MODELING AND SIMULATION OF INDUSTRIAL MEMBRANE PROCESSES USING COMPLEX MIXTURES FOR INTEGRATION IN PROCESS SIMULATION ENVIRONMENTS

A Dissertation
Presented to
The Academic Faculty

By

Dylan Jacob Weber

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Engineering
Department of Chemical and Biomolecular Engineering

Georgia Institute of Technology

December 2023

MODELING AND SIMULATION OF INDUSTRIAL MEMBRANE PROCESSES USING COMPLEX MIXTURES FOR INTEGRATION IN PROCESS SIMULATION ENVIRONMENTS

Thesis committee:

Dr. Elizabeth Cherry
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Matthew Realff
Department of Chemical and Biomolecular Engineering
Georgia Institute of Technology

Dr. William Koros
Department of Chemical and Biomolecular Engineering
Georgia Institute of Technology

Dr. Joseph Scott
Department of Chemical and Biomolecular Engineering
Georgia Institute of Technology

Dr. Ryan Lively
Department of Chemical and Biomolecular Engineering
Georgia Institute of Technology

Date approved: December 14, 2023

"Whatever you do in life will be insignificant, but it is very important that you do it; because you can't know. You can't ever really know the meaning of your life. And you don't need to. Every life has a meaning, whether it lasts one hundred years or one hundred seconds. Every life, and every death, changes the world in its own way. You can't know. So don't take it for granted. But don't take it too seriously. Don't postpone what you want. Don't leave anything misunderstood. Make sure the people you care about know. Make sure they know how you really feel. Because just like that...It could end."

-Mohandas Karamchand Gandhi

For my grandpa, Joseph Anthony Weber. May he rest in peace.

November 25th, 1934–April 6th, 2023

ACKNOWLEDGMENTS

As I have gone through this graduate journey towards completing my PhD in Chemical Engineering, there is no way I could have done it alone. We build on the work from the past few centimillennium of human knowledge, and I could not even begin (nor do I have the time or space) to describe how much gratitude I have for the progress of the human race. Firstly, without my families sacrifice, and time to nurture me to being the human I am, I would not be the man I am today. My mother was always there for me growing up, and her hard work to raise three children–for a majority of the time–alone will never go unnoticed. I'll never forget our late night conversations, and hearing of her experiences growing up. I learned to always be a good person no matter the circumstance you come from. Without her pouring her heart into each of her children, we would not have been able to make it on our own. My father taught me the value of hard work, and dedication; at the end of the day, we only have our name. If we mess that up, we'll never amount to anything in this world. The other idea distilled in my head from him is that the only constant in this world is change. He also started me on computers at a very young age, and since his background is a master carpenter, I also learned how to work with my hands as soon as I could hold a hammer. Without my older brother Cody and younger brother Ethan, I also could have not made it this far without them. Our many hours playing Risk, PS2, Gamecube, N64, etc. we were always there for each other and I love you both; mom and dad as well!

Specifically during my graduate career, I would like to thank my research advisor, Dr. Joseph K. Scott. I always appreciate your careful guidance, and conscious attitude towards allowing your graduate students to flourish while maintaining an end goal in mind. Our research group also had the opportunity to move from my initial graduate study at Clemson University to Georgia Tech in 2019. I am forever grateful for the opportunity. Once here, my thesis work really blossomed after initiating an experimental collaboration with Dr. Ryan Lively and his now graduated student, Dr. Ronita Mathias. After seeing their

need in membrane modeling and simulation first-hand, a piece of my thesis was carved out. From there, I was able to jump-start the rest of my contributions to the field. I would also like to thank my initial project team, the RAPID CPM (Rapid Advancement of Process Intensification Deployment Center for Process Modeling). Dr. Chauchyun Chen was the main PI for the project, and he gave very insightful feedback and a space for us to present our work during our weekly meetings. Also, Dr. Maximilian Gorensek and Dr. Palou-Rivera Ignasi were great mentors to learn from how they conducted research and interacted with our team. Other notable colleagues from the RAPID CPM I would like to thank are Dr. Md Islam, Dr. Micheal Sees, Dr. Tori Kirkes, Dr. Ben Caudle, Dr. Usman Hamid. I would also like to thank Dr. Conrad Roos and Youngjoo Lee of Dr. Ryan Lively's group for initiating additional experimental collaborations. I would also like to thank my thesis committee: Dr. Elizabeth Cherry, Dr. Matthew Realff, Dr. Ryan Lively, and Dr. William Koros for their time and effort in being a part of this research journey. Other notable ChBE faculty at GT I would like to thank are Dr. Martha Grover and Janice Whatley for their excellent guidance and support in our transition to GT, and throughout my graduate career. I also would be broken without my feline partner in crime: PACO!

I could not have done this work without the camaraderie and mentoring from my fellow research group members: Dr. Alphonse Hakizimana, Dr. Yuanxun Shao, Dr. Xuejiao Yang, Dr. Kai Shen, and Dr. Taehun Kim. Seeing their research progress, and the insightful conversations I had with them helped me stay motivated, understand that this program is not impossible, and that patience coupled with perseverance in the key. Other members in my research group as well that were very valuable to grow with (and we almost will all get graduate together): Dillard Robertson, Pengfei Cheng, Jason Ye, and Bowen Mu. Outside of my graduate program, there are many friends that all played a part in enjoying the journey with me as well. Coming all the way form my hometown in Greenville, SC–I want to acknowledge Deron Young, Jimmy Hall, Courtney Hawk, Daniel Hall, Officer Micheal Robertson, Aaron Monroe, Peter Gordon, Brian Goodwin, Jennifer Goodwin,

Dalton Studemund, Chris Edwards, Chuck Clifford, Dalton Kelly, and Josh Pfiffer. Those people have been with me since the beginning of high school, and they're not going anywhere from my life and mind! From my initial Clemson graduate 2018 cohort, I would like to acknowledge our first-year struggle-bus members: Dr. Zach Pittman, Dr. Steven Lee, Dr. Steph Klaubert, Dr. Maddie McCarthy, Dr. Graham Tindall, and Dr. Chinmay Joshi. I'll never forget the day we all made at least a 40 on our grad transport final exam, but still celebrated the end of that semester as if we aced it (with the grading curve, you bet we did)! I would also like to thank Dr. Chris Norfolk from Clemson who I took courses with in undergrad (specifically Unit Ops I & II Lab), and then was a teaching assistant for him during my time at Clemson. His insights/experience as a PhD student, as will as tough-love approach to learning chemical engineering allowed me to sharpen my skills, and really grow as a practicing chemical engineer. Regarding my first research experience, I'd like to thank Dr. Sapna Sarupria for giving me a chance to spend the Summer running ice nucleation molecular simulations on Clemson's supercomputer cluster. That was an interesting time to see how undergraduate research works, and how the possibility to continue for grad school was there for me. From my co-op at Fujifilm, I'd like to thank my manager Susan Robert, and co-workers Daneil Moniot, Brook Philips, Ron Holloway, Dylan Pitts, Joshua Hubbard, Crazy Clay, and many others. Other notable friends from Clemson I would like to acknowledge that I still keep up with are: Brice Barnett, Tara Brooks, Christian Sommerville, James Jackson, Ryan Alawar, and Shannon Roberson. The first three plus me, we're four peas in a pod. Once I got to Georgia Tech, I can't forget the memories and people I met during my time here. I am very grateful to meet such great people and quickly become close friends with many of you: Rahul Venkatesh, Dr. Rohan Murty, Dr. Lukas Bingel, Aaron Phfinning, Dr. Zachary Kilwein, Prasaad Milner, Nishant Deshmukh, Kunal Dani, Jose Lopez Ninantay, Matthew Wallace, JT Smith, Jeffrey Hashiem, Tom Wilson, Fidel Amezcua, Dr. Alan Dou, Yacine Feliachi, Dr. Hannah Viola, Dr. Sumner Dudick, Dr. William Bradley, Dr. Danny Shade, Dr. Aaron Liu (one of the first people

I met at GT during my initial days working here), Dr. Gabriel Gusmão, Tanner Hickman, and Kelly Badilla. I'd like to acknowledge our biker gang: Johnathan O'neal, Valrie Kay, Andres Cano, Sabrina Westgate, Matthew Williams (who also sucks at MTG :p), and Rahul again. Additionally, I would like to thank the many people I met during my internship at Via Separations in Boston, MA. Namely, my manager Marcus Lundgren, who allowed me to really take the project in a direction which he never thought would be possible. I met many others namely: Dr. Shreya Dave, Dr. Brent Keller, Dr. Stephen Freyne, Dr. Gabrielle Lawrence, Sintia Sardinas, Robert Reppuci, Krishna Nanduri, Natalie Vogel, Max Barton, Jeff Fuller, Lucy Kitch-Peck, Zachary Troidle, Alena Stern, Vanessa De Campos, Sarah Ellis, Cole Harris, MKL, Ben Voss, Adriana Garties, Sam Rae, Curtis Nicholas, Moris Lee III, Lili Yao, Forbes Fui, Kara L'Italien, Corey Engle, and many others. Everyone's such individualized outlook, and varied perspectives on existence has allowed me to gain such motivation to push forward to keep making this world a better place.

Lastly, I would like to express a paragraph for Alexandra Dobbs. Although our paths have diverged, I still cannot go without acknowledging the time we spent and helped each other grow over our graduate careers. Since meeting you when I first came to Georgia Tech, and we talked about some of our favorite bands, there was a unique chemistry between us. Growing to know each other from our time at many GT cohort outings, the 2019 Halloween party, a local metal band show from a colleague in the department, late nights when our friends wanted to hangout at my apartment near Piedmont Park, to helping each other in coursework; I'll never forget our initial memories trying to beat each other at pool, and the perfect storm that got us together during the most difficult times from the COVID quarantine. I think the one good thing that came out of that lockdown was us. From the many times we pushed each other to excel in coursework, like you to finish your Master's thesis, transition into your newly matched PhD research, successfully proposing your thesis in Spring 2023, and mine for my thesis proposal writing/presentation; we were always there for each other. I'll never forget our trip to Colorado after finishing my pro-

posal, the Durango to Silverton train ride, our many hikes, backpacking/camping trips, paddle-boarding on Lake Nighthorse, experiencing that triple rainbow first-hand after the worst storm of our lives on the Colorado trail, our conversations about the depths of existence, intense Magic the Gathering games with you and friends, our many chess games, many meals we shared and cooked for each other, exploring the city, seeing the rise/fall of The Highlander, seeing the rise/fall of Orpheus Brewery, hanging out at the park with friends, watching days worth of anime, movies, Adventure Time, Avatar the Last Airbender, enjoying countless hours of music/records together, then melding our tastes to one megacollective of awesome tunes, the live shows we've seen like you properly introducing me to metal, me showing you my favorite band Glass Animals, then seeing them live in 2021, seeing Crumb/Limbo/Carnifex/Knocked Loose/Moon Hooch live, crafting together like your many sewing projects and my jewelry hobby, our eclectic fashion choices and the many times we dressed up like it was Halloween when it was a normal Tuesday (on that vein, our Halloween costumes were awesome and I really enjoyed our trip to Salem in 2022), teaching you ASL, our many times at Piedmont Park enjoying each others company, grill-outs, quiches, throwing Frisbee, rock climbing, mountain biking, sushi rolling, meditating, yogi-ing, etc. I never could have completed this PhD journey without your company. I am forever grateful of the memories we made, and growth we underwent together. I wish you the best in your adventures.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AEM anion-exchange membrane

CAFO concentrated animal farming operation

CAPE-OPEN Computer-Aided Process Engineering Open

CASFER Center for Advancing Sustainable Fertilizer Production

CEM cation-exchange membrane

DAEs differential and algebraic equations

DMS dual-mode sorption

ED electrodialysis

FD finite differences

FH Flory-Huggins

FH-LM Flory-Huggins-Langmuir

IEM ion-exchange membranes

IVP initial value problem

MS Maxwell-Stefan

NELF non-equalibrium lattice fluid

NGIG nodal good initial guess

ODEs ordinary differential equations

OSN organic solvent nanofiltration

OSRO organic solvent reverse osmosis

PDAEs partial differential and algebraic equations

PIM-1 polymer membrane of intrinsic microporosity

RHS right-hand side

RMSPE root mean square percentage error

SBAD-1 spirobifluorene aryl diamine

WWTP wastewater treatment plant

NOMENCLATURE

A	area of membrane [m]
A_i	free-fractional volume diffusion theory molecule-polymer system dependant constant [m^2s^{-1}]
a_i^y	activity of component i in Phase y; if second subscript present; $a_{i,h}^y$ refers to evaluation at $z = h$
$\mathbf{a}^y$	vector of n component activities in Phase y, $\mathbf{a}^y = (a_1^y, a_2^y, \ldots, a_n^y)$; $\mathbf{a}_h^y$ refers to evaluation at $z = h$
b	electrodialysis (ED) channel width [m]
B_i	free-fractional volume diffusion theory molecule dependant constant
b_i	Langmuir affinity parameter for component i [bar]
C_i	ED channel i concentration, where $i = C, D$ for concentrate or dilute channel [mol m^{-3}]
c_i	concentration of component i [mol m$^-3$]
C_i^H	volume Langmuir free volume capacity for component i [m^3 penetrant m^{-3} polymer]
$Đ_{ij}^V$	volume based binary Maxwell-Stefan diffusivity for component i and j [m^2s^{-1}]
D^{IEM}	salt permeability coefficient [m^2 s^{-1}]

F Farday's constant; 96,485 [C mol^{-1}]

f_i° pure component fugacity for component i at defined T^y, P^y, [bar]

f_i^y mixture fugacity for component i in Phase y [bar]

$f_{s,\text{SOL}}$ shadow factor for solution channel [1]

$\mathbf{f}^y$ vector of n Phase y fugacities, $\mathbf{f} = (f_1^y, f_2^y, \ldots, f_n^y)$

I total current [A]

i current density [A m-2]

J_i molar flux of salt where $i = \{\text{tot}, \text{diff}, \text{cond}\}$ [mol m^{-2} hr^{-1}]

k_i volume Henry's law parameter for component i [m^3 penetrant Pa^{-1} m^{-3} polymer]

L_p^{IEM} water permeability coefficient [m^{3}Pa^{-1}s^{-1}m^{-2}]

M_w molecular weight of water; 18 [g mol^{-1}]

n number of permeating component, where component $n+1$ refers to the active layer (Phase II) membrane component

N_i molar flux of component i [mol m^{-2}s^{-1}]

N_i^V volumetric flux of component i [L m^{-2}s^{-1}]

N_s number of ED stages

N_{tot} total molar flux [mol m^{-2}s^{-1}]

N_{cp} number of ED cell pairs

$\mathbf{N}^V$ vector of n volumetric fluxes. $\mathbf{N} = (N_1^V, N_2^V, \ldots, N_n^V)$

P^y total pressure of Phase y [bar]

P_i power of stage i or $i =$ tot for total power of single ED stage, where $P = VI$ [W]

Q_i ED channel i volumetric flowrate, where $i = C, D$ for concentrate or dilute channel [m^3 hr^{-1}]

q_i volumetric flux of water where $i = \{\text{w}, \text{osm}, \text{eosm}\}$ [m^3 m^{-2} hr^{-1}]

R gas constant [bar m^3 $mol^{-1}K^{-1}$] or [J $mol^{-1}K^{-1}$]

R_i areal resistance, where $i = \{\text{tot}, \text{CEM}, \text{AEM}, C, D\}$ [Ωm^{-2}]

T temperature of system [K]

t_i^{counter} transport number of counter ion through membrane where $i = \{\text{CEM}, \text{AEM}\}$ [-]

$\overline{V}_i$ partial molar volume of component i [$m^3 mol^{-1}$]

V_i voltage drop where $i = \{\text{tot}, \text{cp}\}$, cp = cell pair, [V]

V_i° molar volume of pure component i at (T, P^{II}) [$m^3 mol^{-1}$]

$v_{F,II}$ free-fractional volume diffusion theory polymer free volume

$\mathbf{V}^\circ$ vector of n pure component molar volumes, $\mathbf{V}^\circ = (V_1^\circ, V_2^\circ, \ldots, V_n^\circ)$

w total water transport number [1]

x ED channel length coordinate [m]

x_i^y mol fraction of component i in Phase y

$\mathbf{x}^y$ vector of n component mol fractions in Phase y, $\mathbf{x}^y = (x_1^y, x_2^y, \ldots, x_n^y)$

z spacial dimension through membrane layer [m]

γ_i^y activity coefficient of component i in Phase y

$\boldsymbol{\gamma}^y$ vector of activity coefficients in Phase y. $\boldsymbol{\gamma}^y = (\gamma_1^y, \gamma_2^y, \ldots, \gamma_n^y; \boldsymbol{\gamma}_h^y)$ refers to evaluation at $z = h$

δ^{i} thickness of either $i = \{\text{IEM}, \text{SOL}\}$ [m]

$\delta_{y,i}$ Hansen solubility parameters for dispersion forces (y = D), intermolecular forces (y = P), and hydrogen bonding (y = H) [$\text{Pa}^{0.5}$]

η non-ohmic voltage contributions [V]

μ_i° chemical potential of component i at pure component system temperature T and pressure P° reference state [J mol^{-1}]

μ_i^y chemical potential of component i in Phase y [J mol^{-1}]; $\mu_{i,h}^y$ refers to evaluation at $z = h$

Φ_i^{IEM} osmotic coefficient for $i = C, D$ for concentrate or dilute channel

ϕ_i volume fraction of component i dissolved in the Phase II; $\phi_{i,h}$ refers to evaluation at $z = h$

ρ_w density of water [g m^{-3}]

$\boldsymbol{\phi}$ vector of Phase II volume fractions.
$\boldsymbol{\phi} = (\phi_1, \phi_2 \ldots, \phi_n, \phi_m)$; for all methods except discretization, $\boldsymbol{\phi}_h$ refers to evaluation at $z = h$, $\boldsymbol{\phi}_{1:n}$ is ϕ_i for $i = 1, 2, .., n$, for the discretization methods, $\boldsymbol{\phi}_k$ is $\boldsymbol{\phi}$ evaluated at node k

SUMMARY

This dissertation addresses the need for improved industrial membrane process modeling and simulation by developing a general framework that considers complex mixture coupling. As our society shifts from everyday products stemming from distillation-based fossil-fuel refineries, to those same products made by means of bio-refineries; utilizing novel separation technologies such as membranes and complex mixtures will be ever more prevalent. A complex mixture refers to either a liquid or gas stream with no single majority component and usually gives rise to mixture interactions (thermodynamic or diffusional coupling) between species. Membranes are used for complex mixture separations in many applications, including water purification, carbon capture, hydrogen separation, olefin/paraffin separation, benzene derivative concentration, and membrane reactor systems. Currently, overall membrane process modeling is heavily reliant on simple models that do not consider complex mixture interactions. In addition, numerical algorithms for simulating membrane performance using a rigorous modeling framework are inefficient and unreliable for systems with many permeants or strong thermodynamic/diffusional coupling. Moreover, membrane thermodynamic and diffusional modeling capabilities are still lacking for transport predictions based on parameters fit from minimal experimental data. Consequently, a general-purpose membrane simulation method with sufficient accuracy, robustness, and efficiency to be included in process flowsheet simulation environments is non-existent. Therefore, there is a critical need for improved numerical algorithms and modeling capabilities for industrial membrane processes involving complex mixtures.

To address this need, the overall objective of this work is to develop new theory and algorithms for modeling and simulating industrial membrane modules. Such theory and algorithms must be applicable to any complex mixture, membrane material, and module geometry for integration into overall process flowsheet simulation. This dissertation will start by working with the most logical aspect that is microscopic (local) membrane transport. For the detailed contribution, chapter 2 presents improved numerical methods to solve

complex mixture local membrane transport using a Maxwell-Stefan model. For membrane modeling, the most significant challenges have to do with accurate thermodynamic and diffusional predictions that require no multicomponent mixture parameterization (i.e. models with parameters that are fit based solely on pure component data to minimize the number of experiments required). To find the contributions that work towards this goal, chapter 3 presents novel sorption and diffusion models. Additionally, a more compact version of the Flory-Huggins sorption model is presented to enable simulation of complex mixtures with hundreds of components. Then, chapter 4 presents a software package of our contributions for pressure-based industrial membrane processes for use by practicing chemical engineers within process simulation environments. Finally, chapter 5 provides a base case nutrient recovery scenario, simulation framework, preliminary control strategies, preliminary process designs, and a working unit operation code of electrified industrial membranes for nutrient recovery from wastewater treatment plant and concentrated animal farming operation streams. Overall, this work has resulted in novel algorithms, modeling capabilities, and a software package for industrial membrane process simulation. In addition to that, contributions towards electrified industrial membrane processes are presented. Moving forward, the membrane design process involving complex mixtures can compete with the seamless design process of traditional energy or chemically intensive separations such as distillation, extraction, air stripping, or crystallization. Overall, this will enable deployment of complex mixture membrane processes (i.e. more energy efficient and smaller chemical processes). This will benefit society by reducing our environmental footprint, and allow for design of intensified chemical systems.

CHAPTER 1
INTRODUCTION

1.1 Overview

This dissertation provides improved numerical methods, phenomenological relationships, and software tools to enable pressure-based industrial membrane process modeling and simulation that considers complex mixture coupling. In addition to that, this dissertation provides process design tools for electrified industrial membrane nutrient recovery processes to produce nitrogen and phosphorous fertilizers from wastewater treatment plant and concentrated animal farming operation streams. The contributions from this dissertation aim to make it possible for process design of industrial membrane processes utilizing complex mixtures. The end-goal of these modeling and numerical method approaches is to extend them and enable mixture simulations of up to hundreds (or thousands) of components. While the numerical methods herein have been applied to organic solvent reverse osmosis and fertilizer nutrient recovery, these methods are generally applicable to membrane processes that utilizes streams with complex mixture coupling and/or electrochemical potential gradients. Given that, the models used may need alterations to match the given application. The next sections provide background on complex mixture chemical separations, pressure-based industrial membrane processes, existing phenomenological relationships used to model pressure-based industrial membrane transport, existing numerical methods to simulate pressure-based industrial membrane transport, available tools to design pressure-based industrial membrane processes, electrified industrial membrane based-processes, and available software to design electrified industrial membrane processes. Within the last sections, the shortcomings of each piece will be highlighted to define the knowledge gaps which this dissertation specifically addresses. Finally, the chapter ends with an outline of

contributions to each topic, and where it can be found.

1.2 Complex Chemical Mixture Separations

As chemical engineers, we confront separations in everything we work with. Even as a consumer, every product has been through some sort of separation process somewhere along its synthesis. In fact, from the 32% of energy that is used within the US for the industrial sector, 45–55% of that energy is devoted to chemical separations [2]. A good rule of thumb for chemical engineering is that half of the energy is used to synthesize the chemical of interest, while the other half is used to separate it. This project will consider complex mixtures as the fundamental process stream when modeling and simulating industrial membrane processes. This is significant because realistically every chemical process contains at least one complex mixture stream. A complex mixture refers to either a liquid or gas chemical process stream with no single majority component. This usually gives rise to mixture interactions (either through thermodynamic and/or diffusional coupling) between species. In the context of industrial membrane processes, many assume solution ideality such that single component properties carry over to higher component mixtures. However, this assumption misses phenomena only observed in complex mixtures [2]. As an example, membranes for olefin/paraffin separation have been shown to have increased separation performance with a realistic process stream that has five added impurities relative to what would be expected based on pure component olefin and paraffin permeation experiments [3].

When considering the transition from everyday chemicals derived from fossil-fuel refineries to ones derived from environmentally sustainable bio-refineries, complex mixtures are guaranteed to be encountered. Koutinas et al. presents a reaction network consisting of 24 chemical building blocks that can be produced via fermentation [4]. These building blocks range from fuels, polymers, pharmaceuticals, niche/bulk commodity chemicals, etc. Throughout all proposed biological networks, most of them have side-products creating a

complex mixture that will need to be refined. Other sources of complex mixture separations are the valorization of industrial waste and byproduct streams. In the wood, pulp, and paper industries for example, Koutinas et al. states that wood residues account for about 10% of the starting material going to waste, and black liquors are produced at a rate of 7 tons per ton of paper pulp. These streams contain cellulose, hemicellulose, lignin, phenolic compounds, dissolved salts, etc. These complex mixtures can then be either separated as recyclable process streams, biochemicals, biopolymers, bioethanol, or biodiesel. Looking towards solvent use in the chemical and pharmaceutical industries, raw materials can be up to 85% solvent by mass [5]. Producing these solvents by means of sustainable biological precursors will also lead to a complex reaction broth that will need to be refined [6].

As far as everyday chemical separations that utilize complex mixtures, crude oil has to be the largest example of a mixture containing over 10,000 compounds that each end up in almost every product we use [7]. Two final notable examples are wastewater treatment plant (WWTP) and concentrated animal farming operation (CAFO) streams. These streams contain many different organic acids, large organics, and salts that end up creating a complex system of various particle interactions. For specific component breakdowns, Table 1.1 and Table 1.2 show the common components for a synthetic inlet WWTP and CAFO lagoon manure streams, respectively. In addition to these examples, to reach realistic process stream simulations, exemplar complex mixture streams must first be understood and modeled [8].

1.3 Pressure-based Industrial Membrane Processes

1.3.1 Overview

Starting from the beginning of life itself, the development of a physical barrier between an internal system and its environment is thought to have been the crucial prerequisite to chemical evolution for life on Earth [11]. That chemical autonomy theoretically then evolved to human life with the foundational physical barrier being a lipid bi-layer membrane. Then, in 1748, the discovery of osmosis to describe the permeation of water through a pig bladder by

Table 1.1: CAFO lagoon manure composition [9].

Component	CAFOs Manure (mg/L)
NH_4^+-N	407
Organic-N	168
Total Nitrogen	575
P_2O_5	336
K_2O	731
Ca	103
Mg	55
Zn	4
Cu	2
Mn	1
S	37
Na	216

Table 1.2: WWTP inlet stream synthetic recipe [10].

Ingredient	Synthetic WWTPs Inlet solution (mg/L)
Urea	97
NH_4Cl	12.7
Na-acetate	79.4
Peptone	17.4
$MgPO_4{\cdot}H_2O$	29.0
KH_2PO_4	23.4
$FeSO_4{\cdot}7H_2O$	5.8
Starch	122.0
Milk powder	116.19
Yeast	52.2
Soy Oil	29.0
$Cr(NO_3)_3{\cdot}9H_2O$	0.8
$CuCl_2{\cdot}2H_2O$	0.5
$MnSO_4{\cdot}H_2O$	0.1
$NiSO_4{\cdot}6H_2O$	0.3
$PbCl_2$	0.1
$ZnCl_2$	0.2

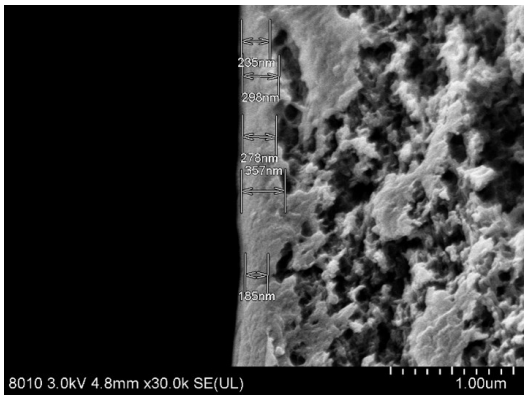


Figure 1.1: FE-SEM image of asymmetric membrane [14].

Abbé Nolet sparked inspiration from nature to continue to study these semipermiable materials. Almost 200 years later in the 1960s, atomically precise physical barriers for large scale chemical separations were first commercialized [12]. Today, membranes are used for water purification, carbon capture, hydrogen separation, olefin/-paraffin separation, benzene derivative concentration, and many other applications [2]. Moreover, membranes are a low-energy alternative when comparing to traditional thermal separation processes such as distillation and evaporation. In fact, membrane processes have been shown to be the most applicable energy-saving alternative for chemical separations across a set of industries that currently account for 98% of the total U.S. separations energy consumption [13]. In these same industries, it has been estimated that a transition to low-energy separations including membrane processes could lead a 16% reduction in the total annual energy consumed by chemical separations [13]. In absolute amounts, this could save 223 trillion BTU/yr, or the equivalent energy of 25.5 Empire State Buildings worth of coal by weight per year. With that energy savings in mind, the next paragraph outlines some general background on industrial membrane structure, material, and module geometry.

Although this dissertation will focus mainly on the local scale membrane transport, it is relevant to understand membrane modules as a whole. This is important because future work may focus on incorporating capabilities to simulate the most industrially relevant membrane processes using numerical methods and models presented in this dissertation.

This will be significant because it provides a complete toolkit for the end user to model and simulate complex membrane processes given any relevant material, module geometry, or flow configuration. While membrane technology was first limited to lab scale performance for many years, increased availability and application of industrial membrane technology in the early 1960s was due to the Loeb-Sourirajan process for making "defect-free, high-flux, anisotropic reverse osmosis membranes" [15]. Figure 1.1 shows an example of asymmetric membrane that utilizes a thin active layer fabricated on a porous support layer. The cited paragraph continues, stating that this technique allowed reverse osmosis membranes to have a 10-fold increase in productivity for water purification, which ultimately led to its commercialization. Asymmetric membranes also allow for a rigid backbone that can withstand harsher environments, while the nanometer-thin active layer can provide the proper separation selectivity and permeation rate.

For those reasons, this dissertation will be restricted to asymmetric membrane process modeling and simulation. Relevant industrial membrane materials can either be inorganic (zeolites, metallic organic frameworks–MOFs, thin metallic layers) or organic (polymeric, carbon molecular sieves, carbon nanotubes) [16]. Combining the two, mixed-matrix membranes provide the high component selectivity of dense polymer membranes and the high permeation rates of inorganic materials [17]. The material chosen affects the variables used to model the system and will be important in later sections. There are transport regimes based on nominal pore size[12]. Simple relationships such as Darcy's law and Knudsen diffusion describe the transport well for pore sizes above 1 nanometer. This project will focus on molecular-scale (subnanometer) nominal pore sizes. This leads to complex interactions, and challenges arise when modeling and simulating such phenomena.

At the global scale, module geometries and flow configurations can take many forms. Geometries include flat plate, tubular, plate-and-frame, hollow fiber, spiral wound, and submerged modules [18]. The main trade-offs between each type of configuration has to do with ease of manufacturing and packing density (area of active membrane layer per unit

volume of membrane module). For hollow fiber membranes, the values can reach 30,000 m^2/m^3. Conversely, plate-and-frame modules can be one the order of 100 m^2/m^3. Different flow geometries and configurations can be considered as well. Cross-flow is more commonly seen in industrial applications where streams flow parallel to the membrane surface. These streams can also flow in co-current, counter current, or perpendicular directions (which is also referenced in certain texts as "cross-flow", so care should be taken to understand the context since "cross-flow" is also a common term for lab-scale/industrial-sized permeators with tangential flow along the membrane surface rather than vertically perpendicular to the membrane surface as found in "dead-end" permeators). Figure 1.2 shows the general system set-up for a co-current cross-flow flat plate membrane module. When thinking about the different scales to model and simulate such processes, the overall membrane module is defined as the global transport scale. Following that, the third part of Figure 1.2 show the local transport scale. In later chapters, the difference between the two scales will entail vastly different simulation strategies. While chapter 2 and chapter 3 will go into rigorous detail of the modeling and simulation of membrane processes using complex mixtures, subsection 1.3.2 provides a high-level overview of the membrane local flux modeling and simulation problem.

1.3.2 Modeling and Simulation

This section aims to present a high-level overview of pressure-based industrial membrane process modeling and simulation. In doing so, the cohesiveness of each objective to the overall goal of this dissertation is established. Moreover, challenges and critical needs will be highlighted to motivate the completed dissertation work. The following subsections will then provide detail to justify claims. While this dissertation does not deal with contributions to global membrane module transport, it is still important to understand the state of the art, and how an interested reader might extend the numerical methods and models presented in later chapters from local scale transport to the global scale transport. That relevant background can be found in Appendix C for the interested reader.

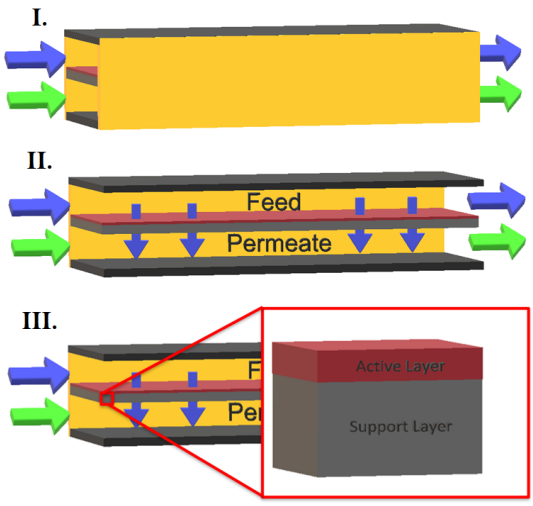


Figure 1.2: Three views of the same flat plate membrane module with co-current flow configurations

To begin, any industrial membrane module is composed of a feed stream that comes into contact with a membrane layer and partially transports through to a permeate stream. From that basis, there are three main aspects to model: the feed channel model, the permeate channel model, and a model that describes the flux through the surface of the membrane (also known as the **local** transport model). Each piece combined creates a coupled system of partial differential and algebraic equations (PDAEs). The **global** transport model is the overall transport model that combines these 3 parts. Note that the physical properties model will be assumed known at all times.

The feed and permeate channel models are comprised of differential species concentration, momentum, and energy balances. The local transport model is dependant on the feed-side species concentrations, permeate-side species concentrations (in certain cases this is not needed, as seen in chapter 2), temperatures, and pressures at each point on the membrane surface. Using those values on each side, a driving force or gradient for transport is created. This gradient dictates the flux that travels across the membrane, which is the main unknown.

The mechanism for local transport can be described by the sorption-diffusion (SD)

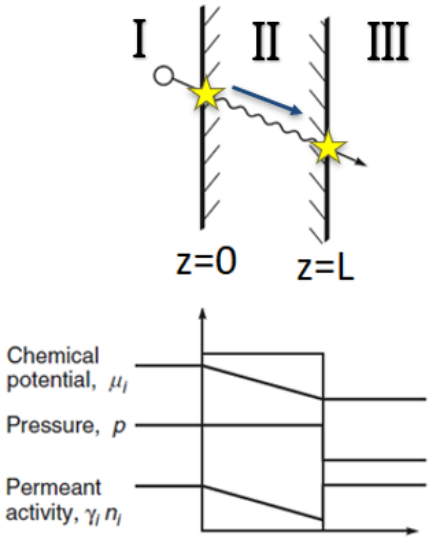


Figure 1.3: Solution-diffusion mechanism, figure adapted from [19].

model. This mechanism can be described by three steps (see Figure 1.3):

1. thermodynamic equilibrium (sorption) between bulk feed (Phase I) and the active layer membrane (Phase II)

2. diffusion through the active layer membrane (Phase II)

3. thermodynamic equilibrium between active layer membrane (Phase II) and support layer membrane (Phase III)

The main driving force for transport is the gradient of chemical potential, which is a function of all species concentrations, pressure, and temperature. The pressure in Phase II has been shown theoretically to be constant (or of negligible importance) and equal to the Phase I pressure [20].

This three-step mechanism implies there are two main sub-models to the local membrane transport problem–a thermodynamic sorption model, and a membrane diffusion model.

The main problems addressed in the dissertation are: (i) simulation of the local transport model, (ii) insight and novel modeling approaches for thermodynamic and diffusional modeling, and (iii) tools to design such processes within chemical process flowsheeting environments. The critical needs for (i) are fast and reliable numerical methods when dealing

with complex mixture separation. For (ii), the critical need is improved models for predicting complex mixture membrane sorption/diffusion with minimal computational complexity and experimental data.

The main flow for the next subsections will start with the most basic piece and build on complexity.

Membrane Sorption Modeling

The local transport framework relies on accurate thermodynamic models to make realistic permeation predictions. This project will focus on thermodynamic models applied to polymeric and inorganic materials. Charged interactions between species will be ignored for now.

The most basic ideal thermodynamic sorption model is Henry's Law. This provides a linear relationship for phase equilibrium partitioning. This model is valid for low species activities (related to concentration) in a penetrant-polymer mixture. When considering higher activities and complex mixtures, this model will fail. This is because of nonlinearity in the experimental pure-component isotherms, and interactions with other mixture species (see Minellie et al. for this trend in the experimental data) [21].

For inorganic nanoporous materials, the sorption phenomena is known as adsorption. Well-known example thermodynamic models take the form of either Langmuir or Ideal Adsorbed Solution Theory (IAST) [22]. The Langmuir model deals with pure component adsorption, and is based on a surface reaction balance between adsorption/desorption. The multicomponent extension of this theory is the second term of (Equation 1.2) [23]. This model works for gas systems with non-interacting species. The latter multicomponent adsorption model, IAST, has been the standard for gas adsorption modeling over the last 50 years [24]. Yet, the assumptions built-in breakdown for complex mixtures. There have been extensions of IAST to predict real mixtures, namely real adsorbed solution theory (RAST) [25]. However, the equations are difficult to parameterize without a multitude of multicomponent mixture data. This is because an underlying activity model is needed

where binary interaction parameters are needed.

In modeling rubbery polymer membranes, the thermodynamics between penetrant-penetrant and penetrant-polymer pairs is well described by the Flory-Huggins polymer solution theory [26] [27]. The challenge lies with glassy polymers, which contain non-equilibrium frozen voids within the polymer matrix. These voids disappear as the polymer swells (plasticize) with increased penetrant activity. The polymer matrix is then rubber-like, and described by a Flory-Huggins type sorption. Current models to describe glassy polymer sorption are the non-equalibrium lattice fluid (NELF), and the dual-mode sorption (DMS) model. The NELF model requires many fitting parameters even for the single component case, and fails to capture the higher activity transition to a rubbery polymer membrane state (see Minelli et al. for this trend in the experimental data) [21]. The NELF equation for a simple permeant (1) and polymer (2) system can be written as [21]

$$\ln S_0 = \ln\left(\frac{T_{STP}}{p_{STP}T}\right) + r_1^0\left\{\left[1+\left(\frac{v_1^*}{v_2^*}-1\right)\right]\ln\left(1-\frac{\rho_{20}}{\rho_2^*}\right)\right\} + \left(\frac{v_1^*}{v_2^*}-1\right) + \frac{\rho_{20}}{\rho_2^*}\frac{T_1^*}{T}\frac{2(1-k_12)}{p_1^*}\sqrt{p_1^* p_2^*}. \quad (1.1)$$

For DMS, the sorption is split into two distinct types, one being this glassy void filling and the other a simple Henry's law dissolution type [28]. Combing the two, the working equation becomes

$$c_i^m = c_{d,i}^m + c_{h,i}^m = k_{d,i} f_i^m + \frac{C_{h,i}^{m,sat} b_i f_i^m}{1+\sum_{j=1}^n b_j \hat{f}_j^m}. \quad (1.2)$$

The predicted isotherm is similar to that of the NELF prediction in Minellie et al. [21].

The main problems outlined for thermodynamic membrane sorption modeling are either (i) the models are simple but do not apply to complex mixtures, or (ii) the models work well for complex mixtures but are difficult to parameterize and implement even in the single permeant case.

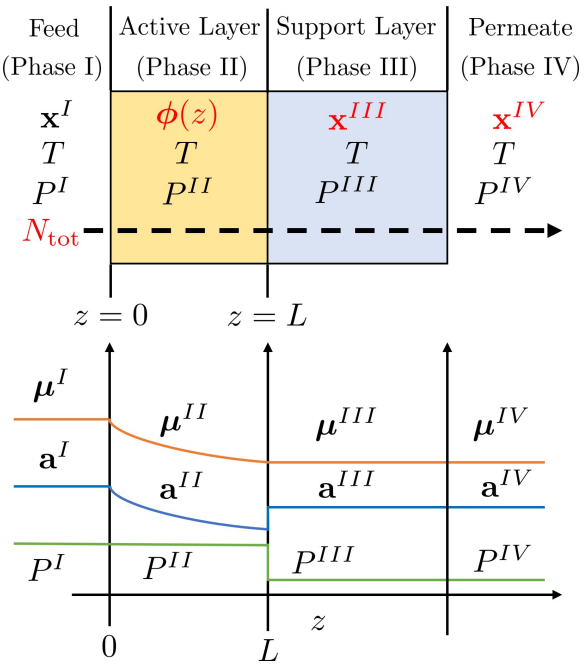


Figure 1.4: Quantitative representation of an asymmetric membrane layer at the local level. We assume that the support layer provides no resistance to transport.

Local Transport Modeling and Simulation

Given a suitable thermodynamic sorption model, the local transport model calculates the component fluxes. All physical properties are assumed to be known. Figure 1.4 shows a quantitative representation of the local transport system.

The simplest uncoupled approach assumes the permeant driving forces are not dependant on other mixture species. The workhorse permeability, P_i, or Permeance, $\tilde{P}_i$, model for membrane transport can be written as

$$N_i = -\frac{D_{i,(m),\text{Fick}} S_i}{l_{mem}} \left(c_i^{III} - c_i^I\right) = -\frac{\hat{P}_i}{l_{mem}} \left(c_i^{III} - c_i^I\right) = -\tilde{P}_i \left(c_i^{III} - c_i^I\right). \tag{1.3}$$

This equation is simple to implement, but has these assumptions: (i) constant permeant activity throughout the membrane layer, (ii) concentration independence from the other components in the mixture, and (iii) Henry's type thermodynamic partitioning. All of these assumptions fail to capture complex mixture species coupling between each ij pair [29] [30].

As a better approach, the Maxwell-Stefan (MS) transport model is built for complex mixture coupling [31]. There are other formulations such as the generalized Fick's law and Onsager formulations. For generalized Fick's law, the derived cross-diffusion coefficients are specific for each multicomponent mixture [32]. For the Onsager formulation, parameters are solvent and reference frame specific for each multicomponent mixture. [33]. The basis for fixed-membrane phase ($N_m = 0$) Maxwell-Stefan expressions are a chemical vs. friction force balance between each molecule in the mixture. This model can be written for an n component mixture in matrix form as system of ordinary differential equations (ODEs)

$$\begin{bmatrix} \frac{d\phi_1}{dz} \\ \frac{d\phi_2}{dz} \\ \vdots \\ \frac{d\phi_n}{dz} \end{bmatrix} = -\mathbf{\Gamma}^{-1}\mathbf{B}\mathbf{n}, \tag{1.4}$$

where,

$$\Gamma_{ij} = \phi_i \frac{\partial \ln(a_i^{II})}{\partial \phi_j} = \frac{\phi_i}{f_i^{II}} \frac{\partial f_i^{II}}{\partial \phi_j}, \quad B_{ii} = \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{\phi_j}{Đ_{ij}^V} \right) + \frac{\phi_m}{Đ_{im}^V}, \quad B_{ij} = -\frac{\phi_j}{Đ_{ij}^V}.$$

This approach is useful, and easily interpreted as thermodynamic coupling between all ij pairs though Γ_{ij}, along with the diffusional coupling though each $Đ_{ij}^V$.

This model is built to handle complex mixture transport. Yet, there are two notable problems. The first is modeling each $Đ_{ij}$ using predictive models with minimal or no mixture experimental data. The second is implementing and simulating the system of equations.

For diffusional models within polymer membranes, the concentration dependence of the penetrant-polymer diffusivity is dependant on the swelling state of the polymer. The

phenomena presented is usually of an exponential form like [34]

$$\text{Đ}_{1m}^{V} = \text{Đ}_{1m,0}^{V} \exp[A_1(\phi_1 + C_{12}\phi_2)]. \tag{1.5}$$

This expression is not easily extended to higher component mixtures without arbitrary extra fitting terms. For rigid polymer matrix diffusion models, they usually have to do with the degree of loading within the nanopourous material, and requires further investigation. Known models are almost always empirical in nature and more work is needed to be done to establish diffusion theories.

Regarding simulation of (Equation 1.4), it cannot be directly integrated like the simple permeability model. The main point is this rigorous approach creates a 2-point boundary value problem since there is are two fixed boundary conditions. The first boundary condition is volume fraction of all components evaluated at $z = 0$. This can be found using Phase I molar compositions that are assumed in equilibrium with the Phase II at $z = 0$. To model and simulate local asymmetric membrane transport in a predictive manner (based on feed composition), a relationship coined by C.Y. Pan is used [35]. This relates the ratio of partial fluxes to the molar composition in the support layer Phase III as

$$x_i^{III} = \frac{N_i}{\sum_{j=1}^{n} N_j}, \; i = 1, 2, ..., n. \tag{1.6}$$

Using this relationship, and assuming equilibrium between Phase II and III, the volume fraction at $z = L$ can be found. This equation pair has been shown in the literature before [36] [37] [38]. However, the methods used to solve them are insufficient for a high number of permeants, strong thermodynamic coupling, or strong diffusional coupling. The method by Hesse et al. uses discretization and partitions Phase II into discrete nodes while keeping equation (Equation 1.4) in terms of chemical potential gradients. This avoids having to evaluate derivatives to generate the thermodynamic factor matrix, $\mathbf{\Gamma}$. Even so, the ODE system begins to take on non-linearity through the $\frac{1}{RT}\frac{d\mu_i^{II}}{dz} = \frac{d\ln a_i^{II}}{dz}$ term. See Figure 1.5 for

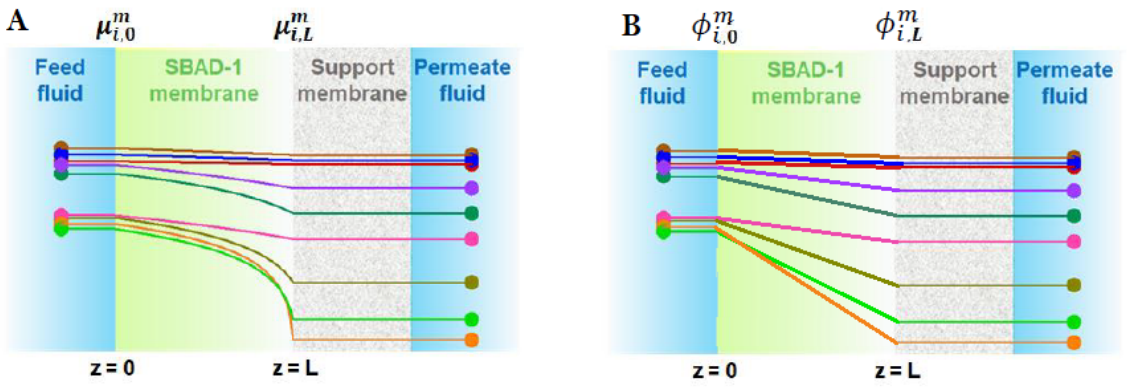


Figure 1.5: (A) Simulated chemical potential gradient, (B) equivalent volume fraction gradient.

comparison of state variable profiles. The introduction of highly nonlinear profiles begins to cause the number of discretization nodes to increase greatly, and make the computational times unreasonable.

Izák et al. presents a form of these ODE equations by assuming a linear profile throughout the membrane phase (only two nodes). The matrices in equation (Equation 1.4) are evaluated at an average concentration of component i in Phase II. This method is also presented by Krishna et al., but does not invoke Pan's relationship [39]. These methods have been shown to work well for a small number of components. However, after looking into 5 or 9 components, we found that the matrices can actually vary significantly across the membrane layer and can cause error from the true solution [40].

In summary, the Maxwell-Stefan transport model is the workhorse for local complex mixture membrane transport. Given that, there are still pitfalls with this approach to modeling and simulation that must be addressed. First, novel diffusional relationships are needed based on minimal computational complexity and parameterization requirements. Along with that, the state-of-the-art methods for simulating this local transport framework present challenges for general use. One challenge is requiring a large inefficient system of equations when discretizing highly nonlinear chemical potential gradients. The other is encountering large error for systems exhibiting strong thermodynamic or diffusional coupling that

varies significantly with concentration throughout the membrane layer.

1.3.3 Available Software

Now that the exact problem is motivated to model and simulate asymmetric (polymeric, inorganic, or mixed-matrix materials) industrial membrane processes (flat plate, plate-and-frame, spiral wound, hollow fiber, and submerged module geometries with co-current, counter-current, or perpendicular flow configurations) using complex mixtures for use in process simulation environments. The most notable known tools (either commercially available or standalone software) are the Chemstations, Inc. membrane process unit model, commercially available example files within AspenPlus process flowsheeting software, a commercially available unit operation in gPROMs, a software package developed by University of Minnesota to simulate spiral-wound membranes called memPy, and MEMSIC (which simulates gas separation in various flow configurations of a flat plate membrane module) [41] [42]. From a recent review article, there are another 17 different membrane modeling and simulation tools released [43]. The article also states that the software packages lack features to interface with proprietary simulators and error handling for model convergence failure. Another observation is that half are restricted to gas separations, including the Chemstations, AspenPlus, and gPROMS unit operations. organic solvent nanofiltration (OSN) Designer is another general purpose tool that actually takes a step in the right direction and implements their *MATLAB* code into Aspen using Computer-Aided Process Engineering Open (CAPE-OPEN) standards [44]. However, the limitations are that the software is limited to binary mixtures, and only applicable for organic solvent nanofiltration. These tools are a good first milestone being commercially available, but they are lacking in being a general purpose tool for complex mixture separations since most are limited to uncoupled gas separations.

1.4 Electrified Industrial Membrane Processes

1.4.1 Overview

Compared to traditional industrial membrane processes, where transport is based on a mechanical pressure and/or concentration gradient that drives transport from the feed (which is usually concentrated in the chemical of interest) to the permeate (usually dilute in the chemical of interest), electrified industrial membrane processes work in the opposite manner by using an electrical potential gradient to drive ions from a dilute stream to a more concentrated stream. The most common application of this is desalination of seawater to produce clean drinking water [45]. Additionally, this electrified industrial membrane process is usually referred to as ED. The basic principles of how it works are as follows. The feed stream is considered a dilute stream with one or more charged ions (salts). The product or waste stream (depending on the application) is a concentrated charged ion stream. Separating these streams are ion-exchange membranes (IEM). The simplest subsystem can be thought of as a cell-pair where from left-to-right there is a concentrate channel (product or waste), cation-exchange membrane (CEM), dilute (feed) channel, and an anion-exchange membrane (AEM). This cell pair repeats as many times as needed to give a full electrodialysis stack. On the ends, there are electrode channels (anode and cathode) to provide an electrical current that passes across the ED stack to drive transport of ions from each respective dilute channel to each of the concentrate channels. For the purposes of this dissertation, ED will be applied to nutrient recovery of ammonium and phosphate salts from WWTP and CAFO streams. This is not a novel application on the lab-scale. However, it is when applied on an industrial scale. There are many intricacies involved in process design which this dissertation aims to address. The next section looks available software to design these processes on an industrial scale.

