UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

CARBON NANOTUBES AND MAGNETIC PARTICLES AS PICKERING EMULSION STABALIZERS: PARTICLE CONTROL FOR PHASE SELECTIVE REACTIONS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

CHASE ALAN BROWN
Norman, Oklahoma
2016

CARBON NANOTUBES AND MAGNETIC PARTICLES AS PICKERING EMULSION STABALIZERS: PARTICLE CONTROL FOR PHASE SELECTIVE REACTIONS

A THESIS APPROVED FOR THE
SCHOOL OF CHEMICAL, BIOLOGICAL AND MATERIALS ENGINEERING

BY

_____
Dr. Steven Crossley, Chair

_____
Dr. Daniel Resasco

_____
Dr. Jeffrey Harwell

Dedicated to my son, Gavin Brown, for propelling me into the field of engineering and motivating me in all that I do.

Acknowledgements

I would like to extend my gratitude to the professors within my committee. Dr. Crossley, whose guidance and passion for exploring the field of reaction engineering in new ways, has shown me the utility of engineering in many areas of research. I am thankful and lucky to have had his mentorship and teaching skill in advanced kinetics and as an advisor. I would also like to present my appreciation to Dr. Resasco, as his course in materials was my first realization of the value in a professor being highly skilled in education. His great teaching abilities and insight into the mechanisms of any physical process were what inspired me to forge a career path in materials science. I would also like to thank Dr. Harwell for showing me the power of scripting in engineering early within my academic career, for always being a respectful and quiet voice of reason, and for his helpful questioning of the fundamentals in emulsion science.

In addition, I would like to acknowledge my friends and family, for if it were not for their support, I would have not been able to make it to this stage in my life. To my parents, I thank you for all of your hard work and money to raise me, your continual support, and listening to me talk about my research when I know you likely didn't want to. To my best friend Grant DeLozier, who has been with me from the first days of my life, I appreciate the interesting philosophical and scientific discussions, long coding sessions, and life's adventures which sculpted my personality, political beliefs, and life's goals. To Nick Briggs, I appreciate you keeping me on track whenever I got too crazy about physics, your ability to keep the lab working despite my best efforts otherwise, and for always cutting to the simplest and most practical solution to a

problem when I needed it but had not yet realized. To all of the chemical engineering and biomedical engineering graduate students, I thank you for always being there for trivia nights, board games, and excellent birthday parties. To Cortes Williams for always being there to brighten the mood either at SWeNT or OU, while being so proper and yet suddenly and unexpectedly strange. To Zach Musset for taking me in to the world of gaming and being a great roommate. And to all the countless graduate students who helped me along the way. Each and every one of you made it all worthwhile. Finally, I would like to extend my gratitude to Needa Virani, for all of the wonderful adventures we continually have together, for listening to my rants about physics when I need to let off steam, for miraculously and consistently making food more amazing than I had ever thought possible, and for eating my attempts at culinary mediocrity despite sometimes missing the mark.

In addition, I would like to thank SouthWest Nanotechnologies and specifically to the head of the research and development division, Ricardo Prada Silvy, for his help and guidance throughout my time at SWeNT, as well as for donating the multi-walled carbon nanotubes used for the studies proposed within this thesis.

Table of Contents

List of Tables

List of Figures

Abstract

Interest in the use of two phase reaction systems for the production of biofuels and specialty chemicals from biomass has been increasing over the past several years, specifically in circumstances in which the feedstock is comprised of both water soluble and oil soluble components. In addition, these immiscible solvents form stabilized emulsions in the presence of solid particles, which provide larger liquid-liquid interfacial areas. In a two phase reaction system, this increased interfacial area improves the phase transfer of any products reacted at the interface into their favored solvent. In particular, Pickering emulsions stabilized by carbon nanotubes (CNTs) have been shown to be exceptionally stable due to their one dimensional nature, as well as their ability to become highly entangled by their large van der Waals forces.[1] This thesis studies the role of carbon nanotube surface functionality alterations in controlling Pickering emulsion properties. It is found that the behavior of CNTs in Pickering emulsions is similar to that of other solid particles such as silica, although with much higher stability.

Liquid biphasic reactors are useful in industry as they allow for a reduction in the amount of capital required if multiple reactions involving different solvents are needed.[2] In hopes to further increase industry interest within these systems, recovery of catalyst via magnetically responsive catalyst supports is studied. This method of recovery may reduce operational cost due to the catalyst loss and environmental issues involved in removing the Pickering particles via filtration.[3] Additionally, we provide some preliminary results that suggest magnetic particles may be of use for controlling reactions in biphasic liquid systems. However, the magnetic support's mode of operation may be fundamentally different from that of many other two phase stimuli responsive emulsion

breaking techniques, as its response is not dictated by surface wetting alterations. We observed that when a hydrophilic catalyst support is forced into a hydrophobic environment within a biphasic reaction system, the catalyst is not as active. As a consequence of this observation, this thesis proposes that the magnetically responsive Pickering emulsion may be of use in applications for which a reversible catalyst deactivation method is required. Discussion of the mechanism for the observation of catalyst inactivity is provided, as it is hypothesized that the particle obtain an adsorbed film of the phase in which they are most soluble.

Chapter 1: Emulsions

History and Introduction

In 1984, a team of archeologists and explorers set out on a journey to the tomb of Nefertari in order to image the various tomb wall paintings, assess the chemical components used for their creation, and attempt to preserve the tomb paintings of the well-known Egyptian Queen. Among the team were Laura and Paolo Mora, two conservators who assisted in identifying the chemical constituents within the pigments and binders for these paintings. These explorers and their colleagues proposed that the wall paintings of Nefertari such as in Figure 1 and other ancient civilizations were created using mixtures of water, egg whites, oils, and berry extract. These common ingredients have a tendency to form emulsions (a mixture of two immiscible liquids homogenized with a stabilizing component). This discovery is therefore suggested to be one of the first recorded uses of emulsions by man.[4,5]

In addition to the long history of artificially created emulsions, these systems have been ubiquitous within our experience from natural sources as well. For instance, dairy products are familiar and well-studied systems in our society, which are – or are derived

Figure 1: Ancient Egyptian Wall Painting.

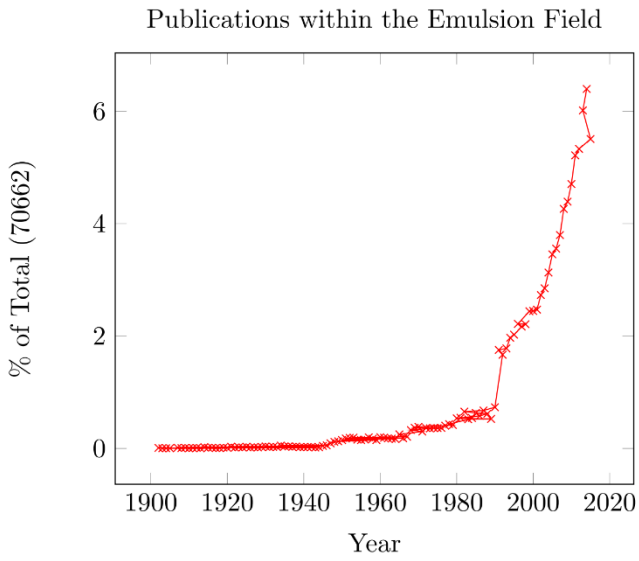


Figure 2: Increase in the amount of publications on the topic of emulsions.

from – the natural emulsion of milk. These emulsions are highly important to our society and provide a major source for protein and its purified supplements (casein and whey) at a rate of approximately 15,800,000 metric tons of protein per year (2013).[6] However, despite our familiarity with emulsions and the success of the many emulsion dependent industries (such as the cosmetic, foods, and pharmaceutical industries) the area of emulsion studies is still not quite well understood. Recently there has been an exponential increase in the area of emulsion science, in efforts to ameliorate the vast gaps in knowledge (see Figure 2). This new attention in emulsion science is primarily due to the ease of which they may be produced in the lab with very little resources, as well as the exciting challenge they pose in creating a complete physical theory to predict an emulsion's properties.

In order to properly explore and predict emulsion behavior, we start by defining the characteristics of an emulsion. An emulsion is defined as a heterogeneous system of two immiscible liquid phases in which one phase (the dispersed phase) consists of small droplets distributed within a second continuous phase. Typically these systems consist

of oil and water, in which the two simple emulsion types are denoted oil-in-water (O/W) or water-in-oil (W/O). In addition to emulsion type, emulsions may be characterized by the emulsion stabilizer properties, thermodynamic or kinetic stability, and droplet size. Emulsions are usually categorized into three different groups – microemulsions (or micellular emulsions), nanoemulsions (or miniemulsions), and macroemulsions (or simply emulsions).[7] These different emulsion types are shown in Figure 3 and discussed in further detail within the following sections.

Stability and Formation

Emulsions must be stabilized by a particle or surfactant in order to prevent rapid coalescence of the droplets, due to the strong thermodynamic driving force to minimize interfacial surface area. The thermodynamics of emulsion formation are given by the Gibbs free energy equation, which can be reduced to the following:[8]

Equation 1

$$\Delta G_{form} = \gamma \Delta A - T \Delta S_{conf}$$

Where γ is the interfacial tension between the two phases, ΔS_{conf} is the conformational entropy change upon emulsification, ΔA is the change in interfacial area after emulsification, and ΔG_{form} is the Gibbs free energy change after emulsification. In many circumstances, the interfacial energy term dominates due to the large interfacial tension between the two phases. This is can be seen in Figure 4 which displays each energy term against the droplet size and number of droplets. It can be observed in these figures that low surface tension values (Figure 4a) provide the means for a negative value of the Gibbs free energy (shown in green), while high surface tension values (Figure 4b) are dominated by very large and positive values of Gibbs free

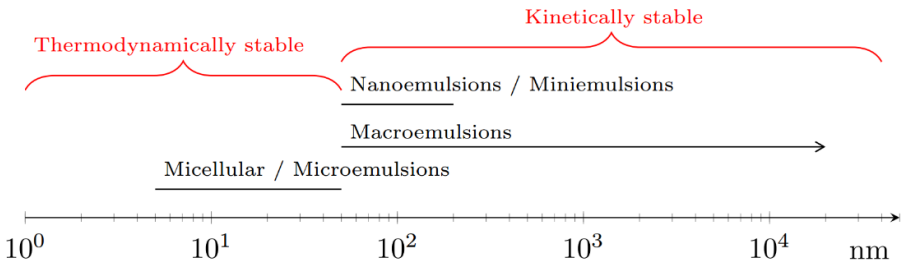


Figure 3: Types of emulsions.
Here we show the likely emulsion stability plotted against the mean emulsion droplet size in nanometers.

energy. This figure was created using the following equation for conformational entropy:[8]

Equation 2

$$\Delta S_{conf} = -\frac{n\, k_b}{\phi}(\phi ln\phi + (1-\phi)\ln(1-\phi))$$

Where ϕ is the volume fraction of the dispersed phase, k_b is Boltzmann's constant, and n is the number of droplets of the dispersed phase. As seen in Figure 4, the Gibbs free energy is negative (spontaneous, thermodynamically stable emulsion formed) for an average droplet radius of 5000 nm or lower for constant temperature (298K), constant total volume, and constant interfacial tension (0.1 nanoNewtons per meter). Extremely low interfacial tensions only occur when using surface active agents, as the typical values for interfacial tensions between oil and water are on the order of milliNewtons per meter (for example, $\gamma_{Decalin-Water} = 51.5\frac{mN}{m}$).

The reduction of surface tension by surface active agents is described by the Gibbs adsorption isotherm for multicomponent systems:

Equation 3

$$-d\gamma = \Gamma_1\mu_1 + \Gamma_2\mu_2$$

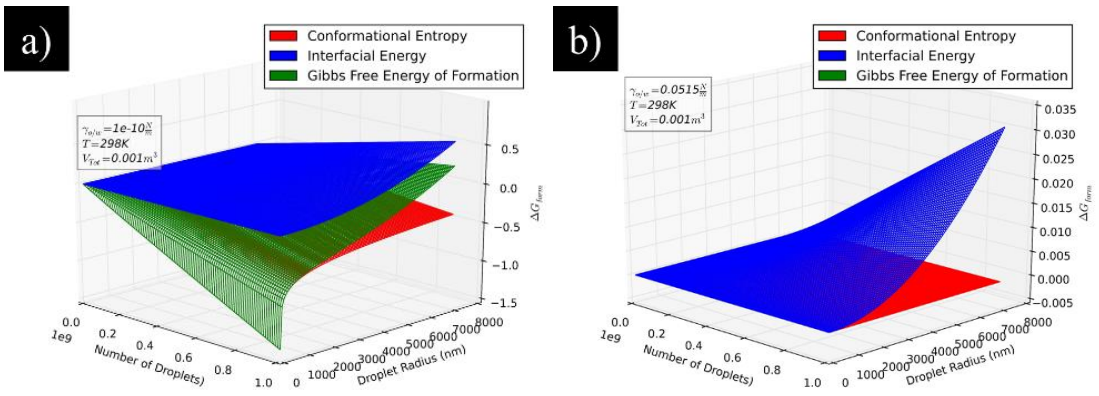


Figure 4: Gibbs free energy plots for emulsion formation.
A small negative area for Gibbs free energy is given at extremely low surface tension (0.1 nanoNewtons per meter), as shown in (a), while a typical oil-water surface tension (b) gives a large positive Gibbs free energy.
Plot created using the python matplotlib script given in Appendix A – S.

Where μ_i is the chemical potential of component i and the surface excess Γ is the excess of solute within the area at the interfacial surface over what would be present if the bulk concentration were the same as the interface concentration. This gives an insight as to whether or not a surface active agent will be more present at the surface (yielding a positive surface excess value) or more present in the bulk (yielding a negative value for the surface excess). The chemical potential for solutes in liquids is a function of the concentration of the solute, and provides an 'availability' metric for the solute. The Gibb's isotherm equation provides insight into the types of emulsions as well as their properties and allows us to predict the general behavior of emulsions under the addition of different chemical constituents. For example, the addition of sodium dodecyl sulfate to an oil-water mixture decreases the surface tension ~20 dynes / cm.[9] However, in the case of solid particles, the surface tension reduction from particle adsorption at interface is very low. Tsuneo Okubo was one of the first main researchers to study the area of solid particles and their effect on interfacial tension. His work showed reductions of interfacial tension of ~2 mN/m for silica spheres and ~20 mN/m for polystyrene spheres

at water-air interfaces, although only at particle concentrations high enough to form crystalline phases within the water. Although these studies by Okubo were at gas-liquid interfaces, many studies of particles at liquid-liquid interfaces have been published, typically in the context of emulsions.

Pickering Emulsions

The work of Okubo gives insight into a special type of emulsion, referred to as a Pickering emulsion. Pickering emulsions are two phase systems which have been kinetically stabilized by solid particles. In a typical Pickering emulsion, liquid droplets (dispersed phase) are stabilized within a continuous phase by these solid particles (often referred to as "Pickering particles") which strongly adsorb between the two phases at the interface. Although Pickering first wrote about this phenomenon over 100 years ago, predicting the properties of a an emulsion created with a given set of parameters remains an active area of research.[10] Pickering emulsion stability is affected by several parameters, such as particle wettability, surface tension of each phase, particle surface roughness, pH, temperature, agitation energy, etc., which provides a great deal of complexity for understanding the formation and breakdown of emulsions.[11–13]

Despite the difficulties in obtaining a complete and predictive theory for emulsion kinetics, Tharwat Tadros *et al.* have provided some preliminary work in this area, such as the ΔG_{form} equations in the previous chapter, as well as several equations describing the breakdown of emulsions.[14] Figure 5 describes the energetics of emulsion breakdown

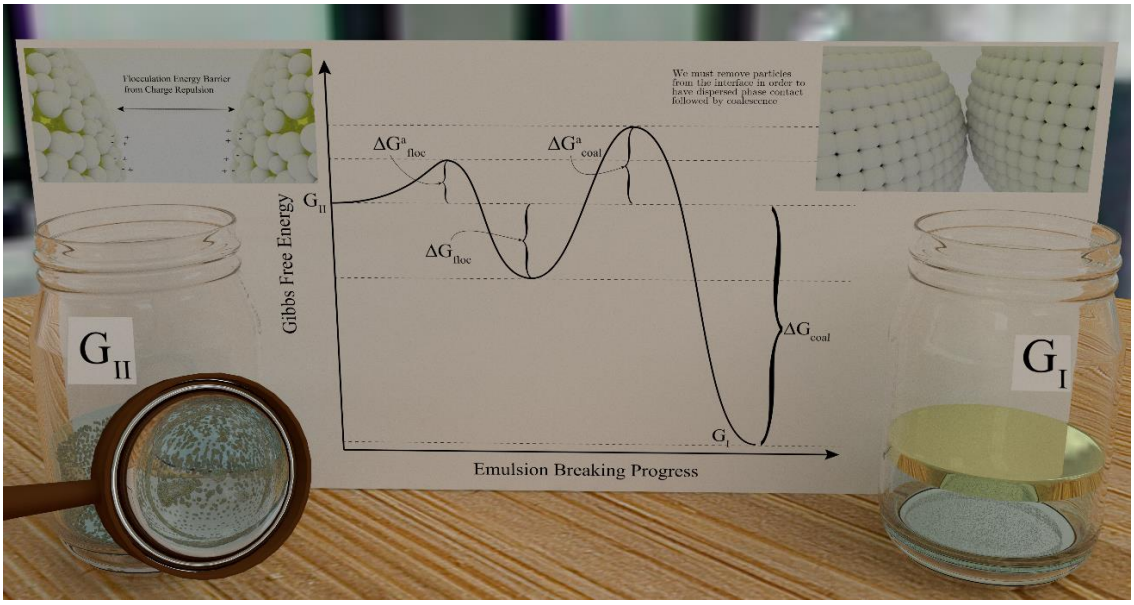


Figure 5: Energy schematic for emulsion breakdown.

when energy barriers are present, as in the case for a Pickering emulsion. This diagram indicates that the energy barriers for flocculation and coalescence are likely the most important energy barriers for providing the stability of kinetically stable emulsions. In addition, Binks *et al.* have shown that energy required to remove a particle from the interface is proportional to the stability of a kinetically stable emulsion, and is likely related to these energy barriers, although this barrier may be affected by many other factors as well.[11,15]

The thermodynamic equations governing particle adsorption energies for Pickering emulsions are often referred to within the literature due to this importance in governing stability. A simple spherical Pickering particle is often cited, as it is the more well understood case from the large availability of polystyrene and silica microspheres:

Equation 4

$$E_{ads} = \pi R^2 \gamma_{ow} (1 + cos\theta)^2$$

Here, E is the energy required to remove the particle from the interface, R is the radius of the particle, γ_{ow} is the surface tension between the oil and water phases, and θ is the

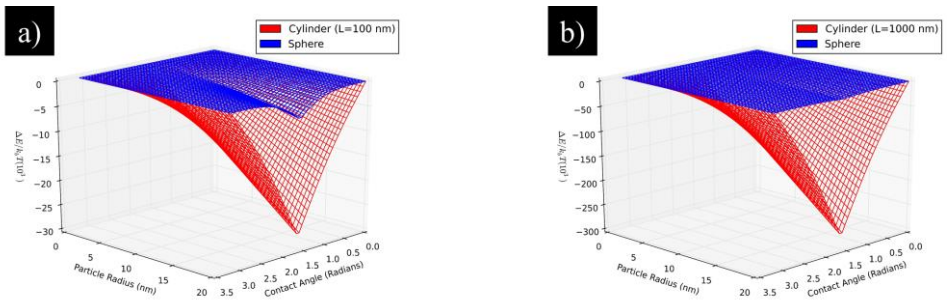


Figure 6: Energy plot from calculation for removing a cylinder from a liquid-liquid interface.
A comparison to the calculation for a sphere (blue) is given for reference. It can be seen that increasing the length ((a) 100 nm and (b) 1 μm) has a large linear effect upon the energy required to desorb the particle from the interface.

contact angle of the particle. The provided equation is simplified such that the particles do not interact with each other and have no surface roughness. However, within the constraints of equal volumes of dispersed and continuous phases (such that Bancroft's rule holds), the particle adsorption energy indicates the trend in emulsion droplet size for both one dimensional as well as zero dimensional particles[11,16]

Given the same assumptions as provided for spherical Pickering particles, it can easily be seen that one dimensional materials can have much larger surface areas per particle when their aspect ratio is large (as in the case for carbon nanotubes), which greatly increases the adsorption energy, given here:[16]

Equation 5

$$\Delta E_{cyl} = \begin{cases} \gamma_{\alpha\beta} RL(\theta \cos\theta - 2\sin\theta), & \theta < 90° \\ -\gamma_{\alpha\beta} RL((\pi - \theta)\cos\theta + 2\sin\theta), & \theta \geq 90° \end{cases}$$

Figure 6 displays that one dimensional particles will increase their surface area contact with both phases as the product of RL (typically much larger than silica's radius term) and can therefore maintain much stronger adhesion at the interface. In addition, some one dimensional materials are flexible or rigid and have varying attraction towards one another. Carbon nanotubes, described in detail within the next chapter,

provide great opportunities for manipulating emulsion stability with their great van der Waals attraction towards one another, as well as their adjustable rigidity which scales with the number of nanotube walls. This creates a tunable manner to create sprawling networks of aggregated and entangled sheets to provide excellent stability for Pickering emulsions.

Chapter 2: Carbon Nanotubes

History and Introduction

Carbon nanotubes (CNTs) have received much attention within the past several decades since their reinvigoration by the scientific community at the presumed discovery by Iijima in 1991.[17] However, it is important to note that an enormous body of literature on carbon filaments previous to this date can easily be found when closely analyzing the history of CNTs. Indeed, Morinobu Endo and Angès Oberlin had taken an image of a carbon nanofilament in 1976 - years before Iijma or Bethune had sparked the excitement within the field in the early 90's.[17,18] However, it is clear that the appearance of Iijima's image of a nanotube (see Figure 7) in a widely read journal such as *Nature* allowed for the large increase in interest to their study over the past few decades.[17]

The excitement brought about by carbon nanotubes was due to the massive amount of applications in which they may be utilized. These materials have been shown to be useful in solar cells,[19,20] computer,[21] polymer composites,[22,23] water treatment,[24] oil recovery,[25–29] Pickering emulsion stabilizers,[2,16,30] catalyst support,[2,31]

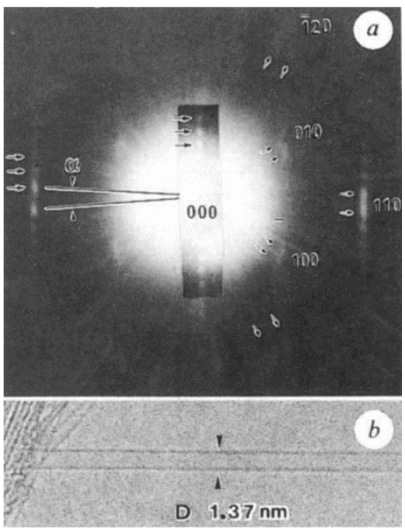


Figure 7: TEM image taken by Sumo Iijima's group of a single-walled carbon nanotube.

chromatography,[32,33] cancer treatment,[34] thin conductive films,[35–37] and many other applications. The quantum mechanical properties are also widely studied by the physics community due to the unique ability to probe the physics of electron-electron interactions[38–42] such as those in Luttinger liquids[43–47] or spin-orbit coupling,[48] as well as phonon,[49–52] plasmon,[53,54] and exciton photophysics.[55–65]

SWCNT Structure

Carbon nanotubes can be divided in to different classes of materials: single-walled carbon nanotubes (SWCNTs), double walled carbon nanotubes (DWCNTs), triple walled carbon nanotubes (TWCNTs), and multi-walled carbon nanotubes (MWCNTs). The structure of a single-walled carbon nanotube was theorized many years before its observation was revealed to the scientific community. The theorized structure was envisioned by rolling a graphene sheet onto itself and connecting the atoms into a tubular construct, as seen in Figure 8.

The chiral vector given below is derived from Figure 8 as in many previous works, and defines the SWCNT via two chiral vector values: (n, m).

Equation 6

$$C_h = n\boldsymbol{a_1} + m\boldsymbol{a_2}$$

Where C_h is the chiral vector of the SWCNT, $\boldsymbol{a_1}$ and are the primitive vectors in the real lattice of graphene (see Figure 10). It follows that diameter and chiral angle can be calculated from geometry as given here:

Equation 7

$$d_t = \frac{|C_h|}{\pi} = \frac{\sqrt{3}a_{cc}}{\pi}\sqrt{n^2 + nm + m^2}$$

Equation 8

$$\theta = \tan^{-1}\frac{\sqrt{3}m}{m + 2n}$$

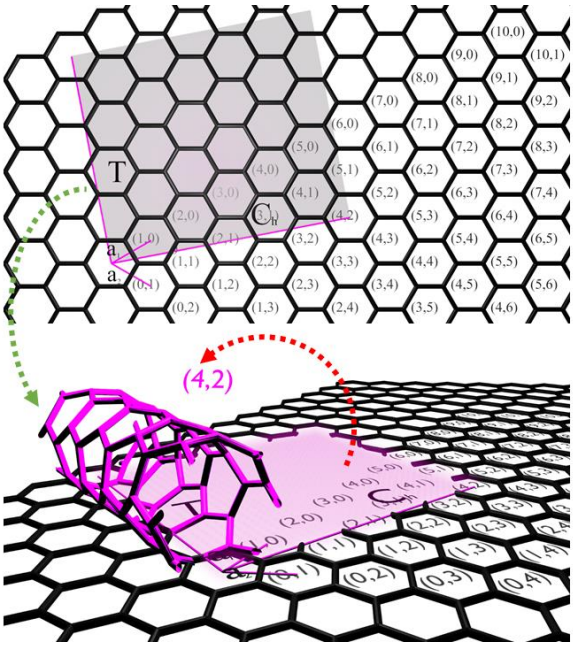


Figure 8: Structure of a (4,2) single-walled carbon nanotube (SWCNT).

Where $a_{cc} = 1.421$ Å is the spacing between carbon atoms in the graphene plane. These equations describing SWCNT crystalline materials are highly important for understanding how to characterize CNT materials. The different SWCNT structures allow for tailored material properties, which can be specified to any application. For instance, the number of walls within a CNT can affect its flexural modulus, and therefore the nanoscale rigidity of double-walled, triple-walled, or multi-walled carbon nanotubes can hinder the ability of the nanotube to adhere its entire axis to liquid-liquid interfaces in very small emulsion droplets.[66] In addition, for the case of SWCNTs, it is often crucial for certain applications to denote the chiral purity, as the optical and electronic properties will greatly vary depending on diameter homogeneity and electronic structure purity.

CNT Characterization

Due to the many synthesis methods for carbon nanotube materials (such as CVD, arc discharge, plasma, laser, CoMoCat, and HiPCO), the as-produced CNT

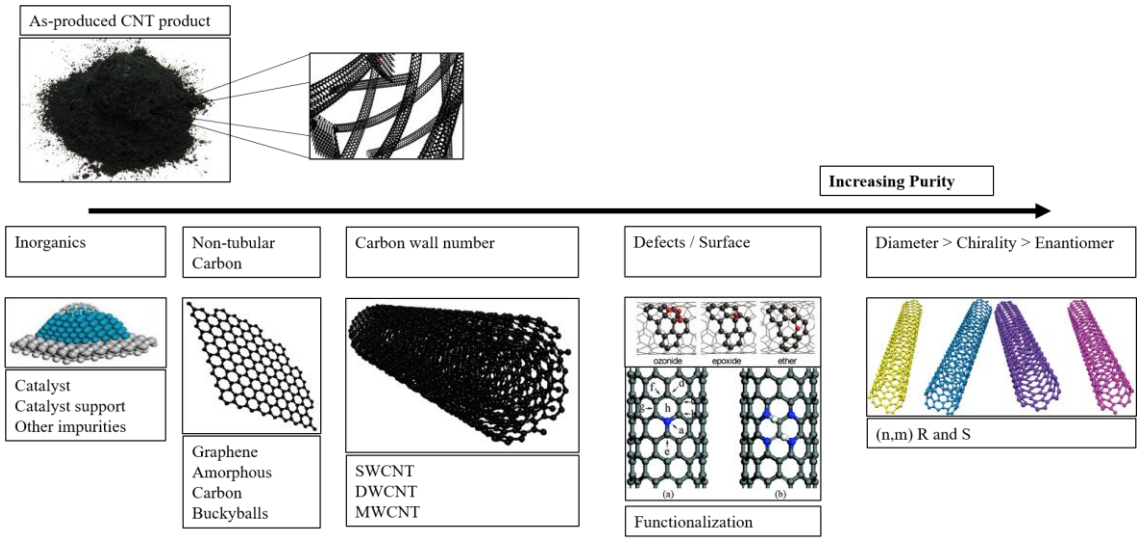


Figure 9: Hierarchy of CNT separations and characterization.

materials are a very diverse and may have varying amounts of impurities, such as transition metals and metal oxides, which are typically used as catalyst support. This large heterogeneity in materials associated with the name 'as-produced carbon nanotubes' is easier understood by constructing a hierarchy of purity, such as the one provided in Figure 9. This hierarchy of purity is important to researchers, because (due to the large diversity of carbon nanotube materials) studies using CNT materials can provide largely erroneous results for some applications by comparison of different manufacturers.

