Non-Linear Inverse Liquid-Solid Chromatography as a Methodology to Characterize Drug Concentration Losses to Polymeric Materials used in Body-on-a-Chip Devices for Drug Discovery

Mark Schnepper
University of Central Florida

Part of the Materials Science and Engineering Commons

Find similar works at: https://stars.library.ucf.edu/etd2020

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2020- by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation
Schnepper, Mark, "Non-Linear Inverse Liquid-Solid Chromatography as a Methodology to Characterize Drug Concentration Losses to Polymeric Materials used in Body-on-a-Chip Devices for Drug Discovery" (2020). Electronic Theses and Dissertations, 2020-. 285.
https://stars.library.ucf.edu/etd2020/285
NON-LINEAR INVERSE LIQUID-SOLID CHROMATOGRAPHY
AS A METHODOLOGY TO CHARACTERIZE
DRUG CONCENTRATION LOSSES TO POLYMERIC MATERIALS
USED IN BODY-ON-A-CHIP DEVICES FOR DRUG DISCOVERY

by

MARK TOSHIRO SCHNEPPER
B.A. Whitman College, 2008
M.S. University of Central Florida, 2014

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Materials Science and Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2020

Major Professor: James J. Hickman
ABSTRACT

Body-on-a-chip and human-on-a-chip systems are currently being used to augment and could eventually replace animal models in drug discovery and basic biological research. However, hydrophobic molecules, especially therapeutic compounds, tend to adsorb to the polymer materials used to create these microfluidic platforms, which may distort the dose-response curves that feed into Pharmacokinetic/Pharmacodynamic (PK/PD) models which translate preclinical data into predictions of clinical outcomes. Adsorption of hydrophobic molecules to these polymer materials needs better characterization.

Inverse Liquid-Solid Chromatography paired with a numerical optimization based on the Langmuir model of adsorption was used to characterize the adsorption isotherm parameters of selected drugs to polydimethylsiloxane (PDMS) and polymethylmethacrylate (PMMA), polymers commonly used in these platforms after extensive modification to an existing HPLC-MS instrument.

Surface modification by organosilanes is one method being explored to modify PDMS, but the effect of organosilanes on drug adsorption isotherms are not well characterized. We utilized Inverse Liquid-Solid Chromatography (ILC) to characterize the adsorption parameters of the selected drugs with native PDMS and organosilane-modified (fluoropolymer (13F) and polyethylene glycol (PEG)) PDMS surfaces to correlate the modifications to changes in drug adsorption. We found that the organosilane modifications significantly changed the energy of adsorption.
The adsorption isotherms were then compared against concentration measurements of drugs recirculated in these platforms. It was found that the adsorption alone does not account for the drug concentration losses, which was expected as the drugs will diffuse into the bulk of the material. Organosilane surface modifications were successfully made to PDMS parts but were not fully adequate in preventing concentration losses, also due to diffusion into the bulk. Future work will characterize the diffusion and be integrated with the work presented in this dissertation to create a larger model of drug adsorption and diffusion which drives the concentration losses.

This research establishes a foundation for a new approach whereby quantifying drug or drug candidate interactions before system dosing and including this data in the PK/PD models, that polymers used in these platforms need not be limited to “less-adsorbing” materials.
I dedicate this *magnum opus* to:

my wife Alisha Colón,

my parents Sasha Kusel and David Schnepper,

my wife’s parents Marie and Carlos Colón,

my brother Alex Schnepper and his wife Kel Peyton,

my wife’s siblings and spouses Dominic and Carissa Colón, Eric and Tara Colón, and Jessica and Frank Peppe

and our dog Ada Colón.

Each of you granted me the grit to finish this trek.
ACKNOWLEDGEMENTS

Many thanks for the years of guidance from my advisor and committee chair, Dr. James J. Hickman, and committee members Dr. Stephen Lambert, Dr. Kevin Coffey, Dr. Andre Gesquiere, and Dr. Lei Zhai. Many thanks to my coauthor Dr. Jeff Roles, for patiently working with me. Many thanks to Dr. Cathleen Wolf for helping me polish off my papers for submission. Many thanks to the surface chemists who have worked alongside me through the years. Many thanks to the Chromatography Forum and HP1050 Service Manual, my dear companions through numerous repairs.

This work was supported with funding from NIH grants 5R01NS050452 and 2R44TR001326.

Content in this dissertation is reprinted with permission from Schnepper et al., Characterization of Drug-Polymer Adsorption Isotherms in Body-on-a-Chip Systems by Inverse Liquid-Solid Chromatography. *ACS Biomaterials Science & Engineering*. ©2020 American Chemical Society.¹ Content in this dissertation is reprinted with permission from Schnepper et al., Inverse Liquid Solid Chromatography to Evaluate Drug Interactions with Organoisilane Modified Polydimethylsiloxane for use in Body-on-a-Chip Systems. *Biotechnology Progress*. ©2020 American Institute of Chemical Engineers.²
# TABLE OF CONTENTS

LIST OF FIGURES .............................................................................................................. xi

LIST OF TABLES ................................................................................................................ xvii

LIST OF ACRONYMS/ABBREVIATIONS ........................................................................ xix

CHAPTER ONE: INTRODUCTION ....................................................................................... 1

Drug discovery overview ................................................................................................. 1

Design of a body-on-a-chip system .................................................................................. 3

Interactions with material surfaces .................................................................................... 9

Factors that affect adsorption & absorption ..................................................................... 12

CHAPTER TWO: INVERSE LIQUID-SOLID CHROMATOGRAPHY .................................. 18

Introduction – The Characterization of Adsorption .......................................................... 18

Adsorption Isotherm Modelling ....................................................................................... 18

Adsorption Characterization by Chromatographic Methods .......................................... 25

Numerical Method ........................................................................................................... 30

Theory ............................................................................................................................. 32

Numerical Algorithm ...................................................................................................... 36
CHAPTER THREE: METHANOL AS A MOBILE PHASE MODIFIER

Introduction ........................................................................................................................................... 53

Results .................................................................................................................................................. 54

Discussion .............................................................................................................................................. 60

Conclusion ............................................................................................................................................. 66

CHAPTER FOUR: ORGANOSILANE SURFACE MODIFICATIONS

Introduction ............................................................................................................................................. 67

Experimental ......................................................................................................................................... 74

Reagents & Materials ............................................................................................................................ 74

Organosilane modification of microspheres .......................................................................................... 74
APPENDIX A: SOURCE CODE ............................................................................................................. 117

Numerical_Method_v0.4.py ............................................................................................................. 118

simtools_branch_class2.py ......................................................................................................... 119

simmethods_branch_class.py ....................................................................................................... 138

Make Summary v4.1.py ................................................................................................................. 164

N Determination v3.0.py ................................................................................................................. 169

CSV to XLSX v2.py ......................................................................................................................... 177

Clean Chromatograms v2.2.py ...................................................................................................... 180

MYMACRO.mac .............................................................................................................................. 190

REFERENCES .................................................................................................................................... 193
LIST OF FIGURES

Figure 1. Simplified microfabrication steps used to create microchips used for cell culture. Not to scale. .................................................................................................................................................. 6

Figure 2. Exploded diagram of an example housing (left), and assembled with screws (right) .................................................................................................................................................. 8

Figure 3. Adsorption patterns of drugs on columns of the silicone-coated porous glass. Reprinted from Journal of Pharmaceutical Sciences, 70(5), Mizutani, Estimation of protein and drug adsorption onto silicone-coated glass surfaces, 493-496, ©1981, with permission from Elsevier.29 .............................................................................................................................................. 11

Figure 4. Histogram of calculated Log P values listed in the databases Available Chemical Directory (ACD) and MACCS-II Drug Data Report (MDDR), and measured Log P values listed in the database Comprehensive Medicinal Chemistry (CMC). Reprinted by permission from Springer Nature: Journal of Computer-Aided Molecular Design, Property distribution of drug-related chemical databases, Oprea, ©2000.35 ........................................................................................................ 15

Figure 5. Histogram of number of H-bond donors and acceptors in drugs listed in the databases Available Chemical Directory (ACDF) and MACCS-II Drug Data Report (MDDRF), filtering out “drug-like” structures. Reprinted by permission from Springer Nature: Journal of Computer-Aided Molecular Design, Property distribution of drug-related chemical databases, Oprea, ©2000.35 ........................................................................................................................................ 16

Figure 6. Schematic of adsorption and associated terms relevant for this dissertation. .. 18
Figure 7. Example of a typical adsorption isotherm. A linear approximation can be made at very dilute concentrations, but as concentration increases the isotherm becomes nonlinear. For this isotherm, the nonlinearity is due to the finite number of adsorption sites with which an analyte can interact.

Figure 8. Five types of van der Waals adsorption isotherms. Reprinted with permission from Brunauer et al., On a Theory of the van der Waals Adsorption of Gases. *Journal of the American Chemical Society*. ©1940 American Chemical Society

Figure 9. Comparison between conventional analytical chromatography (top row) and inverse chromatography (bottom row). Analytical chromatography is used to separate a mixture of unknown analytes for identification and quantification downstream. Inverse chromatography is used to characterize the surface of a material packed into a column.

Figure 10. Comparison of the typical chromatograms and adsorption isotherm measurements of different inverse chromatography methods. Figure adapted with permission from Ylä-Mäihäniemi & Williams, A Comparison of Frontal and Nonfrontal Methods for Determining Solid-Liquid Adsorption Isotherms Using Inverse Liquid Chromatography. *Langmuir*. ©2007 American Chemical Society

Figure 11. Overlapping curves of diclofenac at 2x10^{-4} M and 1x10^{-4} M concentrations, demonstrating the virtual peaks covered by the Elution by Characteristic Points (ECP) method.

Figure 12. Representative SEM images of the PMMA (A) and PDMS (B) microspheres. Histograms and fitted Weibull distributions (C & D) describing the particle size distribution of the microspheres.
Figure 13. Example experimental band profiles and fitting of simulated band profiles of drugs eluting through PMMA and PDMS columns. Verapamil on PDMS would not elute within a reasonable time or with sufficient peak height to be quantified. Scaled to best display the shape of the band profiles and simulated curve fit.

Figure 14. Langmuir adsorption isotherms for drugs on PMMA and PDMS. The horizontal axis shows the mobile phase concentration at equilibrium. For reference, drug concentrations in human-on-a-chip systems are typically in the nM (1x10^-9) to µM (1x10^-6) region. The vertical axis shows the concentration on material surfaces at equilibrium. PMMA (dashed lines) show less adsorption of drugs to the material surfaces than PDMS (solid lines). Drugs are organized by color, with acetaminophen (APAP) in red, diclofenac (DCF) in green, and verapamil (VER) in blue. The adsorption isotherm depicting verapamil adsorption to PDMS uses Langmuir terms derived from an extrapolation of changes in the Langmuir terms with respect to methanol concentration.

Figure 15. Changes to the band profile of verapamil on PDMS for four dosages of methanol in PBS (15 to 40% v/v in 5% v/v increments). Associated simulated curves also shown for both Langmuir and bi-Langmuir adsorption isotherm models. Horizontal axis represents time in minutes, vertical axis represents micromolar concentration. Note the wide difference of the peak elution time between 15% v/v methanol at 80 minutes and 40% v/v methanol at 1 minute.

Figure 16. Changes in the saturation capacities and adsorption equilibrium constants with respect to methanol concentration. Acetaminophen shown in red, diclofenac in green, and
verapamil in blue. PDMS shown with solid lines, PMMA shown with dashed lines. Error bars
denote standard error of the mean.......................... 56

Figure 17. Linear regression of the logarithm of the saturation capacity ( log([S0]) ) and
logarithm of the adsorption equilibrium constant ( log(Keq) ) versus methanol concentration for
acetaminophen, diclofenac, and verapamil eluting against PDMS. Linear regressions were based
upon the 15-30% v/v methanol concentrations and had good agreement with the measured values
for acetaminophen and diclofenac at 0% v/v methanol.................................................. 59

Figure 18. Illustrated mechanisms of hydrophobic recovery of modified PDMS.
Reconstruction (left) is one mechanism that reverses energy-based and organosilane
modifications of the surface. Rotation of functional groups around the polymer backbone,
condensation reactions between surface hydroxyl groups, and the diffusion of low molar mass
chains to the surface are also mechanisms for hydrophobic recovery. Left figure reprinted with
permission from Keefe et al., Suppressing Surface Reconstruction of Superhydrophobic PDMS
Using a Superhydrophilic Zwitterionic Polymer. Biomacromolecules. ©2012 American
Chemical Society.93 Right figure reprinted with permission from Yeh et al., Modification of
©2014 American Chemical Society.94 .................................................................................. 70

Figure 19. XPS spectra of native PDMS-coated microspheres (top), 13F-modified
PDMS-coated microspheres (middle), and PEG-modified PDMS-coated microspheres (bottom).
Presence of the 13F-modification was evident by the addition of a F1s peak at 685 – 688 eV and
fluorinated carbon peaks in the C1s region associated with CF2 and CF3 at 291.5 and 293.6 eV.
Presence of the PEG-modification was evident by the addition of a C-O peak at 286.4 eV in the C1s region.

**Figure 20.** Representative examples of elution band profiles, comparing between materials: PDMS (red), 13F-PDMS (green), and PEG-PDMS (blue). Experimental band profiles (solid lines) and simulated band profiles (dashed lines) were in good agreement. Scales selected to best show the peak shapes. (A) Acetaminophen, 10 µL injection, λ = 297 nm, 1 mL · min⁻¹ flowrate, 0% v/v methanol. (B) Diclofenac, 100 µL injection, λ = 276 nm, 1 mL · min⁻¹ flowrate, 0% v/v methanol. (C) Verapamil, 100 µL injection, λ = 229 nm, 3 mL · min⁻¹ flowrate, 15% v/v methanol. Drugs were most retained by native PDMS.

**Figure 21.** Semi-log plots for the monolayer saturation capacity [S₀] and adsorption equilibrium constant Kₑq with respect to the percentage of methanol present in the mobile phase. Linear regressions are based only on the 15 – 30% v/v methanol measurements. Each regression line represents one packed column (n = 3 for each drug-material combination). The extrapolated values for 0% v/v methanol tended to agree with the measured values.

**Figure 22.** Langmuir adsorption isotherms for acetaminophen, diclofenac, and verapamil versus native PDMS-coated silica microspheres (red), 13F-modified PDMS-coated silica microspheres (green), and PEG-modified PDMS-coated microspheres (blue). Isotherms for acetaminophen and diclofenac were based solely on the measured values at 0% v/v methanol in the mobile phase. Isotherms for verapamil were based on the extrapolated terms as determined by the linear regression of measurements made between 15 – 30% v/v methanol.
**Figure 23.** Effect of the organosilane surface modifications on the energies of adsorption for selected drugs. Error bars denote standard deviation.......................................................... 91

**Figure 24.** Adsorption channel dimensions used. The top was defined by a 5.5 mm thick PMMA plate. The microfluidic channel layer was defined by a 0.5 mm PDMS sheet. Below the channel layer was another PDMS sheet. Below that another PMMA plate for structural integrity. All layers were held together with screws through the device. .......................................................... 99

**Figure 25.** A) Surface plots depicting the percentage of drug predicted to remain as a function of initial drug load [C0] and surface area-to-volume ratio (A / V) as calculated by (40), using the saturation capacity [S0] and adsorption equilibrium constant Keq for the drug-material combination listed. All curves predict domains where the drug is completely retained in solution or completely adsorbed, with a sharp transition between the two domains. B) Lines depict the percentage of drug predicted to remain, specifically for the surface area-to-volume ratio of the device described. As adsorption to PMMA was predicted to be much less than adsorption to PDMS, only the curves for adsorption to PDMS was used. Points represent the measured amount remaining after 24 hours, error bars represent the standard deviation................................. 103

**Figure 26.** XPS spectra and contact angle of native PDMS gasket material (top), 13F-modified PDMS gasket material (middle), and PEG-modified PDMS gasket material (bottom). Contact angle was measured with a sessile 5 µL drop of water. Presence of the 13F-modification was evident by the addition of a F1s peak at 685 – 688 eV and fluorinated carbon peaks in the C1s region associated with CF2 and CF3 at 291.5 and 293.6 eV, as well as a slight increase in the contact angle. Presence of the PEG-modification was evident by the addition of a C-O peak
at 286.4 eV in the C1s region and a large decrease in the contact angle indicating a more hydrophilic surface.

**Figure 27.** Lines depict the percentage of drug predicted to remain, specific to the surface area-to-volume ratio of the device described (750 m⁻¹). Points represent the measured amount remaining after 24 hours, error bars represent the standard deviation.
LIST OF TABLES

Table 1. List of symbols and their meaning ................................................................. 32

Table 2. Drug Information. Properties cited from Wishart, et al. 28 except where otherwise noted. ............................................................................................................ 46

Table 3. Langmuir Terms for Drug Adsorption on Polymeric Materials ................... 51

Table 4. Measured & Extrapolated Langmuir Terms in the Methanol-free Condition.... 60

Table 5. Measured & Extrapolated Langmuir Adsorption Isotherm Terms and Energies .................................................................................................................. 86

Table 6. Initial, Change, Equilibrium Table of Adsorption ........................................ 101

Table 7. Two-way ANOVA of endpoint concentration relative to native PDMS ....... 110
LIST OF ACRONYMS/ABBREVIATIONS

13F trichloro(1H,1H,2H,2H-perfluorooctyl)silane
ANOVA Analysis of variance
APAP Acetaminophen
BSA Bovine serum albumin
C18 Octadecyl carbon chain bonded silica
DCF Diclofenac
DETA 3-trimethoxysilylpropyl)diethylenetriamine
ECP Elution by Characteristic Points
FA Frontal Analysis
FACP Frontal Analysis by Characteristic Points
HETP Height Equivalent of a Theoretical Plate
HPLC High Performance Liquid Chromatography
ILC Inverse Liquid-Solid Chromatography
MEMS Microscale ElectroMechanical Systems
MeOH Methanol
PBS Phosphate Buffered Saline
PDMS Polydimethylsiloxane
PEG 2-[methoxy(polyethyleneoxy)₆-9propyl]trimethoxysilane
PK/PD Pharmacokinetic / Pharmacodynamic Modelling
PM Peak Maximum
PMMA Polymethylmethacrylate
RP-LC Reverse-Phase Liquid Chromatography
SAM Self-Assembled Monolayer
SEM Scanning Electron Microscopy
VER Verapamil
XPS X-ray Photoelectron Spectroscopy
CHAPTER ONE: INTRODUCTION

There is growing concern that drugs and soluble factors have been interacting significantly with materials used to build systems containing experiments between cultured cells and applied drugs, obscuring and distorting these experiments with undesired interactions with the material. This dissertation aims to improve body-on-a-chip systems used for drug discovery and disease modelling by characterizing the adsorption isotherms of the drug to the materials used in body-on-a-chip systems and investigating the effect of surface modifications to the adsorption isotherms.

This work is relevant to the accuracy of pharmacokinetic-pharmacodynamic (PK/PD) models which utilize the data from body-on-a-chip systems to create predictions of clinical outcomes. The following chapter will cover the background information that prompted and drives this line of inquiry.

Drug discovery overview

Medical advances rely on a steady stream of new therapeutics from the drug development pipeline. The discovery process is driven by a specific disease or syndrome that lacks an effective therapeutic agent. The current paradigm of drug discovery is target-driven, and goes through defined stages. Drug studies begin with the hypothesis of a target: a naturally occurring part of the cellular apparatus or system that if manipulated by an outside agent, yet to be discovered or developed, will result in a therapeutic effect on the disease process. An identified target is then evaluated through various methods (e.g., immunoprecipitation, genetic screening of
patients, transgenic animal models) to assess the validity of the target. Finding a drug candidate that can interact with the target requires an exhaustive search through a near-infinite catalog of possible compounds, using a multitude of methods to screen for efficacy and toxicity (e.g., computational models, *in vitro* cell culture of affected tissue, animal models). Only after this battery of filters will a drug candidate undergo a series of phases of clinical trials on human subjects and then onwards to widespread clinical use.

Over the past 50 years there has been increasing political pressure to eliminate animal-based studies, an established platform for preclinical drug candidate screening and disease modelling. The first step toward decreasing animal use began when Russel & Burch published “The principles of humane experimental technique” outlining the three “R’s” of animal research: “Replacement, reduction, & refinement”⁴. Contemporary philosopher Peter Singer then published “Animal Liberation,” a utilitarian argument against animal exploitation, which has been the foundation of the modern Animal Liberation political movement.⁵ In the past few decades, the European Union has passed legislation restricting the use of animals in research,⁶⁻⁷ most notably the Cosmetics Directive which has completely banned the use of animals in cosmetics testing.⁸ More recently, moral philosophy has begun to explore the ethical consequences revealed by research of animal cognition, concluding that current animal welfare policies lag behind or are in contradiction with recent scientific discoveries,⁹ which may prompt additional legislation and incentives for finding replacements.

However, any replacement of animal studies must address the unique aspects of animal models useful to the drug discovery process. For example, the interaction between multiple organs and tissue types, the emergent properties resultant from these interactions, and the
resulting functional and behavioral effects, cannot yet be fully recapitulated by non-animal models. Genetic manipulation via knockout, knockdown, or transgenics allows for the creation of animals that reliably produce analogues of disease states in humans. Animal models are also the only legal means to research injuries, paralysis, addiction, infectious-disease, and chemical exposure (chronic and acute), for the benefit of humans. Because animal studies have these unique applications, legislation limiting the use of animals will hinder drug development unless a replacement that can address these aspects is developed. Body-on-a-chip systems (also known as human-on-a-chip or organ-on-a-chip systems) are currently being used to augment and could eventually replace animal models. This emergent field combines in vitro cell culture of human-derived tissue, microfluidics, pharmacokinetic-pharmacodynamic (PK/PD) modeling, and MEMS (Microscale ElectroMechanical Systems) to produce devices that mimic human organs and tissue for drug screening, disease modelling, and fundamental biological research.

Design of a body-on-a-chip system

In a perspective published in Nature, Horrobin warned the direction of modern biomedical research has been disconnecting with medical reality, that biomedical research pours all of its funding and effort into simulations, cell culture, and animal models without checking on the assumption of congruity between these efforts and human disease. However, a body-on-a-chip system should not be expected to fully simulate a whole human body; they are models, and a useful model simplifies a complex system to reveal useful information. The abstractions made for a body-on-a-chip system may decrease the accuracy of the model, but serves to keep the cost of implementation, size, and usability of the system under control.
Animal models are also approximations for humans, which are broadly accepted for research in medicine, drug discovery, and psychology, but are not expected to reflect the full range of human physiology. However, unlike body-on-a-chip systems, animal models are based on a living organism with all the complex interactions already in place. A body-on-a-chip system, as an artificial construct, relies on a top-down design perspective to engineer a system that reflects the aspects of human physiology we wish to model. Wikswo, et al. 11 provide an overview of various engineering challenges that need to be addressed in the creation and control of an integrated body-on-a-chip system, detailed below.

Wikswo, et al. 11 advance three points relevant to the Hybrid Systems Lab’s designs: organ scale, total volume, and a universal blood surrogate. Regarding organ scale, they point out that a mismatch of scales between coupled organs could yield misleading results: “For instance, if a 0.1 µLung [microscale lung] were coupled to a mLiver [milliscale liver], the mLiver might not respond significantly to drugs or toxins delivered through or metabolized by the µLung, e.g., the conversion of angiotensin I into angiotensin II.” Of similar concern is the total volume of the medium, since if the medium volume is unrealistically large relative to the cells cultured, then any soluble factors they express could be so dilute that any studies would effectively be of two isolated organs. A universal blood surrogate to culture multiple disparate cell types at once also needs to be developed, a line of investigation the Hybrid Systems Lab has worked on extensively which will not be detailed here as it is outside the scope of this work.

The design paradigm selected by the Hybrid Systems Lab for its body-on-a-chip systems is a bottom-up design approach, where discrete parts are developed separately then united to create emergent properties. Each organ or tissue type is at first developed as its own project with
MEMS chips and interrogation methods tailored for that organ. When the model organ sufficiently reflects the physiology in a whole organism, it can be added to another to expand the complexity of the overall system.

The MEMS devices utilized by the Hybrid Systems Lab are designed to interrogate cells non-invasively. These devices are each tailored to assess a single tissue type, following the bottom-up design approach. As an example, the cantilever chips were designed to measure the force output of muscle. As shown by Figure 1, the cantilever chips are made from silicon-on-insulator (SOI) wafers with a device thickness (DT) of 4 µm, buried oxide (BOX) of 1 µm, and handle wafer thickness (HT) of 500 µm. Using deep reactive ion etching (DRIE), a diving-board of monocrystalline silicon is made on the device side. DRIE is then used on the backside of the wafer to create an open hole under the cantilever. Muscle cells are seeded on top of the cantilever where mature myotubes form and contract under stimulation pulling the distal end of the cantilever upwards. The deflection is measured using principles akin to atomic force microscopy (AFM) in which a laser reflects off the cantilever bottom and into a detector. The deflection amplitude, board dimensions, and Young’s modulus of monocrystalline silicon are used to calculate the force output of single myotubes. When this muscle system incorporates another system designed for motoneurons the interactions between the two can be studied, such as the neuromuscular junction which is fundamental to the study of amyotrophic lateral sclerosis (ALS).
While it is easy to imagine a single unified chip with all the cell types of the body on it, it’s not possible to build all at once. Cells are used in three phases: proliferation, differentiation, and maturity. These phases are determined by the soluble factors the cells are bathed in and the cells develop at disparate rates characteristic to each cell type. This fundamental constraint of cell culture means it is far easier to culture different cell types separately on different chips and then integrate them into a common housing when the cells are mature for experimentation. The engineering designs for a common housing need to fulfill minimum characteristics to accommodate a bottom-up approach: 1) sealed chambers that immobilize individual chips, 2) accessible reservoirs to facilitate the removal and replenishment of cell culture medium, 3) connected chambers and reservoirs with defined microfluidic pathways, defined volumes, and pathways to interface electronically with chips, and 4) the ability to assemble a system around chips populated with cells (wet assembly), instead of seeding cells into a system assembled earlier (dry assembly). It is also useful if the device can be made of materials that are optically clear (for microscopy use), easy to machine for rapid prototyping, and are low cost.
Due to these engineering requirements these devices are commonly built using polymers, but which specific polymers to use is an open debate. Microfluidics material selection has been dominated by polydimethylsiloxane (PDMS) ever since Whitesides’ 1998 paper “Soft Lithography” lauding the material’s applicability for rapid prototyping of microfluidic devices,\textsuperscript{13} and has become the standard material for microfluidic engineers. In contrast, cell biologists have adopted polystyrene as their standard material over the past half century. As the fields of cell biology and microfluidics merge, material selection has become a point of contention.\textsuperscript{14}

The integrated common housings used by the Hybrid Systems Lab use a combination of PDMS and poly-(methyl methacrylate) (PMMA), better known as acrylic, which has been investigated as a material to create microfluidic devices.\textsuperscript{15-19} These materials were chosen because of their compatibility with the Universal Laser Systems CO\textsubscript{2} laser cutter (provided to UCF engineering students through the TI Innovation Laboratory, supported by a grant from Texas Instruments), allowing rapid prototyping and revision of housing designs. While there are setbacks in using these materials, discussed at the end of this section, the ability to rapidly prototype and update designs has greatly accelerated the research projects that utilize the housings. PDMS was selected for its ability to make watertight seals and its biocompatibility. PMMA was selected for its mechanical strength and optical clarity allowing for non-invasive study by phase microscopy.

Generally, the housing designs have two PDMS layers and two PMMA layers as demonstrated by Figure 2. The PDMS layers define the chip positions (bottom layer) and the microfluidic pathways (top layer) and are referred to as the “gaskets”. The gasket layers are sandwiched between two PMMA “plates”, which compress and support the whole structure. Two
holes in the top PMMA plate are used as cell medium reservoirs and are the main point of access for feeding and taking sample aliquots.

Figure 2. Exploded diagram of an example housing (left), and assembled with screws (right)

While the PDMS and PMMA allow rapid prototyping, the inherent disadvantage of using these materials, especially PDMS, is undesirable interactions between biomolecules and the material surface. The culturing medium in housings has constant contact with PDMS and PMMA surfaces giving ample opportunity for the materials and medium to interact. Generally, these interactions are classified into three categories: 1) material entering the medium, 2) loss of soluble factors from the medium to the material, and 3) changes to soluble factors in the medium from interaction with the material. Undesired material entering the medium includes, especially in the case of polymeric materials, unreacted monomer or plasticizers – for example, bisphenol A
(a xenoestrogen) leaching from polycarbonate water bottles.\textsuperscript{20-21} An example of changes to a soluble factor is the denaturing of adsorbed proteins, investigated in response to reported adverse health effects linked with silicone coated implants.\textsuperscript{22-23} The loss of soluble factors, specifically drugs, due to adsorption onto or absorption into the material in contact with the cell culture medium is the basis for this dissertation and will be elaborated upon in the following section.

Interactions with material surfaces

Loss of soluble factors via absorption into PDMS is well known, and the absorption of small hydrophobic fluorescent dyes has been reported in microfluidic systems,\textsuperscript{24} penetrating over 100 µm into the channel wall. The hydrophobicity and porosity of PDMS is considered the main driver of the absorption of small, hydrophobic molecules. While undesired for microfluidics, absorption into PDMS is useful for solid-phase extraction of hydrophobic molecules into PDMS-coated fibers to collect small samples (solid phase microextraction) for analysis in HPLC and GC systems,\textsuperscript{25} and for drug delivery by loading drugs into PDMS - mesoporous silica composites.\textsuperscript{26}

In microfluidic cell culture systems, the loss of soluble factors in cell culture medium from interaction with surrounding materials has already been demonstrated as significantly impacting cell culture results. As an example, Regehr, et al.\textsuperscript{27} investigated the interaction of PDMS with estradiol, a small hydrophobic steroid commonly known as estrogen (MW 272.38 g/mol, log P 4.01, ).\textsuperscript{28} MCF-7 human breast cancer cells were transfected with a luciferase reporter for Activator Protein 1 (AP-1), the activation of which is downstream of estradiol and prolactin signaling. Prolactin, in contrast to estradiol, is a protein with a molecular weight of 24
kDa and was not expected to be sequestered by PDMS. When the MCF-7 human breast cancer cells were cultured with PDMS they demonstrated no activity when exposed to estradiol but did exhibit activity when exposed to prolactin. In contrast, cells cultured without PDMS (on polystyrene, the historical standard for cell culture experiments) demonstrated activity to both estradiol and prolactin.

When body-on-a-chip systems are used to assess novel drug candidates, the calculations derived from PK/PD models of the systems will require accurate measurements of the concentrations of these drug compounds in the cell culture medium. If a significant proportion of the drug candidate is lost from the cell culture medium to the microfluidic channel walls it will distort the conclusions drawn from the resulting dose-response curves and quite possibly – as demonstrated above in Regehr, et al. 27 – sequester enough to elicit no response from the cells studied and yield false negatives.

As an example of small molecule drug concentration loss to PDMS, Mizutani 29, 30 characterized adsorption of drugs and proteins to glass and PDMS-coated vials designed to store doses administered to patients. Quantification of the loss was performed by Frontal Affinity, a chromatography technique where a constant concentration of an analyte is passed through a column packed with controlled pore glass beads or PDMS-coated beads and the eluate fractions are quantified. As adsorption sites are populated in the column, the analyte concentration in the eluate rises. The fractions were quantified by UV-Vis to produce a curve of concentration versus fraction number, which was integrated to calculate the total amount lost to adsorption. When eluted through the chromatography column, the drugs adsorbed in the following quantities: insulin, 71 mg/100m$^2$; epinephrine, <0.02 mg/100m$^2$; atropine, 0.81 mg/100m$^2$; physostigmine,
3.88 mg/100cm². These studies were aimed at finding the loss of drug to adsorption inside of a 20 mL vial with a surface area of 50 cm², under high concentrations and in solutions tailored for storage conditions, where the concentration loss is noteworthy but not worrisome. For body-on-a-chip systems, these numbers are more problematic.

**Figure 3.** Adsorption patterns of drugs on columns of the silicone-coated porous glass. Reprinted from Journal of Pharmaceutical Sciences, 70(5), Mizutani, Estimation of protein and drug adsorption onto silicone-coated glass surfaces, 493-496, ©1981, with permission from Elsevier.²⁹

Body-on-a-chip systems use concentrations of drugs and factors that reflect the physiological concentration, which will be orders of magnitude lower than a concentrated dose.
stored in a vial. Furthermore, as in all microfluidic systems, the surface area-to-volume ratio will be very high – much higher than 20 mL to 50 cm² (250 m⁻¹). This low concentration and high surface area-to-volume ratio greatly increases the fractional loss of drugs from interactions with the material surfaces. As an example, in one housing used by the Hybrid Systems Lab the working volume is 770 μL and the surface areas of acrylic and silicone are 973 mm² and 281 mm² respectively (1629 m⁻¹ total area to volume). Using the value for physostigmine cited earlier (3.88 mg/100m²), this would translate to 10.9 ng adsorbed onto the silicone alone (assuming a linear scaling of adsorption). The peak blood serum concentration of physostigmine occurs 3 hours after taking a 15 mg tablet dose of physostigmine succinate, with a concentration of 1.2 ng/mL. The amount of physostigmine in solution in a housing reflecting this concentration would be about 0.92 ng, meaning the amount that could be adsorbed is 12x the amount in solution, enough to sequester the drug entirely.