1.4.2 Available Software

As with the available software to design such processes for pressure-based industrial membrane unit operations utilizing complex mixtures, the available software to design electrified industrial membrane processes is scarce; even for a single salt solution. The publication by Capione et al. describes their implementation is done in gPROMS (a commercial process simulator), but the code is unavailable for open-source use. QSDsan (*https://qsdsan.com/*) is a software package available for open-source use in Python that has many unit operations available that are common in wastewater treatment plants. While the software package does have an electrodialysis unit operation, the model is more rudimentary than the one presented in the previous section such that the model is based on fixed recovery rates. Given that, it is mainly only useful for high level system energy and flow calculations. The only other main open-source software that actually implements the same modeling framework as outlined previously is found in a software package called WaterTAP (*https://www.nawihub.org/knowledge/watertap/*). The embedded unit operation has many great options for ED simulation, and also implements state-of-the-art electrolyte solution activity models. Compared to anything else, this software would seems like the option to do preliminary process design and optimization. While it is for someone really proficient in Python, after installing it and getting it up and running, the code itself is not straightforward to use for a practicing engineer or an experimental academic researcher. Moreover, there are no dynamic capabilities if wanting to model lab-scale experiments to fit parameters and/or look at various control strategies due to process disturbances. That is why there is a need for a more user-friendly tool to design nutrient recovery processes.

1.5 Contributions

Currently, pressure-based membrane process modeling is heavily reliant on simple models that do not consider complex mixture interactions. In addition, numerical algorithms for simulating membrane performance using a rigorous modeling framework are inefficient,

and unreliable for systems with many permeants or strong thermodynamic or diffusional coupling. Moreover, membrane thermodynamic and diffusional modeling capabilities are still lacking for transport prediction when parameters are fit from minimal experimental data. Consequently, a general-purpose membrane simulation method with sufficient accuracy, robustness, and efficiency to be included in process flowsheet simulation environments is non-existent for pressure-based or electrified industrial membrane processes. Therefore, chapter 2 will present improved numerical algorithms and modeling capabilities for industrial membrane processes involving complex mixtures. We start by working with the most logical aspect which is local membrane transport. For membrane modeling, chapter 3 addresses the most significant challenges having to do with accurate thermodynamic/diffusional predictions that require no multicomponent mixture parameterization, and enabling high component number complex mixture simulations. Then, in chapter 4, we will package our contributions applied to pressure-based industrial membrane processe for use by practicing chemical engineers. Finally, chapter 5 will provide preliminary process designs that show various electrodialysis-based membrane cascades to produce a viable fertilizer product. From this work, the membrane design process involving complex mixtures can compete with the seamless design process of traditional energy or chemically intensive separations such as distillation or extraction. Enabling rapid deployment of complex mixture membrane processes will lead to more energy efficient and smaller chemical processes. This work will enable practicing engineers and researchers to help society by reducing our environmental footprint, and allow for design of modular chemical systems that may be used for modular purposes (e.g. on-demand farmer operated fertilizer production, consumer produced bio-fuels, and/or space colonization applications).

CHAPTER 2
IMPROVED NUMERICAL METHODS FOR LOCAL MEMBRANE TRANSPORT

2.1 Introduction

This chapter presents improved numerical methods for predicting complex mixture separation across asymmetric polymer membranes. The term *complex mixture* refers to a multicomponent fluid mixture with no clear majority component, which usually gives rise to interactions between species that strongly affect membrane performance. Complex mixtures are ubiquitous in chemical processes and are most often separated using energy-intensive thermal methods such as distillation. In fact, 45–55% of U.S. in-plant energy is used for separations [13]. Fortunately, membranes offer an alternative that can potentially achieve significant process intensification. Sholl and Lively [2] estimated that membrane processes could significantly reduce energy intensity compared to thermal separation routes in crude oil refining, bio-oil refining, olefin/paraffin separation, benzene derivative concentration, o/p-xylene separation, and more. Unfortunately, the complex hydrocarbon mixtures in these applications are often theromdynamically non-ideal and their separations via membrane processes are difficult to predict. Standard membrane modeling practices involving constant permabilities or uncoupled Fick's law approaches are often unsuitable [29, 30, 46]. This leads to significant challenges in modeling these systems, parameterizing the models from minimal experiments, and numerically solving the resulting models. At present, all three of these issues are major impediments to widespread adoption of membrane processes in the chemical industry [47]. Our recent publication (adapted text can be found in chapter 3) presented a promising theoretical framework for modeling complex solvent mixture separation with glassy polymer membranes, including procedures for parametrizing the model using only pure component experiments [14]. Here, we focus on the numerical so-

lution procedures needed to efficiently and reliably solve this class of models. The methods developed here may also be advantageous for other applications where strong inter-species coupling occurs.

This chapter specifically considers numerical methods for the *local flux problem*. The objective of this problem is to predict the partial flux of each component through a membrane layer in contact with a uniform feed material of known composition, temperature, and pressure, and a uniform permeate material of known temperature and pressure. This is in contrast to simulating a complete (i.e., global) membrane module, where material flows across the membrane surface in some specified configuration such that the feed and permeate conditions are different at each point on the membrane surface. The local flux problem is suitable for predicting the outcome of laboratory permeation experiments with $\ll 1\%$ stage cut, which is valuable for material screening, model testing, and parameter estimation. It is also a critical subtask required for solving global module models since the latter can be viewed as a continuum of local flux problems coupled by feed and permeate conservation laws.

The mathematical form of the local flux problem we aim to solve is based on the assumption that transport is described by a sorption-diffusion mechanism, which is well established for polymer membranes [12, 19]. Accordingly, the feed and permeate phases are taken to be in equilibrium with the adjacent membrane phases. Transport through the active layer of the membrane is assumed to occur by Maxwell-Stefan diffusion. This accounts for both thermodynamic coupling (through the use of chemical potential driving forces) and diffusional coupling (through the use of cross diffusivities). Finally, we assume Pan's relationship for asymmetric membranes [35], which states that the composition in the membrane support layer is specified completely by the partial fluxes through the membrane, and is therefore an additional unknown. The model comprising these basic parts is now well-established [14, 37, 38, 48, 49, 50, 51, 52, 53, 54]. Overall, this results in a challenging nonlinear two-point boundary value problem in differential-algebraic equa-

tions (differential and algebraic equations (DAEs)).

A variety of numerical methods have been proposed for solving the local flux problem, both directly and with the aid of simplifying assumptions. If thermodynamic and diffusional coupling are ignored, then the Maxwell-Stefan model reduces to Fick's law (the standard uncoupled diffusion model). If the sorption model is also assumed to be linear, then the standard constant permeability model is obtained, resulting in a much simpler numerical problem. Notably, however, even this model cannot be solved explicitly when Pan's relationship is used. Moreover, these approximations often lead to large prediction errors for complex mixtures [14, 55, 56, 57, 58]. In [39], Krishna proposed a more accurate model by first including thermodynamic and diffusional coupling matrices derived from a rigorous Maxwell-Stefan model and then evaluating these matrices at the membrane midpoint composition and assuming them to be constant. This model admits a simple explicit solution procedure in the case of pervaporation where the permeate composition is known, but not otherwise. Izák et al. [37] proposed a similar model combined with Pan's relationship to address the general case with unknown permeate composition. The model is solved using a numerical method previously published by Heintz and Stephan [48]. However, this method is only applicable to binary systems and ignores thermodynamic coupling. Hesse et al. [36] proposed a method for solving the complete local flux problem without simplification using finite difference approximation of the governing equations. However, few details are given about the method used for solving the resulting system of equations. Additionally, this approach is highly sensitive to the initial guess provided, and no guidance is given for that choice. Mittal et al. [38] presents a similar Maxwell-Stefan model for butane isomer separation through zeolite membranes and solves the system using gPROMs. However, the exact numerical method used is not specified and no analysis of the methods performance is given. In addition to these specialized methods, a variety of general-purpose methods are available for solving two-point boundary value problems, and their relative merits are well known to domain experts [59, 60, 61, 62]. However, these techniques are not prevalent in

the membrane literature and their suitability for membrane simulations has not been well studied.

This chapter describes improved numerical methods to solve the local flux problem for asymmetric membranes without the simplifying assumptions mentioned above. First, we develop a full discretization method similar to Hesse et al., but with the equations cast differently and a different choice of iteration variables. Next, we propose a shooting algorithm based on the same formulation. We also provide reliable initialization strategies. Finally, we provide a detailed comparison with existing methods using three organic solvent separation case studies with validated models from [14]. We find that the methods that rely on simplifications of the Maxwell-Stefan model give poor results for the realistic hydrocarbon mixtures considered. Moreover, among numerical methods that address the full Maxwell-Stefan model, the proposed methods offer significant advantages in terms of computational efficiency, accuracy, and robustness.

2.2 Problem Statement and Modeling Background

2.2.1 General Problem Statement

The local flux problem is illustrated schematically in Figure 2.1. The membrane active layer is in contact with feed material at a constant composition, temperature, and pressure, all of which are known. The entire system is assumed to be isothermal. While temperature effects can be nontrivial in some cases [63], consideration of these effects is out of the scope of this work. The membrane support layer is in contact with a bulk permeate with known pressure. The objective is to find the permeate compositions and total molar flux through the membrane at steady-state.

A sorption-diffusion mechanism is assumed. Therefore, the feed, active layer, support layer, and permeate are distinct thermodynamic phases (Phases I–IV). The feed and active layer are in equilibrium at $z = 0$, and the active and support layers are in equilibrium at $z = L$. Concentration polarization is assumed to be negligible. Transport through

the active layer occurs by diffusion. We assume the bulk permeate does not mix into the support layer. This implies that the composition in the support layer ($\mathbf{x}^{III}$) is constant and fully determined by the component fluxes through the active layer. In turn, the flux through the active layer depends on the driving force established by $\mathbf{x}^{III}$, so these variables must be determined simultaneously. Since the system is at steady-state, the component fluxes entering the permeate are the same as those entering the support layer. The permeate is simply viewed as a constant reservoir with $\mathbf{x}^{IV} = \mathbf{x}^{III}$. Therefore, only $\mathbf{x}^{III}$ is actually solved for in the proposed methods. In a model of a complete membrane module, where the local flux problem describes just a single point on the membrane surface, $\mathbf{x}^{IV}$ would be different from $\mathbf{x}^{III}$ and would be determined by permeate balances considering the influx of material from the support layer as well as bulk permeate inflows and outflows. However, $\mathbf{x}^{III}$ would be unchanged, as would the driving forces experienced by the active layer. This is the essence of Pan's relationship for asymmetric membranes [35]. Therefore, the local flux problem addressed here involves exactly the same equations that would need to be solved at each point on the membrane in a more complex full-module model.

Pan's relationship is valid for ideal cross-flow contactors and is a good approximation for other types of contactors provided that there is minimal back-mixing of material from the bulk permeate stream into the support layer. However, it may be a poor approximation otherwise. In fact, it may be desirable in some cases to design contactors with a high degree of back-mixing so that the bulk permeate can be used as a sweep stream to enhance the driving force across the membrane. In such cases, a more suitable version of the local flux problem is to omit Pan's relationship and instead assume that $\mathbf{x}^{III} = \mathbf{x}^{IV}$ is given (i.e., perfect back-mixing). We do not consider this variant of the problem in the main text. However, the implications of this change for the numerical methods studied here are discussed in section A.6.

The following subsections present the model equations for each piece of the local flux model.

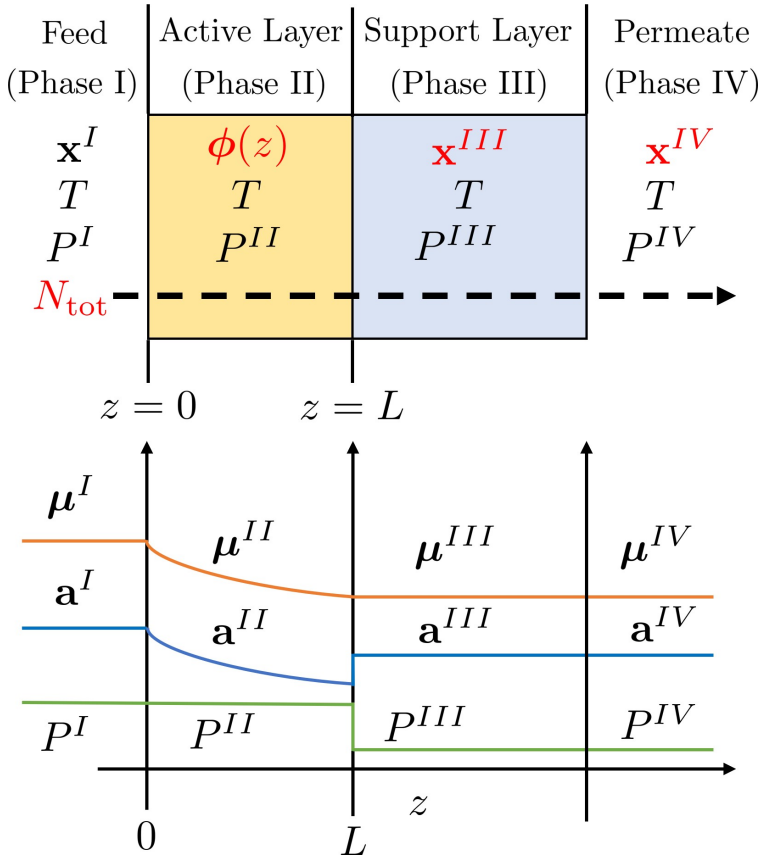


Figure 2.1: Top: Set-up of the local flux problem for an asymmetric membrane with knowns in black font and unknowns in red font (see nomenclature table). Bottom: Schematic of the solution-diffusion model. The chemical potential driving force includes contributions from both activity and pressure. Resistance to transport through the support layer is assumed to be negligible.

2.2.2 Phase Equilibrium

Equilibrium between Phases I and II at $z = 0$ and Phases II and III at $z = L$ implies that, for all $i = 1, \ldots, n$,

$$\mu_{i,0}^{I} = \mu_{i,0}^{II}, \tag{2.1}$$

$$\mu_{i,L}^{III} = \mu_{i,L}^{II}. \tag{2.2}$$

Defining the activity of component i in Phase y, a_i^y, relative to a pure i reference state at the same T and P of Phase y, and letting $^\circ$ denote a pure component property, we have

$$\ln(a_i^y) = \ln\left(\frac{f_i^y}{f_i^\circ(T^y, P^y)}\right) = \frac{\mu_i^y - \mu_i^\circ(T^y, P^y)}{RT^y}. \tag{2.3}$$

To conform with standard property models for each phase, activity coefficients will be used to describe Phases I and III, while fugacity will be used to describe the membrane Phase II. With this convention, combining, (Equation 2.1)–(Equation 2.3) gives

$$\mu_i^\circ(T, P^I) + RT\ln\left(\gamma_i^I x_i^I\right) = \mu_i^\circ(T, P^{II}) + RT\ln\left(\frac{f_{i,0}^{II}}{f_i^\circ(T, P^{II})}\right), \tag{2.4}$$

$$\mu_i^\circ(T, P^{III}) + RT\ln\left(\gamma_i^{III} x_i^{III}\right) = \mu_i^\circ(T, P^{II}) + RT\ln\left(\frac{f_i^{II}}{f_i^\circ(T, P^{II})}\right). \tag{2.5}$$

Using $P^I = P^{II}$ and the approximation $\mu_i^\circ(T, P^{III}) - \mu_i^\circ(T, P^{II}) \approx V_i^\circ(P^{III} - P^{II})$ (the full derivation of this expression can be found in section A.1), and V_i° is the pure component molar volume at (T, P^{II}), (Equation 2.4) and (Equation 2.5) simplify to

$$f_{i,0}^{II} = f_i^{\circ}(T, P^{II})\gamma_i^I x_i^I, \tag{2.6}$$

$$f_{i,L}^{II} = f_i^{\circ}(T, P^{II})\gamma_i^{III} x_i^{III} \exp\left[\frac{-V_i^{\circ}(P^{II} - P^{III})}{RT}\right]. \tag{2.7}$$

Thermodynamic models describing the activity coefficients and fugacities in (Equation 2.6)–(Equation 2.7) as functions of T, P, and composition are assumed to be available in the form of general implicit relationships:

$$\mathbf{g}^{\circ}(\mathbf{f}^{\circ}, T, P) = \mathbf{0}, \tag{2.8}$$

$$\mathbf{g}^{I}(\boldsymbol{\gamma}^{I}, \mathbf{x}^{I}, T, P^{I}) = \mathbf{0}, \tag{2.9}$$

$$\mathbf{g}^{II}(\mathbf{f}^{II}, \boldsymbol{\phi}, T, P^{II}) = \mathbf{0}, \tag{2.10}$$

$$\mathbf{g}^{III}(\boldsymbol{\gamma}^{III}, \mathbf{x}^{III}, T, P^{III}) = \mathbf{0}, \tag{2.11}$$

where $\mathbf{0}$ is an n-dimensional vector of zeroes and the vectors $\boldsymbol{\gamma}^I$, $\boldsymbol{\gamma}^{III}$, $\mathbf{f}^{II}$, $\mathbf{f}^{\circ}$, $\mathbf{f}^{\circ}$, $\mathbf{g}^I$, $\mathbf{g}^{II}$, and $\mathbf{g}^{III}$ all have n components. In subsubsection 3.2.2, several options are presented for the membrane fugacity model $\mathbf{g}^{II}$.

Equations (Equation 2.6), (Equation 2.9), and (Equation 2.10) provide a complete description of the phase equilibrium between Phases I and II, while equations (Equation 2.7), (Equation 2.10), and (Equation 2.11) describe the equilibrium between Phases II and III. Note that equation (Equation 2.10) is used twice, once to relate $\mathbf{f}_0^{II}$ and $\boldsymbol{\phi}_0$ at $z = 0$, and again to relate $\mathbf{f}_L^{II}$ and $\boldsymbol{\phi}_L$ at $z = L$.

2.2.3 Active Layer Transport by Maxwell-Stefan Diffusion

The active layer (Phase II) is viewed as a mixture of $n + 1$ components, with the $(n + 1)^{\text{st}}$ being the membrane itself. The volume fractions ϕ_i and chemical potentials μ_i^{II} for each

component vary with position z, and are related to the volumetric transmembrane fluxes N_i^V by the volume-fraction form of the Maxwell-Stefan model from [50]:

$$-\frac{\phi_i}{RT}\frac{d\mu_i^{II}}{dz} = \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{\phi_j N_i^V - \phi_i N_j^V}{\mathfrak{D}_{ij}^V} + \frac{\phi_{n+1} N_i^V}{\mathfrak{D}_{i,n+1}^V}. \tag{2.12}$$

Our model includes a copy of equation (Equation 2.12) for every $i = 1, \ldots, n$. The $(n+1)^{\text{st}}$ equation is also valid, but is redundant with the first n by the Gibbs-Duhem relation (see section A.2). To clarify notation below, we will replace the subscript $n+1$ with m for 'membrane' wherever it appears.

The chemical potential μ_i^{II} is determined by the independent set of variables $(T, P, \boldsymbol{\phi}_{1:n})$. The last component of $\boldsymbol{\phi}$ is excluded because it depends on the first n via

$$\sum_{j=1}^{n+1} \phi_j = 1. \tag{2.13}$$

Since T and P are constant in the active layer, the left-hand side of (Equation 2.12) can be expanded by the chain rule as

$$-\frac{\phi_i}{RT}\frac{d\mu_i^{II}}{dz} = -\frac{\phi_i}{RT}\sum_{j=1}^{n}\frac{\partial \mu_i^{II}}{\partial \phi_j}\frac{d\phi_j}{dz}. \tag{2.14}$$

Then, using $d\mu_i^{II} = RT d\ln f_i^{II}$,

$$-\frac{\phi_i}{RT}\frac{d\mu_i^{II}}{dz} = -\sum_{j=1}^{n}\frac{\phi_i}{f_i^{II}}\frac{\partial f_i^{II}}{\partial \phi_j}\frac{d\phi_j}{dz}. \tag{2.15}$$

Substituting this into (Equation 2.12) and writing the resulting system of n equations compactly in matrix form gives

$$\boldsymbol{\Gamma}(\boldsymbol{\phi}, \mathbf{f}^{II})\frac{d\boldsymbol{\phi}_{1:n}}{dz} = -\mathbf{B}(\boldsymbol{\phi})\mathbf{N}^V, \tag{2.16}$$

where $\mathbf{N}^V$ is the vector of volumetric fluxes N_i^V and $\mathbf{\Gamma}(\boldsymbol{\phi})$ and $\mathbf{B}(\boldsymbol{\phi})$ are n-by-n matrices with elements $\Gamma_{ij} = \frac{\phi_i}{f_i^{II}} \frac{\partial f_i^{II}}{\partial \phi_j}$ and

$$B_{ij}(\boldsymbol{\phi}) = -\frac{\phi_i}{\text{Đ}_{ij}^V}, \qquad (i \neq j) \tag{2.17}$$

$$B_{ii}(\boldsymbol{\phi}) = \sum_{j=1,\ j\neq i}^{n} \frac{\phi_j}{\text{Đ}_{ij}^V} + \frac{\phi_m}{\text{Đ}_{im}^V}. \tag{2.18}$$

Derivations of the thermodynamic factor matrix $\mathbf{\Gamma}$ for various membrane fugacity models can be found in section A.5. Physically, this matrix depends on T, P^{II}, and $\boldsymbol{\phi}$. Above, we suppress the dependence on T and P^{II} for brevity and add $\mathbf{f}^{II}$ as an additional argument. When the active layer fugacity model (Equation 2.10) is explicit (i.e., simple enough to solve analytically for $\mathbf{f}^{II}$ in terms of $\boldsymbol{\phi}$), it is possible to derive an expression for $\mathbf{\Gamma}$ as a function of $\boldsymbol{\phi}$ only. However, when (Equation 2.10) is implicit, the corresponding equations for $\mathbf{\Gamma}$ involve both $\boldsymbol{\phi}$ and $\mathbf{f}^{II}$. In this case, evaluating $\mathbf{\Gamma}$ at a given $\boldsymbol{\phi}$ involves first using $\boldsymbol{\phi}$ to calculate $\mathbf{f}^{II}$ by solving (Equation 2.10), and then evaluating the $\mathbf{\Gamma}$ equations with the pair $(\boldsymbol{\phi}, \mathbf{f}^{II})$. See section A.5 for further details. The notation $\mathbf{\Gamma}(\boldsymbol{\phi}, \mathbf{f}^{II})$ clarifies the discussion of numerical solution procedures because it correctly reflects the fact that values for both $\boldsymbol{\phi}$ and $\mathbf{f}^{II}$ are needed to compute $\mathbf{\Gamma}$ in general. Regarding $\mathbf{B}(\boldsymbol{\phi})$, we assume the MS diffusivities are known for all i, m pairs and the Vignes relationship is used to evaluate each Đ_{ij} (assuming $\overline{V}_i = V_i^\circ(T, P^{II})$) [64]:

$$\text{Đ}_{ij}^V = V_j^\circ (\text{Đ}_{im}^V / V_j^\circ)^{\phi_i/(\phi_i+\phi_j)} (\text{Đ}_{jm}^V / V_j^\circ)^{\phi_j/(\phi_i+\phi_j)}. \tag{2.19}$$

In [14], the Vignes relationship was shown to result in a reasonable fit between simulations and data for the test mixtures we use for our numerical comparisons in section 2.5. However, it may not work well for certain systems and, in such cases, other correlations such as the Darken relationship or correlations based on liquid diffusivities can be used [65]. None

of the numerical methods we study require the use of any particular correlation here.

2.2.4 Pan's Relationship and Degrees of Freedom Analysis

The local flux model is completed by Pan's relationship, which asserts that the composition in the support layer is completely specified by the molar fluxes entering it [35]:

$$N_i = x_i^{III} \sum_{j=1}^{n} N_j = x_i^{III} N_{\text{tot}}. \tag{2.20}$$

In terms of the volumetric flux N_i^V, this is

$$N_i^V = V_i^\circ x_i^{III} N_{\text{tot}}, \tag{2.21}$$

or, in vector form,

$$\mathbf{N}^V = \text{diag}(\mathbf{V}^\circ)\mathbf{x}^{III} N_{\text{tot}}. \tag{2.22}$$

Substituting this into (Equation 2.16) gives:

$$\mathbf{\Gamma}(\boldsymbol{\phi}, \mathbf{f}^{II}) \frac{d\boldsymbol{\phi}_{1:n}}{dz} = -\mathbf{B}(\boldsymbol{\phi})\text{diag}(\mathbf{V}^\circ)\mathbf{x}^{III} N_{\text{tot}}. \tag{2.23}$$

The complete system of equations we aim to solve is summarized in Figure 2.2. Although different solution strategies will solve these equations in different orders, it is helpful to walk through them sequentially to understand the degrees-of-freedom. The first block considers phase equilibrium at $z = 0$. Since $\mathbf{x}^I$ is known, the unknowns are $\mathbf{f}^\circ$, $\mathbf{f}_0^{II}$, $\boldsymbol{\gamma}^I$, and $\boldsymbol{\phi}_0$ ($4n + 1$ unknowns). Since there are $4n + 1$ equations in this block, all of these unknowns are specified. Thus, this block contributes zero degrees-of-freedom and can be solved independently of the rest of the model. The values of $\boldsymbol{\phi}_0$ and $\mathbf{f}_0^{II}$ provide the initial

condition for the second block, which describes the transport across the active layer using differential algebraic equations (DAEs) derived from (Equation 2.13) and (Equation 2.23). The unknowns here are the trajectories $\boldsymbol{\phi}(z)$ and $\mathbf{f}^{II}(z)$ for all $z \in [0, L]$. If the active layer fugacity model (Equation 2.10) is explicit, then $\boldsymbol{\Gamma}$ can be expressed as a function of $\boldsymbol{\phi}$ only. In that case, the variables $\mathbf{f}^{II}(z)$ and equations $\mathbf{g}^{II}$ are not needed in this block. However, when (Equation 2.10) is implicit, it is necessary to write $\boldsymbol{\Gamma}$ as a function of both $\boldsymbol{\phi}$ and $\mathbf{f}^{II}$ and invoke $\mathbf{g}^{II}$ to relate $\boldsymbol{\phi}$ and $\mathbf{f}^{II}$ at every $z \in [0, L]$, resulting in DAEs. In either case, if N_{tot} and $\mathbf{x}^{III}$ were known, then this system could be solved from 0 to L to obtain $\boldsymbol{\phi}_L$ and $\mathbf{f}_L$. Therefore, this block contributes $n + 1$ degrees-of-freedom. The third block models the phase equilibrium at $z = L$. With $\boldsymbol{\phi}_L$ and $\mathbf{f}_L^{II}$ specified, the only new unknown in this block is $\boldsymbol{\gamma}^{III}$ ($\mathbf{x}^{III}$ and $\mathbf{f}^{\circ}$ were counted above). Since, there are $2n + 1$ equations, this block consumes $n + 1$ degrees-of-freedom. Thus, the overall system is well-posed.

In general, solving systems of DAEs can be much more challenging than solving ODEs due to several issues that are unique to DAEs models; interested readers are referred to [66] for details. Fortunately, the DAEs system in the second block of Figure 2.2 is of a relatively simple type called semi-explicit index 1 DAEs, as shown in section A.7. Consequently, we do not need to consider manual index reduction or specialized methods for high-index systems.

2.3 Existing Solution Methods

This section reviews existing methods for solving the system of equations in Figure 2.2. These methods are compared with the proposed new methods in section 2.4. The methods in subsection 2.3.1 and subsection 2.3.2 invoke various common assumptions to substantially simplify the rigorous Maxwell-Stefan equations in the second block of Figure 2.2 before solving the system. The purpose of comparing against these methods is to assess the impact of these additional assumptions and better understand when it is necessary to solve the rigorous model. In contrast, the method in subsection 2.3.3 solves the rigorous model,

Phase Equilibrium at $z = 0$

$$\mathbf{g}^{\circ}(\mathbf{f}^{\circ}, T, P^{II}) = \mathbf{0}$$

$$\mathbf{g}^{I}(\boldsymbol{\gamma}^{I}, \mathbf{x}^{I}, T, P^{I}) = \mathbf{0}$$

$$\mathbf{g}^{II}(\mathbf{f}_0^{II}, \boldsymbol{\phi}_0, T, P^{II}) = \mathbf{0}$$

$$f_{i,0}^{II} = \gamma_{i,0}^{I} x_i^{I} f_i^{\circ}, \quad i = 1, \ldots, n$$

$$\sum_{j=1}^{n+1} \phi_{j,0} = 1$$

Active Layer Diffusion

$$\frac{d\boldsymbol{\phi}_{1:n}}{dz} = -\boldsymbol{\Gamma}^{-1}(\boldsymbol{\phi}, \mathbf{f}^{II})\mathbf{B}(\boldsymbol{\phi})\text{diag}(\mathbf{V}^{\circ})\mathbf{x}^{III} N_{\text{tot}}$$

$$\frac{d\phi_m}{dz} = -\sum_{j=1}^{n} \frac{d\phi_j}{dz}$$

$$\mathbf{g}^{II}(\mathbf{f}^{II}, \boldsymbol{\phi}, T, P^{II}) = \mathbf{0}$$

Phase Equilibrium at $z = L$

$$\mathbf{g}^{III}(\boldsymbol{\gamma}^{III}, \mathbf{x}^{III}, T, P^{III}) = \mathbf{0}$$

$$f_{i,L}^{II} = \gamma_{i,L}^{III} x_i^{III} f_i^{\circ} \exp\left[\frac{-V_i^{\circ}(P^{II} - P^{III})}{RT}\right], \quad i = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_j^{III} = 1$$

Known variables: $(\mathbf{x}^{I}, T, P^{I}, P^{II}, P^{III})$

Unknown variables: $(\mathbf{f}^{\circ}, \boldsymbol{\gamma}^{I}, \boldsymbol{\gamma}^{III}, \mathbf{f}_0^{II}, \mathbf{f}_L^{II}, \boldsymbol{\phi}_0, \boldsymbol{\phi}_L, \mathbf{x}^{III}, N_{\text{tot}})$

DoF: $5n + 3(n + 1)$ equations
$-5n + 3(n + 1)$ unknowns
= 0 DoF

Figure 2.2: Full set of equations and degrees of freedom analysis for the local transport problem.

but uses a different numerical procedure than the new methods presented in section 2.4. The purpose of comparing against this method is to better understand the pros and cons of different numerical methods for solving the rigorous model in terms of efficiency, accuracy, and robustness.

2.3.1 Fick's Law Approximation

The Fick's Law approximation simplifies the Maxwell-Stefan equations in the second block of Figure 2.2 by assuming that both thermodynamic and diffusional cross-coupling are negligible. This approximation can be derived from (Equation 2.16) by setting $\Gamma_{ij}(\boldsymbol{\phi}, \mathbf{f}^{II}) = 0$ and $B_{ij}(\boldsymbol{\phi}) = 0$ (i.e., $Đ_{ij}^{V} \to \infty$) for all $i \neq j$, and further assuming that the diagonal terms $\Gamma_{ii}(\boldsymbol{\phi}, \mathbf{f}^{II})$ and $B_{ii}(\boldsymbol{\phi})$ are constant. Defining $D_{im,\text{Fick}}^{V} = \Gamma_{ii}/B_{ii}$, (Equation 2.16) simplifies to n independent equations of the form

$$N_i^V = -D_{im,\text{Fick}}^{V} \frac{d\phi_i}{dz}. \tag{2.24}$$

Unlike (Equation 2.16), this version is simple enough to be integrated analytically from $z = 0$ to $z = L$, giving

$$N_i^V = -\frac{D_{im,\text{Fick}}^{V}}{L} \left(\phi_{i,L} - \phi_{i,0}\right). \tag{2.25}$$

This is the classical Fick's law model for membrane transport rewritten in terms of volume fractions. If one assumes the Henry's law sorption relation $\phi_i = S_i x_i$, which we do not do here, (Equation 2.25) simplifies further to the standard constant permeability model with $P_i = S_i D_{im,\text{Fick}}^{V}$ [19]. Fick's law and constant permeability models have been widely used for gas separations, dialysis, reverse osmosis, and more [12].

To construct a complete model analogous to Figure 2.2 based on the Fick's Law approximation, we first plug (Equation 2.21) into (Equation 2.25) to obtain

$$V_i^{\circ} x_i^{III} N_{\text{tot}} = -\frac{D_{im,\text{Fick}}^{V}}{L} \left(\phi_{i,L} - \phi_{i,0}\right). \tag{2.26}$$

These n equations replace the first equation in the second block of Figure 2.2. Since these are algebraic equations that only involve $\boldsymbol{\phi}(z)$ at $z = 0$ and $z = L$, rather than differential equations describing $\boldsymbol{\phi}(z)$ for all $z \in [0, L]$, the second and third equations in that block are no longer needed for all z, but only at $z = L$. Thus, they are replaced with

$$\sum_{i=1}^{n+1} \phi_{i,L} = 1, \tag{2.27}$$

$$\mathbf{g}^{II}(\mathbf{f}_L^{II}, \boldsymbol{\phi}_L, T, P^{II}) = \mathbf{0}. \tag{2.28}$$

The first and third blocks are unchanged.

To solve the complete model, the first block in Figure 2.2 is first solved independently to obtain $\mathbf{f}^{\circ}$, $\boldsymbol{\gamma}^{I}$, $\mathbf{f}_0^{II}$, and $\boldsymbol{\phi}_0$. Next, the modified second and third blocks are solved simultaneously to obtain $\boldsymbol{\gamma}^{III}$, $\mathbf{f}_L^{II}$, $\boldsymbol{\phi}_L, \mathbf{x}^{III}$, and N_{tot}. In our implementation, we actually use (Equation 2.7) to eliminate $\mathbf{f}_L^{II}$ from the remaining equations analytically, resulting in only $3n+2$ equations solved simultaneously for $\boldsymbol{\gamma}^{III}$, $\boldsymbol{\phi}_L$, $\mathbf{x}^{III}$, and N_{tot}. Note that, although the Fick's law approximation decouples and greatly simplifies the MS model (Equation 2.16), the complete model is still coupled and must be solved simultaneously. This is caused by the use of Pan's relationship and is in contrast to many common applications of Fick's law where $\mathbf{x}^{III}$ is assumed to be known (e.g., in pervaporation) [43, 67]. Further simplifications that can be made when $\mathbf{g}^{II}$ is explicit in fugacity are discussed in section A.4.

For the case studies in section 2.5, the required Fickian diffusivities $D_{im,\text{Fick}}^{V}$ were obtained by fitting the model to pure component permeation experiments as described in [14]. Alternatively, they could be calculated by evaluating models for Γ_{ii} and B_{ii} at some reference condition $\boldsymbol{\phi}^*$ and $\mathbf{f}^{II,*}$ as $D_{im,\text{Fick}}^{V} = Đ_{im}^{V}\Gamma_{ii}(\boldsymbol{\phi}^*, \mathbf{f}^{II,*})/\phi_m^*$.

2.3.2 Average Coupling Approximations

This section discusses two methods that simplify the Maxwell-Stefan equations in the second block of Figure 2.2 by assuming that the coupling matrices $\mathbf{B}(\boldsymbol{\phi})$ and/or $\boldsymbol{\Gamma}(\boldsymbol{\phi}, \mathbf{f}^{II})$ can be evaluated at average membrane conditions. In the first method, these matrices are evaluated with the average quantities $\boldsymbol{\phi}_{\text{avg}} = \frac{\boldsymbol{\phi}_0 + \boldsymbol{\phi}_L}{2}$ and $\mathbf{f}^{II}_{\text{avg}} = \frac{\mathbf{f}^{II}_0 + \mathbf{f}^{II}_L}{2}$, which makes them constant but not necessarily diagonal as in the Fick's law approximation. This greatly simplifies the model while still including diffusional and thermodynamic coupling. Making this approximation in (Equation 2.23) and integrating gives

$$\text{diag}(\mathbf{V}^{\circ})\mathbf{x}^{III} N_{\text{tot}} = -\mathbf{B}^{-1}(\boldsymbol{\phi}_{\text{avg}})\boldsymbol{\Gamma}(\boldsymbol{\phi}_{\text{avg}}, \mathbf{f}^{II}_{\text{avg}}) \frac{\boldsymbol{\phi}_{1:n,L} - \boldsymbol{\phi}_{1:n,0}}{L}. \tag{2.29}$$

Similar to the Fick's law approximation, the complete model for this approximation is obtained by replacing the equations in the second block of Figure 2.2 with (Equation 2.27)–(Equation 2.29). The first and third blocks are unchanged. The numerical solution procedure is exactly analogous to that described in subsection 2.3.1.

The second method is similar but begins from a more compact form of the Maxwell-Stefan equations that does not involve $\boldsymbol{\Gamma}$, thereby avoiding the assumption that $\boldsymbol{\Gamma}$ is constant. Specifically, substituting $d\mu_i^{II} = RTd\ln f_i^{II}$ into the LHS of (Equation 2.12) yields

$$\text{diag}(\boldsymbol{\phi}_{1:n}) \frac{d\ln\left(\mathbf{f}^{II}\right)}{dz} = -\mathbf{B}(\boldsymbol{\phi})\mathbf{N}^{V}. \tag{2.30}$$

where the natural log is taken component-wise. Combining with (Equation 2.22) then yields

$$\text{diag}(\boldsymbol{\phi}_{1:n}) \frac{d\ln\left(\mathbf{f}^{II}\right)}{dz} = -\mathbf{B}(\boldsymbol{\phi})\text{diag}(\mathbf{V}^{\circ})\mathbf{x}^{III} N_{\text{tot}}. \tag{2.31}$$

Finally, applying the average concentration approximation to the matrices $\text{diag}(\boldsymbol{\phi}_{1:n})$ and

$\mathbf{B}(\boldsymbol{\phi})$ and integrating yields

$$\text{diag}(\boldsymbol{\phi}_{1:n,\text{avg}}) \left(\frac{\left(\ln\left(\mathbf{f}_L^{II}\right) - \ln(\mathbf{f}_0^{II})\right)}{L} \right) = -\mathbf{B}(\boldsymbol{\phi}_{\text{avg}})\text{diag}(\mathbf{V}^{\circ})\mathbf{x}^{III} N_{\text{tot}}. \tag{2.32}$$

The complete model for this approximation is obtained by replacing the equations in the second block of Figure 2.2 with (Equation 2.27), (Equation 2.28), and (Equation 2.32). The first and third blocks are unchanged. The numerical solution procedure is exactly analogous to that described in subsection 2.3.1. We refer to these two methods as the $\boldsymbol{\phi}$-form and $\mathbf{f}$-form average coupling approximations, respectively, since the first uses the volume-fraction form of the MS equations in (Equation 2.16), while the second uses the fugacity form in (Equation 2.30).

The $\boldsymbol{\phi}$-form approximation has been proposed in multiple papers [37, 39, 48]. Krishna et al. proposed it for simulating pervaporation membranes in [39]. However, since $\boldsymbol{\phi}_L$ is assumed to be zero in pervaporation, $\boldsymbol{\phi}_{\text{avg}}$ is known in that work, whereas it depends on the unknown $\boldsymbol{\phi}_L$ here. The papers [37] and [48] propose similar approximations in conjunction with Pan's relationship for cases with unknown $\boldsymbol{\phi}_L$, but ignore thermodynamic coupling. Izák et al. [37] also make further simplifications that only apply to binary systems. Thus, the $\boldsymbol{\phi}$-form described above is a generalization of the methods in these papers. The $\mathbf{f}$-form approximation was proposed in Mathias et al. [14] and more recently by Marshall et al. [68].

2.3.3 Fugacity Form Finite Difference Method

Hesse et al. [36] proposed a numerical method based on finite differences (FD) for solving the full local flux problem without additional assumptions. This is the most closely related method in the literature to the methods proposed in this paper. Although the model used in [36] is equivalent to our formulation, Hesse et al. cast the equations in terms of membrane-phase mass fractions and mass-based fluxes. This is inconsistent with our derivations so

far and with the thermodynamic models used in our case studies, both of which are cast in terms of volume fractions and volume-based fluxes. Moreover, the final solution procedure in [36] is not described in sufficient detail to reproduce the method exactly. For both reasons, we chose to implement and compare against a variant that is consistent with our formulation but follows the main ideas in [36].

To formulate this method, we first replace the Maxwell-Stefan differential equations in the second block of Figure 2.2 with the equivalent $\mathbf{f}$-form in (Equation 2.31), which does not involve $\mathbf{\Gamma}$. This form is used by Hesse et al. and is a key difference between their approach and the finite difference method proposed in subsection 2.4.1. Next, the spatial domain of the active layer $[0, L]$ is discretized into $S + 1$ equally spaced nodes, $0 = z_0 < z_1 < \cdots < z_S = L$, each with corresponding volume fractions $\boldsymbol{\phi}_s$ and fugacities $\mathbf{f}_s^{II}$. Finally, (Equation 2.10),(Equation 2.13), and (Equation 2.31) are enforced at nodes 1 through S with the derivatives in (Equation 2.31) approximated by FD. The equations for a generic node $0 < s < S$ are

$$\begin{aligned}\mathbf{0} &= \mathbf{h}_s(\boldsymbol{\phi}_{s-1}, \mathbf{f}_{s-1}^{II}, \boldsymbol{\phi}_s, \mathbf{f}_s^{II}, \boldsymbol{\phi}_{s+1}, \mathbf{f}_{s+1}^{II}, \mathbf{x}^{III}, N_{\text{tot}}) \\ &\equiv \begin{bmatrix} \frac{\text{diag}(\boldsymbol{\phi}_{1:n,s}) \ln\left(\frac{\mathbf{f}_{s+1}^{II}}{\mathbf{f}_{s-1}^{II}}\right)}{z_{s+1} - z_{s-1}} + \mathbf{B}(\boldsymbol{\phi}_s)\text{diag}(\mathbf{V}^\circ)\mathbf{x}^{III} N_{\text{tot}} \\ 1 - \sum_{i=1}^{n+1} \phi_{i,s} \\ \mathbf{g}^{II}(\boldsymbol{\phi}_s, \mathbf{f}_s^{II}, T, P^{II}) \end{bmatrix},\end{aligned} \tag{2.33}$$

where again the natural log and the division within it are taken component-wise. The equations for the terminal node S are modified to use backward differences.

The final system of equations is obtained by replacing the entire second block of Figure 2.2 with the $(2n + 1)S$ equations $(\mathbf{h}_1, \ldots, \mathbf{h}_S) = \mathbf{0}$. The first and third blocks are unchanged. The numerical solution procedure is exactly analogous to that described in subsection 2.3.1 but with additional equations and unknowns corresponding to the S nodes.

2.4 Proposed Solution Methods

This section presents two new numerical methods for solving the local flux problem stated in Figure 2.2. The first is based on a FD approximation (subsection 2.4.1), while the second uses a novel shooting approach (subsection 2.4.2). Finally, automated initialization procedures for both methods are presented in subsection 2.4.3.

Our new methods are both based on the ϕ-form of the MS equation in (Equation 2.23) as opposed to the $\mathbf{f}$-form (Equation 2.30) used by Hesse et al. [36]. In fact, this is the only difference between our FD method and the one in subsection 2.3.3. We developed the shooting algorithm because shooting has several well-known advantages relative to FD and other simultaneous solution methods that seemed potentially important for treating complex mixtures (see section 2.5). In contrast, we originally developed our FD method simply to provide a more direct comparison of shooting with FD (i.e., using identical formulations). However, comparisons in subsection 2.5.4 show that this version of FD actually offers some significant advantages over the one in subsection 2.3.3.

2.4.1 Volume-Fraction Form Finite Difference Method

In this method, we again discretize the spatial domain of the active layer $[0, L]$ into S equally spaced nodes, $0 = z_0 < z_1 < \cdots < z_S = L$, and introduce the corresponding volume fractions $\boldsymbol{\phi}_s$ and fugacities $\mathbf{f}_s^{II}$. Next, the equations in the second block of Figure 2.2 are enforced at nodes 1 through S with the derivatives approximated by FD. The equations for a generic node $0 < s < S$ are

$$\mathbf{0} = \mathbf{h}_s(\boldsymbol{\phi}_{s-1}, \mathbf{f}_{s-1}^{II}, \boldsymbol{\phi}_s, \mathbf{f}_s^{II}, \boldsymbol{\phi}_{s+1}, \mathbf{f}_{s+1}^{II}, \mathbf{x}^{III}, N_{\text{tot}}) \tag{2.34}$$

$$\equiv \begin{bmatrix} \boldsymbol{\Gamma}(\boldsymbol{\phi}_s, \mathbf{f}_s^{II}) \frac{(\boldsymbol{\phi}_{1:n,s+1} - \boldsymbol{\phi}_{1:n,s-1})}{z_{s+1} - z_{s-1}} \\ + \mathbf{B}(\boldsymbol{\phi}_s) \text{diag}(\mathbf{V}^\circ) \mathbf{x}^{III} N_{\text{tot}} \\ 1 - \sum_{i=1}^{n+1} \phi_{i,s} \\ \mathbf{g}^{II}(\boldsymbol{\phi}_s, \mathbf{f}_s^{II}, T, P^{II}) \end{bmatrix}.$$

The equations for the terminal node S are modified to use backward differences. The final system of equations is obtained by replacing the entire second block of Figure 2.2 with the $(2n+1)S$ equations $(\mathbf{h}_1, \ldots, \mathbf{h}_S) = \mathbf{0}$. The first and third blocks are unchanged. The numerical solution procedure is exactly analogous to that described in subsection 2.3.1 but with additional equations and unknowns corresponding to the S nodes.

2.4.2 Volume-Fraction Form Shooting Method

This section presents a custom shooting approach for solving the local flux problem in Figure 2.2. The method is described in detail in Figure 2.3. To begin, the equations in the first block of Figure 2.2 are solved independently for $\boldsymbol{\phi}_0$, $\mathbf{f}_0^{II}$, $\mathbf{f}^\circ$, and $\boldsymbol{\gamma}^I$. The rest of the problem is split into outer and inner solves. The outer loop solves for $\mathbf{x}^{III}$ and N_{tot}. With these values fixed for any current iteration, $\boldsymbol{\gamma}^{III}$ can be obtained by solving (Equation 2.11). Moreover, fixing these variables reduces the problem to an initial value problem (IVP) in either ODEs or DAEs depending on whether (Equation 2.10) is explicit or implicit, respectively. In the general DAEs case (left branch of the inner solver box in Figure 2.3), the inner solver integrates the DAEs system consisting of (Equation 2.10), (Equation 2.13), and (Equation 2.23) with the initial conditions $\boldsymbol{\phi}_0$ and $\mathbf{f}_0^{II}$ from $z = 0$ to $z = L$ to obtain $\boldsymbol{\phi}_L$ and $\mathbf{f}_L^{II}$. At this point, all equations in the first and second blocks of Figure 2.2 are satisfied, as is the first equation in the third block. The remaining $n+1$ equations serve as a check on the current guesses for $\mathbf{x}^{III}$ and N_{tot}. Specifically, the outer

solver iteration is completed by explicitly calculating $\mathbf{x}^{III}$ from (Equation 2.5), comparing it to the guessed values, and checking that it sums to one. If these tests are satisfied, then the algorithm terminates. Otherwise, a new guess is generated. Any standard Newton-type equation solver can be used to generate these iterates and converge the outer loop. Similarly, any standard DAEs solver can be used to solve the IVP in the inner loop.

A significant advantage of this approach is that error control mechanisms in modern IVP solvers automatically determine a mesh over $[0, L]$ that effectively balances solution accuracy and efficiency. In contrast, FD methods require a mesh to be specified in advance and manually refined if the solution is insufficiently accurate. The shooting approach also eliminates the need for initial guesses for the nodal values $\boldsymbol{\phi}_s$ and $\mathbf{f}_s^{II}$, which can be a significant challenge for FD methods. These advantages are particularly important for stiff DAEs. Finally, our shooting algorithm is arranged so that the bulk phase activity coefficients $\boldsymbol{\gamma}^I$ and $\boldsymbol{\gamma}^{III}$ are always evaluated by solving (Equation 2.9) and (Equation 2.11) independently rather than within a system of coupled equations. This naturally accommodates "black-box" activity model evaluations using process flowsheet simulation software, which cannot be used easily in FD methods.

2.4.3 Initialization Strategies

All of the methods presented in section 2.3 through section 2.4 require initial guesses for $\boldsymbol{\gamma}^{III}$, $\mathbf{x}^{III}$, and N_{tot}. In addition, the approximation methods require guesses for $\boldsymbol{\phi}_L$ and $\mathbf{f}_L^{II}$, while the FD methods require guesses for all of the nodal values $\boldsymbol{\phi}_s$ and $\mathbf{f}_s^{II}$ with $s = 1, \ldots, S$. This section presents automated methods for providing all of these values. Methods for $(\mathbf{x}^{III}, N_{\text{tot}})$ and $(\boldsymbol{\phi}_s, \mathbf{f}_s^{II})$ are covered in subsubsection 2.4.3 and subsubsection 2.4.3, respectively, and we always guess $\boldsymbol{\gamma}^{III} \leftarrow \mathbf{1}$. As shown in section 2.5, these methods have a significant impact on the robustness and solution times of all algorithms. Moreover, they provide a standard initialization for comparing methods.