Single-walled CNTs possess interesting electronic properties due to quantum confinement effects in the transverse direction (normal to the SWCNT axis), which create CNT diameter dependent effects such as tunable band gaps. Several methods have been developed to probe the electronic and optical properties of SWCNTs, which provide information about their structure, surface chemistry, level of doping, and other environmental effects (via solvatochromic shifting and Fermi level shifts). These

characterization methods include (but are not limited to) Raman spectroscopy, optical absorbance spectroscopy, and near infrared fluorescence spectroscopy.

Optical Absorbance Spectroscopy

SWCNTs display a wide variety of colors (absorption of electromagnetic spectrum) depending upon their crystal orientation, providing an excellent characterization method for determining chiral purity of SWCNT materials. Spectrophotometer equipment for optical absorption in the ultraviolet, visible, and near-infrared domains is quite ubiquitous in academia and industry, and is therefore very desirable as a standardized metrology method. Characterizing SWCNT dispersions via optical absorption spectroscopy has been a difficult problem due to spectral congestion, background contributions, and lack of individual SWCNT absorption cross section information.[67,68] While attempts have been made to create a standardized characterization program for optical absorption analysis[69], these issues remain problematic and understudied in order to fit experimental results with great accuracy and receive complete information from SWCNT photo-absorption spectra.

In order to characterize CNT materials via any spectral data, we must first predict the output we would expect from theoretical or pristine materials. Optical absorption spectra of surfactant suspended SWCNTs have been shown to be approximated by the linear sum of each SWCNT absorption profile:[69]

Equation 9

$$Abs_{total} = Abs_{bkg} + \sum_{(n,m)} C_{(n,m)} * Abs_{(n,m)}$$

Where Abs_{total} is the total calculated spectra to be fit to the experimental data, $Abs_{(n,m)}$ is the calculated optical response from each individual chiral species, Abs_{bkg}

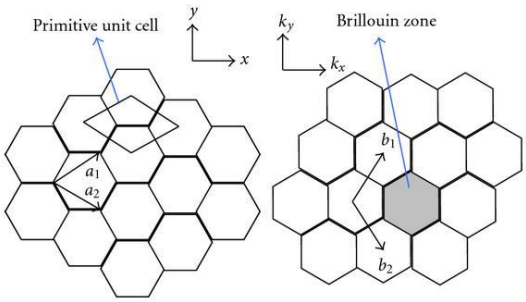


Figure 10: Graphene real space and reciprocal space.

is the background contributions to the optical absorbance from π-plasmon resonances, and $C_{(n,m)}$ is the concentration of the (n, m) nanotube within the solution. Therefore, we must first predict the SWCNT absorption profile $Abs_{(n,m)}$ for each (n,m) species in order to perform a linear fitting. This is typically done by assuming a Lorentzian, Gaussian, or Voigt profile for which the peak center (exciton energy), width, and height (cross section) are estimated using various solid state and optical physics theories.

In order to find the electronic transition energies, we may begin by realizing that the electrons within the system are quantum mechanical in nature, and therefore are described as waves. Therefore, by taking a Fourier transform of the real space lattice to obtain the reciprocal space (i.e. momentum or **k**-space) lattice, we arrive within a lattice space in which the primitive vectors describe the wave vectors of electrons within the system. This is the standard solid state physics approach, in which the energies of transitions are found by creating an electron energy surface within the reciprocal lattice primitive cell (i.e. Brillouin zone or BZ). SWCNT structures are all derived from the graphene crystal structure, and therefore it is customary to start with the graphene Brillouin zone (Figure 10). Subsequently, we may realize that the rolling of graphene onto itself for SWCNT construction quantizes the wave vectors allowable along the SWCNT circumference, creating "cutting lines" along the BZ (see Figure 11). There

are many methods to find the relation between the wave vector (**k)** and the associated energy, including tight binding theory, k·p theory, density functional theory, and others. Tight binding theory has been written about extensively within the SWCNT literature, yielding the dispersion relation given here:[70]

Equation 10

$$E_{g2D}(k_x, k_y) = \pm \mathrm{t}\left\{1 + 4\cos\left(\frac{3k_x a_{cc}}{2}\right)\cos\left(\frac{\sqrt{3}k_y a_{cc}}{2}\right) + 4\cos^2\left(\frac{\sqrt{3}k_y a_{cc}}{2}\right)\right\}$$

Where t is the tight binding overlap energy (~ -3eV), a_{cc} is the carbon carbon bonding distance (0.144 nm), and k_x are the x and y components of the electron wave vector or momentum vector. This famous result for calculating the Fermi surface is shown in Figure 11, where it can be seen that the surface has a linear character near the K point (Dirac point). Indeed, when a Taylor expansion of the above equation is taken, a linear dispersion relation can be extracted:

Equation 11

$$E(k) = \frac{3ta_{cc}}{2}\sqrt{\delta k_x{}^2 + \delta k_y{}^2} = \hbar v_F|\boldsymbol{k}|$$

Where $\frac{3\pi t a_{cc}}{2h} = v_F(\sim 10^6 \frac{m}{s})$ because the Fermi velocity is defined as the group velocity at the Fermi vector $\left(v_F = \frac{1}{\hbar}\frac{\partial E}{\partial k}\right)$. This linear E vs. **k** is typically used to simplify the calculation of transition energies in graphene and SWCNTs, although there are certain deviations from this relation for small diameter SWCNTs and for transitions far away from the Dirac point.

The red lines cutting across the Fermi surface over the graphene Brillouin zone in Figure 11 are shown giving rise to the van Hove singularities in the SWCNT density of states (DOS). The density of states for any material is a hugely important feature to be

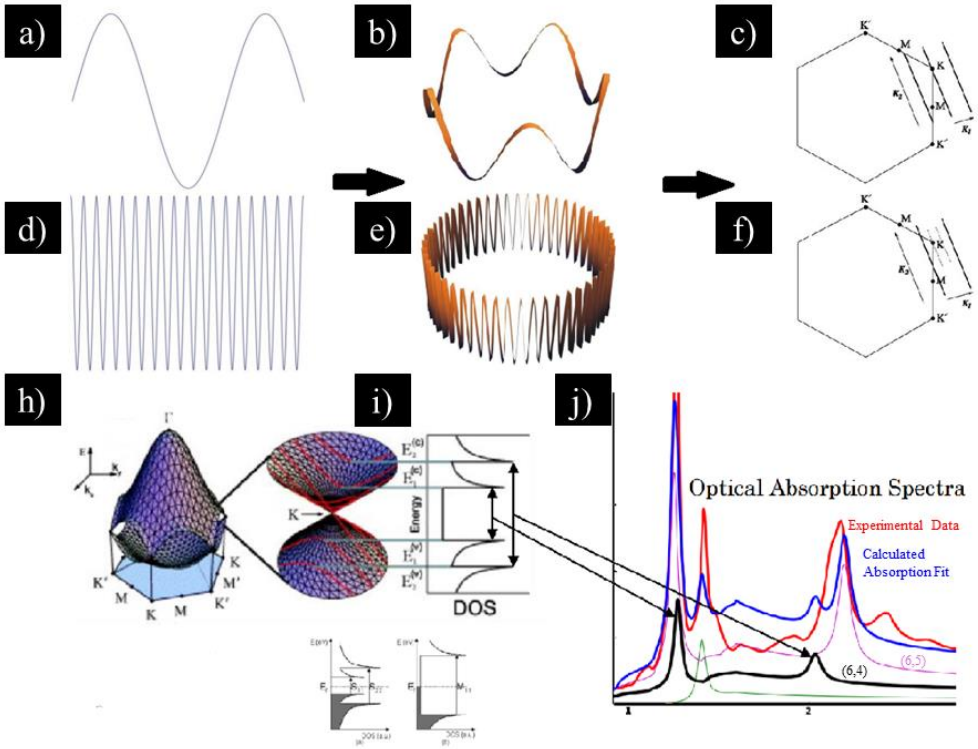


Figure 11: Overview of nanotube electronics and photophysics.
(a, b, d, e) Visualization of periodic boundary conditions (Born-von Karman) for a wave along a closed loop and its effect on quantizing allowable wave vectors within the graphene Brillouin zone (BZ). The BZ in (c) shows the allowable wave vector cutting through the Dirac point (or K point, shown in (h)), creating a metallic SWCNT. Conversely, the bottom (f) BZ has wave vectors offset from the Dirac point, creating a semiconducting CNT with a bandgap (shown in (i)). These bandgaps shown in the DOS (i) give rise to peaks in the optical absorption spectra via excitonic transitions, as shown in (j).

understood for any fundamental study involving electrons and their behavior in a nanoscale system.

This expression of van Hove singularities (vHS) in the SWCNT joint density of states (JDOS) due to the quantum confinement of the electrons in the 1- dimensional wire allows the band gap energies of SWCNTs to be varied over the visible and near infrared regions by manipulating their diameter. It can be seen from this quantum confinement effect that as the diameter of the SWCNT grows and approaches the band

structure of graphene, the distance between allowable wave vectors grows closer and the band gap energy becomes quite small. Similarly, as the SWCNT diameter grows smaller, the band gap grows larger due to the increase in the distance between allowable wave vectors states.

While many past studies have relied heavily on tight-binding formulations of the SWCNT excitonic transition energies, more recent models have been extended to include electron-electron interactions and trigonal warping effects.[38,42,71] The simplest and most predictive model to date has been developed by noticing the linear dispersion given in Equation 11. Higher order terms for anisotropy and other effects are added with an electron wave vector (**k**), and transition number (p) dependence:

Equation 12

$$E_p(\boldsymbol{k}) = 2\hbar v_F(p) \times k + \beta \times k^2 + \eta(p) \times k^2 \cos 3\theta$$

Where E_p is the excitonic energy associated with a transition number p, β accounts for the electron-electron interactions and effective mass, and η accounts for trigonal warping effects, and k is the standard electronic wave vector.

These exciton energies provide the peak centers for the calculated absorption profile for each (n,m) species; however, line widths, cross sections and any other side band features from phonons are also required to create a theoretical SWCNT profile. Several reports have provided these terms with differing theories, such as GW Beth-Salpeter equations, Hubbard ab initio models, and empirical parameters.[56,72–77] The most general of these approaches has led to the following relation, which is applicable to any given (n, m) structure:

Equation 13

$$Abs_{(n,m)} = \frac{\Sigma_p}{\pi} \cdot \frac{w_p}{\left(E - E_p\right)^2 + {w_p}^2} + \frac{\Sigma_p}{a \cdot \pi} \cdot conv\left(\frac{b \cdot w_p}{E^2 + \left(b \cdot w_p\right)^2}, \frac{\Theta\left[E - \left(E_p + \Delta\right)\right]}{\sqrt{E - \left(E_p + \Delta\right)}}\right)$$

The first term within this equation describes the excitonic transition and the second term describes the phonon side band (PSB) contributions. Here, E is an arbitrary energy, w_p is the half linewidth of the excitonic peak transition, Σ_p is the oscillator strength, and E_p is the excitonic energy described previously, while the PSB has terms to modulate these parameters. The $conv$ is the convolution function for convolving the Heaviside step function Θ, placed at an offset of Δ away from the higher energy tail side of the main exciton transition, with modulations of the line width and oscillator strength, b and a respectively. The work done to provide this equation has great reaching power to characterize nanotubes, and to provide support for many fundamental studies involving SWCNTs.

A python script using this SWCNT profile and the exciton energies from above was made by the author of this thesis and is given on in Appendix A – Scripts and GitHub (https://github.com/chaxor/Spectro) with the intention of assisting researchers with the characterization of SWCNTs via optical absorption data automatically and ameliorate some of the issues listed above. The python script provided is capable of fitting experimental data into each individual SWCNT absorption profile, in addition to simultaneously fitting a number of background contributions considered in previous works.[67]

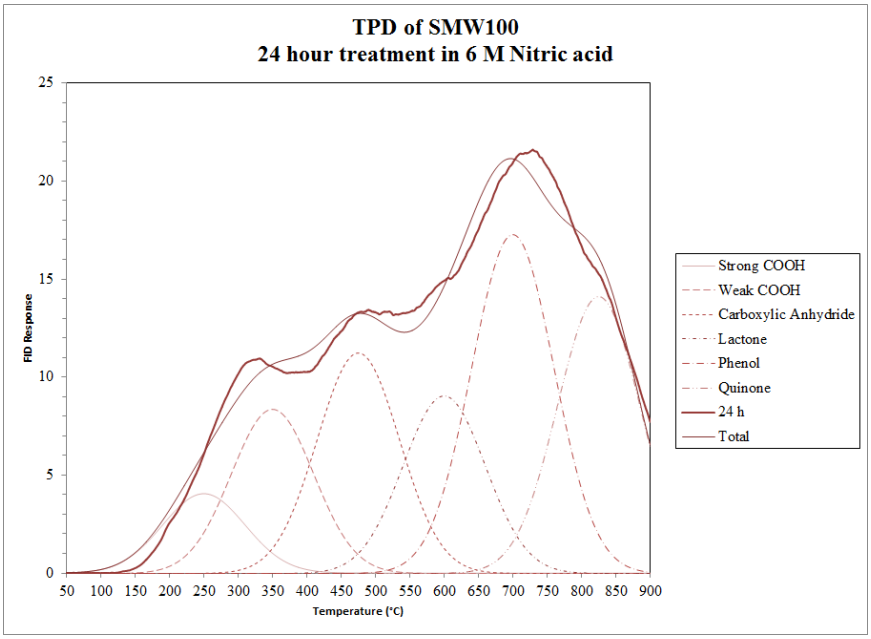


Figure 12: TPD fitting of MWCNT.

Temperature Programmed Desorption

Tuning the surface chemistry of carbon nanotubes has been an intensively studied area for applications in polymer processing, emulsion stability, medicine, catalysis, and batteries. The surface chemistry of carbon can be altered with many different approaches such as ozone, plasma treatments, strong acids, microwave irradiation, and peroxide functionalization. The quantification of these functional groups is useful for determining contact angle and interactions with their environment. One of the most well-known methods for determining the surface functionality of a carbonaceous material is temperature programmed desorption (TPD), wherein a small aliquot of material is slowly heated in an inert environment such that the oxygen groups desorb from the surface as the temperature rises. In the case of CNTs, the profile of the resulting data can be interpreted to provide the amount of differing surface groups such as carboxylic acids, lactones, quinones, phenols, etc.

Surface group	Product	Temperature
Carboxylic Acid	CO_2	453–573 K
anhydrides	CO, CO_2	673–720 K
		710–930 K
Lactones	CO_2	410–430 K
		463–923 K
Peroxides	CO_2	823–873 K
Phenol	CO	873–973 K
Quinones	CO	1073–1173 K

Table 1: Surface functional group temperatures of desorption.

In order to extract the functional group character of a given nanotube material from the experimental TPD data, either the energy of desorption for each functional group and support structure combination within the studied material must be known, or empirical parameters may be given to estimate the desorption profile for each functional group. Ideally, the energy of desorption, heating rate, exponential pre-factor, surface coverage dependence of the energy of desorption, and order of reaction should be used with Arrhenius kinetic modeling in order to fit the experimental data for each functional group. However, many studies estimate surface chemistry of CNTs using the empirical approach as, although it may be less accurate, it requires less data acquisition[78,79].

This method of fitting using Gaussian profiles centered at specific temperature was employed on MWCNT samples treated in nitric acid at different molarities and differing times of treatment. The choice of a Gaussian profile arises from a continuous probability distribution of desorption energies for each associated surface group. The temperatures for each oxygenate species were obtained from the work of Renju Zacharia, which is summarized in Table 1.[80] Data analysis and quantization of total functional groups on the MWCNT surfaces have been discussed in Chapter 3: MWCNT

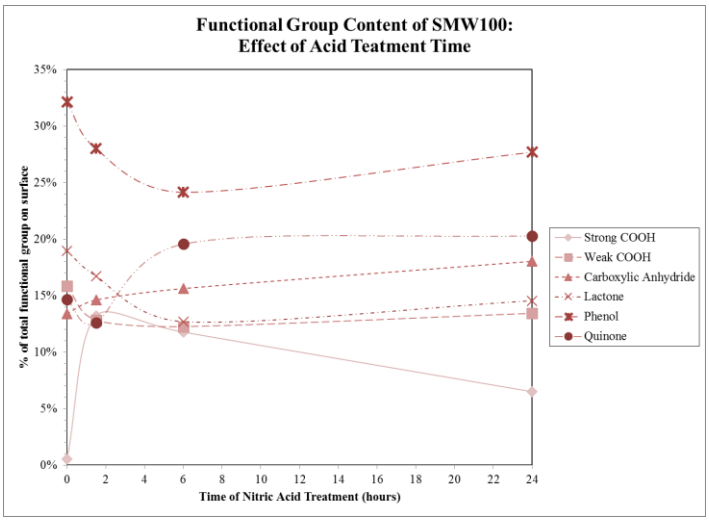


Figure 13: Results of Non-linear fitting of TPD experimental data for SMW100 treated with Nitric acid for varying times.

Sample Characterization. However, since only the total oxygen content is reported in Chapter 3, here we study the features of the experimental profile and gather information about the amount of each surface group. This is done by taking the area of each functional group's corresponding profile, as shown in Figure 12. The results of these analysis can be seen in Figure 13.

While previous reports have shown that quinone groups decrease with increasing oxidation of carbon surfaces, the data we have obtained suggests a different outcome.[81] However, there are several key differences between the study proposing quinone conversion to carboxylic acid and our analysis here. First, the nanotube materials used when finding that quinone decreases with acid functionalization were 25-40 nm in diameter, while our SMW100 material is characterized to be 6-9 nm, which may affect the reactivity of the surface and the selectivity towards forming certain structures. In addition, our report has studied time of reaction in a constant molarity of acid, whereas they studied the effect of concentration. Further study is required to show a plausible mechanism for this reaction. In order to facilitate future works, this thesis

has also provided the python script for TPD analysis which may be helpful as a starting point for researchers studying oxygen groups on carbon materials.

Chapter 3: Wettability of multi-walled carbon nanotubes tuned to control Pickering emulsion properties

Introduction

Despite extensive work done with Pickering emulsions stabilized with silica, much fewer studies have focused on multi-walled carbon nanotubes as the Pickering emulsion stabilizer within these two phase systems. Specifically, one previous study has looked at the behavior of zero dimensional and one dimensional carbon based systems (such as SWCNTs and C_{60}) in Pickering emulsions, where it was noted that zero dimensional carbon (C_{60}) was not a good emulsion stabilizer in comparison to SWCNTs.[82] Some works have commented on the role of CNTs in two phase systems, such as in CNT purification and separations processing, although the emulsion systems were not the main focus of the studies. Typically these newer studies have focused on bulk phase characteristics, such as the increase in nanotube individualization and optical properties after trapping nanotube bundles at liquid interfaces. This process allows for a scalable way to mimic high speed centrifugation experiments for removing bundled SWNTs with less sophisticated equipment.[83–85] Studies which *have* focused on MWCNTs typically focus more on emulsion properties and other macroscopic effects, as the optical and electronic properties and particle physics of MWCNTs are not as precisely tunable as SWCNTs, mimicking that of bulk carbon. Reports have shown that MWCNTs have been oxidized via acids and plasma treating and used to make O/W or W/O emulsions, the latter of which was used to produce microcapsules.[30,86] However, these studies provide little information on the emulsion characteristics, while, to the best of the author's knowledge, there have been no reports in the control of nanotube

functionalization via acid surface modification in order to study the effect on Pickering emulsion properties. Studying Pickering emulsions stabilized with multi-walled carbon nanotubes may provide new opportunities to understand emulsion property fundamentals. In addition, studying the fundamental physical material properties and their manipulation has historically led to discoveries of even further unique features of these materials, allowing for greater impact and research into many fields. For example, MWCNTs have already seen such wide variety of applications in catalyst supports[2], coatings[87], batteries[88], sensors[89,90], field effect transistors[91], high strength yarns[92,93], hydrogen storage material[94,95], thermal management[96], compressible foams[97], and energy adsorbing hybrid composites[98].

The work provided in this thesis aims to study Pickering emulsions stabilized with a range of functionalized MWCNTs. These Pickering emulsions were studied by varying oil type, oil-water ratios, MWCNT concentration, MWCNT functionalization degree mass ratio, and initial nanotube dispersion phase. Three competing hypotheses are provided, each which may contribute to the formation of the Pickering emulsion properties.

First, changing the particle wettability changes the dispersion of the nanotubes, which will increase the available material for covering interfacial area. Second, as the wettability of the MWCNTs is changed, the energy required for particle desorption from the interface will change, affecting the number of MWCNTs at the interface. Lastly, due the differences in particle wettability, the nanotubes can re-orient themselves to occupy higher or lower specific surface areas depending on the energy

	0.8 MWCNT	**2.3 MWCNT**	**3.1 MWCNT**	**4.4 MWCNT**
HNO_3 Molarity [M]	0	6	6	12
Acid Treatment time [h]	0	1.5	24	24

Table 2: Description for the functionalization of each sample

with which the MWCNT is held at the interface. This change in orientation has been hypothesized and shown with fumed silica.[15]

Experimental

Multi-walled carbon nanotubes were obtained from SouthWest Nanotechnologies, Inc. designated by the company as SMW100 (lot 48). The MWCNTs have average diameters of ~6-9 nm, 3-6 walls, and lengths less than 1 μm. Nitric acid (70%), dodecane (99%), heptane (99%), and toluene (99%) were purchased from Sigma Aldrich. 18 MΩ ultra-pure water was obtained from a Cole Parmer filtration system. Oxidation of the MWCNTs was carried out by colleague Nick Briggs, with 1 g SMW100 in 6 M or 12 M HNO_3. This solution was refluxed for 1.5 hours or 24 hours at 110°C (where time was used to control the number of functional groups). The MWCNT solution was then filtered by the colleague Nick Briggs with a 0.22 μm PTFE filter and rinsed with 18 MΩ water until the pH was neutral. MWCNT samples resulting from these treatments have been tabulated in Table 2. MWCNT Pickering emulsions were made with horn sonication using a Fisher Scientific Model 505 Sonic Dismembrator with a 0.5 inch horn tip. MWCNTs were added to reach 0.03-0.26 wt. % (with respect to water) of a 100 mL dispersion in a beaker. A concentration of 0.07 wt. % of MWCNTs was used for all emulsions, unless otherwise stated. Water or dodecane was

then added to this dispersion and sonicated for 5 minutes at 75% amplitude using a 0.5 inch horn tip. This step was done to disperse the MWCNTs, as they are typically highly agglomerated and bundled naturally in the solid state. The opposite phase was then added to this dispersion. Emulsification was performed by horn sonicating the solution for 5 minutes at 100% amplitude using a 0.5 inch tip horn tip. All dispersion were performed by colleagues at OU, either Nick Briggs or Brian Li.

Optical microscopy was used to obtain the average emulsion droplet size for each emulsion. ImageJ software was used to measure the diameter of 100 emulsion droplets from captured images. Emulsion type is determined to be o/w (oil in water) or w/o (water in oil) by placing an aliquot of the emulsion phase into a beaker of pure water and observing if it disperses well. If it disperses well it is an o/w type emulsion and if not, it is a w/o type. The opposite phase was used in the same manner to check the result.

All temperature programmed desorption (TPD) experiments were performed under Helium flow (35 $\frac{ml}{min}$) in a Thermcraft furnace (Model SST-0.75-0-12-1C-D2155 A). Each MWCNT sample was prepared by drying at 120°C in a vacuum oven in order to remove any moisture, weighed to 50 mg, and placed in a thin (3 mm diameter) quartz tube. These samples were heated from room temperature to 900°C at a heating rate of 10°C min and held isothermal for an additional 30 mins. The outlet gas from the furnace was then sent to a Nickel catalyst to convert the carbon oxide gases to methane. This stream is then sent to a flame ionizing detector (FID) (SRI 110 detector Chassis) which provided a TPD spectra. Total integrated TPD signals were used in order to determine the atomic percentage of oxygen groups on the CNT material. This was achieved using

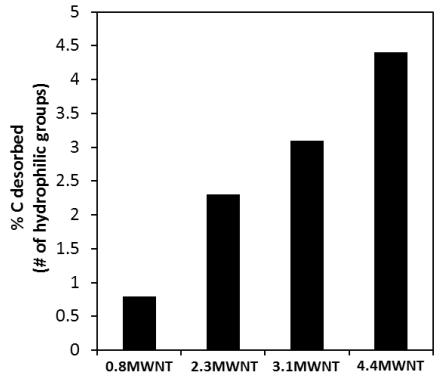


Figure 14: TPD results of MWCNT samples.
Percent carbon desorbed for each MWCNT sample.

Simpson's rule for integration of the FID signal against time. The total moles of carbon desorbed was determined by the product of this integrated area and a constant. This constant was determined by pulsing 64 µL volumes of CO through the TPD system under the same conditions. Treating this CO as an ideal gas provides a molar value associated with the total integrated signal, $1.3\times10^{-13}\ \frac{mol\ C}{s\cdot signal}$. It was assumed that the carbon desorbed from the CNT material during the TPD was equivalent to the oxygen functional groups created by the acid treatments, due to the lower thermal stability of these groups which convert to CO or CO_2. Therefore, the atomic percentage of thermally unstable oxygen groups was found by the ratio of total molar carbon desorbed (found by the process above) and the total molar carbon in the initial sample.

The TPD data suggests that our 4 samples have 0.8, 2.3, 3.1, and 4.4 wt. % thermally unstable carbon on their surfaces as seen in Figure 14. We then designate the sample names used within this study based on their TPD thermal stability, as seen in Table 2.

O:W Ratio	0.8 MWCNT	2.3 MWCNT	3.1 MWCNT	4.4 MWCNT
1:04	w/o	o/w	o/w	o/w
2:03	w/o	o/w	o/w	o/w
1:01	w/o	w/o	o/w	o/w
3:02	w/o	w/o	o/w	o/w
4:01	w/o	w/o	w/o	o/w
1:04	w/o	o/w	o/w	o/w

Table 3: Emulsion type for each o:w ratio of the MWCNT samples.

Results and Discussion

MWCNT Sample Characterization

MWCNTs are inherently hydrophobic and tend to agglomerate due to van der Waals forces. A typical method to increase the hydrophilicity and decrease agglomeration is to oxidize the MWCNTs with nitric acid or a mix of nitric and sulfuric acid to create hydrophilic functional groups.[99–101] This study uses the latter method with HNO_3 in order to control the number of hydrophilic functional groups and the concentration of these groups was determined by TPD (Figure 14). TPD plots showing the carbon desorbed as a function of temperature for each sample are in the supplementary information, Supplemental Figure 1.