Factors that affect adsorption & absorption

Three aspects affect the adsorption or absorption of a molecule with a material: 1) The innate characteristics of the molecule: specifically, the molecular mass, hydrophobicity, and the functional groups that can interact with the material surface. 2) The innate characteristics of the surface: the functional groups presented on the surface and the material porosity and roughness. 3) The pH and ionic strength of the medium, which affects the ionic state of a molecule and the surface, and the solubility of the molecule.
In order to predict what kinds of surfaces drugs will interact with, it’s important to understand the innate characteristics of the drugs themselves. Drug compounds tend to follow Lipinski’s Rule of Five, a ‘rule-of-thumb’ metric that outlines characteristics of a molecule that are common to drug candidates that successfully pass to clinical use. There have been refinements made to the original rules published in 1997, but the basic outline still applies: 1) small molecular mass (MW < 500 Da), 2) cannot be excessively hydrophobic (octanol-water partition coefficient (log P) < 5), 3) fewer than 5 hydrogen bond donors (N-H and O-H bonds), and 4) fewer than 10 hydrogen bond acceptors (N or O atoms).

Perhaps most important for interactions with PDMS (and similarly hydrophobic materials, like PMMA) is the octanol-water partition coefficient (log $K_{ow}$ or log P), which describes the relative hydrophillic/hydrophobic character of the molecule, defined as the logarithm of the ratio of the equilibrium concentration of an unionized molecule in equal volumes of octanol and water in contact with each other, expressed as:

$$\log P = \log \left( \frac{[A]_{octanol}}{[A]_{water}} \right)$$

(1)

Typically, this measurement is done at whichever pH the unionized form of the drug is predominant. However, at physiological pH a drug may be in an ionized form which affects the amount of drug that may partition into water. An adjustment to (1) that accounts for the ionization of drugs in response to pH is the distribution coefficient (log D), expressed as:
\[
\log D = \log \left( \frac{[A]_{\text{unionized}}^{\text{octanol}} + [A]_{\text{ionized}}^{\text{octanol}}}{[A]_{\text{unionized}}^{\text{water}} + [A]_{\text{ionized}}^{\text{water}}} \right)
\]

Which accounts for both the unionized and ionized forms of the drug partitioning between octanol and water at the specified pH. Typical distribution coefficient values are reported at the physiological pH 7.4.

The partition and distribution coefficients have pharmaceutical significance as they predict the relative affinity a particular drug has between the lipid membrane (hydrophobic) or blood serum (hydrophilic),\(^\text{33}\) and has been demonstrated as predictive of absorption quantities into PDMS in early characterization work for solid phase microextraction.\(^\text{34}\) For the population of clinically used drugs, a meta-analysis of partition coefficients by Oprea\(^\text{35}\) showed histogram with a Gaussian distribution centered at “1.78, with mid-50% between 0.34 and 2.85 (90% below 4.0)” shown in Figure 4, implying that developing a hydrophilic surface may be enough to help repel non-specific adsorption by a large population of drugs.
A paper published by Mizutani and Mizutani \cite{Mizutani1996} illustrates the importance of hydrogen bonding in adsorption. It was discovered that basic drugs and protein adsorbed strongly to porous glass, while neutral and acidic drugs did not, and the explanation proposed was hydrogen bonding between the drug and surface silanol groups – note that roughly 75% of all drugs are basic and 20% are acidic, with only 5% non-ionizable,\cite{Oprea2000} so the affected population is in the majority. This highlights the importance of hydrogen bonding with surfaces. The meta-analysis by Oprea \cite{Oprea2000} also looked at the number of hydrogen-acceptor and hydrogen-donor groups, shown in Figure 5. The histogram of hydrogen-acceptors has a right-skewed peak centered at 5, with half the drugs listed possessing between 2-7 hydrogen-acceptors. Conversely, the histogram for hydrogen-donors is heavily skewed with half the drugs listed possessing 0-1 hydrogen-donors. The prevention of hydrogen bonding based adsorption may be achieved by presenting a hydrogen-acceptor rich surface.

**Figure 4.** Histogram of calculated Log P values listed in the databases Available Chemical Directory (ACD) and MACCS-II Drug Data Report (MDDR), and measured Log P values listed in the database Comprehensive Medicinal Chemistry (CMC). Reprinted by permission from Springer Nature: Journal of Computer-Aided Molecular Design, Property distribution of drug-related chemical databases, Oprea, ©2000.\cite{Oprea2000}
Figure 5. Histogram of number of H-bond donors and acceptors in drugs listed in the databases Available Chemical Directory (ACDF) and MACCS-II Drug Data Report (MDDRF), filtering out “drug-like” structures. Reprinted by permission from Springer Nature: Journal of Computer-Aided Molecular Design, Property distribution of drug-related chemical databases, Oprea, ©2000.35

The solution matrix also strongly influences the amount of drug adsorbed to a surface. Mizutani 29 showed that the pH and ionic strength of the solution influenced the adsorption of proteins and drugs to PDMS-coated glass beads. Higher ionic strength solutions showed an increase of adsorbed proteins and drugs. As the ionic strength of a solution increases, the water of solvation around the drug or protein is attracted to the ions and the amount of water that can interact with the charged portions of the drug or protein is decreased – this is the principle for salting-out of proteins (due to lowered solute-solvent interactions, favoring solute-solute interactions leading to aggregation), and has similar repercussions for solute-surface interactions.

For cell culture to function properly, the needs of the cells dictate the pH and ionic strength of the medium and cannot be changed to accommodate non-specific adsorption. We also cannot change the character of the drugs we study. Thus, there are only three options left to address the adsorption: 1) change the material used to be less adsorbing. This is undesirable as
the materials used have plenty of engineering properties that might be sacrificed by adopting a new material. 2) Modify the material or material surface. This has been investigated by others but has not yet yielded a modification to PDMS that can adequately address the problem. 3) Carefully characterize the adsorption and account for it in the dose-response curves that feed into the PK/PD models. This would allow for the continued use of established materials and serve to obviate the problem. To this end, this dissertation demonstrates a new method to characterize adsorption by Inverse Liquid-Solid Chromatography (ILC).
CHAPTER TWO: INVERSE LIQUID-SOLID CHROMATOGRAPHY

Content in this chapter is reprinted with permission from Schnepper et al., Characterization of Drug-Polymer Adsorption Isotherms in Body-on-a-Chip Systems by Inverse Liquid-Solid Chromatography. ACS Biomaterials Science & Engineering. ©2020 American Chemical Society.¹

Introduction – The Characterization of Adsorption

Adsorption Isotherm Modelling

![Diagram of adsorption and associated terms](image)

**Figure 6.** Schematic of adsorption and associated terms relevant for this dissertation.
Formally defined, adsorption is the adhesion of a molecule from a gas or liquid phase to a surface (typically the surface of a solid), as depicted schematically in Figure 6. The adsorption reaction can be described simply by:

\[ C + S \leftrightarrow CS \]  

(3)

Where C is the analyte dissolved in the gas or liquid phase, S is an unoccupied adsorption site on the solid phase surface, and CS is an analyte occupying an adsorption site on the solid surface. The equilibrium state of this reaction is dictated mostly by these main factors: 1) the concentration of analyte dissolved in the solution, 2) the balance of the rates of adsorption to and desorption from the surface, 3) the number and availability of sites on the surface to adsorb to.

When the equilibrium state of the adsorption process is graphed to show the concentration adsorbed to a surface as a function of the concentration dissolved in solution, it describes a curve known as an adsorption isotherm. A typical isotherm is shown in Figure 7 as an example.
**Figure 7.** Example of a typical adsorption isotherm. A linear approximation can be made at very dilute concentrations, but as concentration increases the isotherm becomes nonlinear. For this isotherm, the nonlinearity is due to the finite number of adsorption sites with which an analyte can interact.

At very dilute concentrations an adsorption isotherm is practically linear, meaning the concentration of analyte adsorbed to a surface is proportional to the concentration of analyte dissolved in the solution. Since many analytical applications of adsorption (e.g. analytical chromatography) typically operate at very dilute concentrations the assumption of a linear adsorption isotherm is useful. However, as concentration increases the availability of unoccupied adsorption sites becomes limited and the adsorption isotherm enters a nonlinear domain. The
nonlinear region is important for understanding the mechanisms behind adsorption and finds applications in the field of nonlinear chromatography for preparative chromatography or frontal affinity measurements.

Early modelling of adsorption isotherms was published by Irving Langmuir in 1916 and 1918,\textsuperscript{37-39} which earned him The Nobel Prize in Chemistry 1932. Langmuir demonstrated that the adsorption of gases to a heated filament was limited to a monolayer capacity of the surface and a kinetics-based derivation that could describe the adsorption isotherm was developed as follows (equations adapted to be uniform with the symbols used elsewhere in this dissertation):

We can express the forward rate of adsorption \((r_{ad})\) as:

\[
 r_{ad} = k_{ad}[C][S] 
\]

(4)

where \(r_{ad}\) is the forward rate of adsorption, \(k_{ad}\) is the kinetic constant of adsorption, \([C]\) is the concentration of analyte in solution, and \([S]\) is the concentration of unoccupied adsorption sites on the surface of the adsorbate.

We can also express the reverse rate of desorption \((r_d)\) from the surface as:

\[
 r_d = k_d[CS] 
\]

(5)

where \(r_d\) is the reverse rate of desorption, \(k_d\) is the kinetic constant of desorption, and \([CS]\) is the concentration of analyte adsorbed to the adsorbate surface. Since the equilibrium state is when
these two rates are equal, the equilibrium constant can be defined by combining (4) and (5) such that:

\[
\frac{[CS]}{[C][S]} = \frac{k_{ad}}{k_d} = K_{eq}^C
\]

where \(K_{eq}^C\) is the equilibrium constant for the adsorption of analyte C to the adsorbent surface.

The adsorbent surface site concentration can be defined by:

\[
[S_0] = S_0/\text{area}
\]

where \(S_0\) is the initial number of unoccupied adsorption sites on the surface, area is the total area of the adsorbent surface, which gives a concentration of the initial adsorption sites per unit area represented by \([S_0]\). Since the total number of adsorption sites is conserved, we can also define \([S_0]\) further:

\[
[S_0] = [S] + [CS]
\]

and then use (6) to substitute for \([S]\) in (8):

\[
[S_0] = \frac{[CS]}{K_{eq}^C [C]} + [CS] = \frac{1 + K_{eq}^C [C]}{K_{eq}^C [C]} [CS]
\]

and rearrange to give:
\[ [CS] = \frac{K_{eq}[S_0][C]}{1 + K_{eq}^C[C]} \]

(10)

Which is the equation for a Langmuir adsorption isotherm. It is common to express this as a proportion of the adsorption sites that are occupied, by rearranging (10) slightly:

\[ \theta_C = \frac{[CS]}{[S_0]} = \frac{K_{eq}^C[C]}{1 + K_{eq}^C[C]} \]

which makes it easy to see that:

\[ \lim_{[C] \to \infty} \theta_C = \lim_{[C] \to \infty} \left( \frac{K_{eq}^C[C]}{1 + K_{eq}^C[C]} \right) = 1 \]

meaning that as the concentration of dissolved analyte \([C]\) increases, the concentration of analyte on the adsorbent surface \([CS]\) will approach the monolayer saturation capacity \([S_0]\).

The monolayer saturation capacity as depicted in the Langmuir adsorption isotherm model arises from the fact that adsorbate-adsorbate interactions are typically much weaker than adsorbate-adsorbent interactions. This is true for most gas-solid and almost all liquid-solid adsorbate-adsorbent systems.\textsuperscript{40-41}

However, a monolayer capacity isn’t a universal feature of adsorption isotherms. In certain uncommon cases, the adsorbate-adsorbate interaction may be great enough to build multiple layers of adsorbate. This will result in adsorption isotherms similar to those shown in \textbf{Figure 8}, which depicts common adsorption isotherm types for gas-solid phase equilibria. Type I is the typical Langmuir isotherm. Types II and III model a buildup of multiple layers of
adsorbate, up to the vapor pressure of the adsorbate. Types IV and V model systems where small pores in the adsorbent surface are filled with a few layers of adsorbate. Again, these cases are uncommon and are included here only for completeness.

Figure 8. Five types of van der Waals adsorption isotherms. Reprinted with permission from Brunauer et al., On a Theory of the van der Waals Adsorption of Gases. *Journal of the American Chemical Society*. ©1940 American Chemical Society

The Langmuir adsorption model also only applies under the conditions that there is a single analyte adsorbing and the adsorption sites are energetically homogenous. There are many alternative models that cover the cases where there exists a distribution of adsorption site
energies or there is competitive adsorption with multiple analytes. The basic Langmuir adsorption model can be adjusted to account for multiple adsorption site energies by supposing there exist multiple, independent types of adsorption sites on the adsorbent surface. Since these sites are independent, the multi-Langmuir model is simply a summation of single Langmuir models:

\[
[Q] = \sum_{i=1}^{n} \frac{K_{eq, i} [S_0]_i [C]}{1 + K_{eq, i} [C]}
\]

(11)

where \(K_{eq, i}\) is the adsorption equilibrium constant and \([S_0]_i\) is the monolayer saturation capacity for the \(i^{th}\) type of adsorption site. The concentration of analyte adsorbed to the surface per mass of stationary phase is given by \([Q]\), and is the summation of \(n\) independent Langmuir adsorption sites from \(i = 1\) through \(i = n\).

Adsorption Characterization by Chromatographic Methods

Adsorption isotherm modelling and chromatography are deeply intertwined, as differential adsorption is the underlying mechanism of chromatographic peak separation. To review, the analytical application of Liquid Chromatography is to identify and quantify the components of an unknown sample mixture. The predominant implementation is by Reverse-Phase Liquid Chromatography (RP-LC), in which an unknown sample is injected into an aqueous mobile phase that runs across a bonded, non-polar stationary phase packed into a
stainless-steel column. Since each analyte from the unknown sample will have a different, characteristic affinity for both the mobile and stationary phases, each traverses the analytical column at different linear velocities. This results in a separation of the constituent analytes into discrete peaks measured by the detector (commonly a UV-Vis or mass spectrometer).

Inverse Liquid-Solid Chromatography (ILC) uses the relationship between chromatography and adsorption isotherms to characterize the surface of a stationary phase, as illustrated by Figure 9. The technique of ILC entails loading known analytes one-at-a-time to probe the stationary phase. The retention data of the probe analyte yields information about the stationary phase surface, such as the monolayer saturation capacity and the adsorption equilibrium constant for the given probe. One advantage of ILC specific to the analysis of body-on-a-chip materials is that the mobile phase can have the same temperature, pH, and ionic strength as the cell culture media used for cell culture experiments, which are all factors of adsorption discussed in the last chapter.
Figure 9. Comparison between conventional analytical chromatography (top row) and inverse chromatography (bottom row). Analytical chromatography is used to separate a mixture of unknown analytes for identification and quantification downstream. Inverse chromatography is used to characterize the surface of a material packed into a column.

Introduction of the analyte to the column may be done by either an injection (pulse methods) or by introducing a continuous concentration of the analyte (frontal methods), as illustrated by Figure 10. Frontal methods analyze either 1) the shock front generated by suddenly introducing a constant concentration of the analyte into the mobile phase to chart a breakthrough curve or 2) by suddenly cutting the feed of analyte to chart the desorption curve. Pulse methods inject a bolus of analyte and chart the resulting band profile – thus consuming much less solvent and enabling faster data collection compared to frontal methods.43
**Figure 10.** Comparison of the typical chromatograms and adsorption isotherm measurements of different inverse chromatography methods. Figure adapted with permission from Ylä-Mäihäniemi & Williams, A Comparison of Frontal and Nonfrontal Methods for Determining Solid-Liquid Adsorption Isotherms Using Inverse Liquid Chromatography. *Langmuir.* ©2007 American Chemical Society.

Until the development of the Theory of Characteristics by Rhee et al. (discussed in the following paragraph), inverse chromatography measurements were limited to measuring just a single point on an adsorption isotherm at a time (discrete measurements). Frontal Analysis (FA) works by loading a constant concentration of analyte until a breakthrough curve is observed, meaning that the column adsorption sites have been saturated and the concentration of analyte at the column outlet equals the inlet. Graded increases in concentration will produce additional breakthrough curves to measure additional points along the adsorption isotherm. The Peak Maximum (PM) method injects different concentrations of analyte and the net retention volume is used to build the points along the adsorption isotherm.
The Theory of Characteristics developed by Rhee et al. shows that each point of concentration of an analyte eluting through a column will travel at a characteristic velocity. Consequently, the diffuse boundary of the desorption curve for smaller concentrations of an analyte will lie along the same desorption curve for larger concentrations of an analyte – an example of this is shown in Figure 11. Thus, by studying the diffuse boundary associated with the desorption of analyte through either an injection (ECP – Elution by Characteristic Points) or the sudden cutoff of a constant feed of analyte (FACP – Frontal Analysis by Characteristic Points) multiple points of the adsorption isotherm may be measured simultaneously by one experimental run. The Theory of Characteristics relies on some assumptions: 1) the adsorption isotherm is a Langmuir-like adsorption isotherm, and 2) thermodynamic factors dominate the shape of the band profile and kinetic band-broadening effects are minimized.
Figure 11. Overlapping curves of diclofenac at $2 \times 10^{-4}$ M and $1 \times 10^{-4}$ M concentrations, demonstrating the virtual peaks covered by the Elution by Characteristic Points (ECP) method.

Numerical Method

The Elution by Characteristic Points (ECP) method was selected as the starting point of this work. Unlike frontal methods, a single overloaded elution band profile is obtained after injection of a bolus of analyte. The ECP method is based on the assumption that smaller injections of an analyte will lie on the same desorption curve and that a single large injection of analyte can provide the raw data needed to calculate the isotherm. An overloaded stationary phase results in characteristic non-linear band profiles and enables the capture of the non-linear portion of the isotherm, necessary for accurate calculation of all the isotherm parameters. With surfaces containing multiple types of adsorption sites, the site of highest adsorption energy tends
to be occupied first (before lower energy sites), and this can result in a non-linear system even at relatively low concentrations below the level where the (generally) more abundant lower energy sites are filled. The other cause of non-linearity in a Langmuir-type isotherm (concave toward the concentration axis) is when the concentration exceeds the point where all sites are occupied no further adsorption occurs, reaching a monolayer saturation capacity.

In the ECP method, the isotherm is calculated by simply integrating the band profile, from the low energy limit to the peak maximum, as the shape of the band profile is assumed to be due only to the thermodynamic factors (the shape of the isotherm) and neglects the influence of normal kinetic band-broadening effects of mass transfer and axial dispersion. The ECP method is accurate only when the thermodynamic factors greatly outweigh the normal band-broadening effects, so this method has been largely replaced by numerical techniques that take band-broadening into account more accurately and derive the optimum isotherm by:

1) assuming a starting set of isotherm parameters (in our case they are obtained by an approximation obtained by performing the ECP method).

2) using an optimization strategy to adjust the isotherm parameters and then after each adjustment simulate a band profile.

3) compare the simulated band profile to the experimental band profile.

4) repeat this process until the simulated profile matches the experimental results.\(^\text{40, 46-48}\)
**Table 1.** List of symbols and their meaning

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_g$</td>
<td>mm$^2$</td>
<td>Column Cross Section of (mobile phase permeable portion)</td>
</tr>
<tr>
<td>$[C]$</td>
<td>$\mu$mol $\cdot$ mL$^{-1}$</td>
<td>Concentration of the solute in the mobile phase</td>
</tr>
<tr>
<td>$D$</td>
<td>cm$^2$ $\cdot$ min$^{-1}$</td>
<td>Apparent diffusion coefficient of the solute in the mobile phase</td>
</tr>
<tr>
<td>$F$</td>
<td>mL $\cdot$ min$^{-1}$</td>
<td>Volumetric flow rate of mobile phase</td>
</tr>
<tr>
<td>$H$</td>
<td>cm</td>
<td>Column Height Equivalent of a Theoretical Plate (HETP)</td>
</tr>
<tr>
<td>$L$</td>
<td>cm</td>
<td>Column length</td>
</tr>
<tr>
<td>$m_g$</td>
<td>g $\cdot$ cm$^{-1}$</td>
<td>Mass of stationary phase per unit column length</td>
</tr>
<tr>
<td>$M_s$</td>
<td>g</td>
<td>Mass of stationary phase</td>
</tr>
<tr>
<td>$N$</td>
<td>unitless</td>
<td>Column theoretical plate number</td>
</tr>
<tr>
<td>$[Q]$</td>
<td>$\mu$mol $\cdot$ g$^{-1}$</td>
<td>Concentration of the adsorbed solute</td>
</tr>
<tr>
<td>$[S_{0}]_i$</td>
<td>$\mu$mol $\cdot$ g$^{-1}$</td>
<td>The $i$th adsorbent/solute monolayer saturation capacity</td>
</tr>
<tr>
<td>$t$</td>
<td>min</td>
<td>Time</td>
</tr>
<tr>
<td>$t_0$</td>
<td>min</td>
<td>Unretained compound void time</td>
</tr>
<tr>
<td>$t_R$</td>
<td>min</td>
<td>Solute retention time</td>
</tr>
<tr>
<td>$u$</td>
<td>cm $\cdot$ min$^{-1}$</td>
<td>Mobile phase linear velocity in the $z$ direction</td>
</tr>
<tr>
<td>$V_M$</td>
<td>mL</td>
<td>Volume of mobile phase</td>
</tr>
<tr>
<td>$v_g$</td>
<td>mL $\cdot$ cm$^{-1}$</td>
<td>Volume of mobile phase per unit column length</td>
</tr>
<tr>
<td>$V_N$</td>
<td>mL</td>
<td>Specific retention volume</td>
</tr>
<tr>
<td>$z$</td>
<td>cm</td>
<td>Linear position of solute, measured from the column inlet</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>g $\cdot$ mL$^{-1}$</td>
<td>Phase ratio expressed as stationary phase mass per mobile phase volume</td>
</tr>
</tbody>
</table>
The ability to predict the fate of a drug substance following its contact with a material while in a solution of biological medium is only possible by understanding the adsorption heterogeneity of the material with respect to the drug of interest. A model relating the isotherm to the band profile may be derived by consideration of the migration of a single analyte moving through a packed chromatography column, driven by the laminar flow of a mobile phase and reversibly adsorbing and desorbing to the surface of the stationary phase. As the analyte band passes through a column cross section of thickness $dz$ at a time between $t$ and $t + dt$ a mass balance may be formulated. The advection and diffusion contributions to the flux into the cross section are given by,

$$\text{flux in} = [C] u dt + D \left( \frac{\partial [C]}{\partial z} \right) dt$$

(12)

On the opposite side of the cross section, the flux out is,

$$\text{flux out} = \left\{ [C] + \left[ \frac{\partial [C]}{\partial z} \right] dz \right\} \left\{ u + \frac{\partial u}{\partial z} dz \right\} dt + D \frac{\partial}{\partial z} \left\{ [C] + \left[ \frac{\partial [C]}{\partial z} \right] dz \right\} dt$$

(13)

The amount of analyte accumulated in the region is,

$$\text{net accumulation} = \left\{ \left( \frac{\partial [C]}{\partial t} \right) + \left( \frac{m_g}{v_g} \right) \right\} dz dt$$

(14)
Setting the combination of (12) and (13) equal to (14), dividing through by dt and dz
yields, upon neglecting the $dz^2$ term,

$$u \left( \frac{\partial [C]}{\partial z} \right) + \left( \frac{\partial [C]}{\partial t} \right) + \frac{m_g}{v_g} \left( \frac{\partial [Q]}{\partial t} \right) = D_a \left( \frac{\partial^2 [C]}{\partial z^2} \right)$$

(15)

Since $m_g = \frac{M_S}{L}$ and $v_g = \frac{V_M}{L}$ and $\Phi = \frac{M_S}{V_M},$

$$u \left( \frac{\partial [C]}{\partial z} \right) + \left( \frac{\partial [C]}{\partial t} \right) + \Phi \left( \frac{\partial [Q]}{\partial t} \right) = D_a \left( \frac{\partial^2 [C]}{\partial z^2} \right)$$

(16)

Inherent to the derivation of this mass balance equation are the assumptions that 1) the mobile
phase velocity is constant throughout the length of the column and during the time of the
experiment, 2) that the stationary phase is distributed evenly throughout the length of the
column, and 3) that the mobile phase does not compete with the analyte for adsorption. Further
assumptions must be made to use (16) (with appropriate boundary conditions) to model the
chromatographic process. These include 1) that the normal chromatographic band broadenin
processes are strongly dominated by axial dispersion (i.e., adsorption and desorption are fast and
the system exists in a state of equilibrium), 2) that the system is maintained at constant
temperature, 3) that the effect of column pressure on the adsorption/desorption process may be
neglected, 4) that the effect of mobile phase viscosity on the shape of the band profile may be
neglected, and 5) that the shape and finite width of the injection profile (which deviates from the boundary conditions) may be neglected.

The parameters of the adsorption isotherm model determined in the optimization serve to characterize the adsorption heterogeneity of the system. The multi-Langmuir equation,

\[
[Q] = \sum_{i=1}^{n} \frac{K_{eq,i}[S_0]_i[C]}{1 + K_{eq,i}[C]}
\]  

(17)

provides the Langmuir terms, where \(K_{eq,i}\) is the adsorption equilibrium constant and \([S_0]_i\) is the monolayer saturation capacity of the surface for the \(i^{th}\) type of adsorption site. The concentration of adsorbate on the surface is given by \([Q]\), and is the summation of \(n\) independent Langmuir adsorption sites from \(i = 1\) through \(i = n\). If a surface’s adsorption sites can be described as energetically homogenous, a single Langmuir equation describes the surface. For systems with two types of sites, a bi-Langmuir model yields independent Langmuir terms for each type of site.

This approach to measuring the isotherm follows Guiochon et al.\textsuperscript{48} and others.\textsuperscript{46} After calculating the ECP isotherm, the tentative isotherm is fit to the Langmuir model (17). Using (16) as our chromatographic model, a band profile was simulated. The simulated band profile was then compared to the experimental band profile and the difference between the two was quantified by the least-squares method. To achieve these results, a Nelder-Mead optimization process was used until the optimum fit was found where the simulated band profile matched the experimental band profile.

Chromatographic modeling entails solving the partial differential equation (16), and this is historically difficult because of the need to resolve the discontinuity where the shock front
intersects the tailing rear portion of the band profile (in the case where the isotherm is concave toward the concentration axis). A simplified model may be obtained by setting the apparent diffusion coefficient, \( D \), equal to zero, which was solved for non-linear chromatographic data by Valentin.\(^{49}\) Later, Rouchon\(^{50}\) modeled gas chromatography data by implementing the Godunov algorithm,\(^{51}\) a finite difference scheme. Rouchon found that the normal band-broadening processes could be included in the simulation by choosing the space increment that matched the measured column theoretical plate dimensions and the time increment that provided numerical stability. The small error produced by using the first order Taylor series approximations at each iteration (rather than a more accurate higher order approximation for the differential) propagated through the calculations, which effectively modeled the band broadening effect of finite column efficiency on a simulated band profile.

Numerical Algorithm

Native ChemStation data was converted after each experimental run to a .csv (comma separated variable format) via a custom script and saved to the sequence folder. If an injection artifact was present, a custom Python script was used to remove the artifact by subtracting the chromatogram from a blank injection and removing any residual artifact.

The experimental band profile was read into the program. Also loaded was a file containing the measured experimental parameters including the following:

- void time, \( t_0 \),
- flow rate, \( F \), measured at the detector outlet with a graduated cylinder and stopwatch.
Stationary phase mass, $M_s$,

Column theoretical plates, $N$

Column length, $L$

Linear Calibration Curve Parameters

If the baseline was negative, the entire band profile was shifted in the positive direction by the amount of the negative deviation. Next, the detector calibration parameters were used to convert the band profile from units of detector response to concentration (as a function of retention time).

To obtain the initial estimate of the adsorption isotherm parameters, the Elution by Characteristic Points (ECP) method was used to process the band profile. The tail region of the band profile from the peak maxima to the peak baseline was identified and converted to specific retention volume corrected for the void time,

$$V_N = (t_R - t_0) \frac{F}{M_S}$$

(18)

Next, the isotherm was calculated according to the method of Elution by Characteristic Points\textsuperscript{52}. 

37
Inherent in this method is the assumption that column efficiency is sufficiently high that the shape of the elution profile is determined solely by the shape of the isotherm. Continuing from (16) with the diffusion coefficient set to zero,

\[ u \left( \frac{\partial [C]}{\partial z} \right) + \left( \frac{\partial [C]}{\partial t} \right) + \Phi \left( \frac{\partial [Q]}{\partial t} \right) = 0 \]  

(19)

Since \([C]\) is a function both of \(z\) and \(t\),

\[ d[C] = \left( \frac{\partial [C]}{\partial z} \right) dz + \left( \frac{\partial [C]}{\partial t} \right) dt \]

(20)

At constant \([C]\), (20) may be rearranged to,

\[ \left( \frac{\partial [C]}{\partial z} \right) = - \left( \frac{\partial [C]}{\partial t} \right) \left( \frac{\partial t}{\partial z} \right) \]

(21)

Combining (19) and (21) and rearranging,

\[ \int_{t_0}^{t_R} dt = \int_{0}^{L} \frac{1}{u} \left\{ 1 + \Phi \frac{d[Q]}{d[C]} \right\} \]

(22)
After integrating,

\[ t_R = \frac{L}{u} \left\{ 1 + \Phi \frac{d[Q]}{d[C]} \right\} \]  

(23)

Since \( \frac{L}{u} = t_0 \) and \( \Phi = \frac{M_S}{(F \cdot t_0)} \)

\[ [Q][C] = \int_0^C V_N d[C] \]  

(24)

Another useful relationship may be derived from (23). Consider the Langmuir equation,

\[ [Q] = \frac{K_{eq}[S_0][C]}{1 + K_{eq}[C]} \]  

(25)

Taking the first derivative with respect to concentration,

\[ \frac{d[Q]}{d[C]} = \frac{K_{eq}[S_0]}{1 + 2K_{eq}[C] + K_{eq}^2[C]^2} \]  

(26)

Now, in the limit as \( C \) goes to zero,
\[
\lim_{[C] \to 0} \frac{d[Q]}{d[C]} = K_{eq}[S_0]
\]

(27)

Combining this result with (23) and rearranging,

\[
t_R = t_0\left(1 + \Phi K_{eq}[S_0]\right)
\]

(28)

Following calculation of the approximate isotherm using the ECP method, the values \(K_{eq}\) and \([S_0]\) are estimated. The lowest concentration point in the band profile (at the end of the tail) corresponds to the highest energy point on the isotherm. The value of that point is used to estimate the \(K_{eq}[S_0]\) parameter using (28). The highest concentration point calculated from the ECP isotherm is taken as an estimate of the monolayer capacity, \([S_0]\). Then, using the earlier estimated value of \(K_{eq}[S_0]\), the value of \(K_{eq}\) is calculated. These values of \(K_{eq}\) and \([S_0]\) are used as starting points for the optimization.

For the data reported in this paper, either a single Langmuir or bi-Langmuir model was used. For the cases when the experimental results did not accurately fit a single Langmuir model, a bi-Langmuir was then tried. The starting points for the bi-Langmuir parameters were set by simply dividing the single Langmuir \([S_0]\) term by two and setting both \([S_0]_1\) and \([S_0]_2\) to this value. \(K_{eq,1}\) and \(K_{eq,2}\) were both set to the same value.
All calculations were done using Python and extensive use was made of the Scientific Python libraries. The non-oscillatory Akima cubic spline was used from the Scientific Python Interpolation Module for interpolation in several stages of the algorithm. Because preliminary investigations using just Nelder-Mead optimization resulted in obviously local minima, the following two-stage optimization strategy was utilized:

First, optimization toward the minima was done using the Nelder-Mead algorithm from the Scientific Python optimization package. After each time the Nelder-Mead algorithm resolved, the optimizer would then iterate through a grid of nearby values in the problem domain within a specified range built around the Nelder-Mead solution (a total of $15^2$ points for a single Langmuir optimization or $5^4$ for a bi-Langmuir). Each of these points were evaluated by the function and the lowest score was then used as the starting point for the next Nelder-Mead optimization. If the optimization routine resolved to the same solution five times, that point was deemed as the solution. For each iteration, the evaluated function would load in the assumed Langmuir or bi-Langmuir parameter values and experimental parameters for the column and analyte, then generate a simulated band profile using the Godunov algorithm. The simulated band profile was then compared against the experimental data by interpolating to a common timeline and tabulating the sum of squares of the residuals between the two curves which was returned to the optimizer. In the event that negative Langmuir parameters were handed to the function, a punitively high sum of squares value was simply returned and no simulation was performed.

The full Python source code can be found in APPENDIX: SOURCE CODE.
Experimental

Reagents & Materials

LC-grade methanol and water were purchased from Fisher Scientific (Hampton, NH). 10x concentrated phosphate buffered saline (without magnesium or calcium) was also purchased from Fisher Scientific and was diluted with 18.2 MΩ deionized water from a Milli-Q water purification system made by Millipore (Zug, Switzerland) and sterile filtered before use. Acetaminophen, diclofenac sodium, and verapamil HCl were purchased from Sigma-Aldrich (St. Louis, MO). Empty stainless-steel chromatography columns were purchased from Restek (Bellefonte, PA). PMMA and PDMS microspheres were purchased from Cospheric (Santa Barbara, CA). Spheres of 304 stainless steel (325 mesh) and of known dimensions approximating those of the stationary phase particles were purchased from Atlantic Equipment Engineers (Upper Saddle River, NJ).