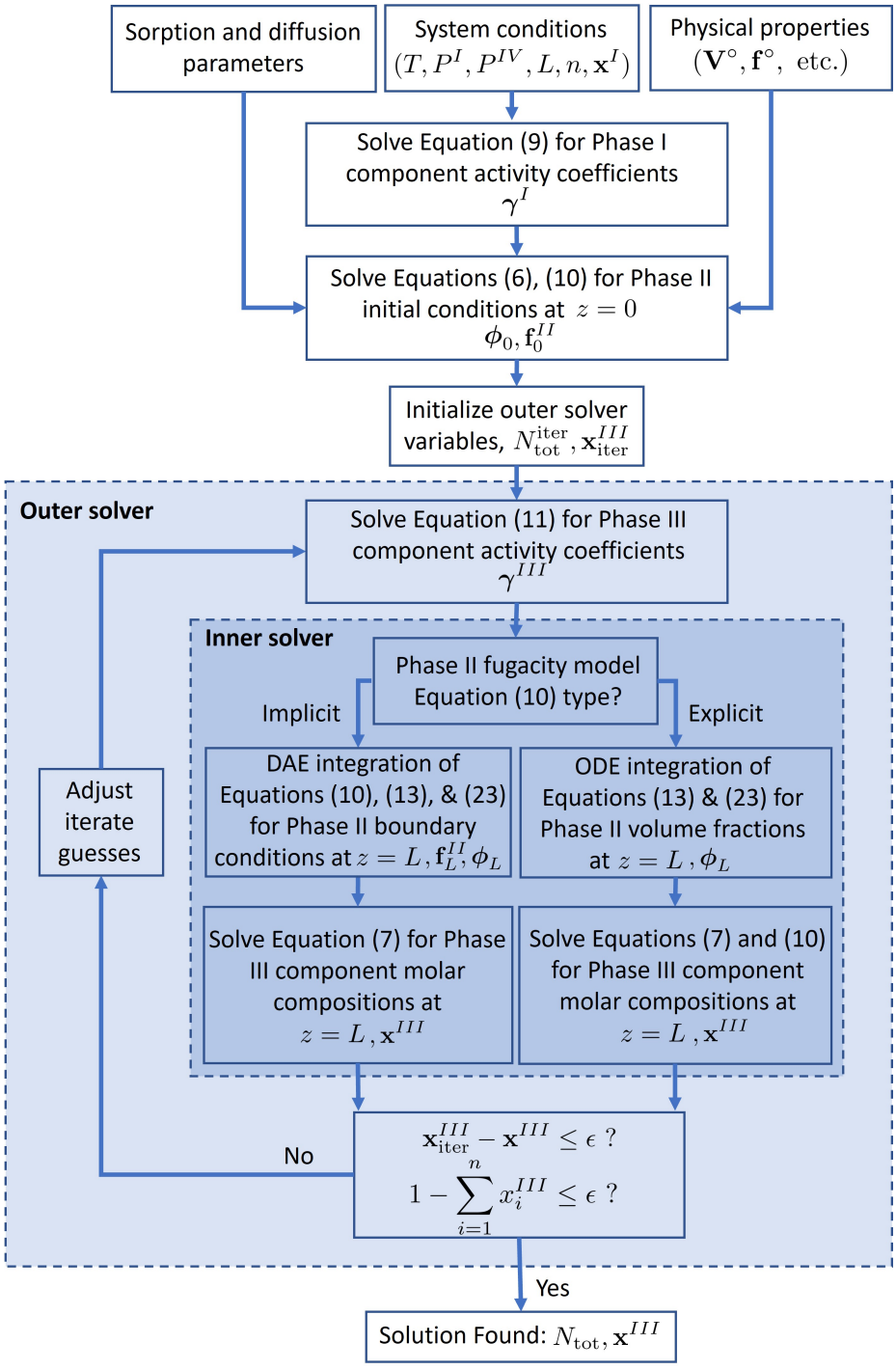


Figure 2.3: Shooting algorithm for the local flux problem in Figure 2.2. Different outer solvers may use slightly different termination criteria.

Explicit One-Step Shooting for Total Flux and Phase III Molar Compositions

A simple but sensible initial guess for $(N_{\text{tot}}, \mathbf{x}^{III})$ is to set $\mathbf{x}^{III} \leftarrow \mathbf{x}^{I}$ and $N_{\text{tot}} \leftarrow 0$. In the numerical comparisons in section 2.5, we refer to this as the "first-pass" initial guess. However, we found that this did not work robustly over a range of systems with different membrane thicknesses and diffusivities of different magnitudes. To generate guesses in a more adaptive manner, we implemented a variant of the shooting algorithm in which the forward integration step in the inner loop is replaced by a single step of the explicit Euler formula. We call this the explicit one-step shooting algorithm. The algorithm follows the flowchart in Figure 2.3 exactly until the outer solver is entered. The first block within the outer solver is replaced with the assumption that $\boldsymbol{\gamma}^{III} = \mathbf{1}$, after which the inner solver is entered. At this point, the initial conditions $\boldsymbol{\phi}_0$ and $\mathbf{f}_0^{II}$ are known. However, instead of numerically integrating the DAEs forward from these conditions, the terminal volume fractions are determined explicitly by

$$\boldsymbol{\phi}_L = \boldsymbol{\phi}_0 + \boldsymbol{\Gamma}^{-1}(\boldsymbol{\phi}_0, \mathbf{f}_0^{II})\mathbf{B}(\boldsymbol{\phi}_0)\text{diag}(\mathbf{V}^{\circ})\mathbf{x}^{III}N_{\text{tot}}L. \tag{2.35}$$

The terminal fugacity $\mathbf{f}_L^{II}$ is then obtained by solving (Equation 2.10). If (Equation 2.10) is explicit w.r.t. $\mathbf{f}^{II}$, then the inner solver becomes completely explicit. The rest of the algorithm proceeds exactly as in Figure 2.3 and the resulting solution $(N_{\text{tot}}, \mathbf{x}^{III})$ serves as the desired initial guess. In the numerical comparisons in section 2.5, we refer to this as the "informed" initial guess. Note that this simplified shooting algorithm itself requires an initial guess for N_{tot} and $\mathbf{x}^{III}$, which we set simply as $N_{\text{tot}} \leftarrow 0$ and $\mathbf{x}^{III} \leftarrow \mathbf{x}^{I}$.

Shooting-Based Initialization of Nodal Variables

For the approximation methods, reasonable guesses for $\boldsymbol{\phi}_L$ and $\mathbf{f}_L^{II}$ are given by $\boldsymbol{\phi}_L \leftarrow \boldsymbol{\phi}_0$ and $\mathbf{f}_L^{II} \leftarrow \mathbf{f}_0^{II}$. Recall that all methods solve the first block of equations in Figure 2.2 independently, so $\boldsymbol{\phi}_0$ and $\mathbf{f}_0^{II}$ are always known before initial guesses for $\boldsymbol{\phi}_L$ and $\mathbf{f}_L^{II}$ are

needed. We used these guesses for both the "first-pass" and "informed" initialization strategies. When the "informed" strategy is used, $\boldsymbol{\phi}_L$ and $\mathbf{f}_L^{II}$ could alternatively be initialized using the values of these variables at the end of the explicit one-step shooting procedure described above. However, we found that this did not add significant benefit.

For the FD methods in subsection 2.3.3 and §subsection 2.4.1, a simple strategy for initializing the nodal variables $\boldsymbol{\phi}_s$ and $\mathbf{f}_s^{II}$ is to guess that, for all components except the membrane itself, both ϕ_i and f_i^{II} decrease linearly from $\phi_{0,i}$ and $f_{0,i}^{II}$ at $s = 0$ to 0.1% of these initial values at $s = S$. As for the membrane volume fraction, it must increase as the other fractions decrease, so it is guessed to increase linearly from its initial value to 0.999. This approach is always used in the "first-pass" initialization strategy. To obtain more accurate guesses, we can alternatively execute a single forward integration from the known initial conditions $\boldsymbol{\phi}_0$ and $\mathbf{f}_0^{II}$ using an IVP solver exactly as in the inner solver in Figure 2.3. This requires values for N_{tot} and $\mathbf{x}^{III}$, which are obtained from the informed method in subsubsection 2.4.3. The results of this forward integration are then interpolated at each mesh point s to obtain guesses for $\boldsymbol{\phi}_s$ and $\mathbf{f}_s^{II}$. This approach is always used in the "informed" initialization strategy.

2.5 Numerical Comparisons

This section presents numerical results for all methods in section 2.3–section 2.4. Sections subsection 2.5.1–subsection 2.5.3 define the test cases used, the implementation details, and the metrics used to compare methods. Results are given in subsection 2.5.4–subsection 2.5.7.

2.5.1 Test Cases

We compare numerical methods using three test systems studied experimentally in Mathias et al. [14]. In these systems, three, five, and nine-component complex hydrocarbon mixtures defined in Table 2.1 permeate through glassy polymer membranes. The five-

component case uses a 1.5 μm polymer membrane of intrinsic microporosity (PIM-1) as the active layer (Phase II), while the other two use a 300 nm spirobifluorene aryl diamine (SBAD-1) membrane as the active layer (Phase II). The three and five component mixture were simulated with a 30 bar transmembrane pressure, while the nine component mixture was simulated with 40 bar transmembrane pressure ($P^{III} = 1$ bar). More details on these systems can be found in [14].

For each test system, we consider two different options for the active layer fugacity model (Equation 2.10): the multicomponent Flory-Huggins (FH) model and the Flory-Huggins-Langmuir (FH-LM) model from [14]. The latter model generalizes the classic dual-mode sorption model by replacing the Henry's law component with a more flexible FH model. This was shown to provide superior predictions for multicomponent permeation experiments with the three test systems in [14]. Both models are described further in subsubsection 3.2.2.

The FH model specifies the fugacity $\mathbf{f}^{II}$ explicitly as a function of $\boldsymbol{\phi}$. In contrast, the FH-LM model is implicit. As discussed in subsection 2.2.4, this distinction implies that the FH-LM case requires the general DAEs formulation presented in Figure 2.1, while the FH case can be formulated more simply as an ODEs model. Specific implementation differences are discussed in section A.4. In the discussions below, the FH and FH-LM models will be referred to as "explicit" and "implicit" fugacity models, respectively.

In all three test systems, we assume that the bulk feed and permeate phases behave ideally; i.e., $\boldsymbol{\gamma}^{I} = \boldsymbol{\gamma}^{III} = \mathbf{1}$. This allows our comparisons to focus on the key differences in the numerical methods, which all relate to how they handle complexity in the membrane model. Since all methods handle the feed-side calculations in essentially the same way, introducing a numerically challenging feed activity model would not provide additional insight and could make the results harder to interpret.

The complete set of test cases is summarized in Table 2.2. All model parameters required to simulate these cases (diffusivities, thermodynamic parameters, feed conditions,

Table 2.1: Feed compositions for each of the three test cases.

n	Mixture Components	Mol%
3	toluene	28.4
	iso-octane	38.8
	iso-cetane	32.8
5	toluene	25.7
	heptane	21.6
	p-xylene	20.5
	o-xylene	26.4
	iso-cetane	5.8
9	toluene	17.1
	methylcyclohexane	28.1
	1-methylnaphthalene	2.0
	decalin	10.7
	n-octane	22.1
	iso-octane	15.0
	tert-butlybenzene	2.1
	1,3,5-triisopropylbenzene	1.6
	iso-cetane	1.3

Table 2.2: Test cases for numerical method comparisons.

Fugacity Model	n, Membrane
Explicit	3, SBAD-1
	5, PIM-1
	9, SBAD-1
Implicit	3, SBAD-1
	5, PIM-1
	9, SBAD-1

etc.) were obtained directly from [14]. For convenience, they are listed in section A.11

2.5.2 Implementation Details

All simulations were done on a laptop with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz and 16 GB of RAM. The code is implemented in MATLAB 2020b using the built-in non-linear equation solver *fsolve*, the stiff ODEs integrator *ode15s*, and the implicit differential equation integrator *ode15i*. The integrators *ode15s* and *ode15i* are used for the shooting algorithm with explicit and implicit thermodynamic models, respectively, while *fsolve* is

used for all methods. We used 10^{-6} for the *fsolve* function tolerance and first-order optimality tolerance, 10^{-4} for the integrator relative tolerances, and 10^{-6} for the integrator absolute tolerances. The *fsolve* default algorithm is "trust-region-dogleg".

To keep the equations and variables well-scaled, the code uses different units than presented in section 2.2. Specifically, we scale $Đ_{ij}^{V}$ to $\mu\text{m}^2\text{s}^{-1}$, z to μm, f° and f_i to torr, b_i to torr, N_{tot} to mol μm^{-2} s^{-1}, and V_i° to μm^3 mol^{-1}. The other known and unknown variables in Figure 2.2 are left the same as defined in the nomenclature table.

The multicomponent FH and FH-LM fugacity models both involve multiple nested summations (see section A.3 and section 3.3 for more details) that are not evaluated efficiently in *MATLAB*. Indeed, in preliminary experiments, the evaluation of these models and the associated $\mathbf{\Gamma}$ matrices consumed significant computational time in all methods. Therefore, we derived alternative matrix-vector forms of the FH and FH-LM formulas, as well as the respective $\mathbf{\Gamma}$ matrices, that can be evaluated using fast linear algebra routines native to *MATLAB*. These formulas decreased convergence time by at least 30% for the approximation methods with FH and led to a $4\times$ speed up for the shooting algorithm using FH-LM. As an added advantage, they are simpler to implement and allow the $\mathbf{\Gamma}$ matrices to be derived much more easily using standard matrix calculus rules. These formulas are used in all numerical experiments. Details are given in section A.3 and section 3.3.

2.5.3 Comparison Metrics

All methods are compared in terms of efficiency, accuracy, and robustness. Efficiency is characterised by reporting the wall-clock time required by each method to converge to a solution. Accuracy is characterized by comparing the converged solution of each algorithm against the true mathematical solution of the complete local flux model in Figure 2.2. We compare against the true model solution rather than experimental data because our primary aim is to assess the ability of each numerical method to solve the postulated model. This question is independent of whether the model matches experiments, and in general these

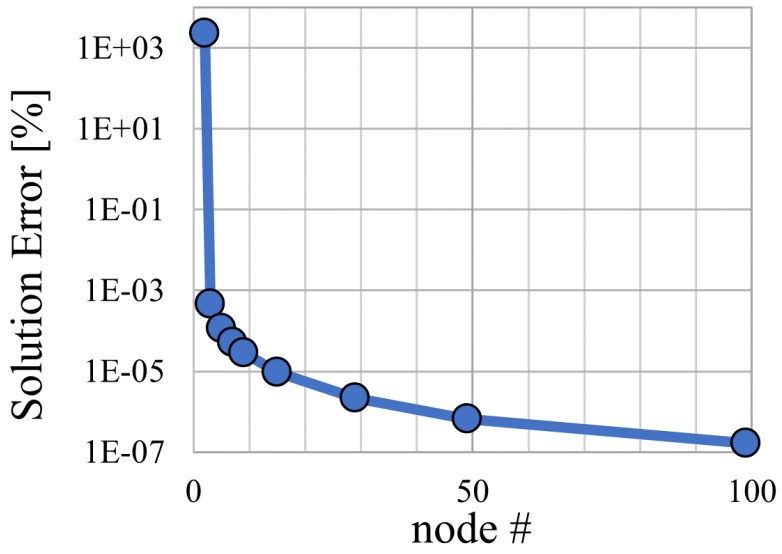


Figure 2.4: Convergence of the solution error between FD and the shooting algorithm with increasing number of nodes for the volume fraction form FD method applied to the nine-component SBAD-1 test case with the Flory-Huggins-Langmuir implicit fugacity model.

two errors may compound or cancel. The interested reader is referred to [14] for comparisons of many of the methods considered here to experimental results.

To obtain the true solution for each case, the complete model was solved from high-quality initial guesses with both the shooting algorithm in subsection 2.4.2 and the FD method in subsection 2.4.1 with the number of nodes increased until the solution no longer changed appreciably. We then verified that both approaches yielded the same true solution to very high accuracy for all test cases (see Figure 2.4).

Given the true solution, the error for the solution obtained by a numerical method is defined as the absolute mean percentage error of all solution components as

$$\text{Error } \% = \frac{\sum_{i=1}^{n} \left| \frac{x_i^{III,*} - x_i^{III}}{x_i^{III,*}} \right| + \left| \frac{N_{\text{tot}}^{*} - N_{\text{tot}}}{N_{\text{tot}}^{*}} \right|}{n+1} \times 100. \tag{2.36}$$

By this definition, our shooting algorithm has near-perfect accuracy whenever it converges because it solves the full model. The same is true of both FD methods provided S is large enough, but more generally these methods should be viewed as providing a trade-off

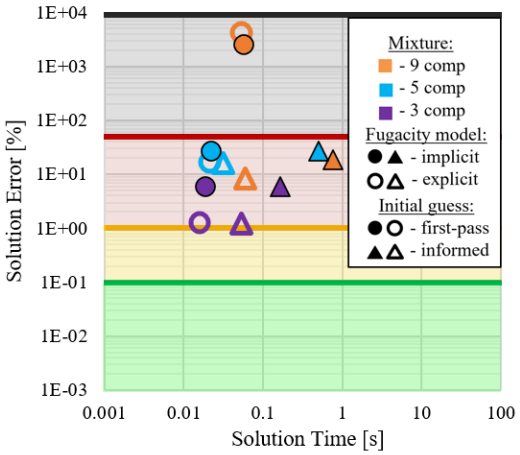
between accuracy and efficiency governed by S. Finally, the approximation methods are expected to have moderate errors in most cases because they solve simplified models. In our comparisons, we view all methods with $\leq$0.1% error as maximally accurate. Furthermore, errors of 0.1–1% are viewed as good enough for most purposes (i.e., well within typical experimental error). Errors of 1–50% are viewed as the result of inaccurate approximations or low S, while those above 50% are most likely due to convergence failures. For later figures, shaded regions correspond to the following error ranges: black >50%, red 1–50%, yellow 0.1–1%, and green <0.1%. These categories are not precise, but are useful for collecting informative statistics. To obtain a favorable S for comparing FD methods, for each FD method and each test case, we increased S starting from S_0 until either: (i) the solution error was less than 0.1%, (ii) the simulation time was more than $2.5\times$ that of the shooting algorithm, or (iii) the maximum number of nodes $S = 30$ was reached. We report the error and solution time for only the last solve with the final S value. We used $S_0 = 1$ in most cases. However, in some cases, solution time was not monotonically increasing with S due to convergence issues at low S, which caused criterion (ii) to trigger prematurely. In such cases, a suitable S_0 was determined manually. Using this procedure, in the absence of convergence failures, FD methods will achieve maximal accuracy unless doing so requires large S or a wall-clock time that is no longer competitive with shooting. Note that this methodology is favorable to the FD methods because it does not consider the computational effort required to determine a suitable S.

Lastly, robustness is assessed by determining the ability of each method to converge from various initial guesses. For each method, we first compare the convergence behavior when using the first-pass and informed initial guess strategies outlined in subsection 2.4.3. Then, in subsection 2.5.7, we evaluate the fraction of cases converged when the method is challenged with 300 randomly generated initial guesses.

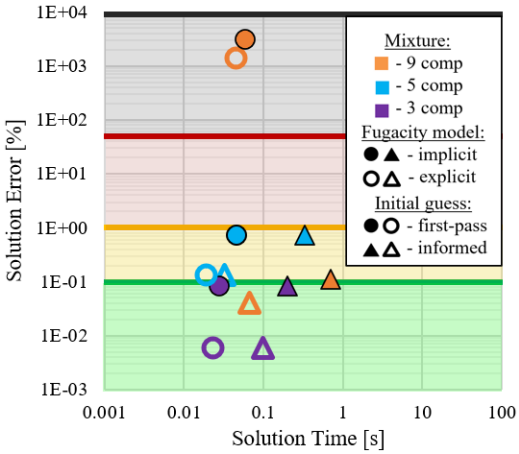
2.5.4 Approximation Methods

This section presents numerical results for the Fick's law and average coupling approximations discussed in subsection 2.3.1 and subsection 2.3.2. Figure 2.5 shows the solution error and solution time for all test cases. The colored regions correspond to the error regions defined in subsection 2.5.3. All methods converge for all test cases except the 9 component case with both explicit and implicit fugacity models. In that case, all methods fail with the first-pass initial guess but succeed with the informed initial guess, which shows the importance of good initial guesses for complex mixtures. On the other hand, the proposed initialization strategy appears to be too heavy-handed for these simple methods, significantly increasing the solution time for all cases and adding nearly an order of magnitude for the implicit fugacity model cases. Even so, all methods are fairly efficient, taking less than a second for all converged cases and only 0.01–0.1s when using first-pass initial guesses.

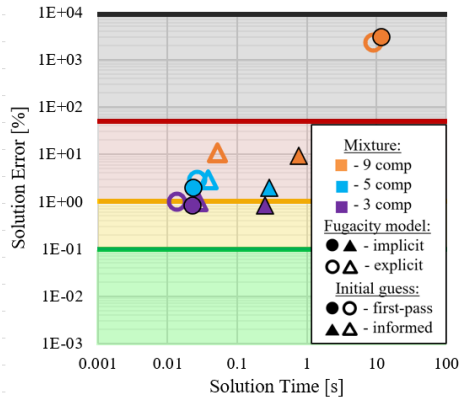
The Fick's law approximation results in large errors for all test cases. The solution error is nearly acceptable (<1%) for the 3 component case with an explicit fugacity model. However, for higher component numbers and the implicit fugacity model, the majority of runs have large error. This is attributed to the assumption that $\mathbf{\Gamma}$ and $\mathbf{B}$ are diagonal and constant. Figure S1 in the SI shows that $\mathbf{\Gamma}$ has significant off-diagonal entries when evaluated at $z = 0$ for the 5 and 9 component test cases. The diffusional coupling matrix, $\mathbf{B}$, also has non-negligible off-diagonal elements (not shown). Moreover, the implicit fugacity model has larger off-diagonal elements compared to the explicit model and contains negative values, as seen in Figure S1. Because of these issues, both of the average coupling approaches are more accurate than Fick's law. The $\boldsymbol{\phi}$-form average coupling approximation is the most accurate for these test cases, with less than 1% error for all cases that converged, while the $\mathbf{f}$-form has less than 10% error for all cases that converged. However, the $\mathbf{f}$-form has been shown to be more competitive with the $\boldsymbol{\phi}$-form for cases without diffusional coupling in [14, 68]. To aid in understanding these results, the transmembrane volume fraction profiles



(a) Fick's Law Approximation



(b) ϕ-form Average Coupling Approximation

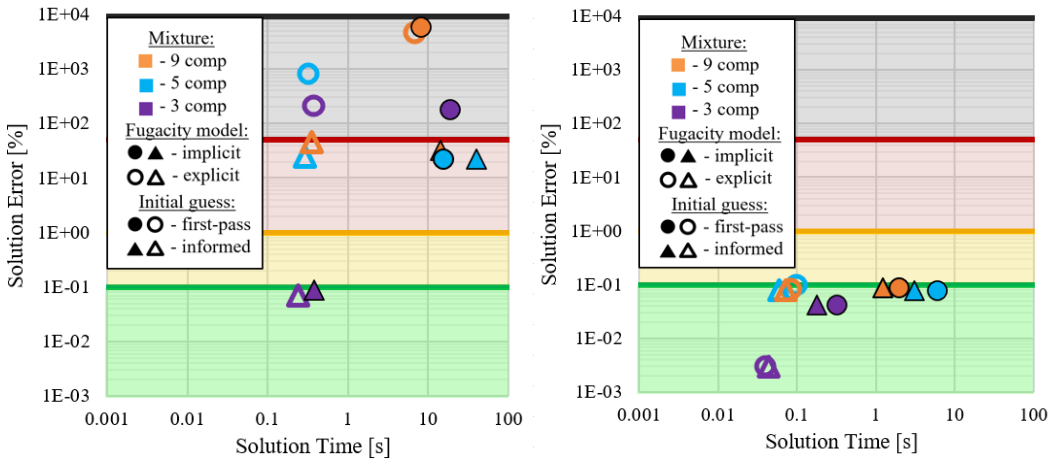


(c) f-form Average Coupling Approximation

Figure 2.5: Error (%) versus solution time (s) for the Fick's law and average coupling approximations.

and total fluxes predicted by each method for the five and nine component test cases with implicit fugacity model are shown in section A.8 and compared to the full model solutions.

To conclude, Fick's law is too inaccurate for these complex mixtures, while the ϕ-form average coupling approximation outperforms the f-form average coupling approximation with respect to accuracy by an order of magnitude. Notably, however, all methods show convergence errors with high component number and first-pass initial guesses.



(a) f-form FD (b) ϕ-form FD

Figure 2.6: Error (%) versus solution time (s) for finite difference methods.

2.5.5 Finite Differences Methods Comparison

Figure 2.6 shows the solution error and solution time for all test cases for the **f**-form and ϕ-form FD methods described in subsection 2.3.3 and subsection 2.4.1, respectively.

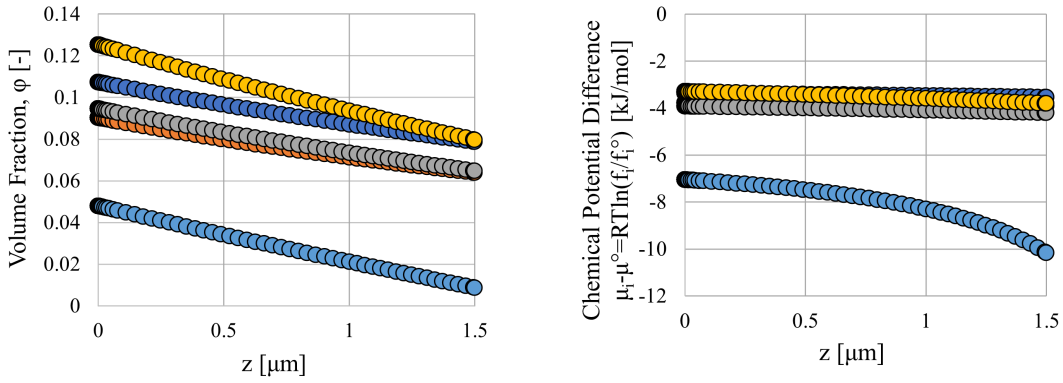
The most notable distinction between these methods is their robustness. The ϕ-form method converges for all cases from both first-pass and informed initial guesses, while the **f**-form fails for all but one case with first-pass guesses (according to the >50% error criterion discussed in subsection 2.5.3). Notably, although the informed initial guesses are not required for convergence of the ϕ-form method, they do lead to faster solution times in five out of six cases, showing that the added cost of good initialization is compensated by faster convergence. The ϕ-form method is also much more accurate, achieving the 0.1% error target in all cases, while the **f**-form only achieves this for the 3 component case. Although both methods can achieve arbitrary accuracy with sufficiently large S, recall that the methodology for determining S described in subsection 2.5.3 will not increase S further if $S = 30$ or the solution time for the current S is more than $2.5\times$ that of the shooting algorithm for the same case. Therefore, the accuracy results in Figure 2.6 actually indicate that the ϕ-form method does not reach these limits for any test cases, while the **f**-form method reaches them in all but two. Further investigation shows that all cases with

$> 0.1\%$ error in Figure 2.6a reached the solution time limit specifically. Thus, the $\mathbf{f}$-form method either fails to converge altogether, or requires significantly more time than the $\boldsymbol{\phi}$-form method to achieve similar accuracy.

One possible explanation for these trends is that, for these complex mixtures, some of the chemical potentials $RT\ln(f_i^{II})$ vary with z in a highly nonlinear way, whereas all of the volume fractions ϕ_i change nearly linearly. This can be seen in Figure 2.7 and Figure 2.8, which show profiles obtained using a high-order numerical integrator with state-of-the-art error control, as in the inner-loop of the shooting algorithm. As a result, more nodes are required to approximate the derivatives $\frac{d\ln(\mathbf{f})}{dz}$ to a desired accuracy as compared to $\frac{d\boldsymbol{\phi}}{dz}$. Indeed, the $\boldsymbol{\phi}$-form FD method requires only 3–9 nodes to solve all test cases to within 0.1% error, whereas the $\mathbf{f}$-form method requires 17 nodes to achieve 0.1% error for the 3 component case with the explicit fugacity model.

The poor results of the $\mathbf{f}$-form method for all other cases in Figure 2.6 are, however, better explained by another critical issue. Namely, the appearance of the terms $\ln(\mathbf{f}_s^{II})$ in the equations for the $\mathbf{f}$-form method causes these equations to be undefined (or produce imaginary values) when evaluated with non-positive f_i^{II} values. Unfortunately, we found that the solver often produces iterates with such f_i^{II} values and, in many cases, never recovers. This happens reliably from the first-pass initial guesses, but also to a lesser extent from the informed guesses. For this reason, the $\mathbf{f}$-form accuracy results in Figure 2.6 do not improve even when the number of nodes is set to 50 and the time limit is relaxed to 2.5 minutes. These results are shown in section A.9.

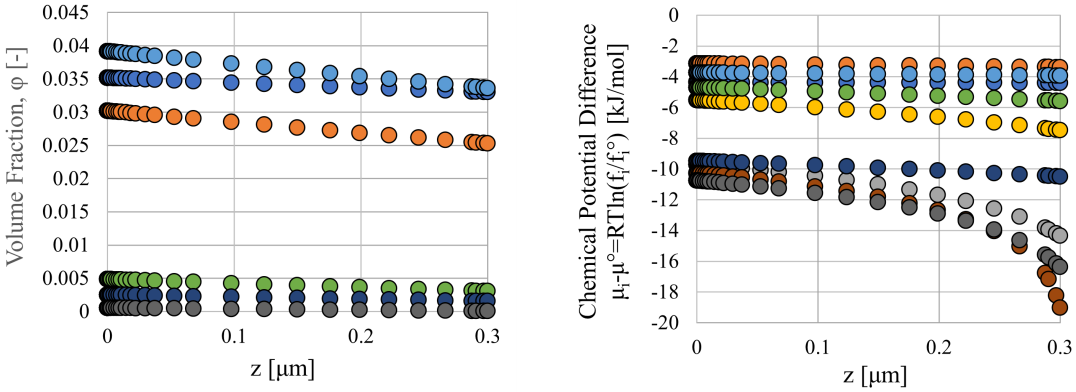
Compared to the best approximation method ($\boldsymbol{\phi}$-form average coupling), the $\boldsymbol{\phi}$-form FD method is significantly more accurate in all cases. It is also more robust since it avoids the convergence failure for the 9 component test cases with first-pass initial guesses. However, it is also slower in most cases, often by a factor of 2–10$\times$, particularly with the implicit fugacity model. Thus, the $\boldsymbol{\phi}$-form average coupling model may be preferable when $\sim$1% error is acceptable and some convergence failures can be tolerated.



(a) Volume Fraction Profiles

(b) Chemical Potential Profiles

Figure 2.7: Volume fraction and chemical potential profiles through the membrane active layer for the five-component mixture with the Flory-Huggins fugacity model. Each line represents a different component. Some components overlap and the membrane phase volume fraction, ϕ_m, is excluded from this figure.



(a) Volume Fraction Profiles

(b) Chemical Potential Difference Profiles

Figure 2.8: Volume fraction and chemical potential profiles through the membrane active layer for the nine-component mixture with the Flory-Huggins fugacity model. Each line represents a different component. Some components overlap and the membrane phase volume fraction, ϕ_m, is excluded from this figure.

2.5.6 Shooting Algorithm

Figure 2.9 shows the solution error and solution time for all test cases for the shooting method described in subsection 2.4.2. No convergence failures were observed and highly accurate solutions were obtained for all cases with solution times between 0.07 and 3 seconds. The informed initial guess had little effect on the convergence behavior but slightly increased computational time in most cases.

Compared to the ϕ-form average coupling results presented in Figure 2.5, the shooting algorithm is consistently more accurate and robust. However, if an informed solution strategy is used and 1% error is acceptable, then the ϕ-form average coupling approximation provides solutions much more efficiently. Compared to the ϕ-form FD method, the shooting algorithm is more accurate for all cases and more efficient in most cases. However, if 0.1% error is acceptable, then the ϕ-form FD method with the proposed initialization strategies provides competitive results. Yet, the FD approach requires the user to manually tune the number of nodes. In contrast, the shooting algorithm automatically determines a discretization that achieves a good balance between accuracy and efficiency by exploiting the adaptive step-size and error control mechanisms within the IVP solver used in the inner solve. Moreover, the solution times for the shooting algorithm could be reduced by relaxing the solver tolerances used by *fsolve* and/or the inner IVP solver such that the resulting solution errors are on the order of 0.1%. However, preliminary experiments with this approach (not shown) indicated that the speed-up is not very significant.

2.5.7 Extended Robustness Comparisons

To provide a more comprehensive test of robustness, all methods were challenged with a large number of random initial guesses for $\mathbf{x}^{III}$ and N_{tot}. The generation of these guesses is discussed below. The remaining unknowns for each method are initialized as follows. As described in subsection 2.5.1, $\boldsymbol{\gamma}^I = \boldsymbol{\gamma}^{III} = \mathbf{1}$ in all cases. For the approximation methods, $\boldsymbol{\phi}_L$ and $\mathbf{f}_L^{II}$ are initialized to their respective values at $z = 0$. For the FD methods, the nodal

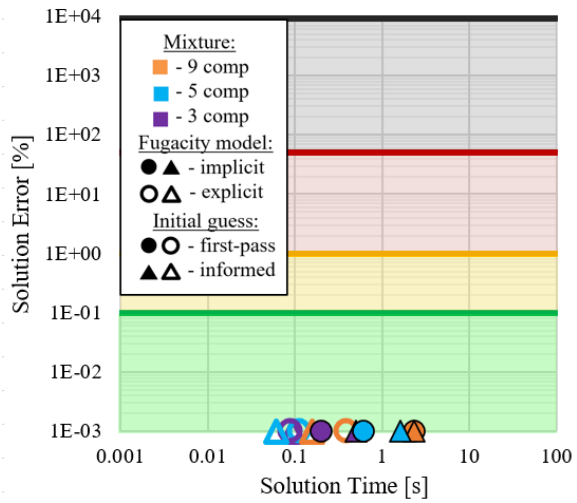


Figure 2.9: Error (%) versus solution time (s) for the shooting algorithm. The true solution errors for all cases are actually less than 1E-10, but the values are clipped at 1E-3 for consistency with the other figures.

variables ϕ_s and $\mathbf{f}_s^{II}$ were handled as described in subsubsection 2.4.3. Specifically, they are specified following either the procedure used in the 'informed' initialization strategy in subsubsection 2.4.3, or the procedure used in 'first-pass' strategy. We refer to these different methods here as either with or without nodal good initial guess (NGIG). These terms are used in place of 'informed' and 'first-pass' in this section because they apply only to the nodal values ϕ_s and $\mathbf{f}_s^{II}$, while we consider random initial guesses for $\mathbf{x}^{III}$ and N_{tot}.

The random initial guesses for $\mathbf{x}^{III}$ and N_{tot} were split into three sets of 100 points each representing base-case, worst-case, and best-case guesses. In the base-case set, each component of the initial guess for $\mathbf{x}^{III}$ is a random number of the form $a \times 10^{-b}$, where a is uniformly distributed in the interval $[0, 1]$ and b is a random integer in the interval $[1, 8]$. The guess for N_{tot} is generated in exactly the same way and then multiplied by 0.01 to account for the fact that the total molar flux is experimentally observed to be on the order of 0.01 $\text{mol}/\text{m}^2\text{s}$. Note that all variables are non-negative in the base-case set.

The best-case and worst-case sets were generated as modifications of unique base-case sets. To generate each worst-case guess from the corresponding base-case guess, each

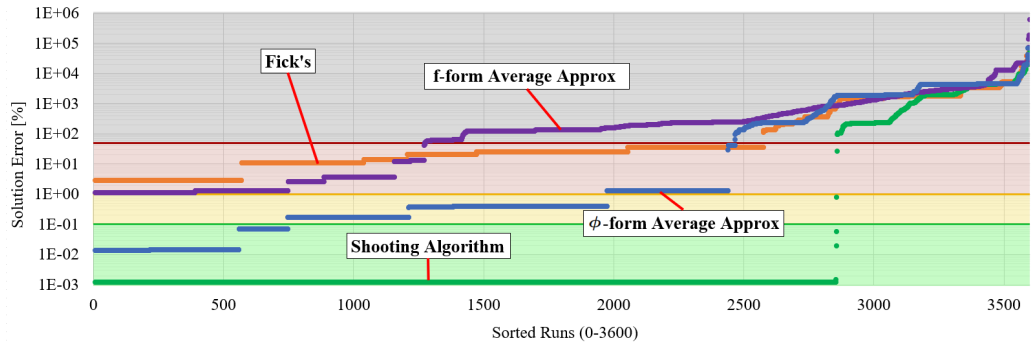
variable in the base-case guess was multiplied by either -1 or 1 with equal probability, specifically to generate non-physical values. In contrast, to generate each best-case guess from the corresponding base-case guess, $\mathbf{x}^{III}$ was scaled so that the mole fractions sum to one, making the guess more physically reasonable.

The rationale for using random guesses, and particularly for using non-physical guesses, is as follows. Although we typically have control over the initial guess provided to a solver, we generally have no control over the guesses that the solver itself generates in subsequent iterations. In practice, it is common for at least some of these guesses to be inaccurate or even non-physical in basic ways, such as having some variables negative. Thus, it is important to understand how different numerical methods and formulations respond to such points. Explicitly providing such points as initial guesses allows us to thoroughly sample the space of possible iterates and ensure that all methods are subjected to the same set of test points.

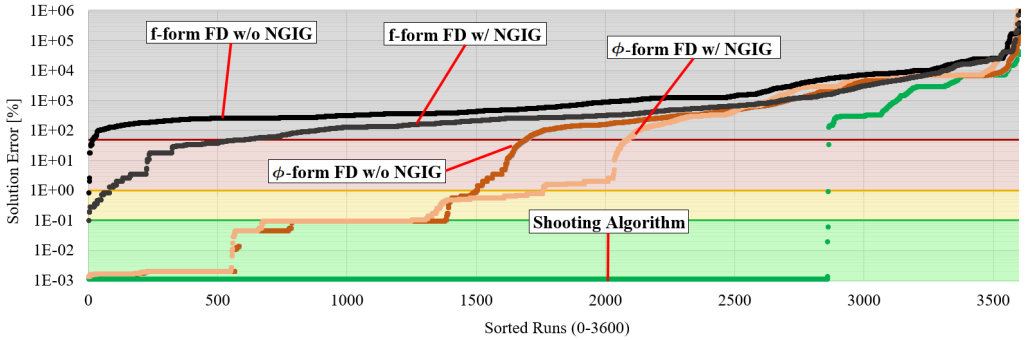
These 300 initial guesses were generated for each of the three mixtures described in subsection 2.5.1 and combined with each of the two fugacity models described in subsection 2.5.1 to create 1800 test cases. For each numerical method, we also considered using *fsolve* with the solver algorithm set to trust-region-dogleg (default) and Levenberg-Marquardt, making 3600 test cases per method. These test cases were solved with each of the six numerical methods presented in section 2.3 and section 2.4. Moreover, the FD methods were tested separately with and without NGIG.

For each numerical method, Figure 2.10 shows the solution errors obtained for all 3600 runs in monotonically increasing order. These results are summarized more concisely in Figure 2.11, which shows the fraction of runs with solution errors $> 50\%$ (did not converge), 1–50% (large approximation error), 0.1–1% (acceptable error), and $< 0.1\%$ (effectively no error).

Figure 2.10a compares the shooting algorithm to all approximation methods. The shooting algorithm converges in 79% of all runs, with high accuracy as expected, and



(a) Approximation Methods Robustness Comparison



(b) Finite Difference Methods Robustness Comparison

Figure 2.10: Solution errors for all methods on 3600 test cases with random initial guesses. The results for each method are sorted in order of increasing error. Shaded regions correspond to the following error ranges: >50%, 1–50%, 0.1–1%, <0.1%. NGIG = nodal good initial guess. For the shooting algorithm, the solution error is often much less than 1E-3, but the values are clipped at 1E-3 for plotting.

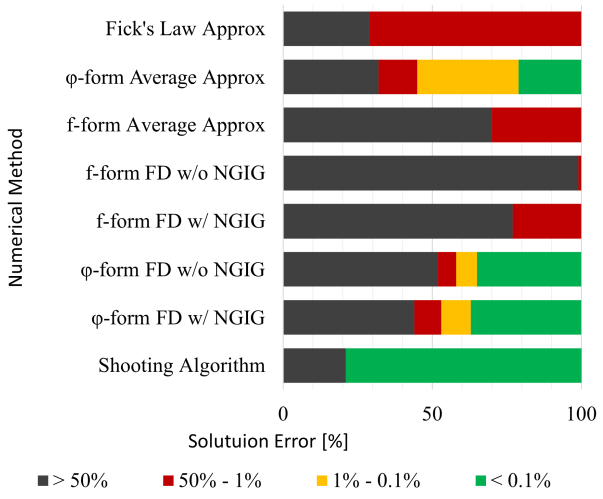


Figure 2.11: Percentage of runs from Figure 2.10 with errors >50%, 1–50%, 0.1–1%, and <0.1%.

fails to converge in the remaining 21%. The next best method is the ϕ-form average coupling approximation, which achieves an acceptable error (<1%) in 58% of cases and fails to converge in 31% (10% more than shooting). Finally, the Fick's law and $\mathbf{f}$-form average approximations fail to converge in 41% and 65% of cases, respectively.

Figure 2.10b compares the shooting algorithm to the $\mathbf{f}$-form and ϕ-form FD methods. The shooting method is evidently much more robust than either FD method. The $\mathbf{f}$-form FD performs the worst, with a convergence failure rate of 99% without using the NGIG strategy and 77% with NGIG. The ϕ-form FD performs significantly better with a convergence failure rate of 54% without NGIG and 44% with NGIG.

To understand the impact of the initial guesses, we also analyzed the results separately for the base, best, and worst-case guess sets. The results for all sets (not shown) were found to be very similar, giving percentages within a few points of the aggregated values in Figure 2.11 for all methods.

Based on these results, we conclude that the shooting algorithm converges much more reliably than any of the competing methods for this test set. The convergence failures that do occur in the shooting algorithm are nearly always caused by failures of the IVP solver used in the inner solve, which may result from the current iterate creating an unstable IVP. In contrast, the more frequent convergence failures in the other methods are likely related to the highly nonlinear phase equilibrium problem at the permeate side of the active layer, for which good initial guesses are not known. The shooting algorithm largely avoids this issue because the inner IVP solver propagates the key variables ϕ and $\mathbf{f}^{II}$ from the feed side $z = 0$ to the permeate side $z = L$ smoothly through a sequence of small, sequential steps. More specifically, within the IVP solver, each incremental step through the active layer involves computing ϕ and $\mathbf{f}^{II}$ at the current location z using very high-quality initial guesses derived from the converged values of ϕ and $\mathbf{f}^{II}$ at previous step $z - \Delta z$. Thus, the shooting method avoids ever solving a phase equilibrium problem from scratch, except the one at feed side itself, which is typically easier since $\mathbf{x}^{I}$ is known. This advantage

of shooting is expected to be more pronounced the more complex the membrane fugacity model becomes, which is important for complex mixtures.

The number of convergence failures observed for these methods is somewhat dependent on our decision to use *fsolve* rather than some other general-purpose nonlinear equation solver. However, the Newton-Raphson-type methods used by *fsolve* are standard in such solvers and provide a good representative benchmark. Notably, changing the *fsolve* algorithm option from trust-region-dogleg to Levenberg-Marquardt had very little impact on the robustness results. This can be seen in §S8 of the SI, which parses the results in Figure 2.10a–Figure 2.10b by the algorithm option used. Moreover, note that the most significant convergence problems arise in the $\mathbf{f}$-form average approximation and FD methods. As discussed in subsection 2.5.5, this is largely caused by iterates falling outside of the domains of the natural logarithms in those formulations, which is likely to cause problems for any general-purpose solver.

2.6 Conclusions

This chapter presented improved numerical methods for solving the local flux problem and provided extensive numerical comparisons with existing methods in the context of organic reverse osmosis with complex hydrocarbon mixtures. Our proposed shooting algorithm provided the best all-around performance in terms of accuracy, efficiency, and robustness. Our proposed ϕ-form FD method showed significant improvement over the existing $\mathbf{f}$-form FD method and nearly matched the shooting method in accuracy and efficiency, but not robustness. Moreover, shooting is preferred due to the manual tuning required by FD methods. Finally, the ϕ-form average approximation method performed surprisingly well for the complex mixtures tested, drastically outperforming Fick's law and offering a low-cost alternative to shooting with reductions of accuracy and robustness that will often be acceptable. Our complete implementation of these methods is available as an open-source *MATLAB* package called *aysMemSim*. See section 2.8 for the download details. The

software features scripts for fitting model parameters to single component data and running predictive simulations with any of the models presented in this paper.

Although the shooting algorithm outperformed the FD approaches tested here, we acknowledge that future advances in FD approaches could make them significantly more powerful. Specifically, their robustness could potentially be improved through the use of alternative nonlinear equation solvers or customized iterative schemes.

While the local flux problem addressed in this paper is a suitable stand-alone model for ideal cross-flow contactors (as in most experimental set-ups), additional work is required to extend our results to the simulation of industrial membrane modules in spiral-wound, hollow-fiber, plate-and-frame, or tubular configurations. Such global module models involve partial differential-algebraic equations describing a continuum of local flux problems coupled together by feed and permeate channel balances. Among the many possible approaches, the shooting algorithm proposed here could be extended to such cases through the use of a method-of-lines-type partial discretization.

2.7 Funding

This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Advanced Manufacturing Office (award no. DE-EE0007888).

2.8 Code Availability

The code used in this work may be found at
https://doi.org/10.5281/zenodo.8040519.

CHAPTER 3
EXTENDED MODELS FOR PREDICTING COMPLEX MIXTURE PERMEATION

3.1 Overview

This chapter outlines two different extended modeling applications (see section 3.2 and section 3.3). In the first application, a framework for predicting fractionation of complex mixture feeds via polymer membranes was developed for organic solvent reverse osmosis. This frameworks presents a novel sorption model, and two diffusion models that better predict the transport through two glassy polymers. The theory behind them is presented and compared to state-of-the-art modeling approaches. The conclusion from this work is that the proposed models are much better at describing the permeation and thermodynamic sorption phenomena observed [14]. In the second application, data-driven predictions of complex organic mixture permeation in polymer membranes is investigated. Given that, section 3.3 will outline how the underlying sorption model was better represented mathematically in order to realistically converge simulations of complex mixture permeation that included hundreds of components [69]. Then, modeling predictions show how the machine-learned parameters were able to predict the separation of crude oil mixtures with polymers that have never before been synthesized.

3.2 Framework for Predicting the Fractionation of Complex Liquid Feeds via Polymer Membranes

3.2.1 Introduction

Most commodities derived from petroleum and biorefining feedstocks involve the separation of molecules in the 100-400 g/mol range, and the majority of these separations are

achieved via thermally-intensive distillation and extraction processes. As industries seek to reduce the energy and resource consumption associated with separation processes, membranes have emerged as attractive low-energy options either in hybrid separation process configurations or as standalone alternatives. Importantly, real mixtures that need to be separated are often concentrated liquids that are "complex" in nature; in this section, we use "complex" to identify streams with many components without a clear majority species. Crude oil refining represents an important target for membrane-based separations, as the initial fractionation of crude oil is commonly done by distillation, which accounts for nearly 15% of all US manufacturing energy [70]. Thermal fractionation of liquid hydrocarbons can, in principle, be translated to membrane separations via membrane cascades containing varying separation modalities (e.g. OSN, organic solvent reverse osmosis (OSRO)) [71, 72]. The ability to predict multicomponent transport in many different classes of membranes is essential to accelerate the development of materials for such cascade systems that would otherwise require lengthy experimental timelines for material synthesis, membrane fabrication, and separation testing.

Significant attention has been paid to experimental and theoretical aspects of membrane-based separations of dilute mixtures [67, 72]; however, there is a lack of methods available for modeling complex mixture permeation data or predicting complex mixture permeations based on easily accessible experimental parameters. Complex liquid permeation of similarly-sized molecules, wherein multiple components are present in high concentrations (such as in crude oil separations), has not yet been successfully modeled and matched with experimental data in a scalable manner for polymer systems. The challenge is accurately describing both multicomponent liquid occupancy throughout a polymer membrane and diffusional cross-coupling for multiple species with similar physicochemical properties. Several studies have modeled liquid transport in polymer membranes via pore-flow and solution-diffusion models [37, 44, 67]. Pore-flow models assume no change in solvent activity across the thickness of a membrane, which does not match experimental observa-

tions for microporous or dense polymer materials, especially in the case of reverse osmosis regimes [73, 74]. It is now general consensus that the permeation of molecules through dense membranes is best described by the solution-diffusion model, whose driving force is a pressure-induced concentration gradient and not a pressure gradient across the membrane thickness [19, 67].

Several researchers have published detailed studies of small organic molecule flux through OSN and OSRO membranes that provide the backdrop for understanding complex organic transport [75, 76, 77, 78]. A number of researchers have applied Paul and Ebra-Lima's solution-diffusion model in Fick's law form to describe the transport of pure molecules or dilute mixtures through polymeric membranes and observed a close agreement with experimental data [79, 80, 81]. Although Fick's law is easily combined with mass balances and requires less complex equations to solve than the Maxwell-Stefan model, the omission of cross-coupling effects, the assumption of thermodynamic ideality and neglecting the dependence of Fickian diffusivity on concentration bring about significant error in describing multicomponent transport [31, 46, 55]. The Maxwell-Stefan model does not possess the shortcomings of this classical approach to the solution-diffusion model, which is why the solution-diffusion form of the Maxwell-Stefan model is preferred to describe multicomponent liquid separations via polymer membranes [82, 83, 84]. However, such predictive studies with liquid mixtures containing three or more species in »1% concentrations are scarce.

Ribeiro et al. first described the Maxwell-Stefan transport equations with volume fraction terms in conjunction with the Flory-Huggins sorption model to predict the permeation of binary mixtures of CO2/C2H6 across crosslinked polyethylene oxide membranes [85]. Krishna further developed explicit analytic expressions such that these fluxes could be described in two-dimensional matrix notation [39]. He used this notation to predict fluxes for CO2/C2H6 in Ribeiro et al. experiments and water/alcohol pervaporation via cellulose acetate and polyimide membranes. While this approach showed potential for experimental

simplicity as mutual diffusion parameters did not have to be experimentally measured, it requires further validation for complex, nonpolar feeds such as those found in crude oil. Here we use Krishna's approach as a starting point to predict the separation of liquid hydrocarbon mixtures containing up to nine similarly-sized molecules with two microporous, glassy polymer materials: PIM-1 and SBAD-1 (see Figure 3.1) [86, 87].

We demonstrate that complex mixture separations can be adequately predicted (i.e., within an average of 10% of the experimental values of permeate compositions and an average of 35% for permeate fluxes) via a Maxwell-Stefan framework with single component molecule-polymer sorption and diffusion parameters as the only experimental input requirements. These values were obtained from a combination of unary gravimetric vapor sorption, unary liquid swelling experiments, and unary liquid permeation. A new sorption model that additively combines Langmuir-type and Flory-Huggins-type sorption contributions is proposed to fit the unique sorption in glassy polymers and serves as a generalizable model, capable of extension to low- and high-swelling polymers. Comparisons to other thermodynamic models such as classical FH and DMS) are made based on unary, binary, and ternary sorption experiments as well as unary and mixture membrane permeation experiments. Vignes mixing rules are investigated as empirical correlations of binary diffusion interactions amongst hydrocarbons. Finally, free volume theory and an alternative average diffusivity concept are investigated to capture the influence of polymer dilation and plasticization on guest diffusivities.