Previous work by Binks *et al.* have shown the emulsion inversion point will change at different oil-water ratios depending on the hydrophobicity of spherical silica particles.[11] A very highly reported rule of emulsions, known as the Bancroft rule, states that hydrophobic particles favor w/o emulsions at a 1:1 o:w ratio, while hydrophilic

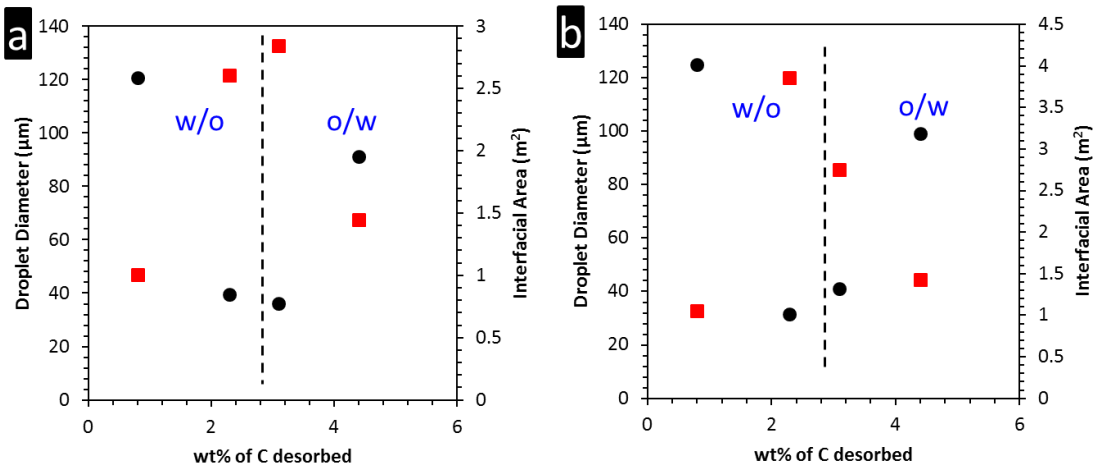


Figure 15: Change in droplet diameter as a function of MWCNT wettability
Results are for dodecane and water system (0.07 wt. % MWCNTs) when dispersing MWCNTs in (a) water or (b) dodecane previous to emulsifying.
Circles (black) denote droplet diameter while squares (red) denote interfacial area.

particles favor o/w emulsions.[11,102] A quick characterization of the particles hydrophobicity can be accomplished using this phenomenon. A high o:w ratio is required for emulsion inversion when hydrophobic particles are used, while more hydrophilic particles require a low o:w ratio. Intermediate hydrophobic group coverage is required in order to achieve an equal o:w ratio. Using this method, we found different o:w ratios were required for emulsion inversion for each MWCNT sample, as shown in Table 3. This indicates the hydrophilicity of each MWCNT sample is unique, which correlates with the TPD results. Initially no inversion in emulsion is obtained, due to the MWCNTs inherent hydrophobicity, while as the number of hydrophilic functional groups increase on the MWCNTs the emulsion inversion points requires a higher o:w ratio.

Role of MWCNT Wettability

Emulsion droplet size and interfacial area change with the hydrophilicity of the MWCNTs as indicated in Figure 15. This trend observed here with MWCNTs is also

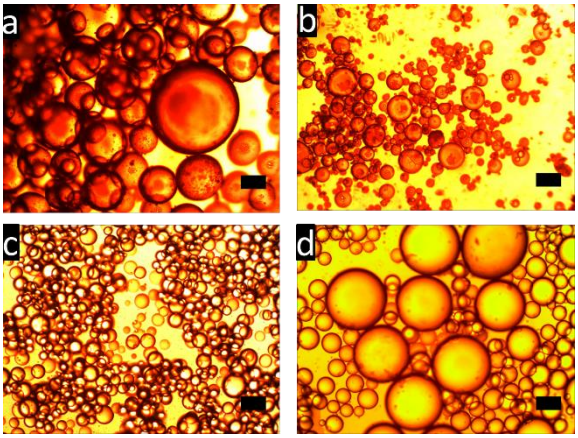


Figure 16: Optical microscope images for dodecane and water system (0.07 wt. % MWCNTs).
Scale bar in image is 100 µm.

observed with fractal silica.[11] The emulsion droplet diameter is large at first then decreases and then increases again with the most hydrophilic and hydrophobic particles being the largest, while the emulsion flips at intermediate hydrophilicity, as can be seen in Figure 16. Interfacial area follows a similar trend; however, the data follows a volcano plot trend, rather than a valley plot which is expressed for droplet diameter (see Figure 15a). Dispersion of silica particles of intermediate hydrophobicity have been shown to control emulsion type based on the phase the particles are initially dispersed in, with the continuous phase of the emulsion being the phase of which the particles were initially dispersed.[102] No difference in emulsion type was found when dispersing the MWCNTs initially in water or oil (Figure 15). Studying the variation in the MWCNT concentration shows the same trend, as shown in Figure 17. Previous work has shown that as the concentration of particles in an emulsion system increases, the emulsion droplet size decreases to a point, and then plateaus.[103] The initial decrease in emulsion droplet size is due to a larger availability of particles to stabilize more interfacial area which keeps droplets from coalescing. However, at some point there are

sufficient particles to cover the emulsion droplets and the droplet size no longer changes.

Emulsion Stability

Agglomeration of carbon nanotubes has been shown to enhance emulsion stability over silica particles. Strong van der Waals forces between carbon nanotubes forms a rigid network over an emulsion droplet surface which stabilizes emulsion droplets during coalescence. Charge repulsion between silica particles creates a weak network over an emulsion droplet surface which is easily destabilized during droplet coalescence. Emulsions stabilized with silica of variable wettability have been shown to have different emulsion stability.[102] The most unstable emulsions were made with hydrophobic and hydrophilic silica. Stable emulsions were made with silica of intermediate wettability. The hypothesis that the change in MWCNT wettability may have induced similar emulsion stabilization alterations as those seen with silica was tested to contrast zero dimensional particles with one dimensional particles. To test if changing the MWCNT wettability weakened the MWCNT network over the emulsion

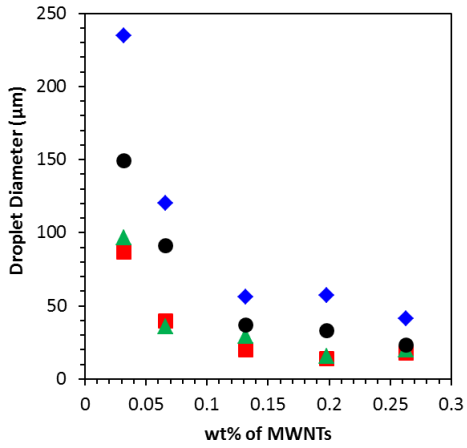


Figure 17: Effect of concentration of MWCNTs on emulsion droplet size. Diamonds (blue) denote 0.8MWCNT, squares (red) denote 2.3MWCNT, triangles (green) denote 3.1MWCNT, and circles (black) denote 4.4MWCNT.

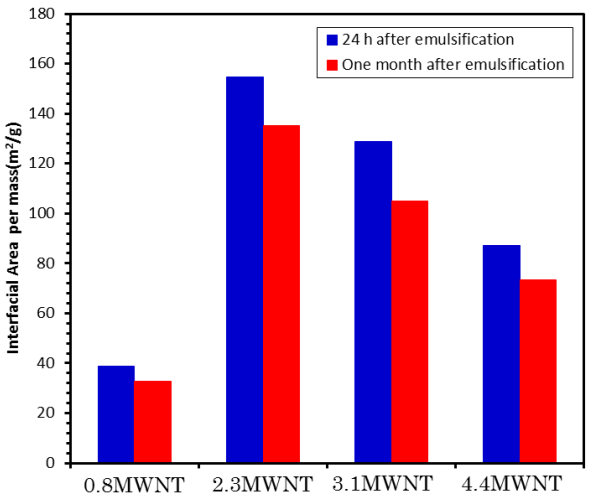


Figure 18: Change in interfacial area 24 hours and one month after emulsification.

droplet surface, emulsion stability tests were conducted by comparing the interfacial area 24 hours and one month after emulsification. Small changes in interfacial area occurred during the emulsion stability test for all samples (as seen in Figure 18). The emulsion stability test shows the emulsion type and droplet size can be changed by changing the MWCNTs wettability and have good emulsion stability. MWCNTs of the most hydrophobic and hydrophilic have high emulsion stability in contrast to hydrophobic and hydrophilic silica which have poor emulsion stability.

Different Oils

Two parameters which affect the energy required for a particle to desorb from a liquid-liquid interface are contact angle and interfacial tension. We may observe the effects of these parameters by varying oil type. Table 4 shows that heptane and dodecane vary the most in contact angle, while toluene and dodecane vary the most with respect to oil-water interfacial tension. Therefore, these two oils allow for one parameter to be held roughly constant while changing the other. Admittedly, the values

O:W Ratio	γ ($\frac{mN}{m}$)	θ
Dodecane	52.5	122
Heptane	50.7	105
Toluene	35	125

Table 4: Information about oils used within this study.

cited are for an oil-water interface and do not take into account the role of the particle at the interface. While the emulsion droplet diameter, interfacial area, and emulsion inversion point change depending on the oil, the trends (decrease in the emulsion droplet diameter or increase in interfacial area at intermediate hydrophobicities) seen for dodecane emulsions are displayed in a similar fashion for heptane and toluene (see Figure 19). This shows the oil type has an effect on the emulsions properties, which may be related to the oil-water contact angle and interfacial tension, in turn affecting the energy required for particle desorption.

Oil to Water Ratio

Changing the oil-water ratio of an emulsion allows for one to tune the emulsion without having to change the particles properties, oil phase, or aqueous phase. When changing the oil-water ratio for hydrophobic MWCNTs (sample 0.8MWCNT), a decrease in emulsion droplet size is observed, as the oil fraction increases while no change in emulsion type occurs (Figure 20a). Stiller et al. showed for a similar trend for hydrophilic titanium dioxide, except the emulsion droplet size increased with increasing oil fraction.[104] Emulsion droplet size decreases due to the MWCNTs having to stabilize less water in the system. As the oil fraction decreases, more water is present

and the emulsion does not invert, which indicates the inherent hydrophobicity of the MWCNTs. Interfacial area also increases with the increase in oil fraction. As the amount of water decreases, the MWCNTs have less emulsion droplets to stabilize consisting of water, allowing the MWCNTs to stabilize smaller emulsion droplets. MWCNTs with intermediate hydrophobicity (sample 2.3MWCNT) display emulsion droplet sizes and interfacial areas which tail off at high and low oil fractions (Figure 20b). The emulsion inverts below an oil fraction of 0.5. Due to the emulsion inversion occurring close to an oil fraction of 0.5, the MWCNTs equally wet the oil and water. This may be why the greatest interfacial area is obtained at an oil fraction close to 0.5 and the change in emulsion droplet size at the inversion point is not catastrophic. Sample 3.1MWCNTs shows this catastrophic phase inversion as the emulsion droplet size is small until an oil fraction of 0.8 is reached (Figure 20c). This catastrophic phase inversion has been observed with hydrophobic and hydrophilic silica.[105] Emulsion droplet size increases drastically once reaching the oil fraction which causes emulsion inversion from o/w to w/o. The drastic change in emulsion droplet size is also seen with

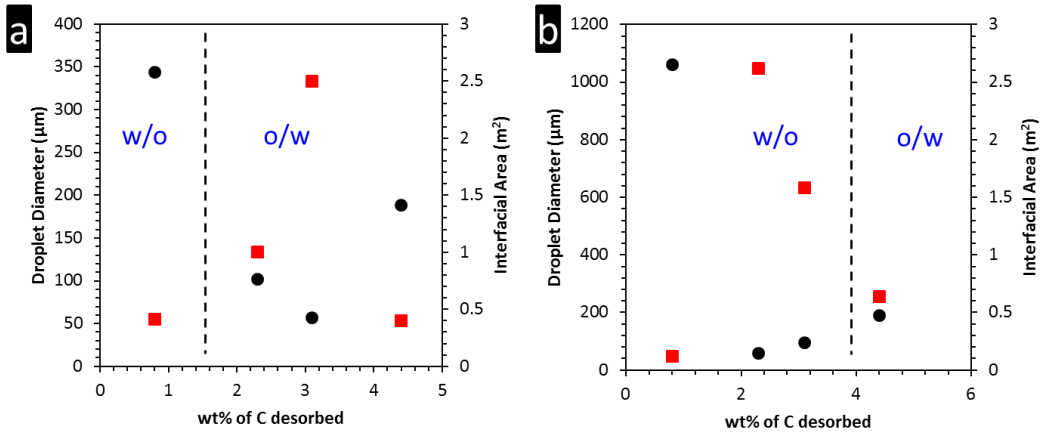


Figure 19: Change in emulsion droplet size and interfacial area for a) heptane and b) toluene oil phases.

Circles (black) denote droplet diameter and squares (red) denote interfacial area.

interfacial area. This catastrophic phase inversion is not seen with sample 2.3MWCNT. This may be due to sample 3.1MWCNT being more hydrophilic than 2.3MWCNT. The point of emulsion inversion is further away from an equal 1:1 ratio for sample 3.1MWCNT than for 2.3MWCNT. The most hydrophilic MWCNTs (sample 4.4MWCNT) shows no change in emulsion type (Figure 20d). Emulsion droplet size decreases as the oil fraction decreases, while interfacial area increases. With less oil in the system, the MWCNTs do not have to stabilize as much interfacial area, allowing for smaller emulsion droplets. At an oil fraction of 0.8, the emulsion droplet size deviates from the trend by decreasing. However, the interfacial area follows the expected trend by decreasing further. We believe this is an indication of the emulsions stability playing

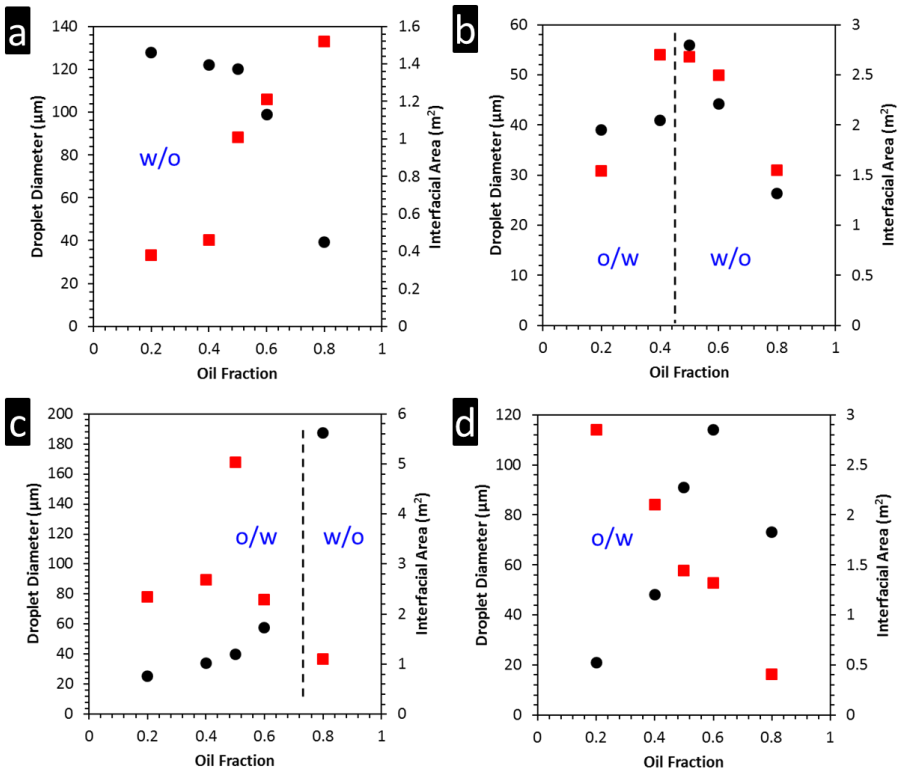


Figure 20: Effect of changing the oil (dodecane) to water ratio on droplet diameter. Sample (a) 0.8MWCNT, (b) 2.3MWCNT, (c) 3.1MWCNT, & (d) 4.4MWCNT. Circles (black) denote droplet diameter and squares (red) denote interfacial area.

a role. During sampling, the emulsion fraction was small and emulsion droplets were difficult to find. Therefore, the emulsion droplets at this oil fraction may be smaller because of high emulsion instability (only the small droplets survive). Hence, we highlight the importance of measuring the oil, emulsion, and aqueous fractions to calculate the interfacial area.

Mixed Types of MWCNTs

Due to the potential challenge of obtaining the desired hydrophilicity of a particle there may be an advantage to mixing two particles each with a different hydrophilicity. Simply by changing the ratio of particles one can tune the emulsions properties. To test this idea 0.8MWCNT (hydrophobic MWCNTs) and 3.1MWCNT (hydrophilic

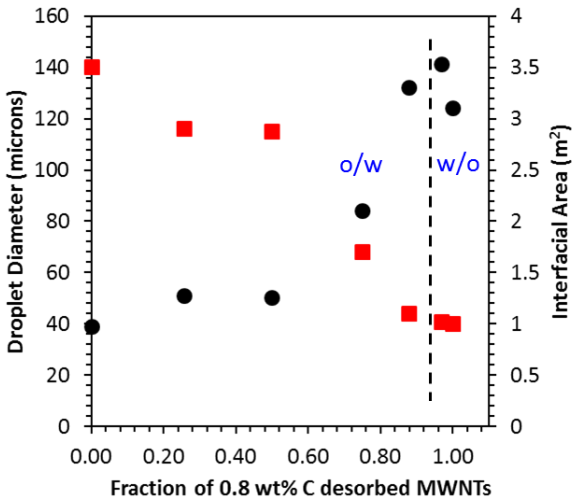


Figure 21: Effect of changing the ratio of MWCNTs with different wettability. 0.7 wt. % total MWCNT was maintained while varying the fraction of 0.8MWCNT to 3.1MWCNT. Change in emulsion droplet size and interfacial area per mass are shown. Circles (black) denote droplet diameter and squares (red) denote interfacial area.

MWCNTs) were mixed together at different fractions, while keeping the total MWCNT concentration constant. Both MWCNTs were initially dispersed in the water. As stated previously, no significant change was observed when dispersing the MWCNTs in oil initially.

Increasing the fraction of 0.8MWCNT to 3.1MWCNT produces an increase in emulsion droplet size initially, but rapidly increases with a fraction of 0.8MWCNT sample followed by an inversion in emulsion type (Figure 21). Emulsion type does not change until reaching a 0.97 fraction of 0.8MWCNT. This trend agrees well with what has been seen when silica particles of different hydrophobicity are mixed together in emulsions.[106]

Emulsion Droplet Size Change

Great difficulty arises when trying to determine the true cause for the change in emulsion droplet size as a function of MWCNT wettability. Carbon nanotubes inherently agglomerate together[99] and can be cut in length during sonication, or when

mixed with acid during oxidation treatment.[107] Furthermore, carbon nanotubes are not perfectly straight. All of these factors make it difficult to come up with an absolute conclusion to why the emulsion droplet size changes with carbon nanotube wettability. Therefore, we have come up with three hypotheses which are based on our results and information about Pickering emulsion in the literature and provide an explanation for which hypothesis we believe in. Our first hypothesis is based on carbon nanotube agglomeration due to van der Waals forces. A common way to increase the dispersion of MWCNTs is to oxidize the MWCNTs, creating hydrophilic functional groups which will generate charge repulsion between the MWCNTs. If the initial MWCNTs are not entirely dispersed or in some cases driven to the interface as thick mats then there will not be as many MWCNTs to stabilize all of the interfacial area created during emulsification. By oxidizing the MWCNTs, the degree of MWCNT agglomeration decreases. With less agglomeration of MWCNTs more MWCNTs are available to cover more interfacial area. However, if this were to be the dominant factor then the emulsion droplet size should decrease further for sample 4.4MWCNT. Instead, the emulsion droplet size increases, which can be seen with two different oils, dodecane (Figure 15) and heptane (Figure 19a). In fact, for the case of toluene, the emulsion droplet size increases for both 3.1MWCNT and 4.4MWCNT samples (Figure 19).

Our second hypothesis is based on the energy to adsorb/desorb a particle from the interface, which is a maximum when the contact angle is 90°. Therefore, when the MWCNTs have contact angles far above or below a contact angle of 90° the energy to remove the particle from the interface is low. When the MWCNTs can be easily removed from the interface there are not as many particles at the interface as when the

particles contact angle is 90°, thus more droplet coalescence occurs. Our calculations (see Equation 4 and Equation 5), as well as others[85,108,109], show a valley plot is obtained for cylinders with the highest energy required to remove the particle from the interface being at 90°. Cylinders at all contact angles require a greater energy to desorb from the interface than a sphere (Figure 6). Even when both the cylinders and spheres are the same diameter, the length of the cylinder increases the energy required much more than the sphere. This may explain why there was no noticeable change in emulsion droplet size or fraction of oil, emulsion, or water after a three month period. Very hydrophobic and hydrophilic fumed silica has been reported to produce emulsions with poor stability and large emulsion droplets[110], the size of which is similar to our most hydrophobic and hydrophilic MWCNTs. MWCNTs inherently agglomerate due to van der Waals forces, which will further increase the energy required to desorb them from the interface. The energy to remove a particle is dependent on its size; larger particles require more energy, as can be seen in Equation 4 and Equation 5. A surfactant,[111,112] polymer,[113] or salt[114] must be added for silica to flocculate together to obtain the same effect. In this regard the shape and agglomeration of MWCNTs makes them ideal emulsion stabilizers. However, the energy required for the MWCNTs to desorb from the interface being greater than a sphere leads us to believe this is not causing the change in emulsion droplet size. Our third hypothesis considers the idea that the MWCNT orientation may change depending upon the particle hydrophilicity which in turn causes a high or low surface area coverage state for the particles at the liquid-liquid interface. This phenomenon was hypothesized by Binks *et al.* for the change in emulsion droplet size when carrying the hydrophobicity of fumed silica particles through

functionalization.[15] Droplet coalescence continues to occur while the particles are adsorbing to the interface under non equilibrium conditions until the particles cover the entire droplet surface. The surface of the droplet becomes saturated with particles at some point; however, due to inhomogeneities in particle hydrophilicity, not all particles will have the same adsorption energy, and therefore may have different affinities for differing orientations. The more extreme hydrophilic or hydrophobic particles (residing more within the bulk phase) may flip their orientations to be in a state which occupies lower surface area upon the droplet, allowing for more coalescence of droplets. Particles strongly held at the interface do not change orientation and maintain their occupancy of high surface area, thereby limiting coalescence. Binks *et al.* attributes the fractal particles of fumed silica with causing changes in emulsion droplet size. Ellipsoidal polystyrene stabilizing particles have been observed in parallel and perpendicular orientations while stabilizing emulsion droplets.[115] In previous studies, we have observed MWCNTs in a perpendicular, slanted and parallel orientations, although most resided in the parallel orientation.[116] MWCNTs are cylindrical shape, not spherical, just as the fractal particles are non-spherical. Taking into account our results and previous studies we believe the same effect is occurring with the MWCNTs. MWCNTs held weakly at the interface can change orientation from occupying a high to low surface area, while MWCNTs that are held strongly at the interface occupy a high surface area. The energy with which the MWCNTs are held to the interface is dependent upon the MWCNTs wettability. MWCNTs held strongest at the interface have contact angles close to 90° and MWCNTs held weakly at the interface have contact angles approaching 0° or 180°.

Conclusion

Controlled functionalization of MWCNTs has allowed the properties of Pickering emulsions to be adjusted, while droplet size and emulsion inversion trends have been observed to be consistent amongst different oil-water systems. Different oil-water interfacial tensions reveal the trends expected from the proposed theory of energy required to remove a particle from the interface. Changing the oil-water ratio allows for emulsion type and emulsion droplet size to be changed when the MWCNTs are intermediately in hydrophobicity. In contrast, the most hydrophobic or hydrophilic MWCNTs do not change emulsion type when changing the oil-water ratio, although the droplet size may change. Mixing MWCNTs of different wettability in different ratios allows for control of emulsion droplet size and type. Our current evidence suggests the emulsion droplet size is dependent on the surface area occupied by MWCNTs at the interface, which is controlled by the MWCNT orientation and therefore the energy required to remove the particle from the interface. Further work may elucidate the mechanism which controls this phenomenon.

Chapter 4: Superparamagnetic Pickering Emulsions – Evidence for Facile and Reversible Catalyst Deactivation in Liquid Biphasic Reaction Systems

Introduction

Iron oxide nanoparticles, and in particular magnetite (Fe_3O_4), are one of the most widely studied superparamagnetic nanomaterials for catalyst supports and general engineering applications. These particles have been observed and studied for ages due to magnetite (lodestone) being one of the only naturally occurring magnetic materials. However, there is still active study to obtain a complete understanding of its magnetic properties at the quantum scale, such as in the Verwey transition of magnetite at ~130K. Recently, many works have focused on using this simple and ubiquitous material in stimuli responsive Pickering emulsions.

Controlling Pickering emulsion properties via stimuli responsive systems has been an intensely researched area in the past several years, with possible applications in oil recovery[117] and liquid phase catalysis[2,118]. Several possible parameters are available for controlling the stability of these emulsions including surfactants[119], pH[120], temperature, light[121], and magnetism. These methods typically involve modulation of the surface properties of a Pickering particle which stabilizes the oil-water interface. These surface modifications can be applied in situ via chemical alterations from an externally available tool (surfactant adsorbs to particle surfaces, heat and light can cause a conformation change in a molecule adsorbed to particle surface, etc.). Typically in these cases, the surface alteration, and therefore alteration of the hydrophobic or hydrophilic nature, causes a shift in the favorability for the particle to wet an oil or

water interface within the system. While this discovery is quite useful in many fields, this alteration of the surface chemistry may be a disadvantage in the application of these features in two phase reactions involving a solid supported catalyst, it potentially changes the catalytic activity of an active site.

In the first case of surfactants, the addition of any ionic surfactants in the system causes catalytic poisoning to take place via adsorption of sulfates, phosphates, or nitrates. Furthermore, in addition to interacting with the Pickering particle surface, there is potential for the surfactants to emulsify the system and potentially alter the solubility of the different small molecule organics or other chemical constituents, thereby creating a more complex system. This can be shown within aqueous two phase extractions, and may extend to any liquid-liquid system, although it is likely severely reduced in high surface tension oil-water environments.[122] In the case of light or heat, the surface typically responds in a manner which alters the Pickering particle's hydrophobicity, which may lead to larger (or smaller) mass transfer limitations for reactants.

However, one exception to these examples is magnetism. In the case of magnetically controlled Pickering emulsions, the particle stabilizers within this system do not change their surface chemistry or require the chemical environment to change in order to destabilize the system, but rather rely on force in order remove themselves from the kinetically stable system (A non-Janus Pickering particle emulsion is typically a kinetically stable system)[123]. One may hypothesize that this provides a few advantages for performing stimuli responsive reaction catalysis at liquid-liquid interfaces, as the emulsion thermodynamic state may remain unaltered. In addition, this

allows recovery of potentially expensive catalyst, which typically employ precious metals. Furthermore, although the effect of magnetism on catalytic properties of reduced metals is an active area of research[124], one may also hypothesize that by using an externally applied field to which only the catalyst support responds, the catalytic properties may be left intact.

However, the work presented here provides evidence to support the contrary. In this study we show that using magnetism to switch the position of magnetic particles within two phase systems using a hydrophilic catalyst support creates a mass transfer limited system via a film layer on the catalyst surface. One feature which segregates the magnetically controllable particles from the other emulsion switching methods is the potential to reversibly deactivate catalyst in situ for two phase reactions by removing the catalyst from its thermodynamically favored phase. Other stimuli responsive emulsion stabilizers are irreversibly deactivated via alteration to the catalytic active sites (salt adsorption and poisoning). In addition, to the potential use for reversible catalytic deactivation, this study provides experimental support for the computational modeling done by Fan *et al.*[125]

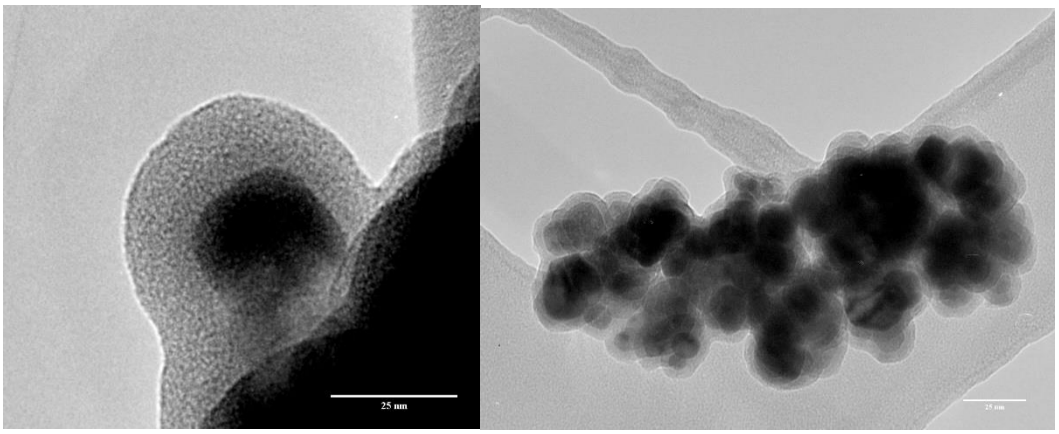


Figure 22: TEM images of Fe_3O_4@SiO_2 particles.