Instrumentation

Experimental data was acquired using a Hewlett-Packard HP1050 liquid chromatograph. The instrument modules included a four channel delivery system, a UV-Vis diode array detector (DAD), an autosampler with a 100 µL sample loop and 21-vial carousel, and ChemStation data collection software (Rev. A.10.02). The UV-Vis DAD sampled at a rate of 20 Hz. The extra-column volume from the autosampler to the detector cell was measured to be 0.116 mL. A 10 mL graduate cylinder was used to collect the eluate and timed with a stopwatch to confirm the accuracy of the mobile phase flow rate.
Material Columns

Microspheres of PDMS and PMMA were purchased from Cospheric and were used to craft material columns for ILC measurements. PDMS microspheres were silica microspheres coated with dimethylpolysiloxane (Cospheric: SiO2MS-DS 2-19μm - 20mL). PMMA microspheres were homogenous material (Cospheric: PMMAMS-1.2 125-150μm - 5g). Microspheres were imaged with a JEOL JSM-6480 Scanning Electron Microscope (SEM) and analyzed with ImageJ to generate a histogram of particle sizes which was fit to a Weibull Distribution. The PDMS microspheres had a mean particle diameter of 7.7 μm and a dispersion of 2.4 μm and the PMMA microspheres had a mean particle diameter of 122.2 μm and a dispersion of 15.1 μm (Figure 12). The expected surface area per gram was calculated as 3.9x10⁻¹ m² · g⁻¹ for the PDMS microspheres and 4.2x10⁻² m² · g⁻¹ for the PMMA microspheres.
Figure 12. Representative SEM images of the PMMA (A) and PDMS (B) microspheres. Histograms and fitted Weibull distributions (C & D) describing the particle size distribution of the microspheres.

The PDMS and PMMA microspheres were packed into columns measuring 30 x 3.2 mm and 250 x 4.6 mm, respectively. The material columns were dry loaded then tamped down with water at a flow rate of 5 mL · min⁻¹ and over 100 bar pressure while submerged in a sonicator to agitate the microspheres. This process was reiterated until the microspheres were flush with the column end and the injection of a void compound yielded a sufficiently symmetric peak. The tailing factor for PMMA columns was 1.03 – 1.08 and PDMS columns was 1.40 – 1.67, using
the $S = W_{0.05h}/2f$ method, where $S$ is the tailing factor, $W_{0.05h}$ is the full-width at 5% of the peak height, and $f$ is the interval between the peak front and the peak maximum. It was difficult to obtain symmetric peaks from the PDMS packed columns, which may affect the accuracy of results. Theoretical plates were low relative to commercial columns, with plate numbers for PMMA columns at 598 – 723 and PDMS columns at 145 – 196, using the $N = 5.54 \times (t_R/W_{0.5h})^2$ method, where $N$ is the theoretical column plate number, $t_R$ is the retention time, and $W_{0.5h}$ is the full-width at half of the peak height. While the plate numbers and symmetry for the PDMS columns were less than ideal, likely a result of the polydispersity of the packing material, it was the best available material and was deemed sufficient to provide ILC measurements with enough accuracy for single-component adsorption isotherms.

Experimental

Calibration curves for the UV-Vis DAD were made using a conventional C18 analytical column (Kinetex 2.6 µm EVO C18 with a pore size of 100 Å (50 x 4.6 mm), manufactured by Phenomenex).

Drug concentrations and injection volumes used are listed in Table 1. In order to overload the columns to study the non-linear responses the maximum concentrations possible were used, constrained by the drug solubility limit in PBS and the UV-Vis detector saturation limit. The UV spectrum of each drug was first confirmed using a Varian Cary 300 BIO UV-Vis spectrometer. For highly UV-absorbing compounds such as acetaminophen, a wavelength with a low molar absorptivity was chosen to prevent detector saturation.
Table 2. Drug Information. Properties cited from Wishart, et al. \(^{28}\) except where otherwise noted.

<table>
<thead>
<tr>
<th>Compound</th>
<th>Acetaminophen</th>
<th>Diclofenac</th>
<th>Verapamil</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviation</td>
<td>APAP</td>
<td>DCF</td>
<td>VER</td>
</tr>
<tr>
<td>CAS</td>
<td>103-90-2</td>
<td>15307-79-6</td>
<td>52-53-9</td>
</tr>
<tr>
<td>MW (g/mol)</td>
<td>151.163</td>
<td>318.129</td>
<td>454.69</td>
</tr>
<tr>
<td>log P</td>
<td>0.91</td>
<td>4.51</td>
<td>3.79</td>
</tr>
<tr>
<td>log D</td>
<td>0.41(^{56})</td>
<td>1.1(^{57})</td>
<td>2.7(^{58})</td>
</tr>
<tr>
<td>Water Solubility (mg/mL)</td>
<td>4.15</td>
<td>0.00237</td>
<td>0.00447</td>
</tr>
<tr>
<td>Physiological Charge</td>
<td>0</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>Polar Surface Area (Å²)</td>
<td>49.33</td>
<td>49.33</td>
<td>63.95</td>
</tr>
<tr>
<td>Polarizability (Å³)</td>
<td>15.52</td>
<td>27.93</td>
<td>51.7</td>
</tr>
</tbody>
</table>

**Experimental:**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Concentration (M)</td>
<td>3E-2</td>
<td>2E-4</td>
<td>2E-4</td>
</tr>
<tr>
<td>Injection Volume (µL)</td>
<td>10</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>λ (nm)</td>
<td>297</td>
<td>276</td>
<td>229</td>
</tr>
</tbody>
</table>

Typical injection volume was 10 µL, but for drugs with low solubility in PBS the volume was increased to 100 µL. Injections were done in quintuplicate. Void measurements were also taken in quintuplicate prior to each experimental run and used to measure the void time and calculate the column’s number of theoretical plates as inputs for the numerical simulation.
The elution was done isocratically with a flow rate of 1 mL · min⁻¹, except for experiments with verapamil on PDMS which was done at a flow rate of 3 mL · min⁻¹. The mobile phase was phosphate buffered saline (PBS), as an analogue for cell culture media as it has the same pH and ionic strength which are the main factors of an aqueous mobile phase that can influence adsorption.

Results

Example experimental elution curves and the simulated band profiles are shown in Figure 13. The shapes of the experimental curves exhibit the expected shock front and rear tailing, indicative of an analyte interacting with a Langmuir-like adsorption isotherm curve and overloading the column.
Figure 13. Example experimental band profiles and fitting of simulated band profiles of drugs eluting through PMMA and PDMS columns. Verapamil on PDMS would not elute within a reasonable time or with sufficient peak height to be quantified. Scaled to best display the shape of the band profiles and simulated curve fit.

Simulated curves were generally in agreement with the experimental elution curves, except for the single Langmuir fitting to verapamil on PMMA for which the tail of the simulated peak was higher than observed. For the systems reported here, bi-Langmuir modeling did not yield better least-square residuals compared to the single Langmuir model. Since the bi-Langmuir model didn’t appear to have more explanatory power than a single Langmuir model, the single Langmuir model was deemed sufficient for the system analysis. Fitting with a single Langmuir model suggests that all adsorption sites fall within a narrow band of adsorption energies.
The elution of verapamil on the PDMS column was not possible without an organic modifier as no peak was detected even after multiple hours. An indirect analysis of verapamil interacting with PDMS was facilitated by utilizing methanol as a mobile phase modifier and the results are described in the next chapter.

Following the approach of Liu, the Gibbs free energy of adsorption was calculated from the adsorption equilibrium constant

\[ \Delta G^\circ = -RT \ln \left( \frac{K_{eq}}{\gamma_e} \right) \text{ (1 mol} \cdot \text{L}^{-1}) \]

(29)

where \( \Delta G^\circ \) is the standard Gibbs free energy, \( R \) is the gas constant, \( T \) is the temperature in Kelvin, \( K_{eq} \) is the adsorption equilibrium constant, and \( \gamma_e \) is the analyte activity coefficient at equilibrium. This activity coefficient is unity for neutral analytes, but can be fully described by the Debye-Hückel equation for activity coefficients in solutions with ionic solutes:

\[ \ln \gamma_e = -\frac{z^2 q^2 \kappa}{8\pi \epsilon_r \epsilon_0 k_B T} = -\frac{z^2 q^3 N_A^{1/2}}{4\pi (\epsilon_r \epsilon_0 k_B T)^{3/2}} \sqrt{\frac{I_e}{2}} = -Az^{2}I_{e}^{1/2} \]

(30)

where \( q \) is the elementary charge, \( \kappa \) is the inverse Debye screening length, \( \epsilon_r \) is the relative permittivity of the solvent (in this case water), \( \epsilon_0 \) is the permittivity of free space, \( k_B \) is the Boltzmann constant, \( T \) is the temperature in Kelvin, \( N_A \) is Avogadro’s number, \( z \) is the charge of the analyte, and \( I_e \) is the ionic strength of the solution. The constants in the equation can be condensed down to the combined term \( A \), which is dependent on temperature. For water
at 25 °C, \( A = 1.172 \text{ mol}^{-1/2} \cdot \text{kg}^{1/2} \). The ionic strength of the solution \( (I_e) \) can be simply calculated with

\[
I_e = \frac{1}{2} \sum_{i=1}^{n} b_i z_i^2
\]

(31)

where \( b_i \) is the molality of the \( i^{\text{th}} \) solute and \( z_i \) is the charge of the \( i^{\text{th}} \) solute. For phosphate buffered saline, the ionic strength is 165.45 mMolal.

At pH 7.4, acetaminophen has a charge of 0, diclofenac has a charge of -1, and verapamil has a charge of +1.\(^{28}\) Taking the equilibrium constants from \textbf{Table 3} and combining with (29) resulted in the energies of adsorption shown in \textbf{Table 3}. These energy values are in agreement with the general range for physisorption between 2.1 to 20.9 kJ \cdot \text{mol}^{-1} and lower than chemisorption within the general range of 80 to 200 kJ \cdot \text{mol}^{-1}.\(^{59}\) Measurements at different temperatures to further derive the enthalpy and entropy of adsorption will be reported in a future manuscript.
Table 3. Langmuir Terms for Drug Adsorption on Polymeric Materials

<table>
<thead>
<tr>
<th>Material</th>
<th>Drug</th>
<th>Saturation Capacity ([S_0]) mol · m(^{-2})</th>
<th>Equilibrium Constant (K_{eq}) L · mol(^{-1})</th>
<th>Energy of Adsorption (kJ \cdot mol^{-1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMMA</td>
<td>APAP</td>
<td>1.99E-7 (± 1.05E-7)</td>
<td>5.89E+3 (± 2.10E+3)</td>
<td>-21.5 (± 0.9)</td>
</tr>
<tr>
<td></td>
<td>DCF</td>
<td>2.54E-7 (± 1.61E-8)</td>
<td>1.99E+4 (± 1.99E+3)</td>
<td>-25.7 (± 0.2)</td>
</tr>
<tr>
<td></td>
<td>VER</td>
<td>2.53E-7 (± 2.58E-8)</td>
<td>8.52E+4 (± 4.93E+4)</td>
<td>-29.3 (± 1.6)</td>
</tr>
<tr>
<td>PDMS</td>
<td>APAP</td>
<td>6.70E-5 (± 1.84E-5)</td>
<td>3.38E+1 (± 9.17E+0)</td>
<td>-8.7 (± 0.7)</td>
</tr>
<tr>
<td></td>
<td>DCF</td>
<td>8.48E-6 (± 1.79E-6)</td>
<td>2.30E+4 (± 3.38E+3)</td>
<td>-26.1 (± 0.4)</td>
</tr>
</tbody>
</table>

The Langmuir parameters in Table 3 were used to build Figure 14, which depicts the Langmuir adsorption isotherms for the drugs on PMMA and PDMS. Comparing between materials, the adsorption isotherm of each drug was multiple orders of magnitude lower on PMMA than on PDMS. When comparing between drugs, the energy of adsorption correlated with the distribution coefficient \((\log D, \text{pH } 7.4)\) which represents the relative hydrophobicity of a drug as measured by the distribution of both ionized and unionized states of a drug between equal volumes of octanol and water at pH 7.4. This is a well characterized value during drug discovery as it is predictive of the partition of a drug between blood plasma (an aqueous medium) and cell membranes (a nonpolar medium).\(^6^0\) Other groups have published correlations of drug concentration losses to PDMS via absorption with either the polar surface area\(^6^1\) or partition coefficient \((\log P)\)\(^6^2\), but neither of these correlate with the adsorption energies measured here (refer to Table 2). It should be noted that neither the distribution coefficient nor the partition coefficient account for the nature of the adsorbent, but only the analyte.
**Figure 14.** Langmuir adsorption isotherms for drugs on PMMA and PDMS. The horizontal axis shows the mobile phase concentration at equilibrium. For reference, drug concentrations in human-on-a-chip systems are typically in the nM (1x10⁻⁹) to µM (1x10⁻⁶) region. The vertical axis shows the concentration on material surfaces at equilibrium. PMMA (dashed lines) show less adsorption of drugs to the material surfaces than PDMS (solid lines). Drugs are organized by color, with acetaminophen (APAP) in red, diclofenac (DCF) in green, and verapamil (VER) in blue. The adsorption isotherm depicting verapamil adsorption to PDMS uses Langmuir terms derived from an extrapolation of changes in the Langmuir terms with respect to methanol concentration.
CHAPTER THREE: METHANOL AS A MOBILE PHASE MODIFIER

Content in this chapter is reprinted with permission from Schneppe et al., Characterization of Drug-Polymer Adsorption Isotherms in Body-on-a-Chip Systems by Inverse Liquid-Solid Chromatography. *ACS Biomaterials Science & Engineering*. ©2020 American Chemical Society.¹

Introduction

It is well known in chromatography that adding an organic modifier such as methanol or acetonitrile to an aqueous mobile phase helps to elute non-polar analytes from reverse-phase columns.⁶³⁻⁶⁴ Previous work by Gritti and Guijochn ⁶⁵ investigated the effect of methanol across a range of water/methanol mixtures on the adsorption isotherms of phenol eluting through a C18-silica column. They were able to track changes to the saturation capacity and energy of adsorption as a function of methanol in the mobile phase. We utilized a similar approach and measured the band profiles of drugs in multiple concentrations of methanol in the PBS mobile phase, then made a linear regression to extrapolate the adsorption parameters of strongly-adsorbing drugs in a methanol-free system.

Experimental

Methanol was added as an organic modifier to the mobile phase in increments of 5% v/v from 0% v/v to 40% v/v in PDMS columns and 0% v/v to 20% v/v in PMMA columns. The
same experimental methods as discussed in the prior chapter were used as the basis for the work discussed in this chapter.

Results

Verapamil adsorption isotherms on PDMS columns were measured with methanol organic modifier between 15% v/v through 40% v/v in 5% v/v increments. The resulting band profiles are shown in Figure 15. As methanol concentration increased the verapamil eluted earlier and the band profile became sharper, with a faster drop-off of the desorption tail. 15% v/v methanol was the lowest methanol concentration condition able to elute verapamil on PDMS, as the detector response was near the lower limit and each injection consumed 900 mL of mobile phase.
Figure 15. Changes to the band profile of verapamil on PDMS for four dosages of methanol in PBS (15 to 40% v/v in 5% v/v increments). Associated simulated curves also shown for both Langmuir and bi-Langmuir adsorption isotherm models. Horizontal axis represents time in minutes, vertical axis represents micromolar concentration. Note the wide difference of the peak elution time between 15% v/v methanol at 80 minutes and 40% v/v methanol at 1 minute.

In order to validate the treatment used to extrapolate the methanol-free parameters for verapamil, the same procedure was applied to the other drugs so that the extrapolated methanol-free parameters could be compared to those measured in an actual methanol-free condition. This treatment was applied to acetaminophen and diclofenac on PDMS, as well as all three drugs on PMMA. The full range of 0% v/v through 40% v/v was used for acetaminophen and diclofenac on PDMS as both drugs could elute within reasonable times under the methanol-free condition. Only 0% v/v through 20% v/v methanol was used for experiments with PMMA, capping at 20% v/v methanol to avoid damaging the material. All sequences started at the lowest methanol
concentration and incremented by 5% v/v to test each concentration. Columns were equilibrated with a minimum of 10 column volumes before testing each concentration.

The changes in the saturation capacity \([S_0]\) and equilibrium constant \(K_{eq}\) were charted against the concentration of methanol in the mobile phase as shown in Figure 5. When charted on a semi-log scale, the trendlines appeared linear with respect to methanol concentration with the exception of verapamil, which appeared to have a linear trend between 15% v/v to 30% v/v but nonlinear between 30% v/v to 40% v/v.

**Figure 16.** Changes in the saturation capacities and adsorption equilibrium constants with respect to methanol concentration. Acetaminophen shown in red, diclofenac in green, and verapamil in blue. PDMS shown with solid lines, PMMA shown with dashed lines. Error bars denote standard error of the mean.
It is known that non-polar organosilane surface coatings such as C18 will swell in the presence of organic solvents like methanol as a result of increased chain solvation, thus increasing the available surface area presented to an adsorbate.\textsuperscript{65} This may explain the increase in saturation capacity with higher methanol concentration observed for diclofenac and verapamil. The monolayer saturation capacity decreased with increasing methanol concentration for acetaminophen. This may be related to competition for the adsorption sites between methanol and the relatively hydrophilic acetaminophen.

Solvophobic theory suggests that repulsive interactions between water and hydrophobic solutes drives adsorption to nonpolar stationary phases.\textsuperscript{66} This explains the observed decrease in adsorption equilibrium constants of diclofenac and verapamil, both hydrophobic drugs, as methanol concentration increased in the mobile phase. The equilibrium constant for acetaminophen adsorbing to PDMS remained low compared to other drug-material combinations except for a slight increase at the highest methanol concentrations tested. The equilibrium constant for acetaminophen adsorbing to PMMA increased with increasing methanol concentration. As acetaminophen is more soluble in methanol and water-methanol mixtures than pure water,\textsuperscript{67-68} solvophobic theory would predict a decreasing adsorption equilibrium constant with increasing methanol concentration. This deviation from the trend observed with the other drugs warrants further investigation. It should be noted that despite the high adsorption equilibrium constant for acetaminophen to PMMA, the drug was minimally retained and eluted close to the void time, perhaps due to the low monolayer saturation capacity.

A linear regression was applied to the changes in saturation capacity and adsorption equilibrium constant for verapamil on PDMS to extrapolate Langmuir parameters for the 0\% v/v
methanol condition, reflective of the use-case for body-on-a-chip systems. Since the data was nonlinear at higher methanol concentrations, only the segment between 15-30% v/v methanol was selected for the linear regression. The same fitting method was applied to acetaminophen and diclofenac, and these results are shown in Figure 17. There was general agreement between the extrapolated values and the experimentally measured values under the methanol-free condition, as shown in Table 4.
Figure 17. Linear regression of the logarithm of the saturation capacity (log([S0])) and logarithm of the adsorption equilibrium constant (log(Keq)) versus methanol concentration for acetaminophen, diclofenac, and verapamil eluting against PDMS. Linear regressions were based upon the 15-30% v/v methanol concentrations and had good agreement with the measured values for acetaminophen and diclofenac at 0% v/v methanol.
Table 4. Measured & Extrapolated Langmuir Terms in the Methanol-free Condition

<table>
<thead>
<tr>
<th></th>
<th>Saturation Capacity $[S_0]$, mol · m$^{-2}$</th>
<th>Equilibrium Constant $K_{eq}$, L · mol$^{-1}$</th>
<th>Energy of Adsorption kJ · mol$^{-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Measured</td>
<td>Extrapolated</td>
<td>Measured</td>
</tr>
<tr>
<td>APAP</td>
<td>$6.70E$-$5$</td>
<td>($±1.84E$-$5$)</td>
<td>$3.38E$+$1$</td>
</tr>
<tr>
<td>DCF</td>
<td>$8.48E$-$6$</td>
<td>($±1.79E$-$6$)</td>
<td>$2.30E$+$4$</td>
</tr>
<tr>
<td>VER</td>
<td>$1.13E$-$6$</td>
<td>($±7.80E$-$8$, $-7.29E$-$8$)</td>
<td>$6.96E$+$7$</td>
</tr>
</tbody>
</table>

The changing adsorption behavior of these drugs in the presence of increasing concentrations of methanol can also be utilized to understand how to remove the drugs from these material surfaces, especially to prevent the carryover of sequestered drugs between experiments should the PDMS devices be used for multiple *in vitro* experiments. Since organic solvents lower the affinity between strongly adsorbing drugs such as verapamil to PDMS, any cleaning protocol should include an extraction step with an organic solvent before reusing PDMS-based devices.

**Discussion**

The utilization of Inverse Liquid-Solid Chromatography enables a quantitative method of determining both hydrophilic and especially hydrophobic molecular interactions with polymer materials commonly used in body-on-a-chip systems. This report details both theoretical and experimental verification of the results.
From a theoretical standpoint, if a sufficiently dilute band migrates through a column, each molecule may be viewed as randomly walking from each adsorption site to the next\(^\text{69}\) so that the retention time is determined by all types of sites. Often, it is the case that the stationary phase is composed of a large number of lower adsorption energy sites and a smaller number of higher energy sites. This is the case even in modern commercially available columns where the organic bonded phase makes up the low energy sites and the uncovered silica surface provides the few high energy sites.\(^\text{70}\) Likewise in affinity chromatography and in chiral separations, the low energy sites correspond to non-specific interactions and the high energy to specific interactions. As eluite concentration increases, there is a greater tendency for high energy sites to be already occupied relative to the low energy sites as additional eluite arrives in the vicinity of a site. The eluite molecules present in a higher concentration band therefore pass through the column faster. This tendency of increasing linear velocity with concentration continues to increase as a non-linear function determined by the derivative of the adsorption isotherm. The higher concentrations cannot pass the lower concentrations since this would result in a concentration having multiple values of velocity (an impossibility)\(^\text{71}\) and so they all pile up in a discontinuity (shock front) where the concentration instantly falls from high concentration to baseline.\(^\text{72}\)

The Equilibrium-Diffusive Model of Chromatography\(^\text{40}\) forms the basis of the chromatographic simulations. This model assumes that the normal kinetic band broadening functions are dominated by axial and eddy diffusion in the mobile phase and that these effects may be lumped into an apparent diffusion coefficient. In order to ensure that non-equilibrium effects may be neglected, the method first suggested by Huber\(^\text{73}\) was used, wherein
chromatograms of a range of concentrations are superimposed to determine whether or not their tails lie on the same curve. This quality assurance procedure is related to the concentration-dependent nature of non-equilibrium effects. An example of this test was shown in Figure 11.

Several well-known techniques have been described for determining the adsorption isotherm from HPLC data and the Frontal Analysis method has been regarded as the most accurate method. For this work, the Inverse Method work flow was used because the time required to collect sufficient data points to construct an accurate Frontal Analysis isotherm is excessive. The Inverse Method work flow is much less costly in terms of solvent and analyte as well, which is important when working with limited amounts of analyte such as for novel drugs.

The Langmuir model of adsorption was selected due to its predominance as the adsorption isotherm present in liquid-solid adsorption systems. The Langmuir adsorption model selected was validated by the shock-front and trailing tail present in the observed band profiles (Figure 13 and Figure 15). Other isotherm types, such as anti-Langmuir and S-shaped isotherms, would have resulted in a sloped front and a sharply cut-off rear tail.

In order to provide an accurate determination of both parameters of the Langmuir or bi-Langmuir isotherm, sufficient material must be injected to produce a non-linear band profile. One complication of loading a large concentration was having to account for the relatively large peak area in the initial space/time increment of the simulation. Ideally, the entire area would be placed in the first increment (a Dirac pulse), however, if the resulting concentration was greater than the solubility of the analyte, the excess area would be moved to the second time increment.
Another difficulty related to the sample size was the detector signal saturation limit. Since the HPLC system was old and few spare parts were available, a preparative flow cell with a shorter optical path was not available to compensate for detector signal saturation. By recording the chromatograms in regions of the absorption spectrum where the molar absorptivity was lower, the need to use a non-linear equation to relate the peak area to the concentration injected was avoided.

The column phase ratio, \( \Phi \), defined as the mass of stationary phase per unit mobile phase volume was a necessary parameter required for the chromatographic simulation. Spheres of 304 stainless steel (325 mesh) and of known dimensions approximating those of the stationary phase particles were purchased from Atlantic Equipment Engineers (Upper Saddle River, NJ). The density of both the stainless-steel spheres and the stationary phase material was determined by weighing a graduated cylinder filled with the respective material. A column packed with the stainless-steel spheres was weighed and the volume of packing calculated to be used as a point of reference. The mobile phase volume of the steel reference column and the material sample columns were calculated using the empty column dimensions, the measured void-time, and the measured flowrate. As both columns were made using the same empty column and the total column volume is the sum of the mobile phase volume and stationary phase column,

\[
V_{M,U} + V_{S,U} = V_{M,K} + V_{S,K}
\]

Where \( V_{M,U} \) and \( V_{M,K} \) are the volumes of the mobile phases of the known (stainless steel) and unknown columns and \( V_{S,U} \) and \( V_{S,K} \) are the volumes of the stationary phases. Rearranging
and replacing the respective mobile phase volumes with their flowrate multiplied with void time yields,

\[
V_{S,U} = F_K t_{0,K} + V_{S,K} - F_U t_{0,U}
\]

(33)

The mass of the unknown stationary phase may then be calculated from its previously determined density. This technique worked well because the mass of the stainless-steel packing was large enough to provide a significant weight difference to the tared empty column (relative to the polymeric packing material).

The void time and flowrate at the column outlet were measured on the same day the overloaded band profile was recorded for transformation to the isotherm. Likewise, because in-house packed columns of PDMS or PMMA material may be less stable than commercially packed columns, the column efficiency was measured before each condition. Ammonium nitrate was injected and the absorbance was measured at 240 nm to determine the column efficiency using the equation,

\[
N = 5.54 \left( \frac{t_R}{(t_b - t_a)} \right)^2
\]

(34)

attributed to Giddings,\textsuperscript{75} where \(t_b\) and \(t_a\) are the retention times of the front and back (respectively) at 50% of the maxima. The retention time, \(t_R\) was simply set to be the peak maxima. As these columns were manually packed in-house, the elution bands for the void compound was not perfectly symmetric, especially in the PDMS columns, which could be
indicative of a small void-volume at the column head. It is possible this affected the experimental band profiles and was not accounted for in the simulated band profiles.

While PBS functioned as a mobile phase analogue for cell culture media with respect to pH and ionic strength, it also limited the range of drugs that could be directly studied. One problem with the mobile phase is that the phosphate and chloride ions conjugate with basic drugs and lower their effective solubility. Since the observation of non-linear effects depends on overloading the material column, this solubility upper-bound capped the quantity of analyte that could be delivered in one injection to overload the column. This limitation was partially compensated for by increasing the injection volume to deliver a larger quantity of analyte. However, it should be noted that in all cases the concentration delivered was much larger than that evaluated within body-on-a-chip systems.

Another issue was that some drugs could not be eluted effectively without an organic modifier. Verapamil was strongly retained and could not be eluted before kinetic band broadening processes reduced its peak concentration below the detection limit. To estimate the thermodynamic parameters of verapamil adsorption in a methanol-free condition, an extrapolation was performed. The same fitting method was applied to acetaminophen and diclofenac, compounds that could be eluted by the PBS mobile phase. There was general agreement between extrapolated values versus the experimentally measured values in the methanol-free condition.

Since the PDMS packing was silica spheres coated with a film of PDMS, one would expect that there could be some higher energy surface silanol sites that were not covered by the PDMS film. Silanol groups are unreacted surface sites leftover from the alkylsilane surface
modification of the silica support bed, can act as ion exchange sites, and contribute to band tailing on reverse phase columns.\textsuperscript{77-78} We expected to find these high-energy silanol groups on the PDMS-coated silica microspheres but the simulated bands were fit adequately with a single Langmuir model (rather than a bi-Langmuir) with an energy of adsorption on the scale of physisorption well below that of a hydrogen-bond with silanol. We did not expect verapamil to interact with these silanol groups as it does not have any hydrogen-bond donor groups, but both acetaminophen and diclofenac have two hydrogen-bond donor groups that could have interacted with a surface silanol group. It’s common to add salts to the mobile phase to suppress band tailing from silanol groups\textsuperscript{78} and the mobile phase we used, phosphate buffered saline, has a very high salt concentration that could have acted to suppress access to any silanol groups present.

Conclusion

We have developed an adaptation of Inverse Liquid-Solid Chromatography and paired it with a Godunov numerical simulation and a Langmuir model of adsorption to characterize the adsorption isotherms of drugs to materials used in body-on-a-chip systems. We also developed a method to indirectly measure the Langmuir terms of a strongly adsorbing drug by studying the changes to the equilibrium constant and monolayer saturation capacity with respect to methanol as an organic modifier and extrapolating to a methanol-free condition.
CHAPTER FOUR: ORGANOSILANE SURFACE MODIFICATIONS

Content in this chapter is reprinted with permission from Schnepper et al., Inverse Liquid Solid Chromatography to Evaluate Drug Interactions with Organosilane Modified Polydimethylsiloxane for use in Body-on-a-Chip Systems. Biotechnology Progress. ©2020 American Institute of Chemical Engineers.²

Introduction

Modifications to surfaces to prevent adsorption date back over 2000 years to the ancient Phoenicians and Carthaginians, who used pitch and lead sheathing to prevent the fouling of ship hulls with barnacles and degradation by sea worm holes,⁷⁹ making a sacrificial layer that would be sloughed off as the ship travelled and needed reapplication in dry dock. Medieval and renaissance mariners would apply mixtures of pitch and tar which were known colloquially as “black stuff” or, when mixed with sulfur, “brown stuff”. In the late 18th century, the English navy began adding copper to the bottoms of their ships, which creates a toxic layer of oxychloride as the copper dissolves away slowly – and was so effective for antifouling that the term copper-bottomed came to mean ‘reliable and low-risk.’ When shipbuilders began building iron-hulled and steel-hulled vessels, copper was no longer used because of its galvanic degradation of the iron, thus antifouling paints became the standard. These antifouling paints have varied in active ingredients through the decades as technology and environmental concern has advanced, including biocides like tetrabutyltin, copper sulfate, and non-stick coatings of fluoropolymers or silicones.
Techniques for antifouling of ship hulls apply directly to microfluidics, where the same paradigms apply for creating antifouling surfaces. In the literature,\textsuperscript{80} methods to discourage adsorption and absorption to PDMS include cladding the surfaces with waxes, adding ‘inert’ sacrificial proteins into solutions to saturate the wall adsorption sites, or grafting with a nonstick polymer layer. Direct modifications to PDMS include exposure to energy sources such as oxygen plasma, UV irradiation, or corona discharge, and the creation of self-assembled monolayers (SAM) with short-chain alkylsilanes. Additionally, replacing PDMS with other materials like polystyrene or PDMS-like fluoropolymers have been investigated.\textsuperscript{14}

A general survey of structures & properties of protein adsorption resistant surfaces by Whitesides\textsuperscript{81} found four aspects that appeared necessary for effective passivation: (a) a hydrophilic surface, (b) with hydrogen bond acceptors, (c) but no hydrogen bond donors, and (d) with a neutral charge. This list matches the suppositions stated in the introduction of this dissertation regarding a surface that would be resistant to drug adsorption.

Early attempts at preventing non-specific adsorption to PDMS surfaces originate with microfluidic ELISA (enzyme-linked immunosorbent assay) designs, in which bovine serum albumen (BSA) was used to block non-specific binding,\textsuperscript{82} also known as ‘pre-fouling’ the surface.\textsuperscript{83} The use of a blocking agent for ELISA appeared early in the life of the technique in 1972,\textsuperscript{84} a year after the first paper describing ELISA was published, with the introduction of the surfactant Tween 20 (polyoxyethylene (20) sorbitan monolaurate), a PEGylated sorbitan esterified with a fatty acid. The addition of BSA, casein, and gelatin were later added to formulations to try to enhance the blocking of non-specific binding in the assay microtiter plates,\textsuperscript{85-87} with mixed results at blocking binding to the polystyrene. Pre-fouling with BSA has
been demonstrated as beneficial for microfluidic devices in decreasing non-specific adsorption of proteins to surfaces,\textsuperscript{18, 88} but may not block the adsorption of small molecules such as drugs.\textsuperscript{83}

Energy-based modifications of PDMS include UV irradiation,\textsuperscript{89} corona discharge,\textsuperscript{90} and exposure to RF-generated plasmas.\textsuperscript{91-92} These methods change the chemistry of the PDMS surface through various mechanisms inducing: chain scissions,\textsuperscript{89} adding functional groups such as hydroxyls, carbonyls, carboxyls, amines, and amides,\textsuperscript{91-92} or through the vitrification of the top 10 nm of the PDMS surface into a silica-like material.\textsuperscript{90} Invariably, energy-induced modifications are transient and the PDMS recovers its hydrophobic surface on a timescale of hours to days. Multiple mechanisms are responsible for hydrophobic recovery: low molecular weight species migrating to the surface, rotation of surface hydrophilic groups into the bulk around sigma bonds, modified chains subsiding into the bulk polymer and away from the surface, and reactions between water and hydrophilic surface groups. Water and elevated temperatures appear to drive the hydrophobic recovery mechanisms, which perfectly describes the inside of a body-on-a-chip system.
Figure 18. Illustrated mechanisms of hydrophobic recovery of modified PDMS. Reconstruction (left) is one mechanism that reverses energy-based and organosilane modifications of the surface. Rotation of functional groups around the polymer backbone, condensation reactions between surface hydroxyl groups, and the diffusion of low molar mass chains to the surface are also mechanisms for hydrophobic recovery. Left figure reprinted with permission from Keeffe et al., Suppressing Surface Reconstruction of Superhydrophobic PDMS Using a Superhydrophilic Zwitterionic Polymer. Biomacromolecules. ©2012 American Chemical Society.93 Right figure reprinted with permission from Yeh et al., Modification of Silicone Elastomer with Zwitterionic Silane for Durable Antifouling Properties. Langmuir. ©2014 American Chemical Society.94

Deposition of films into PDMS-based microfluidic devices is a popular means to passivate the material surfaces. Vapor deposition of parylene (AKA poly(p-xylene)) is an established method for passivating medical implants, first investigated in the late 1980s for coating long-term neuronal electrodes95 and used more recently on PDMS for lab-on-a-chip applications such as miniaturized PCR chips to prevent the adsorption of DNA and proteins or to prevent the absorption of hydrophobic dyes used in fluorescence-based assays.96-97 Parylene has even been used to coat a PDMS-based subdural multi-electrode array used in vivo.98 Polymerization-based passivation methods all require an assembled device before coating, which is not compatible with the device design paradigm used by this lab (bottom-up design).
Additionally, PDMS will not stick to a parylene coated glass substrate, necessitating assembly before adding the parylene coating and prohibiting repeated uses of the device.