3.2.2 Theory and Background

Maxwell-Stephan Transport of Mixed Feeds

This section is analogous to subsection 2.2.3. For the general set-up of the sorption-diffusion membrane transport mechanism, please refer to subsection 2.2.1.

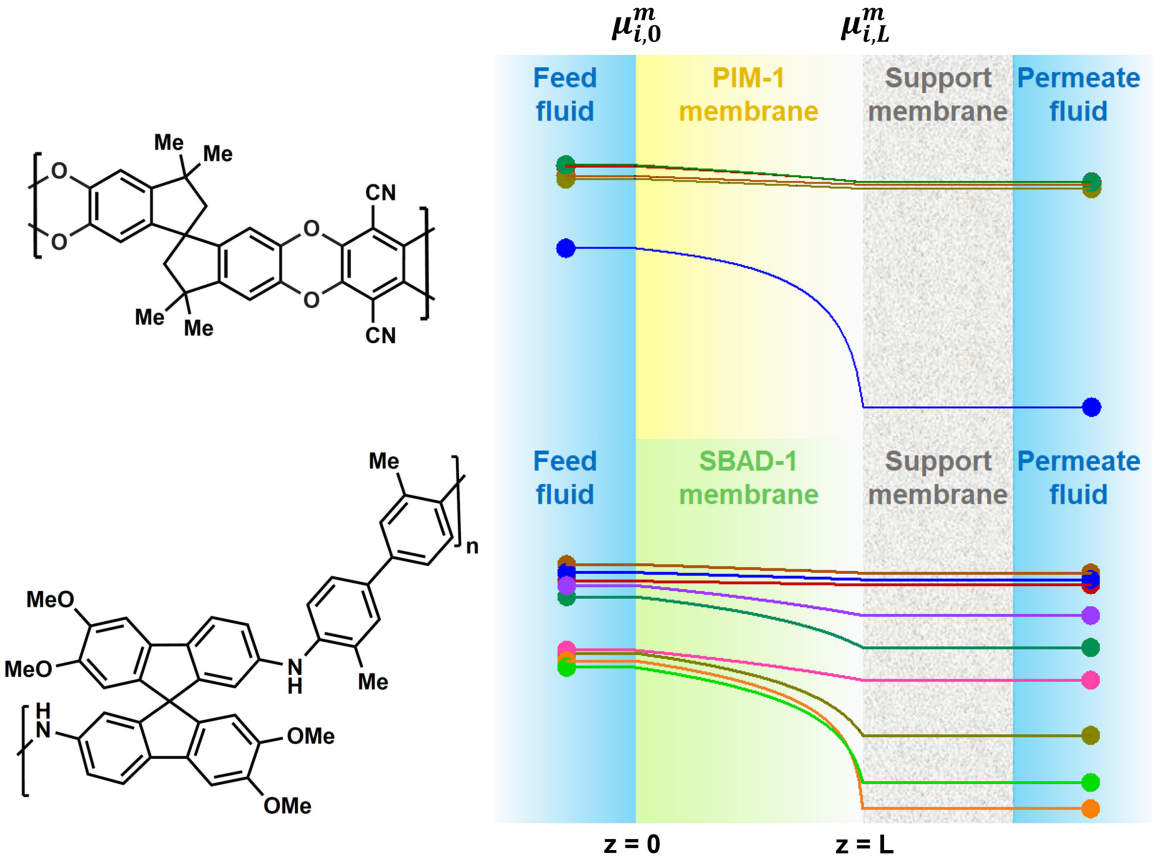


Figure 3.1: **Complex mixture transport via PIM-1 and SBAD-1 membranes.** Exemplar non-linear chemical potential profiles of individual molecules in complex mixtures are shown across the thickness of selective polymer membranes with corresponding chemical structures. It is assumed that the support layer does not hinder transport and chemical potential is unchanged throughout the support. Higher nonlinearity is observed for more dilute or low sorbing components, and as the number of components is increased (i.e., the mixture becomes more complex and each molecule becomes less concentrated in the mixture), Fick's law becomes insufficient to describe liquid hydrocarbon transport. Note that the membrane and support thicknesses are not to scale. [14]

Liquid Sorption in Polymers

This section describes the specific membrane fugacity models used in the phase equilibrium calculations in subsection 2.2.2. Recall that the generic notation for these models in the main text is $\mathbf{g}^{II}(\mathbf{f}^{II}, \boldsymbol{\phi}, T^{II}, P^{II}) = \mathbf{0}$. For brevity in this section, we drop the arguments T^{II} and P^{II} and suppress the superscript II on all variables. Lastly, unless specified, all variables are the same as defined in the main text nomenclature table.

Flory-Huggins Fugacity Model The multicomponent FH model gives the fugacity f_i of component i in the membrane phase at (T, P) as (assuming $\overline{V}_i = V_i^\circ(T, P)$) [26, 88]

$$\ln\left(\frac{f_i}{f_i^\circ(T,P)}\right) = \ln(\phi_i) + 1 - \sum_{j=1}^{n+1}\frac{V_i^\circ}{V_j^\circ}\phi_j + \left(\sum_{j=1}^{i-1}\chi_{ji}\phi_j\frac{V_i^\circ}{V_j^\circ} + \sum_{j=i+1}^{n+1}\chi_{ij}\phi_j\right)(1-\phi_i) - \sum_{\substack{j=1\\ j\neq i}}^{n}\sum_{\substack{k=j+1\\ k\neq i}}^{n+1}\chi_{jk}\frac{V_i^\circ}{V_j^\circ}\phi_j\phi_k, \quad (3.1)$$

where $\phi_{n+1} = \phi_m = 1 - \sum_{i=1}^{n}\phi_i$ is the membrane volume fraction and χ_{ii} = 0. Flory interaction parameters are commonly understood to be composition-dependent. Yang and Lue have explicitly shown that sorption is predicted more accurately when χ_{im} and χ_{ij} are considered to be functions of concentration, and are asymmetric such that χ_{ij} is not equal to χ_{ji} [89]. They show that hyperbolic and polynomial functions need to be fit to obtain $\chi_{im} - \phi_i$ and $\chi_{ij} - \phi_i$ relationships. However, these involve extensive mixture sorption experiments to verify, and are difficult to apply in a Maxwell-Stefan framework with n components. For simplicity, we will assume that χ_{im} is constant across different loading conditions, and χ_{im} is equal to the observed value at unit activity (when the polymer is in contact with pure liquid, see paragraph 3.2.3). Additionally, $\chi_{ij} = \chi_{ji}$ is the Flory-Huggins parameter for component pair ij, which can be approximated for $i \neq j \neq m$ using Hansen solubility parameters as [90]

$$\chi_{ij} = \frac{\left(V_i^\circ V_j^\circ\right)^{0.5}}{RT}\left[(\delta_{D,i} - \delta_{D,j})^2 + 0.25(\delta_{P,i} - \delta_{P,j})^2 + 0.25(\delta_{H,i} - \delta_{H,j})^2\right]. \tag{3.2}$$

Detailed multicomponent sorption experiments and analyses are required to further verify this simplified approach, and will be the focus of future work. Table A.3 shows the Hansen solvent parameters used in this study [91]. The more alike two solvents are, the lower the difference between the parameters will be; and the lower the binary chi parameters will be, resulting in greater binary sorption coupling.

Classical Dual Mode Sorption Model The two most common multicomponent sorption models for glassy polymers are the non-equalibrium lattice fluid (NELF) and the Dual-Mode Soroptin [92, 93]. In subsubsection 1.3.2, the NELF model is shown as infeasible for the purposes of predictive multicomponent sorption predictions based solely on pure component experiments. Additionally, there were a number of parameters to fit even in the single component case. Therefore, the classical DMS is described herein for uptake in glassy polymer membranes as a sum of two contributions: a dissolved phase following Henry's law, and an adsorbed phase following a Langmuir isotherm. The membrane concentration can therefore be written as

$$c_i^p = c_i^{p,\text{Henry}} + c_i^{p,\text{Langmuir}}, \tag{3.3}$$

where the superscript p denotes that the concentration is based on the skeletal polymer volume. Substituting in expressions for the Henry's law and Langmuir adsorption terms gives:

$$c_i^p = k_i' f_i + \frac{C_i^{H'} b_i f_i}{1 + \sum_{k=1}^{n} b_k f_k}, \tag{3.4}$$

where k_i' and $C_i^{H'}$ are based on moles of solvent per skeletal polymer volume.

Recall that the MS model in subsection 2.2.3 describes the membrane phase compo-

sition in terms of volume fractions ϕ_i based on total system volume (defined as skeletal polymer and sorbed molecule volume). To cast (Equation 3.4) in terms of the same variables, first assume that the partial molar volume of component i is the same in both the dissolved and adsorbed phases and $\overline{V}_i = V_i^\circ(T,P)$. Then, c_i^p can be related ϕ_i by

$$\phi_i = \phi_m V_i^\circ(T,P) c_i^p. \tag{3.5}$$

The units above are

$$\phi_i \left[\frac{\text{m}^3 \text{ component } i}{\text{m}^3 \text{ total}}\right] = \phi_m \left[\frac{\text{m}^3 \text{ polymer}}{\text{m}^3 \text{ total}}\right] V_i^\circ \left[\frac{\text{m}^3 \text{ component } i}{\text{mol component } i}\right] c_i^p \left[\frac{\text{mol component } i}{\text{m}^3 \text{ polymer}}\right].$$

Using this relation and letting $k_i = k_i' V_i^\circ(T,P)$ and $C_i^H = C_i^{H'} V_i^\circ(T,P)$, (Equation 3.4) becomes

$$\phi_i = \phi_m \left(k_i f_i + \frac{C_i^H b_i f_i}{1 + \sum_{k=1}^{n} b_k f_k} \right). \tag{3.6}$$

Consequently, (Equation 3.6) is the final form of the DMS model used in this work. This approach has been highly successful for describing gas sorption into glassy polymers, but fails to capture many features of solvent sorption isotherms.

Combined Flory-Huggins-Langmuir Model Before going into details of the derivation, it is beneficial to provide some context on how we derived this relationship theoretically and why the proposed mechanisms are valid for swollen glassy polymer membrane layers. Figure 3.2a shows the proposed mechanism where initially there is a hole-filling glassy soprtion mechanism followed by a transition to lattice theory-based dissolution mechanism. In fact, the concept of additively combining Langmuir hole-filling with lattice theory-based dissolution has been theorized much earlier by Barrer et al., but with a formulation that resembles DMS more closely than the formulation proposed in here [94]. In developing the proposed FH-LM sorption model, we conceptualize the dry polymer system as having

polymer and void regions, while at high activities we imagine the system to be made up of polymer and sorbed guest molecule regions. We assume that sorbates take up the entire free volume space present in the polymer above a certain activity corresponding to the transition of the polymer from the Langmuir-dominant regime to the Flory-Huggins-dominant regime (which often coincides with a glassy to rubbery transition). While the proposed FH-LM sorption model is similar to DMS in that both models describe a combination of a pore-filling and dissolution mechanisms, the convexity of sorption at higher activities for certain hydrocarbon molecules is not accurately captured by the Henry's sorption component of DMS. The competitive sorption of multiple penetrants into free volume elements in a glass can be described by the multi-component Langmuir model as seen in the right hand side term of (Equation 3.4).

Inspired by this dual mode approach, in this work we propose that the sorption of hydrocarbon vapors and liquids in glassy polymers can be more accurately described over the full range of solvent activities by a combination of Langmuir micropore-filling and Flory-Huggins swelling type sorption contributions. In the FH-LM model, the Henry's Law sorption term in DMS is replaced with a Flory-Huggins type sorption term [14]:

$$c_i = c_i^{\mathrm{Flory-Huggins}} + c_i^{\mathrm{Langmuir}}, \tag{3.7}$$

where the concentrations are normalized by the total system volume (defined as skeletal polymer and sorped molecule volume). Assuming again that the partial molar volumes are equal in each phase and equal to the pure component molar volume, the model can be rewritten in terms of volume fractions as

$$\phi_i = \phi_i^{\mathrm{Flory-Huggins}} + \phi_i^{\mathrm{Langmuir}}. \tag{3.8}$$

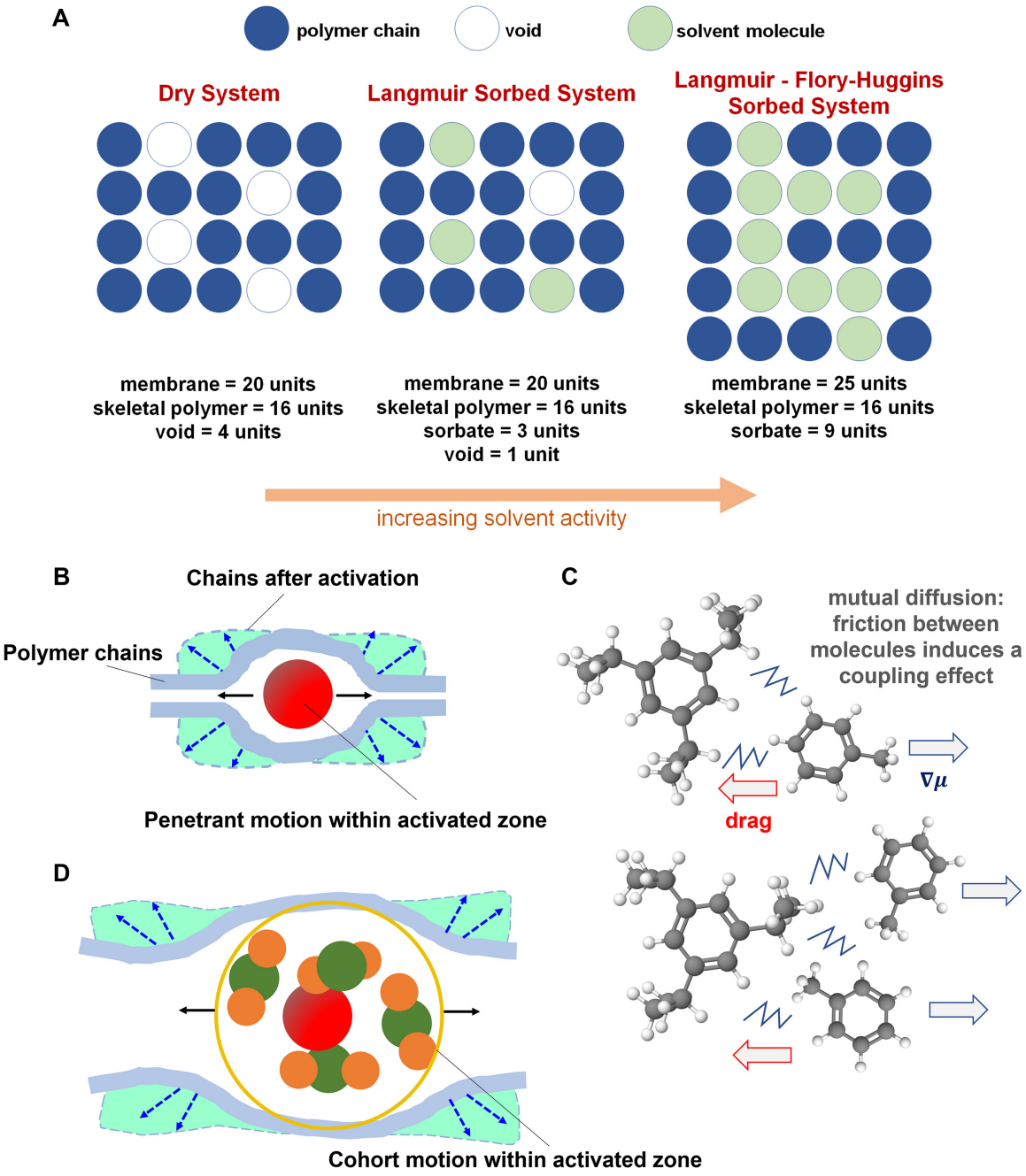


Figure 3.2: **Sorption Regimes and diffusive modes of transport in polymer membranes.** A. Dependence of sorption regime and membrane volume on solvent activity. It is important to note that the two guest populations are in equilibrium but Langmuir-style sorption will dominate at low solvent activities and Flory-Huggins-style sorption will dominate at high solvent activities. B. Conventional diffusion mechanism where a molecule makes diffusive jumps through free space in a polymer network. C. Maxwell-Stefan interpretation of mixture diffusion where frictional forces between molecules cause diffusion coupling such that faster molecules are slowed and slower molecules are sped up, leading to a loss in diffusion selectivity. D. Cohort motion mode of transport where molecules diffuse collectively as a unit and no diffusion selectivity is obtained. [14]

Using Equations (Equation 3.4)–(Equation 3.5) for the Langmuir term gives

$$\phi_i = \phi_i^{\text{FH}} + \phi_m \frac{C_i^H b_i f_i}{1 + \sum_{k=1}^{n} b_k f_k}. \tag{3.9}$$

In this Equation, ϕ_i^{FH} is specified from $\mathbf{f}$ as the solution of (Equation 3.17). Thus, Equations (Equation 3.17) and (Equation 3.9) constitute the final model.

Accessible Free Volume Theory

Penetrant-polymer Maxwell-Stefan diffusivities can be calculated via experimentally measured pure molecule permeation, pure molecule sorption, and (Equation 2.23). However, significantly different polymer swelling in the presence of a range of solvents could result in a wide range of mixture-accessible free volumes; these depend on the various possible upstream and downstream concentration ratios for a given set of molecules. This complicates simple attempts to estimate separation performance of a multicomponent mixture because the diffusion of each molecule through the polymer matrix could also vary with the accessible free volume in the system. Several versions of the free volume theory have been previously used in gas transport studies to correlate penetrant diffusivity with the accessible free volume of the dry polymer. Here, we have used one such model to correlate the same [95]

$$Đ_{im}^V = A_i \text{exp}\left(-\frac{B_i}{v_f}\right), \tag{3.10}$$

where A_i is a molecule-polymer system dependent constant, B_i is a molecule-dependent constant and is related with the molecule size, and v_F is the polymer free volume. Assuming the diffusivities of liquid species are also similarly dependent on accessible volume due to membrane swelling, we can derive the following correlation for each molecule:

$$\frac{Đ_{im,\text{polymer state } I}^V}{Đ_{im,\text{polymer state } II}^V} = \text{exp}\left[B\left(\frac{1}{v_{F,I}} - \frac{1}{v_{F,II}}\right)\right], \tag{3.11}$$

Thus, solvent-polymer Maxwell-Stefan diffusivities that are calculated from permeation experiments with known polymer states (i.e., degree of swelling) can be used in conjunction with (Equation 3.11) to estimate diffusivities in mixtures where the degree of polymer swelling is dependent on multicomponent sorption and is different from the swelling induced by the pure solvent. Here, we take $v_{F,I}$ as the unit activity volume fraction of component i and $v_{F,II}$ to be the mixture accessible volume (i.e., $v_{F,II} = \sum_{j=1}^{n} \phi_j = 1 - \phi_m$).

Cohort-style Average Diffusivity

Going a step further in relating diffusivities with polymer state upon sorption, Damle and Koros observed a loss of diffusion selectivity in glassy polymers that strongly dilate and plasticize in the presence of condensable adsorbates [96]. On the feed side, the mixture undergoes equilibrium partitioning into the membrane, which is swollen to the point that it has negligible microvoids, such that sorption is most appropriately described by Flory-Huggins lattice interactions. Inside the membrane, small collections of molecules act as a unit since the membrane is sufficiently swollen or plasticized, such that they collectively diffuse together. While the Maxwell-Stefan framework naturally takes the thermodynamic and cross-diffusional coupling into account, we noticed that setting all the penetrant-membrane diffusivities to some volume corrected average value gave the best results compared to any other approach we considered (subsection 3.2.4, vide infra). Since $Đ_{im}^{V}$ is proportional to the inverse drag coefficient, this approach can be considered as setting all the friction forces experienced by the permeants to be almost equivalent. When tracking the motion of an individual molecule, it would have a velocity or a displacement that is equivalent to all other molecules as they are moving in this unit. The mixture then partitions into the adjacent permeate phase according to the FH sorption model again. This is different from the free volume vision of a single molecule "hop" from sorption site to sorption site, which is the classic picture for gases and polymers that have not plasticized or dilated. Figure 3.2b–d provides a visual representation of these mechanisms. The cohort-style diffusivity can

be calculated a using a Vignes-style volume-corrected interpolation formula of the pure component Maxwell-Stefan molecule-polymer diffusivities:

$$Đ_{im}^{V} = (V_i)^{-1} \prod_{k=1}^{n} (Đ_{km}^{V})^{\frac{\phi_k}{\sum_{j=1}^{n} \phi_j}}, \tag{3.12}$$

Many averaging and weighting approaches have been tested, and this version gave the best results for our initial description of the complex mixture permeation framework.

Simulation and Parameter Fitting of Modeling Framework

This section outlines how parameters were fit and simulated for the modeling framework. While the experimental aspects are detailed, the main contribution from this dissertation deals with the computation aspect. The experimental work was all carried out by the co-first author, Dr. Ronita Mathias [14].

Sorption Models The Langmuir capacity can be estimated by visual observation of the curvature of the measured isotherms and is typically designated as the initial step uptake at a relative pressure of 0.3 for SBAD-1 (better fits were obtained by using the uptake at a relative pressure of 0.5 for 1-methylnaphthalene in SBAD-1) and 0.4 for PIM-1. For the FH-LM sorption model, the Flory-Huggins contribution is then calculated by subtracting C_i^H from the total sorption at unit activity, which effectively removes sorption in the microvoids such that these contributions are not double counted. For each solvent, χ_{im} can then be calculated based on this Flory-Huggins sorption contribution via (Equation 3.1). This single parameter (assumed to be concentration-independent, see paragraph 3.2.2) is sufficient to describe the Flory-Huggins contribution at different solvent activities. Finally, the Langmuir sorption parameter, b_i, is obtained by performing a least-squares fit of the Langmuir term in (Equation 3.9) to the total experimental sorption isotherm. In the case of the FH sorption model, where the dry polymer fractional free volume is assumed to be negligible, χ_{im} is calculated using sorption at unit activity in (Equation 3.1). For the DMS model,

the three model parameters are fit simultaneously via a least-squares fit of (Equation 3.6) to the experimental sorption isotherm. The error in calculated sorption model parameters was estimated by fitting to upper and lower bounds of the raw sorption isotherms measured from duplicate and triplicate data and is listed in the Supplemental Information along with the isotherm parameters.

Penetrant-polymer Diffusivities Experimental unary fluxes at a fixed transmembrane pressure were used in combination with each sorption model (FH, DMS, and FH-LM) to calculate three sets of $Đ_{im}^{V}$ corresponding to each sorption model. This was done via the Maxwell-Stefan equation reduced to a binary system of one penetrant in a polymer ((Equation 2.23), $n = 1$). These $Đ_{im}^{V}$ values were then utilized in the Maxwell-Stefan framework along with the respective sorption models to predict unary fluxes at higher transmembrane pressures for model validation. These diffusivities are then used as the inputs into the multicomponent transport models. As described in subsubsection 3.2.2, $Đ_{im}^{V}$ is expected to be a function of polymer free volume and we adjust for polymer swelling in multicomponent transport via (Equation 3.11). The free volume theory dictates that B is unique to each molecule and is dependent on penetrant size. Evaluating appropriate B values for each molecule requires at least two distinct experimentally-measured diffusivities for each penetrant at different accessible free volumes of the polymer. This increases the experimental effort required by the predictive framework and is also difficult to calculate for molecules that do not noticeably swell polymers. Therefore, we have explored several different possible B values and, in each sorption model case, a single value was consistently applied to all penetrants for simplicity. This method enables modification of a single component diffusivity (measured at a condition of $v_{F,I}$) to a case where the polymer is swollen to a much different degree due to the sorption of many different molecules resulting in the $v_{F,II}$ polymer condition.

Multi-component Transport Simulation Framework The Maxwell-Stefan framework discussed above was implemented to obtain multi-component permeation predictions using the Dual-Mode sorption model, the Flory-Huggins sorption model, and the proposed Flory-Huggins + Langmuir sorption model. Additionally, several different transport models were tested, which affect how the $\mathbf{B}$ and $\mathbf{\Gamma}$ matrices are specified in the Maxwell-Stefan framework. These are described in the following scenarios:

a) Scenario 1 (Sc1): Fick's Law flux formulation as in (Equation 2.26)

b) Scenario 2 (Sc2): Maxwell-Stefan approach in (Equation 2.23) without diffusional cross-coupling (i.e., $\mathbf{B} \rightarrow \text{diag}(\mathbf{B})$)

c) Scenario 3 (Sc3): identical to Sc2 but with a Vignes correlation (Equation 2.19) to describe diffusion cross-coupling

d) Scenario 4 (Sc4): identical to Sc3 but with $Đ_{im}^{V}$ adjusted for polymer swelling via the free volume theory (Equation 3.11), where B=0.03 is assumed to be a constant value for all molecules for experimental simplicity.

e) Scenario 5 (Sc5): identical to Sc2 but with $Đ_{im}^{V}$ replaced via the average diffusivity concept (Equation 3.12).

The proposed framework makes the following assumption:

i. The partial molar volume of each component is equivalent to its molar volume at pure conditions, 298 K, 1 atm

ii. The proposed framework for modeling transport through the asymmetric membrane is a three-step sorption-diffusion mechanism

To simulate the transport, please refer back to subsection 2.2.4 for the problem statement and subsection 2.5.6 for the proposed numerical method. The only difference to highlight is that the activity coefficients for Phase I and III are calculated by AspenTech UNIQUAC (or PC-SAFT) methods.

3.2.3 Experimental

All experiments were carried out by Dr. Ronita Mathias. This section outlines the complete experimental details that accompany the previously outlined simulation framework [14].

Materials The synthesis of PIM-1 and SBAD-1 were performed according to literature procedures as detailed previously [87, 97]. All other chemicals were purchased from Sigma Aldrich, Acros Organics, Alfa Aesar, Oakwood Chemical, or TCI and used as received.

Helium Pycnometry Dried SBAD-1 powder samples were analyzed via helium gas at 22 °C (AccuPyc II 1340 FoamPyc V3.00, Micromeritics). An average skeleton density of 1.29 g/cm3 was calculated from 10 cycles.

Membrane Fabrication Thin film composite fabrication of SBAD-1 is detailed in previous work [87]. Briefly, a 0.8wt% chloroform solution of SBAD-1 was coated on crosslinked polyetherimide support (surface pore size of 9 nm) via a roll-to-roll process line at a speed of 5m/min and a drying temperature of 55 °C in an air-convection dryer. A 1 wt% chloroform solution of PIM-1 was blade-coated on crosslinked polyetherimide support (surface pore size of 25 nm) via a blade of 25.4-micron thickness. SBAD-1 and PIM-1 dense films were prepared by pouring chloroform solutions of the polymers (10-20 wt%) into leveled Teflon dishes in a glove bag saturated with chloroform vapor. Films stood in the saturated environment for 24 h and were then allowed to dry as the atmosphere was gradually depleted of solvent vapor. The films were then dried under vacuum (-29 mm Hg) at 110 °C overnight.

Vapor Sorption SBAD-1 and PIM-1 powder obtained directly from synthesis were used for vapor sorption experiments. The powder was dried under 29 mm Hg vacuum and 110 °C overnight before analysis and dried again in situ at 110 °C under flowing nitrogen for 200 minutes before sorption. The vapor sorption instrument (VTI SA+, TA Instruments)

utilized Wagner equation constants to determine the saturation vapor pressure of a liquid and the relative pressure $\frac{p_i}{p_{\text{sat}}}$ was controlled by mixing dry nitrogen gas and the headspace of a saturator containing hydrocarbon liquid. Measurements were performed at 25 °C and in duplicate or triplicate to determine experimental error.

Unit Activity (Liquid) Sorption The sorption at unit activity was measured by submerging weighed dense film fragments in liquid hydrocarbon at room temperature (22 °C) for at least 1 month. The resulting solvated films were weighed after wiping the surface dry with a Kimwipe. Each measurement was performed twice to improve accuracy. Measurements with sorption under 5 wt% were not included in the final data as reliable mass uptakes could not be obtained.

Liquid Permeation Thin film composite permeation was measured with a custom-built cross flow system pressurized by an HPLC pump (Azura P 4.1S, Knauer). A constant feed flow rate of 10 mL/min was employed, which satisfied the upper limit of a 1% stage cut for all mixtures. Aliquots from the permeate were taken at 24 h intervals until the permeance and permeate concentrations were stable (typically 3-5 days). The permeate compositions were determined using gas chromatography (Agilent 7890B) which, in combination with the total permeate flux measured, were used to determine individual molecule fluxes. Samples from each membrane sheet were tested in triplicate.

Scanning Electron Microscopy The thickness of thin polymer films in PIM-1 and SBAD-1 were measured via Field Emission Scanning Electron Microscopy (FE-SEM) (Hitachi SU8010). Membrane samples were cut with a sharp razor blade and placed on aluminum mounts using carbon tape. The samples were sputter coated with a gold/palladium alloy using a turbomolecular pumped coater (Quorum Q-150 T ES) under a deposition current of 10 mA for 30 seconds. SEM images were captured at a voltage of 3kV and a current of 10 μA (Figure B.1). The resulting SBAD-1 and PIM-1 film thicknesses were estimated to

be 300 and 1500 nm, respectively.

3.2.4 Results and Discussion

A key factor in the multicomponent transport framework is a sorption isotherm that accurately captures the uptake of multiple penetrants in a polymer system. Unary experimental hydrocarbon sorption isotherms and model fits for PIM-1 and SBAD-1 are shown in Figure 3.3. The higher experimental error observed for low sorbing molecules such as *iso*-cetane is expected as the data approaches the lower end of the instrument accuracy range. It should be noted that due to slow transport in SBAD-1, certain sorption data likely did not reach full equilibrium (Figure B.2). These include sorption of methylcyclohexane, 1-methylnaphthalene, tert-butylbenzene, 1,3,5-triisopropylbenzene and iso-cetane. Despite this, the data at each relative pressure that was collected within reasonable timeframes (up to 2 weeks of equilibration time per point) were used to fit the desired sorption parameters with the understanding that the model isotherms will likely be underpredicting the true uptakes. Beyond a relative pressure of 0.3, vapor uptakes of tert-butylbenzene were difficult to measure due to accumulation and condensation of the fluid within the instrument chamber and so, limited experimental data is available for this molecule. In general, FH-LM and DMS enable good predictions of sorption at different activities while the FH model with constant χ_{im} underpredicts sorption within the entire range of activities. Figure B.3 shows that χ_{im} varying with the volume fraction of the penetrant enables more accurate predictions of sorption via the FH model, but ultimately complicates multicomponent transport solutions. For certain highly sorptive molecules, such as o-xylene in PIM-1 and 1-methylnaphthalene in SBAD-1, the proposed FH-LM sorption model delivers better isotherm fits than DMS. Based on this observation, we suggest that multicomponent sorption of liquid hydrocarbons to be best captured by FH-LM.

It is essential for a model that describes the transport of complex liquid feeds to be capable of describing permeation of simple feeds - the simplest being a pure liquid. There-

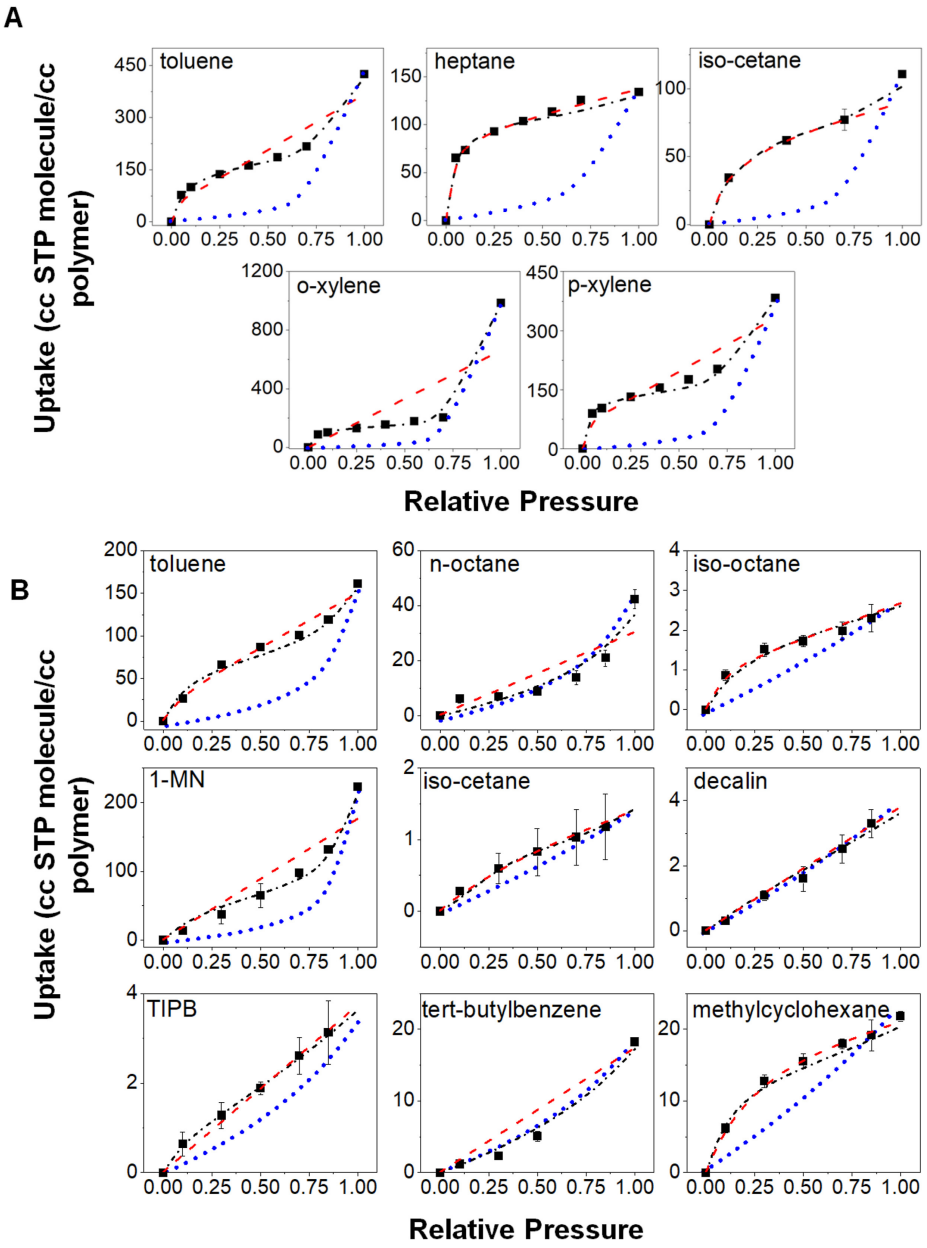


Figure 3.3: **Unary sorption in PIM-1 and SBAD-1.** Experimental hydrocarbon sorption isotherms (▪ ▪) and predictions for PIM-1 (A) and SBAD-1 (B) at 25°C assuming Dual-mode (- -), Flory-Huggins (· · ·), and Flory Huggins + Langmuir (-··-). X-axes indicate relative pressure of the molecule and y-axes represent molecule uptake (cc [STP] molecule/cc polymer). Data are shown as averages of at least two measurements with standard deviation error bars. Abbreviations are shown for 1-methylnaphthalene (1-MN) and 1,3,5-triisopropylbenzene (TIPB) [14].

Table 3.1: Multicomponent separations via SBAD-1 and PIM-1 performed at 22°C.

Separation 1 via PIM-1 at a transmembrane pressure of 30 bar			
Permeate Flux (L/m^2/hr)	6.33±3.96		
	Feed Composition	Permeate Composition	
Component	mole fraction	mole fraction	error %
toluene	0.257	0.267	0.37
heptane	0.216	0.210	0.95
p-xylene	0.205	0.212	0.47
o-xylene	0.264	0.269	0.74
iso-octane	0.058	0.042	4.8
Separation 2 via SBAD-1 at a transmembrane pressure of 40 bar			
Permeate Flux (L/m^2/hr)	0.88±0.52		
	Feed Composition	Permeate Composition	
Component	mole fraction	mole fraction	error %
toluene	0.171	0.201	1.5
methycyclohexane	0.281	0.253	0.79
1-methynaphthalene	0.020	0.028	3.6
decalin	0.107	0.110	0.91
n-octane	0.221	0.245	2.0
iso-octane	0.150	0.123	6.5
tert-butylbenzene	0.022	0.027	3.7
1,3,5-triisopropylbenzene	0.016	8.2×10^{-3}	12
iso-cetane	0.013	4.5×10^{-3}	22
Separation 3 via SBAD-1 at a transmembrane pressure of 30 bar			
Permeate Flux (L/m^2/hr)	0.40±0.12		
	Feed Composition	Permeate Composition	
Component	mole fraction	mole fraction	error %
toluene	0.284	0.318	1.9
iso-octane	0.388	0.422	1.4
iso-cetane	0.328	0.260	4.2

fore, we first apply the Maxwell-Stefan model (Equation 2.23) to unary liquid hydrocarbon permeation at a range of different transmembrane pressures as seen in Figure 3.4. The pure component diffusivities $Đ_{im}^{V}$ were estimated using the experimental unary permeation flux for each solvent-polymer pair at a transmembrane pressure of 20 bar (30 bar for iso-octane) (Figure 3.4 and Figure 3.5a). The calculated $Đ_{im}^{V}$ were then used to predict and compare with experimentally measured fluxes at higher transmembrane pressures of 30, 40, 50, and 60 bar. It should be noted that the high error in the experimental permeation for PIM-1 was

likely due to the greater variation in thickness of the thin films in the thin film composite membranes. To reduce the contribution of experimental error to the predictive framework, the same set of membranes were utilized in all permeation experiments, including complex mixture separations. Figure 3.4 shows that the predicted fluxes fit closely with experimentally measured values in the case of all three sorption models. An anomaly exists in the experimental values for 1,3,5-triisopropylbenzene (TIPB), where the fluxes were so low that an accurate measurement was only recorded for one sample (which was the highest) and error bars could not be calculated. In general, all sorption models enable unary flux predictions that are within error of experimental values. For aromatic molecules such as toluene, *p*-xylene and *o*-xylene in PIM-1 and tert-butylbenzene and 1-methylnaphthalene in SBAD-1, where a clear preference of the FH-LM model was observed in Figure 3.3, no such preference is seen in the prediction of their fluxes in Figure 3.4. It is observed that TIPB flux predictions are out of the range of experimental values. The good matches for all other hydrocarbon fluxes and the excellent TIPB sorption predictions suggest that the measured TIPB fluxes (which are very low) are on the order of the leak rate in the permeation cell resulting in unexpectedly higher values.

Figure 3.5a correlates $Đ_{im}^{V}$ with molecule liquid molar volumes and as expected, higher Maxwell-Stefan diffusivities are calculated for molecules with lower liquid volumes. There is an almost linear negative correlation of the data except for one outlier: 1-methylnapthalene. It is possible that due to the long timescales of 1-methylnaphthalene diffusion within SBAD-1 that the five-day unary permeation experiments were not at the same sorption state as assumed from Figure 3.3. Lower 1-methylnaphthalene uptake in the membrane would result in lower measured fluxes, which would then lead to lower calculated $Đ_{\text{1-MN,SBAD-1}}^{V}$. We conclude that for this molecule, further long-term sorption and permeation experiments may be needed to more accurately estimate $Đ_{\text{1-MN,SBAD-1}}^{V}$. Moreover, the diffusivities of toluene and iso-cetane in PIM-1 are higher than those in SBAD-1 which align with the expected higher fractional free volume in PIM-1 compared to SBAD-1.

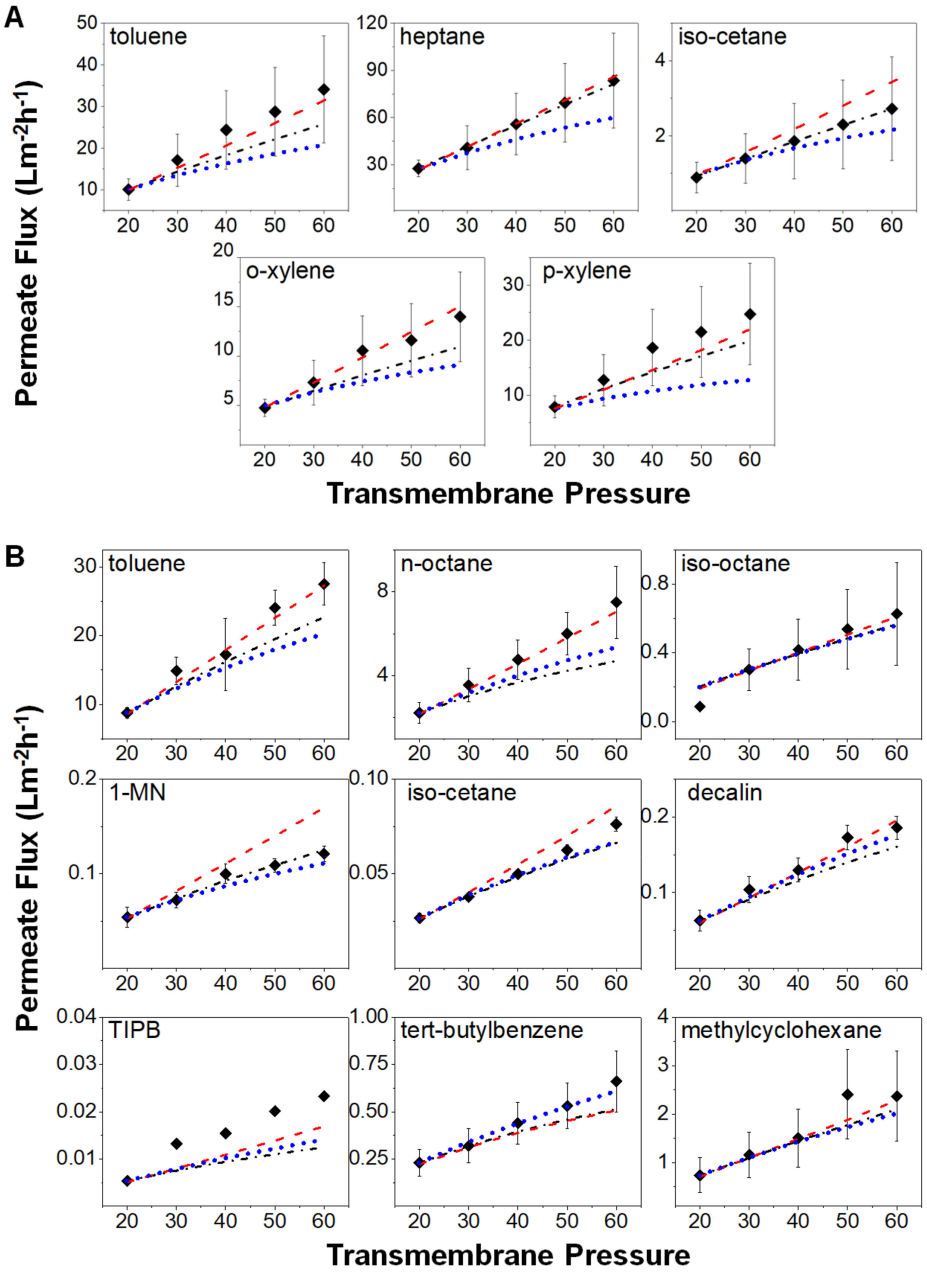


Figure 3.4: **Unary permeation in PIM-1 and SBAD-1.** Experimental liquid hydrocarbon unary flux (⧫) and predicted flux for thin-film composites at 22°C with an estimated film thickness of 1500 nm for PIM-1 (A) and 300 nm for SBAD-1 (B) assuming Dual-mode (- -), Flory-Huggins (· · ·), and Flory Huggins + Langmuir (-··-) sorption models. X-axes indicate transmembrane pressure (bar) and y-axes represent flux ($Lm^{-2}h^{-1}$). Data are shown as averages of three measurements on separate films with standard deviation error (with the exception of TIPB for which only one sample had measurable permeate flux). Abbreviations are shown for 1-methylnaphthalene (1-MN) and 1,3,5-triisopropylbenzene (TIPB)[14].

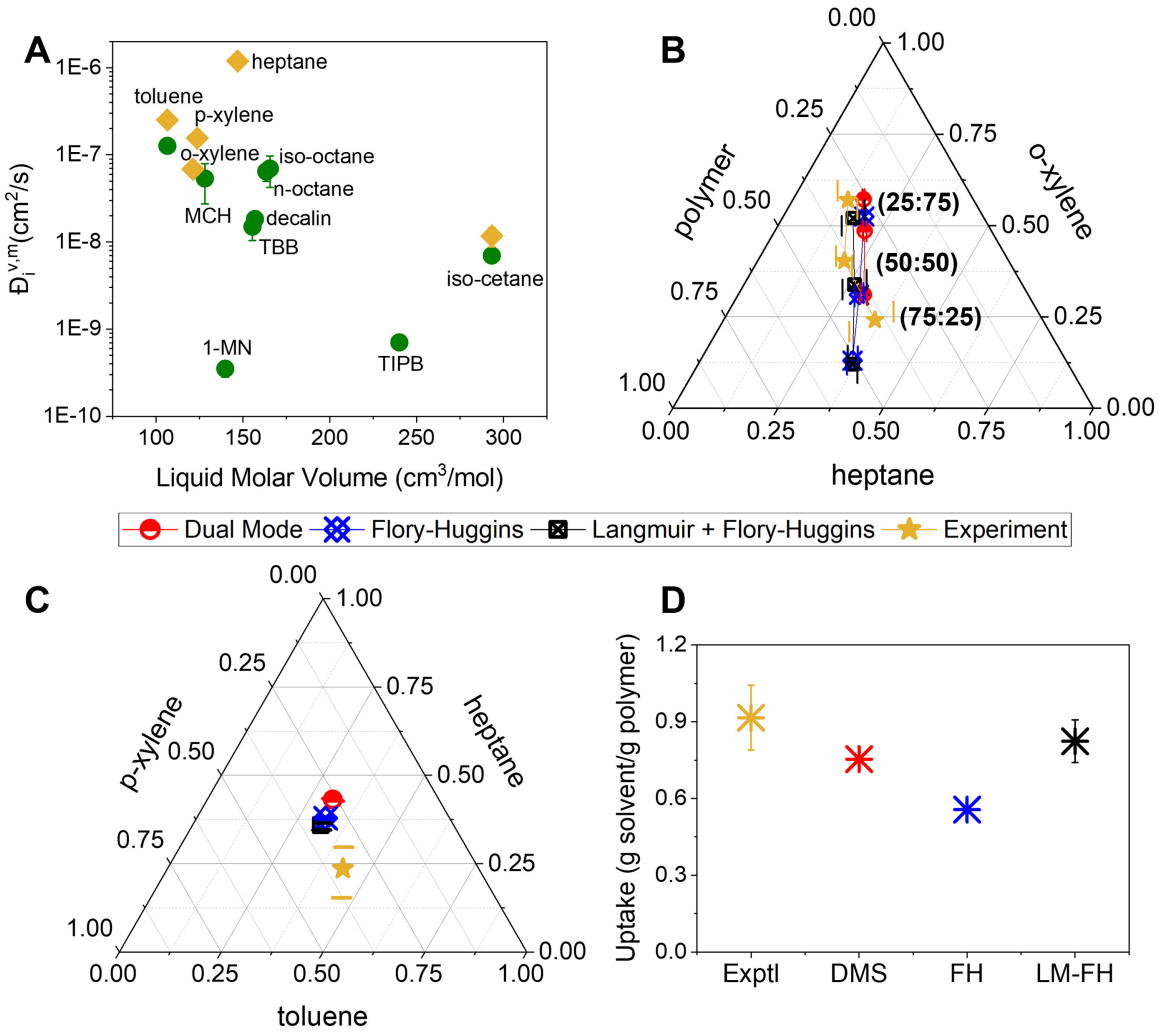


Figure 3.5: **Unary diffusion and multicomponent liquid hydrocarbon sorption.** A. Volume-based Maxwell Stefan diffusivities, $Đ_{im}^{V}$ (cm^2/s), in SBAD-1 (●) and PIM-1 (♦) at 22°C calculated using the Flory Huggins + Langmuir sorption parameters and unary permeate fluxes at 20 bar. Abbreviations are shown for methylcyclohexane (MCH), 1-methylnaphthalene (1-MN), tert-butylbenzene (TBB) and 1,3,5-triisopropylbenzene (TIPB). B-D. Multicomponent experimental sorption in PIM-1 compared with sorption predictions using single component parameter fits and estimates for competitive sorption effects for Dual-Mode, Flory-Huggins, and Flory Huggins + Langmuir models. Experimental measurements are from submerging dense films of PIM-1 in liquid mixtures at 22°C and atmospheric pressure. Molecule activities were taken into account when predicting multicomponent sorption here. B. Binary sorption indicated as volume fractions of swollen polymer system. Values in parentheses indicate initial mole fractions of the surrounding bulk fluid (heptane:*o*-xylene). C. Ternary sorption indicated as volume fractions of sorbed liquid in PIM-1 dense films in bulk fluid initially composed of toluene, heptane and *p*-xylene in mole fractions of 0.35, 0.36 and 0.29 respectively, and D. Total solvent uptake (g solvent / g polymer) in the swollen polymer from the ternary sorption condition in C [14].

Beyond single component sorption, it is important that the new sorption models adequately describe mixture sorption phenomena. While a complete analysis of multicomponent sorption is beyond the scope of this paper due to the large amount of experimental data that is required, we have compared a few binary liquid sorption measurements of heptane and *o*-xylene (Figure 3.5b, Figure B.4) and one ternary mixture sorption of toluene, *p*-xylene and *iso*-cetane Figure 3.5c–d in PIM-1 with predictions from the three sorption models. For the binary uptakes, we observe equally good predictions by the FH and FH-LM sorption models. In the case of ternary sorption, the total uptake (g mixture/g polymer) seems to match predictions in the order of FH-LM > DMS > FH with the latter being out of the range of experimental error. All sorption models predict the composition of the sorbed species in the ternary system to a similar degree of accuracy. We conclude, based on this limited dataset, that the FH-LM mixture sorption model – parameterized with single component data – results in the closest agreement with the experimental uptakes out of the three sorption models.