Results and Discussion

Fe_3O_4@SiO_2-Pd catalyst

In this work we have used magnetite nanoparticles (<50 nm) with a silica coating in order to create stimuli responsive catalyst particles for doing reactions in two phase systems. This has been achieved using a standard method for creating core-shell particles via a modified Stöber process.[126] In order to control the growth of silica on magnetite, rather than growing silica only particles, we have performed the reaction at room temperature, with very little water present, and a small amount of TEOS present. This is due to the selectivity for the condensation reaction over the hydrolysis reaction

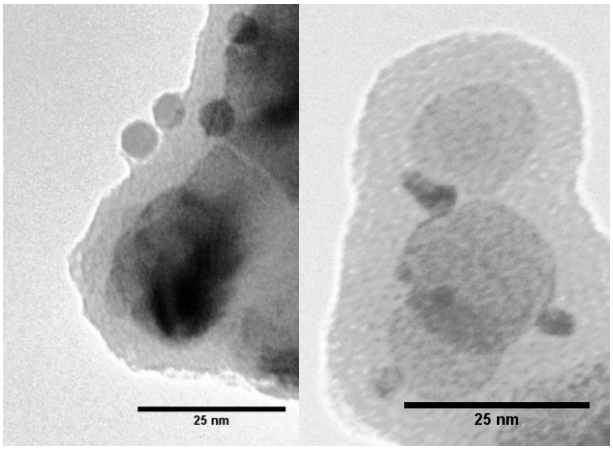


Figure 23: TEM images of Fe_3O_4@SiO_2-Pd particles.
Palladium oxide particles are measured at ~6 nm in diameter.

at low temperature, low pH, low TEOS concentration, and low water content.[126] TEM images were taken to confirm that the magnetite was well coated with a uniform layer of silica. It was found that the approximate thickness of the silica shell is about 15 nanometers on top of Fe_3O_4 particles with a diameter of about 25 nanometers.

Following the silica deposition, we have impregnated and calcined our core-shell support with a water soluble palladium nitrate and found the resulting palladium particles to be approximately 5 nanometers in diameter, as indicated by Figure 23 and Figure 24. TEM images confirm the separation of magnetite from palladium catalyst particles. It is highly unlikely for iron oxide to be reduced to metallic iron until very high temperatures (~600°C) in a hydrogen environment. However, the hydrogen dissociated by palladium at low temperatures may reduce iron oxide to metallic iron at the local active site. Therefore, we have gathered temperature programmed reduction profiles for our catalyst.

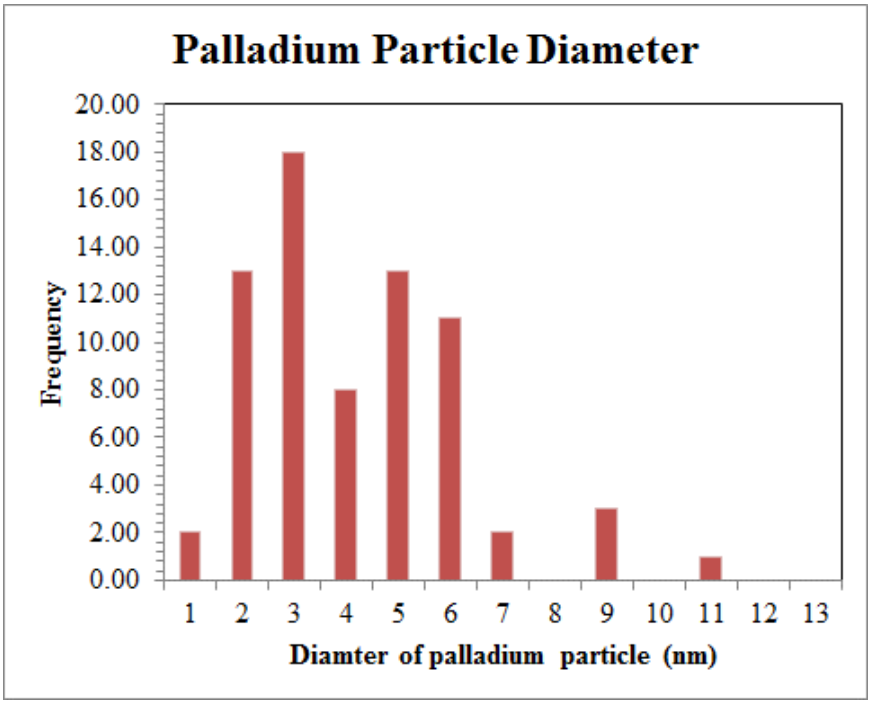


Figure 24: Palladium catalyst particle diameter.

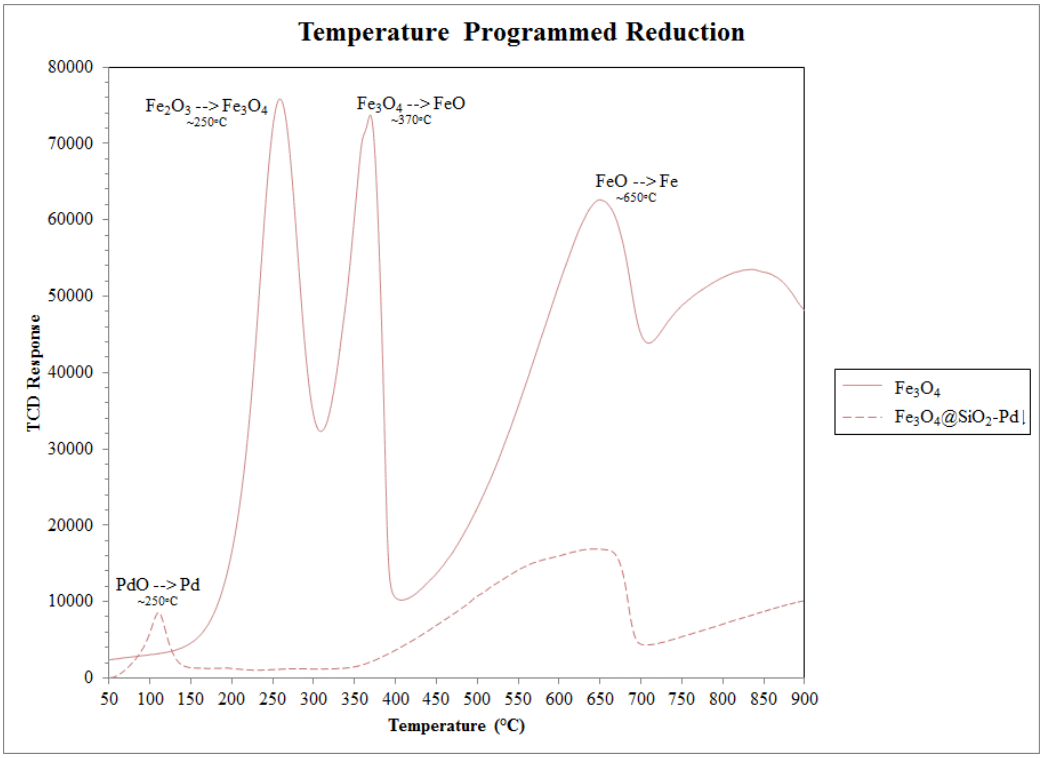


Figure 25: TPR of magnetite and magnetic catalyst.

.

Although the reduction temperatures for magnetite and palladium are well documented[127], temperature programmed reduction (TPR) data were acquired for Fe_3O_4 and Fe_3O_4@SiO_2-Pd in order to demonstrate 1) our material does have some amount of other iron oxide forms which have been reported to reduce at ~250°C (hematite or Fe_2O_3), 2) our reactions will operate at well below the reduction temperature of magnetite or any iron oxide (70°C), and 3) there is an apparent absence of reduction for Fe_3O_4 and Fe_2O_3 in the magnetite silica catalyst. This implies that the silica shell is providing a mass transfer limitation to the hydrogen diffusing to the iron oxide layer. This observation provides support for the iron oxide remaining catalytically inactive while the palladium performs its typical catalyst role.

Reactions

Here, we have chosen to study the hydrogenation of alkenes over palladium, as it is a well-studied reaction that is capable of being performed at low temperature and low pressure. Following our beginning hypothesis and goal of controlling phase selectivity, we have selected two alkenes which are highly soluble and partition strongly in water and oil. Therefore, a very small diol (cis-2-butene-1,4-diol) was chosen for the water phase reactant, providing a high density for hydrogen bonding, and a long, linear alkene (1-dodecene) was chosen for the oil soluble reactant.

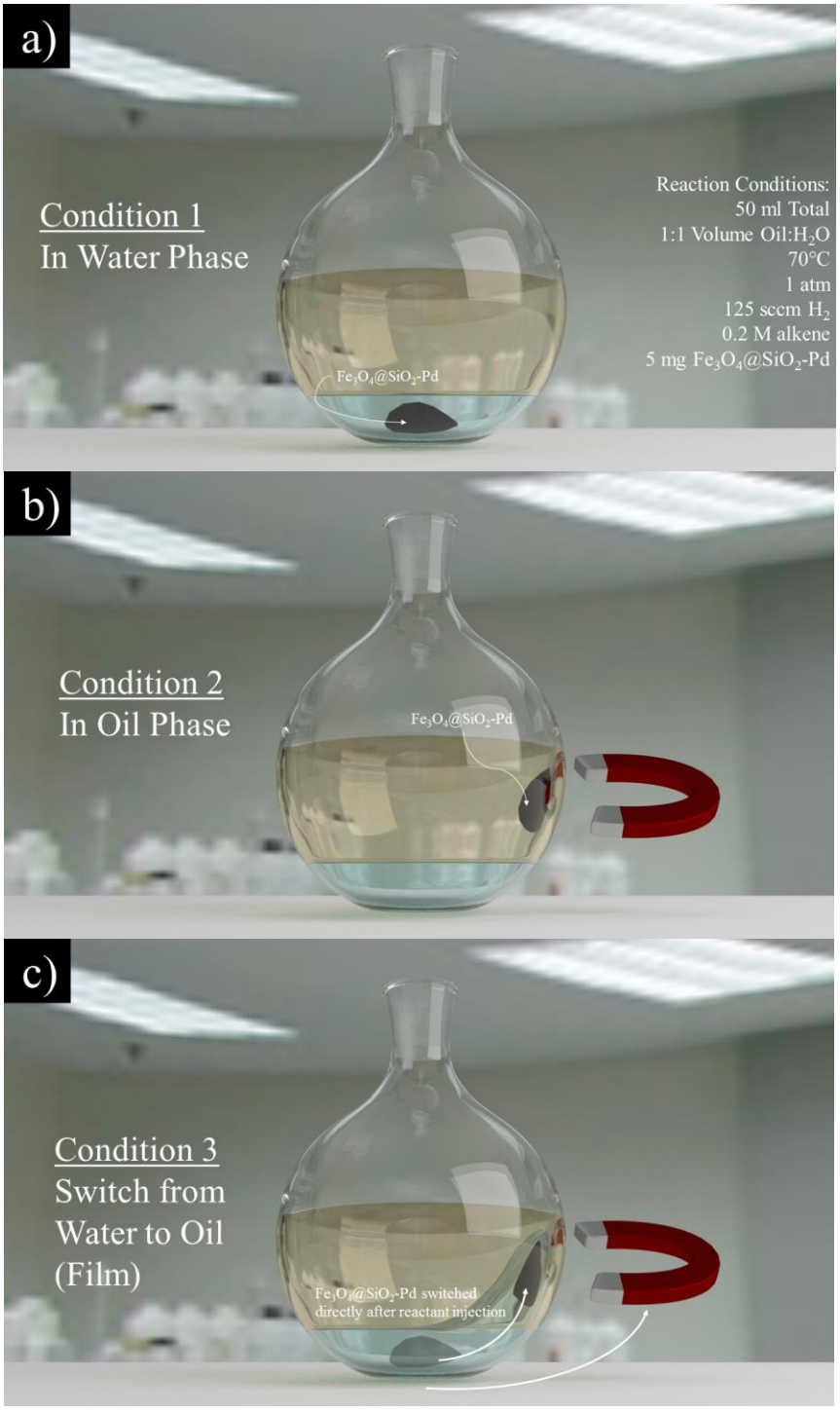


Figure 26: Illustrations of each reaction condition used within this study.

In order to use optically transparent laboratory glassware which allows more precise control and monitoring of magnetically manipulated catalyst placement, the reaction conditions were chosen at 70°C and 1 atm of pure H_2 at 125 sccm (further

reaction conditions given in the Experimental section). This turned out to be a key observation within the study, as the two phase heights were approximately the size of the neodymium magnet used to control the particles, requiring careful adjustment of the magnet placement for exclusion of phase contact with catalyst particles.

The magnetic catalyst's behavior in responsive reactions was characterized via conducting three different magnetically controlled catalyst position conditions for our reaction system. The three positions and an overview of the reactions are shown in Figure 26. The figure displaying the switching of the magnet from the water to oil phase (c) shows an exaggerated image of the film created by pulling the catalyst particles across the interface from the water to oil phase. A more microscopic image of the hypothesized behavior is shown in Figure 27.

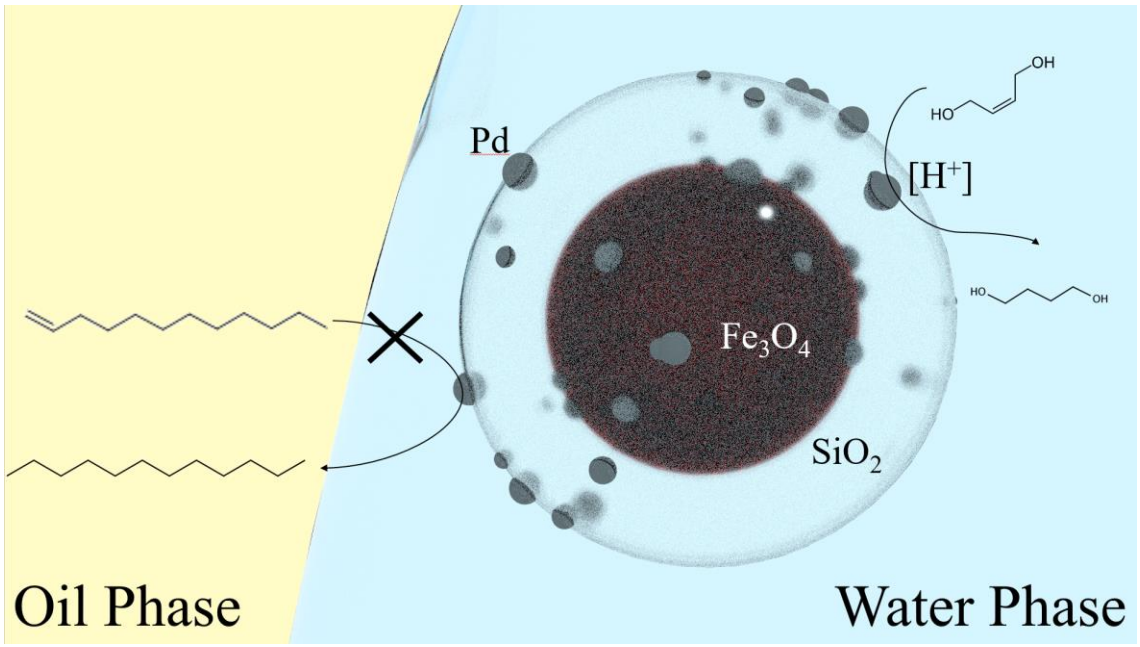


Figure 27: Illustration of a water film around the hydrophilic Fe_3O_4-SiO_2-Pd catalyst.
1-dodecene conversion of dodecane is hindered from mass transfer limitations through the water phase.

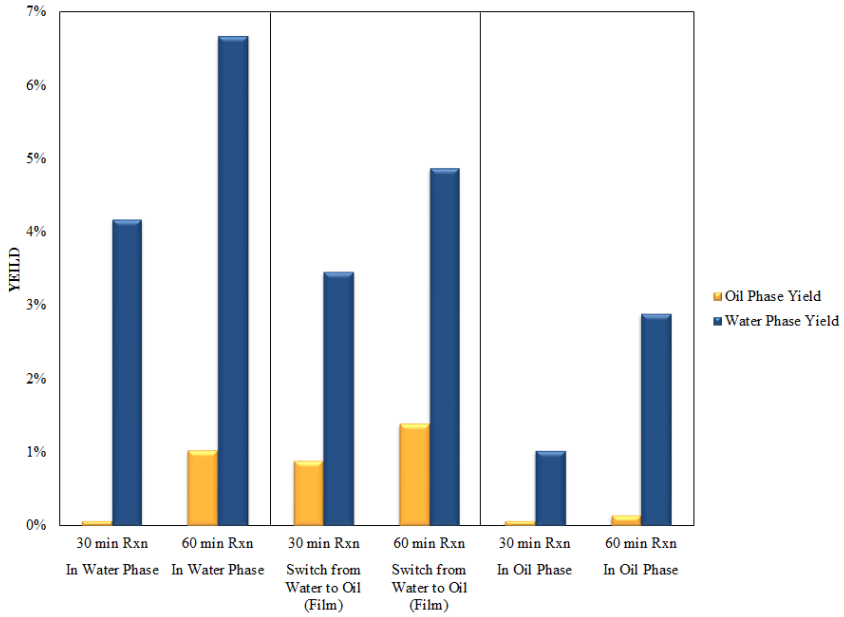


Figure 28: Oil and Water phase yields for different reaction times and magnet locations.

Figure 28 shows the reaction results in terms of yield of each alkane product after reaction in each condition for two reaction times. Each condition indicates an increase in the yield of 1,4-butanediol (water phase product), while the oil phase product is nearly constant and non-reacting. This result is surprising; however, is typical of phase transfer limitations, with reactants of low solubility. While the reactions performed completely in the oil phase are unexpected (the catalyst was carefully not allowed to come into contact with the water phase), one potential explanation is the transfer of moisture from the interface via hydrogen bubbling (the hydrogen inlet was placed at the bottom of the water phase). This may have allowed for the buildup of a film regardless of the placement of the catalyst. Another potential hypothesis is that heat transfer limitations were created by only having agitation available from hydrogen bubbling provided to the system without stirring. Stirring was not used for this study in order to ensure that the reaction was operating under a severe

mass transfer limited regime in order to test for phase selectivity after applying a magnetic field to alter catalyst placement. Therefore, further study is required in order to validate the hypotheses provided here.

Experimental

Materials and Equipment

Fe_3O_4 particles (<50 nm), 1-dodecene (95%), dodecane (99%), 1,4-butanediol (99%), and cis-2-butene-1,4-diol (95%), tetraethyl orthosilicate (95%), ammonium hydroxide (50%), palladium nitrate dihydrate, and ethanol (99%) were all obtained from Sigma Aldrich. All water used within the experiments was obtained from an 18 MΩ ultra-pure Cole Parmer filtration system.

Fe_3O_4@SiO_2: Hydrophilic core-shell catalyst support particle synthesis

5 g of Fe_3O_4 particles were dispersed in 160 ml of H_2O (18 MΩ) with a Fisher Scientific Model 505 Sonic Dismembrator (¼ inch tip) at 20% amplitude for 10 minutes. This dispersion was then added to 700 ml ethanol and 20 ml of 25 wt. % NH_4OH while 80 ml of TEOS was added dropwise. This solution was sonicated at 20% amplitude with a ¼ inch tip for 5 hours, allowed to cool and recovered via Neodymium magnet (~ 1 Tesla) overnight before the paramagnetic Fe_3O_4@SiO_2 solids were collected and washed 5 times with acetone.

Fe3O4@SiO2-Pd: Addition of palladium catalyst for reactions

Following Fe_3O_4@SiO_2 collection, 131.7 mg $Pd(NO_3)_2 \cdot 2\ H_2O$ was dispersed in 200 μL H_2O (2.47 M Pd aqueous solution) and was added to 1 g of the catalyst support (Fe_3O_4@SiO_2) dropwise while mixing with a mortar and pestle for wet impregnation of the Palladium catalyst in order to create a 5 wt. % Pd loading on the Fe_3O_4@SiO_2.

These particles were then calcined at 250°C for 2 hours after a 2 °C/min temperature ramp and recovered for subsequent reactions.

Reactions

25 ml of water, 25 ml decalin, and 5 mg of Fe_3O_4@SiO_2-Pd catalyst were placed into a 50 ml three neck round bottom flask and heated to 70°C in pure H_2 flowing at 125 sccm. The system was held in this state for 1 hour in order to reduce the palladium oxide to palladium metal, after which 5 ml of each liquid phase, containing 1.2 M reactant was injected (taken from a batch of 5.905 g cis-2-buten-1,4-diol in 50 ml H_2O and 13.795 g 1-dodecene in 50 ml decalin) . The reaction was allowed to continue for either 30 minutes or 1 hour, after which the heat was removed via placing the reactor in cool water and purging the system with atmospheric nitrogen. The three different configurations of the reaction systems were manipulated by placing a magnet on the side of the reactor and adjusting the location of the particles until they visually resided within the desired phase. The first configuration (denoted as “water phase”) was the simplest case, in which the catalyst particles were first added to the reactor, followed by water in order to wet them. Careful addition of the oil phase ensured that the oil had no contact with the catalyst particles. The second configuration (denoted as “oil phase”) was performed by placing the magnet at the side of the reactor, followed by pouring in the water and oil phases while ensuring the water phase did not contact the particles. After this, the catalyst was adjusted to visually be within only the oil phase. The third configuration was done by combining the first configuration (no magnet is provided at the start of the reaction), followed by a change in catalyst particle position to the oil phase after injection of the reactant. Approximately 30 seconds were allowed for

mixing of the system's reactants was given before moving the catalyst particles to the oil phase.

The catalyst was then separated from the two liquid phases via magnetic separation by a strong neodymium magnet and syringe filtration through a 0.45 μm filter, followed by the addition of external standards (225 mg decane in decalin/dodecane/dodecane phase, 100 mg butanol in water/cis-2-butene-1,4-diol/1,4-butanediol phase) to 10 ml of each phase in order to more precisely calculate the molarity of the reactants and products. Small aliquots of each phase were then provided to a gas chromatography system (Agilent 7890B GC) for analysis.

Conclusion

Magnetically controllable superparamagnetic particles were synthesized and utilized for hydrogenation in two phase reaction systems. Some evidence has been provided that in systems which contain solvents of differing hydrophobic and hydrophilic character, films are created on catalyst supports consisting of the solvent in which the particle is more soluble. This film is hypothesized to create a special type of mass transfer limitation recently referred to as a phase limitation within the literature. Although more study is required, this work has provided some experimental validation of the water bridging effect noted by Fan *et al* and has potential for controlling and reversibly deactivating catalyst in two phase systems in situ.

Future Prospects

Many other potential avenues for using CNTs within two phase systems are available to study for the coming years. In order to provide the future graduate student or researcher some inspiration, I have provided several ideas which were uncovered throughout this work, yet were not fully explored. These ideas include the following:

- Phase transfer catalysis (PTC) with water solubilized CNTs
- Aqueous two phase systems (ATPS):
    - Phase separation of small sugars from water soluble potential biofuel chemicals
    - Phase selectivity via controllable separation of SWCNT supported catalysts
    - Removal of humins into separate aqueous phase

Phase Transfer Catalysis (PTC) with CNT

While pristine nanotubes are not typically soluble in any liquids other than superacids and strong electrolyte solutions,[128–130] with extensive surface modification they can be made to be much more hydrophilic with the addition of oxygen containing groups and sulfate groups. This allows for a separation of CNTs by length via hydrophobic and hydrophilic interactions by introducing an oil phase to the water dispersed CNTs. In the case of highly sulfonated CNTs, we may employ a cationic surfactant known as CTAB in order to ionically interact with the sulfate groups on the CNT surface. This in turn acts as a standard phase transfer catalyst, creating a more hydrophobic complex, and transfers the CNT into the oil phase. This method has been used to length fractionate CNTs, although it is easy to envision this system within a two

phase reaction for biofuel production with the addition of transition metals via impregnation and calcination of catalyst onto the surface of the CNT.

SWCNT Separationss

In order to have high performance from certain carbon nanotube applications, some level of separation is typically required. For example, Jain *et al.* were able to observe that purification of SWCNTs by pure chirality yielded an increase by 3,000% (30X) in the efficiency of an organic solar cell when comparing an 80:20% (6,5):(6,4) mixture and a 100% (6,5) nanotube solution.[19] It should be noted by the reader that both of these samples are remarkably well separated, and that having a SWCNT material with >45% of any chirality is a very difficult task to achieve. The study performed by Jain *et al.* clearly shows the importance of separating SWCNTs by electronic character for any application in which electronics are contributing to the function of the product.

Various methods for separating CNT materials by length, diameter, defect density, and chirality have been proposed and studied.[131,83,132,129,133] These methods consist of standard silica gel chromatography, adsorption, amine functional group functionalization, polymer wrapping, and two phase transfer systems.

Aqueous Two Phase Extraction (ATPE)

In light of recent reports of interfacial reaction engineering for biofuel production[2], much attention has been paid to particles within two phase systems and their behavior at interfaces. Oil-water systems typically provide large interfacial tensions which (when combined with the low solubility of CNTs in almost any solvent) tend to form Pickering emulsions. However, one may envision a system in which the

solubility of the SWCNT in the bulk phase is more preferable, and the surface tension between the phases is low. This is the case of aqueous two phase systems, which have been developed at NIST for separating SWCNTs[83] and, while the idea is not fully studied here, this observation may be used to perform phase transfer catalysis (PTC) with SWCNT supported catalyst. This idea can be demonstrated in Figure 29.

The benefits of performing reactions at the interface of a two phase system have been shown in many works; however, the pyrolysis of biofuel feedstock provides not only a portion of water and aqueous phase soluble products, but also reactants which form insoluble polymers (humins) which are undesirable for biofuel production. It is therefore of considerable interest to either research facile separation techniques for the polymer (humins) removal, or to hinder their formation during reaction. Although there is much work to be done within the area, there is potential that the aqueous two phase system may provide solutions to this problem using either of these methods (formation hindrance or polymer removal).

The aqueous two phase system typically consists of a sugar polymer (such as dextran) or salt, and a slightly more hydrophobic polymer phase (typically polyethylene glycol). This can be extended to provide systems with up to 5 or more phases, all

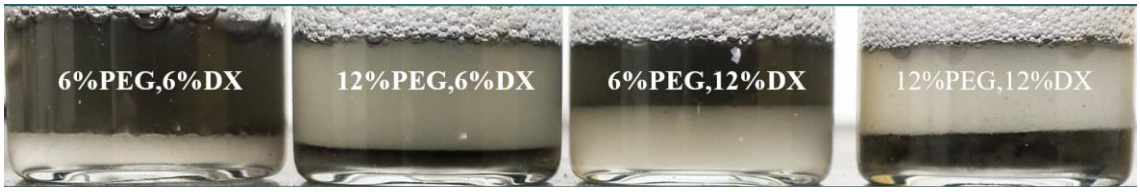


Figure 29: SWCNTs switching phases in ATPS.
PEG and DX denote polyethylene glycol 6000 MW and dextran 75,000 MW. The percentages are given as weight percent of total, while the other constituents are sodium cholate and sodium dodecyl sulfate (SC:SDS) at a 9:7 ratio at 1.47 wt % total surfactant.

consisting of a continuous water environment.[134] These different phases provide different environments which may be tuned *very* precisely by varying temperature, polymer type, polymer molecular weight, polymer weight percent, pH, surfactant type, surfactant amount, as well as chaotropic and kosmotropic salts.

The enormous parameter space given by the ability to vary pH, temperature, molecular weights and loading of polymers, surfactant loading, etc. is a wonderful tool for potentially creating a precisely tuned reaction system for phase transfer catalysis; although one's experimental plan for exploring the number of available variables can become quite burdensome when considering the lab work required.