The Hybrid Systems Lab specializes in alkylsilane surface chemistry for controlled, patterned cell adhesion to silicon and silicon oxide materials. This group uses three alkylsilanes which have been published in the literature, one cytophillic for cell adhesion and two cytophobic for cell repulsion. The cytophillic alkylsilane, (3-trimethoxysilylpropyl)diethylenetriamine (DETA, purchased from United Chem, T2910-KG, CAS 0035141-30-1), presents a triamine functional group which exhibits a positively charged and hydrophilic surface that readily adsorbs proteins assisting with cell adhesion. The cytophobic alkylsilanes used are a perfluorinated silane, (tridecafluoro-1,1,2,2-tetrahydrooctyl)trichlorosilane (13F, purchased from Gelest, SIT8174.0, CAS 78560-45-9), and a polyethylene glycol silane, 2-[methoxy(polyethyleneoxy)6-9propyl]trimethoxysilane (PEG, purchased from Gelest, SIM6492.7, CAS 65994-07-2), which discourage cell adhesion with either a strongly hydrophobic (13F) or strongly hydrophilic surface (PEG). Applied to PDMS surfaces, this would have the advantage of modifying on a per-part basis without the need of preassembly of the system before passivation.

Alkylsilane surface modification of PDMS has been explored for use in microfluidics. Sharma, et al. investigated the use of PEG-silane in comparison to oxygen plasma treatment on PDMS to increase the wettability of the surface. The contact angle of a sessile drop of 15 MΩ water onto the surfaces was used to track the wettability of PEGylated and plasma treated PDMS over time, demonstrating a recovery of hydrophobicity in plasma treated PDMS within an hour but a retention of hydrophilicity in PEGylated PDMS for the duration of their experiments.
However, the extent of their application was limited to fluid velocity in channels – there was no exploration of the antifouling properties of the PEGylated PDMS surfaces.

PEGylation of PDMS to passivate surfaces was explored by Demming, et al. insofar as the adhesion of cells and oil droplets only. A hydrophobic yeast, *Saccharomyces cerevisiae* DSM 2155, was used to test the adhesion of cells on the surface and showed a decrease of adhering cells onto a PDMS substrate in a PMMA microfluidic channel to 2.5 ± 2.5% normalized to the control. The formation of oil droplets on a chip was also demonstrated with the aim for applications in the creation of stable emulsifications of active pharmaceutical compounds. This aspect of the paper was merely a demonstration of stable oil droplet formation with a PEGylated substrate and was not explored further.

Similar to energy exposure methods, modifications using PEG-silane suffer from hydrophobic recovery over time. Demming, et al. demonstrated that PDMS, when modified with PEG-silane, initially presented a contact angle under 10° (water, sessile drop method) - though it was speculated this was due to unbound PEG-silane remaining after “inadequate rinsing”. The samples were stored in four conditions (air, water, sodium chloride salt, and ethanol), and some conditions showed the surface partially recovering hydrophobic character within a day to around 50° (water, sodium chloride salt, and ethanol) and further to near full recovery of hydrophobicity at 5 months to around 100-110° (air, sodium chloride salt). Even when the PDMS substrates were pre-treated by soaking in toluene to remove low molecular weight species that might be migrating to the surface, the recovery of hydrophobicity still occurred. This suggested that the modified PDMS chains were either migrating or rotating the PEG side chains into the bulk of the polymer, as illustrated by Figure 18.
In 2014, Yeh, et al. published their work regarding modified PDMS surfaces with a Zwitterionic silane, sulfobetaine silane (SBSi), synthesized in their lab from (N,N-Dimethylaminopropyl)trimethoxysilane (DMASi) and 1,2-propane sultone. This modification was shown to resist adsorption of bacteria (“[Repelling rates for] P. aeruginosa and S. epidermidis reached 99.86% and 99.78% respectively, with respect to bare PDMS” quantified by LIVE/DEAD BacLight), protein (“98% and 97% for BSA and mucin” quantified by ELISA), and lipids. The water contact angle was also shown to be stable when stored in air with contact angles below 20° maintained as far as 5000 hours (approx. 7 months).

The majority of papers found developing antifouling surface modifications for PDMS addressed cells, protein, and DNA. A few addressed small molecules such as fluorescent dyes and lipids. None directly addressed drugs.

In this study, we utilized ILC to characterize the effect of organosilane surface modifications on the adsorption isotherms of drugs. As the main driver behind adsorption is suspected to be the hydrophobic interaction, drugs were selected which represent a range of hydrophobicity as measured by the dispersion coefficient (log D, pH 7.4): acetaminophen (log D = 0.41), diclofenac (log D = 1.1), and verapamil (log D = 2.7). To test the effect of organosilanes on drug adsorption, a hydrophilic organosilane (polyethylene glycol, PEG) and a hydrophobic organosilane (fluoropolymer, 13F) were selected in an attempt to respectively decrease and increase the energy of adsorption relative to native PDMS.
Experimental

Reagents & Materials

LC-grade methanol, LC-grade water, and 10x Phosphate Buffered Saline (without calcium or magnesium) were purchased from Fisher Scientific (Hampton, NH). 10x PBS was diluted to 1x PBS concentration with 18.2 MΩ water from a Milli-Q water purification system made by Millipore (Zug, Switzerland) and sterile filtered with a 0.22 µm filter before use. Organosilanes used for surface modifications were 2-[methoxy(polyethyleneoxy)₆-9propyl]trimethoxysilane (PEG) purchased from Gelest (Morrisville, PA) and trichloro(1H,1H,2H,2H-perfluorooctyl)silane (13F) purchased from Millipore Sigma (St. Louis, MO). Toluene was purchased from Fisher Scientific (Hampton, NH) and distilled shortly before use. Hydrochloric acid (concentrated, 37 wt %) was purchased from Fisher Scientific (Hampton, NH). Acetaminophen, diclofenac sodium, and verapamil HCl were purchased from Sigma-Aldrich (St. Louis, MO). Empty stainless-steel chromatography columns were purchased from Restek (Bellefonte, PA). PDMS-coated silica microspheres were purchased from Cospheric (Santa Barbara, CA).

Organosilane modification of microspheres

To activate and modify the surface of the microspheres with a self-assembled monolayer of organosilane, a protocol was adapted from previously published protocols for the surface modification and patterning of glass coverslips and MEMS devices for use in body-on-a-chip systems. To activate the microsphere surfaces, approximately 10 g of microspheres were
added to a 250 mL glass bottle containing 1:1 methanol : concentrated hydrochloric acid and stirred for 30 minutes. The microspheres were collected with a Buchner funnel and washed with 500 mL of water. The recovered microspheres were then added to a vessel containing water and boiled for 60 minutes. The activated microspheres were then dried for 2 hours in a 100 °C oven.

To modify the activated microspheres with 13F, 1.2 mL of 13F-silane was added to 600 mL of freshly distilled toluene, mixed, and added to the bottle containing the activated microspheres. The mixture was left to react while stirring for 30 minutes. The modified microspheres were recovered with a Buchner funnel and washed with 500 mL of toluene. The microspheres were transferred into a 20 mL scintillation vial and left to dry in a 70 °C oven for 1 hour.

To modify the activated microspheres with PEG, 1.2 mL of PEG-silane and 0.3 mL of concentrated hydrochloric acid were added to 600 mL of freshly distilled toluene, mixed, and decanted into the bottle containing the activated microspheres. The mixture was left to react while stirring for 30 minutes. The modified microspheres were recovered with a Buchner funnel and washed with 500 mL of toluene, 500 mL of ethanol, then 500 mL of water. The microspheres were transferred into a 20 mL scintillation vial and left to dry in a 70 °C oven for 1 hour. The modified microspheres were then used as chromatography packing materials for the ILC columns.
Instrumentation

Chromatograms were collected using a Hewlett-Packard HP1050 liquid chromatograph. Installed modules included a multisolvent delivery system, a UV-Visible diode array detector (DAD), an autosampler with a 100 µL sample loop and 21-vial carousel, and ChemStation data collection software (Rev. A.10.02). The UV-Vis DAD sampled at a rate of 20 Hz. Extra-column volume from the autosampler needle seat to the detector cell was 0.116 mL, measured by injecting an unretained compound through a zero-volume union. The flowrate accuracy was checked by using a 10 mL graduated cylinder and stopwatch to collect the eluate. Calibration curves were made using a conventional C18 analytical column manufactured by Phenomenex (Kinetex 2.6 µm EVO C18 with a pore size of 100 Å, 50 x 4.6 mm).

X-ray Photoelectron Spectrometry (XPS) data was collected using a FIONS 220i XPS with an aluminum anode, spot size of 250 x 1000 µm, monoXPS lens mode, and a pass energy of 100 eV for survey and 25 eV for high resolution scans. Charge compensation was provided by a low-energy flood gun. Data was collected and analyzed with Thermo Avantage software (v4.36 Build 02611).

Contact angle measurements were performed with a Ramé-Hart Model 250 Goniometer. 5 µL of LC-grade water was dispensed onto the sample surface as a sessile drop. Image analysis was performed by DROPimage Advanced software (v1.4.11).
Packed Columns

Microspheres of PDMS-coated silica were first modified with organosilanes or used directly as native material to craft packed columns for ILC experiments. The microspheres were imaged on a JOEL JSM-6480 SEM and analyzed with ImageJ to create a histogram of microsphere diameters which was fit to a Weibull distribution. The fitted Weibull distribution had a shape parameter $k = 2.6 \, \mu m$ (spread of particle sizes) and a scale parameter $\lambda = 7.7 \, \mu m$ (mean particle size), which was used to calculate the expected surface area per gram as $0.39 \, m^2 \cdot g^{-1}$ for the microspheres.

Columns were packed with the microspheres for ILC experiments as follows: empty stainless-steel columns measuring 30 x 3.2 mm (length x inner diameter) were loaded with microspheres and then tamped down with water at a flow rate of 5 mL · min$^{-1}$ and at least 100 bar pressure while submerged in a sonicator bath. The process was repeated until the packed bed of microspheres was flush with the column inlet and the band profile of a void compound had a sufficiently symmetric peak shape, defined as a tailing factor $S < 1.80$, a plate number $N > 100$, and without any peak shoulders or footing which would indicate insufficient packing. The tailing factor ($S$) was measured to validate the quality of the column packing, and was $1.40 – 1.67$ for PDMS columns, $1.54 – 1.73$ for 13F-PDMS columns, and $1.46 – 1.69$ for PEG-PDMS columns, as calculated using the $S = W_{0.05h}/2f$ method where $W_{0.05h}$ is the full-width at 5% of the peak maxima, and $f$ is the interval between the peak maxima and the peak front at 5% of the peak maxima. Theoretical plates ($N$) for these columns were between $145 – 196$ for PDMS columns, $126 – 181$ for 13F-PDMS columns, and $114 – 208$ for PEG-PDMS columns, using the $N = 5.54 \times \left( t_R/W_{0.5h} \right)^2$ method where $t_R$ is the retention time and $W_{0.5h}$ is the full-width at half of
the peak maxima. Average column permeability was 3.5E-4 cm² for PDMS columns, 4.7E-4 cm² for 13F-PDMS columns, and 3.5E-4 cm² for PEG-PDMS columns as calculated by the Darcy equation: 

\[ B = \left( \frac{FL\Delta \pi}{\eta r^2} \right) \]

where \( B \) is the column permeability in cm², \( F \) is the mobile phase flowrate in cm³ · s⁻¹, \( \eta \) is the mobile phase viscosity in Poise, \( L \) is the column length in cm, \( r \) is the column internal radius in cm, and \( \Delta \pi \) is the pressure drop across the column relative to a union and expressed in Barye. Column outlet flowrate, stationary phase mass, void time, column temperature, and column theoretical plates number were also measured during experimental runs, and the values were used in the numerical algorithm.

Inverse Liquid Chromatography

An HP1050 HPLC was adapted to perform Inverse Liquid Solid Chromatography (ILC) on the packed columns to measure the adsorption isotherms for each drug-material combination. Each column was probed with an injected bolus of drug with sufficient concentration to overload the column, necessary for accurate determination of the Langmuir parameters. Acetaminophen was injected as a 10 µL volume at 3E-2 M. Due to limited solubility in PBS, diclofenac and verapamil were each injected as 100 µL volumes at 2E-4 M to maximize the amount delivered to overload the column. Measurements were collected in quintuplicate. Before each experimental condition, a 10 µL injection of 1.7E-2 M ammonium nitrate was recorded in quintuplicate to establish the void time and theoretical plate number of the column. Each drug-material combination was tested on 3 different packed columns.
The isocratic mobile phase flowrate was 1 mL · min⁻¹ for acetaminophen and diclofenac experiments and 3 mL · min⁻¹ for verapamil experiments. The mobile phase was sterile filtered 1x PBS, mixed with between 0 – 40% v/v methanol at 5% v/v increments. The UV-Vis diode array detector was set to record at these wavelengths for each analyte: acetaminophen (297 nm), diclofenac (276 nm), verapamil (229 nm), ammonium nitrate (240 nm).

Numerical Algorithm

The experimental band profile collected by the ILC instrument was loaded into the program for analysis. Other experimental parameters were also loaded, including: the column length, theoretical plate number, stationary phase mass, the adsorbate concentration and injection volume, the detector calibration curve parameters, the mobile phase flow rate, and the void time.

The first estimate of the Langmuir terms was calculated using the Elution by Characteristic Points method. This method uses the band profile resulting from a single concentrated injection of adsorbate and makes the assumption that smaller peaks would lie on the tail of the larger curve. Thus, the retention volumes for multiple concentrations could be measured simultaneously. This first-pass was used to seed the numerical optimization.

The numerical optimization takes the Langmuir terms and generates a simulated band profile based upon the adsorbate and column parameters entered into the program. The simulated band profile was then compared to the experimental band profile with a least-squares method to calculate the residuals. The Nelder-Mead algorithm was used to adjust the Langmuir terms iteratively until convergence to an optimized result. The surrounding domain was also checked.
within a specified radius to ensure the Nelder-Mead did not settle on a local minimum. This process was repeated until the same solution was found five times, within a specified threshold for convergence.

Data & Results

Organosilane modification of the PDMS-coated silica microspheres was confirmed by XPS, as shown in Figure 19. Microsphere samples were adhered to double-sided carbon tape attached to a piece of a silicon wafer. A sample of only carbon tape was referenced as background. Modification with 13F-silane was confirmed by the presence of a F1s peak at 685 - 688 eV and the addition of higher-energy C1s peaks associated with CF₂ and CF₃ at 291.5 eV and 293.6 eV, respectively. Modification with PEG-silane was confirmed by the presence of an additional higher-energy C1s peak associated with C-O at 286.4 eV. A peak at 281.6 eV was present in all microsphere samples, attributed to silicon carbide as a contaminant from the manufacturing process of the underlying silica support. As the ILC method only interacts with the surface, the presence of the carbide should have negligible effect on the ILC results. Si2p indicated the presence of the silica support at 103.7 eV and the silicone coating at 101.5 eV, and the carbide contaminant at ~99 eV (Si2p³/₂ peak positions reported). Since the sampling depth of XPS is approximately 10 nm and the silica peak from the silica bed substrate is still detectable in all samples, we can conclude that the PDMS and organosilane thicknesses should be less than 10 nm. Contact angle measurements were performed to determine the relative hydrophobicity of the native PDMS and the organosilane modifications, measured with a 5 µL sessile droplet of LC-
grade water. Contact angle on native PDMS was $111.7^\circ \pm 5.9$. Contact angles on glass coverslips coated with 13F and PEG were $114.0^\circ \pm 2.4$ and $40.9^\circ \pm 1.2$, respectively.
Figure 19. XPS spectra of native PDMS-coated microspheres (top), 13F-modified PDMS-coated microspheres (middle), and PEG-modified PDMS-coated microspheres (bottom). Presence of the 13F-modification was evident by the addition of a F1s peak at 685 – 688 eV and fluorinated carbon peaks in the C1s region associated with CF2 and CF3 at 291.5 and 293.6 eV. Presence of the PEG-modification was evident by the addition of a C-O peak at 286.4 eV in the C1s region.

Representative examples of the experimental and simulated band profiles are shown in Figure 20. The band profiles have a shock front and an extended tailing rear, indicative of an analyte overloading the column which is necessary to access the non-linear regions of the adsorption isotherm curve. The triangular band profile is characteristic of a Type I Langmuir adsorption isotherm,\textsuperscript{40, 47} validating the choice of this model of adsorption. Simulated band profiles were in good agreement with the experimental band profiles.
**Figure 20.** Representative examples of elution band profiles, comparing between materials: PDMS (red), 13F-PDMS (green), and PEG-PDMS (blue). Experimental band profiles (solid lines) and simulated band profiles (dashed lines) were in good agreement. Scales selected to best show the peak shapes. (A) Acetaminophen, 10 µL injection, λ = 297 nm, 1 mL · min⁻¹ flowrate, 0% v/v methanol. (B) Diclofenac, 100 µL injection, λ = 276 nm, 1 mL · min⁻¹ flowrate, 0% v/v methanol. (C) Verapamil, 100 µL injection, λ = 229 nm, 3 mL · min⁻¹ flowrate, 15% v/v methanol. Drugs were most retained by native PDMS.

As verapamil did not elute within a reasonable time when phosphate buffered saline was used as the mobile phase, an indirect method was utilized to extrapolate the Langmuir terms by measuring the change in saturation capacity [S₀] and the adsorption equilibrium constant K_{eq} in response to graded additions of methanol in the mobile phase. This approach follows that of
Gritti and Guiochon, who tracked changes in the saturation capacity and adsorption equilibrium constant of phenol adsorbing to a C18 column with respect to methanol concentration in a mixed methanol/water mobile phase.

Methanol concentrations from 0% v/v to 40% v/v in 5% v/v increments were tested for all drug-material combinations. However, the lowest methanol concentration that could elute verapamil was 15% v/v, as the peak height approached the detector lower limit and each chromatographic run consumed 900 mL of mobile phase. A linear regression was applied to the logarithms of both the measured saturation capacities and the adsorption equilibrium constants with respect to methanol concentration to extrapolate the Langmuir terms for the methanol-free condition, as shown in Figure 21. Changes in the Langmuir terms of verapamil with respect to methanol concentration appeared linear between 15 – 30% v/v but nonlinear between 30% - 40% v/v, so the linear regression was based on the 15 – 30% v/v methanol set to extrapolate Langmuir terms for the methanol-free condition for verapamil. To validate the method, the same extrapolation process was applied to ILC measurements of diclofenac and acetaminophen and compared to the Langmuir terms measured from the methanol-free condition, also shown in Figure 20. A comparison between extrapolated and measured Langmuir parameters for each adsorbent material and drug combination is listed in Table 5.
Figure 21. Semi-log plots for the monolayer saturation capacity \([S_0]\) and adsorption equilibrium constant \(K_{eq}\) with respect to the percentage of methanol present in the mobile phase. Linear regressions are based only on the 15 – 30% v/v methanol measurements. Each regression line represents one packed column (\(n = 3\) for each drug-material combination). The extrapolated values for 0% v/v methanol tended to agree with the measured values.
### Table 5. Measured & Extrapolated Langmuir Adsorption Isotherm Terms and Energies

<table>
<thead>
<tr>
<th></th>
<th>Measured</th>
<th></th>
<th>Extrapolated</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Saturation Capacity $[S_0]$</td>
<td>Equilibrium Constant $K_{eq}$</td>
<td>Energy of Adsorption</td>
<td>Saturation Capacity $[S_0]$</td>
</tr>
<tr>
<td></td>
<td>mol · m$^{-2}$</td>
<td>L · mol$^{-1}$</td>
<td>kJ · mol$^{-1}$</td>
<td>mol · m$^{-2}$</td>
</tr>
<tr>
<td>Acetaminophen</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDMS</td>
<td>6.70E-5</td>
<td>3.38E+1</td>
<td>-8.6</td>
<td>6.91E-5</td>
</tr>
<tr>
<td>13F-</td>
<td>4.34E-5</td>
<td>4.52E+1</td>
<td>-9.4</td>
<td>4.04E-5</td>
</tr>
<tr>
<td>PEG-</td>
<td>5.92E-5</td>
<td>3.55E+1</td>
<td>-8.8</td>
<td>9.49E-5</td>
</tr>
<tr>
<td>PDMS</td>
<td>(± 2.17E-5)</td>
<td>(± 1.08E+1)</td>
<td>(± 0.8)</td>
<td>(± 2.07E-5)</td>
</tr>
<tr>
<td>13F-</td>
<td>(± 1.85E-6)</td>
<td>(± 7.28E-1)</td>
<td>(± 0.0)</td>
<td>(± 8.85E-6)</td>
</tr>
<tr>
<td>PEG-</td>
<td>(± 7.58E-6)</td>
<td>(± 7.64E+0)</td>
<td>(± 0.6)</td>
<td>(± 4.05E-5)</td>
</tr>
<tr>
<td>PDMS</td>
<td>8.48E-6</td>
<td>2.30E+4</td>
<td>-26.0</td>
<td>6.10E-6</td>
</tr>
<tr>
<td>PEG-</td>
<td>(± 2.06E-6)</td>
<td>(± 3.84E+3)</td>
<td>(± 0.4)</td>
<td>(± 1.52E-6)</td>
</tr>
<tr>
<td>PDMS</td>
<td>3.39E-6</td>
<td>4.24E+4</td>
<td>-27.6</td>
<td>2.80E-6</td>
</tr>
<tr>
<td>13F-</td>
<td>(± 4.54E-7)</td>
<td>(± 3.64E+3)</td>
<td>(± 0.2)</td>
<td>(± 7.66E-7)</td>
</tr>
<tr>
<td>PEG-</td>
<td>3.91E-6</td>
<td>3.95E+4</td>
<td>-27.4</td>
<td>2.05E-6</td>
</tr>
<tr>
<td>PDMS</td>
<td>(± 1.47E-6)</td>
<td>(± 2.53E+3)</td>
<td>(± 0.2)</td>
<td>(± 2.95E-7)</td>
</tr>
<tr>
<td>PEG-</td>
<td>1.14E-6</td>
<td>7.71E+7</td>
<td>-45.9</td>
<td>(± 1.49E-7)</td>
</tr>
<tr>
<td>PDMS</td>
<td>(± 2.20E-6)</td>
<td>(± 2.22E+6)</td>
<td>(± 0.8)</td>
<td></td>
</tr>
<tr>
<td>13F-</td>
<td>7.87E-6</td>
<td>6.40E+6</td>
<td>-39.9</td>
<td></td>
</tr>
<tr>
<td>PEG-</td>
<td>4.97E-6</td>
<td>2.63E+6</td>
<td>-37.8</td>
<td></td>
</tr>
<tr>
<td>PDMS</td>
<td>(± 7.60E-7)</td>
<td>(± 2.47E+5)</td>
<td>(± 0.2)</td>
<td></td>
</tr>
</tbody>
</table>
The Langmuir parameters determined for each material and drug combination listed in Table 5 were used to build the adsorption isotherms shown in Figure 22. Generally, the saturation capacities and adsorption equilibrium constants between materials were similar, within an order of magnitude. However, the extrapolated adsorption equilibrium constant for verapamil to PDMS was an order of magnitude higher than the organosilane-modified materials.

Figure 22. Langmuir adsorption isotherms for acetaminophen, diclofenac, and verapamil versus native PDMS-coated silica microspheres (red), 13F-modified PDMS-coated silica microspheres (green), and PEG-modified PDMS-coated microspheres (blue). Isotherms for acetaminophen and diclofenac were based solely on the measured values at 0% v/v methanol in the mobile phase. Isotherms for verapamil were based on the extrapolated terms as determined by the linear regression of measurements made between 15 – 30% v/v methanol.
The adsorption equilibrium constants were converted to the standard Gibbs free energy of adsorption using the approach published by Liu (2009):

\[ \Delta G^\circ = -RT \ln \left[ \frac{K_{eq}}{\gamma_e} \left(1 \text{ mol} \cdot \text{L}^{-1}\right) \right] \]

(35)

where \( \Delta G^\circ \) is the standard Gibbs free energy, \( R \) is the gas constant, \( T \) is the temperature in Kelvin, \( K_{eq} \) is the adsorption equilibrium constant, and \( \gamma_e \) is the adsorbate activity coefficient at equilibrium. The activity coefficient can be calculated with the Debye-Hückel equation for activity coefficients in solutions with ionic solutes:

\[ \ln \gamma_e = -\frac{z^2q^2\kappa}{8\pi\varepsilon\varepsilon_0k_B T} = -\frac{z^2q^3N_A^{1/2}}{4\pi(\varepsilon\varepsilon_0k_BT)^{3/2}} \frac{I_e}{2} = -Az^2I_e^{1/2} \]

(36)

where \( q \) is the elementary charge, \( \kappa \) is the inverse Debye screening length, \( \varepsilon_\tau \) is the relative permittivity of the solvent (water), \( \varepsilon_0 \) is the permittivity of free space, \( k_B \) is the Boltzmann constant, \( T \) is the temperature in Kelvin, \( N_A \) is Avogadro’s number, \( z \) is the charge of the adsorbate, and \( I_e \) is the ionic strength of the solution which is 165.45 mM for phosphate buffered saline. At pH 7.4, acetaminophen has a neutral charge, diclofenac has a -1 charge, and verapamil has a +1 charge.

Energies of adsorption remained within or close to the level of physisorption (2.1 to 20.9 kJ · mol\(^{-1}\)) except for verapamil which was higher, though all energies of adsorption remained
below the range expected for chemisorption (80 to 200 kJ \cdot \text{mol}^{-1}). All measurements were performed at the same temperature. Measurements will be conducted at different temperatures to determine the enthalpy and entropy of the adsorption and reported in a future publication.

As a hydrophobic interaction was thought to be the main adsorption process between the drugs and the materials tested, the energies of adsorption were expected to increase in correlation with the drugs’ dispersion coefficient (log D, pH 7.4) a measurement of a drugs’ hydrophobicity which is quantified by measuring the relative distribution of both ionized and unionized forms of a drug between equal volumes of octanol and water at pH 7.4. This is a well-characterized property of drugs and drug candidates, as it predicts the partition of drug between blood (hydrophilic) and cell membranes (lipophilic). For each drug, it was also expected that the energies of adsorption would increase in correlation with the hydrophobicity of the stationary phase surface, with the smallest energy to PEG-modified microspheres and the greatest to 13F-modified microspheres, and energy of adsorption to native PDMS microspheres in the middle.

As predicted, energies of adsorption correlated strongly with the drugs’ dispersion coefficient. As shown in Figure 5, energy of adsorption progressed in increasing magnitude from acetaminophen (log D = 0.41) to diclofenac (log D = 1.1) to verapamil (log D = 2.7). However, the relative energies of adsorption of the drugs to the different stationary phases could not be explained by the simple hydrophobic model. A One-Way ANOVA followed by a Dunnett’s test was applied to analyze the effect of the organosilane surface modifications on the energy of adsorption. Acetaminophen, a relatively hydrophilic drug, showed no significant difference between native PDMS and the modified PDMS stationary phases. Diclofenac had small but significant differences in energies of adsorption between stationary phases, with the
least energy of adsorption to native PDMS, and slightly greater energies of adsorption to 13F-PDMS (p = 0.001) and PEG-PDMS (p = 0.002). This was surprising as diclofenac was less retained by 13F-PDMS and PEG-PDMS than by native PDMS. However, since chromatographic retention is largely determined by the slope of the isotherm at infinite dilution, which for the Langmuir isotherm is equal to the product of the monolayer saturation capacity and the equilibrium constant, we can posit that in the case of diclofenac the retention order was dictated mostly by the relative number of adsorption sites available on the material surface. Verapamil showed the greatest differences in energies of adsorption between stationary phases. The greatest energy of adsorption was to PDMS, was less for adsorption to 13F-PDMS (p < 0.001), and was the least for adsorption to PEG (p < 0.001). The lesser energy of adsorption to PEG-PDMS followed the expectations of a hydrophobic interaction mediating adsorption, as the hydrophilic PEGylated surface should decrease the adsorbate-adsorbent attraction. The adsorption energy of verapamil to 13F-PDMS was expected to be greater than to native PDMS, as 13F is a more hydrophobic surface, but was found to be less.
Figure 23. Effect of the organosilane surface modifications on the energies of adsorption for selected drugs. Error bars denote standard deviation.

Discussion

We have utilized a new technique to understand the dynamic interactions of drugs with organosilane-modified PDMS materials. ILC was paired with a numerical optimization to characterize the adsorption parameters of these well-known drugs to native PDMS and organosilane-modified PDMS. This unique approach allows quantitative measurement of
adsorption parameters to body-on-a-chip materials so that surface modifications can be best implemented to control drug adsorption. This provides an alternative to utilizing alternative polymeric materials optimized for reduced drug adsorption but may have less desirable material properties in terms of fabrication or optical clarity, which are still desirable for body-on-a-chip devices.

Peak symmetry measured for the unretained compound was less than ideal and could have affected the accuracy of the technique. The source of the asymmetry was likely from the polydispersity of the packing material or from settling of the packing that may have resulted in a void at the inlet which would have resulted in kinetic tailing. This problem might be remedied in the future by accounting for the asymmetry as a part of the simulation.

Another source of error may have been in the choice of algorithm used to simulate the band profiles. The Godunov finite difference scheme was used to numerically solve the Equilibrium-Diffusive model of chromatography equation\(^\text{50-51}\) and while the Godunov method is computationally fast it has been reported to be less accurate when applied to multicomponent systems.\(^\text{109}\) We are developing an implementation of the Orthogonal Collation on Finite Elements algorithm,\(^\text{110}\) and future work will include comparing the accuracy of OCFE with the Godunov finite difference scheme for our single component ILC experiments.

According to solvophobic theory, adsorption of a non-polar adsorbate to a non-polar adsorbent in the presence of an aqueous solution is mainly driven by a net repulsion between the adsorbate and water, since the solvent-solvent interactions are much stronger than the adsorbate-solvent interactions.\(^\text{66}\) It would follow that decreasing the polarity of the solution (e.g. adding methanol) would also decrease the net repulsive interaction driving adsorption. In accordance
with the solvophobic theory, we observed a strong correlation between the drugs’ dispersion coefficients (log D) and the measured adsorption energies. Additionally, the adsorption equilibrium constants for diclofenac and verapamil decreased with increasing methanol concentration, also in accordance with solvophobic theory. This appears to be the main factor determining adsorption interactions in the conditions studied.

The hydrophobicity of the stationary phase surface was less of a determining factor for adsorption energies. Modification with the hydrophobic fluoropolymer 13F was expected (based on the solvophobic model) to increase the energy of adsorption of all drugs, which it did only with diclofenac. The 13F modification had no significant effect on acetaminophen and counter to expectations reduced the energy of adsorption of verapamil. The PEG modification was expected to reduce the energy of adsorption for all drugs, and while it did reduce the energy of adsorption greatly for verapamil, PEG was ineffective in changing the adsorption energy for acetaminophen and the energy was slightly greater for diclofenac.

From these observations, we suspect that there may be a secondary effect driving adsorption besides just the solvophobic effect and nonspecific van der Waals forces. Both verapamil and diclofenac have multiple polar functional groups and may therefore have multiple modes of adsorption. Both species are charged at physiological pH as well, verapamil positively and diclofenac negatively. More specific adsorption modes (e.g., by hydrogen bonding) may be present. In our model, we applied a single Langmuir adsorption model to the ILC measurements, which assumes an energetically homogenous surface populated by just one type of adsorption site. Our model may have instead found a statistical average of multiple types of adsorption sites. A bi-Langmuir model (two independent adsorption sites) was explored to test for multiple
adsorption sites but did not reduce the residual relative to the single Langmuir model. So, one must conclude that even if multiple types of adsorption sites are present, they must all fall within a relatively narrow range of adsorption energies.

Silanols were considered as a possible high-energy adsorption site present on the surface that could have interacted with the drugs. Silanols could have been originated either from 1) the silica support bed or 2) since the PDMS surface must be activated with silanols before the organosilanes can covalently attach to the surface, it is possible there were some unreacted silanols present on the modified surfaces. If diclofenac adsorbs through hydrogen bonding to these silanols whereas acetaminophen and verapamil don’t, that might explain why only diclofenac showed a greater energy of adsorption to the organosilane-modified surfaces. However, since the addition of salt to the mobile phase is commonly used to suppress the activity of silanols and we used a PBS as the mobile phase (a high-salinity solution) we expected that any silanols present would have been fully suppressed.