Experimentally-measured permeate fluxes and compositions for three complex mixtures via PIM-1 and SBAD-1 membranes are detailed in Table 3.1. The three separations vary in complexity: a five-component separation via PIM-1, a nine-component separation via SBAD-1 and a three-component separation via SBAD-1. SBAD-1, being less susceptible to dilation than PIM-1, as observed in Figure 4, results in better separation of mixtures (significant decrease in concentration of large molecules such as iso-cetane and TIPB). It is of interest to include both polymers in the Maxwell-Stefan transport predictions because of their differing behavior in solvents, despite both being glassy and rigid in the dry state. The transport of these mixtures is predicted via the varying sorption and diffusion scenarios described in paragraph 3.2.2 and the resulting predicted partial fluxes are summarized in Figure B.5, Figure B.6, and Figure B.7. The experimental partial flux of a molecule is calculated as the product of its volume-based composition in the permeate and the total permeate volume flux. It is informative to define the success of a transport model by

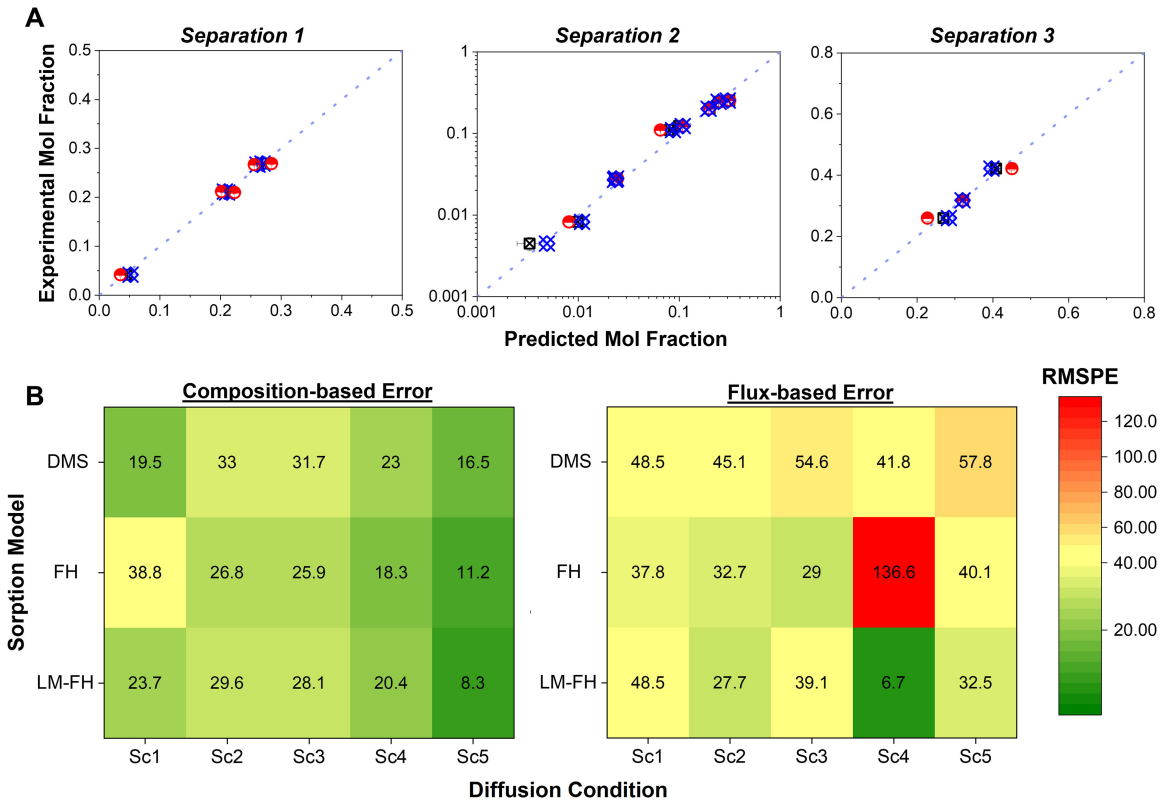


Figure 3.6: **Permeate flux and composition-based prediction of multicomponent separations in Table 1.** A. Comparison of predicted experimental permeate compositions with predicted values for Separations 1, 2, and 3 where the $Đ_{im}^{V}$ for all molecules are assumed to be equal (average diffusivity approach: Sc5). For each separation, Dual-mode (red), Flory-Huggins (blue) and Flory Huggins + Langmuir (2) sorption models are investigated. Dotted parity lines (x=y) are included as a guide for comparisons between predicted and experimental values. Error bars are included but are too small to be visible in some cases. B. Heatmaps showing composition based and total flux based root mean square percentage error (RMSPE) of each combination of sorption and diffusion assumptions. Y-axes vary sorption between Dual-mode, Flory-Huggins, and Flory Huggins + Langmuir models while x-axes vary diffusion conditions as: Sc1 = Fickian transport, Sc2 = no diffusion coupling, Sc3 = Vignes diffusion coupling, Sc4 = Vignes diffusion coupling + free volume theory, Sc5 = average diffusivity assumption [14].

how precise the predicted permeate compositions are and to a lesser extent, by how precise the predicted total permeate fluxes are. This precision can be calculated for any single experiment via RMSPE calculations where:

$$\text{RMSPE} = \left(\frac{\sum_{i=1}^{n} \left[\frac{|\text{Predicted Value}_i - \text{Experimental Value}_i|}{\text{Experimental Value}_i} \right]^2}{n} \right)^{\frac{1}{2}}. \tag{3.13}$$

The average RMSPE of the separations listed in Table 1 are reported for different combinations of the three sorption models and five diffusion scenarios and are summarized in Figure 3.6b.

The best overall agreement of permeate composition and total permeate flux with experimental data is obtained with the FH-LM sorption model and Sc4 where the free volume theory is used within the Maxwell-Stefan framework to adjust $Đ_{im}^{V}$ based on polymer swelling. When the permeate composition needs to be more accurate but the flux can have a larger band of error, the cohort-style average diffusivity approach (Sc5) excels with all sorption models, particularly FH-LM (Figure 3.6a). For thin film membranes, such as the ones used in this work, it is not uncommon to encounter membranes with 30-40% variability in thickness. Despite the dependence of permeate flux on membrane thickness, the permeate compositions are expected to be independent of thickness if a defect-free membrane is utilized. The low experimental error in permeate compositions in Figure 3.6a is reflective of defect-free membranes. Therefore, we place greater importance on the precision of permeate composition predictions than permeate flux predictions which are sensitive to variations in membrane thickness.

Interestingly, the simple Fick's law formulation (Sc1) generates better predictions of the permeate composition via the DMS and FH-LM model than the Maxwell-Stefan framework (Sc2) although the latter predicts the total flux slightly more accurately (Figure 7B). On the other hand, the FH sorption model behaves as expected with Fick's law (Sc1) performing worse than Maxwell-Stefan (Sc2) in both permeate composition and permeate flux

predictions. Vignes cross-coupling only provides minimal improvements in the accuracy of permeate composition predictions and, in fact, generates worse permeate flux predictions. This indicates that the Vignes equation does not sufficiently describe the various intermolecular coupling of diffusive transport of molecules in a liquid mixture through a glassy polymer. Even when the cross-diffusivities $Đ_{im}^{V}$ are deliberately fit to match the experimental compositions (Figure B.8), there is still an undesirably large discrepancy in the flux predictions suggesting that manipulation of cross-diffusivities is insufficient to capture the transport in these glassy polymers.

As discussed in subsubsection 3.2.2, the $Đ_{im}^{V}$ can be correlated with the degree of polymer swelling. Said more plainly, the pure component diffusivity is strongly dependent on the state of dilation of the polymer, and the polymer will exist at different states of dilation depending on the solvent mixture it is in contact with. Our simplified free volume theory expression (Equation 3.11) enables a first pass estimate of this complex process (estimates of $Đ_{im}^{V}$ for a range of swollen FFVs is shown in Figure B.9 and are maintained below self-diffusivities [98]. A clear issue associated with assuming a constant B parameter is that the polymer's diffusion selectivity for specific molecular pairs is maintained at various states of dilation, whereas it is almost certain that the selectivity will be reduced at higher levels of dilation. For this reason, individual B values may be estimated by fitting the equation to self-diffusivities, although we have not pursued this approach in an effort to simplify the framework such that it can be more easily generalized and applied. The B value of 0.03 was chosen as the optimum value in a range of arbitrary values investigated (Figure B.10 and Figure B.11). In Sc4, the free volume theory adjustment of diffusivity when applied with an optimized B value, offers noticeable improvement in slow molecule flux predictions for Separations 2 and 3 via SBAD-1 (Figure B.6 and Figure B.7, Sc4 vs Sc1-Sc3) but not for Separation 1 via PIM-1 (Figure B.5, Sc4 versus Sc1-Sc3).

As discussed earlier, when considering the composition-based error, the average diffusivity approach (Sc5) results in the lowest RMSPE, with small differences across the

three sorption models. It appears that without sufficient adjustment of $Đ_{im}^{V}$ with polymer dilation and plasticization as described in the average diffusivity concept, the compositions of the slower molecules (decalin, 1-methylnaphthalene, 1,3,5-triisopropylbenzene and *iso*-cetane) are underpredicted in Separation 2 via SBAD-1 (Figure B.6, Sc5 versus Sc1-Sc4). The next lowest error is when both Vignes cross-coupling and free volume theory are employed (Sc4), where an RMSPE of <30% is maintained for all sorption models. In Sc5, all molecule diffusivities are equivalent, caused by strong coupling of molecules such that they cannot diffuse independently, resulting in no diffusion selectivity. Note that this level of coupling is not possible Vignes cross-coupling approaches. On the other hand, in Sc4, a constant diffusion selectivity is maintained for each pair of molecules, but absolute diffusivities change with polymer sorption, caused by swelling of the polymer and an increase in accessible volume. The differentiating transport mechanisms thus are i) guest cohort motion across the membrane where molecules move collectively in small units versus ii) individual molecules hopping from one open site to another, which move faster when more sorption sites become available upon membrane dilation.

Based on unary, binary and ternary sorption experiments, if we narrow down the results to just the FH-LM model, we observe a consistent decrease in composition-based RMSPE from Sc2 to Sc5. One could deduce that this aligns with increasing solvent-solvent and solvent-polymer diffusive coupling within a Maxwell-Stefan framework in the ascending order of scenarios. The error in total permeate fluxes does not follow a similar trend and cannot be correlated as easily but it remains low enough (<40%), such that preference is given to the predictions of composition based RMSPE. This point is further supported by the difficulty in precisely measuring thin film thicknesses that are on the order of hundreds of nanometers and the large relative variabilities in thicknesses across several samples during membrane production, that lead to low confidence in measured thicknesses and therefore, expected permeance (permeate flux that is normalized by the thickness of a given sample). We may therefore conclude that both the free volume theory approach and the

average diffusivity approach show potential in making fast predictions of permeate compositions and fluxes of multicomponent liquid mixtures via glassy spirocyclic polymers like PIM-1 and SBAD-1.

3.2.5 Conclusions

We explored three sorption models – FH, FH-LM, and DMS – as distinct conceptual approaches to the problem of membrane-based separation of complex mixtures. When considered in combination with the unary and multicomponent sorption experiments, FH-LM is the most robust isotherm model that could be implemented with numerical and experimental ease. A variety of transport scenarios were also compared in this study. We find that the Vignes correlation did not offer significant improvements in the predictability of multicomponent transport and while the Vignes equation was included here as a stand-in for cross-coupling, such types of empirical correlations that do not have a well-defined physical significance hinder us from improving diffusional coupling correlations at this point. On the other hand, with a reasonable estimate of individual B values used in the free volume theory, varying degrees of improvements in predictions were obtained. The free volume theory improves the predictions of slow molecule fluxes but introduces more parameters that need to be fit or chosen arbitrarily, which would require further experiments or molecular dynamics simulations to evaluate fully. Critically, the average diffusivity approach based on an average guest diffusivity effectively had the best success in predicting permeate compositions in all 3 multicomponent separations and does not require the fitting of additional parameters. This is an important observation, as it has the potential to dramatically simplify deployment of this multicomponent transport framework in the case of more complex mixtures.

When drawing conclusions from the presented data, it is important to keep a few things in mind. First, the slow kinetics of bulky molecules such as 1-MN shown in Figure B.2 could very well mean that experimental permeate compositions and fluxes were measured

at what appears to be a pseudo-equilibrium state in the unary permeation testing period due to slow solvent-induced relaxations of the polymer structure. Further experiments will be needed to confirm this and accurately calculate the diffusivity of 1-MN and other such molecules as a function of penetrant activity and polymer free volume simultaneously. Second, investigating the predictability of a small dataset of experiments is just a start – an extensive database of experimental data for a variety of solvent and polymer systems is required to validate the generalizability, accuracy, and ease of computing a multicomponent transport model. Furthermore, there is a need to standardize the success of transport models. A model with a low RMSPE for a random set of experiments (preferably >100), could be deemed a suitable framework.

Overall, we demonstrated that complex multicomponent transport in polymers can be quantitatively predicted with some degree of accuracy, using only pure molecule-polymer sorption and diffusion parameters. While this is a complex problem, the average diffusivity simplification provides a potentially simplistic approach which could pave a practical path to multi-component diffusion modeling of complex mixtures in PIM-1 and SBAD-1. For more fundamentally accurate predictions, the diffusional and thermodynamic cross coupling of molecule pairs could be better defined with more complex approaches. Although our aim is to eventually describe the transport of real crude feeds containing thousands of molecules, we focused on less complex feeds here due to the extensive experimental effort required to validate the model. Future computational efforts that can also predict pure solvent sorption and diffusion that replace experimental measurements will be needed before more complex feeds can be investigated.

3.2.6 Funding

This work was supported by ExxonMobil Research and Engineering. K.A.T. acknowledges support from the Department of Education Graduate Assistance in Areas of National Need (GAANN) program at Georgia Institute of Technology (award no. P200A180075). This

material is also based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Advanced Manufacturing Office (award no. DE-EE0007888).

3.3 Extension of the Transport Model for Large Component Simulations

This section outlines the work done in order to enable simulations of crude oil mixture permeation through glassy polymers. Specifically, the work presented here was a crucial piece of a co-authored publication titled "Data-driven Predictions of Complex Organic Mixture Permeation in Polymer Membranes" [69]. This paper presented a framework using the Polymer Genome (*https://www.polymergenome.org/*) such that machine learned polymer parameters could be used to predict permeation of complex mixture feeds for polymers that were never before synthesised based on a training set of known polymer properties. In order to scale the simulation to make it feasible to go from 9 components to 353 component crude oil mixtures, a serious revamping of the implemented sorption and transport models had to be undertaken. The need for such a change was that the simulation was estimated to take almost a year to converge with the initial implementation of the model. This estimation is based on taking the time for evaluating a single entry of the Jacobian matrix of the outer solver as shown in Figure 2.3 (which for the shooting-algorithm specified in subsection 2.5.6 was a two forward integration of DAEs that took about 4 hours each), multiplying that by the number of evaluations required to get a complete Jacobian matrix (353), and again multiplying that by the number of average iterations required to converge a solution (8). After some clever rearrangements of the underlying models using matrix-vector algebra, the convergence time dropped to around 16 hours for the same simulation.

The major change implemented was taking the Flory-Huggins model and transforming it from something that was written in terms of summations (which in code was represented in terms of three nested *FOR* loops), to something that was compacted into a single line of matrix-vector multiplications and additions. To see exactly how this process works, please

first refer to (Equation 3.1) for original form. This is the main sorption model that was used in Lee et al. [69]. In order to understand the full reduction in model complexity, the speed up did not come just from vectorization of the model. While solely vectorization would only benefit in interpreted programming languages such as *MATLAB* or Python, compiled programming languages such as C++ would not see much of a speed-up. That is why it is imperative to understand that while vectorization was part of the speed-up, the Flory-Huggins fugacity model was first rewritten to allow for a more compact matrix-vector implementation. First, (Equation 3.1) was simplified by making some clever substitutions for the membrane phase volume fractions, and also combining summation terms to yield a more compact equation that takes the form

$$\ln\left(\frac{f_i}{f_i^\circ(T,P)}\right) = \ln(\phi_i) + 1 - \sum_{j=1}^{n+1}\frac{V_i^\circ}{V_j^\circ}\phi_j + \left(\sum_{j=1}^{i-1}\chi_{ji}\phi_j\frac{V_i^\circ}{V_j^\circ} + \sum_{j=i+1}^{n+1}\chi_{ij}\phi_j\right) - \sum_{j=1}^{n}\sum_{k=j+1}^{n+1}\chi_{jk}\frac{V_i^\circ}{V_j^\circ}\phi_j\phi_k. \quad (3.14)$$

Here, (Equation 3.14) has less multiplications, and the double summation is simplified greatly to not have multiple $i \neq j$ terms compared to (Equation 3.1). After a lengthy but fairly straightforward set of rearrangements, (Equation 3.14) can be written compactly in matrix-vector form as

$$\ln\left(\frac{\mathbf{f}}{\mathbf{f}^\circ}\right) = \ln\left(\boldsymbol{\phi}_{1:n}\right) + \mathbf{C}\boldsymbol{\phi} - \mathbf{V}_{n+1}^\circ(\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Q}\boldsymbol{\phi}), \quad (3.15)$$

where the division and natural logs are meant componentwise and $\mathbf{C} \in \mathbb{R}^{n\times(n+1)}$, $\mathbf{V}^\circ \in \mathbb{R}^{n+1}$, and $\mathbf{Q} \in \mathbb{R}^{(n+1)\times(n+1)}$ are defined by

$$c_{ij} = \begin{cases} \frac{V_i^\circ}{V_j^\circ}(\chi_{ij} - 1) & \text{if } j \leq i \\ \chi_{ij} - \frac{V_i^\circ}{V_j^\circ} & \text{otherwise} \end{cases},$$

$$\mathbf{V}^{\circ}_{n+1} = (V^{\circ}_1, V^{\circ}_2, \ldots, V^{\circ}_{n+1}),$$

$$\mathbf{Q} = \mathbf{U}^{\mathrm{T}} \left[\mathrm{diag}(\mathbf{V}^{\circ})\right]^{-1},$$

$$\mathbf{U} = \begin{bmatrix} 0 & \chi_{12} & \cdots & \cdots & \chi_{1,n+1} \\ 0 & 0 & \chi_{23} & & \vdots \\ \vdots & & 0 & \ddots & \vdots \\ 0 & & & \ddots & \chi_{n,n+1} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

Using (Equation 3.15), the complete vector of fugacities can be computed from $\boldsymbol{\phi}$ with a number of floating point operations that scales as n^2, whereas evaluating (Equation 3.1) separately for each component scales as n^3. The new version (Equation 3.15) also benefits from the speed-up of vectorization in interpreted languages such as *MATLAB* and Python. Thus, (Equation 3.15) offers a significant speed-up for simulations with many components. The change shown in (Equation 3.15) also makes it significantly easier to derive the thermodynamic factors $\left(\Gamma_{ij} = \phi_i \frac{\partial \ln(f_i)}{\partial \phi_j} = \frac{\phi_i}{f_i}\frac{\partial f_i}{\partial \phi_j}\right)$ using standard matrix-vector calculus rules. However, to obtain the correct derivatives, we must first express $\mathbf{f}$ as a function of $\boldsymbol{\phi}_{1:n}$ only by eliminating the membrane volume fraction ϕ_m using $\phi_m = 1 - \sum_{i=1}^{n} \phi_i$. This is done most simply by observing that $\boldsymbol{\phi} = \mathbf{K}\boldsymbol{\phi}_{1:n} + \boldsymbol{\epsilon}$ with $\mathbf{K} \in \mathbb{R}^{(n+1)\times n}$ and $\boldsymbol{\epsilon} \in \mathbb{R}^{n+1}$ defined by

$$\mathbf{K} = \begin{bmatrix} \mathbf{I}_{n\times n} \\ -\mathbf{1}_n^{\mathrm{T}} \end{bmatrix}, \qquad \boldsymbol{\epsilon} = \begin{bmatrix} \mathbf{0}_n \\ 1 \end{bmatrix}, \tag{3.16}$$

where $\mathbf{I}_{n\times n}$ is the $n \times n$ identity matrix, $\mathbf{1}_n$ is an n-dimensional vector of ones, and $\mathbf{0}_n$ is an n-dimensional vector of zeros. Substituting this relation into (Equation 3.15) gives

$$\ln\left(\frac{\mathbf{f}}{\mathbf{f}^{\circ}}\right) = \ln\left(\boldsymbol{\phi}_{1:n}\right) + \mathbf{p} + \tilde{\mathbf{C}}\boldsymbol{\phi}_{1:n} - \mathbf{V}^{\circ}_{n+1}(\boldsymbol{\phi}^{\mathrm{T}}_{1:n}\tilde{\mathbf{Q}}\boldsymbol{\phi}_{1:n}), \tag{3.17}$$

where

$$\mathbf{p} = \mathbf{1}_n + \boldsymbol{\chi}_m - \frac{1}{V_m^\circ}\mathbf{V}^\circ,$$
$$\tilde{\mathbf{C}} = \mathbf{C}\mathbf{K} - \mathbf{V}^\circ\boldsymbol{\epsilon}^{\mathrm{T}}\mathbf{Q}^{\mathrm{T}}\mathbf{K},$$
$$\tilde{\mathbf{Q}} = \mathbf{K}^{\mathrm{T}}\mathbf{Q}\mathbf{K},$$

and $\boldsymbol{\chi}_m = (\chi_{1m}, \chi_{2m}, \ldots, \chi_{nm})$.

Additionally, this transition from a multi-line implementation to a single line compact matrix-vector equation allows for storage of constant matrices in parameter structs as well to streamline. The real difference in speed comes from differentiating this model to find the thermodynamic factor matrices. Before the compaction of this model, the Flory-Huggins thermodynamic matrix took the form (differentiating (Equation 3.1) with respect to $\boldsymbol{\phi}_{1:n}$), the thermodynamic factors $\Gamma_{ij} = \phi_i \frac{\partial \ln(a_i^{II})}{\partial \phi_j} = \frac{\phi_i}{\hat{f}_i^{II}} \frac{\partial \hat{f}_i^{II}}{\partial \phi_j}$ can be evaluated as:

For $i = j$

$$(\Gamma)_{ij} = \phi_i \Bigg(\frac{1}{\phi_i} - 1 - \overline{V}_i \sum_{k=1}^{i-1} \frac{\phi_k \chi_{ki}}{\overline{V}_k} + \overline{V}_i \sum_{\substack{k=1 \\ k\neq i}}^{n} \frac{\phi_k \chi_{mk}}{\overline{V}_j} + \frac{\overline{V}_i}{\overline{V}_m} - \sum_{k=i+1}^{n} \chi_{ik}\phi_k - \overline{V}_i \sum_{k=1}^{i-1} \frac{\phi_k \chi_{ki}}{\overline{V}_k} - \chi_{im} \sum_{\substack{k=1 \\ k\neq i}}^{n} \phi_k - 2\phi_m \chi_{im} \Bigg). \quad (3.18)$$

For $i < j$

$$(\Gamma)_{ij} = \phi_i \Bigg(-\frac{\overline{V}_i}{\overline{V}_j} + \overline{V}_i \sum_{k=1}^{i-1} \frac{\phi_k \chi_{ki}}{\overline{V}_k} + \sum_{\substack{k=i+1 \\ k\neq j}}^{n} \phi_k \chi_{ik} + \chi_{ij} \sum_{\substack{k=1 \\ k\neq i,j}}^{n} \phi_k + 2\phi_j \chi_{ij} - \overline{V}_i \sum_{\substack{k=1 \\ k\neq i}}^{j-1} \frac{\phi_k \chi_{kj}}{\overline{V}_k}$$
$$- \frac{\overline{V}_i}{\overline{V}_j} \sum_{k=j+1}^{n} \phi_k \chi_{jk} + \frac{\overline{V}_i}{\overline{V}_m} - \sum_{\substack{k=i+1 \\ k\neq j}}^{n} \chi_{ik}\phi_k - \overline{V}_i \sum_{k=1}^{i-1} \frac{\phi_k \chi_{ki}}{\overline{V}_k} - \chi_{im} \sum_{\substack{k=1 \\ k\neq i,j}}^{n} \phi_k + \overline{V}_i \sum_{\substack{k=1 \\ k\neq i,j}}^{n} \frac{\phi_k \chi_{mk}}{\overline{V}_j}$$
$$+ (\phi_m - \phi_j)(\chi_{im} + \chi_{ij} - \frac{\overline{V}_i}{\overline{V}_j}\chi_{jm}) - 2\phi_m \chi_{im} \Bigg). \quad (3.19)$$

For $i > j$

$$(\Gamma)_{ij} = \phi_i \Bigg(-\frac{\overline{V}_i}{\overline{V}_j} + \overline{V}_i \sum_{\substack{k=1 \\ k\neq j}}^{i-1} \frac{\phi_k \chi_{ki}}{\overline{V}_k} + \sum_{k=i+1}^{n} \phi_k \chi_{ik} + \chi_{ji} \frac{\overline{V}_i}{\overline{V}_j} \sum_{\substack{k=1 \\ k\neq i,j}}^{n} \phi_k + 2\phi_j \chi_{ij} \frac{\overline{V}_i}{\overline{V}_j} - \overline{V}_i \sum_{k=1}^{j-1} \frac{\phi_k \chi_{kj}}{\overline{V}_k}$$
$$- \frac{\overline{V}_i}{\overline{V}_j} \sum_{k=j+1}^{n} \phi_k \chi_{jk} + \frac{\overline{V}_i}{\overline{V}_m} - \sum_{k=i+1}^{n} \chi_{ik} \phi_k - \overline{V}_i \sum_{\substack{k=1 \\ k\neq j}}^{i-1} \frac{\phi_k \chi_{ki}}{\overline{V}_k} - \chi_{im} \sum_{\substack{k=1 \\ k\neq i,j}}^{n} \phi_k + \overline{V}_i \sum_{\substack{k=1 \\ k\neq i,j}}^{n} \frac{\phi_k \chi_{mk}}{\overline{V}_j}$$
$$+ (\phi_m - \phi_j)(\chi_{mi} + \chi_{ij} \frac{\overline{V}_i}{\overline{V}_j} - \frac{\overline{V}_i}{\overline{V}_j} \chi_{jm}) - 2\phi_m \chi_{im} \Bigg). \quad (3.20)$$

Now, instead of a page of equations and an implementation that requires nested *FOR* loops, (Equation 3.17) can be differentiated to yield:

$$\boldsymbol{\Gamma}(\boldsymbol{\phi}_{1:n}) = \mathbf{I}_{n\times n} + \text{diag}(\boldsymbol{\phi}_{1:n}) \left[\tilde{\mathbf{C}} - \mathbf{V}^{\circ} \boldsymbol{\phi}_{1:n}^{\mathrm{T}} \left(\tilde{\mathbf{Q}}^{\mathrm{T}} + \tilde{\mathbf{Q}} \right) \right], \quad (3.21)$$

where the required matricies are defined in (Equation 3.17). Using (Equation 3.21), the full $\boldsymbol{\Gamma}(\boldsymbol{\phi}_{1:n})$ matrix can be evaluated in order n^2 operations. Whereas (Equation 3.18), (Equation 3.20), and (Equation 3.21) is evaluated in n^3 operations. Additionally, this equation can also be found in subsection A.3.1.

Now that the vectorization of the model is complete, the results from the publication can be shown. The first thing to show is how the transport model fit into the overall data-driven machine learning framework. Again, the contribution in this dissertation is not the machine learning aspect, but how the transport model was tweaked to enable permeation simulations of complex crude oil mixtures with hundreds of components. Figure 3.7, Figure 3.8, and Figure 3.9 outline the framework, trained diffusivities, experimental vs. predicted Permain crude oil permeation through SBAD-1, and experimental vs. predicted Arab Light crude permeation through DUCKY-9. Overall, the transport modeling framework predicts the transport with reasonable accuracy.

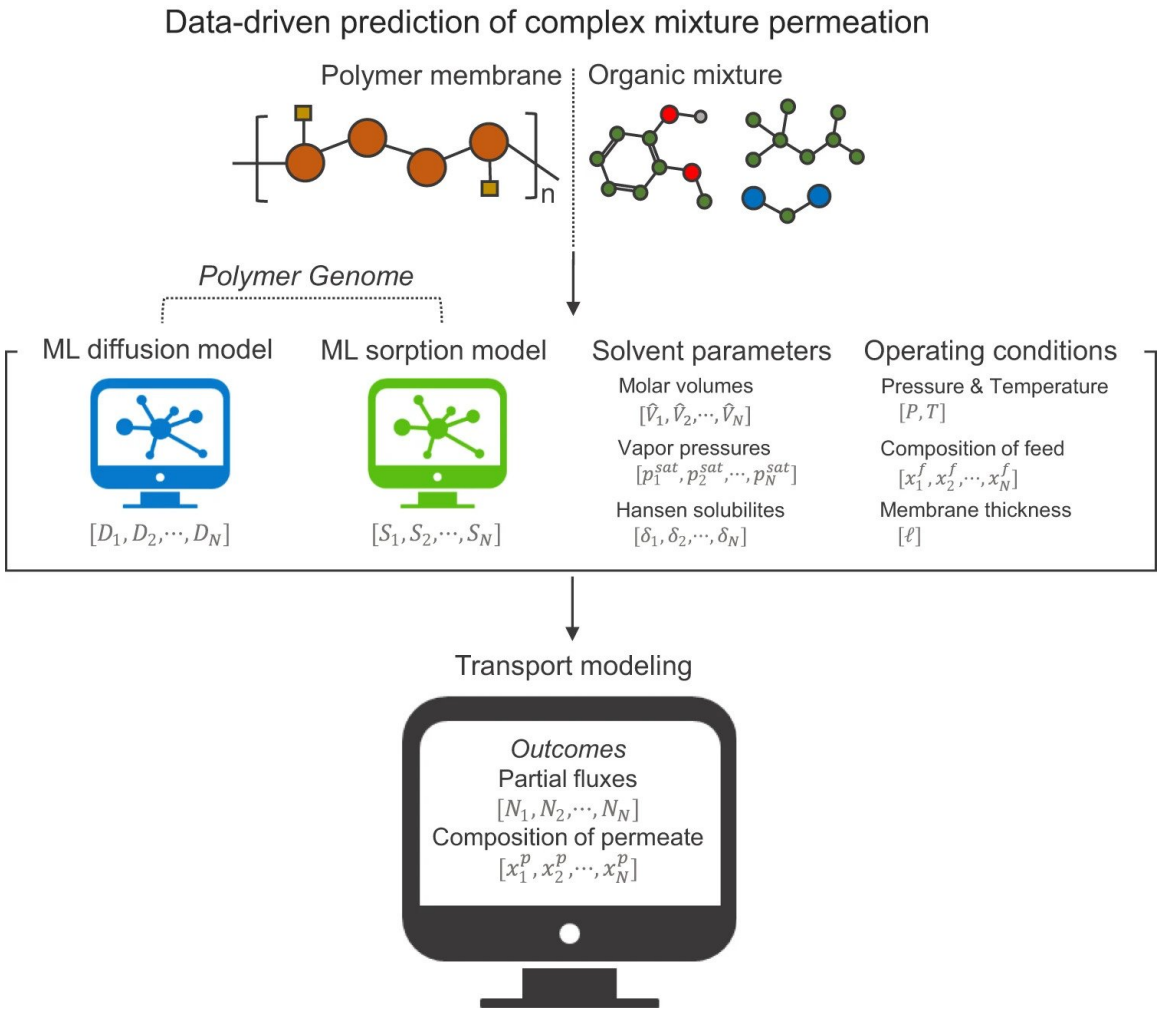


Figure 3.7: Polymer structures and solvent mixtures are converted to simplified molecular-input line entry system (SMILES) strings and used as inputs for machine-learning algorithms designed to relate polymer-solvent structure to solvent diffusivities (D) and solubilities (S) within polymer membranes. These parameters – in addition to the various physicochemical properties of the solvents (e.g., molar volumes (V_i), vapor pressures (p_{sat}), Hansen solubility parameters (δ_i)) at the desired operating conditions (e.g., pressure (P), temperature (T), composition of the feed mixture ($\mathbf{x}_I$), membrane thickness (l) – are then used as inputs into an N-component Maxwell-Stefan model that outputs a vector of fluxes ($\mathbf{N}$) and compositions ($\mathbf{x}^{IV}$) for each component permeating through the membrane.[69].

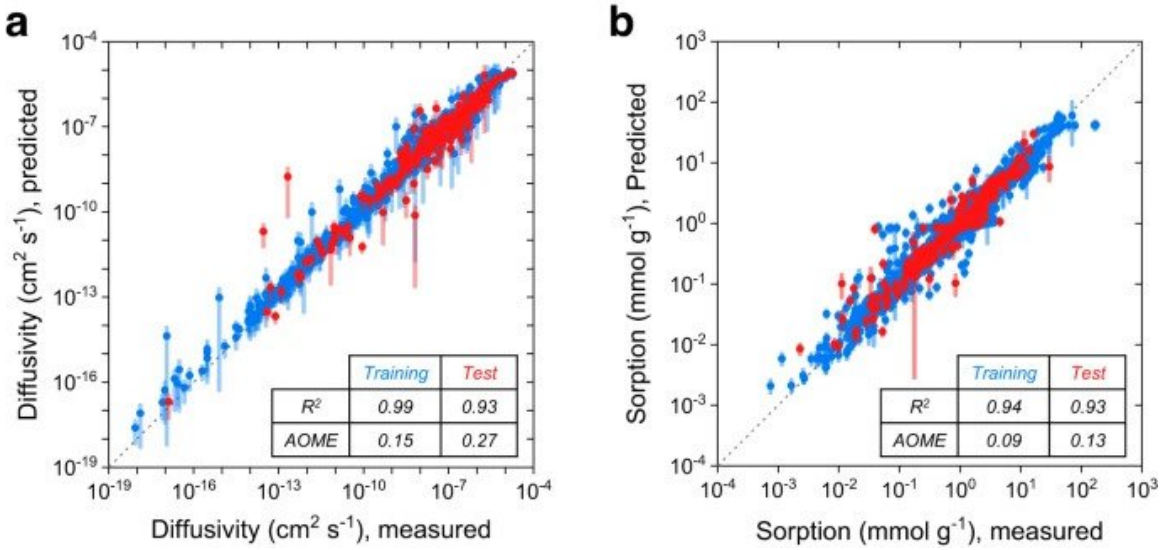


Figure 3.8: **a, b** Parity plots between experimentally obtained and ML predicted diffusion coefficients and sorption uptakes, respectively (the methods of the model development are described in "Methods" section of Lee et al.). Both diffusion and sorption models are trained using 10-fold cross-validation (CV). The 10 models from the 10 CV splits were used to make predictions on the 90% training (blue) and the 10% test set (red). The error bars on each point represent the standard deviations from 10 predictions. R^2 and AOME in the plots are defined as the coefficient of determination and averaged order of magnitude error, respectively [69].

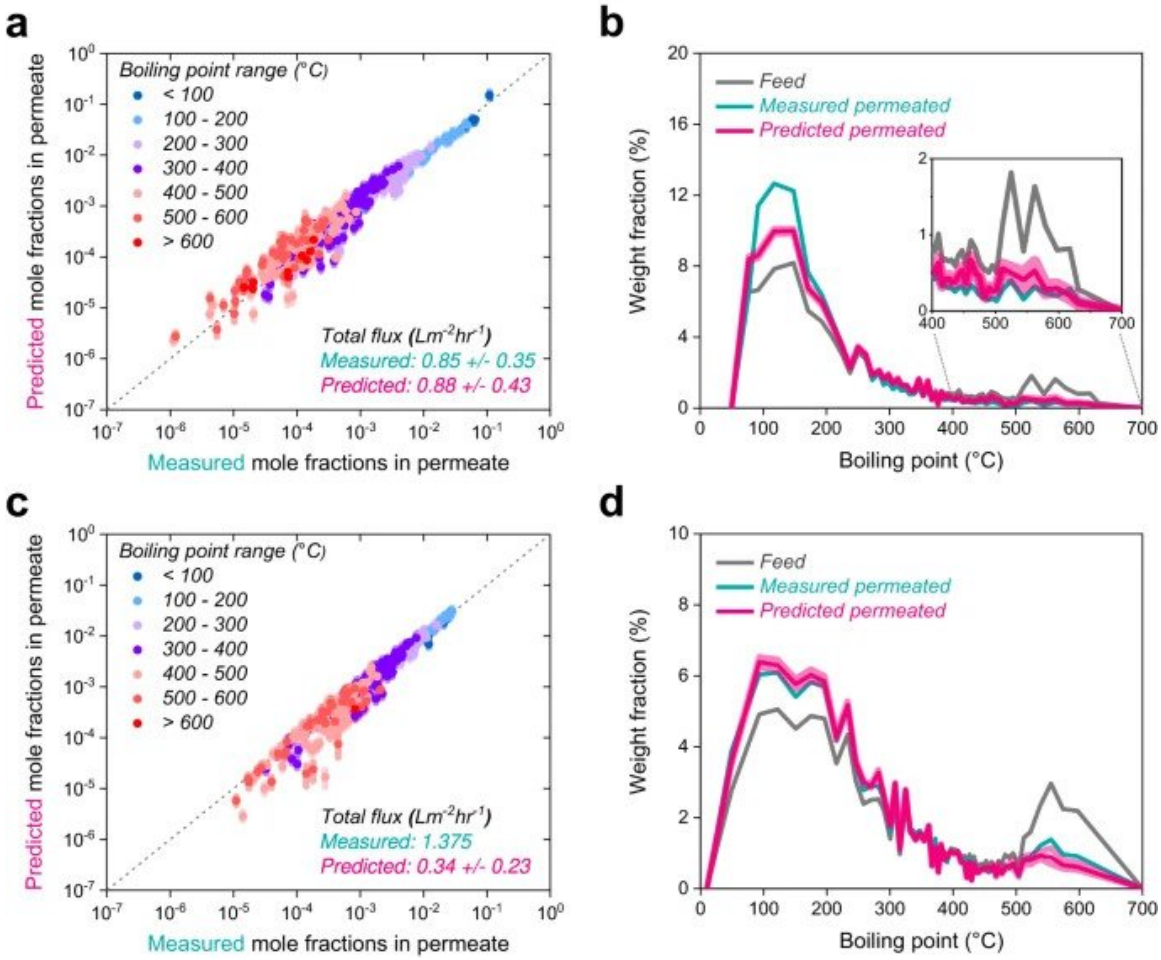


Figure 3.9: **a, c** Parity plots comparing experimental and predicted permeate mole fractions after the fractionation of Permain crude oil by SBAD-1 membrane (**a**) and Arab Light crude oil by DUCKY-9 membrane (**c**). Blue-to-red colors are assigned to different boiling point ranges of molecules in the crude oil mixtures. The shaded area around each point represents the standard deviation of the permeate concentration predictions for each molecule. The deviations are from the uncertainty in the machine learning (ML) sorption model parameter predictions. The deviations in the total flux predictions are the uncertainty in the ML diffusion model predictions. b, d Differential weight fraction relative to boiling points of molecules in the Permian crude oil mixture before and after fractionation by SBAD-1 membrane (**b**) and in the Arab Light crude oil mixture before and after fractionation by DUCKY-9 membrane (**d**). The curve shows the local slope of the concentration/boiling point over a period of 6 molecules. The lighter shade displays the deviation in the predicted weight fractions of the permeate [69].

CHAPTER 4
DEVELOPMENT OF A SOFTWARE PACKAGE FOR USE IN PROCESS SIMULATION ENVIRONMENTS

4.1 Overview

This chapter outlines the software package, *asyMemSim*, that was developed over the course of this research project in collaboration with Dr. Ryan Lively, Dr. Ronita Mathias, and Youngjoo Lee. The purpose of this chapter is to present a succinct and complete description of the simulation problem this code aims to solve, how the code is structured (input/output files, coding standards applied, modeling and simulation capabilities, code file indexes, etc.), an exemplar tutorial simulation, custom model implementation details, and finally the extension of this code to be used within process simulation environments. In doing so, this dissertation will serve as a document for practicing researchers and engineers to use this open-source software in their own applications, and possibly build other simulation codes for different applications using this structure. The motivation for this software is that such an open-source pressure-based industrial membrane modeling and simulation package is non-existent (as described in subsection 1.3.3). Thus, this chapter will be a worthwhile reference for years to come.

4.2 Simulation Problem Description

This code implements the exact simulation problem outlined in section 2.2. Please refer back to that section for a complete description of the *local transport problem* for complex mixture asymmetric membrane transport. For the specific modeling and simulation capabilities this software can execute, see subsection 4.3.3.

4.3 *asyMemSim* Code Structure

In order to properly run this code, the user needs to be familiar with the general philosophy used to organize and construct *asyMemSim*. The next set of subsections will expand upon this idea. When applicable, the direct code is presented to show the exact inner workings, and how each piece comes together to create a single working piece of software. Additionally, the general naming conventions and etiquette are presented in subsection 4.3.1.

As an overview of the code structure, the general types of code files found within are as follows: centralized databank files, single mixture simulation initialization scripts, experimental specification files, model and solver specification files, diffusion coefficient parameter fitting scripts, single-component permeation simulations with varied system condition loop scripts, multiple mixture/model/membrane simulation loop scripts, ODEs solver initialization files, ODEs right-hand side (RHS) functions, DAE solver initialization files, DAEs RHS functions, shooting algorithm numerical method scripts, shooting algorithm RHS functions, full-discretization solver initialization functions, full-discretization RHS functions, fugacity model matrix and model evaluations for transport modeling framework (e.g. thermodynamic factor evaluation functions, Phase I/II phase equilibrium solvers with respective RHS functions, and Phase II/III phase equilibrium solvers with respective RHS functions), diffusion/fugacity model parameter correlation evaluations, numerical method initialization functions with respective RHS functions. The complete code file index can be found in subsection 4.3.4.

4.3.1 General Code Etiquette and Naming Conventions

Throughout my graduate career, a number of helpful code writing tools have been learned that should be mentioned in this dissertation before going into the specific files within this piece of software. The first piece of advice is to always include a preamble in every function that gives details on the function name, a short description, and the required input/output

variables. A quick example is shown below

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function:    evalFH_RHS(stateVar,params)                                    %
% Description: Evaluate RHS of Flory-Huggings equation.                       %
% Input:       stateVar - (ODE, FH) n+1 dimensional vector of volume          %
%                           fractions in membrane phase                       %
%                         (DAE, FH-LM or DSM) 2*n+1 dimensinal vector         %
%                           of n+1 volume fractions and n fugacities          %
%                           of membrane phase                                 %
%              params   - struct of system parameters                         %
%                           (see dataBank function for specs)                 %
% Output:      FH_RHS   - nonlinear function value of FH RHS equation         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

With these preambles, the effort required to dig and debug by an end-user will be greatly reduced since they can instantly understand the purpose of the function. Another good code standard that has been applied throughout all files is to name variables by starting with lower case letters and then having words begin with an uppercase letter rather than separating them by "_". In some cases, this is necessary if there are short variable extensions or names (e.g. "stateVar_1", "FH_RHS"). Overall this has made the code more readable, especially when straying away from single letter variable names like "B". Sometimes, the single letter variable names are obvious if one is familiar with the equations from the modeling framework documentation such that exceptions can be made. However, most of the time, it is not. The last piece of advice that is exemplified throughout the code is proper commenting where it is needed. Without this, again the end-user who needs to dig and debug will again have many hours wasted trying to understand the purpose of certain variables and sections of code. By utilizing these coding standards, the software presented in this dissertation aims to alleviate the difficulty in reproducing, re-purposing, and reusing *asyMemSim* for any possible application.

4.3.2 Input File Descriptions

Here, the details for each of the inputs files is described. Generally, there are three main input files required to run the code. The first is the central *dataBank.m* file that is a repository for penetrant and membrane parameters. The second is *solverModelSpec.m*, which

allows the user to specify exactly the different fugacity models, diffusion models, numerical method, and solver specifications. The last input file is *expSpec.m*, which allows for exact specification of the experimental conditions such as the mixture components, mixture compositions, membrane, Phase I pressure, Phase III pressure, system temperature, etc. Please refer to the following sub-subsections for more detail.

Parameter Database Function – dataBank.m

This function is the database for all membrane and component parameters. There is functionality for custom polymers to be added by copy and pasting the "if...then" statement that checks for a certain polymer name as a template. See subsection 4.5.1 for more details on adding additional membrane material and mixture parameters.

Please see the code-snippet below for a good visual aid

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function:    dataBank(sysInfo)                                          %
% Description: Build parameter matricies based on system specifications.  %
% Input:       sysInfoExt - external struct defining simulation specs     %
%                             (memID, mixID, yf, n, lmem, Pu, Pd, T, R    %
%                              memPhaseModel, diffModel, swlDiffModel)    %
% Output:      params     - struct of system parameters                   %
%                             (compID, n, Vs, HanSolParam, psat, chis [FH %
%                             or FH-LM], diffs, Ch & bs [FH-LM or DSM],   %
%                             ks [DSM], unitActPhis, Bffv, and all        %
%                             fields listed in sysInfo struct)            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [params] = dataBank(sysInfo)
%--------------------------------------------------------------------------------------------------------------------%
%full parameter/property sets
    if contains(sysInfo.memID,'SBAD1')
      % Lively/Ronita's 9-comp data TOL/MCH/1MN/DCN/NOC/IOC/TBB/TPB/ICE/SBAD-1
        params.lmem = 0.3; % thickness of active membrane layer um
        params.compID = struct('TOL', 1, 'MCH', 2, 'MNP', 3, 'DEC', 4, 'NOC', 5, 'IOC', 6, 'TBB', 7, 'TPB', 8, 'ICT', 9)
              ;
        params.n = 9;
        params.Vs = [106.521;128.123;139.823;156.962;163.42;165.552;155.529;240.069;293.267;62326]; %cm3/mol
        params.HanSolParam = [18,1.4,2;16,0,1;20.6,0.8,4.7;18,0,0;15.5,0,0;14.1,0,0;17.4,0.1,1.1;18,0,0.6;16.3,0,0]; %[
              delD,delP,delH;...]
        params.psat = [28.998;46.596;0.059;0.975;14.805;49.087;2.115;0.0352;0.0458];%torr
        if contains(sysInfo.memPhaseModel,'F-H')
            params.chis_im = [0.871;1.672;0.705;2.783;1.163;3.049;1.648;2.5;3.130];
            if sysInfo.lowDiffErrorBar == 0
                params.diffs_im = [12.7;2.8617;2.8617;2.8617;2.8617;2.8617;2.8617;2.8617;2.8617]; %um2/s FH Exp Based
                     tweaked for single comp MS diffs
```

```
        elseif sysInfo.lowDiffErrorBar == 1
            params.diffs_im = [14.157;2.870;0.042;1.296;4.458;4.2057;0.975;0.069;0.536]; %Mixed Exp (LOW ERROR)
                 Based FH Tweaked to match single comp FH um2/s
        end
        if sysInfo.memPhaseModel_FicksOG == 1
            if sysInfo.lowDiffErrorBar == 0
                params.diffs_im = [3.622;3.329;0.009;1.395;2.369;6.091;0.833;0.055;0.520]; %um2/s FICKS FH-BC
                     tweaked for single comp 20bar
            elseif sysInfo.lowDiffErrorBar == 1
                params.diffs_im = [3.316;1.709;0.0076;1.085;1.838;3.6712;0.580;0.055;0.473]; %um2/s FICKS (LOW ERROR
                     ) FH-BC tweaked for single comp 20 bar
            end
        end
    elseif contains(sysInfo.memPhaseModel,'DSM')
    ...
```

Please note that the above code snippet is not the full function but just the first complete fugacity model and membrane parameter definition block. Additionally, the *dataBank.m* function is separated by sorption model since for any given membrane, there may be different fugacity model parameters. Then with each fugcity model, there may be different Maxwell-Stefan (MS) penetrant-membrane diffusivites to be fit. The order of magnitudes should be the same, but the values will vary.

Other notes are that there are also separate sections for "lowError" MS penetrant-membrane diffusivities, and Fick's law diffusivities that assume a constant thermodynamic contribution (Γ_{ii} constant and $\Gamma_{ij} = 0$) that will need to be updated after running each respective diffusivity fitting script has been run (if desired).

Units are given in the nomenclature section and also commented out to each side. If not units are given, then it is assumed that the variable is unitless. The units were chosen to give order of magnitude of equations to be $O(1)$. The main list of parameters in this function will be listed below to help understanding:

- params.lmem – thickness of active layer (Phsae II), μm
- params.compID – struct defining component name and order of parameters to be defined in subsequent parameter set definitions
- params.n – total number of components for which parameters are defined below

(equal to number of components listed in "params.compID")

- params.Vs – penetrant pure component molar volumes, cm^{3}mol^{-1}, and V_m is the membrane molar volume that is assumed to be equal to a large estimate $\bar{V}_m = 62,326$ cm^{3}mol^{-1} based on polymer weight average molecular weight and bulk density

- params.HanSolParam – Hansen solubility parameters (dispersion, intermolecular forces, and hydrogen bonding), note $= [\delta_{D1}, \delta_{P1}, \delta_{H1}; ...; \delta_{Dn}, \delta_{Pn}, \delta_{Hn}]$ where n is equal to params.n, MPa$^{0.5}$

- params.psat – permeant vapor pressures evaluated at system T, torr

- params.chis – **Flory-Huggins or combined Flory-Huggins-Langmuir sorption model** χ_{im} where m refers to the membrane component and $\chi_{ij} = 1$ are placeholders until relevant Hansen solubility parameter relationships are applied

- params.bs – **Dual-mode or Flory-Huggins-Langmuir sorption** Langmuir affinity constants, torr^{-1}

- params.Ch – **Dual-mode or Flory-Huggins-Langmuir sorption** volume based Langmuir free volume capacities, mol $(\text{cm}^3 \text{ polymer})^{-1}$

- params.ks – **Dual-mode sorption** Henry's type sorption parameter, $(\text{cm}^3 \text{ solvent})(\text{cm}^3 \text{ polymer})^{-1}\text{torr}^{-1}$

- params.diffs Maxwell-Stefan penetrant-membrane Đ$_{im}$ and Đ$_{ij} = 1$ as placeholder until cross-diffusivity relationship is applied or user-specified cross-diffsivity Đ$_{ij}$ are given, μm$^2 s^{-1}$; also note there are 4 sets of diffusivities for one given soprtion model (normal MS, lowFluxError MS, normal Fick's law, and lowFluxError Fick's law) and if doing error calculations

With all these parameter pieces in place, one can move on to the model and numerical method specifications.

Numerical Method and Model Specifications — solverModelSpec.m

This paragraph deals with the more technical specifications of how to solve the local flux complex mixture modeling framework presented in chapter 2. Below is a code-snippet example of the input file.