Reports within the SWCNT separations community have shown the phase transfer of different SWCNT (n,m) species at very specific conditions (~0.04 wt % sodium deoxycholate, ~6 wt % polyethylene glycol [6000 molecular weight], ~6 wt % dextran) by varying the surfactant sodium dodecyl sulfate between ~0.5-1.4 wt %.[83] The

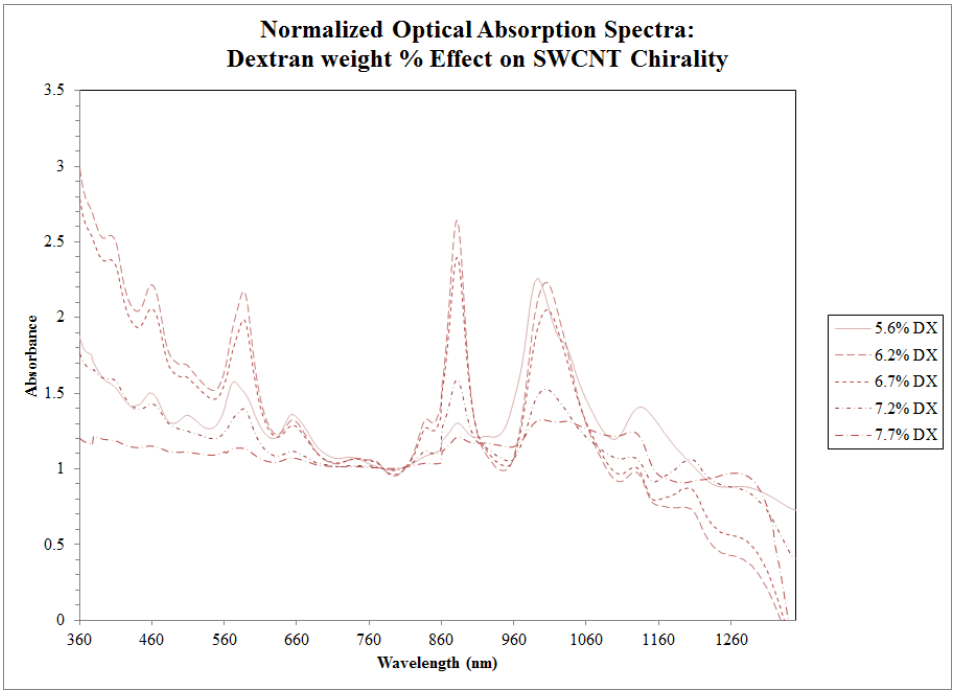


Figure 30: Effect of dextran on chirality of SWCNTs in dextran phase.

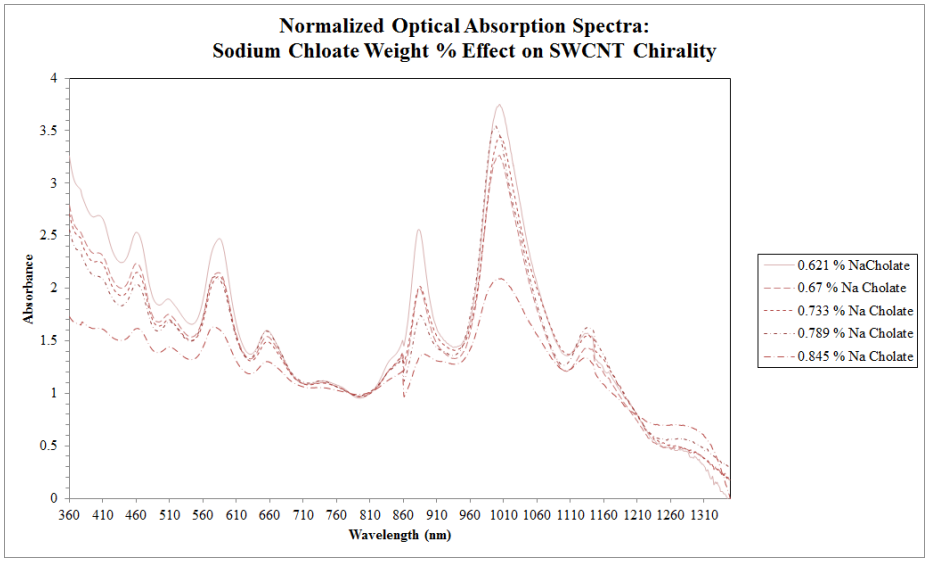


Figure 31: Effect of sodium cholate on chirality of SWCNTs in dextran phase.

literature has documented the partitioning behavior of most common (n,m) species, revealing that the large diameter semiconductors partition to more hydrophobic environments (PEG), while the metallic SWCNTs and smaller diameter SWCNTs (<1 nm) prefer the hydrophilic phase. The partitioning of the smaller diameter semiconductors has been attributed to the ability for structures less than ~1 nm to 'hide' within the hydrogen bonding network of water.[135]

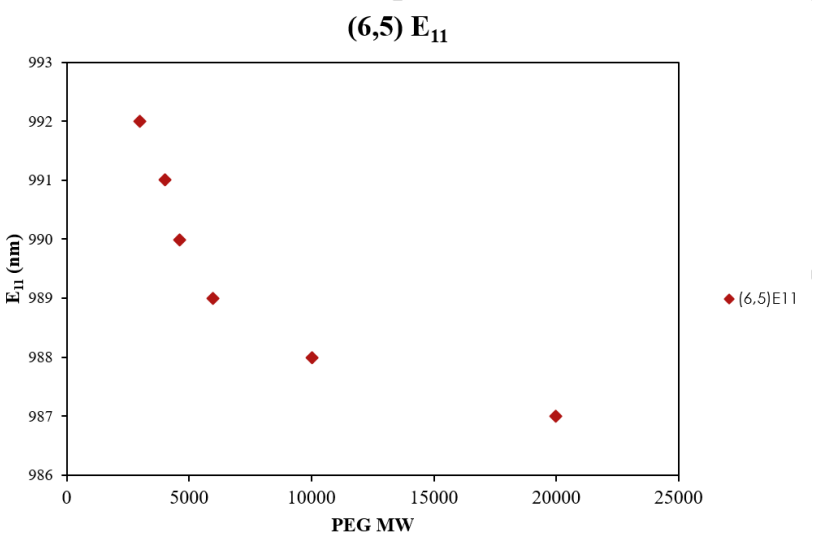


Figure 32: Optical effect from increasing PEG molecular weight.

.

Some preliminary data was gathered for studies which changed the surfactant concentration, polymer loading, and polymer molecular weight. Increasing the dextran loading appears to quickly shift the smaller diameter SWCNTs to the dextran (more hydrophilic) phase, although it is quickly quenched. The subsequent quenching may be due to bundle formation, which leads to the observed linewidth increases or broader energy spectra. The increase of sodium cholate causes the SWCNTs and any other forms of carbon to migrate to the dextran phase, as indicated in Figure 31. We may also vary the polymer molecular weights, which appear to wrap the SWCNTs better within the top phase, creating smaller line widths for the peaks, indicating better individualization. Additionally, we can see optical effects of solvatochromic shifts occurring in the E_{11} transition for (6,5) in Figure 32.

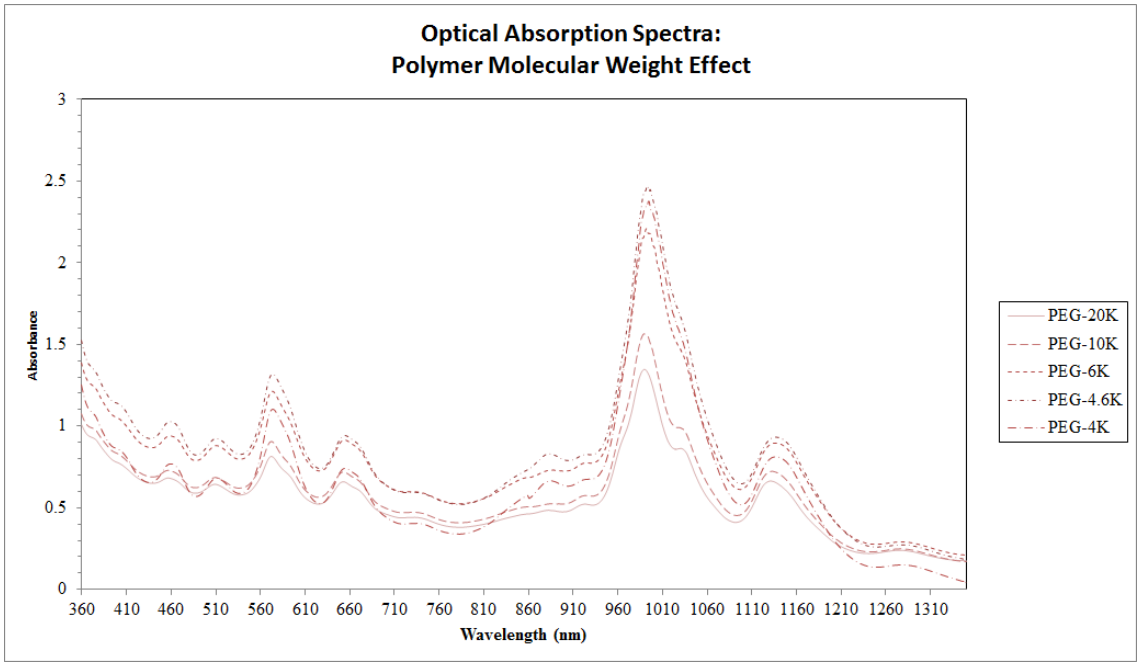


Figure 33: Effect of PEG molecular weight on the line widths of the SWCNT optical absorption excitonic transitions in the PEG phase.

.

Further, one may use a different facile SWCNT separation technique such as the chirality dependent gel adsorption method developed at JIST[131,136,137,137–140,133], in order to obtain specific SWCNT (n,m) species, and subsequently place those within the NIST developed aqueous two phase system. In this manner, one may impregnate and calcine a different metal catalysts upon two differing SWCNT (n,m) catalyst supports, and have the ability to perform different reactions within each phase. This allows for the treatment of strikingly similar molecules in completely different manners.

Although the potential applications for highly tuned environments in reaction engineering with aqueous environments may run into difficult issues such as poisoning and deactivation of catalyst from salts and surfactants, mass transfer problems caused from presence of large polymers, and competitive reactions from catalytic breakdown of polymer, other useful applications may still exist for using ATPS in reaction engineering. For example, some work has shown that the two phase system has the potential of removing humic acids, which are a large problem within biofuel production due to the condensation of furfural, HMF, furfuyl alcohol, etc.[141] Therefore, the ATPS may be tuned such that the catalyst is soluble to perform reactions within the hydrophilic aqueous phase, while the hydrophobic aqueous phase is left to remove the humic acids.

Conclusion

Methods for the facile characterization of carbon nanotube materials have been presented along with procedures and scripts that may provide a starting point for future scientists researching CNTs and their applications. These methods have been applied to understanding the role of CNT wettability in emulsions and their stability, and paved the way for creating stable and tunable biphasic reaction systems. Furthermore, while studying stimuli responsive Pickering emulsions, we have elucidated a potential fundamental difference for magnetically controllable particles within stimuli responsive liquid-liquid systems. Although, further work is required to confirm this hypothesis, this work has provided some promising preliminary results for evidence of reversible in-situ catalyst deactivation.

References

1. Wu, T. *et al.* Multi-body coalescence in Pickering emulsions. *Nat. Commun.* **6,** (2015).
2. Crossley, S., Faria, J., Shen, M. & Resasco, D. E. Solid nanoparticles that catalyze biofuel upgrade reactions at the water/oil interface. *Science* **327,** 68–72 (2010).
3. Purceno, A. D. *et al.* Magnetic amphiphilic hybrid carbon nanotubes containing N-doped and undoped sections: powerful tensioactive nanostructures. *Nanoscale* **7,** 294–300 (2015).
4. Rickerby, S. Original painting techniques and materials used in the Tomb of Nefertari. *Art Eternity Nefertari Wall Paint. Conserv. Proj.* **1992,** 43–53 (1986).
5. Scholz, H. A. History of Water-Thinned Paints. *Ind. Eng. Chem.* **45,** 709–711 (1953).
6. Lagrange, V., Whitsett, D. & Burris, C. Global Market for Dairy Proteins. *J. Food Sci.* **80,** A16–A22 (2015).
7. Tadros, T. F. *Applied surfactants: principles and applications*. (John Wiley & Sons, 2006).
8. McClements, D. J. *Food emulsions: principles, practices, and techniques*. (CRC press, 2015).
9. Cockbain, E. The adsorption of sodium dodecyl sulphate at the oil-water interface and application of the Gibbs equation. *Trans. Faraday Soc.* **50,** 874–881 (1954).
10. Pickering, S. U. CXCVI.—Emulsions. *J. Chem. Soc. Trans.* **91,** 2001–2021 (1907).
11. Binks, B. & Lumsdon, S. Influence of particle wettability on the type and stability of surfactant-free emulsions. *Langmuir* **16,** 8622–8631 (2000).

12. San-Miguel, A. & Behrens, S. H. Influence of Nanoscale Particle Roughness on the Stability of Pickering Emulsions. *Langmuir* **28,** 12038–12043 (2012).
13. Nguyen, B. T., Wang, W., Saunders, B. R., Benyahia, L. & Nicolai, T. pH-Responsive Water-in-Water Pickering Emulsions. *Langmuir* **31,** 3605–3611 (2015).
14. Tadros, T. F. *Emulsion formation and stability*. (John Wiley & Sons, 2013).
15. Binks, B. P. *et al.* Drop sizes and particle coverage in emulsions stabilised solely by silica nanoparticles of irregular shape. *Phys. Chem. Chem. Phys.* **12,** 11967–11974 (2010).
16. Briggs, N. *et al.* Wettability of Multi-Walled Carbon Nanotubes Tuned to Control Pickering Emulsions Properties.
17. Iijima, S. & Ichihashi, T. Single-shell carbon nanotubes of 1-nm diameter. *Nature* **363,** 603–605 (1993).
18. Oberlin, A., Endo, M. & Koyama, T. Filamentous growth of carbon through benzene decomposition. *J. Cryst. Growth* **32,** 335–349 (1976).
19. Jain, R. M. *et al.* Polymer-Free Near-Infrared Photovoltaics with Single Chirality (6,5) Semiconducting Carbon Nanotube Active Layers. *Adv. Mater.* **24,** 4436–4439 (2012).
20. Dang, X. *et al.* Virus-templated self-assembled single-walled carbon nanotubes for highly efficient electron collection in photovoltaic devices. *Nat. Nanotechnol.* **6,** 377–384 (2011).
21. Shulaker, M. M. *et al.* Carbon nanotube computer. *Nature* **501,** 526–530 (2013).

22. Bauhofer, W. & Kovacs, J. Z. A review and analysis of electrical percolation in carbon nanotube polymer composites. *Compos. Sci. Technol.* **69,** 1486–1498 (2009).
23. Moniruzzaman, M. & Winey, K. I. Polymer Nanocomposites Containing Carbon Nanotubes. *Macromolecules* **39,** 5194–5205 (2006).
24. Saleh, T. A. & Gupta, V. K. Column with CNT/magnesium oxide composite for lead(II) removal from water. *Environ. Sci. Pollut. Res.* **19,** 1224–1228 (2011).
25. Li, Y.-Q., Samad, Y. A., Polychronopoulou, K., Alhassan, S. M. & Liao, K. Carbon Aerogel from Winter Melon for Highly Efficient and Recyclable Oils and Organic Solvents Absorption. *ACS Sustain. Chem. Eng.* **2,** 1492–1497 (2014).
26. Sui, Z., Meng, Q., Zhang, X., Ma, R. & Cao, B. Green synthesis of carbon nanotube–graphene hybrid aerogels and their use as versatile agents for water purification. *J. Mater. Chem.* **22,** 8767 (2012).
27. Sun, H. *et al.* Hydrophobic carbon nanotubes for removal of oils and organics from water. *J. Mater. Sci.* **49,** 6855–6861 (2014).
28. Gui, X. *et al.* Elastic shape recovery of carbon nanotube sponges in liquid oil. *J. Mater. Chem.* **22,** 18300 (2012).
29. Du, R., Zhao, Q., Zhang, N. & Zhang, J. Macroscopic Carbon Nanotube-based 3D Monoliths. *Small* **11,** 3263–3289 (2015).
30. Chen, W., Liu, X., Liu, Y., Bang, Y. & Kim, H.-I. Preparation of O/W Pickering emulsion with oxygen plasma treated carbon nanotubes as surfactants. *J. Ind. Eng. Chem.* **17,** 455–460 (2011).

31. Planeix, J. M. *et al.* Application of Carbon Nanotubes as Supports in Heterogeneous Catalysis. *J. Am. Chem. Soc.* **116,** 7935–7936 (1994).
32. Singhal, R., Mochalin, V. N., Lukatskaya, M. R., Friedman, G. & Gogotsi, Y. Separation and liquid chromatography using a single carbon nanotube. *Sci. Rep.* **2,** (2012).
33. Saridara, C. & Mitra, S. Chromatography on Self-Assembled Carbon Nanotubes. *Anal. Chem.* **77,** 7094–7097 (2005).
34. Liu, Z., Robinson, J. T., Tabakman, S. M., Yang, K. & Dai, H. Carbon materials for drug delivery & cancer therapy. *Mater. Today* **14,** 316–323 (2011).
35. Blackburn, J. L. *et al.* Transparent Conductive Single-Walled Carbon Nanotube Networks with Precisely Tunable Ratios of Semiconducting and Metallic Nanotubes. *ACS Nano* **2,** 1266–1274 (2008).
36. Mirri, F. *et al.* High-Performance Carbon Nanotube Transparent Conductive Films by Scalable Dip Coating. *ACS Nano* **6,** 9737–9744 (2012).
37. Simien, D. *et al.* Influence of Nanotube Length on the Optical and Conductivity Properties of Thin Single-Wall Carbon Nanotube Networks. *ACS Nano* **2,** 1879–1884 (2008).
38. Zhao, H. & Mazumdar, S. Electron-Electron Interaction Effects on the Optical Excitations of Semiconducting Single-Walled Carbon Nanotubes. *Phys. Rev. Lett.* **93,** 157402 (2004).
39. Elias, D. C. *et al.* Dirac cones reshaped by interaction effects in suspended graphene. *Nat. Phys.* **7,** 701–704 (2011).

40. Attaccalite, C. & Rubio, A. Fermi velocity renormalization in doped graphene: Fermi velocity renormalization in doped graphene. *Phys. Status Solidi B* **246,** 2523–2526 (2009).
41. Lin, H. *et al.* Measuring many-body effects in carbon nanotubes with a scanning tunneling microscope. *ArXiv Prepr. ArXiv09070344* (2009).
42. Liu, K. *et al.* An atlas of carbon nanotube optical transitions. *Nat. Nanotechnol.* **7,** 325–329 (2012).
43. Ishii, H. *et al.* Direct observation of Tomonaga–Luttinger-liquid state in carbon nanotubes at low temperatures. *Nature* **426,** 540–544 (2003).
44. Bockrath, M. *et al.* Luttinger-liquid behaviour in carbon nanotubes. *Nature* **397,** 598–601 (1999).
45. Angelakis, D. G., Huo, M., Kyoseva, E. & Kwek, L. C. Luttinger Liquid of Photons and Spin-Charge Separation in Hollow-Core Fibers. *Phys. Rev. Lett.* **106,** 153601 (2011).
46. Shi, Z. *et al.* Observation of a Luttinger-liquid plasmon in metallic single-walled carbon nanotubes. *Nat. Photonics* **9,** 515–519 (2015).
47. Tame, M. Nanophotonics: Liquid quantum photonics. *Nat. Photonics* **9,** 485–487 (2015).
48. Kuemmeth, F., Ilani, S., Ralph, D. C. & McEuen, P. L. Coupling of spin and orbital motion of electrons in carbon nanotubes. *Nature* **452,** 448–452 (2008).
49. Dresselhaus, M. S., Jorio, A., Hofmann, M., Dresselhaus, G. & Saito, R. Perspectives on Carbon Nanotubes and Graphene Raman Spectroscopy. *Nano Lett.* **10,** 751–758 (2010).

50. Dresselhaus, M. S., Dresselhaus, G., Saito, R. & Jorio, A. Raman spectroscopy of carbon nanotubes. *Phys. Rep.* **409,** 47–99 (2005).
51. Ferrari, A. C. & Basko, D. M. Raman spectroscopy as a versatile tool for studying the properties of graphene. *Nat. Nanotechnol.* **8,** 235–246 (2013).
52. Ganzhorn, M., Klyatskaya, S., Ruben, M. & Wernsdorfer, W. Strong spin-phonon coupling between a single-molecule magnet and a carbon nanotube nanoelectromechanical system. *Nat. Nanotechnol.* **8,** 165–169 (2013).
53. Polini, M. *et al.* Plasmons and the spectral function of graphene. *Phys. Rev. B* **77,** (2008).
54. Takase, M. *et al.* Selection-rule breakdown in plasmon-induced electronic excitation of an isolated single-walled carbon nanotube. *Nat. Photonics* **7,** 550–554 (2013).
55. Dresselhaus, M. S., Dresselhaus, G., Saito, R. & Jorio, A. Exciton Photophysics of Carbon Nanotubes. *Annu. Rev. Phys. Chem.* **58,** 719–747 (2007).
56. Choi, S., Deslippe, J., Capaz, R. B. & Louie, S. G. An Explicit Formula for Optical Oscillator Strength of Excitons in Semiconducting Single-Walled Carbon Nanotubes: Family Behavior. *Nano Lett.* **13,** 54–58 (2013).
57. Blackburn, J. L., Holt, J. M., Irurzun, V. M., Resasco, D. E. & Rumbles, G. Confirmation of K-Momentum Dark Exciton Vibronic Sidebands Using ^{13}C-labeled, Highly Enriched (6,5) Single-walled Carbon Nanotubes. *Nano Lett.* **12,** 1398–1403 (2012).

58. Bindl, D. J. & Arnold, M. S. Efficient Exciton Relaxation and Charge Generation in Nearly Monochiral (7,5) Carbon Nanotube/C_{60} Thin-Film Photovoltaics. *J. Phys. Chem. C* **117,** 2390–2395 (2013).

59. Cognet, L. *et al.* Stepwise Quenching of Exciton Fluorescence in Carbon Nanotubes by Single-Molecule Reactions. *Science* **316,** 1465–1468 (2007).

60. Duque, J. G. *et al.* Quantum Interference between the Third and Fourth Exciton States in Semiconducting Carbon Nanotubes Using Resonance Raman Spectroscopy. *Phys. Rev. Lett.* **108,** (2012).

61. Haroz, E., Bachilo, S., Weisman, R. & Doorn, S. Curvature effects on the E33 and E44 exciton transitions in semiconducting single-walled carbon nanotubes. *Phys. Rev. B* **77,** (2008).

62. Koyama, T., Miyata, Y., Kishida, H., Shinohara, H. & Nakamura, A. Photophysics in Single-Walled Carbon Nanotubes with (6,4) Chirality at High Excitation Densities: Bimolecular Auger Recombination and Phase-Space Filling of Excitons. *J. Phys. Chem. C* **117,** 1974–1981 (2013).

63. Wang, F. *et al.* Observation of Excitons in One-Dimensional Metallic Single-Walled Carbon Nanotubes. *Phys. Rev. Lett.* **99,** (2007).

64. Rocha, J.-D. R., Bachilo, S. M., Ghosh, S., Arepalli, S. & Weisman, R. B. Efficient Spectrofluorimetric Analysis of Single-Walled Carbon Nanotube Samples. *Anal. Chem.* **83,** 7431–7437 (2011).

65. Tsyboulski, D. A., Rocha, J.-D. R., Bachilo, S. M., Cognet, L. & Weisman, R. B. Structure-Dependent Fluorescence Efficiencies of Individual Single-Walled Carbon Nanotubes. *Nano Lett.* **7,** 3080–3085 (2007).

66. Bending single-walled carbon nanotubes into nanorings using a Pickering emulsion-based process. Available at: http://www.sciencedirect.com/science/article/pii/S0008622311009699. (Accessed: 10th April 2016)
67. Naumov, A. V., Ghosh, S., Tsyboulski, D. A., Bachilo, S. M. & Weisman, R. B. Analyzing Absorption Backgrounds in Single-Walled Carbon Nanotube Spectra. *ACS Nano* **5,** 1639–1648 (2011).
68. Ohmori, S. *et al.* Fundamental Importance of Background Analysis in Precise Characterization of Single-Walled Carbon Nanotubes by Optical Absorption Spectroscopy. *J. Phys. Chem. C* **114,** 10077–10081 (2010).
69. Nair, N., Usrey, M. L., Kim, W.-J., Braatz, R. D. & Strano, M. S. Estimation of the (*n,m*) Concentration Distribution of Single-Walled Carbon Nanotubes from Photoabsorption Spectra. *Anal. Chem.* **78,** 7689–7696 (2006).
70. Hazeghi, A. Carbon Nanotube Electronics. (Stanford University, 2011).
71. Kane, C. L. & Mele, E. J. Electron Interactions and Scaling Relations for Optical Excitations in Carbon Nanotubes. *Phys. Rev. Lett.* **93,** (2004).
72. Liu, K. *et al.* Systematic determination of absolute absorption cross-section of individual carbon nanotubes. *Proc. Natl. Acad. Sci.* **111,** 7564–7569 (2014).
73. Islam, M., Milkie, D., Kane, C., Yodh, A. & Kikkawa, J. Direct Measurement of the Polarized Optical Absorption Cross Section of Single-Wall Carbon Nanotubes. *Phys. Rev. Lett.* **93,** (2004).
74. Vialla, F. *et al.* Chirality Dependence of the Absorption Cross Section of Carbon Nanotubes. *Phys. Rev. Lett.* **111,** 137402 (2013).

75. Oudjedi, L., Parra-Vasquez, A. N. G., Godin, A. G., Cognet, L. & Lounis, B. Metrological Investigation of the (6,5) Carbon Nanotube Absorption Cross Section. *J. Phys. Chem. Lett.* **4,** 1460–1464 (2013).

76. Streit, J. K., Bachilo, S. M., Ghosh, S., Lin, C.-W. & Weisman, R. B. Directly Measured Optical Absorption Cross Sections for Structure-Selected Single-Walled Carbon Nanotubes. *Nano Lett.* **14,** 1530–1536 (2014).

77. Malapanis, A., Perebeinos, V., Sinha, D. P., Comfort, E. & Lee, J. U. Quantum Efficiency and Capture Cross Section of First and Second Excitonic Transitions of Single-Walled Carbon Nanotubes Measured through Photoconductivity. *Nano Lett.* **13,** 3531–3538 (2013).

78. Li, N. *et al.* Maximizing the number of oxygen-containing functional groups on activated carbon by using ammonium persulfate and improving the temperature-programmed desorption characterization of carbon surface chemistry. *Carbon* **49,** 5002–5013 (2011).

79. Kundu, S., Wang, Y., Xia, W. & Muhler, M. Thermal Stability and Reducibility of Oxygen-Containing Functional Groups on Multiwalled Carbon Nanotube Surfaces: A Quantitative High-Resolution XPS and TPD/TPR Study. *J. Phys. Chem. C* **112,** 16869–16878 (2008).

80. Zacharia, R. Desorption of gases from graphitic and porous carbon surfaces. (Freie Universität Berlin, Freie Universität Berlin, Germany, 2004).

81. Likodimos, V. *et al.* Controlled surface functionalization of multiwall carbon nanotubes by HNO3 hydrothermal oxidation. *Carbon* **69,** 311–326 (2014).

82. Wang, H. & Hobbie, E. K. Amphiphobic carbon nanotubes as macroemulsion surfactants. *Langmuir* **19,** 3091–3093 (2003).

83. Fagan, J. A. *et al.* Isolation of Specific Small-Diameter Single-Wall Carbon Nanotube Species via Aqueous Two-Phase Extraction. *Adv. Mater.* **26,** 2800–2804 (2014).

84. Wang, R. K., Reeves, R. D. & Ziegler, K. J. Interfacial Trapping of Single-Walled Carbon Nanotube Bundles. *J. Am. Chem. Soc.* **129,** 15124–15125 (2007).

85. Wang, R. K. *et al.* Improving the effectiveness of interfacial trapping in removing single-walled carbon nanotube bundles. *J. Am. Chem. Soc.* **130,** 14721–14728 (2008).

86. Chen, W., Liu, X., Liu, Y. & Kim, H.-I. Novel synthesis of self-assembled CNT microcapsules by O/W Pickering emulsions. *Mater. Lett.* **64,** 2589–2592 (2010).

87. Lau, K. K. *et al.* Superhydrophobic carbon nanotube forests. *Nano Lett.* **3,** 1701–1705 (2003).

88. Welna, D. T., Qu, L., Taylor, B. E., Dai, L. & Durstock, M. F. Vertically aligned carbon nanotube electrodes for lithium-ion batteries. *J. Power Sources* **196,** 1455–1460 (2011).

89. Star, A., Joshi, V., Skarupo, S., Thomas, D. & Gabriel, J.-C. P. Gas sensor array based on metal-decorated carbon nanotubes. *J. Phys. Chem. B* **110,** 21014–21020 (2006).

90. Yang, J., Zhang, R., Xu, Y., He, P. & Fang, Y. Direct electrochemistry study of glucose oxidase on Pt nanoparticle-modified aligned carbon nanotubes electrode by

the assistance of chitosan–CdS and its biosensoring for glucose. *Electrochem. Commun.* **10,** 1889–1892 (2008).