Polyethylene glycol (PEG) is commonly used to passivate surfaces, and the main mode of this passivation is through stearic repulsion. As the PEG chains are solvated by water, any adsorption by an adsorbate to a PEG-coated surface must first overcome the water associated to the chains, which is entropically unfavorable. This explains the smaller adsorption energy for verapamil measured on PEG-modified PDMS microspheres relative to the native PDMS microspheres.

Wettability and hydrogen bonding to fluoropolymers is dictated by the degree of fluorination of the polymer. Electron density in organic fluorine moieties (C-F) are heavily shifted to the fluorine, resulting in depletion of electron density around the bonded carbon. If the
polymer is partially fluorinated (e.g. poly(vinylidene fluoride) (PVDF) or poly(vinyl fluoride) (PVF)), these organic fluorine functional groups produce strong dipoles that shift electron density from the remaining C-H groups. These strong dipoles allow partially fluorinated polymers to participate in van der Waals interactions and form hydrogen bonds.\textsuperscript{113} In contrast, perfluorinated polymers (e.g. poly(tetrafluoroethylene) (PTFE) or polyperfluoroalkyxyethylene (PFA)), which are similar to the organosilane 13F, have very weak polarity and, despite the lone electron pairs on fluorine, typically do not accept hydrogen bonds.\textsuperscript{114} Thus, the lower than expected adsorption energy of verapamil to 13F-modified PDMS microspheres resulted from the weak polarity inherent to perfluorinated polymers, so only a small van der Waals interaction was available for adsorption.

The conclusions we can draw about the material surfaces are limited as we were trying to measure these specific drug-material adsorption isotherms and not fully probe the material surface. The drugs used here have several functional groups so there may be multiple possible nonspecific interactions. Usually in ILC experiments simpler molecules (e.g. with only one functional group) are used as probes so that the known functional group can be used to elucidate the chemical structure of the surface through a specific targeted interaction. Analytes are usually chosen for ILC experiments for which the retention and shape of the band profile is strongly dominated by thermodynamic factors, relative to the normal band-broadening effects. We would like to conduct a more detailed analysis of these organosilane surface modifications in future work to draw further conclusions.

The packing material used in this work was a PDMS-coated silica with a thin layer of PDMS, which would have limited penetration of analyte molecules to the depth of the film.
Besides adsorption, PDMS is known to sequester small molecules by absorption, whereby the small molecule diffuses into the bulk. This process can perhaps still be modeled as an interfacial process, but one where concentration changes are measured over days or even weeks. Because of the slow kinetics, absorption cannot be measured by ILC. ILC is however an important technique that may be used to prevent losses by absorption, since the analyte must adsorb before it can be absorbed. This observation leads one to ask the question, does the absorption process preclude accurate ILC measurements? This seems to not be the case. Loss of analyte would be readily apparent in an ILC experiment as the calibration process requires accounting for the analyte that is injected. The plastic matrix close to the surface is likely easily penetrable by the constant stream of smaller solvent molecules which therefore could extract the retained analyte molecules back into the mobile phase.

Conclusion

Polydimethylsiloxane (PDMS) has many useful material properties for making body-on-a-chip devices, including optical clarity, ease of fabrication, and has FDA approval for numerous medical applications. Drug adsorption is a problematic property of PDMS, but less-adsorbing polymeric materials don’t all have the same engineering properties that made PDMS the material of choice for microfluidics and body-on-a-chip devices. Inverse Liquid-Solid Chromatography (ILC) allows for the characterization of drug adsorption to PDMS. Here, we have applied this methodology to evaluate surface modifications to PDMS that may reduce drug adsorption.
We have demonstrated a method utilizing Inverse Liquid-Solid Chromatography (ILC) to measure differences in the adsorption isotherms of three drugs with respect to organosilane surface modifications of PDMS. ILC measurements found differences in the adsorption equilibrium constants $K_{eq}$ and the saturation capacities $[S_0]$ of three drugs adsorbing to native PDMS microspheres and organosilane-modified PDMS microspheres. The energy of adsorption correlated strongly with the drugs’ dispersion coefficient (log D), a measurement of hydrophobicity. As the hydrophobic interaction was expected to be the main factor determining energies of adsorption between drugs and materials, organosilane modifications that increased (13F) or decreased (PEG) the hydrophobicity of the material were used as modifications. However, it was found that the hydrophobicity of the material used as the stationary phase did not correlate with observed changes to the adsorption energy for a specific drug.

Analysis of the native PDMS and organosilane modified PDMS surfaces will be explored in future publications by additional drugs and non-drug monofunctional probes to elucidate the modes of adsorption. Additional organosilane modifications, such as charged and zwitterionic organosilanes, will also be explored in future work.
CHAPTER FIVE: ABSORPTION MEASUREMENTS FROM BODY-ON-A-CHIP SYSTEMS

Content in this chapter is reprinted with permission from Schnepper et al., Characterization of Drug-Polymer Adsorption Isotherms in Body-on-a-Chip Systems by Inverse Liquid-Solid Chromatography. *ACS Biomaterials Science & Engineering*. ©2020 American Chemical Society.¹ Content in this chapter is reprinted with permission from Schnepper et al., Inverse Liquid Solid Chromatography to Evaluate Drug Interactions with Organosilane Modified Polydimethylsiloxane for use in Body-on-a-Chip Systems. *Biotechnology Progress*. ©2020 American Institute of Chemical Engineers.²

Introduction

Experimental I – Unmodified PDMS

A housing was designed for adsorption testing as shown in Figure 24, and fabricated in a similar manner to housings used previously for drug testing in multi-organ systems by this group.¹¹⁶⁻¹¹⁹ The device was made with 5.5 mm thick PMMA sheets (McMaster Carr) and 0.5 mm thick PDMS sheets (Grace BioLabs) cut to specifications with an IR carbon dioxide laser (Trotec Speedy 400). Two 5 mm radius cylindrical cuts were made in the top PMMA plate to act as reservoirs. The channel was 90 mm long and 10 mm wide with a 5 mm radius of curvature at each end, connecting to the reservoirs in the top PMMA plate. The sidewalls and bottom of the channel were PDMS and the top of the channel was PMMA. Overall, the device held approximately 1.29 mL and had 13.19 cm² of combined surface area, 3.49 cm² of PMMA and 9.7 cm² of PDMS. This gave a surface area to volume ratio of 1020 m⁻¹ (combined), 270 m⁻¹ of PMMA and 750 m⁻¹ of PDMS.
**Figure 24.** Adsorption channel dimensions used. The top was defined by a 5.5 mm thick PMMA plate. The microfluidic channel layer was defined by a 0.5 mm PDMS sheet. Below the channel layer was another PDMS sheet. Below that another PMMA plate for structural integrity. All layers were held together with screws through the device.

Standard in-house cleaning protocols were used to prepare the housing materials for testing. After laser cutting, the PDMS sheets were washed with 70% isopropyl alcohol, left to soak overnight in 70% isopropyl alcohol, rinsed with water, sonicated for 60 minutes in 1%wt tergazyme, rinsed with water, and left to dry. The PMMA plates were sonicated for 60 minutes in 1%wt tergazyme, rinsed with water, and left to air-dry. Before assembly, these parts were sterilized in a biosafety cabinet with 70% isopropyl alcohol and left to dry overnight.
Adsorption was measured by administering 1000 µL of initial concentration of the drug [C$_0$] premixed and dissolved in PBS into the device. The device was left on a rocker platform set to 3° tilt and 1 oscillation per minute to ensure the solution was thoroughly mixed throughout the device. 55 µL samples were taken at time points of 10 min, 1 hour, 4 hours, and 24 hours and then stored frozen at -4 °C until time for analysis. Samples were diluted 1:20 in PBS and quantified by HPLC using a conventional C18 analytical column under isocratic conditions.

Results I

Prediction of the expected drug concentration at equilibrium was calculated using the adsorption parameters determined earlier, the initial concentration, and the surface area in contact with the solution. The equilibrium state is given by:

$$K_{eq} = \frac{[Q]}{[S][C]}$$

(37)

where $K_{eq}$ is the equilibrium constant, [C] is the concentration of the drug in mol · L$^{-1}$, [S] is the concentration of available surface sites in mol · dm$^{-2}$, and [Q] is the concentration of occupied surface sites in mol · dm$^{-2}$. The change in concentration from the initial state to the equilibrium state is given by Table 6:
Table 6. Initial, Change, Equilibrium Table of Adsorption

<table>
<thead>
<tr>
<th></th>
<th>Available Surface Sites</th>
<th>Analyte in Solution</th>
<th>Occupied Surface Sites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change</td>
<td>−x</td>
<td>−x</td>
<td>+x</td>
</tr>
<tr>
<td>Equilibrium</td>
<td>A[S₀] − x</td>
<td>V[C₀] − x</td>
<td>x</td>
</tr>
<tr>
<td>Expressed as Concentration</td>
<td>[S₀] − ( \frac{x}{A} )</td>
<td>[C₀] − ( \frac{x}{V} )</td>
<td>( \frac{x}{A} )</td>
</tr>
</tbody>
</table>

where [C₀] is the initial concentration of the drug in mol · L\(^{-1}\), [S₀] is the saturation capacity determined earlier in mol · dm\(^{-2}\), [Q₀] is the initial concentration of occupied surface sites in mol · dm\(^{-2}\) and set to 0, x is moles of adsorbed drug, A is the surface area of the device in dm\(^{-2}\), and V is the volume of the solution in L.

Combining (37) and Table 6 gives:

\[
K_{eq} = \frac{x}{A} = \frac{1}{([S₀] - \frac{x}{A})([C₀] - \frac{x}{V})}
\]  

(38)

Which can be arranged as a quadratic equation in standard form:

\[
0 = \frac{K_{eq}}{V} x^2 - (K_{eq}[S₀]\frac{A}{V} + K_{eq}[C₀] + 1) x + K_{eq}A[S₀][C₀]
\]

(39)

And solved with the quadratic formula, yielding:
\[ x = \frac{(K_{eq}[S_0]^A + K_{eq}[C_0] + 1) \pm \sqrt{(K_{eq}[S_0]^A + K_{eq}[C_0] + 1)^2 - 4K_{eq}^2[S_0][C_0]^A}}{2K_{eq}} \]  

This equation can be used to relate the initial drug concentration \([C_0]\), equilibrium constant of adsorption \(K_{eq}\), saturation capacity \([S_0]\), and device surface area-to-volume ratio \((A / V)\) to give the percentage of drug expected to remain in solution once equilibrium is reached.

The predicted percentage of drug concentration remaining for the two materials is shown in Figure 25. It should be noted that with sufficiently high surface area and low initial drug concentration, any drug will be effectively completely adsorbed. Similarly, with low surface area and high initial drug concentration, any drug concentration losses due to adsorption will be negligible. When graphed as a function of the initial concentration \([C_0]\) and the surface area-to-volume ratio \((A / V)\) on a log scale, the retention curves all exhibit a sharp drop-off similar to a titration curve.
Figure 25. A) Surface plots depicting the percentage of drug predicted to remain as a function of initial drug load [C0] and surface area-to-volume ratio (A / V) as calculated by (40), using the saturation capacity [S0] and adsorption equilibrium constant Keq for the drug-material combination listed. All curves predict domains where the drug is completely retained in solution or completely adsorbed, with a sharp transition between the two domains. B) Lines depict the percentage of drug predicted to remain, specifically for the surface area-to-volume ratio of the device described. As adsorption to PMMA was predicted to be much less than adsorption to PDMS, only the curves for adsorption to PDMS was used. Points represent the measured amount remaining after 24 hours, error bars represent the standard deviation.
The point at which 50% of the initial drug concentration is expected to remain can be calculated by setting (40) equal to half the initial concentration (expressed as \( x = \frac{V[C_0]}{2} \)) and solving for \( [C_0] \) to give:

\[
[C_0]_{50\%} = 2 \left( [S_0]_A^A - \frac{1}{K_{eq}} \right)
\]

(41)

The predicted and measured remaining concentration for each of the drugs is charted on Figure 25 for the device described above. Acetaminophen was predicted to have a negligible loss of concentration in solution, diclofenac up to a 15% loss for low initial concentrations (1E-7 to 1E-5 M), and verapamil up to a 98% loss for low initial concentrations. Experimental readings in the device showed a negligible loss of concentration for both acetaminophen and diclofenac, and verapamil decreased to 0-8% of initial concentration for all initial values after approximately 24 hours. The higher than predicted loss of verapamil was likely due to diffusion of adsorbed verapamil into the bulk of the material, a phenomenon not measured by ILC. The negligible loss in concentration of diclofenac was not predicted but could be explained by an underestimation of the surface area in the chromatographic column. The microspheres were treated as non-porous spheres with a negligible PDMS coating thickness and either of these assumptions could result in an underestimation of surface area. Since the system was under high pressure during the ILC experiments, but under ambient pressure during the housing experiments, it is possible the high pressure could have resulted in increased adsorption for the ILC experiment by enabling the adsorbate to travel deeper into the polymer. To investigate this, the housing experiment could be repeated inside a pressurized chamber. Pressure effects were observed by Fornstedt et al on the
adsorption energy distribution measured on a C18 column and attributed to higher pressure making the higher energy silanols on the silica support bed more accessible to the adsorbate. 

Experimental II – Modified PDMS

Organosilane modification of the surface of the PDMS gaskets was performed in a similar manner as described in CHAPTER FOUR: ORGANOSILANE SURFACE MODIFICATIONS.

To activate the gaskets, they were added to a 1:1 hydrochoric acid: methanol mixture in a beaker on an oscillating platform running at 10 rpm for 30 minutes. The gaskets were removed and rinsed in water before transferring to another beaker of water and boiled for 60 minutes. The PDMS gaskets were removed and dried under a stream of dry nitrogen before modification.

To modify the activated PDMS gaskets with 13F, 3.0 mL of 13F-silane was added to 1.5 L of freshly distilled toluene, mixed, and added to a beaker containing the activated PDMS gaskets and then placed on an oscillating platform running at 10 rpm and left to react for 30 minutes. The modified gaskets were then rinsed three times in toluene and left to dry in a 70 °C oven for 1 hour.

To modify the activated PDMS gaskets with PEG, 3.0 mL of PEG-silane and 0.75 mL of concentrated hydrochloric acid were added to 1.5 L of freshly distilled toluene, mixed, and added to a beaker containing the activated PDMS gaskets and then placed on an oscillating platform running at 10 rpm and left to react for 60 minutes. The modified gaskets were then rinsed in
toluene twice, in ethanol twice, and in water once before being left to dry on the benchtop overnight.

Gasket materials swelled to approximately double their original size in the presence of toluene during the modification reactions. This swelling may have allowed for the impregnation of organosilanes into the bulk of the material. Depth profiling with an ion gun in an XPS may be performed in the future to investigate this.

Results II

Organosilane modification to the PDMS gasket material was confirmed by XPS and goniometry, as shown in Figure 26. As PDMS is an insulator, an electron flood gun and a metal sample mask were used in conjugation to compensate for charge buildup on the material surface. XPS spectra of native PDMS gasket material had a single C1s peak. Atomic percentages were roughly at the expected ratio for Si:C:O of 1:2:1, at 27% : 43% : 30%. The contact angle of a 5 µL sessile droplet of water was measured as 111.7° ± 5.9. Modification with 13F-silane was again confirmed by the presence of F1s on the modified material and the addition of higher-energy peaks associated with CF2 and CF3. The contact angle of the 13F-modified gasket was increased to 119.3° ± 1.0. Modification with PEG-silane was confirmed by the addition of a higher-energy peak associated with C-O. The contact angle of the PEG-modified gasket was decreased to 74.3° ± 4.3, indicative of a more hydrophilic surface.
Figure 26. XPS spectra and contact angle of native PDMS gasket material (top), 13F-modified PDMS gasket material (middle), and PEG-modified PDMS gasket material (bottom). Contact angle was measured with a sessile 5 µL drop of water. Presence of the 13F-modification was evident by the addition of a F1s peak at 685 – 688 eV and fluorinated carbon peaks in the C1s region associated with CF2 and CF3 at 291.5 and 293.6 eV, as well as a slight increase in the contact angle. Presence of the PEG-modification was evident by the addition of a C-O peak at 286.4 eV in the C1s region and a large decrease in the contact angle indicating a more hydrophilic surface.

The curves predicting the remaining percentage of initial concentration for each drug-material combination and the amounts measured from the device are depicted in Figure 27.
Figure 27. Lines depict the percentage of drug predicted to remain, specific to the surface area-to-volume ratio of the device described (750 m⁻¹). Points represent the measured amount remaining after 24 hours, error bars represent the standard deviation.

As typical drug concentrations tested in body-on-a-chip devices are between the nanomolar to millimolar range, this domain was selected for testing adsorption. Initial concentrations tested for acetaminophen and diclofenac were low (1x10⁻⁷ M, 1x10⁻⁶ M, and 1x10⁻⁵ M), selected to check for any losses with an overwhelming abundance of empty adsorption sites present. Acetaminophen was predicted to maintain concentration at the ranges tested and this complete maintenance of concentration was observed in the test housing. Diclofenac was predicted to have a small concentration loss, but no concentration loss was observed. An underestimation of the surface area in the inverse chromatography columns could explain this difference. The microspheres were treated as non-porous spheres with a negligible PDMS coating thickness, and these assumptions could have resulted in an underestimation of surface area.
Verapamil was tested at higher initial concentrations (1x10^{-5} M, 1x10^{-4} M, 1x10^{-3} M) than acetaminophen or diclofenac as the amount of verapamil was expected to decrease at low initial concentrations. Although some minimal loss was predicted at the highest initial concentration tested, all samples’ concentrations tended to decay down to 1-10% of the initial concentration, and the two lowest initial concentrations converged to between 5.0x10^{-7} to 1.0x10^{-6} M at the 24-hour endpoint for all materials. The greater than predicted loss of concentration was attributed to diffusion into the PDMS bulk, which is not measured by ILC.

Concentration measurements were analyzed by a two-way ANOVA for statistical significance, and the results for the 24-hour endpoint are shown in Table 7. Although statistical significance was detected for some endpoint concentrations between native PDMS and the modified materials, the measured concentration differences were very small. In most cases no statistical significance was detected in the drug concentrations retained between the native material and the modified materials.
Table 7. Two-way ANOVA of endpoint concentration relative to native PDMS

<table>
<thead>
<tr>
<th>Acetaminophen</th>
<th>Diclofenac</th>
<th>Verapamil</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>13F-</td>
<td>PEG-</td>
</tr>
<tr>
<td>Concentration</td>
<td>PDMS</td>
<td>PDMS</td>
</tr>
<tr>
<td>1E-7</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>1E-6</td>
<td>0.005</td>
<td>0.048</td>
</tr>
<tr>
<td>1E-5</td>
<td>0.007</td>
<td>NS</td>
</tr>
</tbody>
</table>

While the adsorption equilibrium constant was an order of magnitude lower for verapamil on organosilane-modified PDMS versus the native PDMS, the saturation capacities were higher which drove the predicted adsorption to be greater on organosilane-modified PDMS. The higher saturation capacity is likely from the added organosilane chains on the microsphere surfaces, adding more effective surface area on a given microsphere. However, the verapamil concentrations measured at the 24-hour endpoint were not significantly different between the native PDMS gaskets and the organosilane-modified PDMS gaskets.

Discussion

The ILC methodology does not account for diffusion into the bulk of a material. The chromatographic model is based on a reversible system, implying rapid movement of adsorbate to and from the stationary phase. Practically, one would expect the process of diffusion of
adsorbate molecules into the material to be a slow process, not even measurable within the timeframe of a LC experiment. By using a PDMS-coated silica material instead of a homogenous PDMS material, the measurement was limited to just the surface interactions between the stationary phase and the analyte. This is useful as we can specifically measure the adsorption between the PDMS surface and analyte to establish the surface concentration, then add migration into the bulk to this model in a later manuscript.

The sequestration of drugs and hydrophobic factors in soft polymers can cause problems in long-term experiments and part reusability as these sequestered compounds may slowly release. This feature of PDMS has been investigated as useful for the sustained release of drugs,\textsuperscript{121-122} but would cause problems for the study of dose-response curves using a body-on-a-chip system. Non-polar drugs, when administered at a high concentration as an acute dose, may absorb into PDMS and slowly release over days at a lower dose. Furthermore, if PDMS parts are reused this slow release may carryover to unrelated experiments if the parts are not thoroughly cleaned to remove sequestered drugs.

Using PBS as the mobile phase in ILC experiments and as the base solution in adsorption housing experiments omits the effect of dissolved proteins common to cell culture media, such as albumin which is a major component in serum containing cell culture media, and has relevance clinically regarding the bioavailability of drugs.\textsuperscript{123} Drugs \textit{in vivo} bind to serum proteins and translating this interaction from \textit{in vitro} models into \textit{in vivo} measurements has been investigated.\textsuperscript{124} This interaction may affect the availability of drugs to adsorb to housing surfaces as the serum proteins would compete with adsorption sites. Additionally, serum proteins have been utilized to block surface adsorption sites in microfluidic channels, a technique known as
‘pre-fouling’,\textsuperscript{80,83} which would lessen the population of available surface adsorption sites to interact with drugs. As PBS was used in these experiments, these results would not reflect the effects of drug-protein binding and the pre-fouling of surfaces on the drug concentration losses due to adsorption to housing materials.

In the adsorption housing experiments, as no significant difference in concentration losses was observed when presented with either a more hydrophobic (13F-modified) or less hydrophobic (PEG-modified) material, it seems the hydrophobicity of the material surface doesn’t correlate with drug concentration losses to PDMS-based materials. Despite the marked decrease in the energy of adsorption associated with 13F and PEG modified PDMS surfaces as measured by ILC, this did not measurably prevent the sequestration of verapamil. Since adsorption must occur before absorption, we speculated that an organosilane surface modification might be able to at least slow down the diffusion into the bulk of the material, but this does not seem to be the case. Diffusion from the surface and into the bulk might occur faster than expected, rendering even the decreased energy of adsorption ineffective.

In our future work, we would like to integrate direct measurements of the diffusivity constant for drugs through PDMS. We plan to base this work on a publication by Belles et al.,\textsuperscript{115} who used the following methodology to measure the diffusivities of a variety of volatile organic compounds (VOCs) through PDMS: a thin slice of PDMS is loaded with an analyte and added to a stack of PDMS slices. The entire stack is then wrapped in foil and the analyte is allowed to diffuse through for a prescribed number of days. Analyte is later extracted from each slice of PDMS and the concentration is measured to build a diffusion profile. By adding this information...
to our model, we can build a two-step model of drug adsorption and diffusion into the PDMS bulk.
CHAPTER SIX: CONCLUDING REMARKS

This dissertation has demonstrated the development of Inverse Liquid-Solid Chromatography as a technique to characterize the adsorption of drugs to the surfaces of materials used in body-on-a-chip systems, to gain insight about the process and to test organosilane surface modifications as a means to affect the adsorption.

We implemented the Inverse Method by taking an analytical HPLC and converting it for use in inverse liquid-solid chromatography experiments. The data was then processed by a custom program to simulate band profiles by using a Gudonov numerical implementation of the Equilibrium-Diffusive model of chromatography paired with a Langmuir adsorption isotherm model. This approach had various advantages over other attempts at characterizing drug-material interactions: 1) this technique is specific to adsorption, making it well suited to assess surface modifications. 2) this method consumes little analyte, which is advantageous when assessing novel drugs which might be scarce since they are not yet produced commercially.

It was found that the dispersion coefficient (log D, pH 7.4) appeared to correlate strongly with the energy of adsorption. Further experiments will be conducted in the future to build a more robust dataset to confirm this trend and dissect the energy of adsorption to characterize the enthalpic and entropic contributions.

Direct measurements of strongly-adsorbing drug-material interactions was difficult since the drug would be retained until kinetic band broadening effects would decrease the peak below the detection threshold. To counter this, a method was developed to track the changes in the Langmuir parameters with respect to the concentration of an organic modifier, methanol, added
to the mobile phase. This allowed for the extrapolation of Langmuir parameters and could be used in the future assessment of other strongly-adsorbing drugs.

Organosilane surface modifications were successfully grafted to the PDMS surface and their effects on adsorption isotherms was assessed. Since the hypothesized mechanism driving drug adsorption was the Solvophobic Effect, organosilanes were selected that would change the hydrophobicity of native PDMS. A polyethyleneglycol based silane (PEG) was used to modify the surface to be more hydrophillic and a perfluoropolymer based silane (13F) was used to make it more hydrophobic. It was found that the hydrophobicity of the surface did not correlate with the observed changes (if any) in the energy of adsorption of drugs to the surface. No changes were observed in the energy of adsorption for the hydrophillic drug acetaminophen. Both modifications slightly increased the energy of adsorption for diclofenac, a somewhat hydrophobic drug. Both modifications greatly decreased the energy of adsorption for verapamil, a very hydrophobic drug, but to a much greater extent with PEG. It seems that the Solvophobic Effect might not be the main determinant for the energy of adsorption.

The adsorption isotherms measured by ILC were then used to predict the amount of drug concentration loss that would occur in an actual-use case. An equation was made to convert the adsorption isotherm parameters into a prediction of the amount of drug predicted to remain, dependent on the initial drug concentration loaded and the surface area-to-volume ratio of the device. A device was made to test the model and drug concentration losses were measured. Unfortunately, the model was not as predictive as was hoped, largely because the model did not account for diffusion into the bulk of the PDMS material. Future work will be done to measure
the diffusivity of drugs through PDMS by building a diffusion profile. This will be used to create a fuller model that covers both adsorption and diffusion into the bulk.

Since adsorption must occur before absorption, it was hypothesized that by modifying the PDMS surface with the organosilanes could prevent or slow the diffusion into the PDMS bulk. PDMS sheets were modified with the aforementioned PEG and 13F organosilanes and drug concentration losses were again measured. There was no measurable effect by the organosilanes on the concentration loss as measured in the devices despite the significant effect on the energy of adsorption observed in ILC measurements. This suggests that the diffusion into the bulk is the main driver of concentration losses and surface modifications that affect adsorption may not be adequate in creating a barrier to these losses.

While the end application of this work was not successful, a significant contribution to the methods used to assess adsorption and the effect of surface modifications to adsorption has been made. This technique will be developed further to create a fuller picture of the adsorption processes of drugs to polymer materials used in body-on-a-chip devices.
import os
import time
import simtools_branch_class2 as simtools
import simmethods_branch_class as simmethods

#START
directory = simtools.set_directory(subdirectory="!Work Space/")
batch_flag = input("Batch mode? (y/n): ")
yes = ['y', 'Y']

if batch_flag in yes:
    file_list = [f for f in os.listdir(directory) if f.endswith('-data.xlsx')]

    #batchparams = simtools.set_batchparams(directory)
    #chromparams = simtools.set_chromparams(directory)

    register = simtools.setup_parameters(directory)
simmethods.initialize_monitor()
    for fname in file_list:
        #gathering file parameters first
        line = int(fname[4:-12])
        optparams = simtools.get_parameters(register, line)
        simmethods.simulate(directory, fname, optparams)
        time.sleep(10)
        simtools.makereport(directory, register.batchparams)

else:
    inputdata = simtools.set_inputdata(directory)
    chromparams = simtools.set_chromparams(directory)
    optparams = simtools.get_params(directory, chromparams, None, -1)
    print(optparams)
simmethods.initialize_monitor()
simmethods.simulate(directory, inputdata, optparams)

print("Finished")
import scipy.interpolate
import numpy as np
import pandas as pd
import os
import datetime as dt
import secrets
import simmethods

class parameters():
    '''
    Class: parameters
    Contains all of the setup information and values for runtime & results reports.
    '''
    def __init__(self, N, Ms, t0, F, L, Vloop, Cinj, Calparam0, Calparam1):
        #First-order, primary values:
        self.N = np.int(N)  #Number of Theoretical plates (integer)
        self.Ms = Ms  #Mass stationary phase (g)
        self.t0 = t0  #Void time (min)
        self.F = F  #Flow rate (mL/min)
        self.L = L  #Column length (cm)
        self.Vloop = Vloop  #Injection volume (mL)
        self.Cinj = Cinj  #Injection concentration (uM)
        self.Calparam0 = Calparam0  #Intercept
        self.Calparam1 = Calparam1  #Slope

        self.area = 0.0  #Curve area (uM * mL)
        self.time_start = 0.0  #Start Timestamp where 1 = 24 hours (Excel format)
        self.time_end = 0.0  #End Timestamp where 1 = 24 hours (Excel format)
        self.time_elapsed = 0.0  #Duration Time where 1 = 24 hours (Excel format)
        self.iterations_langmuir = 0  #Number of Langmuir Fit iterations
        self.iterations_bilangmuir = 0  #Number of BiLangmuir Fit iterations
        self.abort_flag = False  #Flag for if the simulation aborted
        self.tR = 0.0  #Elution time (min)

        #Properties for second-order and third-order derived values
```python
@property
def Vm(self):
    # Void volume (mL)
    return self.F * self.t0

@property
def Phi(self):
    # Phase ratio (g / mL)
    return self.Ms / self.Vm

@property
def H(self):
    # Theoretical plate height (cm)
    return self.L / self.N

@property
def mol(self):
    # Amount of analyte injected (uM [1E-6] * mL [1E-3] = nmol [1E-9])
    return self.Cinj * self.Vloop

@property
def u(self):
    # Mobile phase linear velocity in the z direction (cm / min)
    return self.L / self.t0

def report(self):
    """
    Class function: parameters.report()
    Collects parameters into a list which is later written
to the *-results.xlsx file.
    """

    Input variables:
    NONE

    Returned variables:
    dict object: 'label' & 'values' paired lists
    containing the parameter class values.
    """

    label = ['N', 'Ms', 't0', 'F', 'L', 'Vloop', 'Cinj',
             'Calparam0', 'Calparam1', 'Mol', 'Area',
             'Start', 'End', 'Elapsed',
             'Langmuir Fit Iterations', 'Bilangmuir Fit Iterations',
             'Abort Flag', 'Elution Time']
              self.Calparam0, self.Calparam1, self.mol, self.area,
              self.time_start, self.time_end, self.time_elapsed,
              self.iterations_langmuir, self.iterations_bilangmuir,
              self.abort_flag, self.tR]
    return {'label': label, 'values': values}
```
class register():
    """
    Class: register
    Contains all of the file setup information.
    """

def __init__(self, directory, batchparams,
             columnparams="ColumnParams", analyteparams="AnalyteParams"):  
    #Setup register of file information
    self.directory = directory
    self.batchparams = batchparams
    self.columnparams = columnparams
    self.analyteparams = analyteparams

def love_and_support():
    """
    Thank you to everyone who helped, supported, or encouraged me through this doctorate.

    Input variables:
    Variable and countless.

    Returned variables:
    Thank yous and this dissertation.
    """
    print("Thank you to my wife, Dr. Alisha Colón!")
    print("Thank you to my amazing parents, Sasha Kusel and David Schnepper!")
    print("Thank you to my in-laws, Marie and Carlos Colón!")
    print("Thank you to my brother and his wife, Alex Schnepper and Dr. Kel Peyton!")
    print("Thank you to my wife's siblings and spouses, Dominic and Carissa Colón, Eric and Tara Colón, and Jessica and Frank Peppe!")
    print("Thank you to my grandparents, Genny Jenkins, Fred and Lynda Schneppeer, and LeRoy Steck!")
    print("Thank you to my wife's grandparents, Denise Ganthier, Anna and Angel Torres")
    print("Thank you to our dog, Ada Colón!")

def set_directory(subdirectory=''):  
    """
    Sets the input directory used for other functions.

    Input variables:
    *subdirectory (string)[OPTIONAL] - subdirectory, appended to output

    Returned variables:
    directory (string) - os path leading to the desired directory

    NOTE: Add more main directories for new computers/setups.
    """
# Main directory list - update as necessary

dir_list = [["Jeff Work", \\
"C:/Users/jroles/My Documents/Simulated Chromatograms/"], 
["Jeff Home", \\
"C:/JEFF HOME"], 
["Mark Work", \\
"C:/Users/mark/Desktop/Python Work Space/"], 
["Mark Home", \\
"C:/Users/Mark Schnepper/Desktop/Python Work Space/"]
]

directory = "Not Set"
# subdirectory = dict['subdirectory']
dir_set_flag = -1
path_exists = "Not Set"
number_found = 0

# requests user input to select the directory
while dir_set_flag == -1:
    print("Known Directories:")
    i = 1
    for Name, Path in dir_list:
        if os.path.isdir(Path):
            path_exists = "FOUND"
            number_found += 1
            directory_temp = Path
        else:
            path_exists = "-"
            print(i, ":", Name, ":", path_exists)
            i += 1
        if number_found == 1:
            directory_sub = directory_temp + subdirectory
            if os.path.isdir(directory_sub):
                directory = directory_sub
                print("Using: ", directory)
                dir_set_flag = 1
            else:
                directory = directory_temp
                print("ERROR: Subdirectory specified by program not found.", \\
"Reverting to known parent directory at:", directory)
                dir_set_flag = 1
        if number_found > 1:
            dir_set = int(input("Which directory?: "))
            # test if the input is a valid index
            if dir_set > 0 and dir_set <= len(dir_list):
                directory = dir_list[dir_set - 1][1]
                # checking if user-selected directory actually exists
                if os.path.isdir(directory):
                    print("Directory set to: ", directory)
                    dir_set_flag = 1
                else:
...


```python
print("Directory not found. Try again.")
else:
    print("Please enter a valid number.")
if number_found == 0:
    print("ERROR: No valid directory found")
    break

if dir_set_flag == 1:
    return directory

def set_batchparams(directory):
    
    Sets the batch parameters file to be used for other functions.

    Input variables:
    directory (string) - os pathway to designated directory

    Returned variables:
    batchparams (string) - batch parameter file name, including .xlsx suffix

    Last Updated: 2018/03/21
    Depreciated: 2020/02/17 (Date flagged)
    
    #Local variables
    batchparams = "Not Set"
batch_set = False

    batch_list = [f for f in os.listdir(directory) if f.startswith('BatchParams')]
    print('
', "===== Batch Parameter Files =====")
    for number, name in enumerate(batch_list):
        print(number + 1, ":", name)
    print("=================================", '
')
    number_found = len(batch_list)
    print("Number of batches found:", number_found)

    while not batch_set:
        if number_found == 1:
            batchparams = batch_list[0]
batch_set = True
        if number_found > 1:
            batch_selected = int(input("Which batch file?: "))
            if batch_selected > 0 and batch_selected <= len(batch_list):
                batchparams = batch_list[batch_selected - 1]
batch_set = True
            else:
                print("Please enter a valid number.")
        if number_found == 0:
            print("ERROR: No valid batch file found.")
            break

    if batch_set:
```

123
def set_chromparams(directory):
    
    """
    Sets the chromatography parameters file to be used for other functions.
    