```
function [sysInfo] = solverModelSpec_EXAMPLE()
%-------------------------------------------------------------------------------------------------------------------%
%sorption/diffusion model and thermodynamic assumpstions spec
  %sorption membrane phase model
    sysInfo.memPhaseModel = "F-H"; %other options: "FH-LM", "F-H", and "DSM"
  %specify thermodynamic assumptions
    sysInfo.noThermoCoupling = 0;
    sysInfo.memPhaseModel_FicksOG = 0;
  %specify diffusional relationships
    sysInfo.diffModel = "Vignes";
    sysInfo.swlDiffModel = "none";
%-------------------------------------------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------------------------------------------%
%numerical method sepcs
  sysInfo.numMethod = "NormShootAlg"; %classical shooting algorithm (SA) [Recommended]
  %  sysInfo.numMethod = "FullDis";
  %  sysInfo.numNodes = 1;
  %  sysInfo.numMethod = "bvp4cTEST";
  %solver specs
    sysInfo.solverSpec = "trust-region-dogleg"; %default, use for FUD
    sysInfo.iterDetail = 1; %0 = all main solver output will be suppressed
  %initial guess specs
    sysInfo.initGuessApprox = 0; %0 = use feed mol fraction vector and small total flux value (use if Forward Euler IVP
          approx fails)
    sysInfo.nodalGuessApprox = 0; %only applicable to FUD methods N>1
    sysInfo.customInitGuessApprox = []; %set to match your experimental compostion number plus total flux as [n+1 by 1]
          vector for custom initial guess
  %sys of eqns spec
    sysInfo.currentStateLit_eqnSetup = 0; %default == 0 (no difference to genreal simulations)
    sysInfo.noGammaFugacityODEs = 0; % ONLY USE FOR FULL MS and SA or FUD method (FICK == 0 and noThermo == 0)! option
          keep MS system interms of fugacity gradients and do not make chain rule trick to get gamma matrix
%-------------------------------------------------------------------------------------------------------------------%
```

The following choices are for "sorption membrane phase model" where **sysInfo.memPhaseModel** can equal ("F-H", "DSM", "FH-LM") which stand for classical Flory-Huggins, Dual-mode sorption, or our novel combined Flory-Huggins-Langmuir. Model derivations can be found in subsubsection 3.2.2. Note that, for some scripts a vector

sysInfo.modSpec will be specified instead to test different sorption models at once. Please see the diffusivity fitting script and single component prediction script EXAMPLE inputs for direct use.

The thermodynamic assumptions are a simple logic 1 == yes or 0 == no. Then for cross-diffusion models, **sysInfo.diffModel** can equal ("NoCoupling", "Vignes", "Darken", or "Fudge") where the first are self explanatory, and the last option is used if user specified cross-diffusivites are used.

Then, there are the specifications of the membrane swelling diffusivity model where **sysInfo.swlDiffModel** can equal ("none", "FFV", "Avg-Diff") where "none" uses constant diffusivities, "FFV" uses the fractional free volume novel diffusion model presented in subsubsection 3.2.2. Then "Avg-Diff" utilizes the novel sorption-vection average cohort style diffuson presented in subsubsection 3.2.2.

Next, one can specify which method they would like to use to simulate the local flux transport where **sysInfo.numMethod** can equal ("NormShootAlg", "FullDis", and "MultShootAlg"). See chapter 2 for extended details. Regardless, the first is the single shooting algorithm, second is the full discritization style solve, and the last is a multiple shooting numerical method (which is a future direction of the code for advanced users, and not to be used by the end-user).

****Note** if **sysInfo.numMethod** equals "FullDis" then the additional variable **sysInfo.numNodes** must be provided with good initial value of "5".

****Note** if **sysInfo.numMethod** equals "MultShootAlg" then the additional variable **sysInfo.numShootPoints** must be provided with good initial value of "5". Also **sysInfo.casADi** is a logical that must be provided and only used for FH-LM sorption model, with Fick's law and no thermodynamic coupling options disabled (==0) since casADi multiple shooting capability is still in development for the other models (it really only needs to be used for the difficult system to solve using FH-LM and full Maxwell-Stefan).

The **recommended method** is our shooting algorithm, but for a quick estimate that is

probably within less than 20% error in most cases (see section 2.5), full discritization is recommended. For strong diffusional coupling ($Đ_{ij} << 1$), the multiple shooting point algorithm is recommended with number of shooting points starting at 5, and increasing if needed.

The next part of the numerical method specifications is the outer solver algorithm specifications **sysInfo.solverSpec**, iterate detail option, and if different approximation of the initial guess is required to make the problem converge with less iterations–or vice-versa **sysInfo.initGuessApprox** or **sysInfo.nodalGuessApprox**. If simulating a large component system that is fairly sensitive to the initial guess, the option **sysInfo.customInitGuessApprox** allows for the input of custom $n+1$ unknowns (n component phase III molar compositions and 1 total volumetric flux variable). If the value is equal to "[ ]", then the custom initial guess option will not be used. Alternatively, if "$[x_1^{III}; \ldots; x_n^{III}; N_{tot}^V]$ is set equal to the custom initial guess variable, other outer solver initialization strategies will be circumvented and that custom initial guess will be used. The other solver options are those found in the *fsolve* solver options page (i.e. **sysInfo.iterDetail**). The specified algorithm has been found to be generally applicable.

****Note** for the multiple mixture/model prediction loop, only the numerical method specifications are needed.

Experimental Specifications – expSpec.m

This is the last required input file, and is the most pertinent to the experiment or industrial process stream the end-user is trying to simulate. The function takes the form

```
function [sysInfo] = expSpec_EXAMPLE()
%-------------------------------------------------------------------------------------------------------------------%
%specify single simulation
%mixture components, compositions, system parameters
  %YJL MA/DU9/DU10
%       sysInfo.memID = "Matrimidd";  % polymer spec
    %9 COMP M1
%     sysInfo.mixID = ["IOC","NOC","MCH","TOL","DEC","TBB","ICT","TPB","MNP"];  % mixture spec
%     sysInfo.yf = [0.117;0.188;0.197;0.327;0.089;0.0385;0.0109;0.013;0.016];  % composition spec (must sum to 1)
%     PuM = 40; %bar
  %SBAD1 Data
```

```
    sysInfo.memID = "SBAD1";  % polymer spec
%     %9 COMP M1
    sysInfo.mixID = ["TOL","MCH","MNP","DEC","NOC","IOC","TBB","TPB","ICT"];  % mixture spec
    sysInfo.yf = [0.171;0.281;0.0199;0.107;0.221;0.15;0.0217;0.0158;0.013];  % composition spec (must sum to 1)
    PuM = 40; %bar
    %3 COMP M1
%     sysInfo.mixID = ["TOL","IOC","ICT"];  % mixture spec
%     sysInfo.yf = [0.282;0.385;0.333];  % composition spec (must sum to 1)
%     PuM = 30; %bar
%     sysInfo.memID = "PIM1";  % polymer spec
% %     %5 COMP M1
%     sysInfo.mixID = ["TOL","HEP","PXY","OXY","ICT"]; % mixture spec
%     sysInfo.yf = [0.257;0.216;0.205;0.264;0.058];  % composition spec (must sum to 1)
%     PuM = 30; %bar
%     %3 COMP M2
%      sysInfo.mixID = ["TOL","HEP","PXY"]; % mixture spec
%      sysInfo.yf = [0.345;0.362;0.293]; % composition spec (must sum to 1)
%      PuM = 1; %bar
     %2 COMP M2
%     sysInfo.mixID = ["HEP","OXY"]; % mixture spec
%     sysInfo.yf = [0.01;0.99]; % composition spec (must sum to 1)
%     PuM = 1; %bar
%     sysInfo.memID = "MATRIMID";
%     sysInfo.mixID = ["TOL"];
%     PuM = 1;
%     sysInfo.yf = 1;
  %system specs
    sysInfo.n = length(sysInfo.yf);  % number of permeants
    sysInfo.Pu = PuM*0.9869;  % feed side pressure [bar]*0.9832 = [atm]
    sysInfo.Pd = 1;  % support side pressure (atm)
    sysInfo.pervapMode = 0; % %BETA(UNSTABLE) pervap capability -- yes if == 1 (assume downstream pressure = 0 bar)
    sysInfo.T = 295;  % system temperature (K)
    sysInfo.R = 82.05;  % gas constant (atm cm^3/ mol K)
    sysInfo.diffFit = 0; %see code below for diffusivity fitting
    sysInfo.crossDiffFudge = 0; %if = 0 then no specified cross-diffusivities D_ij will be used
    if sysInfo.crossDiffFudge == 1
        sysInfo.crossDiffSpecs = [1,2,1,3]; % e.g. [i,j] = D_ij^MS
        sysInfo.crossDiffVals = [0.1,0.5]; %um2/s
    end
    sysInfo.lowDiffErrorBar = 0; % if = 1 then diffusivities fit to low error sing comp will be used
%-------------------------------------------------------------------------------------------------------------%
end
```

A good rule-of-thumb when using *asyMemSim* is that the lower "system specs" section is usually constant and self explanatory across the different main scripts, whereas the top section will change based on the different main script used. This will be explained further in the next subsection. The above code snippet also has various mixtures and membrane mixture simulations that can be easily commented in and out. Another good feature to note is that even though the *dataBank.m* file can have as many component parameters

defined as needed, the end-user has the ability to pick any custom mixture such that the code will automatically build the required and correctly sized parameter vectors to simulate the complex mixture membrane permeation.

4.3.3 Modeling and Simulation Capabilities

This subsection deals with the main software scripts of *asyMemSim* and what it is generally capable of in terms of modeling and simulation. As an overview, the current software can either fit diffusivities based on a set of experimental single component permeation flux data across either one or multiple fugacity models at a time, simulate a single mixture permeation, iterate over a loop of single component permeations with different experimental conditions and/or models, and run multiple mixture simulations for different numerical methods/fugacity models/diffusion models/initial guess strategies. The latter script was used extensively for the numerical comparisons in section 2.5, the extended robustness tests in subsection 2.5.7, and the experimental validations for different modeling scenarios presented in subsubsection 3.2.2.

Diffusivity Fitting

This script deals with input of single component permeation flux data (including the low flux error), to fit model diffusivities for each polymer and sorption model specified. See expSpec.m input file below

```
function [sysInfo] = expSpec_EXAMPLE()
%-------------------------------------------------------------------------------------------------------------%
%specify exp

%mixture components, compositions, system parameters
  %SBAD1 Data
    sysInfo.memID = "SBAD1";  % polymer spec
    sysInfo.mixID_OG = ["TOL","MCH","MNP","DEC","NOC","IOC","TBB","TPB","ICT"];  % mixture spec **NUMBER of entries
          should equal params.compID number listed in dataBank.m
    sysInfo.singFluxVec = [8.76;0.74;0.054;0.063;2.23;0.302;0.23;0.0054;0.0266]; %LMH 20 bar flux (30 IOCT)
    sysInfo.singFluxVec_Error = [0.74;0.36;0.011;0.014;0.5;0.120;0.07;0;0.0024]; %LMH 20 bar flux error (30 IOCT)
%     sysInfo.memID = "PIM1";  % polymer spec
%     sysInfo.mixID_OG = ["TOL","HEP","PXY","OXY","ICT"];
%     sysInfo.singFluxVec = [10.04;27.7;7.88;4.74;0.93]; %LMH %20baru flux
%     sysInfo.singFluxVec_Error = [2.58;5.20;1.98;0.87;0.424]; %LMH %20bar flux error
```

```
    PuM = 20; %feed-side pressure (bar)
%     sysInfo.memID = "MATRIMID";  % polymer spec
%     sysInfo.mixID_OG = ["TOL","MES","TPB"];
%     sysInfo.singFluxVec = [0.798;0.0666;0.00734]; %LMH %20baru flux
%     sysInfo.singFluxVec_Error = [0;0;0]; %LMH %20bar flux error
%     PuM = 20; %feed-side pressure (bar)
  %system specs
    sysInfo.n = 1;  % number of permeants
    sysInfo.Pu = PuM*0.9869;  % feed side pressure [bar]*0.9832 = [atm]
    sysInfo.Pd = 1;  % support side pressure (atm)
    sysInfo.pervapMode = 0; % %BETA(UNSTABLE) pervap capability -- yes if == 1 (assume downstream pressure = 0 bar)
    sysInfo.T = 295;  % system temperature (K)
    sysInfo.R = 82.05;  % gas constant (atm cm^3/ mol K)
    sysInfo.diffFit = 0; %see code below for diffusivity fitting
    sysInfo.crossDiffFudge = 0; %if = 0 then no specified cross-diffusivities D_ij will be used
    if sysInfo.crossDiffFudge == 1
        sysInfo.crossDiffSpecs = [1,2,1,3]; % e.g. [i,j] = D_ij^MS
        sysInfo.crossDiffVals = [0.1,0.5]; %um2/s
    end
    sysInfo.lowDiffErrorBar = 0; % if = 1 then diffusivities fit to low error sing comp will be used
%-----------------------------------------------------------------------------------------------------------------%
end
```

The vectors "singFluxVec" and "singFluxVec_error" are to be the exact size and component order and the defined "sysInfo.mixID_OG" is set to. The only other input spec for this script that is slightly tweaked is the *solverModelSpec.m* as shown below

```
function [sysInfo] = solverModelSpec_EXAMPLE()
%-----------------------------------------------------------------------------------------------------------------%
%sorption/diffusion model and thermodynamic assumpstions spec
  %sorption membrane phase model
    sysInfo.modSpec = [1 3]; %specify which models to fit diffusivities for -- FH - 1 ; DSM - 2; FH-LM - 3

  %specify thermodynamic assumptions
    sysInfo.noThermoCoupling = 0;
    sysInfo.memPhaseModel_FicksOG = 1; %==1 if you want to fit Fick's diffusivities

  %specify diffusional relationships **set for parameter fitting**
    sysInfo.diffModel = "NoCoupling";
    sysInfo.swlDiffModel = "none";
%-----------------------------------------------------------------------------------------------------------------%
%-----------------------------------------------------------------------------------------------------------------%
%numerical method sepcs **best specs for fitting code**
    sysInfo.numMethod = "FullDis"; %classical shooting algorithm (SA) [Recommended]
    sysInfo.numNodes = 1;
  %solver specs
    sysInfo.solverSpec = "trust-region-dogleg"; %default, use for FUD
    sysInfo.iterDetail = 0; %0 = all main solver output will be suppressed
  %initial guess specs
    sysInfo.initGuessApprox = 0; %0 = use feed mol fraction vector and small total flux value (use if IVP approx fails)
    sysInfo.nodalGuessApprox = 0;
    sysInfo.customInitGuessApprox = 0; %set to match your experimental compostion number plus total flux as [n+1 by 1]
          vector for custom initial guess
```

```
  %sys of eqns spec
    sysInfo.currentStateLit_eqnSetup = 0; %default == 0 (no difference to genreal simulations)
    sysInfo.noGammaFugacityODEs = 0; % ONLY USE FOR FULL MS (FICK == 0 and noThermo == 0)! option keep MS system interms
          of fugacity gradients and do not make chain rule trick to get gamma matrix
%-------------------------------------------------------------------------------------------------------------------%
```

Now the main diffusivity script can be shown and ran as

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% script:    diffFit_asyMemLocal.m                                        %
% Description: Asymmetric membrane local flux model/method solver for:    %
%                -single component diffsivity fitting from single exp     %
%                (See tutorial document for specific info regarding use)  %
%                                                                         %
% Copyright 2021 Georgia Tech Research Corporation                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%-------------------------------------------------------------------------------------------------------------------%
%add /SOURCE/ folder and /casADi-v3.5.5/ to MATLAB session search path
    soFolder = fullfile(pwd,'..','..','SOURCE');
    mainFolder = fullfile(pwd,'..','..');
    addpath(fullfile(soFolder,'casadi-v3.5.5'));
    addpath(genpath(soFolder));
    addpath(genpath(mainFolder));
%-------------------------------------------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------------------------------------------%
%specify simulation input file names and load data
    sysInfoEXP = expSpec_EXAMPLE(); %change function name to match input file name
    sysInfoSMS = solverModelSpec_EXAMPLE(); %change function name to match input file name
    sysInfo = cell2struct([struct2cell(sysInfoEXP);struct2cell(sysInfoSMS)],[fieldnames(sysInfoEXP);fieldnames(
         sysInfoSMS)]);
    sysInfo.dataBankName = "dataBank_EXAMPLE";
%-------------------------------------------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------------------------------------------%
%diffusivity fit from sing comp permeation
    singFluxVec = sysInfo.singFluxVec; %LMH 20 bar flux (30 IOCT)
    singFluxVec_Error = sysInfo.singFluxVec_Error; %LMH 20 bar flux error (30 IOCT)
    singFluxVec_LOW = singFluxVec-singFluxVec_Error; %LMH
    modSpec = sysInfo.modSpec;
  %run fitting function
    [allModSingCompDiff,allModSingCompDiffLOW] = ...
        asyMemDiffFitSolve(sysInfo,singFluxVec,singFluxVec_Error,modSpec);
%-------------------------------------------------------------------------------------------------------------------%
```

With those scripts ran, and using the output vectors, one can update the *dataBank.m* function for each normal and low diffusivity for each polymer fit. Then to fit Fick's law diffusivities set "sysInfo.memPhaseModel_FicksOG = 1" as seen in the first line of the above figure and rerun the script/update of "dataBank.m".

Single Mixture Simulation

This script is used when one wants to look at a single mixture local flux membrane permeation simulation. The main script takes the form

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% script:    singMixSim_asyMemLocal.m                                      %
% Description: Asymmetric membrane local flux model/method solver for:    %
%                -single mixture permeation simulation                     %
%                (See tutorial document for specific info regarding use)  %
%                                                                          %
% Copyright 2023 Georgia Tech Research Corporation                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%-----------------------------------------------------------------------------------------------------------------%
%add /SOURCE/ folder and /casADi-v3.5.5/ to MATLAB session search path
    soFolder = fullfile(pwd,'..','SOURCE');
    mainFolder = fullfile(pwd,'..');
    addpath(fullfile(soFolder,'casadi-v3.5.5'));
    addpath(genpath(soFolder));
    addpath(genpath(mainFolder));
    addpath(fullfile(pwd,'DATA_353_NEW'));
    addpath(fullfile(pwd,'DATA_bio_3'));
    addpath(fullfile(pwd,'Aug_2023_Vignes_Sim_Data'));
%-----------------------------------------------------------------------------------------------------------------%

%-----------------------------------------------------------------------------------------------------------------%
%specify simulation input file names and load data
    sysInfoEXP = expSpec_EXAMPLE_VIG(); %change function name to match input file name
    sysInfoSMS = solverModelSpec_EXAMPLE_VIG(); %change function name to match input file name
    sysInfo = cell2struct([struct2cell(sysInfoEXP);struct2cell(sysInfoSMS)],[fieldnames(sysInfoEXP);fieldnames(
         sysInfoSMS)]);
    sysInfo.dataBankName = "dataBank_EXAMPLE_VIG";
%-----------------------------------------------------------------------------------------------------------------%
%run single simulation
    [localCompFlux,partialFlux,totVolFlux,exitFlag,output] = asyMemLocalSolve(sysInfo)
%-----------------------------------------------------------------------------------------------------------------%
```

Again, the *expSpec.m* takes the form

```
function [sysInfo] = expSpec_EXAMPLE()
%-----------------------------------------------------------------------------------------------------------------%
%specify single simulation
%mixture components, compositions, system parameters
  %SBAD1 Data
    sysInfo.memID = "SBAD1";  % polymer spec
%     %9 COMP M1
    sysInfo.mixID = ["TOL","MCH","MNP","DEC","NOC","IOC","TBB","TPB","ICT"];  % mixture spec
    sysInfo.yf = [0.171;0.281;0.0199;0.107;0.221;0.15;0.0217;0.0158;0.013];  % composition spec (must sum to 1)
    PuM = 40; %bar
    %3 COMP M1
%     sysInfo.mixID = ["TOL","IOC","ICT"];  % mixture spec
```

```
%     sysInfo.yf = [0.282;0.385;0.333];  % composition spec (must sum to 1)
%     PuM = 30; %bar
  %system specs
    sysInfo.n = length(sysInfo.yf);  % number of permeants
    sysInfo.Pu = PuM*0.9869;  % feed side pressure [bar]*0.9832 = [atm]
    sysInfo.Pd = 1;  % support side pressure (atm)
    sysInfo.pervapMode = 0; % %BETA(UNSTABLE) pervap capability -- yes if == 1 (assume downstream pressure = 0 bar)
    sysInfo.T = 295;  % system temperature (K)
    sysInfo.R = 82.05;  % gas constant (atm cm^3/ mol K)
    sysInfo.diffFit = 0; %see code below for diffusivity fitting
    sysInfo.crossDiffFudge = 0; %if = 0 then no specified cross-diffusivities D_ij will be used
    if sysInfo.crossDiffFudge == 1
        sysInfo.crossDiffSpecs = [1,2,1,3]; % e.g. [i,j] = D_ij^MS
        sysInfo.crossDiffVals = [0.1,0.5]; %um2/s
    end
    sysInfo.lowDiffErrorBar = 0; % if = 1 then diffusivities fit to low error sing comp will be used
%-------------------------------------------------------------------------------------------------------------%
end
```

The *solverModelSpec.m* file is exactly as described in subsubsection 4.3.2.

Multiple Mixture Simulations

This script has two forms. The first is based on cycling through many different single component permeation at various experimental conditions and/or fugacity models to generate figures similar to Figure 3.4. This allows for assessment of prediction capabilities of single component permeation based on diffusivities fit at a single transmembrane pressure. The *expSpec.m* function takes the form

```
function [sysInfo] = expSpec_EXAMPLE()
%-------------------------------------------------------------------------------------------------------------%
%specify experiment

%mixture components, compositions, system parameters
  %SBAD1 Data
%     sysInfo.memID = "SBAD1";  % polymer spec
%     sysInfo.mixID_OG = ["TOL","MCH","MNP","DEC","NOC","IOC","TBB","TPB","ICT"];  % mixture spec **NUMBER of entries
      should equal params.compID number listed in dataBank.m
%     sysInfo.memID = "PIM1";  % polymer spec
%     sysInfo.mixID_OG = ["TOL","HEP","PXY","OXY","ICT"];
    sysInfo.memID = "MATRIMID";
    sysInfo.mixID_OG = ["TOL","MES","TPB"];
    sysInfo.pressureVec = [10,20,30,40,50,60,70,80,90,100]; %bar -- specify pressures to loop over
    sysInfo.yf = 1;

  %system specs
    sysInfo.n = 1;  % number of permeants
    sysInfo.Pd = 1;  % support side pressure (atm)
    sysInfo.pervapMode = 0; % %BETA(UNSTABLE) pervap capability -- yes if == 1 (assume downstream pressure = 0 bar)
```

```
    sysInfo.T = 295;  % system temperature (K)
    sysInfo.R = 82.05;  % gas constant (atm cm^3/ mol K)
    sysInfo.diffFit = 0; %see code below for diffusivity fitting
    sysInfo.crossDiffFudge = 0; %if = 0 then no specified cross-diffusivities D_ij will be used
    if sysInfo.crossDiffFudge == 1
        sysInfo.crossDiffSpecs = [1,2,1,3]; % e.g. [i,j] = D_ij^MS
        sysInfo.crossDiffVals = [0.1,0.5]; %um2/s
    end
    sysInfo.lowDiffErrorBar = 0; % if = 1 then diffusivities fit to low error sing comp will be used
%-----------------------------------------------------------------------------------------------------------%
end
```

And the *solverModelSpec.m* looks like

```
function [sysInfo] = solverModelSpec_EXAMPLE()
%-----------------------------------------------------------------------------------------------------------%
%sorption/diffusion model and thermodynamic assumpstions spec
  %sorption membrane phase model
    sysInfo.modSpec = [1]; %specify which models to fit diffusivities for -- FH - 1 ; DSM - 2; FH-LM - 3

  %specify thermodynamic assumptions
    sysInfo.noThermoCoupling = 0; %==1 if want to predict using no thermodynamci coupling
    sysInfo.memPhaseModel_FicksOG = 0; %==1 if you want to predict using Fick's Law

  %specify diffusional relationships **set for single comp loop**
    sysInfo.diffModel = "NoCoupling";
    sysInfo.swlDiffModel = "none";
%-----------------------------------------------------------------------------------------------------------%
%-----------------------------------------------------------------------------------------------------------%
%numerical method sepcs **best specs for fitting code**
    sysInfo.numMethod = "NormShootAlg"; %classical shooting algorithm (SA) [Recommended]
  %solver specs
    sysInfo.solverSpec = "trust-region-dogleg"; %default, use for FUD
    sysInfo.iterDetail = 0; %0 = all main solver output will be suppressed
  %initial guess specs
    sysInfo.initGuessApprox = 0; %0 = use feed mol fraction vector and small total flux value (use if IVP approx fails)
    sysInfo.nodalGuessApprox = 0;
    sysInfo.customInitGuessApprox = 0; %set to match your experimental compostion number plus total flux as [n+1 by 1]
         vector for custom initial guess
  %sys of eqns spec
    sysInfo.currentStateLit_eqnSetup = 0; %default == 0 (not applicable to genreal simulations)
    sysInfo.noGammaFugacityODEs = 0; % ONLY USE FOR FULL MS (FICK == 0 and noThermo == 0)! option keep MS system interms
          of fugacity gradients and do not make chain rule trick to get gamma matrix
%-----------------------------------------------------------------------------------------------------------%
```

Finally, the single component permeation prediction loop script can be seen below

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% script:    singCompLoop_asyMemLocal.m                                                                    %
% Description: Asymmetric membrane local flux model/method solver for:    %
%                -single component permeation predictions loop            %
%                (See tutorial document for specific info regarding use)  %
%                                                                         %
% Copyright 2021 Georgia Tech Research Corporation                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%-------------------------------------------------------------------------------------------------------------------%
%add /SOURCE/ folder and /casADi-v3.5.5/ to MATLAB session search path
    soFolder = fullfile(pwd,'..','..','SOURCE');
    mainFolder = fullfile(pwd,'..','..');
    addpath(fullfile(soFolder,'casadi-v3.5.5'));
    addpath(genpath(soFolder));
    addpath(genpath(mainFolder));
%-------------------------------------------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------------------------------------------%
%specify simulation input file names and load data
    sysInfoEXP = expSpec_EXAMPLE(); %change function name to match input file name
    sysInfoSMS = solverModelSpec_EXAMPLE(); %change function name to match input file name
    sysInfo = cell2struct([struct2cell(sysInfoEXP);struct2cell(sysInfoSMS)],[fieldnames(sysInfoEXP);fieldnames(
        sysInfoSMS)]);
    sysInfo.dataBankName = "dataBank_EXAMPLE";
%-------------------------------------------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------------------------------------------%
%pure comp predictions loop
    pressureVec = sysInfo.pressureVec; %bar
    modSpec = sysInfo.modSpec;
    [allMPSCF,failedRuns] = asyMemSingCompEval(sysInfo,pressureVec,modSpec); %MPSCF = model, pressure, single component
        fluxes
%-------------------------------------------------------------------------------------------------------------------%
```

Please refer to the variable comments and structure to help with understanding. The outputs are matrices detailing predicted fluxes for each column to be at the given pressure defined from the **sysInfo.pressureVec**, each component defined from the **sysInfo.mixID_OG**, and **sysInfo.memID** string vector. Then the given matrix has n component fluxes for each sorption model. So if three sorption models are specified then the output matrices will contain $3n$ rows. Additionally, within the code folder, there is an example experimental data and simulated data file titled "Example_Data.xlsx" for provided example input files to see how the output matrices look like if the data is separated out.

The second prediction loop script is the most involved, and mimics the last loop presented. However, instead of varying the single component pressures, each defined mixture, sorption model, diffusion models, and thermodynamic assumptions are looped through to give a comprehensive set of predictive modeling data. Please refer to the examples within the *asyMemSim* code package for exact details. The code is very similar to the previous sections, except there are many embedded *FOR* loops to iterate over different defined mixtures, fugacity models, numerical methods, diffusion models, etc. There are many included

example files that were used in the data generation for Mathais et al., and Roos et al., and Weber et al. [14, 17, 40]

Since the output is a vary large matrix, organized simulated data is provided as a template to figure out how to trace the for loops to what the data is actually representing. See file "Example_Data_Error_Analysis.xlxs" and workbook titled "SIM DATA". The main script that this data was generated from was *mixLoop_asyMemLocal_Lively_Ronita_Pub.m*.

4.3.4 Complete Code File Index

Throughout the software package, a number of functions are used for various purposes. To be exact, there 3 main input file types as described in subsection 4.3.2, four main scripts as described in section subsection 4.3.3, and 63 helper functions that are mostly all essential for the entire piece of software to run properly. Certainly, for one model and one numerical method, less helper functions are needed. The following list provides the name and brief definition of what the function does for the interested user. Note that, some of the functions may not be used unless for advanced options that should only be for code expansion or exploration.

- asyMemDiffFit_RHS: RHS function for diffusivity fitting function
- asyMemDiffFitSolve: solver function for diffusivity script described in subsubsection 4.3.3
- asyMemGlobalRHS: (UNUSED) RHS function for old code that solved local flux problem within a idealized cocurrent flat-plate global module where the results were presented at AIChE 2020 Annual Meeting
- asyMemGlobalSolve: (UNUSED) solver function for problem outlined in previous function description

- asyMemLocal_bvp4c_BC_DAE: (UNUSED) boundary condition function for attempting to solve a ODEs with a built-in *MATLAB* boundary value problem solver that can only handle DAEss (it did not work)

- asyMemLocal_bvp4c_BC_ODE: (UNUSED): boundary condition function for ODEs BVP solver to test functionality compared to numerical methods presented in chapter 2

- asyMemLocal_IVP: function that evaluates the IVP for the ODEs/DAEs implementations outlined in section 2.2

- asyMemLocal_IVP_JAC_FH: (UNUSED) function for evaluating the Jacobian of the Flory-Huggins (FH) IVP that was for testing the speed-up when supplying the Jacobian to the integrator (never fully had time to test but initial tests showed not much of a speed-up)

- asyMemLocal_IVP_JAC_FHLM: (UNUSED) function for evaluating the Jacobian of the FH-LM IVP that was not fully tested or implemented

- asyMemLocal_IVP_pervap: (EXPERIMENTAL) function for solving the local transport when assuming pervaporation happening (Phase III pressure goes approaches 0 so $\phi_L \rightarrow 0$ since a vacuum is being pulled on the downstream membrane side)

- asyMemLocalALG_RHS_casADi: (UNUSED) function for solving system with casADi (was only tested with FH-LM when trying to solve local flux problem using multiple shooting point numerical method)

- asyMemLocalFoEu_QUICK: implementation of single step forward Euler method for genreating informed initial guesses as described in subsubsection 2.4.3

- asyMemLocalFoEu_QUICK_RHS: RHS for intialization strategy described in subsubsection 2.4.3

- asyMemLocalFUD: implementation of full FD method to solve the local flux problem outlined in chapter 2
- asyMemLocalFUD_RHS: RHS function for the above function description
- asyMemLocalFUD_RHS_pervap: (EXPERIMENTAL) function for solving the local transport when assuming pervaporation happening (Phase III pressure goes approaches 0 so $\phi_L \to 0$ since a vacuum is being pulled on the downstream membrane side) using the FUD numerical method
- asyMemLocalSA: implementation of shooting algorithm (SA) method to solve the local flux problem outlined in chapter 2
- asyMemLocalSA_IVP_TimeStep: function to manually integrate the IVP for the local flux problem
- asyMemLocalSA_IVP_TimeStep_RHS: RHS function to manually integrate the IVP for the local flux problem
- asyMemLocalSA_RHS: RHS function to sole the implementation of the SA to solve the local flux problem
- asyMemLocalSA_RHS_OG_eqn: (OUTDATED) RHS function to solve the implementation of the SA to solve the local flux problem
- asyMemLocalSA_RHS_pervap: RHS function for solving the local transport when assuming pervaporation happening (Phase III pressure goes approaches 0 so $\phi_L \to 0$ since a vacuum is being pulled on the downstream membrane side) using the SA numerical method
- asyMemLocalSolve: main function that all simulation scripts call to solve for initial phase equilibrium, initialize the parameter struct, and solve the local flux problem using the specified numerical method

- asyMemLoopDeLoop: (OUTDATED) main function that was the same as asyMemLocalSolve but with paramters hard-coded
- asyMemSingCompEval: function called when using the diffusivity fitting code that runs the single component simulations
- correlationEval: helper function that evaluates all fugacity model and diffusion model paramters (e.g. Vignes, Darken, Cohort, Hansen, etc.)
- correlationEval_casASi: (UNUSED) same as previous function description but written for casADi
- DAEevalDSM_RHS: DAEs RHS evaluation function for Dual-Mode Sorption (DMS) fugacity model
- DAEevalFH_LM_RHS: DAEs RHS evaluation function for FH fugacity model
- DAEevalFH_RHS: DAEs RHS evaluation function for FH-LM fugacity model
- dataBank: main paramter, component property, membrane property database function described in subsubsection 4.3.2
- diffFit_asyMemLocal: one of the main scripts described in subsubsection 4.3.3
- expSpec: experimental specification function described in subsubsection 4.3.2
- evalFH_LM_RHS_casADi: (UNUSED) function similar to DAEevalFH_LM, but written to be used with casADi
- matrixEvalB: function that evaluate the $\mathbf{B}$ within the Maxwell-Stefan model
- matrixEvalGammaB_DSM function that evaluates the $\mathbf{B}$ and $\mathbf{\Gamma}$ within the Maxwell-Stefan model when assuming the DMS fugacity model

- matrixEvalGammaB_FH: function that evaluates the $\mathbf{B}$ and $\mathbf{\Gamma}$ within the Maxwell-Stefan model when assuming the FH fugacity model
- matrixEvalGammaB_FH_LM: function that evaluates the $\mathbf{B}$ and $\mathbf{\Gamma}$ within the Maxwell-Stefan model when assuming the FHLM fugacity model
- matrixEvalGammaB_FH_LM_casADi: (UNUSED) function that evaluates the $\mathbf{B}$ and $\mathbf{\Gamma}$ within the Maxwell-Stefan model when assuming the FHLM fugacity model, but written to be used with casADi
- mixLoop_asyMemLocal: one of the main scripts detailed in subsubsection 4.3.3
- modelFH_functionMapping: (UNUSDE) funtion that tests teh feasible set that the FH fugacity model can predict based on different parameter sets
- outfun: (UNUSED) old function used to have custom solver output functions
- phi2fugPhaseEq_DSM: function to solve DMS fugacity model for an input of volume fractions and output fugacities
- phi2fugPhaseEq_FH_LM: function to solve FHLM fugacity model for an input of volume fractions and output fugacities
- phi2fugPhaseEq_FH_RHS: RHS function to solve FHLM fugacity model for an input of volume fractions and output fugacities
- singCompLoop_asyMemLocal: one of the main scripts to loop over single component simulations as described in subsubsection 4.3.3
- singMixSim_asyMemLocal: one of the main scripts to simulate the local flux problem for a single complex mixture as described in subsubsection 4.3.3
- solverModelSpec: function to specify the numerical method and model options as described in subsubsection 4.3.2

- vol2molFrac: (UNUSED) helper code to take volume fractions and convert to mole fractions
- vol2molFrac_casADi: (UNUSED) helper code to take volume fractions and convert to mole fractions but written for casADi
- vol2molFrac_RHS: (UNUSED) RHS for helper code to take volume fractions and convert to mole fractions
- y2evalFH_LM_RHS: function similar to DAEeval_FHLM
- y2phiPhaseEq_DSM: function to solve phase equilibrium for Phase I and Phase II assuming the DSM fugacity model
- y2phiPhaseEq_DSM_RHS: RHS function to solve phase equilibrium for Phase I and Phase II assuming the DSM fugacity model
- y2phiPhaseEq_FH: function to solve phase equilibrium for Phase I and Phase II assuming the FH fugacity model
- y2phiPhaseEq_FH_LM: function to solve phase equilibrium for Phase I and Phase II assuming the FHLM fugacity model
- y2phiPhaseEq_FH_LM_RHS: RHS function to solve phase equilibrium for Phase I and Phase II assuming the FHLM fugacity model
- y2phiPhaseEq_FH_RHS: RHS function to solve phase equilibrium for Phase I and Phase II assuming the FH fugacity model

4.4 *asyMemSim* Tutorial Examples For Use in *MATLAB*

The purpose of this section is to outline how to find the respective minimum working example files for each main script described in subsection 4.3.3. Since each main script shares

the same central database function, it can be found in the main folder with the name *dataBank_EXAMPLE.m*. This is not a requirement for the code to function properly since the database can be stored in any subfolder as long as a line of code is added to make sure the folder is in the *MATLAB* path (i.e. use the "addpath()" function). Documentation can be found here (*https://www.mathworks.com/help/matlab/ref/addpath.html*). Next, within each main script folder, one can find a respective *expSpec_EXAMPLE.m* and *solerModelSpec_EXAMPLE.m* files that will allow one to run the code without modification until necessary or warranted. Please use these files as templates for future applications. Again, all main scripts are ready to run as-is when the code is downloaded and the current version can be found here [99].

4.5 *asyMemSim* Custom Model Implementations

This section is written for advanced end-users that are interested in expanding this code for additional membrane materials, mixture components, fugacity models, diffusion models, and alternative modeling frameworks. The expansion and addition of these capabilities is fairly straightforward, but the subsequent subsections will describe exactly what functions to edit in order to get a custom addition working properly.

4.5.1 Additional Membrane Materials and Mixture Parameters

In order to add additional membrane materials and mixture parameters, the file that needs to be edited is *dataBank.m*. Within this file, there will be sections like the one shown below

```
function [params] = dataBank(sysInfo)
%-------------------------------------------------------------------------------------------------------------------%
%full parameter/property sets
    if contains(sysInfo.memID,'SBAD1')
      % Lively/Ronita's 9-comp data TOL/MCH/1MN/DCN/NOC/IOC/TBB/TPB/ICE/SBAD-1
        params.lmem = 0.3; % thickness of active membrane layer um
        params.compID = struct('TOL', 1, 'MCH', 2, 'MNP', 3, 'DEC', 4, 'NOC', 5, 'IOC', 6, 'TBB', 7, 'TPB', 8, 'ICT', 9)
             ;
        params.n = 9;
        params.Vs = [106.521;128.123;139.823;156.962;163.42;165.552;155.529;240.069;293.267;62326]; %cm3/mol
        params.HanSolParam = [18,1.4,2;16,0,1;20.6,0.8,4.7;18,0,0;15.5,0,0;14.1,0,0;17.4,0.1,1.1;18,0,0.6;16.3,0,0]; %[
             delD,delP,delH;...]
```

```
params.psat = [28.998;46.596;0.059;0.975;14.805;49.087;2.115;0.0352;0.0458];%torr
...
```

To add additional membrane materials, simply copy and paste the entire block from the example database and change the name from "SBAD1" to the new membrane material to be added. Then, one can fit and add fugacity model/diffusion parameters based on the defined mixture components. Additionally, to change the mixture components that permeate through the membrane material, simply update the *params.compID* to reflect the components of interest. The parameters *Vs, HanSolParam, n*, and *psat* will also need to be updated. Then, the fugacity model and diffusion model parameters will also need to be entered within the same order and size. Once that is done, the end-user can specify the desired *sysInfo.memID* and *sysInfo.mixID* in *expSpec.m*, which will then pull the required parameters from *dataBank.m*.

4.5.2 Membrane Fugacity Models

If the end-user wants to add a custom fugacity model, one will need first make sure the form of the fugacity model is in terms of volume fractions such that it is compatible with the implemented Maxwell-Stefan formulation of the local flux problem. If it is not, then please refer to subsection 4.5.5 first for tweaking the modeling framework basis. Once that is verified, one can add respective fugacity model parameters in *dataBank.m*, create a function to solve the phase equilibrium between Phase I/II (*y2phisPhaseEq_customModelName.m* and accompanying *y2phisPhaseEq_customModelName_RHS.m*, add a section of code to *asyMemLocalSolve.m* to get the required boundary conditions, add a section of code to evaluate the initial value problem function for the custom fugacity model in *asyMemLocal_IVP.m*, (if the fugacity model is implicit) create a *DAEevalCustomModelName_RHS.m* function, (if the fugacity model is explicit) create a *phis2yPhaseEq_customModelName_RHS.m*, add a section of code to *asyMemLocalSA_RHS.m* to solve the inner ODEs/DAEs properly, add a section of code to *asyMemSim* and create a *matrixEvalGammaB_customModelName.m*

function (the code to evaluate the $\mathbf{B}$ does not change, but the $\mathbf{\Gamma}$ evaluation will need to be updated). For each modification required, the next paragraphs will provide the specific details to add the custom fugacity model.

For the addition of custom fugacity model parameters, the *databank.m* needs to be edited as seen below (assuming the membrane material is SBAD1)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function:    dataBank(sysInfo)                                          %
% Description: Build parameter matricies based on system specifications.  %
% Input:       sysInfoExt - external struct defining simulation specs     %
%                           (memID, mixID, yf, n, lmem, Pu, Pd, T, R      %
%                            memPhaseModel, diffModel, swlDiffModel)      %
% Output:      params     - struct of system parameters                   %
%                           (compID, n, Vs, HanSolParam, psat, chis [FH   %
%                           or FH-LM], diffs, Ch & bs [FH-LM or DSM],     %
%                           ks [DSM], unitActPhis, Bffv, and all          %
%                           fields listed in sysInfo struct)              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [params] = dataBank(sysInfo)
%------------------------------------------------------------------------------------------------------------------%
%full parameter/property sets
    if contains(sysInfo.memID,'SBAD1')
      % Lively/Ronita's 9-comp data TOL/MCH/1MN/DCN/NOC/IOC/TBB/TPB/ICE/SBAD-1
        params.lmem = 0.3; % thickness of active membrane layer um
        params.compID = struct('TOL', 1, 'MCH', 2, 'MNP', 3, 'DEC', 4, 'NOC', 5, 'IOC', 6, 'TBB', 7, 'TPB', 8, 'ICT', 9)
             ;
        params.n = 9;
        params.Vs = [106.521;128.123;139.823;156.962;163.42;165.552;155.529;240.069;293.267;62326]; %cm3/mol
        params.HanSolParam = [18,1.4,2;16,0,1;20.6,0.8,4.7;18,0,0;15.5,0,0;14.1,0,0;17.4,0.1,1.1;18,0,0.6;16.3,0,0]; %[
             delD,delP,delH;...]
        params.psat = [28.998;46.596;0.059;0.975;14.805;49.087;2.115;0.0352;0.0458];%torr
        if contains(sysInfo.memPhaseModel,'F-H')
            params.chis_im = [0.871;1.672;0.705;2.783;1.163;3.049;1.648;2.5;3.130];
            if sysInfo.lowDiffErrorBar == 0
                params.diffs_im = [12.7;2.8617;2.8617;2.8617;2.8617;2.8617;2.8617;2.8617;2.8617]; %um2/s FH Exp Based
                     tweaked for single comp MS diffs
            elseif sysInfo.lowDiffErrorBar == 1
                params.diffs_im = [14.157;2.870;0.042;1.296;4.458;4.2057;0.975;0.069;0.536]; %Mixed Exp (LOW ERROR)
                     Based FH Tweaked to match single comp FH um2/s
            end
            if sysInfo.memPhaseModel_FicksOG == 1
                if sysInfo.lowDiffErrorBar == 0
                    params.diffs_im = [3.622;3.329;0.009;1.395;2.369;6.091;0.833;0.055;0.520]; %um2/s FICKS FH-BC
                         tweaked for single comp 20bar
                elseif sysInfo.lowDiffErrorBar == 1
                    params.diffs_im = [3.316;1.709;0.0076;1.085;1.838;3.6712;0.580;0.055;0.473]; %um2/s FICKS (LOW ERROR
                         ) FH-BC tweaked for single comp 20 bar
                end
            end
        elseif contains(sysInfo.memPhaseModel,'DSM')
        ...
```

From there, one simply needs to add an *elseif constains(sysInfo.memPhaseModel, 'custom-ModelName')* followed by all the respective parameters for the model (e.g. for FH, all χ_{im} must be specified for each component). Additionally, the diffusion parameters (Maxwell-Stefan, Ficks, low error bar if all are required) must be specified and fit for the respective custom fugacity model. Refer to subsubsection 4.3.3 for more details after all the other custom fugacity model modifications have been made. Lastly, the bottom of the *databank.m* function needs to have a line added to transform the string variable to a number for faster performance when calling the parameter struct. See code below for where to add it

```
%-------------------------------------------------------------------------------------------------------------------%
%carry over needed sysInfo to params
    params.T = sysInfo.T;
    params.R = sysInfo.R;
    params.mixID = sysInfo.mixID;
    if contains(sysInfo.memID,'SBAD1')
        params.memID = 1;
    elseif contains(sysInfo.memID,'PIM1')
        params.memID = 2;
    end
    params.compID = cell2struct(mat2cell(1:sysInfo.n,1,ones(sysInfo.n,1)),params.mixID,2);
    params.Pu = sysInfo.Pu;
    params.Pd = sysInfo.Pd;
    params.yf = sysInfo.yf;
    params.n = sysInfo.n;
    if contains(sysInfo.memPhaseModel,'F-H')
        params.memPhaseModel = 1;
    elseif contains(sysInfo.memPhaseModel,'DSM')
        params.memPhaseModel = 2;
    elseif contains(sysInfo.memPhaseModel,'FH-LM')
        params.memPhaseModel = 3;
    end
    if contains(sysInfo.diffModel,'NoCoupling')
        params.diffModel = 1;
    elseif contains(sysInfo.diffModel,'Vignes')
        params.diffModel = 2;
    elseif contains(sysInfo.diffModel,'Darken')
        params.diffModel = 3;
    elseif contains(sysInfo.diffModel,'Fudge')
        params.diffModel = 4;
    end
    if contains(sysInfo.swlDiffModel,'none')
        params.swlDiffModel = 1;
    elseif contains(sysInfo.swlDiffModel,'FFV')
        params.swlDiffModel = 2;
    elseif contains(sysInfo.swlDiffModel,'Avg-Diff')
        params.swlDiffModel = 3;
    end
    if contains(sysInfo.numMethod,'FullDis')
        params.numNodes = sysInfo.numNodes;
```

```
    elseif contains(sysInfo.numMethod,'MultShootAlg')
        params.numShootPoints = sysInfo.numShootPoints;
        params.casADi = sysInfo.casADi;
    end
    params.solverSpec = sysInfo.solverSpec;
    params.iterDetail = sysInfo.iterDetail;
    params.crossDiffFudge = sysInfo.crossDiffFudge;
    params.memPhaseModel_FicksOG = sysInfo.memPhaseModel_FicksOG;
    params.noThermoCoupling = sysInfo.noThermoCoupling;
    if sysInfo.crossDiffFudge == 1
        params.crossDiffSpecs = sysInfo.crossDiffSpecs;
        params.crossDiffVals = sysInfo.crossDiffVals;
    end
    params.pervapMode = sysInfo.pervapMode;
    params.currentStateLit_eqnSetup = sysInfo.currentStateLit_eqnSetup;
    params.noGammaFugacityODEs = sysInfo.noGammaFugacityODEs;
    params.nodalGuessApprox = sysInfo.nodalGuessApprox;
%-----------------------------------------------------------------------------------------------------------------%
end
```

Right after the *elseif contains(sysInfo.memPhaseModel,'FH-LM')*, a line should be copy/-pasted to have *elseif contains(sysInfo.memPhaseModel,'customModelName')*, followed by *params.memPhaseModel = 4;*.

For the phase equilibrium problem between Phase I/II, the functions *y2phisPhaseEq_FH.m* and *y2phisPhaseEq_FH_RHS.m* can be used as templates. As long as one if familiar with solving nonlinear equations in *MATLAB*, the same idea applies here. After that is completed, a short section of code needs to be added to actually solve for the boundary conditions in *asyMemLocalSolve.m* as seen below

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function:    asyMemLocalSolve(sysInfoExt)                               %
% Description: Solve boundary value problem for local flux model of       %
%                asymetric membrane layer using Maxwell-Stefan (MS)       %
%                coupled transport framework in terms of volume fractions %
%                that is useful for modeling polymeric materials.         %
% Input:       sysInfo       - external struct defining simulation specs  %
%                                (memID, mixID, yf, n, lmem, Pu, Pd, T, R %
%                                memPhaseModel, diffModel, swlDiffModel)  %
%                                *see descriptions below.                 %
% Output:      localCompFlux - n+1 dimensional vector of support layer    %
%                                compositions and total local mem flux    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [localCompFlux,partialFlux,totVolFlux,exitFlag,output,params] = asyMemLocalSolve(sysInfo)
%-----------------------------------------------------------------------------------------------------------------%
%assign/unpack parameters
    [params] = feval(sysInfo.dataBankName,sysInfo);
    n = params.n;
```

```
    if contains(sysInfo.memPhaseModel,'DSM')||contains(sysInfo.memPhaseModel,'FH-LM')
        bs = params.bs;
        Ch = params.Ch;
    end
    psat = params.psat;
    T = params.T;
    R = params.R;
    Vs = params.Vs;
    Pu = params.Pu;
    yf = params.yf;
    if contains(sysInfo.memPhaseModel,'DSM')
        ks = params.ks;
        params.chis = ones(n+1);
    end
    params.purFug = psat.*exp(Vs(1:n).*(Pu-psat*0.00131)/(R*T));
    params.chisDone = 0;
    params.diffsDone = 0;
    params.phiFeed = ones(n+1);
%-------------------------------------------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------------------------------------------%
%solve for feed conditions
    if params.memPhaseModel == 1
        phiFeed_FH_IS = y2phiPhaseEq_FH(params); %FH ideal solution
        phiFeed = phiFeed_FH_IS; %can change accordingly for IS or RM
    elseif params.memPhaseModel == 2
        phiFeed_DSM_IS = y2phiPhaseEq_DSM(params);
        phiFeed = phiFeed_DSM_IS; %can change accordingly for IS or RM
        params.chis = ones(n+1,n+1);
    elseif params.memPhaseModel == 3
        phiFeed_FH_LM_IS = y2phiPhaseEq_FH_LM(params);
        phiFeed = phiFeed_FH_LM_IS; %can change accordingly for IS or RM
    end
    fugFeed_IS = yf.*psat.*exp(Vs(1:n).*(Pu-psat*0.00131)/(R*T)); %0.00131 converts torr to bat
%     fugFeed_IS = yf; %if assuming fs = activity
    [chis,diffs] = correlationEval(phiFeed,params.diffs,params.chis,params);
    params.diffs = diffs;
    params.chis = chis;
    params.chisDone = 1;
    params.diffsDone = 1;
    params.phiFeed = phiFeed;
    params.fugFeed = fugFeed_IS;
%-------------------------------------------------------------------------------------------------------------------%
```

similar to the previous modifications in the *databank.m*, *elseif params.memPhaseModel == 4* needs to be added along with *phiFeed_customModelName_IS = y2phiPhaseEq_customModelName(params);* and *phiFeed = phiFeed_customModelName_IS;*.