91. Javey, A. *et al.* Carbon nanotube field-effect transistors with integrated ohmic contacts and high-κ gate dielectrics. *Nano Lett.* **4,** 447–450 (2004).
92. Jiang, K., Li, Q. & Fan, S. Nanotechnology: Spinning continuous carbon nanotube yarns. *Nature* **419,** 801–801 (2002).
93. Zhang, M., Atkinson, K. R. & Baughman, R. H. Multifunctional carbon nanotube yarns by downsizing an ancient technology. *Science* **306,** 1358–1361 (2004).
94. Cao, A. *et al.* Hydrogen storage of dense-aligned carbon nanotubes. *Chem. Phys. Lett.* **342,** 510–514 (2001).
95. Zhu, H. *et al.* Hydrogen adsorption in bundles of well-aligned carbon nanotubes at room temperature. *Appl. Surf. Sci.* **178,** 50–55 (2001).
96. Lin, W., Zhang, R., Moon, K.-S. & Wong, C. Synthesis of high-quality vertically aligned carbon nanotubes on bulk copper substrate for thermal management. *Adv. Packag. IEEE Trans. On* **33,** 370–376 (2010).
97. Cao, A., Dickrell, P. L., Sawyer, W. G., Ghasemi-Nejhad, M. N. & Ajayan, P. M. Super-compressible foamlike carbon nanotube films. *Science* **310,** 1307–1310 (2005).
98. Zhang, Q. *et al.* Energy-Absorbing Hybrid Composites Based on Alternate Carbon-Nanotube and Inorganic Layers. *Adv. Mater.* **21,** 2876–2880 (2009).
99. Esumi, K., Ishigami, M., Nakajima, A., Sawada, K. & Honda, H. Chemical treatment of carbon nanotubes. *Carbon* **34,** 279–281 (1996).

100. Bower, C., Kleinhammes, A., Wu, Y. & Zhou, O. Intercalation and partial exfoliation of single-walled carbon nanotubes by nitric acid. *Chem. Phys. Lett.* **288,** 481–486 (1998).

101. Shaffer, M. S., Fan, X. & Windle, A. Dispersion and packing of carbon nanotubes. *Carbon* **36,** 1603–1612 (1998).

102. Binks, B. & Lumsdon, S. Effects of oil type and aqueous phase composition on oil–water mixtures containing particles of intermediate hydrophobicity. *Phys. Chem. Chem. Phys.* **2,** 2959–2967 (2000).

103. Binks, B. P. & Whitby, C. P. Silica particle-stabilized emulsions of silicone oil and water: aspects of emulsification. *Langmuir* **20,** 1130–1137 (2004).

104. Stiller, S. *et al.* Investigation of the stability in emulsions stabilized with different surface modified titanium dioxides. *Colloids Surf. Physicochem. Eng. Asp.* **232,** 261–267 (2004).

105. Binks, B. & Lumsdon, S. Catastrophic phase inversion of water-in-oil emulsions stabilized by hydrophobic silica. *Langmuir* **16,** 2539–2547 (2000).

106. Binks, B. & Lumsdon, S. Transitional phase inversion of solid-stabilized emulsions using particle mixtures. *Langmuir* **16,** 3748–3756 (2000).

107. Yu, H. *et al.* Optimizing sonication parameters for dispersion of single-walled carbon nanotubes. *Chem. Phys.* **408,** 11–16 (2012).

108. Dong, L. & Johnson, D. T. Adsorption of acicular particles at liquid-fluid interfaces and the influence of the line tension. *Langmuir* **21,** 3838–3849 (2005).

109. He, J. *et al.* Drying droplets: A window into the behavior of nanorods at interfaces. *Small* **3,** 1214–1217 (2007).

110. Binks, B. P. & Lumsdon, S. O. Influence of Particle Wettability on the Type and Stability of Surfactant-Free Emulsions†. *Langmuir* **16,** 8622–8631 (2000).

111. Binks, B. P. & Rodrigues, J. A. Enhanced stabilization of emulsions due to surfactant-induced nanoparticle flocculation. *Langmuir* **23,** 7436–7439 (2007).

112. Binks, B. P., Rodrigues, J. A. & Frith, W. J. Synergistic interaction in emulsions stabilized by a mixture of silica nanoparticles and cationic surfactant. *Langmuir* **23,** 3626–3636 (2007).

113. Midmore, B. Preparation of a novel silica-stabilized oil/water emulsion. *Colloids Surf. Physicochem. Eng. Asp.* **132,** 257–265 (1998).

114. Binks, B. & Lumsdon, S. Stability of oil-in-water emulsions stabilised by silica particles. *Phys. Chem. Chem. Phys.* **1,** 3007–3016 (1999).

115. Madivala, B., Vandebril, S., Fransaer, J. & Vermant, J. Exploiting particle shape in solid stabilized emulsions. *Soft Matter* **5,** 1717–1727 (2009).

116. Briggs, N. M. *et al.* Multiwalled Carbon Nanotubes at the Interface of Pickering Emulsions. *Langmuir* **31,** 13077–13084 (2015).

117. Understanding Water-Based Bitumen Extraction from Athabasca Oil Sands - Masliyah - 2004 - The Canadian Journal of Chemical Engineering - Wiley Online Library. Available at: http://onlinelibrary.wiley.com/doi/10.1002/cjce.5450820403/full. (Accessed: 19th April 2016)

118. Wiese, S., Spiess, A. C. & Richtering, W. Microgel-Stabilized Smart Emulsions for Biocatalysis. *Angew. Chem.* **125,** 604–607 (2013).

119. Zhu, Y., Jiang, J., Liu, K., Cui, Z. & Binks, B. P. Switchable Pickering Emulsions Stabilized by Silica Nanoparticles Hydrophobized in Situ with a Conventional Cationic Surfactant. *Langmuir* **31,** 3301–3307 (2015).

120. Ngai, T., Behrens, S. H. & Auweter, H. Novel emulsions stabilized by pH and temperature sensitive microgels. *Chem. Commun.* 331–333 (2005). doi:10.1039/B412330A

121. Chen, Z. *et al.* Light Controlled Reversible Inversion of Nanophosphor-Stabilized Pickering Emulsions for Biphasic Enantioselective Biocatalysis. *J. Am. Chem. Soc.* **136,** 7498–7504 (2014).

122. Liu, Y. *et al.* Selective separation of flavones and sugars from honeysuckle by alcohol/salt aqueous two-phase system and optimization of extraction process. *Sep. Purif. Technol.* **118,** 776–783 (2013).

123. Aveyard, R., Binks, B. P. & Clint, J. H. Emulsions stabilised solely by colloidal particles. *Adv. Colloid Interface Sci.* **100–102,** 503–546 (2003).

124. Melander, M., Laasonen, K. & Jónsson, H. Effect of Magnetic States on the Reactivity of an FCC(111) Iron Surface. *J. Phys. Chem. C* **118,** 15863–15873 (2014).

125. Fan, H., Resasco, D. E. & Striolo, A. Amphiphilic Silica Nanoparticles at the Decane−Water Interface: Insights from Atomistic Simulations. *Langmuir* **27,** 5264–5274 (2011).

126. Hui, C. *et al.* Core-shell Fe3O4@SiO2 nanoparticles synthesized with well-dispersed hydrophilic Fe3O4 seeds. *Nanoscale* **3,** 701–705 (2011).

127. Igor, B., Yevhen, K. & Peter, S. TPR study of core-shell Fe@ Fe 3 O 4 nanoparticles supported on activated carbon and carbon nanotubes. *Adv. Mater. Phys. Chem.* **2012,** (2012).

128. Ramesh, S. *et al.* Dissolution of Pristine Single Walled Carbon Nanotubes in Superacids by Direct Protonation. *J. Phys. Chem. B* **108,** 8794–8798 (2004).

129. Hodge, S. A., Bayazit, M. K., Tay, H. H. & Shaffer, M. S. P. Giant cationic polyelectrolytes generated via electrochemical oxidation of single-walled carbon nanotubes. *Nat. Commun.* **4,** (2013).

130. Fogden, S., Howard, C. A., Heenan, R. K., Skipper, N. T. & Shaffer, M. S. P. Scalable Method for the Reductive Dissolution, Purification, and Separation of Single-Walled Carbon Nanotubes. *ACS Nano* **6,** 54–62 (2012).

131. Liu, H., Nishide, D., Tanaka, T. & Kataura, H. Large-scale single-chirality separation of single-wall carbon nanotubes by simple gel chromatography. *Nat. Commun.* **2,** 309 (2011).

132. Heller, D. A. *et al.* Concomitant Length and Diameter Separation of Single-Walled Carbon Nanotubes. *J. Am. Chem. Soc.* **126,** 14567–14573 (2004).

133. Silvera-Batista, C. A., Scott, D. C., McLeod, S. M. & Ziegler, K. J. A Mechanistic Study of the Selective Retention of SDS-Suspended Single-Wall Carbon Nanotubes on Agarose Gels. *J. Phys. Chem. C* **115,** 9361–9369 (2011).

134. Mace, C. R. *et al.* Aqueous Multiphase Systems of Polymers and Surfactants Provide Self-Assembling Step-Gradients in Density. *J. Am. Chem. Soc.* **134,** 9094–9097 (2012).

135. Chandler, D. Interfaces and the driving force of hydrophobic assembly. *Nature* **437,** 640–647 (2005).

136. Hirano, A., Tanaka, T., Urabe, Y. & Kataura, H. pH- and Solute-Dependent Adsorption of Single-Wall Carbon Nanotubes onto Hydrogels: Mechanistic Insights into the Metal/Semiconductor Separation. *ACS Nano* **7,** 10285–10295 (2013).

137. Hirano, A., Tanaka, T. & Kataura, H. Adsorbability of Single-Wall Carbon Nanotubes onto Agarose Gels Affects the Quality of the Metal/Semiconductor Separation. *J. Phys. Chem. C* **115,** 21723–21729 (2011).

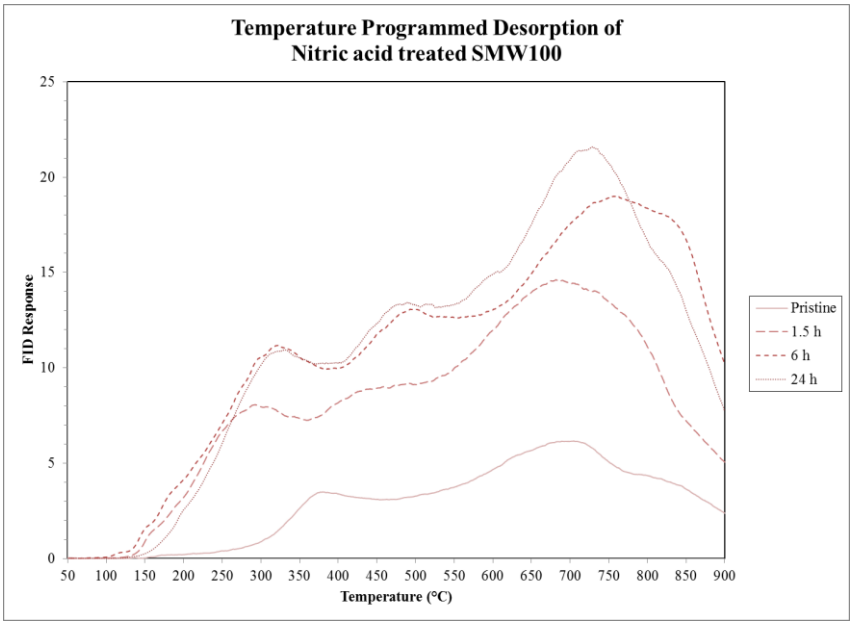
138. Tvrdy, K. *et al.* A Kinetic Model for the Deterministic Prediction of Gel-Based Single-Chirality Single-Walled Carbon Nanotube Separation. *ACS Nano* **7,** 1779–1789 (2013).

139. Hirano, A., Tanaka, T. & Kataura, H. Thermodynamic Determination of the Metal/Semiconductor Separation of Carbon Nanotubes Using Hydrogels. *ACS Nano* **6,** 10195–10205 (2012).

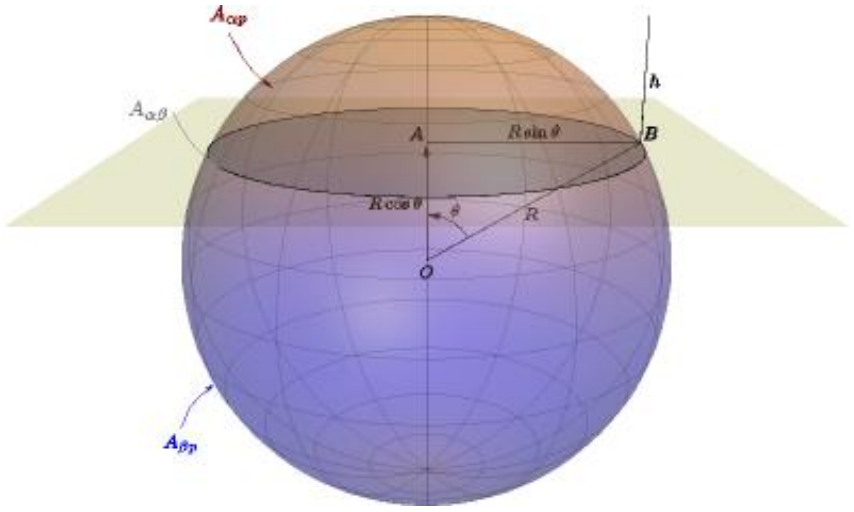
140. Hirano, A., Tanaka, T., Urabe, Y. & Kataura, H. Purification of Single-Wall Carbon Nanotubes by Controlling the Adsorbability onto Agarose Gels Using Deoxycholate. *J. Phys. Chem. C* **116,** 9816–9823 (2012).

141. Zavarzina, A. G. *et al.* Extraction of humic acids and their fractions in poly(ethylene glycol)-based aqueous biphasic systems. *Anal. Chim. Acta* **452,** 95–103 (2002).

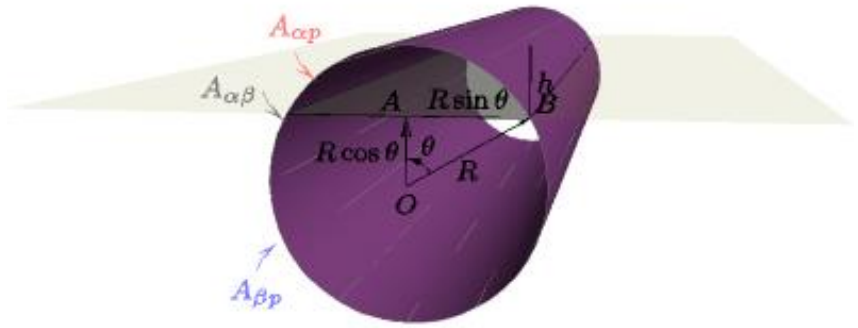
Supplementary Figures



Supplemental Figure 1: TPD data of all nitric acid treated SMW100 MWCNT.



Supplemental Figure 2: Illustration of sphere adsorbed at a liquid-liquid interface.



Supplemental Figure 3: Illustration of cylinder adsorbed at a liquid-liquid interface.

Appendices

Appendix A – Scripts

Data Retrieval Script (required import for TPD program)

```
import numpy as np

def retrieve_XY(file_path):
    # XY data is read in from a file in text format
    file_data = open(file_path).readlines()

    # The list of strings (lines in the file) is made into a list of lists while splitting by whitespace and removing commas
    file_data = map(lambda line: line.rstrip('\n').replace(',',' ').split(), file_data)

    # Remove empty lists, make into numpy array
    xy_array = np.array(filter(None, file_data))

    # Each line is searched to make sure that all items in the line are a number
    where_num = np.array(map(is_number, xy_array))

    # The data is searched for the longest contiguous chain of numbers
    contig = contiguous_regions(where_num)
    try:
        # Data lengths
        data_lengths = contig[:,1] - contig[:,0]
        # All maximums in contiguous data
        maxs = np.amax(data_lengths)
        longest_contig_idx = np.where(data_lengths == maxs)
    except ValueError:
        print 'Problem finding contiguous data'
        return np.array([])
    # Starting and stopping indices of the contiguous data are stored
    ss = contig[longest_contig_idx]

    # The file data with this longest contiguous chain of numbers
    # Float must be cast to each value in the lists of the contiguous data and cast to a numpy array
    longest_data_chains = np.array([[map(float, n) for n in xy_array[i[0]:i[1]]] for i in ss])

    # If there are multiple sets of data of the same length, they are added in columns
    column_stacked_data_chain = np.hstack(longest_data_chains)

    return column_stacked_data_chain

#http://stackoverflow.com/questions/4494404/find-large-number-of-consecutive-values-fulfilling-condition-in-a-numpy-array
def contiguous_regions(condition):
    """Finds contiguous True regions of the boolean array "condition". Returns
    a 2D array where the first column is the start index of the region and the
    second column is the end index."""

    # Find the indicies of changes in "condition"
    d = np.diff(condition)
    idx, = d.nonzero()

    # We need to start things after the change in "condition". Therefore,
    # we'll shift the index by 1 to the right.
    idx += 1
```

```
    if condition[0]:
        # If the start of condition is True prepend a 0
        idx = np.r_[0, idx]

    if condition[-1]:
        # If the end of condition is True, append the length of the array
        idx = np.r_[idx, condition.size] # Edit

    # Reshape the result into two columns
    idx.shape = (-1,2)
    return idx

def is_number(s):
    try:
        np.float64(s)
        return True
    except ValueError:
        return False
```

TPD Analysis Script

```
import sys
import os

from data_retrieval import *

from Tkinter import Tk
from tkFileDialog import askopenfilenames
from os.path import splitext, basename
import matplotlib.pyplot as plt
from PyQt4.QtGui import QMessageBox
import numpy as np
import scipy.integrate
import scipy.optimize as opt

def transform_TPD(data, hertz, temp_ramp, max_temp, t0, tf):
    end_time = len(data) / hertz / 60. # time in minutes
    dt = 1./hertz/60.
    time = np.arange(0., end_time, dt)
    # Temperature is calculated from the time array
    # and taking into account when the temperature controller is turned on
    temp = temp_ramp * time - t0
    # Room temperature assumed to be 22C
    RT = 22
    temp[temp<RT] = RT
    # isothermal at max temperature
    temp[temp>max_temp] = max_temp
    # Assume temps are the same
    furnace_temp = temp
    sample_temp = temp

    rawFID = data[:,0]
    bkg_subFID = data[:,1]

    my_bkg_FID = np.zeros(len(data))
    # All of the Data
    tTy = np.array( zip(time, sample_temp, rawFID , bkg_subFID, my_bkg_FID), dtype=[('time', float), ('sample_temp',
float),('raw_FID', float), ('machine_bkg_sub_FID', float), ('my_bkg_FID', float)])
    # Now we get the data when the experiemnt was run
    sample_data = np.array([])
    cond_between_times = np.logical_and(t0<=tTy['time'], tTy['time']<=tf)
    cond_after_t0 = t0<=tTy['time']
    if tf!=-1:
        sample_data = tTy[np.where(cond_between_times)]
        # Subtract linear background for each detector type
        y = np.copy(tTy['raw_FID'][cond_between_times])
        x = np.copy(tTy['sample_temp'][cond_between_times])

    else:
        sample_data = tTy[np.where(cond_after_t0)]
        # Subtract linear background for each detector type
        y = np.copy(tTy['raw_FID'][cond_after_t0])
        x = np.copy(tTy['sample_temp'][cond_after_t0])

    bkg_func = TPD_bkg(x, y)
    # Final Sample Data
    sample_data['my_bkg_FID'] = sample_data['raw_FID'] - bkg_func(x)

    return tTy, sample_data

def TPD_bkg(iter_x, iter_y):
    for i in range(10):
        z = np.polyfit(iter_x, iter_y, 1)
        p = np.poly1d(z)
        iter_bkg = p(iter_x)
        diff = (iter_y-iter_bkg)
        mask = np.where(diff<0.0)
        iter_y = iter_y[mask]
        iter_x = iter_x[mask]
    return p

def TPD_fit(temp, FID):
    temp_vary = 75
    #amp_vary = None
    width_vary = 100.
    default_amp = np.max(FID)
    default_width = 100.
    # Weak acidic carboxylic acid
    WA_temp = 300 #C
    WA_temp_min = WA_temp - temp_vary/2.
    WA_temp_max = WA_temp + temp_vary/2.
    WA_amp = default_amp
    WA_amp_min = 0.
    WA_amp_max = None
    WA_width = default_width
    WA_width_min = WA_width - width_vary/2.
    WA_width_max = WA_width + width_vary/2.
    # Strong acidic carboxylic acid
```

```
    SA_temp = 400.
    SA_temp_min = SA_temp - temp_vary/2.
    SA_temp_max = SA_temp + temp_vary/2.
    SA_amp = default_amp
    SA_amp_min = 0.
    SA_amp_max = None
    SA_width = default_width
    SA_width_min = SA_width - width_vary/2.
    SA_width_max = SA_width + width_vary/2.
    # Carboxylic anhydride
    CAn_temp = 500.
    CAn_temp_min = CAn_temp - temp_vary/2.
    CAn_temp_max = CAn_temp + temp_vary/2.
    CAn_amp = default_amp
    CAn_amp_min = 0.
    CAn_amp_max = None
    CAn_width = default_width
    CAn_width_min = CAn_width - width_vary/2.
    CAn_width_max = CAn_width + width_vary/2.
    # Lactone
    LC_temp = 650.
    LC_temp_min = LC_temp - temp_vary/2.
    LC_temp_max = LC_temp + temp_vary/2.
    LC_amp = default_amp
    LC_amp_min = 0.
    LC_amp_max = None
    LC_width = default_width
    LC_width_min = LC_width - width_vary/2.
    LC_width_max = LC_width + width_vary/2.
    # Phenol
    PH_temp = 700.
    PH_temp_min = PH_temp - temp_vary/2.
    PH_temp_max = PH_temp + temp_vary/2.
    PH_amp = default_amp
    PH_amp_min = 0.
    PH_amp_max = None
    PH_width = default_width
    PH_width_min = PH_width - width_vary/2.
    PH_width_max = PH_width + width_vary/2.
    # Carbonyl Quinone
    CQ_temp = 900.
    CQ_temp_min = CQ_temp - temp_vary/2.
    CQ_temp_max = CQ_temp + temp_vary/2.
    CQ_amp = default_amp
    CQ_amp_min = 0.
    CQ_amp_max = None
    CQ_width = default_width
    CQ_width_min = CQ_width - width_vary/2.
    CQ_width_max = CQ_width + width_vary/2.

    temps = np.array([WA_temp, SA_temp, CAn_temp, LC_temp, PH_temp, CQ_temp])
    widths = np.array([WA_amp, SA_amp, CAn_amp, LC_amp, PH_amp, CQ_amp])
    amps =  np.array([WA_width, SA_width, CAn_width, LC_width, PH_width, CQ_width])
    params0 = np.ravel(np.column_stack((temps, amps, widths)))
    params_bounds = np.array([(WA_temp_min, WA_temp_max), (WA_amp_min, WA_amp_max), (WA_width_min, WA_width_max),\
                              (SA_temp_min, SA_temp_max), (SA_amp_min, SA_amp_max), (SA_width_min, SA_width_max),\
                              (CAn_temp_min, CAn_temp_max), (CAn_amp_min, CAn_amp_max), (CAn_width_min, CAn_width_max),\
                              (LC_temp_min, LC_temp_max), (LC_amp_min, LC_amp_max), (LC_width_min, LC_width_max),\
                              (PH_temp_min, PH_temp_max), (PH_amp_min, PH_amp_max), (PH_width_min, PH_width_max),\
                              (CQ_temp_min, CQ_temp_max), (CQ_amp_min, CQ_amp_max), (CQ_width_min, CQ_width_max)])
    params, f, d = opt.fmin_l_bfgs_b(func = residual, x0 = params0, bounds = params_bounds, args = (temp, FID), approx_grad =
True)
    models = TPD_models(temp, FID, params)
    return models

def residual(params, *args):
    temp_array = args[0]
    FID = args[1]
    model = TPD_models(temp_array, FID, params)['model']
    return sum((model - FID)**2)

def TPD_models(temp_array, FID, params):
    WA_temp = params[0]
    WA_amp = params[1]
    WA_width = params[2]
    # Strong acidic carboxylic acid
    SA_temp = params[3]
    SA_amp = params[4]
    SA_width = params[5]
    # Carboxylic anhydride
    CAn_temp = params[6]
    CAn_amp = params[7]
    CAn_width = params[8]
    # Lactone
    LC_temp = params[9]
    LC_amp = params[10]
    LC_width = params[11]
    # Phenol
    PH_temp = params[12]
    PH_amp = params[13]
    PH_width = params[14]
```

```
    # Carbonyl Quinone
    CQ_temp = params[15]
    CQ_amp = params[16]
    CQ_width = params[17]

    WA = gaussian(temp_array, WA_temp, WA_amp, WA_width)
    SA = gaussian(temp_array, SA_temp, SA_amp, SA_width)
    CAn = gaussian(temp_array, CAn_temp, CAn_amp, CAn_width)
    LC = gaussian(temp_array, LC_temp, LC_amp, LC_width)
    PH = gaussian(temp_array, PH_temp, PH_amp, PH_width)
    CQ = gaussian(temp_array, CQ_temp, CQ_amp, CQ_width)
    model = WA + SA + CAn + LC + PH + CQ
    models = np.array(zip(model, WA, SA, CAn, LC, PH, CQ) , dtype=[('model', float), ('Weak Carboxylic', float), ('Strong
Carboxylic', float), ('Carboxylic Anhydride', float), ('Lactone', float), ('Phenol', float), ('Quinone', float)])
    return models


def gaussian(x, center, amp, width):
    if width==0: width=0.00000001
    return amp * np.e**(-((x-center)**2./(2.*width**2.)))

def desorbed_dict(time_array, FID_array, mol_sample, cal):
    area = scipy.integrate.simps(FID_array, time_array)
    mol_desorbed = area*cal
    mol_percent_desorbed = mol_desorbed/mol_sample*100.

    return {'area':area,'mol desorbed':mol_desorbed,'mol% desorbed':mol_percent_desorbed}

def TPD_properties(sample_data, mol_sample, cal):
    prop_dict = {}
    raw = desorbed_dict(sample_data['time'], sample_data['raw_FID'], mol_sample, cal)
    my_bkg = desorbed_dict(sample_data['time'], sample_data['my_bkg_FID'], mol_sample, cal)
    machine_bkg = desorbed_dict(sample_data['time'], sample_data['machine_bkg_sub_FID'], mol_sample, cal)
    prop_dict['mol sample'] = mol_sample
    prop_dict['raw'] = raw
    prop_dict['my_bkg'] = my_bkg
    prop_dict['machine_bkg'] = machine_bkg

    return prop_dict

def plot_TPD(tTy, sample_data):
    fig = plt.figure()
    ax = fig.add_subplot(131)
    ax2 = fig.add_subplot(132)
    ax3 = fig.add_subplot(133)
    ax.plot(tTy['time'], tTy['raw_FID'])
    ax2.plot(sample_data['sample_temp'], sample_data['raw_FID'])
    ax3.plot(tTy['time'], tTy['sample_temp'])
    plt.show()

def get_data():
    # Get and manipulate data
    data = {}
    Tk().withdraw()
    Tkfilepaths = askopenfilenames()
    filepaths = Tk().tk.splitlist(Tkfilepaths)

    for filepath in filepaths:
        file_name = splitext(basename(filepath))[0]
        xy = retrieve_XY(filepath)
        OU_bool = False
        if "<TYPE>=CHROM" in open(filepath).readline():
            OU_bool = True
        if len(xy)>0:
            data[file_name] = xy, OU_bool
        else:
            QMessageBox.about(self, 'Error', 'File at\n' + str(filepath) + '\ncould not be parsed to get the xy data.\nTry
making a tab delimited text file with two columns of x and y data.')
    return data

if __name__== "__main__":
    print "This program will determie the % of carbon desorbed from a sample of pure carbon (carbon nanotubes) in a temperature
programmed desorption experiment."

    # Use this calibration from CO2 pulses
    # 1.24 to 1.3 *10^-11
    cal = 1.3*10**-11 # mol C / area

    main_fig = plt.figure()
    main_ax = main_fig.add_subplot(111)

    print "Please choose the '.asc' files given from the TPD/TPO machine"
    file_data = get_data()
    for name, (xy, OU_bool) in file_data.iteritems():
        hertz = float(input("Input data rate in Hertz.\n(OU's instrument typically gathers data at 2 Hertz):"))
        temp_ramp = float(input("Input temperature ramp in K/min.\n(OU's instrument typically ramps at 10 K/min):"))#10. # K/min
        max_temp = float(input("Input max temperature in Celsius.\n(OU's instrument typically ends at 900 C):"))#900. # C
        t0 = float(input("Input time in which the experiment started and the furnace was turned on (minutes).\n(It is easiest to
save separate files when waiting for the baseline to stabalize, then to get the sample data starting at time 0):"))#0 # min
        tf_bool = input("Input 0 if you want the ending time to be when the final temperature is reached,\nor 1 if you would
like the final time to be the last data point gathered:")
```

```
        if tf_bool == 0:
            tf = max_temp/temp_ramp
        elif tf_bool == 1:
            tf = -1 # end of data
        sample_mass = float(input("Input carbon sample mass in grams:"))#0.050 # grams
        mol_sample = sample_mass / 12.

        tTy, sample_data = transform_TPD(xy, hertz, temp_ramp, max_temp, t0, tf)




        print
        print str(name), ":"
        for k, v in TPD_properties(sample_data, mol_sample, cal).iteritems():
            try:
                print str(k), " is: ", str(round(v['mol% desorbed'],2))
            except:
                print str(k), " is: ", str(round(v,2))


        models = TPD_fit(sample_data['sample_temp'], sample_data['raw_FID'])

        for name in models.dtype.names:
            d = desorbed_dict(sample_data['time'], models[name], mol_sample, cal)
            print str(name), " is: ", str(round(d['mol% desorbed'],2))




        main_ax.plot(sample_data['sample_temp'], sample_data['raw_FID'], 'r-', \
                     sample_data['sample_temp'], models['model'], 'g-',\
                     sample_data['sample_temp'], models['Weak Carboxylic'], 'r--',\
                     sample_data['sample_temp'], models['Strong Carboxylic'], 'r--',\
                     sample_data['sample_temp'], models['Carboxylic Anhydride'], 'r--',\
                     sample_data['sample_temp'], models['Lactone'], 'r--',\
                     sample_data['sample_temp'], models['Phenol'], 'r--',\
                     sample_data['sample_temp'], models['Quinone'], 'r--')

    plt.legend(file_data.keys(), loc='upper left')
    plt.show()
```