    Input variables:
    directory (string) - os pathway to designated directory
    
    Returned variables:
    chromparams (string) - chromatography parameter file name, including .xlsx suffix
    
    Last Updated: 2018/03/21
    Deprecated: 2020/02/17 (Date flagged)
    """

    #Local variables
    chromparams = "Not Set"
    chrom_set = False
    
    chrom_list = [f for f in os.listdir(directory) if f.startswith('ExpChromParams')]
    print("\\n", "===== Chrom Parameter Files =====")
    for number, name in enumerate(chrom_list):
        print(number + 1, ":", name)
    print(" ==================================", \\
    number_found = len(chrom_list)
    print("Number of chromparams found:", number_found)

    while not chrom_set:
        if number_found == 1:
            chromparams = chrom_list[0]
            chrom_set = True
        if number_found > 1:
            chrom_selected = int(input("Which parameter file?: "))
            if chrom_selected > 0 and chrom_selected <= len(chrom_list):
                chromparams = chrom_list[chrom_selected - 1]
                chrom_set = True
            else:
                print("Please enter a valid number.")
        if number_found == 0:
            print("ERROR: No valid batch file found.")
            break
        if chrom_set:
            print("Using: ", chromparams)
            return chromparams

def setup_parameters(directory):
Setup the parameters file(s) that will be used for the simulation.
First gets the batchparam file name, & tests if there's a chromparam sheet.
If there's no chromparam sheet, it will ask which sheet to use
from the chromparam library spreadsheet.

Passed variables:
  directory (string): OS pathway to the specified directory

Returned variable: register (register class object):
  register.directory (string): directory path (same as input for this function)
  register.batchparams (string): batch parameter file name, including .xlsx suffix
  register.columnparams (string):
    Case 1: None - default case: grab from the batchparams file,
           ColumnParams sheet
    Case 2: column parameter sheet name from the column library.
  register.analyteparams (string):
    Case 1: None - default case: grab from the batchparams file,
           AnalyteParams sheet
    Case 2: analyte parameter sheet name from the analyte library.

Written: 2018/07/23
Updated: 2018/12/14

#Local variables
batchparams = "Not Set"
batch_set = False
columnlibraryfname = directory + "Local Libraries/"\
  + "Library - Column Parameters.xlsx"
analytelibraryfname = directory + "Local Libraries/"\
  + "Library - Analyte Parameters.xlsx"

batch_list = [f for f in os.listdir(directory) if f.startswith('BatchParams')]
for number, name in enumerate(batch_list):
  print(number + 1, ":", name)
print("Number of batches found:", number_found)

while not batch_set:
  if number_found == 1:
    batchparams = batch_list[0]
    batch_set = True
  if number_found > 1:
    batch_selected = int(input("Which batch file?: "))
    if batch_selected > 0 and batch_selected <= len(batch_list):
      batchparams = batch_list[batch_selected - 1]
      batch_set = True
    else:
      print("Please enter a valid number.")
  if number_found == 0:
print("ERROR: No valid batch file found.")
break

if batch_set:
    print("Using: ", batchparams)

batchxl = pd.ExcelFile(directory + batchparams)
if len(batchxl.sheet_names) < 3:
    column_set = False
    analyze_set = False
    print("No associated column or analyze sheets in this batchparams file.",
          "Please select from these:"
    )
columnxl = pd.ExcelFile(columnlibraryfname)
column_list = columnxl.sheet_names
print(columnxl)
print('
', "===== Column Parameter Library =====")
for number, name in enumerate(column_list):
    temp_df = pd.read_excel(columnlibraryfname, sheetname=name)
    print(number + 1, ":", name, " - ",
          '{:%Y%m%d}'.format(temp_df.at[0, "Updated"]))
print(" =========== Number of entries: ", len(column_list))

while not column_set:
    column_selected = int(input("Which column entry?: "))
    if column_selected > 0 and column_selected <= len(column_list):
        columnparams = column_list[column_selected - 1]
        column_set = True
    else:
        print("Please enter a valid number.")

analytexl = pd.ExcelFile(analytelibraryfname)
analyze_list = analytexl.sheet_names

print('
', "===== Analyte Parameter Library =====")
for number, name in enumerate(analyze_list):
    temp_df = pd.read_excel(analytelibraryfname, sheetname=name)
    print(number + 1, ":", name, " - ",
          '{:%Y%m%d}'.format(temp_df.at[0, "Updated"]),
          ", " , temp_df.at[0, "Vloop"]*1000, "ul", ", ",
          '{0:.1E}'.format(temp_df.at[0, "Cinj"]/1E+6), "M ")
print(" =========== Number of entries: ", len(analyze_list))

while not analyze_set:
    analyze_selected = int(input("Which analyte entry?: "))
    if analyze_selected > 0 and analyze_selected <= len(analyze_list):
        analyteparams = analyze_list[analyze_selected - 1]
        analyze_set = True
    else:
        print("Please enter a valid number.")
else:
    columnparams = "ColumnParams"
    analyteparams = "AnalyteParams"

print("Using: ", columnparams, " and ", analyteparams)

setup = register(directory, batchparams,
    columnparams=columnparams, analyteparams=analyteparams)
return setup

def set_inputdata(directory):
    
    Sets the input experimental data file used for other functions.

    Passed variables:
    directory (string): OS pathway to the specified directory

    Returned variables:
    inputdata (string): file name

    Updated:
    
    #local variables
    file_set_flag = -1
    overwrite_flag = 0
    inputdata = "Not Set"
    inputfname = "Not Set"
    outputfname = "Not Set"

    #requests user input for the input data file name
    while file_set_flag == -1:
        print("Please type the data file to process. (.xlsx files only)"")
        inputdata = input("Filename?: ")
        if inputdata[-5:] == ".xlsx":
            inputdata = inputdata[:-5]
        inputfname = directory + inputdata + ".xlsx"
        outputfname = directory + inputdata + "-results.xlsx"
        if os.path.isfile(inputfname):
            print("File found.")
            if os.path.isfile(outputfname):
                print("Existing output file", inputdata + "-results.xlsx found.")
                overwrite_flag = input("Overwrite? (y/n): ")
                if overwrite_flag == "y" or overwrite_flag == "Y":
                    print("Overwriting.")
                else:
                    continue
            return inputdata + ".xlsx"
        else:
            print("File not found. Please check filename and retype.")
def get_params(register, line):
    """
    Gets the parameters from the specified files and returns them as an array.
    
    Input variables:
    directory (string) - os pathway to the directory specified
    chromatography parameters file, ends with .xlsx
    batchparams (string) - batch parameters file, ends with .xlsx
    line (string) - line to reference inside of the batch parameters file
    
    Returned variables:
    optparams (parameters) - parameters class object used for the simulation
    
    Written to disk:
    NONE
    """

    chromfname = directory + chromparams
    chromparams_df = pd.read_excel(chromfname, header=None, usecols=[0])
    optparams = list(chromparams_df.values.flatten())
    #print(optparams)

    if batchparams != None:
        batchfname = directory + batchparams
        batchparams_df = pd.read_excel(batchfname, index_column=0)
        optparams[0] = batchparams_df.at[line, "N"]  # replacing with batch value
        optparams[2] = batchparams_df.at[line, "Void"]  # replacing with batch value
        #print(optparams[0], optparams[2])

    params = parameters(optparams[0], optparams[1], optparams[2], optparams[3],
                        optparams[4], optparams[5], optparams[6], optparams[8],
                        optparams[9])

    return params

def get_parameters(reg, line):
    """
    Gets the parameters from the batchparams file & chromparams library, and returns a parameters class object.
    
    Input variables:
    reg (register class object):
    reg.directory (string) - os pathway to the directory specified
    reg.batchparams (string): batch parameter file name, including .xlsx suffix
    reg.columnparams (string):
    """
Case 1: None - default case: grab from the batchparams file, ColumnParams sheet
Case 2: column parameter sheet name from the column library.
reg.analyteparams (string):
Case 1: None - default case: grab from the batchparams file, AnalyteParams sheet
Case 2: analyze parameter sheet name from the analyte library.
line (int) - line to reference inside of the batch parameters file

Returned variable:
optparams (parameters) - parameters class object used for the simulation

Written: 2018/07/23
Updated: 2018/12/14

""

batchfname = reg.directory + reg.batchparams
subdirectory = "Local Libraries/"
column_library = "Library - Column Parameters.xlsx"
analyte_library = "Library - Analyte Parameters.xlsx"

if reg.columnparams == "ColumnParams":
    print("Reading in Column Parameters from Batch File")
    columnparams_df = pd.read_excel(batchfname, sheetname=reg.columnparams)
else:
    print("Reading in Column Parameters from Library File")
    columnparams_df = pd.read_excel(reg.directory + subdirectory + column_library,
                                    sheetname=reg.columnparams)

if reg.analyteparams == "AnalyteParams":
    print("Reading in Analyte Parameters from Batch File")
analyteparams_df = pd.read_excel(batchfname, sheetname=reg.analyteparams)
else:
    print("Reading in Analyte Parameters from Library File")
analyteparams_df = pd.read_excel(reg.directory + subdirectory + analyte_library,
                                    sheetname=reg.analyteparams)

print("Reading in Batch File")
batchparams_df = pd.read_excel(batchfname, sheetname="BatchParams")
print(batchparams_df)

columnparams_df.at[0, "N"] = batchparams_df.at[line, "N"]
columnparams_df.at[0, "t0"] = batchparams_df.at[line, "Void"]

#N, Ms, t0, F, L, Vloop, Cinj, Calparam0, Calparam1

optparams = parameters(columnparams_df.at[0, "N"],
                        columnparams_df.at[0, "Ms"],
                        columnparams_df.at[0, "t0"],
                        analyteparams_df.at[0, "F"],
                        columnparams_df.at[0, "L"],
                        columnparams_df.at[0, "Vloop"],
                        columnparams_df.at[0, "Cinj"],
                        columnparams_df.at[0, "Calparam0"],
                        columnparams_df.at[0, "Calparam1"])

129
analyteparams_df.at[0, "Vloop"],
analyteparams_df.at[0, "Cinj"],
analyteparams_df.at[0, "Calparam0"],
analyteparams_df.at[0, "Calparam1"])

return optparams

def timestamp():
    """
    Gets the current time and outputs it in an Excel-readable format.
    
    Input variables:
    None
    
    Returned variables:
    time (string)
    """

time_now = dt.datetime.now().time()
return time_now

def timestamp_excel(time):
    """
    Converts date & time from datetime library into an Excel-readable format.
    
    Passed variables:
    time (datetime object)
    
    Returned variables:
    excel_time (float)
    """

    seconds = time.hour*3600 + time.minute*60 + time.second

return float(seconds / 86400)

def makereport(directory, batchparams):
    """
    Generates a report for a batch of files.
    
    Passed variables:
    directory (string) - os path to the data
    batchparams (string) - os path to the '-batchparams.xlsx'
    
    Returned variables:
    None
    
    Written to disk:
"-Report.xlsx" (Excel file)

Written: 2018/04/05
Updated: 2020/02/16

print("Generating report...")

#Setup
file_list = [f for f in os.listdir(directory) if f.endswith('-results.xlsx')]

#Load in batch information
batchfname = directory + batchparams
batchinfo = pd.read_excel(batchfname, index_col=0)
EndTime = batchinfo.EndTimes.max(axis=0)
num_points = EndTime * 240 #1 point every 250 ms
#num_points = 1000

experimentname = batchparams[11:-5]
date = '-v' + dt.datetime.today().strftime('%Y%m%d')
outputfname = directory + "Report" + experimentname + date + ".xlsx"
report_writer = pd.ExcelWriter(outputfname)

#Setup report dataframes
LangmuirFit = pd.DataFrame({"ID":[],
"MeOH":[],
"Sample":[],
"A1":[],
"B1":[],
"Residuals":[]})

BiLangmuirFit = pd.DataFrame({"ID":[],
"MeOH":[],
"Sample":[],
"A1":[],
"B1":[],
"A2":[],
"B2":[],
"Residuals":[]})

SimTimes = pd.DataFrame({"ID":[],
"MeOH":[],
"Sample":[],
"Time":[],
"Langmuir Iterations":[],
"BiLangmuir Iterations":[],
"Void Time":[],
"Elution Time":[],
"Time Delta":[]})

PeakTimes = pd.DataFrame({"ID":[],
"MeOH":[],
"Sample":[],
"Peak Time":[]})
"Peak Height"=[],
"Peak Index"=[]})

#Continued setup
mastertime = scipy.linspace(0, EndTime, num_points)
SampledChrom = pd.DataFrame({"Time": mastertime})
AllChrom = pd.DataFrame({"Time": mastertime})
abort_count = 0

#Randomly select chromatograms from each "line" to print out in the report
line_list = []
print("Checking for aborted files...")
for fname in file_list:
    #First check if this "-results.xlsx" file was aborted.
    print("Checking:", fname)
    abort_flag = False
    Checkme = pd.read_excel(directory + fname, 'Parameters',
                           header=None, index_col=None)
    Checkme.columns = ['Label', 'Data']
    abort_flag = Checkme.Data[16]
    line = fname[4:-15]
    tag = fname[:-13]
    if line not in [item[0] for item in line_list]:
        line_list.append([line])
        index = line_list.index([line])
    if tag not in line_list[index]:
        if abort_flag:
            abort_count = abort_count + 1
        else:
            line_list[index].append(tag)

#Randomly select from the completed files, one per batch line
selections = []
for item in line_list:
    line = item[0]
    options = item[1:]  
    if len(options) > 0:
        selection = secrets.choice(options)
        selections.append([line, selection])
print("Files randomly selected to represent this batch:", selections)

#Begin collecting from "-results.xlsx" files in directory
for fname in file_list:
    inputfname = directory + fname
    print("Processing:", fname)

    line = fname[4:-15] #Line of batch file
    indexme = int(line) #Line to index to bachinfo DataFrame
    tag = fname[:-13] #
    checkme = [line, tag]
Harvest batchinfo data
meoh = batchinfo.MeOH[indexme]
sample = batchinfo.Sample[indexme]

Harvest Langmuir & BiLangmuir fit data
ResultsFile = pd.ExcelFile(inputfname)

Get Langmuir fit data
Langmuir_readme = pd.read_excel(ResultsFile, 'Langmuir Fit', index_col=None)
Langmuir_addme = pd.DataFrame({'ID':[tag],
                               'MeOH':[meoh],
                               'Sample':[sample],
                               'A1':[Langmuir_readme.A1[0]],
                               'B1':[Langmuir_readme.B1[0]],
                               'Residuals':
                               [Langmuir_readme.Residuals[0]])
LangmuirFit = LangmuirFit.append(Langmuir_addme, ignore_index=True)

Get BiLangmuir fit data
BiLangmuir_readme = pd.read_excel(ResultsFile, 'BiLangmuir Fit',
                                   index_col=None)
B1 = BiLangmuir_readme.B1[0]

Order B-term results from greatest to smallest
if B1 < B2:
    Swap Langmuir term results
    A1 = BiLangmuir_readme.A1[0]
    B1 = BiLangmuir_readme.B2[0]
    A2 = BiLangmuir_readme.A2[0]
else:
    Load remaining A terms if order is OK
    A1 = BiLangmuir_readme.A1[0]
    A2 = BiLangmuir_readme.A2[0]

BiLangmuir_addme = pd.DataFrame({'ID':[tag],
                                 'MeOH':[meoh],
                                 'Sample':[sample],
                                 'A1':[A1],
                                 'B1':[B1],
                                 'A2':[A2],
                                 'B2':[B2],
                                 'Residuals':
                                 [BiLangmuir_readme.Residuals[0]])
BiLangmuirFit = BiLangmuirFit.append(BiLangmuir_addme, ignore_index=True)

Get simulation duration
SimTimes_readme = pd.read_excel(ResultsFile, 'Parameters',
                                header=None, index_col=None)
SimTimes_readme.columns = ['Label', 'Data']
print(SimTimes_readme)
time = SimTimes_readme.Data[13]
langmuir_iterations = SimTimes_readme.Data[14]
bilangmuir_iterations = SimTimes_readme.Data[15]
abort_flag = SimTimes_readme.Data[16]
voidtime = SimTimes_readme.Data[2]
elutiontime = SimTimes_readme.Data[17]
time_delta = elutiontime - voidtime
SimTimes_addme = pd.DataFrame({'ID': [tag],
                               'MeOH': [meoh],
                               'Sample': [sample],
                               'Time': [time],
                               'Langmuir Iterations': [langmuir_iterations],
                               'BiLangmuir Iterations': [bilangmuir_iterations],
                               'Void Time': [voidtime],
                               'Elution Time': [elutiontime],
                               'Time Delta': [time_delta]})
SimTimes = SimTimes.append(SimTimes_addme, ignore_index=True)

if not abort_flag:
    # harvest chromatograms:
    # Load in chromatogram data:
    LangmuirChrom_readme = pd.read_excel(ResultsFile,
                                          'Langmuir Chromatogram',
                                          index_col=None)
    BilangmuirChrom_readme = pd.read_excel(ResultsFile,
                                           'BiLangmuir Chromatogram',
                                           index_col=None)

    # load in time lists
    LangmuirChrom_readme_time = LangmuirChrom_readme.Time.values
    BilangmuirChrom_readme_time = BilangmuirChrom_readme.Time.values

    # use the longer time list and its associated experimental chromatogram
    if len(BilangmuirChrom_readme_time) > len(LangmuirChrom_readme_time):
        Chrom_readme_time = BilangmuirChrom_readme_time
        Chrom_readme_exp = BilangmuirChrom_readme.Experimental.values
    else:
        Chrom_readme_time = LangmuirChrom_readme_time
        Chrom_readme_exp = LangmuirChrom_readme.Experimental.values

    # load in the simulated chromatograms
    LangmuirChrom_readme_sim = LangmuirChrom_readme.Simulated.values
    BilangmuirChrom_readme_sim = BilangmuirChrom_readme.Simulated.values

    # added 2020/02/16: Collect the peak heights/times for the experimental
    peak_max_Langmuir = LangmuirChrom_readme.Experimental.max()
    peak_max_Bilangmuir = BilangmuirChrom_readme.Experimental.max()
    if peak_max_Langmuir < peak_max_Bilangmuir:
        peak_max = peak_max_Bilangmuir
        peak_index = BilangmuirChrom_readme.Experimental.idxmax()
        peak_time = BilangmuirChrom_readme.Time.iloc[peak_index]
    else:
peak_max = peak_max_Langmuir
peak_index = LangmuirChrom_readme.Experimental.idxmax()
peak_time = LangmuirChrom_readme.Time.iloc[peak_index]

PeakTimes_addme = pd.DataFrame({"ID":[tag],
"MeOH":[meoh],
"Sample":[sample],
"Peak Time":[peak_time],
"Peak Height":[peak_max],
"Peak Index":[peak_index]})
PeakTimes = PeakTimes.append(PeakTimes_addme, ignore_index=True)

#Interpolate curves to the mastertime list
Chrom_exp = simmethods.interpolate(Chrom_readme_time, 
Chrom_readme_exp, mastertime)
LangmuirChrom_sim = simmethods.interpolate(Chrom_readme_time, 
LangmuirChrom_readme_sim, mastertime)
BiLangmuirChrom_sim = simmethods.interpolate(Chrom_readme_time, 
BiLangmuirChrom_readme_sim, mastertime)

exp_label = tag + " Experimental"
langmuir_label = tag + " Langmuir"
bilangmuir_label = tag + " BiLangmuir"
Chrom_addme = pd.DataFrame({exp_label: Chrom_exp,
langmuir_label: LangmuirChrom_sim,
bilangmuir_label: BiLangmuirChrom_sim})
Chrom_addme = Chrom_addme[[exp_label, langmuir_label, bilangmuir_label]]
AllChrom = pd.concat([AllChrom, Chrom_addme], axis=1)

#Check if this "-results.xlsx" file was selected for its chromatogram
if checkme in selections:
    print("This file has been selected for its chromatograms:", tag)
    SampledChrom = pd.concat([SampledChrom, Chrom_addme], axis=1)

#Add in abort count summary
AbortReport = pd.DataFrame({"Abort count":[abort_count]})

#Resort the dataframes into the right column orders
PeakTimes = PeakTimes[["ID", "MeOH", "Sample", "Peak Time", "Peak Height", "Peak Index"]]

135
# Write out the report

```python
PeakTimes.to_excel(report_writer, sheet_name="Peak Times", index=False)
LangmuirFit.to_excel(report_writer, sheet_name="Langmuir Fit", index=False)
BiLangmuirFit.to_excel(report_writer, sheet_name="BiLangmuir Fit", index=False)
SampledChrom.to_excel(report_writer, sheet_name="Sampled Chromatograms",
                      index=False)
AllChrom.to_excel(report_writer, sheet_name="All Chromatograms", index=False)
SimTimes.to_excel(report_writer, sheet_name="Sim Times", index=False)
AbortReport.to_excel(report_writer, sheet_name="Abort Report", index=False)
```

```python
delete report_writer
```

```python
def abort(resultsfname, optparams):
    ""
    Protocol to generate a "-results.xlsx" file in the case that the simulation
    Passed variables:
    resultsfname (string) - os pathway to the "-results.xlsx" file
    optparams (parameters class object) - experimental parameters
    Returned variables:
    NONE
    Written to disk:
    "-results.xlsx" - results file, but with an abort sheet instead of sim data
    Written: 2018/04/13
    Updated: 2018/07/12
    ""
    # Set up file handling
    results_writer = pd.ExcelWriter(resultsfname)
    # Populate "-results.xlsx" sheet
    empty = -1
    langmuirfit = pd.DataFrame({"Residuals": [empty],
                                "A1": [empty],
                                "B1": [empty]})
    bilangmuirfit = pd.DataFrame({"Residuals": [empty],
                                   "A1": [empty],
                                   "B1": [empty],
                                   "A2": [empty],
                                   "B2": [empty]})
    stats = optparams.report()
    parameters_report = pd.DataFrame({"Parameters": stats["label"],
                                       "Values": stats["values"]})
    langmuirfit.to_excel(results_writer, sheet_name="Langmuir Fit", index=False)
```
bilangmuirfit.to_excel(results_writer, sheet_name="BiLangmuir Fit", index=False)
parameters_report.to_excel(results_writer, sheet_name="Parameters",
header=False, index=False)

# Make an abort sheet report to flag the Results
abort_report = pd.DataFrame({'Another Castle': ['Your Princess']})
abort_report.to_excel(results_writer, sheet_name="Abort", index=False)
results_writer.save()
SimMethods Library
Last Updated: 2020/02/17

This library is for containing all of the simulation methods. Everything here supports the main simulate() function.

```python
import itertools
import numpy as np
import scipy.interpolate
import scipy.optimize
import scipy.signal
from scipy.signal import savgol_filter
import pandas as pd
import simtools_branch_class2 as simtools
from matplotlib import pyplot

```

GLOBAL VARIABLES USED TO TRACK THE SIMPLEX FIT PROGRESS

```python
residual_tracker = 0.0
langmuir_tracker = pd.DataFrame({"Residuals": [],
                               "A1": [],
                               "B1": []})
bilangmuir_tracker = pd.DataFrame({"Residuals": [],
                                   "A1": [],
                                   "B1": [],
                                   "A2": [],
                                   "B2": []})

```

END GLOBAL VARIABLES

```python
def initialize_monitor():
    
```
Data visualization. Initializes the "Monitor" figure object.

Input variables:
   NONE

Returned variables:
   NONE

Written to disk:
   NONE

Displayed:
   "Monitor" figure object

Written: 2018/06/01
Updated: 2018/06/01

```
# Label the figure (window)
pyplot.figure("Monitor")

# Create the Chromatogram comparison subplot
pyplot.subplot(211)
pyplot.title("Chromatogram Fit")
pyplot.xlabel("Time (minutes)")
pyplot.ylabel("Concentration (uM)")

# Create the 1st Langmuir Terms subplot
pyplot.subplot(223)
pyplot.title("1st Langmuir Terms")
pyplot.xlabel("A1")
pyplot.ylabel("B1")

# Create the 2nd Langmuir Terms subplot
pyplot.subplot(224)
pyplot.title("2nd Langmuir Terms")
pyplot.xlabel("A2")
pyplot.ylabel("B2")

# Resize axes objects (the subplots) so they all fit in the figure
pyplot.tight_layout()
```

```python
def clear_monitor():
    ""
    Data visualization. Clears the "Monitor" figure object.
    ""
    Input variables:
       NONE
    Returned variables:
       NONE
```
Written to disk:
   NONE

Displayed:
   "Monitor" figure object

Written: 2018/06/01
Updated: 2018/06/01

```python
#Clear the chromatogram comparisons & relabel
pyplot.subplot(211)
pplot.cla()
pplot.title("Chromatogram Fit")
pplot.xlabel("Time (minutes)")
pplot.ylabel("Concentration (μM)"")

#Clear the 1st Langmuir Terms & relabel
pyplot.subplot(223)
pplot.cla()
pplot.title("1st Langmuir Terms")
pplot.xlabel("A1")
pplot.ylabel("B1")
pplot.yscale("log")
pplot.xscale("log")

#Clear the 2nd Langmuir Terms & relabel
pyplot.subplot(224)
pplot.cla()
pplot.title("2nd Langmuir Terms")
pplot.xlabel("A2")
pplot.ylabel("B2")
pplot.yscale("log")
pplot.xscale("log")
```

```python
def update_monitor(chromatograms, scores):
    
    Data visualization. Updates the "Monitor" with this iteration's data.

    Input variables:
        chromatograms (list of lists):
            chromatograms[0] (list): time series
            chromatograms[1] (list): experimental values
            chromatograms[2] (list): simulated values
        scores (dataframe): list of terms evaluated & their residuals

    Returned variables:
        NONE

    Written to disk:
        NONE
```
Displayed:
  Updates the "Monitor" figure with the current data.

NOTE: MUST initialize the Monitor first.
NOTE: lengths of time, exp_chrom, and sim_chrom are assumed to be the same

Written: 2018/06/01
Updated: 2020/02/17

```
#Sort scores by descending value in the Residuals column
scores = scores.sort_values(by='Residuals', ascending=False)
scores = scores.reset_index(drop=True)

#Unpack scores:
size = scores.shape[1]  #Number of columns
Residuals = scores.Residuals.values
minindex = scores.Residuals.idxmin(axis=0)
Residuals = np.log(Residuals)  #Mapping residuals to their log values
#Residuals_max = max(Residuals)

#Setup colormap settings
cmap = pyplot.cm.RdYlGn_r  #Load this colormap
#norm = pyplot.Normalize(0.0, 20.0)  #Normalize colormap scale
#cmap.set_over(color = 'k')  #Color for points outside the range

#Plot experimental and simulated chromatograms
pyplot.subplot(211)
pyplot.cla()
pyplot.title("Chromatogram Fit")
pyplot.xlabel("Time (minutes)")
pyplot.ylabel("Concentration (uM)")
pyplot.plot(chromatograms[0], chromatograms[1], 'k-')
pyplot.plot(chromatograms[0], chromatograms[2], 'r:')
pyplot.text(0, 0, scores.Residuals.iloc[minindex])

#Plot 1st Langmuir Terms & Residuals
pyplot.subplot(223)
A1 = scores.A1.values
B1 = scores.B1.values
pyplot.scatter(A1, B1, c=Residuals, s=5, cmap=cmap)
A1min = scores.A1.iloc[minindex]
B1min = scores.B1.iloc[minindex]
pyplot.scatter(A1min, B1min, s=70, facecolors='none', edgecolors='k')

#Plot 2nd Langmuir Terms & Residuals (if BiLangmuir fit)
if size == 5:
    pyplot.subplot(224)
    A2 = scores.A2.values
    B2 = scores.B2.values
```
pyplot.scatter(A2, B2, c=Residuals, s=5, cmap=cmap)
A2min = scores.A2.iloc[minindex]
B2min = scores.B2.iloc[minindex]
pyplot.scatter(A2min, B2min, s=70, facecolors='none', edgecolors='k')
pyplot.pause(0.1)

def make_range(term, radius, n):
    """
    Function to make a 1D range around the given term.
    Used to build the Hypercube Grid for searching parameters.

    Input variables:
    term (float): Langmuir Term
    radius (float): Relative size of range. eg, 0.5 for 50%
    n (int): total number of terms to return in the list

    Returned variables:
    term_range (list): list of floats for the term range, log spacing

    Written: 2018/06/05
    Updated: 2020/02/17
    """

    if term == 0.0:
        return [0.0]

    #LINEAR IMPLEMENTATION
    term_radius = term * radius
    left = term - term_radius * 2 #EDIT: extended left (lower) side
    if left < 0.0:
        left = 0.0
    right = term + term_radius
    if right < 0.0:
        right = 0.0

    term_range = np.linspace(left, right, n)

    #LOGARITHMIC IMPLEMENTATION
    left = np.log10(term * (10 ** -radius))
    right = np.log10(term * (10 ** radius))

    term_range = np.logspace(left, right, num=n)
    print(term_range)
    return term_range
def interpolate(x, y, xi):
    """
    Interpolate function.
    Takes the paired points (x,y) and evaluates them at xi
to yield a set of (xi, yi) points.
    """

    Input variables:
    x (list of doubles)
y (list of doubles, paired to x)
xi (list of evenly-spaced points along the x-axis)

    Returned variables:
    Returns cs.__call__(xi, nu = 0, extrapolate = None)

    Last Updated: 2018/05/09
    """

    #Check if the x and y lists are the same size. If not, resize to the shortest.
    if len(x) > len(y):
        while len(x) > len(y):
            del x[len(x)-1]
    if len(y) > len(x):
        while len(y) > len(x):
            del y[len(y)-1]

    #Feed x and y into the Interpolator
    cs = scipy.interpolate.Akima1DInterpolator(x, y)
    return cs.__call__(xi, nu=0, extrapolate=None)

def specific_volume(times, concentrations, optparams, imax, imin):
    """
    Converts a paired list of time & concentration into
    specific volume & concentration across the specified index range (imax -> imin)
    which is the range from the peak maxima to the end of the peak tail.
    """

    Input variables:
    times (list of doubles)
    concentrations (list of doubles, paired to time)
optparams (list) - list of the parameters. Only need Ms, t0, and F
    imax (int) - index within the time/conc pair for the peak maxima
    imin (int) - index within the time/conc pair for the peak tail end

    Returned variables:
    package (dict): Contains the specific_volume/conc pair for imax -> imin
    package["specific_volumes"]: spec_vol
    package["specific_concentrations"]: spec_conc

    Written to disk:
    NONE
# Initialize lists
spec_conc = []
spec_vol = []

# Populate lists
for i in range(imax, imin, 1):
    spec_vol.append((times[i] - optparams.t0) * optparams.F / optparams.Ms)
    spec_conc.append(concentrations[i])

# Package up output
package = {
    "specific_volumes": spec_vol,
    "specific_concentrations": spec_conc
}
return package

def forward_backward(terms, optparams):
    ```
    Core function. Executes the forward-backward discretization for the given
    input parameters.

    Input variables:
    - terms (list) - Langmuir terms
    - optparams (parameters class object) - Chromatography parameters

    Returned variables:
    - simchromatogram (dict) -
      simchromatogram["time"] - simulated chromatogram time series
      simchromatogram["conc"] - simulated chromatogram concentration series

    Written to disk:
    NONE
    ```

Written: 2018/04/10
Updated: 2018/07/13

# Unpacking input parameters

# Langmuir terms
if len(terms) == 2:
    A1 = terms[0]
    B1 = terms[1]
    A2 = 0
    B2 = 0

if len(terms) == 4:
    A1 = terms[0]
    B1 = terms[1]
A2 = terms[2]
B2 = terms[3]

# Chromatography parameters
N = optparams.N
t0 = optparams.t0
L = optparams.L
area = optparams.area
Phi = optparams.Phi
H = optparams.H
u = optparams.u

tr = t0 * (1.0 + Phi * (A1 + A2))
ur = L / tr
dt = 2.0 * H / ur
locdelQ = 0.0
locdelC = 0.0
c = np.zeros((N + 1, N + 1))
q = np.zeros((N + 1, N + 1))
time = []
for i in range(0, N):
    time.append(0.)
for i in range(1, N):
    time[i] = time[i - 1] + dt

# Dirac Pulse
c[1, 1] = area / dt

# Forward-Backward Discretization
j = 1
while j <= N - 1:
    i = 1
    while i <= N:
        q[i, j] = (A1 * c[i, j]) / (1.0 + B1 * c[i, j]) + \ 
            (A2 * c[i, j]) / (1.0 + B2 * c[i, j])
        locdelQ = q[i, j] - q[i - 1, j]
        locdelC = c[i, j] - c[i - 1, j]
        c[i, j + 1] = c[i, j] - H / (u * dt) * (locdelC + Phi * locdelQ)
        i = i + 1
    j = j + 1
conc = []
i = 0
while i <= N - 1:
    conc.append(c[i, N])
    i = i + 1

simchromatogram = {"time": time, "conc": conc}
# print("Time End:", time[-1])
return simchromatogram

def scale_chromatograms(exp_time, exp_conc, sim_time, sim_conc):
Scales the experimental and simulated chromatograms to the sim time axis scale.