To modify the IVP evaluation that ends up being the RHS equations for the integrator, it is similar to the modification of the database in terms of simply copy/pasting and adding an additional *elseif constains(sysInfo.memPhaseModel, 'customModelName')* line. See code

below for the specific section

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function:    asyMemLocalSA_IVP(z,stateVar,diffVar,localCompFlux,params) %
% Description: RHS function for solveing of initial value problem         %
% Input:       z             - spacial variable to integrate over (um)    %
%              stateVar      - (ODE, FH) n+1 dimensional vector of volume %
%                                fractions in membrane phase              %
%                              (DAE, FH-LM or DSM) 2*n+1 dimensinal vec   %
%                                of n+1 volume fractions and n fugacities %
%                                of membrane phase                        %
%              diffVar       - (ODE) diffVar = 0 (n/a)                    %
%                              (DAE) 2*n+1  dimensinal derivative vcector %
%                                of n+1 volume fractions and n fugacities %
%                                of membrane phase                        %
%              localCompFlux - n+1 dimensional vector of support layer    %
%                                compositions and total local mem flux    %
%              params        - struct of system parameters                %
%                                     (see dataBank function for specs)   %
% Output:      funIVP        - function value for integrator solver       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [funIVP] = asyMemLocal_IVP(z,stateVar,diffVar,localCompFlux,params)
%-------------------------------------------------------------------------------------------------------------------------%
%unpack parameters and modify as needed
    diffs = params.diffs;
    if params.memPhaseModel == 1 || params.memPhaseModel == 3
        chis = params.chis; %note chi_ji = chi_ij
    elseif params.memPhaseModel == 2
        chis = zeros(params.n+1);
    end
    n = params.n;
    HanSolParam = params.HanSolParam;
    T = params.T;
    R = params.R;
    psat = params.psat;
    Vs = params.Vs;
    Pu = params.Pu;
    funIVP = zeros(n+1,1);
    [chis,diffs] = correlationEval(stateVar,diffs,chis,params);
    params.diffs = diffs;
    params.chis = chis;
%-------------------------------------------------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------------------------------------------------%
%evaluate IVP residual functions
    if params.memPhaseModel == 1
      %solving for volume fluxes and mol fractions
        if params.memPhaseModel_FicksOG == 1
            B = matrixEvalB(stateVar,params);
            funIVP(1:n)=-B*(localCompFlux(1:n)...
                .*Vs(1:n).*localCompFlux(n+1));%./sum(Vs(1:n).*localCompFlux(1:n)))*1000/3600; %note correct convertsion
                      from LMH to UM3/UM2/s
            funIVP(n+1)=-sum(funIVP(1:n));
        elseif params.noGammaFugacityODEs == 0
            if params.noThermoCoupling == 1
                B = matrixEvalB(stateVar,params);
```

```
            funIVP(1:n)=-B*(localCompFlux(1:n)...
               .*Vs(1:n).*localCompFlux(n+1));%./sum(Vs(1:n).*localCompFlux(1:n)))*1000/3600; %note correct
                    convertsion from LMH to UM3/UM2/s
            funIVP(n+1)=-sum(funIVP(1:n));
        else
            %ODE
            [B,invGam] = matrixEvalGammaB_FH(stateVar,params);
            funIVP(1:n)=-invGam*B*(localCompFlux(1:n)...
  .*Vs(1:n).*localCompFlux(n+1));%./sum(Vs(1:n).*localCompFlux(1:n)))*1000/3600;
            funIVP(n+1)=-sum(funIVP(1:n));
        end
    else
        B = matrixEvalB(stateVar,params);
        funIVP(1:n) = -diffVar(n+2:end)./stateVar(n+2:end).*stateVar(1:n)-B*(localCompFlux(1:n)...
            .*Vs(1:n).*localCompFlux(n+1));%./sum(Vs(1:n).*localCompFlux(1:n)))*1000/3600; %note correct convertsion
                from LMH to UM3/UM2/s
        funIVP(n+1) = sum(diffVar(1:n+1));
        funIVP(n+2:n+1+n) = DAEevalFH_RHS(stateVar,params); %FH
    end
elseif params.memPhaseModel == 2
%solving for volume fluxes and mol fractions
    if params.memPhaseModel_FicksOG == 1
        B = matrixEvalB(stateVar,params);
        funIVP(1:n) = -diffVar(1:n)-B*(localCompFlux(1:n)...
            .*Vs(1:n).*localCompFlux(n+1));%./sum(Vs(1:n).*localCompFlux(1:n)))*1000/3600; %note correct convertsion
                from LMH to UM3/UM2/s
        funIVP(n+1) = sum(diffVar(1:n+1));
        funIVP(n+2:n+1+n) = DAEevalDSM_RHS(stateVar,params); %DSM
    elseif params.noGammaFugacityODEs == 0
        if params.noThermoCoupling == 1
            B = matrixEvalB(stateVar,params);
            funIVP(1:n) = -diffVar(1:n)-B*(localCompFlux(1:n)...
                .*Vs(1:n).*localCompFlux(n+1));%./sum(Vs(1:n).*localCompFlux(1:n)))*1000/3600; %note correct
                    convertsion from LMH to UM3/UM2/s
            funIVP(n+1) = sum(diffVar(1:n+1));
            funIVP(n+2:n+1+n) = DAEevalDSM_RHS(stateVar,params); %DSM
        else
            [B,invGam] = matrixEvalGammaB_DSM(stateVar,params);
            funIVP(1:n) = -diffVar(1:n)-invGam*B*(localCompFlux(1:n)...
               .*Vs(1:n).*localCompFlux(n+1));%./sum(Vs(1:n).*localCompFlux(1:n)))*1000/3600; %note correct
                    convertsion from LMH to UM3/UM2/s
            funIVP(n+1) = sum(diffVar(1:n+1));
            funIVP(n+2:n+1+n) = DAEevalDSM_RHS(stateVar,params); %DSM
        end
    else
        B = matrixEvalB(stateVar,params);
        funIVP(1:n) = -diffVar(n+2:end)./stateVar(n+2:end).*stateVar(1:n)-B*(localCompFlux(1:n)...
            .*Vs(1:n).*localCompFlux(n+1));%./sum(Vs(1:n).*localCompFlux(1:n)))*1000/3600; %note correct convertsion
                from LMH to UM3/UM2/s
        funIVP(n+1) = sum(diffVar(1:n+1));
        funIVP(n+2:n+1+n) = DAEevalDSM_RHS(stateVar,params); %FH-DSM
    end
elseif params.memPhaseModel == 3
```

From the above code snippet, one only needs to copy/paste the section and change the *el-*

seif params.memPhaseModel == 4 to the number that the custom fugacity model number was set to in the *databank.m* function. The only other change (if the fugacity model is explicit, as in the above snippit of code when *params.memPhaseModel == 1*) is to update the *matrixEvalGammaB_customModelName.m* in the *funIVP(1:n)* line of code. If the fugacity model is implicit (as in the above snippit when *params.memPhaseModel == 2*), then *matrixEvalGammaB_customModelName.m* needs to be updated as well as *DAEeval-CustomModelName_RHS.m* needs to be created and updated in *asyMemLocalIVP.m*. Any of the other DAEs evaluation RHS functions can be used as a template.

Next, the *matrixEvalBGamma_customModelName.m* must be created and can be made from the other fugacity model functions as a template. Finally, a line of code must be added in *asyMemLocalSA_RHS.m* to add the functionality for the custom fugacity model

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function:    asyMemLocalSA_RHS(phiFeed,localCompFlux,fugFeed,params)    %
% Description: RHS function for shooting algorithm of asyMem local flux.  %
% Input:       phiFeed       - n+1 dimensional vector of feed side        %
%                                    membrane phase volume fractions    %
%              localCompFlux - n+1 dimensional vector of support layer    %
%                                  compositions and total local mem flux    %
%              fugFeed       - n dimensional vector of penetrant          %
%                                  feed side fugacities (torr)          %
%              params        - struct of system parameters                %
%                                  (see dataBank function for specs)    %
% Output:      funRHS        - function value vector for nonlinear solver %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function   [funRHS] = asyMemLocalSA_RHS(phiFeed,localCompFlux,fugFeed,params)
%-------------------------------------------------------------------------------------------------------------------------%
%unpack parameters and modify as needed
    T = params.T;
    R = params.R;
    psat = params.psat;
    Vs = params.Vs;
    Pu = params.Pu;
    Pd = params.Pd;
    n = params.n;
    diffs = params.diffs;
    if params.memPhaseModel == 1 || params.memPhaseModel == 3
        chis = params.chis; %note chi_ji = chi_ij
    elseif params.memPhaseModel == 2
        chis = zeros(params.n+1);
    end

    if  params.swlDiffModel == 1
        params.Bffvtype = 'NA';
```

```
    elseif  any(params.Bffv == 0)
        params.Bffvtype = 'some0some1';
    else
        params.Bffvtype = 'cons1';
    end
%-------------------------------------------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------------------------------------------%
%IVP solve
    zspan=[0 params.lmem]; %um
    if params.memPhaseModel == 1 && params.noGammaFugacityODEs == 0
      % ode15s route
        diffVar = 0;
        funIVP = @(z,stateVar)asyMemLocal_IVP(z,stateVar,diffVar,localCompFlux,params); %solve IVP
        %funIVP = @(z,stateVar)asyMemLocal_IVP(z,stateVar,diffVar,[ypGuess;localCompFlux(n+1)],params); %solve IVP
        opt = odeset('RelTol',1e-5,'AbsTol',1e-7);%,'Jacobian',@(z,stateVar)asyMemLocal_IVP_JAC_FH(z,stateVar,
             localCompFlux,params));
        [z,stateVars] = ode15s(funIVP,zspan,phiFeed,opt);
        phiFinal = stateVars(end,1:n+1).';
        [params.chis,~] = correlationEval(phiFinal,diffs,chis,params);
        ypFinal = phis2yPhaseEq_FH_RHS(phiFinal,params);% FH based
    elseif params.memPhaseModel == 2 || params.memPhaseModel == 3 || (params.memPhaseModel == 1 && params.
          noGammaFugacityODEs == 1)
      % DAE ode15i route
        funIVP = @(z,W,D)asyMemLocal_IVP(z,W,D,localCompFlux,params); %solve IVP
        D0 = ones(n+1+n,1); %FH-DSM
        opt = odeset('InitialSlope', D0,'RelTol',1e-6,'AbsTol',1e-8);
        [W0_new,D0_new] = decic(funIVP,0,[phiFeed;fugFeed],[ones(n+1,1);zeros(n,1)],D0,zeros(1,n+1+n),opt); %FH-DSM
        opt = odeset(opt,'InitialSlope', D0_new,'RelTol',1e-3,'AbsTol',1e-5);
        [z,stateVars] = ode15i(funIVP,zspan,W0_new,D0_new,opt);
        fugFinal = (stateVars(end,n+2:end).');
         ypFinal = fugFinal./(exp(-Vs(1:n).*(Pu-Pd)/(R*T)))./params.purFug;
    end
%-------------------------------------------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------------------------------------------%
%fsolve residual functions
    funRHS(1:n) = localCompFlux(1:n)-ypFinal(1:n);
    funRHS(n+1) = 1-sum(ypFinal(1:n));


%-------------------------------------------------------------------------------------------------------------------%
```

If the custom fugacity model is explicit, then the *params.memPhaseModel == 4* can be appended as *params.memPhaseModel == 1 && params.noGammaFugacityODEs == 0* to *(params.memPhaseModel == 1 && params.noGammaFugacityODEs == 0)* || *(params.memPhaseModel == 4 && params.noGammaFugacityODEs == 0)*. Similarly, if the fugacity model is implicit, then a simple || *params.memPhaseModel == 4* can be added. If wanting to use a custom fugacity model with the FD numerical methods, the same would need to be done for *asyMemLocalFUD_RHS.m*.

4.5.3 Membrane Diffusion Models

To add custom membrane diffusion models for $Đ_{im}^{V}$, one simply needs to edit *dataBank.m* and *correlationEval.m*. First, the bottom of *dataBank.m* needs to have a line of code similar to the previous section when adding a custom fugactiy model

```
%-------------------------------------------------------------------------------------------------------------------%
%carry over needed sysInfo to params
    params.T = sysInfo.T;
    params.R = sysInfo.R;
    params.mixID = sysInfo.mixID;
    if contains(sysInfo.memID,'SBAD1')
        params.memID = 1;
    elseif contains(sysInfo.memID,'PIM1')
        params.memID = 2;
    end
    params.compID = cell2struct(mat2cell(1:sysInfo.n,1,ones(sysInfo.n,1)),params.mixID,2);
    params.Pu = sysInfo.Pu;
    params.Pd = sysInfo.Pd;
    params.yf = sysInfo.yf;
    params.n = sysInfo.n;
    if contains(sysInfo.memPhaseModel,'F-H')
        params.memPhaseModel = 1;
    elseif contains(sysInfo.memPhaseModel,'DSM')
        params.memPhaseModel = 2;
    elseif contains(sysInfo.memPhaseModel,'FH-LM')
        params.memPhaseModel = 3;
    end
    if contains(sysInfo.diffModel,'NoCoupling')
        params.diffModel = 1;
    elseif contains(sysInfo.diffModel,'Vignes')
        params.diffModel = 2;
    elseif contains(sysInfo.diffModel,'Darken')
        params.diffModel = 3;
    elseif contains(sysInfo.diffModel,'Fudge')
        params.diffModel = 4;
    end
    if contains(sysInfo.swlDiffModel,'none')
        params.swlDiffModel = 1;
    elseif contains(sysInfo.swlDiffModel,'FFV')
        params.swlDiffModel = 2;
    elseif contains(sysInfo.swlDiffModel,'Avg-Diff')
        params.swlDiffModel = 3;
    end
    if contains(sysInfo.numMethod,'FullDis')
        params.numNodes = sysInfo.numNodes;
    elseif contains(sysInfo.numMethod,'MultShootAlg')
        params.numShootPoints = sysInfo.numShootPoints;
        params.casADi = sysInfo.casADi;
    end
    params.solverSpec = sysInfo.solverSpec;
    params.iterDetail = sysInfo.iterDetail;
    params.crossDiffFudge = sysInfo.crossDiffFudge;
```

```
    params.memPhaseModel_FicksOG = sysInfo.memPhaseModel_FicksOG;
    params.noThermoCoupling = sysInfo.noThermoCoupling;
    if sysInfo.crossDiffFudge == 1
        params.crossDiffSpecs = sysInfo.crossDiffSpecs;
        params.crossDiffVals = sysInfo.crossDiffVals;
    end
    params.pervapMode = sysInfo.pervapMode;
    params.currentStateLit_eqnSetup = sysInfo.currentStateLit_eqnSetup;
    params.noGammaFugacityODEs = sysInfo.noGammaFugacityODEs;
    params.nodalGuessApprox = sysInfo.nodalGuessApprox;
%-------------------------------------------------------------------------------------------------------------%
end
```

The only change needed is to add *elseif contains(sysInfo.swlDiffModel,'customModelName')* and *params.swlDiffModel = 4;*. For the next part in *correlationEval.m* see the section of code below

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function:    correlationEval(stateVar,diffs,chis,params)                %
% Description: Evaluate diffusional and sorption correlations based on    %
%                stateVar values and component properties.                %
% Input:       stateVar    - (ODE, FH) n+1 dimensional vector of volume   %
%                              fractions in membrane phase                %
%                            (DAE, FH-LM or DSM) 2*n+1 dimensinal vec     %
%                              of n+1 volume fractions and n fugacities   %
%                              of membrane phase                          %
%              diffs       - n x n+1 dimensinal matrix of volume based MS %
%                              diffusivities (um^2/s)                     %
%              chis        - n+1 x n+1 dimensional matrix of FH or FH-LM  %
%                              chi parameters                             %
%              params      - struct of system parameters                  %
%                              (see dataBank function for specs)          %
% Output:      diffs, chis - evaluated matricies as defined above         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [chis,diffs] = correlationEval(stateVar,diffs,chis,params)
%-------------------------------------------------------------------------------------------------------------%
%unpack parameters
    yf = params.yf;
    Vs = params.Vs;
    R = params.R;
    T = params.T;
    HanSolParam = params.HanSolParam;
    n = params.n;
    pDiffs = params.diffs;
    unitActPhis = params.unitActPhis;
    B = params.Bffv;
%     molFracStateVar = vol2molFrac(stateVar,params);
    purFug = params.purFug;
    if length(stateVar) < 2*n+1
        stateVar = [stateVar;ones(n,1)];
    end
%     stateVar = stateVar./1000;
```

```
%-------------------------------------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------------------------------------%
%diffusivity modify for swelling model
    if params.swlDiffModel == 2 && params.diffsDone == 0
        for i = 1:n
              diffs(i,n+1) = pDiffs(i,n+1)*exp(B(i)*(1/unitActPhis(i)-1/abs((1-stateVar(n+1)))));
        end
    elseif params.swlDiffModel == 3
        for i = 1:n
    ...
```

From there, another line of code needs to be added *elseif params.swlDiffModel == 4* followed by the custom diffusion model equation. The equation must be explicitly dependant on the volume fractions but other methods and constant correlations can be explored.

4.5.4 Permeant Parameter Correlation Models

This section deals with different diffusion parameter model correlations such as Vignes or Darken for $Đ_{ij}^{V}$. Also, certain fugacity models may have parameter evaluations required such as the Hansen solubility parameter evaluation for χ_{ij}. Similar to the last section, *elseif contains(sysInfo.diffModel,'customModelName')* and *elseif params.diffModel == 5* needs to be added to *dataBank.m*. Then, the same procedure as the previous subsection needs to be applied except for *elseif params.diffModel == 5*.

For the fugacity model parameter evaluation, a simple line of code will need to be added to evaluate correlations based on the parameters and is highly model dependant. See code below of *correlationEval.m* for details of how the χ_{ij} was implemented