Figure 4 Python Matplotlib Script

```
# Import required libraries
import numpy as np
from pylab import *
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d, Axes3D

# Decalin-water surface tension
# Rose, William E. "The interfacial tension of several hydrocarbons against water." Retrospective Theses and Dissertations,
1919-2007 (1949).
gam_ow = 0.0000000001#0.0515#0.0000000001#0.0515 # N/m
# Boltzmann Constant
k_b = 1.3806488*10**-23 # m2 kg s-2 K-1
Temp = 298 # K
# Total Volume
V_t = 0.001 # meters cubed (0.001 m^3 = 1 L)

# Number of points in mesh grid in one direction
num_steps = 100


# Drop Radius initial and final to graph
Ri = 1 * 10**-9 # meters (1*10^-9 m = 1 nm)
Rf = 7000 * 10**-9 # meters (20,000 *10^-9 m = 20 um)

# Maximum and minimum droplet number
n_max = 1*10**9
n_min = 1

# Drop Radius array
R_arr = np.linspace(Ri, Rf, num_steps) # meters
# number for droplets array
n_arr = np.linspace(n_min, n_max, num_steps)

# Single droplet volume
def V_drop(r):
    return (4./3.) * np.pi * r ** 3

# Volume Fraction of dispersed phase
def phi_d(n, r):
    # Volume of dispersed phase
    V_d = n * V_drop(r)
    phi_d_raw = V_d / V_t
    problem_loc = np.where(V_d>=V_t)[0]
    phi_d_raw[problem_loc] = 1
    return phi_d_raw

# Function for calculating entropy of conformation
def S_conf(n, r):
    S_conf = Temp * n * k_b * (np.log(phi_d(n,r)) + ((1-phi_d(n,r))/phi_d(n,r)) * np.log(1-phi_d(n,r)))
    return S_conf

# Surface area change from adding droplets
def delta_area(n, r):
    return n * 4 * np.pi * r**2

# Function for interfacial energy
def E_int(n, r):
    return delta_area(n, r) * gam_ow

def create_plot(fig_name):
    # Reshape Contact Angle and Radius into meshgrid for 3D plotting
    n, r = np.meshgrid(n_arr, R_arr)

    # Calculate Energies with functions
    S_arr = S_conf(n, r)
    E_int_arr = E_int(n, r)
    G_form = E_int_arr + S_arr
    G_form_neg = np.where(G_form<0, G_form, np.nan)

    # Plotting functions
    fig = plt.figure()

    ax = fig.add_subplot(1, 1, 1, projection='3d') # num rows, num columns, subplot number, 3d projection
    p_S = ax.plot_wireframe(n, r*10**9, S_arr, color="Red")
    p_E = ax.plot_wireframe(n, r*10**9, E_int_arr, color="Blue")
    p_G = ax.plot_wireframe(n, r*10**9, G_form, color="Green")
    #p_Gn = ax.plot_wireframe(n, r*10**9, G_form_neg, color="Gray")
    axes = fig.gca(projection='3d')
    xLabel = axes.set_xlabel("Number of Droplets)")
    yLabel = axes.set_ylabel("Droplet Radius (nm)")
    zLabel = axes.set_zlabel(r"$\Delta G_{form}$")

    constants_label =
r'$\gamma_{o/w}=$'+str(gam_ow)+r'$\frac{N}{m}$'+'\n'+'$T=$'+str(Temp)+'K\n'+r'$V_{Tot}=$'+str(V_t)+r'$m^3$'

    axes.text(0,0, np.max(G_form), constants_label, zdir=None, style='italic', bbox={'facecolor':'white', 'alpha':0.5,
'pad':10})
```

```
    p1 = Rectangle((0, 0), 1, 1, fc="r")
    p2 = Rectangle((0, 0), 1, 1, fc="b")
    p3 = Rectangle((0, 0), 1, 1, fc="g")
    ax.legend([p1,p2,p3], ["Conformational Entropy", "Interfacial Energy", "Gibbs Free Energy of Formation"])
    ax.view_init(elev=20, azim=-45)
    plt.tight_layout()

    plt.savefig(str(fig_name) + '.eps', format='eps', dpi=1000)
    plt.savefig(str(fig_name) + '.jpg', format='jpg', dpi=1000)

    plt.show()
create_plot('G_form')
```