Input variables:
- exp_time (list): experimental times
- exp_conc (list): experimental values
- sim_time (list): simulated times
- sim_conc (list): simulated values

Returned variables:
- scaled_chromatograms (list of lists):
  - scaled_chromatograms[0]: master time
  - scaled_chromatograms[1]: experimental values
  - scaled_chromatograms[2]: simulated values

Written to disk:
NONE

Written: 2018/06/01
Updated: 2018/07/12

#Compare duration of experimental & simulated chromatograms
if exp_time[-1] > sim_time[-1]:
    end_time = sim_time[-1]
else:
    end_time = exp_time[-1]

#Build the common time list to interpolate the chromatograms onto
mastertime = []
mastertime.append(sim_time[0])
mastertime_last = sim_time[0]
dt = sim_time[1] - sim_time[0]
while mastertime_last <= end_time:
    mastertime.append(mastertime[-1] + dt)
mastertime_last = mastertime[-1]

#Interpolate experimental and simulated chromatograms across the mastertime
exp_compare = interpolate(exp_time, exp_conc, mastertime)
sim_compare = interpolate(sim_time, sim_conc, mastertime)

#Check for NaN and replace with 0.0
exp_compare = np.nan_to_num(exp_compare, copy=True)
sim_compare = np.nan_to_num(sim_compare, copy=True)

scaled_chromatograms = [mastertime, exp_compare, sim_compare]
return scaled_chromatograms

def fun2dv(terms, *args):
    """
Numerical simulation function for Langmuir adsorption model. Takes in the Langmuir Terms, and returns the residuals. Will also return a punitive residual for certain non-sensical inputs or results.

Global variables:
- residual_tracker - used to hold the residuals result for this run

Input variables:
- terms (list) - Input from optimizer
  - terms[0]: Langmuir Term A1
  - terms[1]: Langmuir Term B1
  - terms[2]: Langmuir Term A2
  - terms[3]: Langmuir Term B2
- *args (tuple) - additional arguments
  - args[0]: optparams (parameters class object) - chromatography parameters
  - args[1]: concentration (list) - concentrations
  - args[2]: time (list) - times matched to concentration list

Returns:
- sumsq (double)
  - Normal Case: returns the residuals between the experimental and simulated chromatograms
  - Error Case: returns a punitive value for the following conditions:
    - Negative Langmuir Term inputs
    - vmax <= vmin
    - NAN

Last Updated: 2019/03/17

```python
#Check input Langmuir Terms for negatives & penalize
for n in range(len(terms)):
    if terms[n] < 0.0:
        #DEBUG PRINT:
        print("Negative term generated. Exiting fun2dv().")
        sumsq = 1E+33
        residual_tracker = sumsq
        return sumsq

#Chromatography parameters input
optparams = args[0]

#Input from experimental data
exp_conc = args[1] #Experimental concentration
exp_time = args[2] #Experimental time

#Forward-backward discretization
simchromatogram = forward_backward(terms, optparams)
```
sim_time = simchromatogram["time"]
sim_conc = simchromatogram["conc"]

scaled_chromatograms = scale_chromatograms(exp_time, exp_conc,
                                          sim_time, sim_conc)

#Unpack scaled chromatograms
mastertime = scaled_chromatograms[0]
exp_compare = scaled_chromatograms[1]
sim_compare = scaled_chromatograms[2]

#Check master time length:
if len(mastertime) < 30:
    print("Sim time too short")
sumsq = 2E+33
    residual_tracker = sumsq
    return sumsq
...

#Testing for peak size & position
exp_peak_index = np.argmax(exp_compare)
exp_peak = exp_compare[exp_peak_index]
sim_peak_index = np.argmax(sim_compare)
sim_peak = sim_compare[sim_peak_index]

#DEBUG PRINT:
print("Exp peak:", exp_peak_index, '{:4f}'.format(exp_peak),
      "Sim peak:", sim_peak_index, '{:4f}'.format(sim_peak))

#Check for peak position on chromatogram:
peak_index_diff = abs(sim_peak_index - exp_peak_index)
eject_index = False
if peak_index_diff > 2:
    #sumsq = 5E+30 + 1E+30 * peak_index_diff
    #print(peak_index_diff)
    print("Eject: Peak index")
    eject_index = True
    #residual_tracker = sumsq
    #return sumsq

#Check for peak height difference on chromatogram:
threshold_diff = 0.05
peak_signal_diff = abs(sim_peak - exp_peak)/exp_peak
eject_peak = False
if peak_signal_diff < threshold_diff:
    #sumsq = 3E+30 + 1E+30 * peak_signal_diff
    #print("Eject: Peak difference")
    eject_peak = True
    #residual_tracker = sumsq
    #return sumsq
    #DEBUG PRINT:
    #print("Peak travelled outside threshold for simulation.",
           '{:.2f}'.format(sim_peak), "versus", '{:.2f}'.format(exp_peak))
#Check eject conditions:
if eject_index or eject_peak:
    sumsq = 3E+30 + (3E+30 * peak_index_diff) + (1E+30 * peak_signal_diff)
    residual_tracker = sumsq
    return sumsq

#Tabulate residuals
sumsq = 0.0
mastertime_length = len(mastertime)
for i in range(mastertime_length):
    sumsq = sumsq + (sim_compare[i] - exp_compare[i])**2

#Check if this is a number
if np.isnan(sumsq):
    #DEBUG PRINT:
    #print("NAN detected.")
    sumsq = 5E+32
    #2017/10/11 increased penalty for NAN
    #DEBUG PRINT:
    #print("Residual ", sumsq)

#========================================
#Update the realtime monitor (useful for debugging)
#========================================

#Find number of Langmuir terms
if len(terms) == 2:
    tracker = langmuir_tracker
elif len(terms) == 4:
    tracker = bilangmuir_tracker
if len(tracker) > 0:
    update_monitor(scaled_chromatograms, tracker)

#DEBUG PRINT:
#difference = mastertime_length - optparams.N
#if difference > 3:
    #print("Mismatch of N and mastertime:", optparams.N, mastertime_length)
residual_tracker = sumsq
#print("Residuals:", sumsq)
return sumsq

def callme(terms):
    # Callback function for scipy.optimize
Used to update the tracking variables while running the simplex search.

Global variables:
- residual_tracker (float) - holds the temporary residual
- langmuir_tracker (DataFrame) - holds the langmuir fit results
- bilangmuir_tracker (DataFrame) - holds the bilangmuir fit results

Input variables:
- terms - fit results
  0: A1
  1: B1
  2: A2
  3: B2

Updated: 2020/02/17

```python
# Determine number of Langmuir sites we're fitting to (only supports 1 or 2)
n_langmuir = 0
if len(terms) == 2:
    n_langmuir = 1
if len(terms) == 4:
    n_langmuir = 2

sumsq = residual_tracker

if n_langmuir == 1:
    temp = pd.DataFrame(
        "Residuals": [sumsq],
        "A1": [terms[0]],
        "B1": [terms[1]])
    langmuir_tracker = langmuir_tracker.append(temp, ignore_index=True)

elif n_langmuir == 2:
    temp = pd.DataFrame(
        "Residuals": [sumsq],
        "A1": [terms[0]],
        "B1": [terms[1]],
        "A2": [terms[2]],
        "B2": [terms[3]])
    bilangmuir_tracker = bilangmuir_tracker.append(temp, ignore_index=True)
```

```python
def Hypersphere(resultsfname, terms, optmethod, inputbundle):
    
Hypersphere search method
Seeks a minimum within the solution space by testing at equidistant points from the current solution.

Global variables:
- langmuir_tracker (DataFrame) - holds the langmuir fit results
```

150
bilangmuir_tracker (DataFrame) - holds the bilangmuir fit results

Input variables:
resultsfname (string) - os pathway to the "-results.xlsx" file location
terms (list, length 2 or 4) - Langmuir Terms in this order: A1, B1, A2, B2
optmethod (string) - which optimization method to use
  (e.g., NelderMead, SLSQP, etc.)
inputbundle (dict):
  optparams (parameter class object): contains the experiment parameters
  conc (list): concentrations list
  time (list): time list, paired to concentrations

Returned variables:
fopt (1-row DataFrame object) - with these columns:
  "Residuals","Iterations","A1","B1","A2","B2"

Written to disk:
  score.xlsx - adds the current solution for each iteration

Last Updated: 2018/07/12
"

global langmuir_tracker
global bilangmuir_tracker

#Search setup parameters (hard-coded for now: 2018/03/26)

"""Nelder-Mead parameters:"""
#Maximum number of iterations to run
maxiter = 200
#Absolute error in xopt (guesses) between iterations to converge
xatol = 1E-7
#Absolute error in fatol (results) between iterations to converge
fatol = 1E-3

"""Hypercube Grid parameters:"""
#Default: 0.5
radius_single = 0.5
#Default: 11. Will give 11^2 = 121 points per iteration to test
points_single = 11
#Default: 0.5
radius_bilangmuir = 0.5
#Default: 5. Will give 5^4 = 625 points per iteration to test
points_bilangmuir = 5

"""Optimizer loop parameters:"""
#Maximum number of iterations to run. Default: 30
niter = 30
#Number of same-value iterations required to exit loop. Default: 5
niter_same = 5
#threshold to consider the residuals "similar enough". Default: 1E-6
eject_threshold = 1E-6
#Flag to eject from the main optimization search loop
eject = False
# Iteration counter
k = 0

# Determine number of Langmuir sites we're fitting to (only supports 1 or 2)
n_langmuir = 0
if len(terms) == 2:
    n_langmuir = 1
if len(terms) == 4:
    n_langmuir = 2

# Clear the Monitor
clear_monitor()

#################################
# MAIN OPTIMIZATION SEARCH LOOP #
#################################
while (k < niter) and (eject == False):
    timestamp = simtools.timestamp()
    print("Currently running iteration #", k, "at", timestamp)
    # Initial Nelder-Mead simplex search
    res = scipy.optimize.minimize(fun2dv, terms, args=(inputbundle["optparams"],
                                      inputbundle["conc"],
                                      inputbundle["time"],
                                      method=optmethod, callback=callme,
                                      options={'maxiter': maxiter, 'fatol': fatol,})
    print("Simplex evaluations:", res.nit)

    # Retrieve Nelder-Mead results & rerun fun2dv once to get the residuals result
    terms[0] = res.x[0]  # A1
    terms[1] = res.x[1]  # B1
    if n_langmuir == 2:
        # Load 2nd B-term and check that B-terms are ordered greatest to smallest
        if terms[1] < terms[3]:  # If B1 < B2:
            # Swap term order
            terms[0] = res.x[2]
            terms[1] = res.x[3]
            terms[2] = res.x[0]
        else:
            # Order is OK, just load in 2nd A-term
    sumsq = fun2dv(terms, inputbundle["optparams"],
                   inputbundle["conc"], inputbundle["time"],
                   inputbundle["conc"], inputbundle["time"]
                   method=optmethod, callback=callme,
                   options={'maxiter': maxiter, 'fatol': fatol,})
    simplex_sumsq = sumsq
    print("Simplex results:", terms, "Residuals:", sumsq)

#===========================#
#BEGIN HYPERCUBE GRID SEARCH#
#===========================#
# Check surrounding solution space

myterms = terms  # Initialize with the terms from Nelder-Mead

# Initializing Hypercube Grid search area with the Nelder-Mead results
if n_langmuir == 1:
    mylistresults = pd.DataFrame({"Residuals": [sumsq],
                                   "A1": [terms[0]],
                                   "B1": [terms[1]]})
elif n_langmuir == 2:
    mylistresults = pd.DataFrame({"Residuals": [sumsq],
                                   "A1": [terms[0]],
                                   "B1": [terms[1]],
                                   "A2": [terms[2]],
                                   "B2": [terms[3]]})
else:
    print("CRITICAL ERROR! Impossible number of Langmuirs.")

# Build the search area in the solution space:
if n_langmuir == 1:
    A1_range = make_range(terms[0], radius_single, points_single)
    B1_range = make_range(terms[1], radius_single, points_single)
    A2_range = [0.0]
    B2_range = [0.0]
elif n_langmuir == 2:
    A1_range = make_range(terms[0], radius_bilangmuir, points_bilangmuir)
    B1_range = make_range(terms[1], radius_bilangmuir, points_bilangmuir)
    A2_range = make_range(terms[2], radius_bilangmuir, points_bilangmuir)
    B2_range = make_range(terms[3], radius_bilangmuir, points_bilangmuir)
else:
    print("CRITICAL ERROR! Impossible number of Langmuirs.")

# Iterate through all possible combinations of these term ranges.
for myterms in itertools.product(A1_range, B1_range, A2_range, B2_range):
    mysumsq = fun2dv(myterms, inputbundle["optparams"],
                      inputbundle["conc"], inputbundle["time"])

    # Storing terms tested & resulting residuals
    if n_langmuir == 1:
        mydatum = pd.DataFrame({"Residuals": [mysumsq],
                                "A1": [myterms[0]],
                                "B1": [myterms[1]]})
    elif n_langmuir == 2:
        mydatum = pd.DataFrame({"Residuals": [mysumsq],
                                "A1": [myterms[0]],
                                "B1": [myterms[1]],
                                "A2": [myterms[2]],
                                "B2": [myterms[3]]})
    else:
        print("CRITICAL ERROR! Impossible number of Langmuirs.")

    mylistresults = mylistresults.append(mydatum, ignore_index=True)
### WARNING: GLOBAL VARIABLE USAGE IN THIS SECTION!!

**Inject simplex tracker here:**

```python
if n_langmuir == 1:
    mylistresults = mylistresults.append(langmuir_tracker, 
    ignore_index=True)

elif n_langmuir == 2:
    mylistresults = mylistresults.append(bilangmuir_tracker, 
    ignore_index=True)
```

**PURGE simplex tracker**

```python
langmuir_tracker = pd.DataFrame({
    "Residuals": [],
    "A1": [],
    "B1": []})
```

```python
bilangmuir_tracker = pd.DataFrame({
    "Residuals": [],
    "A1": [],
    "B1": [],
    "A2": [],
    "B2": []})
```

### END OF GLOBAL VARIABLE USAGE

**Find minimum from the mix of Nelder-Mead and Hypersphere searches**

```python
minindex = mylistresults.Residuals.idxmin(axis=0)
size = mylistresults.shape[0]  # Outputs number of rows
```

**Retrieve associated values to the minimum**

```python
    terms[0] = mylistresults.A1.iloc[minindex]
    if n_langmuir == 2:
    sumsq = mylistresults.Residuals.iloc[minindex]
```

```python
    print("This iteration results:", terms, "Residuals:", sumsq)
    if sumsq == simplex_sumsq:
        print("Simplex result used.")
        solver = "Simplex"
    else:
        print("Hypersphere result used.")
        solver = "Hypersphere"
```

**Update the monitor:**

```python
exp_conc = inputbundle["conc"]
exp_time = inputbundle["time"]
```
simchromatogram = forward_backward(terms, inputbundle["optparams"])
sim_time = simchromatogram["time"]
sim_conc = simchromatogram["conc"]
scaled_chromatogram = scale_chromatograms(exp_time, exp_conc,\sim_time, sim_conc)
update_monitor(scaled_chromatogram, mylistresults)

#Load, append, & store: term results & residuals
score_df = pd.read_excel(resultsfname, sheet_name="Score", index=False)
if n_langmuir == 1:
datum = pd.DataFrame({"Residuals": [sumsq],
            "A1": [terms[0]],
            "B1": [terms[1]],
            "Simplex": [res.nit],
            "Hypersphere": [size],
            "Solver": [solver]})
all_scores = score_df.append(datum, ignore_index=True)
all_scores = all_scores[["Residuals", "A1", "B1",\"Simplex", "Hypersphere", "Solver"]]
if n_langmuir == 2:
datum = pd.DataFrame({"Residuals": [sumsq],
            "A1": [terms[0]],
            "B1": [terms[1]],
            "A2": [terms[2]],
            "B2": [terms[3]],
            "Simplex": [res.nit],
            "Hypersphere": [size],
            "Solver": [solver]})
all_scores = score_df.append(datum, ignore_index=True)
all_scores.to_excel(resultsfname, sheet_name="Score",\header=True, index=False)

#Check if we can eject from the main optimization loop
#by seeing how many sumsq results are similar
#by a threshold of eject_threshold

listsumsq = all_scores.Residuals.values
minsumsq = min(listsumsq)
min_threshold = eject_threshold * minsumsq
#samecount = 0

#Only Look at the Last few results (Length equal to niter_same):
selection = listsumsq[-niter_same:]
concensus_flag = True
for selection_sumsq in selection:
    if np.abs(selection_sumsq - minsumsq) > min_threshold:
concensus_flag = False
if concensus_flag and (minsumsq < 1E+30):
    eject = True

k = k + 1

# Extract results from the scores list
minindex = all_scores.Residuals.idxmin(axis=0)
results = all_scores[minindex:minindex+1]  # Slice to a single row with the minima
iterations = pd.DataFrame({"Iterations": [len(all_scores)]}, index=[minindex])
results = pd.concat([results, iterations], axis=1)  # Add iterations to output
results = results.reset_index(drop=True)  # Resets index so it's 0

# Package up results
if n_langmuir == 1:
    fopt = results["Residuals", "Iterations", "A1", "B1"]
if n_langmuir == 2:

return fopt

def simulate(directory, inputdata, optparams):
    """
    Main simulation loop.
    Loop made to enable batch processing.
    """
    Input variables:
    directory (string) - os pathway to active directory
    inputdata (string) - experimental data file. Format "[file name].xlsx"
    optparams (parameters) - parameters class object for the simulation

    Returned variables:
    NONE

    Written to disk:
    TEMP_score.xlsx - sets up the sheet with
    the initial guess of the solution derived from ECP
    -*results.xlsx - output

    Updated: 2018/07/06
    """

    # Begin timestamp
    time_start = simtools.timestamp_excel(simtools.timestamp())

    # File-handling setup
    if inputdata[-10:] == "-data.xlsx":
        datatag = inputdata[:-10]
    else: datatag = inputdata[:-5]
    resultsfname = directory + datatag + "-results.xlsx"
    results_writer = pd.ExcelWriter(resultsfname)
# read in datapoints
inputdatafname = directory + inputdata # input data file info
times = []
response = []

# Read in datapoints
experimental_df = pd.read_excel(inputdatafname, header=None, names=list('AB'))
times = experimental_df.A.values
response = experimental_df.B.values

# Convert to concentration
concentrations = []
for i in range(len(response)):
    concentrations.append((response[i] - optparams.Calparam0) / optparams.Calparam1)

# Shift baseline to average of last 10 seconds of file (assumed to be zero)
index_size = 200 # Last 10 seconds, at a sampling rate of 20 Hz
conc_end = np.mean(concentrations[-index_size:-1])
conc_start = np.mean(concentrations[0:index_size])
delta = conc_end - conc_start
con_max = max(concentrations)
threshold = conc_max * 0.01
if delta > threshold:
    print("Baseline drift is too great."
    # Do something like abort maybe?

# Remove negatives from the chromatogram
# ECP method throws error if negative values are present
for i in range(len(concentrations)):
    concentrations[i] = concentrations[i] - conc_end
    if concentrations[i] < 0.0:
        concentrations[i] = 0.0

# Integrate
area = 0.
dt = times[1] - times[0]
for i in range(len(concentrations) - 1):
    area = area + ((concentrations[i] + concentrations[i + 1]) / 2.0) * dt
optparams.area = area

# Determine peak maxima
conmax = 0.0
timemax = 0.0
timemin = 0.0
for i in range(len(concentrations)):
    if concentrations[i] > conmax:
        conmax = concentrations[i]
imax = i
timemax = times[i]

#############################################################################
# ABORT CHECK: Check if peak time is before void time: #
#############################################################################
if timemax < optparams.t0:
    #Gather information to give to the "abort()" function
    print('\n', "COLLECTING ABORT STATS", '\n')
    results_writer.save()
    results_writer.close()
    time_end = simtools.timestamp_excel(simtools.timestamp())
    time_elapsed = time_end - time_start
    optparams.time_start = time_start
    optparams.time_end = time_end
    optparams.time_elapsed = time_elapsed
    optparams.abort_flag = True
    optparams.TR = timemax
    print('\n', "REQUESTING ABORT PROTOCOL", '\n")
    simtools.abort(resultsfname, optparams)
    return

#Determine baseline - threshold version
threshold_trigger = 0.05
threshold_conc = conmax * threshold_trigger
for i in range(len(times)-1, 0, -1): #Backwards counting list
    if concentrations[i] > threshold_conc:
        timemin = times[i]
        imin = i
        break

#Convert time to specific volume
package = specific_volume(times, concentrations, optparams, imax, imin)
#Unpack returned items:
vn = package["specific_volumes"]
cn = package["specific_concentrations"]
dptsvn = len(vn)

#Set curve resolution to fixed number
dptst = 1000 #EDIT: Increased to 1000 on 2018/04/17
vnnew = np.linspace(vn[0], vn[-1], dptst)
cnnew = np.linspace(0, 0, dptst)
smoothCN = savgol_filter(cn, window_length=31, polyorder=2, deriv=0)

#select new points on curve
if np.isnan(cnnew[dptst-1]):
    cnnew[dptst-1] = 0.0
del dptsvn
del response
# Write specific volume to the "-results.xlsx" file
specificvol_df = pd.DataFrame({"Concentration": cnnew,
   "Specific Volume": vnnew})
specificvol_df.to_excel(results_writer, sheet_name="Specific Volume",
   header=True, index=False)

# Make ECP-based first guess
dptst = optparams.N
q = []
for i in range(0, dptst):
    q.append(0.)
# Integrate
q[-1] = 0.0
j = len(q) - 2
while j >= 0:
    q[j] = q[j+1] + (cnnew[j]-cnnew[j+1])*(vnnew[j+1] + vnnew[j])/2.0
    j = j - 1
kprime = (timemin-optparams.t0)/optparams.t0

terms = []
terms.append(0.)
terms.append(0.)
A1 = kprime/optparams.Phi
B1 = A1/q[0]
i = 0
terms[i] = A1
terms[i+1] = B1

# Quick check to see what the residual with this would be
ecp_residual = fun2dv(terms, optparams, concentrations, times)
print("ECP Results:
   "Terms:", terms,
   "Residuals", ecp_residual)

# Make file to track the score. Add in first guess.
""
# NOTE: This is done because I cannot pass writer to Hypersphere() or
write to existing "-_results.xlsx" file.
Will have to instead save to a temporary "TEMP_score.xlsx" file,
and then extract that information using the results_writer object.
""
scorefname = directory + "TEMP_score.xlsx"
score_df = pd.DataFrame({"Residuals": [ecp_residual],
   "A1": [A1],
   "B1": [B1],
   "Simplex": [0],
   "Hypersphere": [0],
   "Solver": [0]})
score_df = score_df[['Residuals', 'A1', 'B1', 'Simplex', 'Hypersphere', 'Solver']]
score_df.to_excel(scorefname, sheet_name='Score', header=True, index=False)

# Optimization parameters setup
inputbundle = (optparams, concentrations, times)  # BRANCH: directfit change
inputbundle = {"optparams": optparams, "conc": concentrations, "time": times}
optmethod = "Nelder-Mead"

#========== Main simulation function call here ==============
fopt = Hypersphere(scorefname, terms, optmethod, inputbundle)

# Recover Score information from "TEMP_score.xlsx" & add to "-results.xlsx" file
score_df = pd.read_excel(scorefname, sheet_name="Langmuir Score", index=False)
score_df.to_excel(results_writer, sheet_name="Langmuir Score", header=True, index=True)

# Unpack fit results from Hypersphere()
sumsq = fopt.at[0, "Residuals"]
A1 = fopt.at[0, "A1"]
B1 = fopt.at[0, "B1"]
optparams.iterations_langmuir = fopt.at[0, "Iterations"]
print("Langmuir fit results:", A1, B1, sumsq)

# Write fit results to "-results.xlsx"
fit_df = pd.DataFrame({"Residuals":[sumsq], "A1":[A1], "B1":[B1]})
fit_df.to_excel(results_writer, sheet_name="Langmuir Fit", header=True, index=False)

# Forward-backward discretization of final output
finalterms = [A1, B1]
simchromatogram = forward_backward(finalterms, optparams)
time = simchromatogram["time"]
ccc = simchromatogram["conc"]

# Print using lower frequency
ftime_print = []
if times[len(times)-1] > time[len(time)-1]:
    fdp = len(time)
    endtime = time[fdp-1]
    time_print.append(0.)
    i = 0
    ithtime = 0.0
    while ithtime <= endtime:
        i = i + 1
time_print.append(time_print[i-1]+dt)
ithtime = time_print[i]
else:
    fdp = len(times)
    endtime = times[fdp-1]
    time_print.append(0.0)
    i = 0
    ithtime = 0.0
    while ithtime <= endtime:
        i = i + 1
        time_print.append(time_print[i-1]+dt)
        ithtime = time_print[i]
    expC = interpolate(times, concentrations, time_print)
    csim = interpolate(time, ccc, time_print)

#Print out chromatograms
chromatograms_df = pd.DataFrame({"Time": time_print,
                                "Experimental": expC,
                                "Simulated": csim})

#Added to preserve column order
chromatograms_df = chromatograms_df[["Time", "Experimental", "Simulated"]]
chromatograms_df.to_excel(results_writer, sheet_name="Langmuir Chromatogram",
                           header=True, index=False)

#=============
#========== BEGIN BI-LANGMUIR TESTS =========
#=============

#Make a first guess for the Bi-Langmuir Fit
ratio = 0.55
A1 = A1 * ratio
B1 = B1 * ratio
A2 = A1 * (1 - ratio)
B2 = B1 * (1 - ratio)
terms = [A1, B1, A2, B2]

guess_residual = fun2dv(terms, optparams, concentrations, time_print)

#Make file to track the score. Add in first guess.
"""
#NOTE: This is done because I cannot pass writer to Hypersphere()
#or write to existing -*results.xlsx* file.
#Will have to instead save to a temporary -*TEMP_score.xlsx* file,
#and then extract that information using the results_writer object.
"""

scorefname = directory + "TEMP_score.xlsx"
score_df = pd.DataFrame({"Residuals": [guess_residual],
                        "A1": [A1],
                        "B1": [B1],
                        "A2": [A2],
"""
"B2": [B2],
"Simplex": [0],
"Hypersphere": [0],
"Solver": [0])

score_df = score_df[['Residuals', 'A1', 'B1', 'A2', 'B2',
"Simplex", "Hypersphere", "Solver"]]

score_df.to_excel(scorefname, sheet_name="Score",
header=True, index=False)

# Optimization parameters setup
inputbundle = (optparams, concentrations, times)

inputbundle = {
"optparams": optparams, "conc": concentrations, "time": times
}

optmethod = "Nelder-Mead"

# Main simulation function call here
fopt = Hypersphere(scorefname, terms, optmethod, inputbundle)

# Recover Score information from "TEMP_score.xlsx" & add to "*_results.xlsx" file
score_df = pd.read_excel(scorefname, sheet_name="Score", index=False)
score_df.to_excel(results_writer, sheet_name="BiLangmuir Score",
header=True, index=True)

# Unpack fit results from Hypersphere()

sumsq = fopt.at[0, "Residuals"]
A1 = fopt.at[0, "A1"]
B1 = fopt.at[0, "B1"]
A2 = fopt.at[0, "A2"]
B2 = fopt.at[0, "B2"]

optparams.iterations_bilangmuir = fopt.at[0, "Iterations"]

# Write fit results to "*_results.xlsx"
fit_df = pd.DataFrame(
{"Residuals": [sumsq],
"A1": [A1],
"B1": [B1],
"A2": [A2],
"B2": [B2]})

fit_df.to_excel(results_writer, sheet_name="BiLangmuir Fit",
header=True, index=False)

# Forward-backward discretization of final output
finalterms = [A1, B1, A2, B2]
simchromatogram = forward_backward(finalterms, optparams)
time = simchromatogram["time"]
ccc = simchromatogram["conc"]

def time_print():
    times_len = len(times)
    if times[times_len-1] > time[time_len-1]:
        fdp = len(time)
        endtime = time[fdp-1]
        time_print(0.)
i = 0
ithtime = 0.0

while ithtime <= endtime:
    i = i + 1
    time_print.append(time_print[i-1]+dt)
    ithtime = time_print[i]

else:
    fdp = len(times)
    endtime = times[fdp-1]
    time_print.append(0.0)
    ithtime = 0.0
    while ithtime <= endtime:
        i = i + 1
        time_print.append(time_print[i-1]+dt)
        ithtime = time_print[i]

expC = interpolate(times, concentrations, time_print)
csim = interpolate(time, ccc, time_print)

#Print out chromatograms
chromatograms_df = pd.DataFrame({'Time': time_print,
                                  'Experimental': expC,
                                  'Simulated': csim})

#Added to preserve column order
chromatograms_df = chromatograms_df[['Time', 'Experimental', 'Simulated']]
chromatograms_df.to_excel(results_writer, sheet_name="BiLangmuir Chromatogram",
                           header=True, index=False)

#Saving parameters used for this file

time_end = simtools.timestamp_excel(simtools.timestamp())
time_elapsed = time_end - time_start
if time_elapsed < 0.0:
    time_elapsed = 1.0 + time_elapsed #Converts it into a positive.

optparams.time_start = time_start
optparams.time_end = time_end
optparams.time_elapsed = time_elapsed
optparams.abort_flag = False
optparams.tR = timemax

stats = optparams.report()

parameters_df = pd.DataFrame({'Parameters': stats['label'],
                              'Values': stats['values']})
parameters_df.to_excel(results_writer, sheet_name="Parameters",
                        header=False, index=False)

print("Finished simulating file. Results written to", resultsfname)
def add_summary_sheet(summary_writer, file_sublist, ws_title):
    """
    Converts .csv files to .xlsx files with two columns of data.
    Will read column A (used for time) and column data_column (used for data)
    and write just these columns of data to a new .xlsx file.
    """
    gap = pd.DataFrame({'-': []})
    sheet = pd.DataFrame({})
    for index, fname in enumerate(file_sublist):
        inputfname = directory + fname
        print("Processing: ", fname)
        df = pd.read_csv(inputfname, header=None)
        sheet = pd.concat([sheet, df, gap], axis=1)
    sheet.to_excel(summary_writer, sheet_name=ws_title, header=None, index=False)

def determine_averages(scores):
    """
    Automatically generates a summary of the average tR of the files.
    """
    Input variables:
scores (dataframe): Pandas dataframe object with the scores

Output variables:
- averages (dataframe): Pandas dataframe object with these values:
  - Line
  - Vial
  - tR

Written: 2019/01/10
Updated: 2019/01/25

# Break "File ID" into vial and line columns

# First add Vial and Line columns to the scores DataFrame
scores.loc[:, "Vial"] = pd.Series(0, index=scores.index)
scores.loc[:, "Line"] = pd.Series(0, index=scores.index)

for row in scores.itertuples():
    fileID = row.Filename
    vial = int(fileID[1:3])
    line = int(fileID[4:-2])
    scores.at[row.Index, "Vial"] = vial
    scores.at[row.Index, "Line"] = line

# Get a list of the line entries
print(scores)

# Generate a list of the unique line members
line_list = []
for row in scores.itertuples():
    if row.Line not in line_list:
        line_list.append(row.Line)
p

print(line_list)

averages = pd.DataFrame({"Line":[],
                          "Vial":[],
                          "tR":[],
                          "Height":[]})

# Calculate average values for each line entry
for line in line_list:
    # Initialize Temporary List
    tR = []
    height = []

    # Iterate through the allstats DataFrame collecting stats for this line
    for row in scores.itertuples():
        if line == row.Line:
            tR.append(row.Max)
            height.append(row.Height)
            vial = row.Vial

    # Dump in averages for this Line

76
averages_datum = pd.DataFrame({
    "Line": [line],
    "Vial": [vial],
    "tR": [np.mean(tR)],
    "Height": [np.mean(height)]
})

averages = averages.append(averages_datum, ignore_index=True)

averages = averages[['Line', 'Vial', 'tR', 'Height']]

return averages

#==============================================================================
#=========================== START PROGRAM ========
#Directory & batch setup
directory = simtools.set_directory(subdirectory="!Conversion Space")
batch = simtools.set_batchparams(directory)

#Collect list of .csv files to summarize
file_list = [f for f in os.listdir(directory) if f.endswith('.csv')]
lines_list = []
for fname in file_list:
    fname_line = fname[4:-6]
    if fname_line not in lines_list:
        lines_list.append(fname_line)

#Setup output file
experimentname = batch[11:-5]
date = ' - v' + dt.datetime.today().strftime('%Y%m%d')
outputfname = directory + "Summary" + experimentname + date + "xlsx"
print("Generating:", outputfname)
summary_writer = pd.ExcelWriter(outputfname)