Python Optical Absorbance Fitting

```
## Spectro
## =======
## This program analyzes UV-VIS absorption spectra from aqueous surfactant suspended dispersions of carbon nanotubes.
## It does so by fitting absorption profile models from the literature with a linear regression at each step of a non-linear
regression fitting of the background (amorphous carbon and pi plasmon resonances) model.
##
## This program is written using python 2.7 with a Qt4 Gui and has the following dependencies:
##
## pythonxy:
## http://www.mirrorservice.org/sites/pythonxy.com/Python(x,y)-2.7.6.1.exe
##
## lmfit:
## https://pypi.python.org/packages/2.7/l/lmfit/lmfit-0.7.4.win32-py2.7.exe
##
## This program was written by Chase Brown and is under the GNU General Public License.  (A Copyleft License)
## This program is free to use for any research, as long as this this thesis and github page are cited:
## https://github.com/chaxor/Spectro
## Brown, Chase. “CARBON NANOTUBES AND MAGNETIC PARTICLES AS PICKERING EMULSION STABALIZERS:
## PARTICLE CONTROL FOR PHASE SELECTIVE REACTIONS” Master’s thesis, University of Oklahoma, 2016.

from PyQt4 import QtCore, QtGui
import sys
from math import factorial
import matplotlib
matplotlib.use('Qt4Agg')
from matplotlib.backends.backend_qt4agg import FigureCanvasQTAgg as FigureCanvas
import numpy as np
from scipy import *
import scipy
import lmfit
import multiprocessing as mp
import time
import datetime
import xlsxwriter
import operator

##
## CONSTANTS
##

# A value of 1 for APP_SCREEN_RATIO will open the program in a 'maximized' type of position.
APP_SCREEN_RATIO = 0.7
# Due to the Global Interepreter Lock (GIL) in python, we have to switch the main thread from updating the GUI front end to
doing calculations in the back end
UPDATE_TIME_IN_MILLISECONDS = 300

# E=hc/lambda , and hc=1240 eV*nm
WAVELENGTH_TO_ENERGY_CONVERSION = 1240.0 # eV*nm

# Using Dresselhaus' convention for carbon nanotubes (CNTs), (n,m), we can create all CNTs within two (n,m) values
lowestNM = 5
highestNM = 10

## Pi Plasmon ##
# Starting pi plasmon amplitude as a ratio of the absorbance at the peak center
AMP_RATIO_PI = 0.6
# Starting peak center for the pi plasmon
PI_PLASMON_CENTER = 5.6
# Allowable variance in pi plasmon center
PI_PLASMON_CENTER_VAR = 0.6
PI_PLASMON_CENTER_MAX = (PI_PLASMON_CENTER + PI_PLASMON_CENTER_VAR)
PI_PLASMON_CENTER_MIN = (PI_PLASMON_CENTER - PI_PLASMON_CENTER_VAR)
# Full width at half maximum for pi plasmon
PI_PLASMON_FWHM = 0.6
PI_PLASMON_FWHM_MAX = 5.0
PI_PLASMON_FWHM_MIN = 0.1

## Graphite Lorentzian ##
# Starting graphite amplitude as a ratio of the absorbance at the peak center
```

```
AMP_RATIO_GRAPHITE = 0.5
# Starting peak center for the graphite
GRAPHITE_CENTER = 4.1
# Allowable variance in graphite
GRAPHITE_CENTER_VAR = 0.4
GRAPHITE_CENTER_MAX = (GRAPHITE_CENTER + GRAPHITE_CENTER_VAR)
GRAPHITE_CENTER_MIN = (GRAPHITE_CENTER - GRAPHITE_CENTER_VAR)
# Full width at half maximum for graphite
GRAPHITE_FWHM = 0.6
GRAPHITE_FWHM_MAX = 5.0
GRAPHITE_FWHM_MIN = 0.5

# Extinction coefficients for types of amorphous carbon
# Source: "Analyzing Absorption Backgrounds in Single-Walled Carbon Nanotube Spectra" (Anton V. Naumov, Saunab Ghosh, Dmitri A.
Tsyboulski, Sergei M. Bachilo, and R. Bruce Weisman)
alpha_N134 = 0.155 # L/mg
alpha_aCB = 0.082 # L/mg
b_N134 = 0.0030 # nm^-1
b_aCB = 0.00155 # nm^-1

# Metallic Background coefficients
alpha_METAL = 0.048 # L/mg
b_METAL = 0.00155 # nm^-1

# Energy Transitions given by Reference 2: "An atlas of carbon nanotube optical transitions"
# Equation used for energy transitions can be found in the supplementary information
# Below is the list of anisotropy prefactors and fermi velocity renormalization constants used in the equation

# Supporting Info:
beta = -0.620 # eV * nm
alpha = [1.532, 1.474, 1.504, 1.556, 1.560, 1.576, 1.588, 1.593, 1.596, 1.608] # eV * nm
eta   = [0.148, 0.097, 0.068, 0.058, 0.058, 0.061, 0.050, 0.052, 0.058, 0.058] # eV * nm^2
gamma = [-0.056,-0.009,-0.002,0.014, 0.016, 0.009, 0.000, 0.000, 0.011, 0.004] # eV * nm^2
# Main Paper:
beta_mainpaper = -0.173 # eV * nm^2
eta_mainpaper = [0.142, 0.097, 0.068, 0.058, 0.058, 0.058, 0.047, 0.052, 0.047, 0.054] # eV * nm^2
vF_mainpaper =  [1.229, 1.152, 1.176, 1.221, 1.226, 1.236, 1.241, 1.244, 1.248, 1.256] # 10^6 m s^-1


## Useful functions for getting data from arbitrary data files
## Since most data files contain contiguous sets of data in columns, these functions extract the data and disregard the other
information
def retrieve_XY(file_path):
          # XY data is read in from a file in text format
          file_data = open(file_path).readlines()

          # The list of strings (lines in the file) is made into a list of lists while splitting by whitespace and removing
commas
          file_data = map(lambda line: line.rstrip('\n').replace(',',' ').split(), file_data)

          # Remove empty lists, make into numpy array
          xy_array = np.array(filter(None, file_data))

          # Each line is searched to make sure that all items in the line are a number
          where_num = np.array(map(is_number, xy_array))

          # The data is searched for the longest contiguous chain of numbers
          contig = contiguous_regions(where_num)
          try:
                    # Data lengths
                    data_lengths = contig[:,1] - contig[:,0]
                    # All maximums in contiguous data
                    maxs = np.amax(data_lengths)
                    longest_contig_idx = np.where(data_lengths == maxs)
          except ValueError:
                    print 'Problem finding contiguous data'
                    return np.array([])
          # Starting and stopping indices of the contiguous data are stored
          ss = contig[longest_contig_idx]

          # The file data with this longest contiguous chain of numbers
          # Float must be cast to each value in the lists of the contiguous data and cast to a numpy array
          longest_data_chains = np.array([[map(float, n) for n in xy_array[i[0]:i[1]]] for i in ss])

          # If there are multiple sets of data of the same length, they are added in columns
          column_stacked_data_chain = np.hstack(longest_data_chains)

          return column_stacked_data_chain

#http://stackoverflow.com/questions/4494404/find-large-number-of-consecutive-values-fulfilling-condition-in-a-numpy-array
def contiguous_regions(condition):
          """Finds contiguous True regions of the boolean array "condition". Returns
          a 2D array where the first column is the start index of the region and the
          second column is the end index."""

          # Find the indicies of changes in "condition"
          d = np.diff(condition)
          idx, = d.nonzero()

          # We need to start things after the change in "condition". Therefore,
          # we'll shift the index by 1 to the right.
```

```
            idx += 1

            if condition[0]:
                        # If the start of condition is True prepend a 0
                        idx = np.r_[0, idx]

            if condition[-1]:
                        # If the end of condition is True, append the length of the array
                        idx = np.r_[idx, condition.size] # Edit

            # Reshape the result into two columns
            idx.shape = (-1,2)
            return idx

def is_number(s):
            try:
                        np.float64(s)
                        return True
            except ValueError:
                        return False


## Useful functions for spectral processing

heaviside = lambda x: 0.5 * (np.sign(x) + 1)

# https://gist.github.com/RyanHope/2321077
def savitzky_golay(y, window_size, order, deriv=0, rate=1):
            r"""Smooth (and optionally differentiate) data with a Savitzky-Golay filter.
            The Savitzky-Golay filter removes high frequency noise from data.
            It has the advantage of preserving the original shape and
            features of the signal better than other types of filtering
            approaches, such as moving averages techniques.
            Parameters
            ----------
            y : array_like, shape (N,)
                        the values of the time history of the signal.
            window_size : int
                        the length of the window. Must be an odd integer number.
            order : int
                        the order of the polynomial used in the filtering.
                        Must be less then `window_size` - 1.
            deriv: int
                        the order of the derivative to compute (default = 0 means only smoothing)
            Returns
            -------
            ys : ndarray, shape (N)
                        the smoothed signal (or its n-th derivative).
            Notes
            -----
            The Savitzky-Golay is a type of low-pass filter, particularly
            suited for smoothing noisy data. The main idea behind this
            approach is to make for each point a least-square fit with a
            polynomial of high order over a odd-sized window centered at
            the point.
            Examples
            --------
            t = np.linspace(-4, 4, 500)
            y = np.exp( -t**2 ) + np.random.normal(0, 0.05, t.shape)
            ysg = savitzky_golay(y, window_size=31, order=4)
            import matplotlib.pyplot as plt
            plt.plot(t, y, label='Noisy signal')
            plt.plot(t, np.exp(-t**2), 'k', lw=1.5, label='Original signal')
            plt.plot(t, ysg, 'r', label='Filtered signal')
            plt.legend()
            plt.show()
            References
            ----------
            .. [1] A. Savitzky, M. J. E. Golay, Smoothing and Differentiation of
               Data by Simplified Least Squares Procedures. Analytical
               Chemistry, 1964, 36 (8), pp 1627-1639.
            .. [2] Numerical Recipes 3rd Edition: The Art of Scientific Computing
               W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery
               Cambridge University Press ISBN-13: 9780521880688
            """

            try:
                        window_size = np.abs(np.int(window_size))
                        order = np.abs(np.int(order))
            except ValueError, msg:
                        raise ValueError("window_size and order have to be of type int")
            if window_size % 2 != 1 or window_size < 1:
                        raise TypeError("window_size size must be a positive odd number")
            if window_size < order + 2:
                        raise TypeError("window_size is too small for the polynomials order")
            order_range = range(order+1)
            half_window = (window_size -1) // 2
            # precompute coefficients
            b = np.mat([[k**i for i in order_range] for k in range(-half_window, half_window+1)])
            m = np.linalg.pinv(b).A[deriv] * rate**deriv * factorial(deriv)
            # pad the signal at the extremes with
```

```
            # values taken from the signal itself
            firstvals = y[0] - np.abs( y[1:half_window+1][::-1] - y[0] )
            lastvals = y[-1] + np.abs(y[-half_window-1:-1][::-1] - y[-1])
            y = np.concatenate((firstvals, y, lastvals))
            return np.convolve( m[::-1], y, mode='valid')


# Optical transition from i (valence) to j (conduction)
# Metallics may have a splitting between high and low energies (trigonal warping effect)
class Transition:
            def __init__(self, swcnt, i, j=None, k = 0):
                        self.swcnt = swcnt
                        self.i = i
                        self.k = k
                        if j is not None:
                                    self.j = j
                        else:
                                    self.j = i

                        # Solvents affect optical transitions (redshifting or blueshifting) due to differences in dielectric
constants and refractive indexes, etc.
                        # Silvera-Batista, Carlos A., Randy K. Wang, Philip Weinberg, and Kirk J. Ziegler.
                        # "Solvatochromic shifts of single-walled carbon nanotubes in nonpolar microenvironments."
                        # Physical Chemistry Chemical Physics 12, no. 26 (2010): 6990-6998.
                        self.epsilon = self.swcnt.spectra.epsilon
                        self.eta = self.swcnt.spectra.eta
                        self.Dswcntsolvent = self.swcnt.spectra.Dswcntsolvent

                        # Calculate the properties of the transition and store them
                        self.p = self.p()
                        self.k_p = self.k_p()
                        self.theta_p = self.theta_p()
                        self.Eii = self.Eii()
                        self.fs = self.fs()
                        self.FWHM = self.FWHM()
                        self.a = self.a()
                        self.b = self.b()
                        self.delta = self.delta()

                        self.shape = 1.0
                        self.amp = self.fs
                        if self.swcnt.spectra.X is not None:
                                    E = self.swcnt.spectra.X
                        else:
                                    E = np.array([0])

                        # Absorption profile from:
                        # Liu, Kaihui, Xiaoping Hong, Sangkook Choi, Chenhao Jin, Rodrigo B. Capaz, Jihoon Kim, Shaul Aloni et al.
                        # "Systematic Determination of Absolute Absorption Cross-section of Individual Carbon Nanotubes."
                        # arXiv preprint arXiv:1311.3328 (2013).
                        self.line = self.fs/np.pi * \
                                                self.FWHM/((E-self.Eii)**2+self.FWHM**2) + \
                                                self.fs/(self.a*np.pi) * \
                                                np.convolve( (self.b*self.FWHM)/(E**2+(self.b*self.FWHM)**2), \
                                                                                    (heaviside(E-(self.Eii +
self.delta))/np.sqrt(abs(E-(self.Eii + self.delta))))
                                                                                    , mode='same')



            # Energy Transition Functions from Reference 2:
            # Optical transition index 'p'
            def p(self):
                        if(self.swcnt.mod_type==0.): p = 3.*self.i
                        elif(self.swcnt.mod_type==1. or self.swcnt.mod_type==2.):
                                    if(self.i%2.==0.): p = self.i+(self.i/2.)-1. # If i is even
                                    elif(self.i%2.==1.): p = self.i+int(self.i/2.) # If i is odd
                        else: print("Error in electronic type")
                        p = int(p)
                        return p

            # Length of polar coordinates vector from the K point (Reference 2):
            def k_p(self):
                        k_p = 2.*self.p/(3.*self.swcnt.dt)
                        return k_p

            # Angle of wave vector around K point in radians
            def theta_p(self):
                        theta_p = []
                        if(self.swcnt.mod_type==0.):
                                    if(self.k==0):
                                                # lower energy sub band
                                                theta_p = self.swcnt.theta + np.pi
                                    if(self.k==1):
                                                # higher energy sub band
                                                theta_p = self.swcnt.theta
                        elif(self.swcnt.mod_type==1.):
                                    theta_p = self.swcnt.theta + self.i*np.pi
                        elif(self.swcnt.mod_type==2.):
                                    theta_p = self.swcnt.theta + (self.i+1.)*np.pi
                        return theta_p
```

```
          # Energy Optical Transitions from Valence Band (i) to Conduction Band (i) given by Reference 2: "An atlas of carbon
nanotube optical transitions"
          # Equation used for energy transitions can be found in the supplementary information of Reference 2
          def Eii_vacc(self):
                    theta_p = self.theta_p
                    k_p = self.k_p
                    p = self.p

                    # Supporting info algorithm
                    Eii = alpha[p-1]*k_p + beta*k_p*np.log10(1.5*k_p) + (k_p**2.)*(eta[p-1]+gamma[p-
1]*np.cos(theta_p*3.))*np.cos(theta_p * 3.)
                    return Eii

          def Eii(self):
                    Eii_vacc = self.Eii_vacc()
                    # Silvera-Batista, Carlos A., Randy K. Wang, Philip Weinberg, and Kirk J. Ziegler.
                    # "Solvatochromic shifts of single-walled carbon nanotubes in nonpolar microenvironments."
                    # Physical Chemistry Chemical Physics 12, no. 26 (2010): 6990-6998.
                    Eii_shift = -self.Dswcntsolvent*self.Onsager()/(Eii_vacc**3.*self.swcnt.dt**5.)
                    Eii = Eii_vacc + Eii_shift
                    return Eii

          def Onsager(self):
                    fe = 2.*(self.epsilon-1.)/(2*self.epsilon+1)
                    fn = 2.*(self.eta**2-1.)/(2*self.eta**2+1)
                    return fe - fn


          # Optical osciallator strength per atom for semiconducting tubes
          # Liu, Kaihui, Xiaoping Hong, Sangkook Choi, Chenhao Jin, Rodrigo B. Capaz, Jihoon Kim, Shaul Aloni et al.
          # "Systematic Determination of Absolute Absorption Cross-section of Individual Carbon Nanotubes."
          # arXiv preprint arXiv:1311.3328 (2013).
          def fs(self):
                    return 45.9/((self.p + 7.5)*self.swcnt.dt)

          # Full width at half maximum
          def FWHM(self):
                    if(self.swcnt.electronic_type!=0):
                              FWHM = 0.0194 * self.Eii
                    else:
                              FWHM = 0.0214 * self.Eii
                    return FWHM

          def a(self):
                    if(self.swcnt.electronic_type!=0):
                              a = 4.673 - 0.747 * self.swcnt.dt
                    else:
                              a = 3.065 - 0.257 * self.swcnt.dt#0.976 + 0.186 * self.swcnt.dt
                    return a

          def b(self):
                    if(self.swcnt.electronic_type!=0):
                              b = 0.97 + 0.256 * self.swcnt.dt
                    else:
                              b = 0.976 + 0.186 * self.swcnt.dt#3.065 - 0.257 * self.swcnt.dt
                    return b

          def delta(self):
                    if(self.swcnt.electronic_type!=0):
                              delta = 0.273 - 0.041 * self.swcnt.dt
                    else:
                              delta = 0.175 - 0.0147 * self.swcnt.dt
                    return delta

          # Returns a string which gives information about the transition
          def transition_string(self):
                    NM_Eii_string = "SWCNT" + self.swcnt.NM() + '_E' + str(self.i) + str(self.j)
                    if(self.swcnt.electronic_type!=0):
                              transition_string = NM_Eii_string
                    if(self.swcnt.electronic_type==0):
                              if(self.k==0):
                                        transition_string = NM_Eii_string + "_low"
                              if(self.k==1):
                                        transition_string = NM_Eii_string + "_high"
                    else:
                              transition_string = NM_Eii_string
                    return transition_string

# Single-walled carbon nanotube object
# each SWCNT is defined by its characteristic (n,m) chirality
class SWCNT:
          def __init__(self, n, m, spectra):
                    n = float(n)
                    m = float(m)
                    self.n = n
                    self.m = m
                    if spectra is not None:
                              self.spectra = spectra
                    else:
                              self.spectra = Spectra('default', np.zeros(0), np.zeros(0))
```

```
                    # Electronic type test:
                    # 0 - metallic
                    # 1 - semiconducting type 1 (S1)
                    # 2 - semiconducting type 2 (S2)
                    self.electronic_type = np.mod(2.*n + m,3.)

                    # Alternative nomenclature:
                    # 0 - metallic
                    # 1 - mod type 1 <=> semiconducting type 2 (MOD1 <=> S2)
                    # 2 - mod type 2 <=> semiconducting type 1 (MOD2 <=> S1)
                    self.mod_type = np.mod(n - m,3.)

                    # Basic Nanotube Properties from Reference 1:
                    # Chiral vector length in angstroms (1.421 is the C-C distance in angstroms)
                    self.Ch = np.sqrt(3.0)*1.421*np.sqrt(n**2. + m**2. + n*m)
                    # CNT diameter in angstroms  (/10. --> nanometers)
                    self.dt = self.Ch/np.pi/10.
                    # Chiral angle in radians
                    self.theta = np.arctan(np.sqrt(3.)*m/(m + 2.*n))
                    # Consider S11, S22, S33, S44, S55, S66, S77 and M11 M22, M33, with high and low tranisitions for metals
                    upper_ij_metal = 3
                    upper_ij_sc = 7
                    if(self.electronic_type==0):
                              if(self.n==self.m):
                                        self.allowed_transitions = [Transition(self, i) for i in range(1,upper_ij_metal+1)]
                              else:
                                        self.allowed_transitions = [Transition(self, i, i, k) for i in
range(1,upper_ij_metal+1) for k in range(0,2)]
                    else:
                              self.allowed_transitions = [Transition(self, i) for i in range(1,upper_ij_sc+1)]

          # A model line for each nanotube is the sum of each of the transitions that can occur for the nanotube
          @property
          def line(self):
                    arrays = np.array([transition.line for transition in self.allowed_transitions])
                    return np.sum(arrays, axis=0)

          # Other useful functions for debugging and documentation
          def print_electronic_type(self):
                    if(self.mod_type==0): return "Metallic"
                    elif(self.mod_type==1 or self.mod_type==2): return "Semiconducting"
                    else: return "Error in n,m indices"

          # Returns the Dresselhaus nomenclature "(n,m)" for each nanotube
          def strNM(self):
                    string_tube = "(" + str(np.int(self.n)) + "," +  str(np.int(self.m)) + ")"
                    return string_tube

          # For paramters, we cannot store "(,)" symbols
          # So this is just a string of "NM" - such as "66" for "(6,6)"
          def NM(self):
                    NM = str(np.int(self.n)).rstrip(".") + str(np.int(self.m)).rstrip(".")
                    return NM

# Psuedo Voigts can create lorentz and Guassian functions or a convolution of both
# It is useful for background creation
def pseudoVoigt(x, amp, center, width, shapeFactor):
          LorentzPortion = (width**2./((x-center)**2.+width**2.))
          GaussianPortion = np.e**(-((x-center)**2./(2.*width**2.)))
          try:
                    Voigt = amp*(shapeFactor*LorentzPortion+(1.-shapeFactor)*GaussianPortion)
          except ZeroDivisionError:
                    width += 0.001
                    pseudoVoigt(x, amp, center, width, shapeFactor)
          return Voigt

# Function to create all of the tubes and store them in a list
def initialize_SWCNTs(lowestNM, highestNM, spectra):
          SWCNTs=[]
          # Create a list of all tube species we are interested in
          for n in range(0,highestNM+1):
                    for m in range(0,highestNM+1):
                              if(n<lowestNM and m<lowestNM): break
                              elif(n<m): break
                              else: SWCNTs.append(SWCNT(n, m, spectra))
          return SWCNTs

# Spectra object which holds the X and Y data and the sample name
class Spectra(QtCore.QObject):

          # Signal/slot setup uses signals to tell the program when to update the GUI
          update_signal = QtCore.pyqtSignal(QtCore.QObject)
          done_signal = QtCore.pyqtSignal()

          def __init__(self, spectra_name, X, Y):
                    QtCore.QObject.__init__(self)
                    self.spectra_name = spectra_name
                    self.X = X
                    # The given spectra is smoothed using the savitsky golay smoothing algorithm
                    self.Y = savitzky_golay(y=Y,window_size=5, order=2)
                    self.model = Y*0
```

```
                    self.background_model = Y*0
                    self.model_without_background = Y*0
                    self.step = 0

                    # Solvents affect optical transitions (redshifting or blueshifting) due to differences in dielectric
constants and refractive indexes, etc.
                    # Silvera-Batista, Carlos A., Randy K. Wang, Philip Weinberg, and Kirk J. Ziegler.
                    # "Solvatochromic shifts of single-walled carbon nanotubes in nonpolar microenvironments."
                    # Physical Chemistry Chemical Physics 12, no. 26 (2010): 6990-6998.
                    self.epsilon = 2.3
                    self.eta = 1.33
                    self.Dswcntsolvent = 0.09 # eV^4*nm^5

                    # All the single-walled carbon nanotubes to be used in the deconvolution process
                    # The list will be an array of SWCNTs from (n,m)=(lowestNM, 0) to (n,m)=(highestNM,highestNM)
                    self.SWCNT_list = initialize_SWCNTs(lowestNM, highestNM, self)
                    self.transition_list = [transition for swcnt in self.SWCNT_list for transition in
swcnt.allowed_transitions] #if(self.in_spectrum(transition)==True)]
                    # First, create our SWCNT profile matrix
                    self.SWCNT_matrix = np.matrix(np.column_stack([swcnt.line for swcnt in self.SWCNT_list]))
                    self.swcnts_soln = np.ones(self.SWCNT_matrix.shape[1], dtype=np.float64)

                    self.params = lmfit.Parameters()

                    self.species_percentage_dictionary = {}
                    self.species_percentage_error_dictionary = {}
                    self.metallic_percentage = 0.0
                    self.mean_diameter = 0.0
                    graphite_amp = AMP_RATIO_GRAPHITE * np.interp(GRAPHITE_CENTER, self.X, self.Y)
                    PP_amp = AMP_RATIO_PI * np.interp(PI_PLASMON_CENTER, self.X, self.Y)

                    self.bkg_soln = np.array([0.0, 0.0, 0.0, 0.0, \
                                                                                graphite_amp, GRAPHITE_CENTER,
GRAPHITE_FWHM, \
                                                                                PP_amp, PI_PLASMON_CENTER,
PI_PLASMON_FWHM])
                    self.bkg_soln_bounds = np.array([(0.0,None), (0.0, None), (0.0, None), (0.0, None), \
                                                                                                        (0.0, None),
(GRAPHITE_CENTER_MIN, GRAPHITE_CENTER_MAX), (GRAPHITE_FWHM_MIN, GRAPHITE_FWHM_MAX), \
                                                                                                        (0.0, None),
(PI_PLASMON_CENTER_MIN, PI_PLASMON_CENTER_MAX), (PI_PLASMON_FWHM_MIN, PI_PLASMON_FWHM_MAX)])


                    self.sample_params = np.array([self.epsilon, self.eta, self.Dswcntsolvent])
                    self.sample_params_bounds = np.array([(1.,5.),(1.,5.),(0.0,0.1)])

                    self.timer = QtCore.QTimer()
                    self.timer.timeout.connect(self.send_update_GUI)
                    self.timer.start(UPDATE_TIME_IN_MILLISECONDS)

          def calc_species_norm_amps_dictionary(self):
                    species_norm_amp_dict = {}
                    for i, swcnt in enumerate(self.SWCNT_list):
                              species_norm_amp_dict[swcnt] = self.swcnts_soln[i]/swcnt.allowed_transitions[0].fs
                    return species_norm_amp_dict

          def calc_species_norm_amps_error_dictionary(self):
                    species_norm_amp_error_dict = {}
                    for swcnt in self.SWCNT_list:
                              for transition in swcnt.allowed_transitions:
                                        try:
                                                  Eii_amp_error_value = self.params[transition.transition_string() +
'_amp'].stderr
                                        except KeyError:
                                                  Eii_amp_error_value = -1.0
                                        try:
                                                  species_norm_amp_error_dict[swcnt] = Eii_amp_error_value/transition.fs
                                        except TypeError:
                                                  species_norm_amp_error_dict[swcnt] = -1.0
                                        break
                    return species_norm_amp_error_dict

          def calc_species_percentage_dictionary(self):
                    species_percentage_dict = {}
                    species_norm_dict = self.calc_species_norm_amps_dictionary()
                    # First get the sum of all of the amplitudes, while normalizing them using the optical oscillator strength
                    sum_Eiis_norm_by_fs = sum(species_norm_dict.values())
                    for swcnt in self.SWCNT_list:
                              try:
                                        species_percentage_dict[swcnt] = 100.*species_norm_dict[swcnt] / sum_Eiis_norm_by_fs
                              except (ZeroDivisionError, KeyError):
                                        species_percentage_dict[swcnt] = 0.0
                    return species_percentage_dict

          def calc_species_percentage_error_dictionary(self):
                    species_percentage_error_dict = {}
                    species_norm_error_dict = self.calc_species_norm_amps_error_dictionary()
                    # First get the sum of all of the amplitudes, while normalizing them using the optical oscillator strength
                    sum_Eiis_norm_by_fs = sum(species_norm_error_dict.values())
                    for swcnt in self.SWCNT_list:
                              try:
```

```
                                        species_percentage_error_dict[swcnt] = 100.*species_norm_error_dict[swcnt] /
sum_Eiis_norm_by_fs
                              except (ZeroDivisionError, KeyError, TypeError):
                                        species_percentage_error_dict[swcnt] = -1.0
                    return species_percentage_error_dict

          def calc_metallic_percentage(self):
                    metallic_percentage = 0.0
                    for swcnt in self.SWCNT_list:
                              if(swcnt.electronic_type==0.0):
                                        metallic_percentage += self.species_percentage_dictionary[swcnt]
                    return metallic_percentage

          def calc_mean_diameter(self):
                    mean_diameter = 0.0
                    for swcnt in self.SWCNT_list:
                              mean_diameter += self.species_percentage_dictionary[swcnt]/100. * swcnt.dt
                    return mean_diameter


          @QtCore.pyqtSlot()
          def deconvolute(self):
                    self.state = 0
                    x, f, d = scipy.optimize.fmin_l_bfgs_b(func = self.residual, x0=self.bkg_soln,
bounds=self.bkg_soln_bounds, approx_grad = True, factr = 10)

                    self.done_signal.emit()

          def residual(self, bkg_params):
                    self.step += 1
                    # Initialize
                    residual_array = np.zeros(len(self.X))
                    temp_background_model = np.zeros(len(self.X))
                    temp_model = np.zeros(len(self.X))
                    temp_model_without_background = np.zeros(len(self.X))

                    # Calculate the background first and add SWCNT voigt profiles on later
                    self.bkg_soln = bkg_params

                    aCBConc = bkg_params[0]
                    N134Conc = bkg_params[1]
                    aCBy0 = bkg_params[2]
                    N134y0 = bkg_params[3]
                    GLamp = bkg_params[4]
                    GLcenter = bkg_params[5]
                    GLFWHM = bkg_params[6]
                    PPamp = bkg_params[7]
                    PPcenter = bkg_params[8]
                    PPFWHM = bkg_params[9]

                    aCB = aCBConc * alpha_aCB * (aCBy0 + np.exp(-b_aCB * (WAVELENGTH_TO_ENERGY_CONVERSION/self.X)))
                    N134 = N134Conc * alpha_N134 * (N134y0 + np.exp(-b_N134 * (WAVELENGTH_TO_ENERGY_CONVERSION/self.X)))
                    GL = pseudoVoigt(self.X, GLamp, GLcenter, GLFWHM, 1)
                    PP = pseudoVoigt(self.X, PPamp, PPcenter, PPFWHM, 1)

                    temp_background_model = aCB + N134 + GL + PP

                    # In the first state, solve the fitting for just the background parameters and nanotube amplitudes
simultaneously
                    if(self.state==0):
                              # bkg_sub, if the background is fit properly, will contain just the absorption profile from van
hove singularity transitions from nanotubes
                              bkg_sub = self.Y - temp_background_model

                              # Solve the system with swcnts:
                              self.swcnts_soln, residual = scipy.optimize.nnls(self.SWCNT_matrix, bkg_sub)

                              # Change the amplitudes for each SWCNT in the SWCNT object
                              for i, swcnt in enumerate(self.SWCNT_list):
                                        swcnt.line = self.swcnts_soln[i] * np.array(self.SWCNT_matrix[:,i])

                              temp_model_without_background = np.inner(self.SWCNT_matrix, self.swcnts_soln)

                    temp_model = temp_model_without_background + temp_background_model

                    # The background model should not exceed the given spectrum at any point
                    # Therefore, if it does, apply a large residual to that point
                    for x_index in range(0, len(self.X)):
                              if(temp_background_model[x_index] > self.Y[x_index]):
                                        residual_array[x_index] = -999.*(temp_model[x_index] - self.Y[x_index])
                              else:
                                        residual_array[x_index] = temp_model[x_index] - self.Y[x_index]

                    # Update and store all of the values in the Spectra object
                    self.model_without_background = temp_model_without_background
                    self.background_model = temp_background_model
                    self.model = temp_model
                    self.species_percentage_dictionary = self.calc_species_percentage_dictionary()
                    self.species_percentage_error_dictionary = self.calc_species_percentage_error_dictionary()
                    self.metallic_percentage = self.calc_metallic_percentage()
                    self.mean_diameter = self.calc_mean_diameter()
```

```
                        return np.sum(residual_array**2)

            def send_update_GUI(self):
                        self.update_signal.emit(self)
                        return

# Each tab which holds a spectra gets its own object
class Spectra_Tab(QtGui.QTabWidget):
            start_comp = QtCore.pyqtSignal()
            kill_thread = QtCore.pyqtSignal()
            def __init__(self, parent, temp_spectra):
                        self.parent = parent
                        QtGui.QTabWidget.__init__(self, parent)
                        self.temp_spectra = temp_spectra
                        self.top_layer_grid = QtGui.QGridLayout(self)

                        self.canvas_frame = QtGui.QFrame(self)
                        self.canvas_frame.setFrameShape(QtGui.QFrame.StyledPanel)
                        self.results_frame = QtGui.QFrame(self)
                        self.results_frame.setFrameShape(QtGui.QFrame.StyledPanel)

                        self.top_layer_grid.addWidget(self.canvas_frame)
                        self.top_layer_grid.addWidget(self.results_frame)

                        self.canvas_grid = QtGui.QGridLayout(self.canvas_frame)

                        self.top_left_frame = QtGui.QFrame(self.canvas_frame)
                        self.top_left_frame.setFrameShape(QtGui.QFrame.StyledPanel)
                        self.canvas_grid.addWidget(self.top_left_frame)

                        self.bottom_canvas_frame = QtGui.QFrame(self.canvas_frame)
                        self.bottom_canvas_frame.setFrameShape(QtGui.QFrame.StyledPanel)
                        self.canvas_grid.addWidget(self.bottom_canvas_frame)

                        vertical_splitter = QtGui.QSplitter(QtCore.Qt.Vertical)
                        vertical_splitter.addWidget(self.top_left_frame)
                        vertical_splitter.addWidget(self.bottom_canvas_frame)
                        self.canvas_grid.addWidget(vertical_splitter)

                        self.results_grid = QtGui.QGridLayout(self.results_frame)
                        self.treeWidget = QtGui.QTreeWidget(self.results_frame)
                        self.treeWidget.setFocusPolicy(QtCore.Qt.WheelFocus)
                        self.treeWidget.setAutoFillBackground(True)
                        self.treeWidget.setAlternatingRowColors(True)
                        self.treeWidget.setSelectionMode(QtGui.QAbstractItemView.SingleSelection)
                        self.treeWidget.setSelectionBehavior(QtGui.QAbstractItemView.SelectRows)
                        self.treeWidget.setHorizontalScrollMode(QtGui.QAbstractItemView.ScrollPerItem)
                        self.treeWidget.setAutoExpandDelay(-1)
                        self.treeWidget.setHeaderLabels(["(n,m)/Property","%, [value]"])
                        self.other_properties = QtGui.QTreeWidgetItem(self.treeWidget, ["Properties"])
                        self.nm_species = QtGui.QTreeWidgetItem(self.treeWidget, ["(n,m)"])
                        self.semiconducting = QtGui.QTreeWidgetItem(self.other_properties, ["Semiconducting %"])
                        self.metallic = QtGui.QTreeWidgetItem(self.other_properties, ["Metallic %"])
                        self.avg_diameter = QtGui.QTreeWidgetItem(self.other_properties, ["Average Diameter"])
                        self.step_in_tree = QtGui.QTreeWidgetItem(self.other_properties, ["Iteration #"])
                        self.dict_of_nm_tree = {}
                        for swcnt in temp_spectra.SWCNT_list:
                                    self.dict_of_nm_tree[swcnt] = QtGui.QTreeWidgetItem(self.nm_species, [swcnt.strNM()])
                        self.results_grid.addWidget(self.treeWidget)

                        graph_results_splitter = QtGui.QSplitter(QtCore.Qt.Horizontal)
                        graph_results_splitter.addWidget(self.canvas_frame)
                        graph_results_splitter.addWidget(self.results_frame)
                        self.top_layer_grid.addWidget(graph_results_splitter)

                        policy = QtGui.QSizePolicy(QtGui.QSizePolicy.Preferred,QtGui.QSizePolicy.Preferred)
                        policy.setHorizontalStretch(8)
                        self.canvas_frame.setSizePolicy(policy)

                        # Make figure for original line, background line, and total fit line
                        self.top_left_fig = matplotlib.figure.Figure()
                        self.top_left_plot = self.top_left_fig.add_subplot(111)
                        self.top_left_plot.set_ylabel('Absorbance [a.u.]')
                        self.top_left_plot.set_xlabel('Photon Energy [eV]')
                        self.top_left_plot.set_title('Total Absorbance Fit')
                        init_values = np.zeros(len(self.temp_spectra.X))
                        self.top_left_line, = self.top_left_plot.plot(self.temp_spectra.X, self.temp_spectra.Y, 'r-')
                        self.top_left_background_line, self.top_left_total_fit_line, =
self.top_left_plot.plot(self.temp_spectra.X, init_values, 'k-', self.temp_spectra.X, init_values, 'b-', animated=True)
                        self.top_left_canvas = FigureCanvas(self.top_left_fig)
                        plotLayout = QtGui.QVBoxLayout()
                        plotLayout.addWidget(self.top_left_canvas)
                        self.top_left_frame.setLayout(plotLayout)
                        self.top_left_canvas.show()
                        self.top_left_canvas.draw()
                        self.top_left_canvas_BBox = self.top_left_plot.figure.canvas.copy_from_bbox(self.top_left_plot.bbox)
                        self.ax1 = self.top_left_plot.figure.axes[0]
                        self.ax1.set_xlim(self.temp_spectra.X.min(), self.temp_spectra.X.max())
                        self.ax1.set_ylim(0, self.temp_spectra.Y.max() + .05*self.temp_spectra.Y.max())
                        self.top_left_plot_old_size = self.top_left_plot.bbox.width, self.top_left_plot.bbox.height
```

```
                    # Make bottom figure
                    self.bottom_fig = matplotlib.figure.Figure()
                    self.bottom_plot = self.bottom_fig.add_subplot(111)
                    self.bottom_plot.set_ylabel('Absorbance [a.u.]')
                    self.bottom_plot.set_xlabel('Photon Energy [eV]')
                    self.bottom_plot.set_title('Background Subtracted Fit')
                    self.bottom_line_original_without_background, = self.bottom_plot.plot(self.temp_spectra.X,
self.temp_spectra.Y, 'r-', linewidth=3, animated=True)
                    self.bottom_line, = self.bottom_plot.plot(self.temp_spectra.X, init_values, 'b-', linewidth=3,
animated=True)
                    self.swcnt_line_dict = {}
                    for swcnt in temp_spectra.SWCNT_list:
                              self.swcnt_line_dict[swcnt], = self.bottom_plot.plot(self.temp_spectra.X, swcnt.line,
animated=True)
                    self.bottom_canvas = FigureCanvas(self.bottom_fig)
                    bottomplotLayout = QtGui.QVBoxLayout()
                    bottomplotLayout.addWidget(self.bottom_canvas)
                    self.bottom_canvas_frame.setLayout(bottomplotLayout)
                    self.bottom_canvas.show()
                    self.bottom_canvas.draw()
                    self.bottom_canvas_BBox = self.bottom_plot.figure.canvas.copy_from_bbox(self.bottom_plot.bbox)
                    self.bottom_ax1 = self.bottom_plot.figure.axes[0]
                    self.bottom_ax1.set_xlim(self.temp_spectra.X.min(), self.temp_spectra.X.max())
                    self.bottom_ax1.set_ylim(0, self.temp_spectra.Y.max() + .05*self.temp_spectra.Y.max())
                    self.bottom_plot_old_size = self.bottom_plot.bbox.width, self.bottom_plot.bbox.height

                    # Make Thread associated with the tab
                    thread = QtCore.QThread(parent=self)
                    self.worker = self.temp_spectra
                    self.worker.moveToThread(thread)
                    self.worker.update_signal.connect(self.update_GUI)
                    self.worker.done_signal.connect(self.closeEvent)
                    self.start_comp.connect(self.worker.deconvolute)
                    self.kill_thread.connect(thread.quit)
                    thread.start()

          @QtCore.pyqtSlot(Spectra)
          def update_GUI(self, tmp_spectra):
                    # change the GUI to reflect changes made to Spectra
                    # Get the first background of the plots to blits lines to
                    if(tmp_spectra.step==1):
                              self.top_left_canvas_BBox =
self.top_left_plot.figure.canvas.copy_from_bbox(self.top_left_plot.bbox)
                              self.bottom_canvas_BBox = self.bottom_plot.figure.canvas.copy_from_bbox(self.bottom_plot.bbox)
                    # If the size of the box changes, get that background instead
                    top_left_plot_current_size = self.top_left_plot.bbox.width, self.top_left_plot.bbox.height
                    bottom_plot_current_size = self.bottom_plot.bbox.width, self.bottom_plot.bbox.height
                    if( self.top_left_plot_old_size != top_left_plot_current_size or self.bottom_plot_old_size !=
bottom_plot_current_size):
                              self.top_left_plot_old_size = top_left_plot_current_size
                              self.top_left_plot.clear()
                              self.top_left_canvas.draw()
                              self.top_left_canvas_BBox =
self.top_left_plot.figure.canvas.copy_from_bbox(self.top_left_plot.bbox)
                              self.top_left_plot.set_ylabel('Absorbance [a.u.]')
                              self.top_left_plot.set_xlabel('Photon Energy [eV]')
                              self.top_left_plot.set_title('Total Absorbance Fit')
                              self.bottom_plot_old_size = bottom_plot_current_size
                              self.bottom_plot.clear()
                              self.bottom_canvas.draw()
                              self.bottom_canvas_BBox = self.bottom_plot.figure.canvas.copy_from_bbox(self.bottom_plot.bbox)
                              self.bottom_plot.set_ylabel('Absorbance [a.u.]')
                              self.bottom_plot.set_xlabel('Photon Energy [eV]')
                              self.bottom_plot.set_title('Background Subtracted Fit')

                    # Write to the Top Left Plot with original data, background data, and total fit
                    self.top_left_background_line.set_ydata(tmp_spectra.background_model)
                    self.top_left_total_fit_line.set_ydata(tmp_spectra.model)
                    self.top_left_plot.figure.canvas.restore_region(self.top_left_canvas_BBox)
                    if( tmp_spectra.background_model.max() > tmp_spectra.Y.max()):
                              self.ax1.set_ylim(0, 1.05*tmp_spectra.background_model.max())
                    elif(tmp_spectra.model.max() > tmp_spectra.Y.max()):
                              self.ax1.set_ylim(0, 1.05*tmp_spectra.model.max())
                    else:
                              self.ax1.set_ylim(0, 1.05*tmp_spectra.Y.max())
                    self.top_left_plot.draw_artist(self.top_left_line)
                    self.top_left_plot.draw_artist(self.top_left_background_line)
                    self.top_left_plot.draw_artist(self.top_left_total_fit_line)
                    self.top_left_plot.figure.canvas.blit(self.top_left_plot.bbox)

                    # Write to the Bottom Plot with each nanotube peak
                    self.bottom_line_original_without_background.set_ydata(tmp_spectra.Y-tmp_spectra.background_model)
                    self.bottom_line.set_ydata(tmp_spectra.model_without_background)
                    try:
                              for swcnt in self.dict_of_nm_tree:
                                        self.swcnt_line_dict[swcnt].set_ydata(swcnt.line)
                                        self.swcnt_line_dict[swcnt].set_linewidth(1)
                              current_swcnt = None
                              for swcnt in self.dict_of_nm_tree:
                                        if(self.dict_of_nm_tree[swcnt] == self.treeWidget.currentItem()):
                                                  current_swcnt = swcnt
```

```
                                                  break
                                        self.swcnt_line_dict[current_swcnt].set_linewidth(4)
                              except KeyError:
                                        pass
                              self.bottom_plot.figure.canvas.restore_region(self.bottom_canvas_BBox)
                              if( np.amax(tmp_spectra.Y-tmp_spectra.background_model) > np.amax(tmp_spectra.model_without_background) ):
                                        self.bottom_ax1.set_ylim(0, 1.05*np.amax(tmp_spectra.Y-tmp_spectra.background_model))
                              if( np.amax(tmp_spectra.model_without_background) < 0.05):
                                        self.bottom_ax1.set_ylim(0, 0.05)
                              else:
                                        self.bottom_ax1.set_ylim(0, 1.05*np.amax(tmp_spectra.model_without_background))
                              self.bottom_plot.draw_artist(self.bottom_line_original_without_background)
                              self.bottom_plot.draw_artist(self.bottom_line)
                              for swcnt in tmp_spectra.SWCNT_list:
                                        self.bottom_plot.draw_artist(self.swcnt_line_dict[swcnt])
                              self.bottom_plot.figure.canvas.blit(self.bottom_plot.bbox)

                              # Show percentage of species on the side bar
                              try:
                                        percent_dict = tmp_spectra.species_percentage_dictionary
                                        percent_error_dict = tmp_spectra.species_percentage_error_dictionary
                                        for swcnt in tmp_spectra.SWCNT_list:
                                                  self.dict_of_nm_tree[swcnt].setText(1, str(round(percent_dict[swcnt], 0)).rstrip('0')
+ ' % +-' + str(round(percent_error_dict[swcnt], 1)))
                                        self.semiconducting.setText(1, str(round(100.-tmp_spectra.metallic_percentage, 0)).rstrip('0') +
' %')
                                        self.metallic.setText(1, str(round(tmp_spectra.metallic_percentage, 0)).rstrip('0') + ' %')
                                        self.avg_diameter.setText(1, str(round(tmp_spectra.mean_diameter,2)) + ' nm')
                                        self.step_in_tree.setText(1, str(tmp_spectra.step))
                              except KeyError:
                                        pass


          def output_results(self):
                              print "Making Excel Workbook..."

                              date_time = datetime.datetime.now().strftime("%Y-%m-%d(%H-%M-%S)")
                              name = str(self.temp_spectra.spectra_name)
                              book = xlsxwriter.Workbook(name + ' -- ' + date_time +'_OA_Results.xlsx')
                              OA_sheet_name = "Optical Absorption Data"
                              Results_sheet_name = "Results"
                              Other_params_name = "Other Parameters"
                              OA_sheet = book.add_worksheet(OA_sheet_name)
                              Results_sheet = book.add_worksheet(Results_sheet_name)
                              Other_params_sheet = book.add_worksheet(Other_params_name)

                              # Write x, y data for main and all species
                              OA_sheet.write('A1', "Energy (eV)")
                              OA_sheet.write_column('A2', self.temp_spectra.X)
                              OA_sheet.write('B1', name )
                              OA_sheet.write_column('B2', self.temp_spectra.Y)
                              OA_sheet.write('C1', "Model")
                              OA_sheet.write_column('C2', self.temp_spectra.model)
                              OA_sheet.write('D1', "Background")
                              OA_sheet.write_column('D2', self.temp_spectra.background_model)
                              for i, swcnt in enumerate(self.temp_spectra.SWCNT_list):
                                        OA_sheet.write(0, 3+i, swcnt.strNM())
                                        OA_sheet.write_column(1, 3+i, swcnt.line)

                              Results_sheet.write('A1', "(n,m)")
                              Results_sheet.write('B1', "%")
                              quant_dict = self.temp_spectra.species_percentage_dictionary.iteritems()
                              for i, (swcnt, amount) in enumerate(sorted(quant_dict)):
                                        Results_sheet.write(i+1, 0, swcnt.strNM())
                                        Results_sheet.write(i+1, 1, round(amount,1))
                              Results_sheet.write('D1', "Semiconducting %")
                              Results_sheet.write('D2', "Metallic %")
                              Results_sheet.write('E1', round(100-self.temp_spectra.calc_metallic_percentage(),1))
                              Results_sheet.write('E2', round(self.temp_spectra.calc_metallic_percentage(),1))


                              Other_params_sheet.write("A1", "(n,m)")
                              Other_params_sheet.write_column("A2", [swcnt.strNM() for swcnt in self.temp_spectra.SWCNT_list])
                              Other_params_sheet.write("B1", "SWCNT Solution Vector")
                              Other_params_sheet.write_column("B2", self.temp_spectra.swcnts_soln)

                              book.close()
                              print "Excel Workbook Made."



          def start_computation(self):
                              self.start_comp.emit()
                              return

          def closeEvent(self):
                              print 'done with processing'
                              self.kill_thread.emit()

class MainWindow(QtGui.QMainWindow):
          def __init__(self, parent = None):
```

```
                    self.spectra_list = []
                    self.tab_list = []
                    QtGui.QMainWindow.__init__(self)
                    self.setWindowTitle("Spectro")
                    screen_height = app.desktop().screenGeometry().height()
                    screen_width = app.desktop().screenGeometry().width()
                    self.resize(int(screen_width*APP_SCREEN_RATIO), int(screen_height*APP_SCREEN_RATIO))

                    self.setTabShape(QtGui.QTabWidget.Rounded)
                    self.centralwidget = QtGui.QWidget(self)
                    self.top_level_layout = QtGui.QGridLayout(self.centralwidget)

                    self.tabWidget = QtGui.QTabWidget(self.centralwidget)
                    self.top_level_layout.addWidget(self.tabWidget, 1, 0, 25, 25)

                    open_spectra_button = QtGui.QPushButton("Open Spectra")
                    self.top_level_layout.addWidget(open_spectra_button, 0, 0)
                    QtCore.QObject.connect(open_spectra_button, QtCore.SIGNAL("clicked()"), self.open_spectra)

                    process_spectra_button = QtGui.QPushButton("Process Spectra")
                    self.top_level_layout.addWidget(process_spectra_button, 0, 1)
                    QtCore.QObject.connect(process_spectra_button, QtCore.SIGNAL("clicked()"), self.process_spectra)

                    save_results_button = QtGui.QPushButton("Save Results")
                    self.top_level_layout.addWidget(save_results_button, 0, 2)
                    QtCore.QObject.connect(save_results_button, QtCore.SIGNAL("clicked()"), self.output_results)

                    self.setCentralWidget(self.centralwidget)
                    self.centralwidget.setLayout(self.top_level_layout)

          def open_spectra(self):
                    fileNameList = QtGui.QFileDialog.getOpenFileNames(caption="Select Files for Processing")
                    for file_name in fileNameList:
                              # file_name is form ~ "C:/Users/you/someData.asc", so split it after the last "/" and before the
"." in ".asc" or ".xls"
                              spectra_name = file_name.split('\\')[-1].split('.')[0]
                              xy = retrieve_XY(file_name)
                              X = WAVELENGTH_TO_ENERGY_CONVERSION/xy[:,0]
                              Y = xy[:,1]
                              self.spectra_list.append(Spectra(spectra_name, X, Y))
                              self.tab_list.append(Spectra_Tab(self.tabWidget, self.spectra_list[-1]))
                              self.tabWidget.addTab(self.tab_list[-1], spectra_name)
                    return

          def process_spectra(self):
                    for tab in self.tab_list:
                              tab.start_computation()
                    return

          def output_results(self):
                    for tab in self.tab_list:
                              tab.output_results()
                    return

if __name__ == "__main__":
          app = QtGui.QApplication([])
          win = MainWindow()
          win.show()
          sys.exit(app.exec_())
```