#Load in batch info
batchfname = directory + batch
batchinfo = pd.read_excel(batchfname, index_column=0)

#Generate scores dataframe (empty)
scores = pd.DataFrame({"Filename": [],
    "MeOH": [],
    "Max": [],
    "Height": [],
    "End": []})

#Comparative chromatogram holder
chromcompare = pd.DataFrame({})

for line in lines_list:
    #Extract info for this Line
    line_index = int(line)
print(line_index)
label = batchinfo.at[line_index, "Label"]
meoh = batchinfo.at[line_index, "MeOH"]
data_column = batchinfo.at[line_index, "DataColumn"]

# List the files associated with this line
file_sublist = []
for fname in file_list:
    fname_line = fname[4:-6]
    if fname_line == line:
        file_sublist.append(fname)
print(file_sublist)

# Add in chromatograms for each line
gap = pd.DataFrame({'-': []})
sheet = pd.DataFrame({})
for index, fname in enumerate(file_sublist):
    inputfname = directory + fname
    replicate = int(fname[6:-4])
    print("Processing:", fname)
    df = pd.read_csv(inputfname, header=None)
    columnnames = ['Time', 'Signal']
    chrom = pd.read_csv(inputfname, header=None,
                        usecols=[0, data_column], names=columnnames)

    # Find the peak maxima. Since the injection artifact is usually around t0,
    # let's only look for peaks after that.
    dt = chrom.at[5, "Time"] - chrom.at[4, "Time"]
    crop_index = int(0.1 / dt)
    print("Cropping from:", crop_index, "dt:", dt)
    chrom_cropped = chrom[crop_index:]
    max_index = chrom_cropped.Signal.idxmax(axis=0)
    print(max_index)
    height = chrom.Signal.iloc[max_index]
    time_max = chrom.Time.iloc[max_index]
    end_time = chrom.Time.iloc[-1]

    datum = pd.DataFrame({'Filename': [fname[:-4]],
                          'MeOH': [meoh],
                          'Max': [time_max],
                          'Height': [height],
                          'End': [end_time]})
    scores = scores.append(datum, ignore_index=True)

    sheet = pd.concat([sheet, df, gap], axis=1)

if replicate == 1:
chromcompare = pd.concat([chromcompare, df, gap], axis=1)

sheet.to_excel(summary_writer, sheet_name=label, header=None, index=False)

#Dump scores dataframe into the summary file
scores.to_excel(summary_writer, sheet_name="Summary", index=False)

#Create averages from the scores dataframe
averages = determine_averages(scores)
averages.to_excel(summary_writer, sheet_name="Averages", index=False)

chromcompare.to_excel(summary_writer, sheet_name="Chromatogram Comparison",
header=None, index=False)

summary_writer.save()

del summary_writer

command = 'start EXCEL.EXE ' + outputfname + '

os.system(command)

print("Finished")
# -*- coding: utf-8 -*-

Created on Wed May  9 12:35:48 2018

@author: Mark T. Schnepper

import os
import pandas as pd
import numpy as np
import simtools

def peak_chord(chromatogram, tR_index, height):
    """
    Finds the indices for the points to the left and right of the peak maximum at tR_index at the specified height (must be <1)
    
    passed variables:
    - chromatogram (DataFrame): chromatogram, with columns Time & Signal
    - tR_index (int): index for the time at which the peak elutes
    """

def determine_stats(directory, fname):
    """
    Calculates the chromatographic stats associated with the input file for void elutions:
    - N: # theoretical plates, calculated by the void time & FWHM
    - Void time: Set to the peak maxima, tR
    - FWHM: Full width at half-max
    - Asymmetry: Calculated at the 10% height level
    
    Passed variables:
    - directory (string): os pathwaay to the active directory
    - fname (string): file name

    Returned variables:
    - stats (dataframe): PANDAS dataframe of stats associated with this file:
        - File ID: tag for the file
        - N : # of theoretical plates
        - Void Time: Elution time at the peak maxima
        - FWHM: Full width at half-max
        - Asymmetry: Calculated at the 10% height level

    Written to disk:

"""
inputfname = directory + fname
tag = fname[:-4]
columns = ['Time', 'Signal']
chromatogram = pd.read_csv(inputfname, header=None, names=columns)
print(chromatogram)

max_index = chromatogram.Signal.idxmax()
max_signal = chromatogram.at[max_index, 'Signal']
tR = chromatogram.at[max_index, 'Time']

#Find FWHM:
threshold = 0.5
threshold_signal = threshold * max_signal

#Find leftside index:
left_index = max_index
left_signal = max_signal
while left_signal > threshold_signal:
    left_index = left_index - 1
    left_signal = chromatogram.at[left_index, 'Signal']
left_time = chromatogram.at[left_index, 'Time']

#Find rightside index:
right_index = max_index
right_signal = max_signal
while right_signal > threshold_signal:
    right_index = right_index + 1
    right_signal = chromatogram.at[right_index, 'Signal']
right_time = chromatogram.at[right_index, 'Time']
FWHM = right_time - left_time

#Find asymmetry factor at 10% max height:
threshold = 0.1
threshold_signal = threshold * max_signal

#Find a-side index:
a_index = max_index
a_signal = max_signal
while a_signal > threshold_signal:
    a_index = a_index - 1
    a_signal = chromatogram.at[a_index, 'Signal']
a_time = chromatogram.at[a_index, 'Time']

#Find b-side index:
b_index = max_index
b_signal = max_signal
while b_signal > threshold_signal:
    b_index = b_index + 1
    b_signal = chromatogram.at[b_index, 'Signal']
b_time = chromatogram.at[b_index, 'Time']
a_chord = tR - a_time
b_chord = b_time - tR
asymmetry = b_chord/a_chord

N = 5.54*((tR/FWHM)**2)

datum = pd.DataFrame(
    {
        "File_ID": [tag],
        "Stats_N": [N],
        "t0": [tR],
        "FWHM": [FWHM],
        "Asymmetry": [asymmetry]
    }
)

return datum

def determine_moments(directory, fname):
    """
    Calculates the chromatographic moments associated with
    the input file for void elutions.
    m0 - peak area (integral of signal over time)
    m1 - mean (integral of time*signal over time, divided by m0)
    m2 - variance (integral of (t-m1)^2*signal over time, divided by m0)
    N: # theoretical plates, calculated by m1^2/m2
    Void time: Set to the peak maxima, tR

    Threshold for the integration is set at 5% peak max height.

    Passed variables:
    directory (string): os pathway to the active directory
    fname (string): file name

    Returned variables:
    stats (dataframe): PANDAS dataframe of moments associated with this file:
    File ID: tag for the file
    N : # of theoretical plates
    Void Time: Elution time at the peak maxima
    M0: peak area
    M1: mean
    M2: variance

    Written to disk:
    NONE
    """

    inputfname = directory + fname
    tag = fname[: -4]
    print(tag)

    columns = ['Time', 'Signal']
    chromatogram = pd.read_csv(inputfname, header=None, names=columns)
time = chromatogram.Time.values
signal = chromatogram.Signal.values

#Bump up baseline
min_signal = chromatogram.Signal.min()
for i in range(len(signal)):
    signal[i] = signal[i] - min_signal

max_index = chromatogram.Signal.idxmax()
max_signal = chromatogram.at[max_index, 'Signal']
tR = chromatogram.at[max_index, 'Time']

#Find left & right indices at 5% peak height level
threshold = 0.05
threshold_signal = threshold * max_signal

#Find leftside index:
left_index = max_index
left_signal = max_signal
while left_signal > threshold_signal:
    left_index = left_index - 1
    left_signal = chromatogram.at[left_index, 'Signal']
left_time = chromatogram.at[left_index, 'Time']

#Find rightside index:
right_index = max_index
right_signal = max_signal
while right_signal > threshold_signal:
    right_index = right_index + 1
    right_signal = chromatogram.at[right_index, 'Signal']
right_time = chromatogram.at[right_index, 'Time']

#Find statistical moments:
dt = chromatogram.at[1, 'Time'] - chromatogram.at[0, 'Time']
M0 = 0.0 #Area under curve
for i in range(left_index, right_index):
    signal = chromatogram.at[i, 'Signal']
    M0 = M0 + signal*dt

M1 = 0.0 #Mean
for i in range(left_index, right_index):
    signal = chromatogram.at[i, 'Signal']
    time = chromatogram.at[i, 'Time']
    M1 = M1 + signal*time*dt
M1 = M1 / M0

M2 = 0.0 #Variance
for i in range(left_index, right_index):
    signal = chromatogram.at[i, 'Signal']
    time = chromatogram.at[i, 'Time']
    M2 = M2 + ((time - M1)**2) * signal * dt
M2 = M2 / M0

N = ((M1)**2)/M2
stats = pd.DataFrame({'File_ID': [tag],}
"Moments_N":[N],
"M0":[M0],
"M1":[M1],
"M2":[M2])

print(stats)

return stats

def determine_averages(stats, moments):
    ""
    Automatically generates a summary of the average N and t0 of the files.
    ""
    #Input variables:
    #stats (dataframe): Pandas dataframe object with the stats-based values
    #moments (dataframe): Pandas dataframe object with the moments-based values
    #Output variables:
    #summary (dataframe): Pandas dataframe object with these values:
    Line
    Vial
    Stats N
t0
    FWHM
    Asymmetry
    Moments N
    M0
    M1
    M2
    ""

    #Merge dataframes
    allstats = stats.merge(moments)
    print(allstats)

    #Break "File ID" into vial and line columns
    allstats.loc[:, "Vial"] = pd.Series(0, index=allstats.index)
    allstats.loc[:, "Line"] = pd.Series(0, index=allstats.index)

    for row in allstats.itertuples():
        fileID = row.File_ID
        vial = int(fileID[1:3])
        line = int(fileID[4:-2])
        allstats.at[row.Index, "Vial"] = vial
        allstats.at[row.Index, "Line"] = line

    #Get a list of the Line entries
    print(allstats)
# Generate a list of the unique Line members
line_list = []
for row in allstats.itertuples():
    if row.Line not in line_list:
        line_list.append(row.Line)
print(line_list)

averages = pd.DataFrame({
    "Line": [],
    "Vial": [],
    "Stats_N": [],
    "t0": [],
    "FWHM": [],
    "Asymmetry": [],
    "Moments_N": [],
    "M0": [],
    "M1": [],
    "M2": []})

# Calculate average values for each Line entry
for line in line_list:
    # Initialize temporary lists
    Stats_N = []
    t0 = []
    FWHM = []
    Asymmetry = []
    Moments_N = []
    M0 = []
    M1 = []
    M2 = []
    # Iterate through the allstats DataFrame collecting stats for this line
    for row in allstats.itertuples():
        if line == row.Line:
            Stats_N.append(row.Stats_N)
            t0.append(row.t0)
            FWHM.append(row.FWHM)
            Asymmetry.append(row.Asymmetry)
            Moments_N.append(row.Moments_N)
            M0.append(row.M0)
            M1.append(row.M1)
            M2.append(row.M2)
    vial = row.Vial
    # Dump in averages for this line
    averages_datum = pd.DataFrame({
        "Line":[line],
        "Vial":[vial],
        "Stats_N":[np.mean(Stats_N)],
        "t0":[np.mean(t0)],
        "FWHM":[np.mean(FWHM)],
        "Asymmetry":[np.mean(Asymmetry)],
        "Moments_N":[np.mean(Moments_N)],
        "M0":[np.mean(M0)],
        "M1":[np.mean(M1)],
        "M2":[np.mean(M2)]})
averages = averages.append(averages_datum, ignore_index=True)

averages = averages[['Line', 'Vial', 'Stats_N', 't0', 'FWHM', 'Asymmetry',
                     'Moments_N', 'M0', 'M1', 'M2']]
outputdata = str(input("Name output file: "))
outputfname = directory + outputdata + ".xlsx"

excelwriter = pd.ExcelWriter(outputfname)
stats.to_excel(excelwriter, sheet_name='ChromStats')
moments.to_excel(excelwriter, sheet_name='ChromMoments')
summary.to_excel(excelwriter, sheet_name='Summary')
delexcelwriter.save()

command = 'start EXCEL.EXE ' + outputfname + '"

os.system(command)

print("Finished")
import os
import pandas as pd

def set_directory():
    """Sets the input directory used for other functions."
    Global variables used:
    directory
dir_list
    """
    #global variables
goal directory
    #local variables
dir_set = -1
    path_exists = "Not Set"
    number_found = 0
    #requests user input to select the directory
    while True:
        print("Known Directories:")
        i = 1
        for Name, Path in dir_list:
            print(i, Name, Path)
            i += 1
        user_input = input("Enter number of directory: ")
        if user_input.isdigit():
            user_dir = dir_list[int(user_input) - 1]
            path_exists = user_dir[1]
            number_found = 1
            break
        else:
            print("Invalid input. Please enter a number.")
    return path_exists

#global variables
directory = "Not Set"
dir_list = (("Jeff Work", "C:/Users/jroles/My Documents/Simulated Chromatograms/"),
            ("Jeff Home", "C:/JEFF HOME"),
            ("Mark Work", "C:/Users/mark/Desktop/Python Work Space/!Convers...")
inputdata = "ExpChrom"
inputparams = "ExpChromParams"
batchparams = "BatchParams"
if os.path.isdir(Path):
    path_exists = "FOUND"
    number_found += 1
directory = Path
else:
    path_exists = "-"
print(i, ":", Name, ":", path_exists)
i += 1
if number_found == 1:
    print("Using: ", directory)
    break
if number_found > 1:
    dir_set = int(input("Which directory?: "))
    # test if the input is a valid index
    if dir_set > 0 and dir_set <= len(dir_list):
        directory = dir_list[dir_set - 1][1]
        # checking if user-selected directory actually exists
        if os.path.isdir(directory):
            print("Directory set to: ", directory)
            break
        else:
            print("Directory not found. Try again.")
    else:
        print("Please enter a valid number.")
if number_found == 0:
    print("ERROR: No valid directory found")
    break

def csv_to_xlsxSelective(inputfile, data_column):
    """
    Converts .csv files to .xlsx files with two columns of data.
    Will read column A (used for time) and column data_column (used for data)
    and write just these columns of data to a new .xlsx file.
    """
    inputfname = directory + inputfile
    outputfile = inputfile[:-4]
    outputfname = directory + outputfile + ":-data.xlsx"
    df = pd.read_csv(inputfname, header=None, usecols=[0, data_column])
    print(df)
    df.to_excel(outputfname, sheet_name="ExpChrom", header=None, index=False)
#START PROGRAM

```python
set_directory()
file_list = [f for f in os.listdir(directory) if f.endswith('.csv')]
data_column = int(input("Keep which data column?:"))
for fname in file_list:
    csv_to_xlsx_selective(fname, data_column)

print("Finished")
```

179
def subtract_blank(chromatogram, blank_chromatogram):
    ""
    This function removes the blank from the chromatogram.
    Assumptions: It's assumed that the time indicies
    match the chromatogram being modified.
    Input variables:
    chromatogram (dataframe) - the chromatogram being modified
    blank_chromatogram (dataframe) - the blank chromatogram
    Returned variables:
    chromatogram (dataframe) - the chromatogram, minus the signal from the blank
    Written to disk:
    NONE
    Last Updated: 2018/04/09
    ""
    print("Subtracting Blank...")

    # Check for the "No Blank" case
    if blank_chromatogram.at[0, "Time"] == -1:
        return chromatogram

    new_chromatogram = pd.DataFrame({"Signal": chromatogram.Signal -
                                    blank_chromatogram.Signal})
    chromatogram.update(new_chromatogram)
    return chromatogram

def get_blank(directory, batchinfo, line):
    ""
    This function finds the blank listed in batchinfo & loads it into a dataframe.
**Input variables:**
- directory (string) - os pathway to the directory being used
- batchinfo (dataframe) - batch information
- line (integer) - line to index from the batchinfo to get the blank file name

**Returned variables:**
- blank_chromatogram (dataframe) - chromatogram for the blank.
  It's assumed that the time indices match the chromatogram being modified.

**Written to disk:**
NONE

**Last updated:** 2018/04/09

```python
blank = batchinfo.at[line, "Blank"]
if blank == 0:
    blank_chromatogram = pd.DataFrame({"Time":[-1], "Signal":[-1]})
    return blank_chromatogram
blankfname = directory + blank + "-data.xlsx"
columns = ["Time", "Signal"]
blank_chromatogram = pd.read_excel(blankfname, header=None, names=columns)
return blank_chromatogram
```

```python
def remove_artifact(chromatogram, batchinfo, line):
    print("Removing artifacts...")
    #Get time information
```

**Input variables:**
- chromatogram (dataframe) - chromatogram to be modified
- batchinfo (dataframe) - batch information
- line (integer) - line to index from the batchinfo to get the void time

**Returned variables:**
- chromatogram (dataframe) - modified chromatogram

**Written to disk:**
NONE

**Last updated:** 2018/10/22

```python
print("Removing artifacts...")
```
t0 = batchinfo.at[line, "Void"]
t0_index = len(chromatogram[chromatogram["Time"] <= t0])
dt = chromatogram.at[1, "Time"] - chromatogram.at[0, "Time"]
peak = batchinfo.at[line, "PeakTime"]
peak_index_temp = len(chromatogram[chromatogram["Time"] <= peak])
print(peak_index_temp, peak)

peak_signal_temp = chromatogram.at[peak_index_temp, "Signal"]

# Smooth out chromatogram with a low-pass filter
smooth_me = chromatogram.Signal
filter_time = 1.0 / 60  # Time in minutes to filter
filter_size = int(filter_time / dt)
if filter_size < 3:
    filter_size = 3
if (filter_size % 2) == 0:  # Check if even
    filter_size = filter_size + 1  # Make it odd
smoothed = savgol_filter(smooth_me, window_length=filter_size, polyorder=2, deriv=0)
smoothed_df = pd.DataFrame({"Signal": smoothed})
chromatogram.update(smoothed_df)

# Check around preliminary peak for a better peak candidate.
region = 30.0 / 60  # Number of minutes radius to search around tR
region_index = int(region / dt)
search_start_index = peak_index_temp - region_index
search_end_index = peak_index_temp + region_index
search_segment = chromatogram[search_start_index:search_end_index]
peak_signal = search_segment.Signal.max()
peak_index = search_segment.Signal.idxmax()
print("Peak update:
",
    peak_index_temp, '{:.2f}'.format(peak_index_temp), '\n',
    peak_index, '{:.2f}'.format(peak_signal))

# Set up region to remove around the void time
# USE: 18/60 for 30 mm columns, and 30/60 for 250 mm columns.
region = 16.0 / 60  # Number of minutes radius to remove around t0
region_index = int(region / dt)

# Setup starting point for removal (arbitrary seconds before t0)
start_index = t0_index - region_index
if start_index < 1:
    start_index = int((3.0 / 60) / dt)
    print(start_index)
    if start_index < 1:
        start_index = 1
start_signal = chromatogram.at[start_index, "Signal"]

# Establish baseline level
baseline = chromatogram[:start_index]
baseline_signal = baseline.mean(axis=0)
print("BASELINE SIGNAL:", '{:.2f}'.format(baseline_signal.Signal))
new_chromatogram = pd.DataFrame({"Signal": chromatogram.Signal - \  baseline_signal.Signal})

chromatogram.update(new_chromatogram)

#Establish baseline level across the whole chromatogram
baseline_start = chromatogram[0:start_index]
last_region = 60.0 / 60 #number of minutes for the end baseline
last_index_size = int(last_region / dt)
baseline_end = chromatogram[-last_index_size:]
x_diff = len(chromatogram)
y_diff = baseline_start.mean(axis=0) - baseline_end.mean(axis=0)
slope = y_diff / x_diff
print("Baseline slope:", slope)

baseline_indices = list(range(0, len(chromatogram)))
baseline_start_signal = baseline_start.mean(axis=0)
baseline_end_signal = baseline_end.mean(axis=0)
print(baseline_start_signal, baseline_end_signal)

baseline_values = np.linspace(baseline_start.Signal.mean(axis=0),
    baseline_end.Signal.mean(axis=0),
    len(chromatogram))

baseline_signal = pd.DataFrame({"Signal": baseline_values})

new_chromatogram = pd.DataFrame({"Signal": chromatogram.Signal - \  baseline_signal.Signal})

chromatogram.update(new_chromatogram)

#Update peak signal value
peak_signal = chromatogram.at[peak_index, "Signal"]

#Update start signal value
start_signal = chromatogram.at[start_index, "Signal"]

#Preliminary ending point for removal (arbitrary seconds after t0)
end_index = t0_index + region_index

#check if end_index comes after peak
if end_index > peak_index:
    #If so, then set the end_index to right before the peak
    end_index = peak_index - 1

end_signal = chromatogram.at[end_index, "Signal"]

#Setup ending point for removal (min between starting point for removal and peak)
#Get the minimum between end_index and peak max:
#Segment only used to check between t0 and peak
segment = chromatogram[end_index:peak_index]
print(segment)
min_signal = segment.Signal.min()
min_index = segment.Signal.idxmin()}
end_signal = min_signal
end_index = min_index

# If min_signal isn't between start_signal and 0.5% of peak, # then move until we're in that range.
threshold = 0.005
threshold_signal = threshold * peak_signal
print("Threshold signal: ", '{:.2f}'.format(threshold_signal))
if threshold_signal < 0.01:
    threshold_signal = 0.01

print("Start", start_index, '{:.2f}'.format(start_signal),
      "\nEnd", end_index, '{:.2f}'.format(end_signal),
      "\nThreshold", '{:.2f}'.format(threshold_signal))

# Move forward towards peak until it's at or above start_signal value
move_forward_flag = False
if end_signal < start_signal:
    move_forward_flag = True
    print(start_signal, end_signal)
    print("Moving forward...")
while move_forward_flag:
    end_index = end_index + 1
    end_signal = chromatogram.at[end_index, "Signal"]
    if end_signal > start_signal:
        segment = chromatogram[end_index:peak_index]
        min_signal = segment.Signal.min()
        # print(min_signal)
        if min_signal > start_signal:
            move_forward_flag = False
        if end_index > peak_index:
            print("Went too far!")
    move_forward_flag = False

print("Start", start_index, '{:.2f}'.format(start_signal),
      "\n", "End", end_index, '{:.2f}'.format(end_signal))

# Move backwards from peak until it's below the threshold
locmin_signal = peak_signal * 0.90
locmin_index = -1
locmin_flag = False
while end_signal > threshold_signal:
    end_index = end_index - 1
    end_signal = chromatogram.at[end_index, "Signal"]
    print(end_index, '{:.2f}'.format(end_signal))
    # Check for a local minimum
    past_signal = chromatogram.at[end_index + 1, "Signal"]
    if past_signal < end_signal and past_signal < locmin_signal:
        locmin_signal = past_signal
        locmin_index = end_index + 1
        print("Local minimum found at", locmin_index,
              '{:.2f}'.format(locmin_signal))
    locmin_flag = True
#Check that the end_signal isn't worse than the one ahead of it.
next_index = end_index + 1
if end_signal < 0.0:
    end_index = next_index
    end_signal = start_signal

print("Start", start_index, start_signal,
    \"\n", "End", end_index, end_signal,
    \"\n", "Threshold", threshold_signal)
replacement_indices = list(range(start_index, end_index))
replacement_values = np.linspace(start_signal, end_signal,
    len(replacement_indices))
replacement_chrom = pd.DataFrame({"Signal": replacement_values},
    index=replacement_indices)
chromatogram.update(replacement_chrom)

#Smooth segment between end_index and locmin_index
#if a local minimum has been found
if locmin_flag:
    print("Removing residual peak ahead of main peak...")
    replacement_indices = list(range(end_index, locmin_index))
    replacement_values = np.linspace(end_signal, locmin_signal,
        len(replacement_indices))
    replacement_chrom = pd.DataFrame({"Signal": replacement_values},
        index=replacement_indices)
    chromatogram.update(replacement_chrom)

#Smooth out chromatogram with a LARGER low-pass filter
smooth_me = chromatogram.Signal
total_time = chromatogram.Time.max()
filter_time = 0.01 * total_time  #Time in minutes to filter (1% of total time)
print("Total time, filter window", total_time, filter_time)
filter_size = int(filter_time / dt)
if (filter_size % 2) == 0:  #Check if even
    filter_size = filter_size + 1  #Make it odd
smoothed = savgol_filter(smooth_me, window_length=filter_size,
    polyorder=2, deriv=0)
smoothed_df = pd.DataFrame({"Signal": smoothed})
chromatogram.update(smoothed_df)

return chromatogram

def flatten_baseline(chromatogram, batchinfo, line):
    """
    This function flattens the baseline of the chromatogram
    """
to lessen integration error.
It takes the average of the first few seconds and last few seconds
of the chromatogram, then subtracts the line from the chromatogram signal.
It will alert the user if the difference between the start and end is
greater than a given threshold value relative to the signal maximum.

Assumptions:
> The baseline is linear and is due to signal drift over time\n> The data was collected at a rate of 20 Hz (1 point every 50 ms)

Input variables:
  chromatogram (dataframe) - chromatogram to be modified
  batchinfo (dataframe) - batch information
  line (integer) - line to index from the batchinfo to get the void time

Returned variables:
  chromatogram (dataframe) - modified chromatogram

Written to disk:
  NONE

First written: 2018/05/22
Last updated: 2018/05/22
Depreciated: 2020/02/18 (date flagged)

```python
# Get time information
dt = chromatogram.at[1, "Time"] - chromatogram.at[0, "Time"]
peak_max = chromatogram.Signal.max()
threshold = 0.01
threshold_signal = threshold * peak_max

n_seconds = 5
sampling_rate = 20 # Hertz
index_size = n_seconds * sampling_rate

start_ave = chromatogram.Signal[0:index_size].mean()
end_ave = chromatogram.Signal[-index_size:-1].mean()
del_signal = end_ave - start_ave # Travel from start to end
size = chromatogram.Signal.size
del_index = size - index_size # Travel from start to end, less the indexed length

if del_signal > threshold_signal:
    print("WARNING: This chromatogram baseline drift exceeds threshold.")

slope = del_signal / del_index
```

```python
def make_testreport(directory, batchinfo, reportfname):
    file_list = [f for f in os.listdir(directory) if f.endswith('test.xlsx')]
    lines_list = []
    for fname in file_list:
        # Code continues here
```
fname_line = fname[4:-12]
if fname_line not in lines_list:
    lines_list.append(fname_line)

print(outputfname)
summary_writer = pd.ExcelWriter(outputfname)

scores = pd.DataFrame({'Filename': [],
                      'MeOH': [],
                      'Max': [],
                      'End': []})

for line in lines_list:
    #print("Label for line", line, ",\nLabel = str(input("Label: "))
    line_index = int(line)
    label = batchinfo.at[line_index, "Label"]
    meoh = batchinfo.at[line_index, "MeOH"]
    data_column = batchinfo.at[line_index, "DataColumn"]

    #List the files associated with this line
    file_sublist = []
    for fname in file_list:
        fname_line = fname[4:-12]
        if fname_line == line:
            file_sublist.append(fname)
    print(file_sublist)

    #Add in chromatograms for each line
    gap = pd.DataFrame({'-': []})
    sheet = pd.DataFrame({})

    for index, fname in enumerate(file_sublist):
        inputfname = directory + fname
        print("Processing:", fname)
        df = pd.read_excel(inputfname, header=None, usecols=[0, 1])
        columnnames = ['Time', 'Signal']
        chrom = pd.read_excel(inputfname, header=None, \n                       usecols=[0, 1], names=columnnames)

        max_index = chrom.Signal.idxmax(axis=0)
        time_max = chrom.Time.iloc[max_index]
        end_time = chrom.Time.iloc[-1]

        datum = pd.DataFrame({'Filename':[fname],
                              'MeOH':[meoh],
                              'Max':[time_max],
                              'End':[end_time]})

        scores = scores.append(datum, ignore_index=True)

        sheet = pd.concat([sheet, df, gap], axis=1)
#Add in blank chromatogram for this set
blank_chromatogram = get_blank(directory, batchinfo, int(line))
sheet = pd.concat([sheet, blank_chromatogram], axis=1)
sheet.to_excel(summary_writer, sheet_name=label, header=None, index=False)
scores.to_excel(summary_writer, sheet_name="Summary", index=False)
summary_writer.save()

#Get the directory first
directory = simtools.set_directory(subdirectory="!Conversion Space/")
batchparams = simtools.set_batchparams(directory)

#Get the list of files to process
file_list = [f for f in os.listdir(directory) if f.endswith('-data.xlsx')]

#Load the batch information
batchfname = directory + batchparams
batchinfo = pd.read_excel(batchfname, index_col=0)

#Remove blank files from the file list
file_list_temp = []
for fname in file_list:
    line = int(fname[4:-12])
    if (batchinfo.at[line, "Blank"] != -1):
        file_list_temp.append(fname)

file_list = file_list_temp

#Get the list of files to process
for fname in file_list:
    print("Processing:", fname)
    line = int(fname[4:-12])
    columns = ['Time', 'Signal']
    chromatogram = pd.read_excel(directory + fname, header=None, names=columns)
Main work done by these functions:
blank_chromatogram = get_blank(directory, batchinfo, line)
chromatogram = subtract_blank(chromatogram, blank_chromatogram)
chromatogram = remove_artifact(chromatogram, batchinfo, line)

DEBUG ITEMS: Adds the original chromatogram to the output for comparison
if debug_flag:
    original_chromatogram = pd.read_excel(directory + fname, 
        header=None, names=columns)
    chromatogram = pd.concat([chromatogram,
        original_chromatogram.Signal, 
        blank_chromatogram.Signal], axis=1)
    chromatogram.to_excel(directory + fname[: -10] + "-test.xlsx", 
        header=None, index=False)
else:
    chromatogram.to_excel(directory + fname, header=None, index=False)

if debug_flag:

    experimentname = batchparams[-5]
    outputfname = directory + "Blanking Report" + experimentname + ".xlsx"
    print("Generating:", outputfname)
    make_testreport(directory, batchinfo, outputfname)
    command = 'start EXCEL.EXE ' + '"' + outputfname + '"'
    os.system(command)

print("Finished.")
MYMACRO.mac

Name MYMACRO
print "Hello, World!"
endmacro ! MYMACRO

Name MAKECSV
print "MAKECSV is loaded"
sleep 1

! Loading Variables
Local cols, runtime, signal, i, chroms, path$, Button, filename$, SeqLine, vial, empty$

! Extracting data from register
cols = DataCols(Chromreg)
print cols
chroms = RegSize(Chromreg)
filename$ = "test"

! Making data file
Button = Input("Enter the filename", filename$)
path$ = "C:\"+filename$+.csv"

! Designating data file for dumping
Open path$ For Output as #5
print#5,"time", ",", "signal", ",", "number", ",", "SeqLine", ",", "vial"

! Dumping data into file
For j=1 To chroms
    print "working on chromatogram ", j
    For i=1 To cols
        runtime = Data(Chromreg[j],0,i)
        print Data(Chromreg[j],0,i)
        signal = Data(Chromreg[j],1,i)
        SeqLine = ObjHdrVal(ChromReg[j],"SeqLine")
        Vial = ObjHdrVal(ChromReg[j],"Vial")

        print#5, runtime, ",", signal, ",", i, ",", SeqLine, ",", vial
    Next i
    print#5, ",", ","
Next j

Print path$
sleep 1

Close#5

endmacro ! MAKECSV
Name MAKECSV2

print "MAKECSV2 is loaded"
sleep 1

! Loading Variables
Local cols, runtime, signal, i, chrams, path$, Button, filename$, SeqLine, vial, empty$

! Loading data into register
LoadChromatogram

! Extracting data from register
cols = DataCols(Chromreg)
print cols
chrams = RegSize(Chromreg)
filename$ = "test"

! Making data file
If _SequenceOn = 1 then
    path$ = _datapath$ + _dataseqsubdir$ + "\" + _dataname$ + ".csv"
Else
    Button = Input("Enter the filename", filename$)
    path$ = _datapath$ + _DATASEQSUBDIR$ + "\" + filename$ + ".csv"
EndIf

! Designating data file for dumping
Open path$ For Output as #5
print#5,"time",","signal",","number",","SeqLine",","vial"

! Dumping data into file
For j=1 To chrams
    print "working on chromatogram ", j
    For i=1 To cols
        runtime = Data(Chromreg[j], 0, i)
        print Data(Chromreg[j], 0, i)
        signal = Data(Chromreg[j], 1, i)
        SeqLine = ObjHdrVal(ChromReg[j], "SeqLine")
        Vial = ObjHdrVal(ChromReg[j], "Vial")

        print#5, runtime, ",", signal, ",", j, ",", SeqLine, ",", vial
    Next i
    print#5, "," " 
Next j

print path$
Sleep 1

Close#5
Name LoadChromatogram

! Load the chromatogram by selecting a data file

If _SequenceOn = 1 then  ! Check if sequence running
  File _DataPath$ + _DataFile$  ! Load current data file
  LoadSignal ,"A",, SignalReg
Else
  _Button = Input("Enter data file name:" , FileName$)  ! ask for file
  LoadSignal ,"A",, SignalReg
EndIf
Return

EndMacro ! LoadChromatogram
REFERENCES


14. Berthier, E.; Young, E. W.; Beebe, D., Engineers are from PDMS-land, Biologists are from Polystyrenia. Lab on a Chip 2012, 12 (7), 1224-1237.


115. Belles, A.; Franke, C.; Alary, C.; Aminot, Y.; Readman, J. W., Understanding and predicting the diffusivity of organic compounds in polydimethylsiloxane material for passive