```
%-------------------------------------------------------------------------------------------------------------%
%cross-chi-parm calc
if params.chisDone == 0
    for m = 1:n
        for k = m+1:n
           % chis(m,k) = 2.5;
            chis(m,k) = ((Vs(m)*Vs(k))^0.5)/(R*T/9.86)*((HanSolParam(m,1)-HanSolParam(k,1))^2+...
                0.25*(HanSolParam(m,2)-HanSolParam(k,2))^2+0.25*(HanSolParam(m,3)-HanSolParam(k,3))^2); %atm/9.86 = MPa
            chis(k,m) = chis(m,k);
        end
    end
elseif params.chisDone == 1
    0;
end
%-------------------------------------------------------------------------------------------------------------%
```

As a final note, the *params.chisDone* variable was needed for the high component number simulations to ensure that some constant parameters were not evaluated every single iteration, but only when needed.

4.5.5 Membrane Transport Model Modifications

Lastly, for changing the basis of units (e.g. from volume fractions to mole fractions), *asyMemLocal_IVP.m* would need to be overhauled. Additionally, all inputs and phase equilibrium functions would need to be rewritten in terms of these new units. This would be a fairly lengthy and rigorous modification. However, with the complete state of the code, and the numerical methods implemented; the overall benefit would outweigh the effort in rewriting the entire code implementation for a simple change of units in the Maxwell-Stefan framework. The general equation structure would be preserved, but careful modifications would be necessary to ensure units properly line-up.

4.6 *asyMemSim* Extension to Process Simulation Environments

Now that the tool, *asyMemSim* has been presented in its entirety, the next logical step is to extend the capability of *asyMemSim* to be used in a process simulation environment. This is a natural next step in order to realize the full potential to design membrane cascades for any application in OSRO. As an example of use within a process simulation environment, the main process simulator will be AspenPlus, and the custom unit operation software will be AmsterChem's *MATLAB* Unit-op (https://www.amsterchem.com/matlabunitop.html). The level of comfort with AspenPlus will be assumed to be fairly high for the next steps of this section (e.g. setting up flowsheet, component specification, thermodynamic model specification, enabling CAPE-OPEN unit-op library through "manage libraries tab", etc.). Regardless, this custom unit operation package allows for a seamless transition of any *MATLAB* code to be used within any CAPE-OPEN compliant process simulator (see Figure 4.1). Before using this piece of software, some time should be spent installing, activating the li-

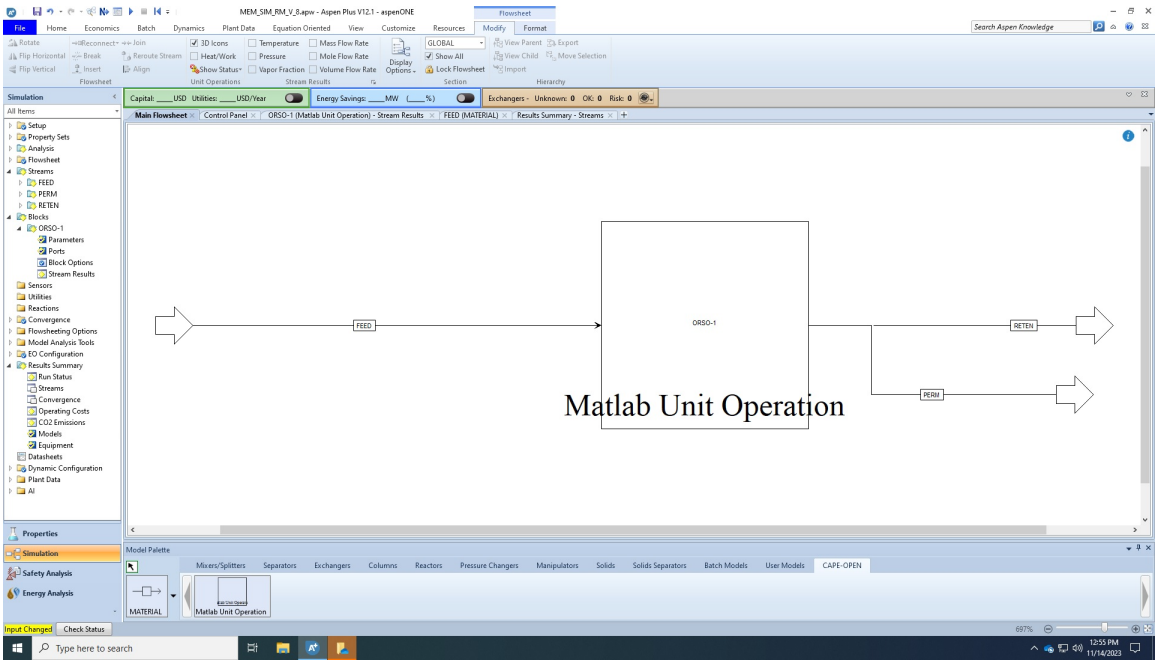


Figure 4.1: AspenPlus flowsheet with *MATLAB* CAPE-OPEN unit operation.

cense (the software is free for academic use, and a small fee is incurred for commercial use), and reading the great help documentation (https://www.amsterchem.com/matlabunitophelp.php). Note that, similar versions this software are also available for implementing custom unit operations into process simulation environments using tools created with Microsoft Excel, Python, and SciLab. Figure 4.2 shows how to first initialize the ports. For the membrane module, a feed, retentate, and permeate stream is shown for a simple cross-flow permeator set-up. This is the simplest global module membrane model such that the driving force is assumed constant across the entire membrane area. With this assumption, the total flux across the membrane multiplied by the total area, and permeate compositions fully specifies all streams for the membrane module. This assumption is great for a first-pass system design and lab-scale, cross-flow permeators. Future work will be focused on extending these local transport models to more complex global membrane module transport (as briefly described in Appendix C and section 6.3).

Next, Figure 4.3 shows the parameter tab shows the only specified parameter required–

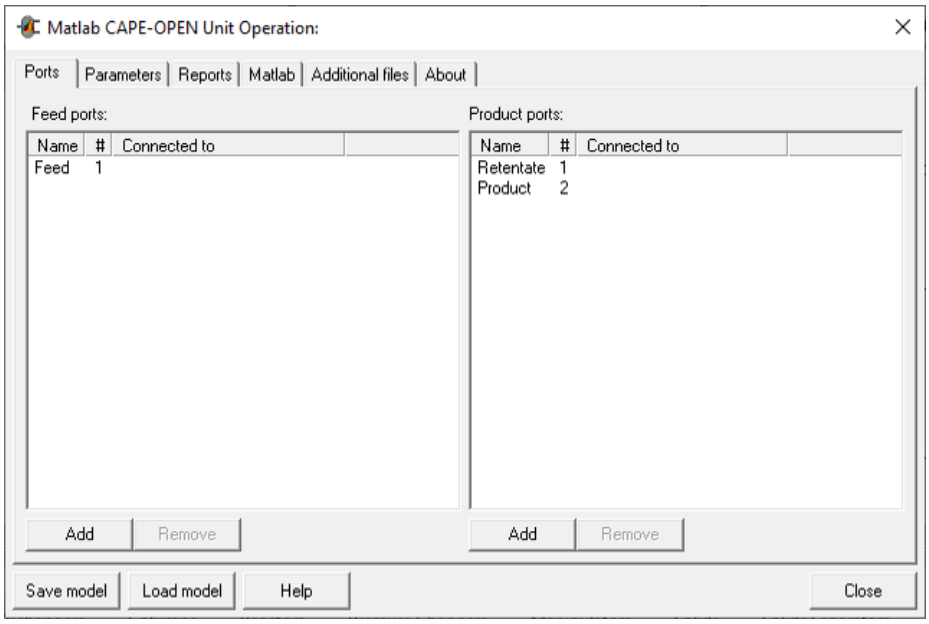


Figure 4.2: AmsterChem *MATLAB* CAPE-OPEN unit operation user interface – Ports tab.

the total membrane area (units of m^2). This allows for output stream calculations based on the permeate compositions, and total flux through the membrane based on soling the local transport problem described in chapter 2. The reports tab is unused in this case, so the *MATLAB* tab is the most important (as seen in Figure 4.4), and that is where the main single mixture simulation is found. Lastly, Figure 4.5 allows a place for all the supplemental functions to be added. Also, excel spreadsheets could be added if there are membranes that have many components and the "xlsread" *MATLAB* function is used. The main simulation script found in the *MATLAB* tab is exactly the single component simulation script described in subsubsection 4.3.3. The only difference is that the *expSpec.m* file is set partially through the feed stream specification and also in the *MATLAB* tab. The other difference is that the *solerModelSpec.m* is also manually set within the *MATLAB* tab as well to allowe for a single specification file. The *dataBank.m* function is the same as described in subsubsection 4.3.2, and can be found in the Additional files tab.

The current implementation is based in a flowsheet that is proprietary to ExxonMobil. However, the custom unit operations can be saved as a separate file with a ".MUM" file extension, and can be loaded within any AspenPlus flowsheet that has the *MATLAB* custom

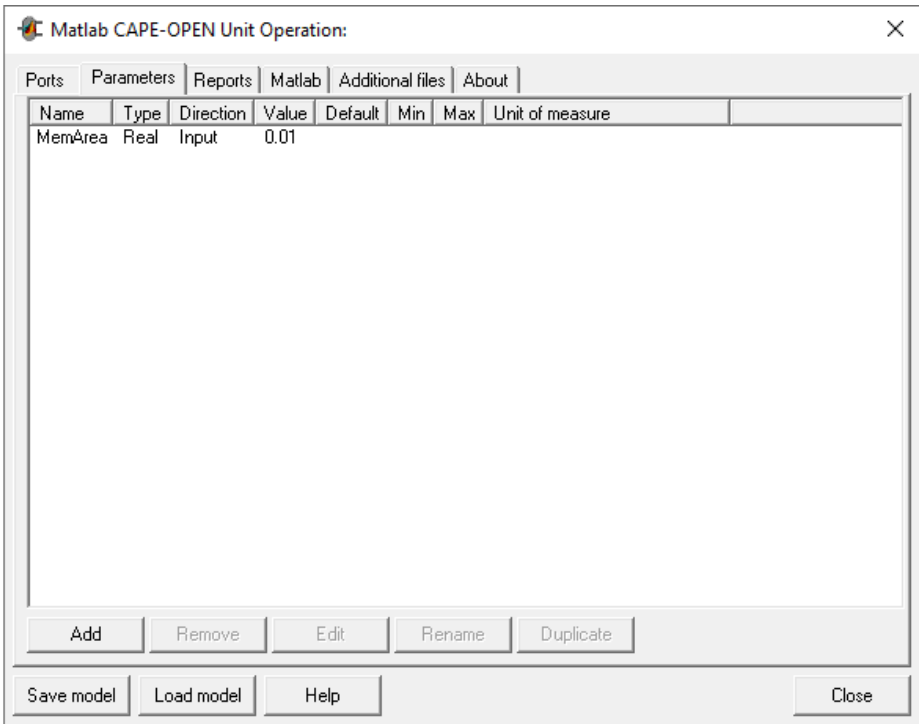


Figure 4.3: AmsterChem *MATLAB* CAPE-OPEN unit operation user interface – Parameters tab.

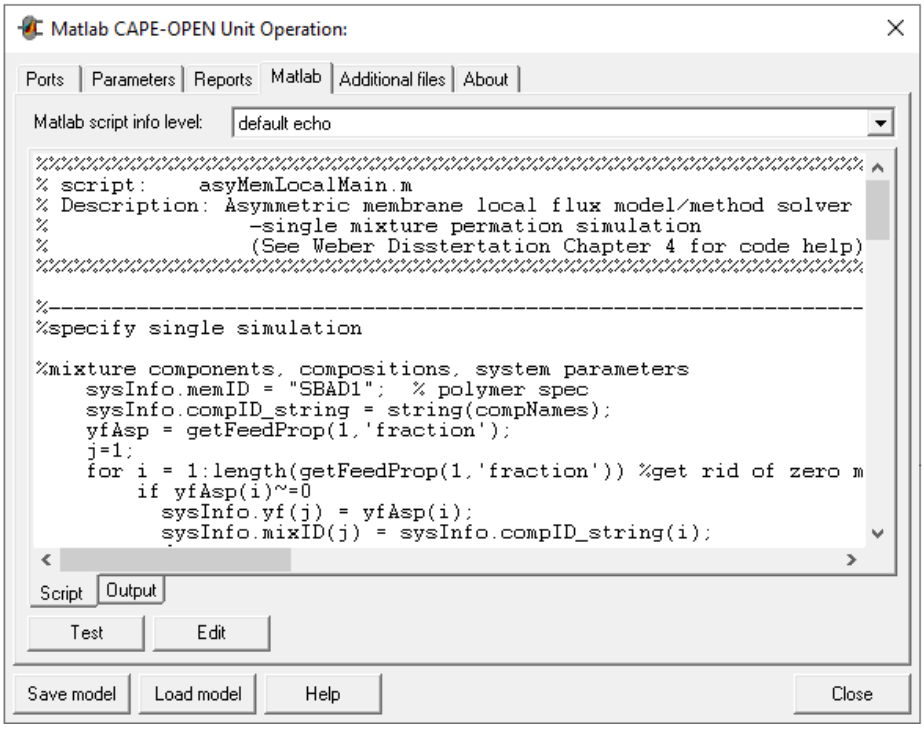


Figure 4.4: AmsterChem *MATLAB* CAPE-OPEN unit operation user interface – *MATLAB* tab.

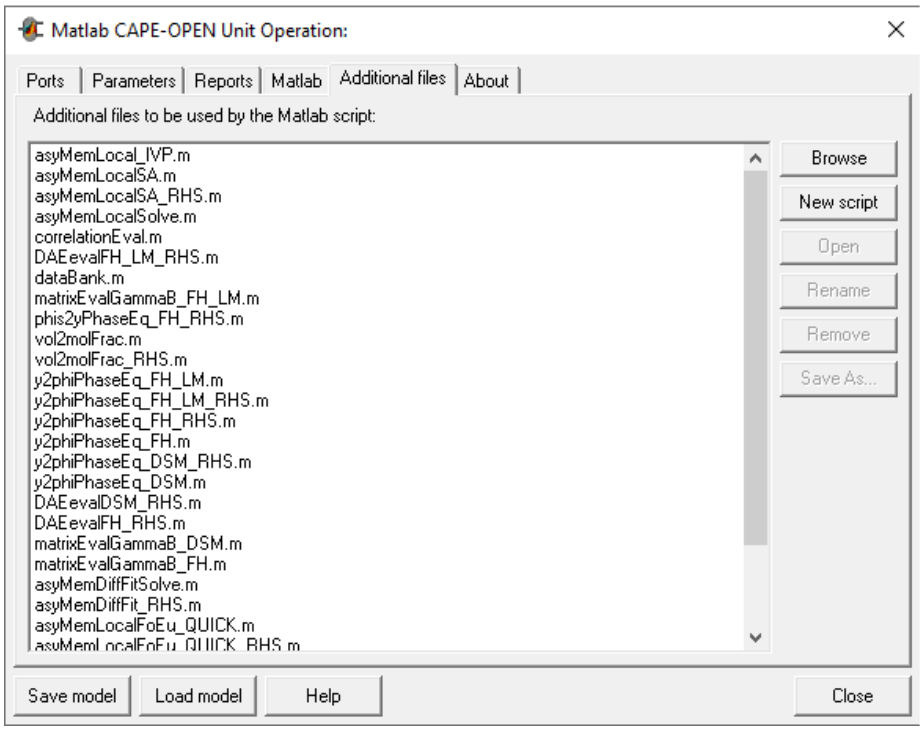


Figure 4.5: AmsterChem *MATLAB* CAPE-OPEN unit operation user interface – Additional files tab.

CAPE-OPEN unit op installed. The model file can be found at the respective GitHub citation [99].

4.7 Conclusions

Throughout this chapter, the software package *asyMemSim* was introduced and comprehensively described such that any future user can understand this code from the inside out. In particular, the simulation problem description, code standards, input files descriptions, modeling/simulation capabilities, code file index, tutorial files, various custom code modifications, and the natural extension of the software package was described in full detail. Along with that, code snippets and screenshots were included to help the end-user as much as possible. Using this chapter as a solid set of software documentation, *asyMemSim* will be able to be used and added to for many years to come. In doing so, realistic process designs of industrial membrane processes utilizing complex mixtures can be constructed.

CHAPTER 5
MODELING AND SIMULATION OF ELECTRIFIED MEMBRANE PROCESSES

5.1 Introduction

This chapter presents preliminary results on control strategies and process design of electrodialysis processes for nutrient recovery from wastewater. While these results are unpublished, these initial process designs will be refined, optimized, and made into a solid future contribution towards a modular nutrient recovery process. Additionally, a succinct modeling and simulation framework is shown for these processes based on either: steady-state (1D and 0D approximation), batch, or dynamic operation modes. The software is also available for use in *MATLAB*, and the steady-state mode has the capability to be used in a process simulation environment for proper process design. This work is supported by the Center for Advancing Sustainable Fertilizer Production (CASFER), *https://www.casfer.us/*, NSF Award#: 2133576.

Looking towards motivation for the nutrient recovery problem, of the total applied fertilizer to farms, 80% of that is washed into our environments [100]. This causes a slough of environmental problems (e.g. algae blooms) that have large health and socioeconomic impacts [101]. Since this agricultural runoff is in the form of nitrogen and phosphorous based fertilizer, the target nutrients to be recovered are nitrogen and phosphorous (N/P) for agricultural production. The end goal of this 10-year engineering research center is to foster a circular N/P economy. In addition to agricultural runoff, there are many different industrial wastewater streams with theses nutrients available in non-negligible concentrations including: municipal wastewater, concentrated animal farming operations, steam electric power generation, textile industrial effluent, ion-exchange brine, landfill leachate, agricultural runoff, and fertilizer production effluent [102, 103, 104, 105, 106, 107]. Given

that, the target waste streams this nutrient recovery process is focused on are those from WWTPs and CAFOs. These streams have dilute concentrations of nutrients, but massive throughput available that make them ideal economical and environmental targets for nutrient recovery [9]. Additionally, other streams can contain impurities that make it hard for biological use, while those from WWTPs and CAFOs are already biologically sourced and safe for biological-use once treated for bio-hazardous contaminants. Since WWTPs are readily known, CAFOs may not be as commonly known. These operations are used for meat/dairy/egg production raising animals such as poultry, swine, and beef for a profit such that the concentrations of animals on a given property is high. This creates a large amount of organic waste that needs to be treated on-site.

For the last few decades, nutrient recovery has been applied on a commercial scale (as seen in Table 5.1 and Table 5.2). However, the scale has been restricted to a few key successful companies, and the technology mainly relies on traditional separation methods that have major drawbacks. Additionally, WWTPs and CAFOs have been treating these streams for N/P using biological methods for many decades. The issue is that these nutrients are converted to either gaseous nitrogen, lost to the environment in a nonviable product form, or remain untreated (which pollutes the environment). A key difference is that these industries have been utilizing treatment rather than recovery. Of the commercial installations that have been marketing nutrient recovery, the traditional technologies are: air stripping, crystallisation, thermal distillation, biological digestion, or pressure-based membrane processes. Each of these processes bring about some real disadvantages such as high energy requirements, production of greenhouse gases, constant addition of chemicals required, or large process footprints that do not allow for modular system installations [108, 109, 13, 110, 111, 112].

Electrodialysis is a promising technology that has already seen success in the desalination industry [113]. Using this key body of knowledge and experience gained with that mature application, ED can be applied to nutrient recovery by using the same approach. The

difference is that, instead of considering the concentrated brine stream as waste, ED applied to nutrient recovery has the potential valorize the concentrated waste to yield a total of two product stream. The first is the concentrated salt stream that can be marketed as a fertilizer product. The second is a dilute steam that can be treated to a level of safe environmental impact such that it is considered treated water for discharge to waterways or recycled on-site of WWTPs/CAFOs. These systems have been applied vastly in the literature on nutrient recovery at the lab scale or small pilot-plant testing [45, 114, 115, 116, 117]. However, there is a void in the literature for a systems level design and optimization of these systems for nutrient recovery on an industrial scale. The main challenges hindering the ability to do so is the existence of a user-friendly ED unit operation model that can be used within a process simulation environment. Even though there are a number of current tools available (as discussed subsection 1.4.2), the tools either quick user-friendly design implementation/iteration compatibility (as with WaterTAP, *https://www.nawihub.org/knowledge/watertap/*), are not available for open-source use (Capione et al. 2019), and/or do have the required modeling and simulation fidelity required for detailed ED process design for nutrient recovery (as with QSDsan, *https://qsdsan.com/*).

Hence, this work describes a set of modeling and simulation contributions for different ED operation modes that is compatible and easily translatable to a user-friendly tool for use in process simulation environments. Namely, we provide a steady-state, batch, and dynamic ED simulation code. In addition to that the steady-state software is available for use within AspenPlus (a standard process simulation environment for chemical engineers). With these tools, this chapter provides preliminary work towards: a base case scenario for an average-sized CAFO that will enable level comparisons of current technologies with novel CASFER developed technologies, control strategies given realistic nutrient recovery process disturbances, and a ED cascade process design that shows a proof-of-concept process that could be implemented to recover recycle quality water along with a 5.6 wt% nitrogen-based fertilizer product. With these preliminary contributions, a step forward can

be taken to solidify the CASFER vision of installing a modular process that can recover a 10 wt% nitrogen fertilizer product to realize a N/P circular economy. By doing so, our society can shift towards scaling-back the need for energy and capital intensive processes that are unsustainable, and detrimental to our environments.

Table 5.1: Current Nutrient Recovery Commercial Installations

Company Name	**Technology**	**Company Footprint**	**Feedstock(s)**	**Product(s)**	**Reference**
Bion Env. Tech.	biological, thermal, mechanical	Since 1987, 4 projects listed	CAFO cattle manure waste streams	Clean water, natural gas, fertalizer	[118]
Ductor	biological	Since 2009, first commercial plant installed 2019	poultry waste	biogas, biofertalizer	[119]
EnviroKure	biological	Since 2011, upcycles millions of tons of chicken manure annually	chicken manure	fertilizer	[120]
sistema.bio	biological	Since 2010, 75k+ digesters installed	cow, swine, poultry manure	biogas, biofertalizer	[121]
CNP Cycles (AirPrex/LysoPhos)	crystalizer, thermal hydrolysis	Since 2011, 5 AirPrex installations, 3 AirPrex licensed installations, and 2 LysoPhos installations	WWTP streams	struvite	[122]
NuReSys	crystalizer and pH air stripping	Since 2011, 7 commercial systems installed	Industrial/Municipal waste streams (centrate/digestate/hybrid)	struvite	[123]
Ostera	crystalizer	Since 2005, 23 commercial installations worldwide	WWTP streams	stuvite	[124]
N2 Applied	electrolyzer	Since 2010, 10 projects installed	animal manure, air, electricity	nitrogen fertilizer	[125]
ReMo Energy Inc.	electrolyzer	Since 2020, MOU signed with Trammo	Air, water, renewable energy	nitrogen fertilizer	[126]
Sedron Tech.	thermal	Since 2017, successful installation handling 15.6 gallons of manure per year	WW biosolids, WW sidestreams, Dairy waste, LCFS digestate, Raw septage	Clean water, dry fertilizer, aqueous ammonia	[127]

Table 5.2: Current Nutrient Recovery Commercial Installations (continued)

Company Name	Technology	Company Footprint	Feedstock(s)	Product(s)	Reference
Digested Organics	membrane filtration	Since 2013, 20+ commercial installations	manure, digestates, food/beverage, industrial/-municipal, landfill	biogas, biofertalizer	[128]
Organics Group	thermal air stripping	Since 1992, unknown number of installations	Landfill leachate	ammonia contaminated stream	[129]
Byoflex	pH air stripping	Since 2011, 32 listed projects	WWTP, Manure, food waste	Ammonia-rich streams	[130]
Hansa Eng.	pH air stripping	Since 1954, uses RVT Process Equipment technology	ammonia contaminated stream	Ammonia-rich streams	[131]
Heil Eng. Process Equipment	pH air stripping	Since 1929, general air stripper vendor	ammonia contaminated stream	Ammonia-rich streams	[132]
Indusco Env.	pH air stripping	Since mid 1970s, general ammonia air stripper vendor	ammonia contaminated stream	Ammonia-rich streams	[133]
Mach Eng.	pH air stripping	Since 2005, unknown number of installations (ammonia recovery equipment rental)	WWTP, energy applications, utility applications	Ammonia-rich streams	[134]
Nijhuis Industries	pH air stripping	Since 1904, general ammonia stripper vendor	digestate	Ammonia-rich streams	[135]
RVT Process Equipment	pH air stripping	Since 1976, main equipment manufacturer	WWTP streams	Ammonia-rich streams	[136]
Task Env. Eng.	pH air stripping	Since 1988, main equipment manufacturer	WWTP streams	Ammonia-rich streams	[137]
Burnham RNG	unknown	Since 2021, 1 public partnership	WWTP streams	water, RNG, fertilizer	[138]

5.2 Theory and Background

In this section, a modeling and simulation framework for electrodialysis is presented for a number of process operation modes. The modes include: steady-state, batch, and dynamic operation. For the steady-state model, a zero-dimensional approximation is presented (in addition to a one-dimensional boundary value problem model) that assumes a constant inlet and constant ED module channel concentrations, current density, and channel volumetric flowrates. This approximation is also used to reduce the partial differential equation systems in the batch and dynamic operation mode to systems of DAEs. The models derived below are for a single cell-pair, and adapted from Campione et al. [1, 139]. The only difference in the current implementation is that the activity coefficient in the dilute/concentrate channels, $\gamma_i^{\text{int,IEM}}$, is assumed to be unity (as well as the osmotic coefficient in the dilute and concentrate channel, Φ_i^{IEM}). Additionally, the equivalent conductivity, $\Lambda_{\text{SOL}}(x)$ is fixed. These expressions can easily be adjusted and accounted for to reflect non-ideal systems with the addition of simple equations found in the Appendix of Campione et al. [1].

5.2.1 Steady-State Electrodialysis Model

First, in order to properly simulate an ED unit operation, the complexity of model must be chosen. Here, a simplified single salt model will be assumed. For later iterations of the simulation framework that include multiple salt solutions and other complex molecules, a Nernst-Plank model should be used that is similar to the Maxwell-Stefan model. The difference is the former takes into account electrical potential gradients while neglecting thermodynamic interactions (the Nernst-Plank model assumes a co-existing concentration gradient rather than an activity gradient). The main model to be listed below is adapted from Campione et al. [1]. Additionally, Figure 1 from Campione et al. shows a great schematic of the ED system set-up to be simulated [1].

Modeling Steady-State ED

This derivation will focus on a single cell pair as described in the previous section. The first piece of the model is to provide mass balances for the dilute and concentrate channels. The process is assumed to be at steady-state (unchanging with respect to time), and the main dynamics to model are the concentration and flow rate changes along the channels (changing with respect to the channel length variable, x). The components to be modeled are the single salt (in terms of concentration), and the water (in terms of a volumetric flow rate). For the concentrate and dilute channels, the equations for each component can be written as

$$\frac{d\left(Q_D(x)C_D(x)\right)}{dx} = -bJ_{\text{tot}}(x), \tag{5.1}$$

$$\frac{d\left(Q_C(x)C_C(x)\right)}{dx} = bJ_{\text{tot}}(x), \tag{5.2}$$

$$\frac{dQ_D(x)}{dx} = -bq_w(x), \tag{5.3}$$

$$\frac{dQ_C(x)}{dx} = bq_w(x). \tag{5.4}$$

The next piece is the constitutive equations that describe the mechanisms of transport across the IEM. The different mechanisms for salt transport are either a concentration and/or a electrical potential gradient. Additionally, the water can transport via an osmotic

or electroosmotic gradient. Those equations take the form

$$J_{\text{tot}}(x) = J_{\text{cond}}(x) + J_{\text{diff}}^{\text{AEM}}(x) + J_{\text{diff}}^{\text{CEM}}(x), \tag{5.5}$$

$$J_{\text{cond}}(x) = [t_{\text{CEM}}^{\text{counter}} - (1 - t_{\text{AEM}}^{\text{counter}})] \frac{i(x)}{F}, \tag{5.6}$$

$$J_{\text{diff}}^{\text{IEM}}(x) = -\frac{D^{\text{IEM}}}{\delta^{\text{IEM}}} \left(C_C(x) - C_D(x)\right), \tag{5.7}$$

$$q_w(x) = q_{\text{eosm}}(x) + q_{\text{osm}}(x), \tag{5.8}$$

$$q_{\text{eosm}}(x) = \frac{w J_{\text{tot}}(x) M_w}{\rho_w}, \tag{5.9}$$

$$q_{\text{osm}}^{\text{IEM}}(x) = L_p^{\text{IEM}} \left[vRT \left(\Phi_C^{\text{IEM}} C_C(x) - \Phi_D^{\text{IEM}} C_D(x)\right)\right], \tag{5.10}$$

where IEM $= \{\text{CEM}, \text{AEM}\}$. The final piece of the model relates the electrical constraints to the constitutive expressions. Those equations manifest as

$$V_{cp} = \eta(x) + R_{\text{tot}}(x) i(x), \tag{5.11}$$

$$V_{\text{tot}} = \frac{R_{\text{blank}} I}{A} + \sum_{i=1}^{N_{cp}} V_{cp,i}, \tag{5.12}$$

$$R_{\text{tot}}(x) = R_{\text{CEM}}(x) + R_{\text{AEM}}(x) + R_C(x) + R_D(x), \tag{5.13}$$

$$R_{\text{SOL}}(x) = f_{s,\text{SOL}} \frac{\delta_{\text{SOL}}}{\Lambda_{\text{SOL}}(x) C_{\text{SOL}}(x)}, \tag{5.14}$$

$$\eta(x) = \eta_{\text{CEM}}(x) + \eta_{\text{AEM}}(x), \tag{5.15}$$

$$\eta_{\text{IEM}}(x) = \alpha_{\text{IEM}} \frac{R_G T}{F} \ln\left[\frac{\gamma_C(x) C_C(x)}{\gamma_D(x) C_D(x)}\right], \tag{5.16}$$

where SOL $= \{\text{C}, \text{D}\}$ for either the concentrate or dilute channel. A few useful process metrics are specific energy consumption per mass of salt processes $\left(E_{\text{spec}}^{\text{salt}}\right)$, and the pro-

ductivity of salt produced (J_p)

$$E_{\text{spec}}^{\text{salt}} = \frac{\sum_{i=1}^{N_s} P_i}{C_{D,N_s}^{\text{IN}} Q_{D,N_s}^{\text{IN,tot}} - C_{D,N_s}^{\text{OUT}} Q_{D,N_s}^{\text{OUT,tot}}}, \tag{5.17}$$

$$J_p = \frac{C_{C,N_s}^{\text{OUT}} Q_{C,N_s}^{\text{OUT,tot}}}{2AN_{cp}}. \tag{5.18}$$

With these equations, the complete model is specified, and the next part deals with how to simulate the process. As a quick reference, Figure 5.1 shows the full set of equation required to simulate the ED module cell-pair at steady-state.

Simulation of Steady-State ED

Continuing on with the general theme of this dissertation, the model is one very important problem to understand. However, the actual implementation to successfully simulate the model is another problem entirely, and is not found as much within the academic membrane literature. From the previous section, (Equation 5.1)-(Equation 5.16) creates a system of DAEs. Moreover, the fact that the total current is required within the system of DAEs through (Equation 5.12) creates a 2-point boundary value problem. In order to solve for this system of equations, a guess of the total current, I, must be supplied to the DAEs solver. In order to calculate this total current at the end of the integration, an additional ordinary differential equation must be added

$$\frac{dI}{dx} = i(x)b. \tag{5.19}$$

The above equation should be initialized to $I = 0$ at $x = 0$, and then the total current will be found at $x = L$. With that value, the guess can be updated, and a simple outer loop solver could be coded up to do this automatically. However, compared to the 2-point boundary value problem outlined in chapter 2, the observed convergence behavior here is much simpler, and can be done manually in one or two iterations. With that in mind,

the simplified ED unit operation can be implemented fairly easily in any scientific coding language. One other tweak to the model that should be listed is whether the user would like to specify total voltage, or total power. The model listed in the previous section is already set-up to specify total voltage, but if the user would like to specify the total power, this simple equation should be substituted for the total voltage in the above model

$$V_{\text{tot}} = \frac{P_{\text{tot}}}{I}. \tag{5.20}$$

This completes the full modeling and simulation of an ED process. The next section deals with available software to design such processes.

Zero-dimensional Approximation

To reduce the steady-state ED DAEs problem to that of simply algebraic equations, (Equation 5.1) through (Equation 5.4) can be modified by evaluating the total salt and water flux, J_{tot} and q_w, respectively, at average channel concentrations and volumetric flowrates based on the inlet and outlet conditions. This average flux model reduces the steady-state ED DAEs model to a zero-dimensional model that is a system of nonlinear algebraic equations. This numerical approach is analogous to that presented in subsection 2.3.2. Figure 5.2 shows the final system of equations.

5.2.2 Batch Electrodialysis Model

In order to simulate lab-scale experimental set-ups, a batch operation that includes differential equations for the tanks will be derived below. Figure 8 from Campione et al. show this process arrangement in great detail [1]. This operation mode assumes the temporal dynamics of the tanks are on a much longer time scale than the temporal dynamics of the ED module such that the 0D-ED model from the previous subsection will be assumed. Without this assumption, the system of equations would be that of PDAEs, which is computation-

Conservation Equations

$$\frac{d\left(Q_D(x)C_D(x)\right)}{dx} = -bJ_{\text{tot}}(x)$$

$$\frac{d\left(Q_C(x)C_C(x)\right)}{dx} = bJ_{\text{tot}}(x)$$

$$\frac{dQ_D(x)}{dx} = -bq_w(x)$$

$$\frac{dQ_C(x)}{dx} = bq_w(x)$$

Constitutive Equations

$$J_{\text{tot}}(x) = J_{\text{cond}}(x) + J_{\text{diff}}^{\text{AEM}}(x) + J_{\text{diff}}^{\text{CEM}}(x)$$

$$J_{\text{cond}}(x) = \left[t_{\text{CEM}}^{\text{counter}} - (1 - t_{\text{AEM}}^{\text{counter}})\right]\frac{i(x)}{F}$$

$$J_{\text{diff}}^{\text{IEM}}(x) = -\frac{D^{\text{IEM}}}{\delta^{\text{IEM}}}\left(C_C(x) - C_D(x)\right)$$

$$q_w(x) = q_{\text{eosm}}(x) + q_{\text{osm}}(x)^{\text{AEM}} + q_{\text{osm}}(x)^{\text{CEM}}$$

$$q_{\text{eosm}}(x) = \frac{wJ_{\text{tot}}(x)M_w}{\rho_w}$$

$$q_{\text{osm}}^{\text{IEM}}(x) = L_p^{\text{IEM}}\left[vR_GT\left(\Phi_C^{\text{IEM}}C_C(x) - \Phi_D^{\text{IEM}}C_D(x)\right)\right]$$

Electrical Constraints

$$V_{cp} = \eta(x) + R_{\text{tot}}(x)i(x)$$

$$V_{\text{tot}} = \frac{R_{\text{blank}}I}{A} + \sum_{i=1}^{N_{cp}} V_{cp,i}$$

$$R_{\text{tot}}(x) = R_{\text{CEM}}(x) + R_{\text{AEM}}(x) + R_C(x) + R_D(x)$$

$$R_{\text{SOL}}(x) = f_{s,\text{SOL}}\frac{\delta_{\text{SOL}}}{\Lambda_{\text{SOL}}(x)C_{\text{SOL}}(x)}$$

$$\eta(x) = \eta_{\text{CEM}}(x) + \eta_{\text{AEM}}(x)$$

$$\eta_{\text{IEM}}(x) = \alpha_{\text{IEM}}\frac{R_GT}{F}\ln\left[\frac{\gamma_C(x)C_C(x)}{\gamma_D(x)C_D(x)}\right]$$

Figure 5.1: Full set of equations to simulate an electrodialysis module cell-pair at steady-state.

Conservation Equations

$$\frac{\left(Q_D^{\text{OUT}} C_D^{\text{OUT}} - Q_D^{\text{IN}} C_D^{\text{IN}}\right)}{L_c} = -b J_{\text{tot}}^{\text{AVG}}$$

$$\frac{\left(Q_C^{\text{OUT}} C_C^{\text{OUT}} - Q_C^{\text{IN}} C_C^{\text{IN}}\right)}{L_c} = b J_{\text{tot}}^{\text{AVG}}$$

$$\frac{Q_D^{\text{OUT}} - Q_D^{\text{IN}}}{L_c} = -b q_w^{\text{AVG}}$$

$$\frac{Q_C^{\text{OUT}} - Q_C^{\text{IN}}}{L_c} = b q_w^{\text{AVG}}$$

Constitutive Equations

$$J_{\text{tot}}^{\text{AVG}} = J_{\text{cond}}^{\text{AVG}} + J_{\text{diff}}^{\text{AEM,AVG}} + J_{\text{diff}}^{\text{CEM,AVG}}$$

$$J_{\text{cond}}^{\text{AVG}} = \left[t_{\text{CEM}}^{\text{counter}} - (1 - t_{\text{AEM}}^{\text{counter}})\right] \frac{I}{F b L_c}$$

$$J_{\text{diff}}^{\text{IEM,AVG}} = -\frac{D^{\text{IEM}}}{\delta^{\text{IEM}}} \left(C_C^{\text{AVG}} - C_D^{\text{AVG}}\right)$$

$$q_w^{\text{AVG}} = q_{\text{eosm}}^{\text{AVG}} + q_{\text{osm}}^{\text{AEM,AVG}} + q_{\text{osm}}^{\text{CEM,AVG}}$$

$$q_{\text{eosm}}^{\text{AVG}} = \frac{w J_{\text{tot}}^{\text{AVG}} M_w}{\rho_w}$$

$$q_{\text{osm}}^{\text{IEM,AVG}} = L_p^{\text{IEM}} \left[v R_G T \left(\Phi_C^{\text{IEM}} C_C^{\text{AVG}} - \Phi_D^{\text{IEM}} C_D^{\text{AVG}}\right)\right]$$

Electrical Constraints

$$V_{cp} = \eta^{\text{AVG}} + R_{\text{tot}}^{\text{AVG}} \frac{I}{b L_c}$$

$$V_{\text{tot}} = \frac{R_{\text{blank}} I}{A} + \sum_{i=1}^{N_{cp}} V_{cp,i}$$

$$R_{\text{tot}}^{\text{AVG}} = R_{\text{CEM}}^{\text{AVG}} + R_{\text{AEM}}^{\text{AVG}} + R_C^{\text{AVG}} + R_D^{\text{AVG}}$$

$$R_{\text{SOL}}^{\text{AVG}} = f_{s,\text{SOL}} \frac{\delta_{\text{SOL}}}{\Lambda_{\text{SOL}}^{\text{AVG}} C_{\text{SOL}}^{\text{AVG}}}$$

$$\eta^{\text{AVG}} = \eta_{\text{CEM}}^{\text{AVG}} + \eta_{\text{AEM}}^{\text{AVG}}$$

$$\eta_{\text{IEM}}^{\text{AVG}} = \alpha_{\text{IEM}} \frac{R_G T}{F} \ln\left[\frac{\gamma_C^{\text{AVG}} C_C^{\text{AVG}}}{\gamma_D^{\text{AVG}} C_D^{\text{AVG}}}\right]$$

Figure 5.2: Full set of equations to simulate a 0D-ED module cell-pair at steady-state.

ally intractable for the scope of using these tools within process simulation environments. Dedicated PDAEs solvers would be required such as those found in the *COMSOL* fluid dynamics simulation environment (which is only capable of simulating a single system, and not an entire process flowsheet). In addition to the 0D-ED model presented in the previous section, a set of differential equations in terms of time derivatives will be required to fully specify the system. These equations take the form:

$$\frac{d\left(V_{\text{tank}}(t)C_{\text{tank}}(t)\right)}{dt} = C_{\text{tank}}(t)^{\text{IN}}Q_{\text{tank}}^{\text{IN}} - C_{\text{tank}}(t)^{\text{OUT}}Q_{\text{tank}}^{\text{OUT}}, \tag{5.21}$$

$$\frac{dV_{\text{tank}}}{dx} = Q_{\text{tank}}^{\text{IN}} - Q_{\text{tank}}^{\text{OUT}}, \tag{5.22}$$

where the tank can either be the dilute or concentrate tank. The above equations for each copy can directly be added to the conservation equations block in Figure 5.2 to create a system of DAEs in time. The only difference is that the "IN" and "OUT" for the 0D model is switched notation-wise since the "IN" and "OUT" refer to the tanks in the batch operation mode.

5.2.3 Dynamic Electrodialysis Model

When disturbances in feed conditions (e.g. concentrations, power, and volumetric flowrates) are experienced, a dynamic ED model is required to simulate the observed process behavior. This model allows for detailed dynamic operation so a proper control strategy can be implemented (e.g. the results presented in section 5.4). Again, the dynamic model is adapted from Campione et al. [139]. The approach is similar to the 0D-ED model, except the main mass balance equations for the salt and water have an additional time derivative term to describe the temporal dynamics experienced by the ED module (i.e. $\frac{dC_{\text{SOL}}}{dt} \neq 0 \neq \frac{dQ_{\text{SOL}}}{dt}$ where SOL $= \{\text{C,D}\}$). Again, the system of equations to solve takes a form similar to Figure 5.2 with time derivatives, and can be found in Figure 5.3.

Conservation Equations

$$b\delta_D \frac{dC_D^{\text{OUT}}}{dt} = \frac{\left(Q_D^{\text{OUT}} C_D^{\text{OUT}} - Q_D^{\text{IN}} C_D^{\text{IN}}\right)}{L_c} + bJ_{\text{tot}}^{\text{AVG}}$$

$$b\delta_C \frac{dC_C^{\text{OUT}}}{dt} = \frac{\left(Q_C^{\text{OUT}} C_C^{\text{OUT}} - Q_C^{\text{IN}} C_C^{\text{IN}}\right)}{L_c} - bJ_{\text{tot}}^{\text{AVG}}$$

$$\frac{Q_D^{\text{OUT}} - Q_D^{\text{IN}}}{L_c} = -bq_w^{\text{AVG}}$$

$$\frac{Q_C^{\text{OUT}} - Q_C^{\text{IN}}}{L_c} = bq_w^{\text{AVG}}$$

Constitutive Equations

$$J_{\text{tot}}^{\text{AVG}} = J_{\text{cond}}^{\text{AVG}} + J_{\text{diff}}^{\text{AEM,AVG}} + J_{\text{diff}}^{\text{CEM,AVG}}$$

$$J_{\text{cond}}^{\text{AVG}} = \left[t_{\text{CEM}}^{\text{counter}} - (1 - t_{\text{AEM}}^{\text{counter}})\right] \frac{I}{FbL_c}$$

$$J_{\text{diff}}^{\text{IEM,AVG}} = -\frac{D^{\text{IEM}}}{\delta^{\text{IEM}}} \left(C_C^{\text{AVG}} - C_D^{\text{AVG}}\right)$$

$$q_w^{\text{AVG}} = q_{\text{eosm}}^{\text{AVG}} + q_{\text{osm}}^{\text{AEM,AVG}} + q_{\text{osm}}^{\text{CEM,AVG}}$$

$$q_{\text{eosm}}^{\text{AVG}} = \frac{wJ_{\text{tot}}^{\text{AVG}} M_w}{\rho_w}$$

$$q_{\text{osm}}^{\text{IEM,AVG}} = L_p^{\text{IEM}} \left[vR_G T \left(\Phi_C^{\text{IEM}} C_C^{\text{AVG}} - \Phi_D^{\text{IEM}} C_D^{\text{AVG}}\right)\right]$$

Electrical Constraints

$$V_{cp} = \eta^{\text{AVG}} + R_{\text{tot}}^{\text{AVG}} \frac{I}{bL_c}$$

$$V_{\text{tot}} = \frac{R_{\text{blank}} I}{A} + \sum_{i=1}^{N_{cp}} V_{cp,i}$$

$$R_{\text{tot}}^{\text{AVG}} = R_{\text{CEM}}^{\text{AVG}} + R_{\text{AEM}}^{\text{AVG}} + R_C^{\text{AVG}} + R_D^{\text{AVG}}$$

$$R_{\text{SOL}}^{\text{AVG}} = f_{s,\text{SOL}} \frac{\delta_{\text{SOL}}}{\Lambda_{\text{SOL}}^{\text{AVG}} C_{\text{SOL}}^{\text{AVG}}}$$

$$\eta^{\text{AVG}} = \eta_{\text{CEM}}^{\text{AVG}} + \eta_{\text{AEM}}^{\text{AVG}}$$

$$\eta_{\text{IEM}}^{\text{AVG}} = \alpha_{\text{IEM}} \frac{R_G T}{F} \ln\left[\frac{\gamma_C^{\text{AVG}} C_C^{\text{AVG}}}{\gamma_D^{\text{AVG}} C_D^{\text{AVG}}}\right]$$

Figure 5.3: Full set of equations to simulate a time-dynamic module cell-pair.

5.3 CAFO Base Case Process

Prior to showing different control strategies and process designs for nutrient recovery, a realistic base case scenario must be built. This is so a nutrient recovery process can be designed and controlled based on this scenario. Even though WWTPs are a target application, CAFO waste streams have about an order of magnitude higher concentration of nutrients. Consequently, they will be the first target installation for creating a fertilizer product. For future work, WWTPs may only be economical for directly treating the water and getting a more dilute fertilizer product. Conversely, CAFOs have more realistic direct use available since CAFOs are located in rural areas where farmland is already located. Additionally, WWTPs are in densely populated, urban areas that would incur higher transportation costs.

To create this base case, first it is imperative to understand the relative sizes of different CAFO operations based on different types of animals, and also the traditional means to treat the manure stream. For the latter part, a lagoon is a pond that is built on the property with biological treatment depths that will treat the wastewater for high nitrogen, high phosphorous, and harmful biological species so that it can be recycled or put on specific crops called "spray crops". These crops are usually as certain grasses that are meant to absorb the excess nitrogen to turn the contaminants into organic matter. See Chastain et al. for an example of the different treatment volumes [9]. One major problem with these lagoons is that over half of the nitrogen is lost as gaseous ammonia, which is a greenhouse gas, and one of the largest polluters of our atmosphere along with CO_2. For the former part, the Clemson Extension Confined Animal Manure Managers Program has many training manuals with useful information [9]. Chapter 3 of that training manual shows information on the generation of swine wastewater per animal unit (AU). Utilizing a table from the EPA which breaks down the number of animals per AU for different types (*https://www3.epa.gov/npdes/pubs/sector_table.pdf*), coupled with the ranges for an average CAFO size, the base case CAFO is chosen to be 1000 AU. Given the information

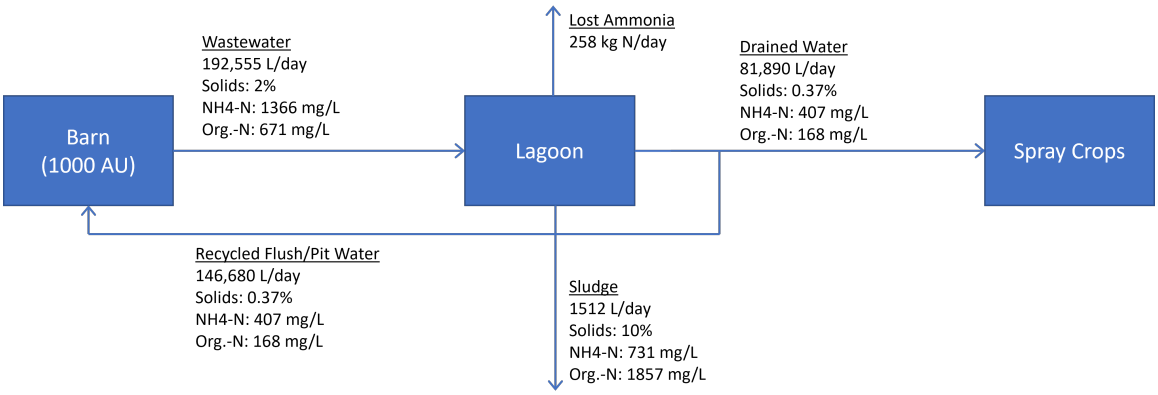


Figure 5.4: Lagoon treatment volume diagram. Flows were calculated based on concentrations and flowrates found in [9].

presented in Table 3.3 of the Clemson Extension Confined Animal Manure Managers Program manual, relative flowrates can be calculated. The final base case scenario is presented in Figure 5.4. The ED system will fit into this figure by replacing the lagoon and spray crops to create a fertilizer product, and recycle quality water.

5.4 Control Strategies for Electrodialysis Processes

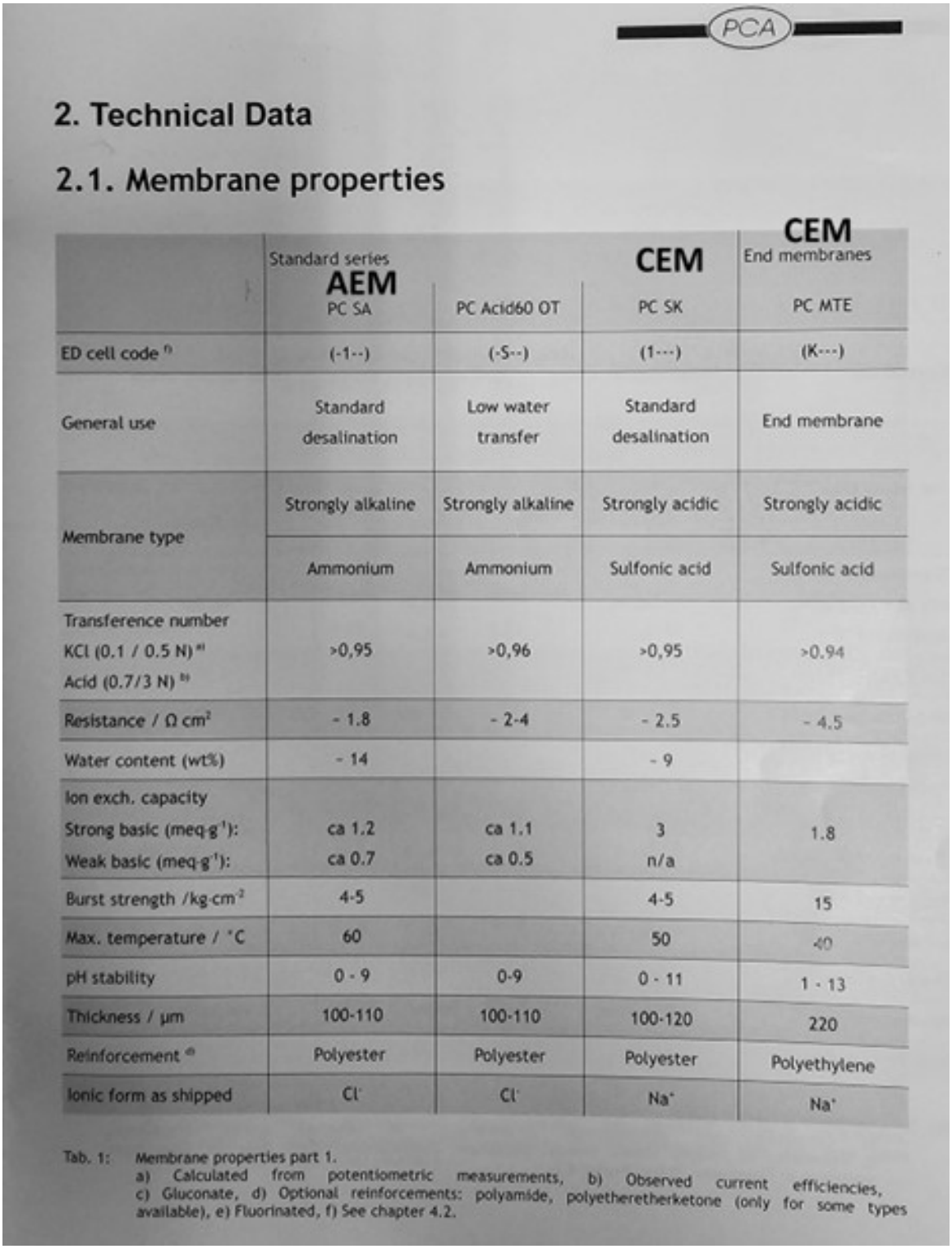
Using the model presented in subsection 5.2.3, along with the base case scenario built in section 5.3, an identification of disturbed, control, and manipulated variables can be found for the nutrient recovery system based on electrodialysis. Even though the final system design will be an entire ED cascade comprised of multiple units, the initial control strategies investigated here only consider a single dynamic ED unit operation. Since the system is anticipated to be powered from renewable energy sources, which are dynamic in nature, the first disturbance has been identified to be the input power (or total voltage). Moreover, because the variability in waste depends on the number of animals present and the biological treatment rate (which both will vary from season to season and year to year), the second disturbance is considered to be the feed concentration. This unit operation has the controlled variable as the dilute concentration since for this example the dilute stream needs to be of recycle water quality for the on-site reuse throughout the CAFO.

Lastly, the manipulated variable will be the dilute flowrate. In the end, this will cause the CAFO to use more recycle water if the dilute flowrate is decreased. However, keeping the concentration of the recycle water quality was chosen to be more important such that an additional concentration control-loop to keep the recycle quality water mixed with the fresh water at a constant concentration is not required. Instead, the concentration is fixed, and only the total CAFO inlet volumetric flowrate needs to be consistent.

Figure 5.6 shows the dynamic process behavior when the ED module is initialized at the inlet concentrations, C_C^{IN} and C_D^{IN}. The ED system design was chosen based on the base case presented in section 5.3. The ED module size was based on the pilot plant ED desalination units from Campione et al. [1]. Table 5.3 and Figure 5.5 show the exact details to replicate this simulation. Note that the flowrates were chosen based on the wastewater stream from the CAFO barn shown in Figure 5.4. The stream was then split into the respective dilute and concentrate streams such that the treated dilute stream can be sent back as the recycled flush/pit water with the same flowrate listed in Figure 5.4. For the initial control strategy analysis, an open-loop response for a 50% decrease in the total voltage (seen in Figure 5.7a) and 50% increase in the feed concentration (seen in Figure 5.8a) input to the ED module was first tested. Then, a simple proportional control law of the form is presented below

$$Q_D^{\text{OUT}} = Q_{D,\text{SS}}^{\text{OUT}} + K_p \left(C_C^{\text{SET}} - C_C^{\text{OUT}} \right), \tag{5.23}$$

with $C_C^{\text{SET}} = 65$ mol/m^3, and $K_p = 1$ able to properly handle each disturbance (as shown in Figure 5.7b and Figure 5.8b). From these results, initial control strategies and identification of key control variables have been elucidated for ED nutrient recovery applied to a realistic CAFO base case.



PCA

2. Technical Data

2.1. Membrane properties

	Standard series AEM		CEM	CEM End membranes
	PC SA	PC Acid60 OT	PC SK	PC MTE
ED cell code [f]	(-1--)	(-S--)	(1---)	(K---)
General use	Standard desalination	Low water transfer	Standard desalination	End membrane
Membrane type	Strongly alkaline	Strongly alkaline	Strongly acidic	Strongly acidic
	Ammonium	Ammonium	Sulfonic acid	Sulfonic acid
Transference number KCl (0.1 / 0.5 N) [a] Acid (0.7/3 N) [b]	>0,95	>0,96	>0,95	>0.94
Resistance / Ω cm²	- 1.8	- 2-4	- 2.5	- 4.5
Water content (wt%)	- 14		- 9	
Ion exch. capacity Strong basic (meq·g⁻¹): Weak basic (meq·g⁻¹):	ca 1.2 ca 0.7	ca 1.1 ca 0.5	3 n/a	1.8
Burst strength /kg·cm⁻²	4-5		4-5	15
Max. temperature / °C	60		50	40
pH stability	0 - 9	0-9	0 - 11	1 - 13
Thickness / µm	100-110	100-110	100-120	220
Reinforcement [d]	Polyester	Polyester	Polyester	Polyethylene
Ionic form as shipped	Cl^-	Cl^-	Na^+	Na^+

Tab. 1: Membrane properties part 1.
a) Calculated from potentiometric measurements, b) Observed current efficiencies, c) Gluconate, d) Optional reinforcements: polyamide, polyetheretherketone (only for some types available), e) Fluorinated, f) See chapter 4.2.

Figure 5.5: Technical membrane parameter data for AEM, CEM, and end cap CEM from PCA "Ion Exchange Membranes for Electromembrane Processes" – Membrane Handling Guide.

Electrodialysis Module Specifications	
Total Voltage	260 V
Channel Length	0.5 m
Channel Width	0.15 m
Channel Dilute Flowrate	0.012 m^3/hr
Channel Concentrate Flowrate	0.004 m^3/hr
Calculated Total Current	59 A
Calculated Specific Energy Consumption	0.25 kWh/kg NH_4

Table 5.3: ED Module Specifications for control strategy test cases. Membrane/ED system parameters from manufacturer, transport number based on KCl can be found in Figure 5.5

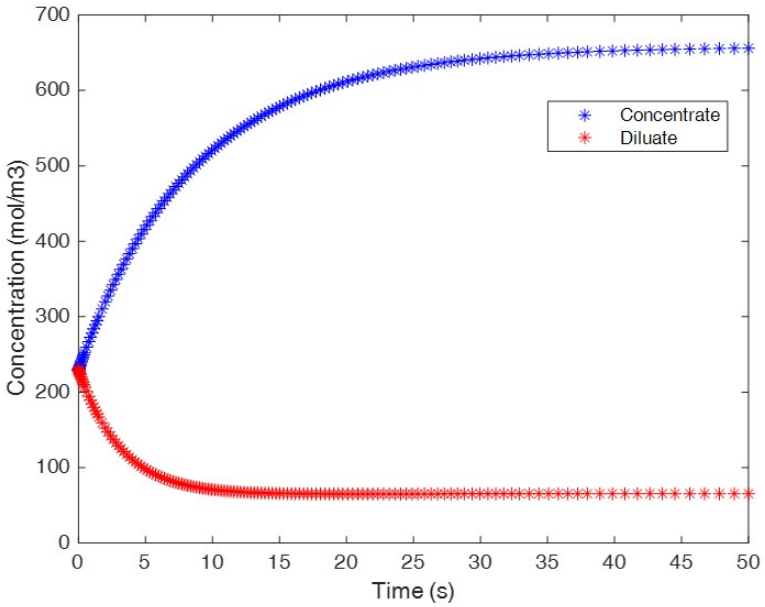
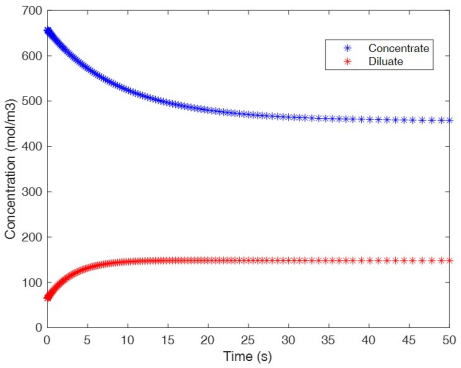
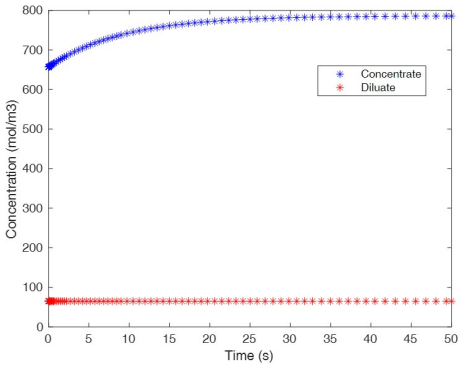


Figure 5.6: Convergence of ED module to steady-state with C_C^{OUT} and C_D^{OUT} initialized to C_C^{IN} and C_D^{IN}, respectively.

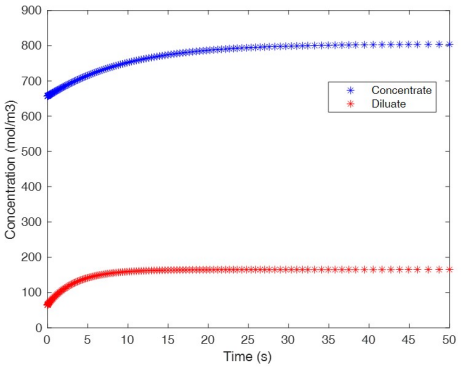


(a) Open-loop response to disturbance

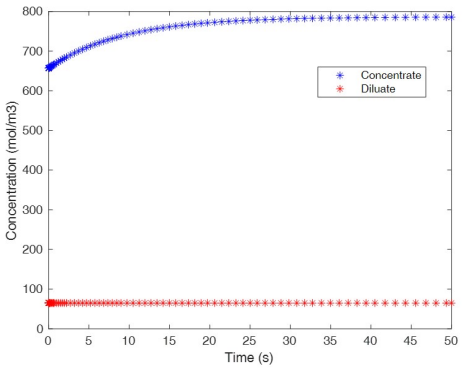


(b) Closed-loop response with proportional control

Figure 5.7: Process disturbance responses for a 50% decrease in the total voltage input to the ED module. The green dotted line represents the set-point dilute concentration.



(a) Open-loop response to disturbance



(b) Closed-loop response with proportional control

Figure 5.8: Process disturbance responses for a 50% increase in the feed concentration input to the ED module. The green dotted line represents the set-point dilute concentration.

5.5 Processes Design of ED Processes

This section details: (i) the implementation of the steady-state model presented in subsection 5.2.1 into a process simulation environment using AmsterChem's *MATLAB* custom unit operation and (ii) the preliminary process design capabilities of the custom ED unit-op. While the presented design is rough and unoptimized, the simulated ED process is shown to be more energy efficient by almost an order of magnitude compared to fertilizer production using the Haber-Bosch process.

5.5.1 ED Custom Unit Operation

Similar to the custom unit operation implemented in section 4.6, the steady-state ED model presented in subsection 5.2.1 can be used within any process simulation environment using AmsterChem's *MATLAB* unit operation. The only difference is that this ED code is much less involved given that there is only a main script to initialize the DAEs solver, and the accompanying DAEs residual function. Screenshots of the AspenPlus flowsheet, port tab, parameter tab, *MATLAB* tab, and additional files tab can be found in Figure 5.9, Figure 5.10, Figure 5.11, Figure 5.12, and Figure 5.13, respectively. Again, the modeling and simulation framework presented is exactly that in subsection 5.2.1 and subsubsection 5.2.1. This tool is available on the *transport-modeling* GitHub [99].

5.5.2 Preliminary ED Process Design Results

Using the tool built from the previous subsection, a preliminary ED cascade process design can be done for the first time. Given that capability, there are still many degrees of freedom including the module size, power consumption for each module, split ratios of feed streams, recycle stream placements, and overall process goal (e.g. to create a fertilizer product for an economical gain, to treat the water fully for an environmental gain, or a mix of both). To create a first-pass process, the module size will be fixed to the one from Campione et

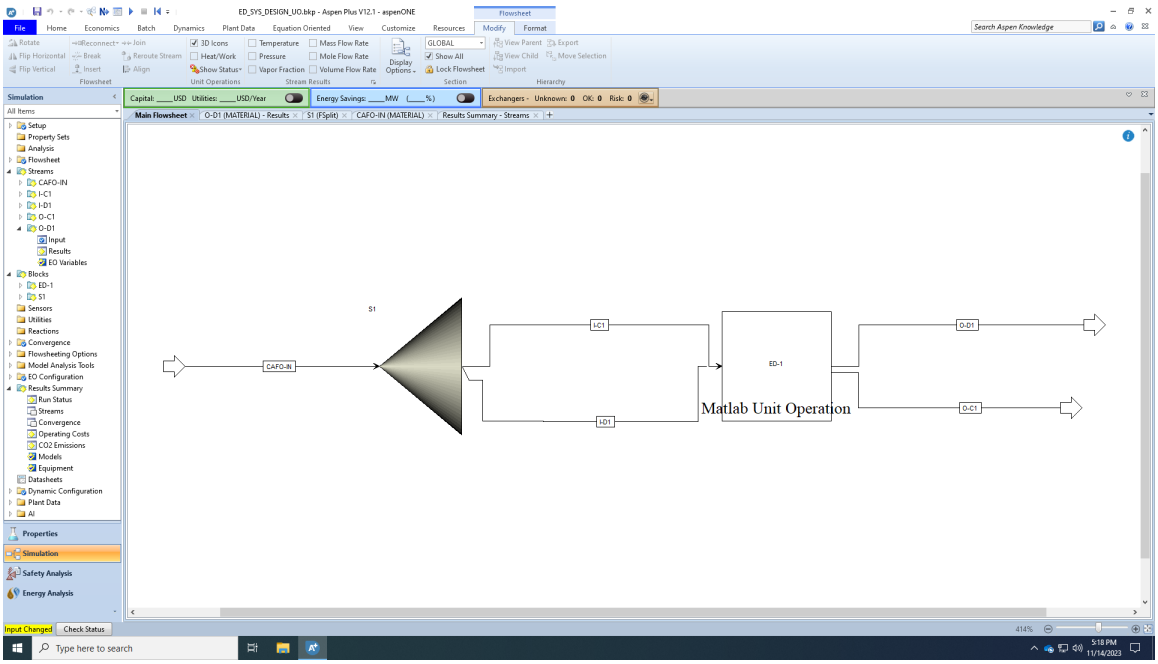


Figure 5.9: Basic AspenPlus flowsheet with *MATLAB* CAPE-OPEN ED unit operation.

al. [1]. The use of recycle streams will be left out, and the split ratio of the feed streams will be fixed at 50%. Lastly, the total voltage was fixed at the maximal separation possible before the extra power would cause dead zones within the ED module (i.e. minimize ED module parts where no concentration change would be occurring, similar to Figure 5.6, but with the x-axis being channel length rather than time).

Figure 5.14 shows the first-pass process design to see how concentrated of a fertilizer product was achievable. Eventually, water transport due to osmotic pressure and salt transport due to an applied voltage began to cancel each other out such that no product stream concentration would occur (i.e. concentrate stream dilution due to the water transport from the feed stream). Given that, the final fertilizer product was found to be 5.6 wt% NH_4-N. For context, the 5 year goal for the CASFER project is to create a technology that can create 10 wt% NH_4-N. Considering this is bringing a stream all the way from 0.1 wt%, this is a massive concentration factor. At the same time, all dilute streams from all 10 ED modules were of recycle quality, and mixed into a single stream to be sent back to the CAFO barn. Figure 5.15 shows the entire ED cascade put back into the base case process flowsheet to

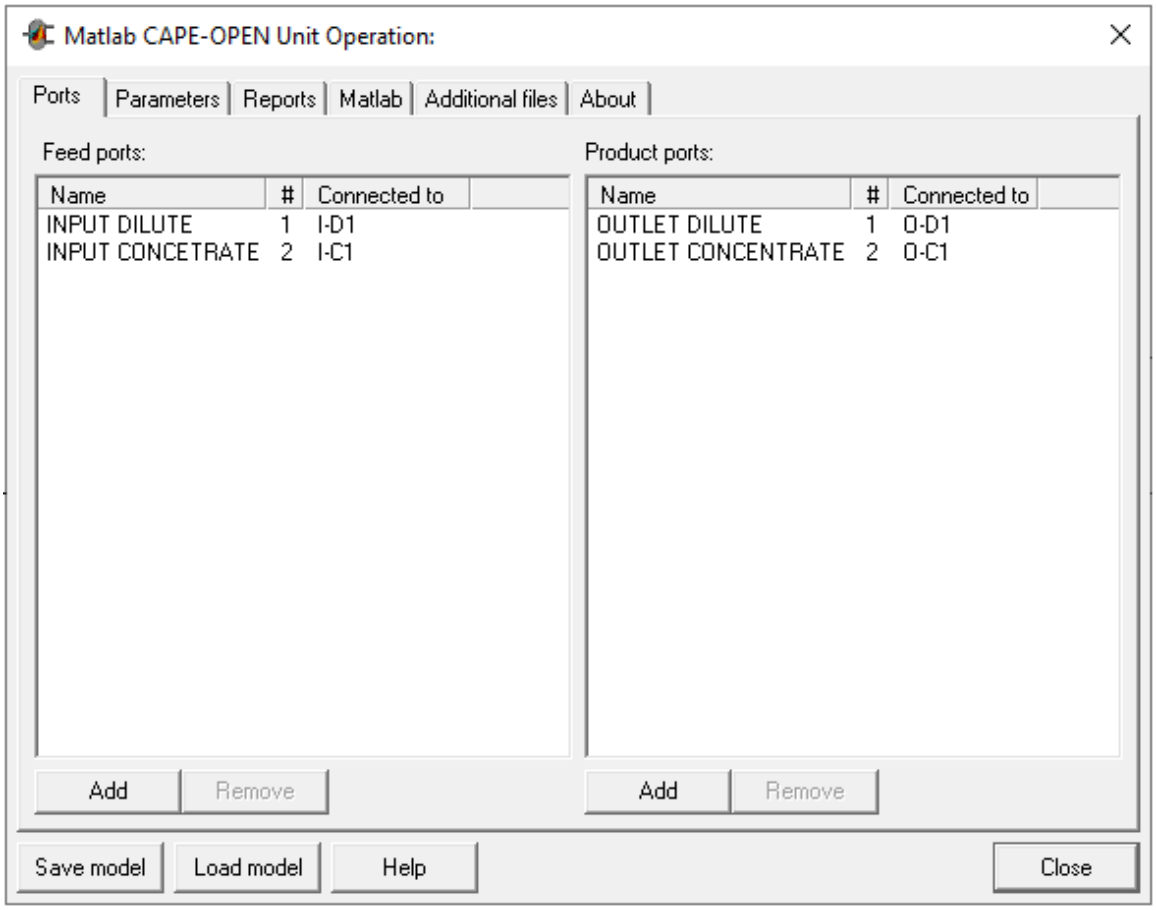


Figure 5.10: AmsterChem *MATLAB* CAPE-OPEN ED unit operation user interface – Ports tab.

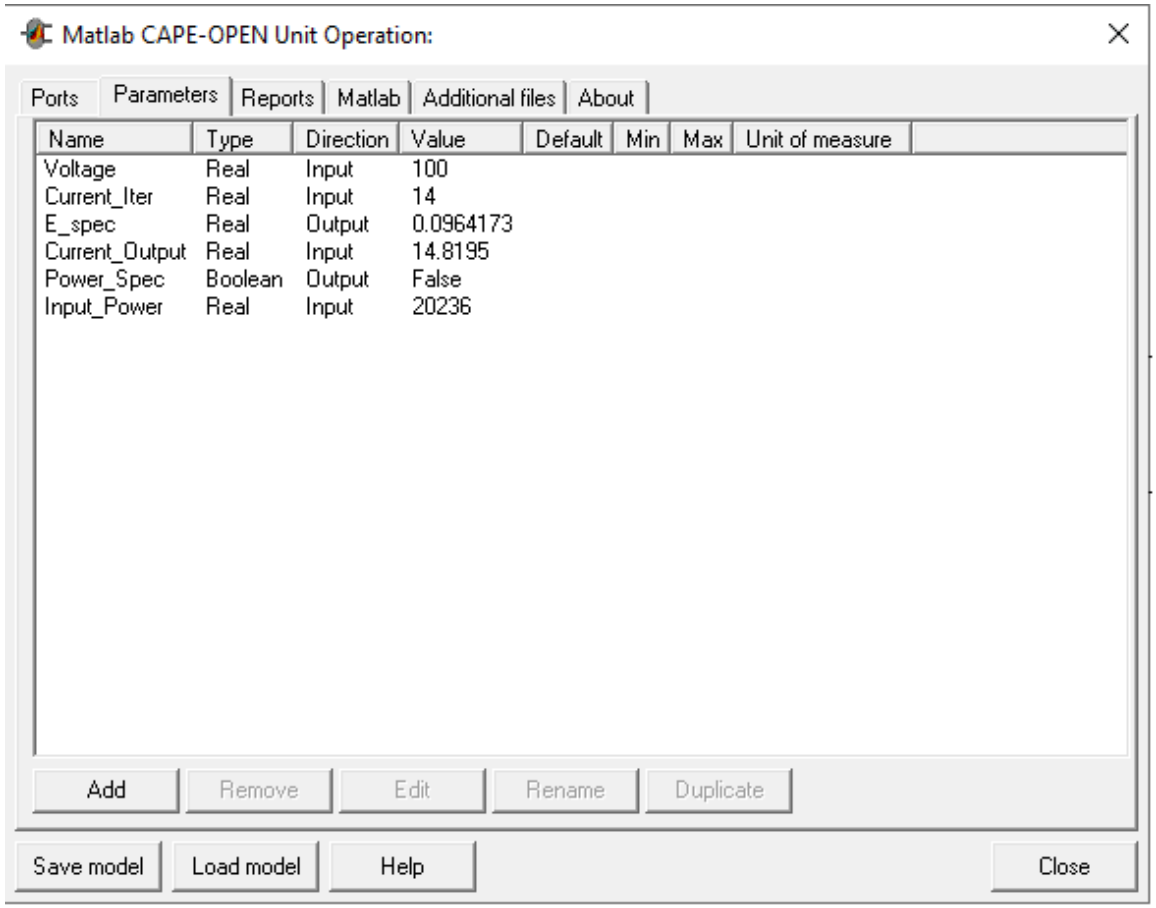


Figure 5.11: AmsterChem *MATLAB* CAPE-OPEN ED unit operation user interface – Parameter tab. The parameter units are the same as used in Table 5.3.

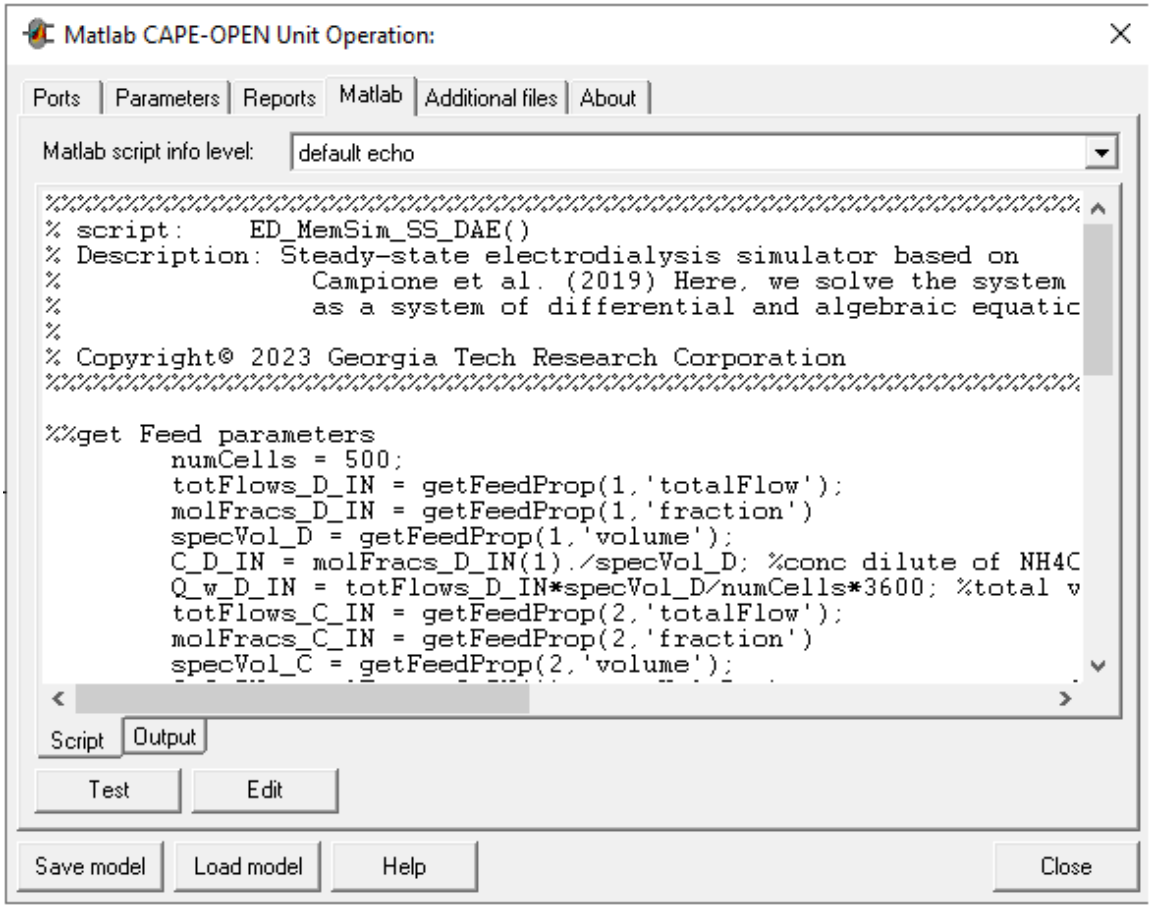


Figure 5.12: AmsterChem *MATLAB* CAPE-OPEN ED unit operation user interface – *MATLAB* tab.

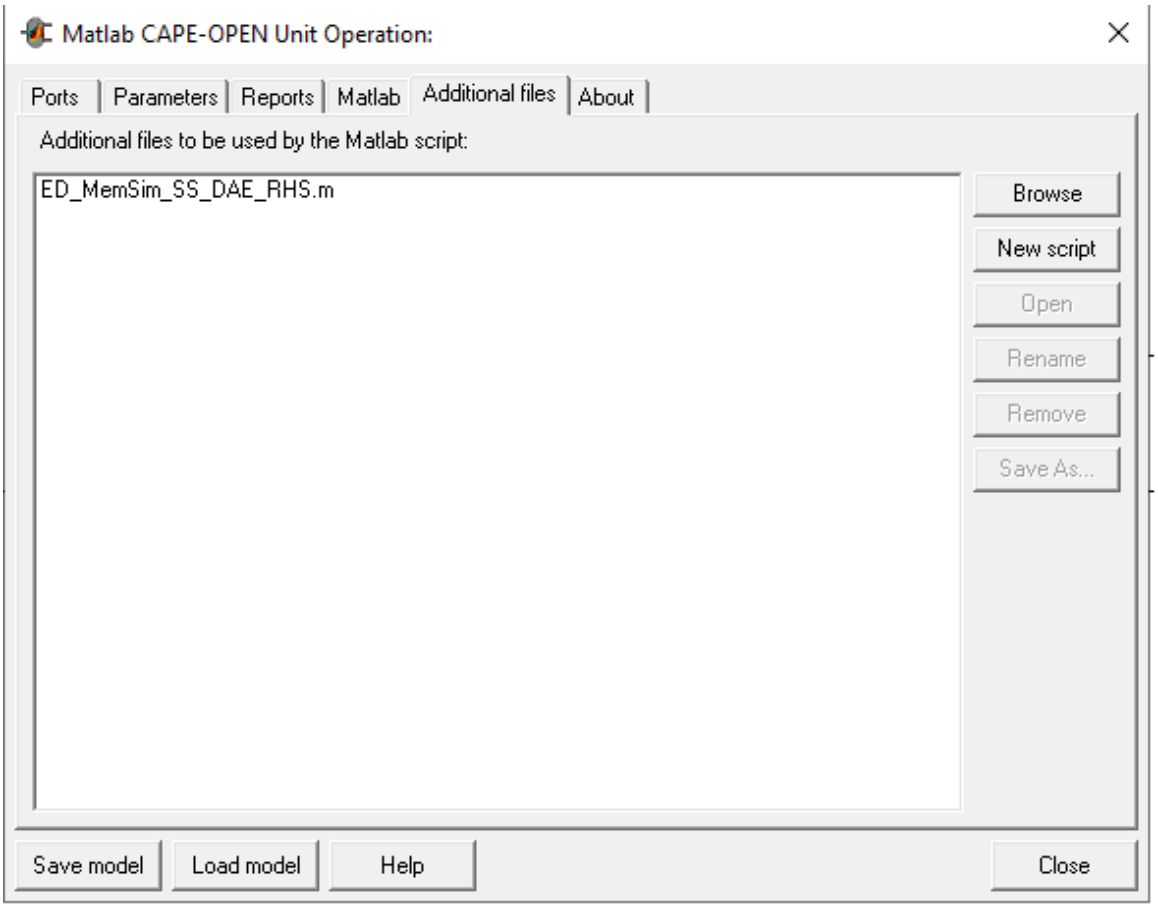


Figure 5.13: AmsterChem *MATLAB* CAPE-OPEN ED unit operation user interface – Additional files tab.

see the overall stream amounts. From this design, there is a 72% recovery of ammonium from the CAFO wastewater. Additionally, the energy consumption is calculated to be 2.26 kWh/kg NH_4-N. This value is almost 4 times lower than ammonia production using conventional Haber-Bosch (which is reported to be 8.44 kWh/kg NH_4-N) [140]. Lastly, the system footprint is estimated to be the size of a fridge, which can fit within a modular process with no problem. Given these great results, they are unoptimized and a first-pass. One major drawback that needs to be address is the maximum flowrate an ED module of this size can handle. If the limit is much lower than what is assumed now, then there will need to be additional ED modules added. This will bring the specific energy consumption up, but also possibly down since the power consumption can be spread out across multiple modules that will possibly create an overall lower power consumption. Regardless, the tool and first-pass process design are a step forward in the right direction to determine the limits and applicability of ED from a system design standpoint. There are other CASFER technologies such as ion magnetic adsorption, biologically-assisted struvite crystallization, and sludge electrolysis that also need to be simulated on a systems level to create a library of different technologies that can be coupled together in many possible configurations. From this, a toolkit to design modular nutrient recovery processes can be realized.

5.6 Conclusions

Within this chapter, electrodialysis was presented as a promising technique to decarbonize the nitrogen and phosphorous economy through electrification. From a modeling and simulation standpoint, many different operation modes were shown including: steady-state, zero-dimensional approximation, batch, and dynamic modes. The code is open-source and available through *MATLAB*. For the dynamic ED model, initial control strategies and process behavior were investigated for realistic disturbances in the total available power and the feed concentration. The results show that a simple proportional control law works well for the test cases. For the steady-state ED model, the code was extended to be used within

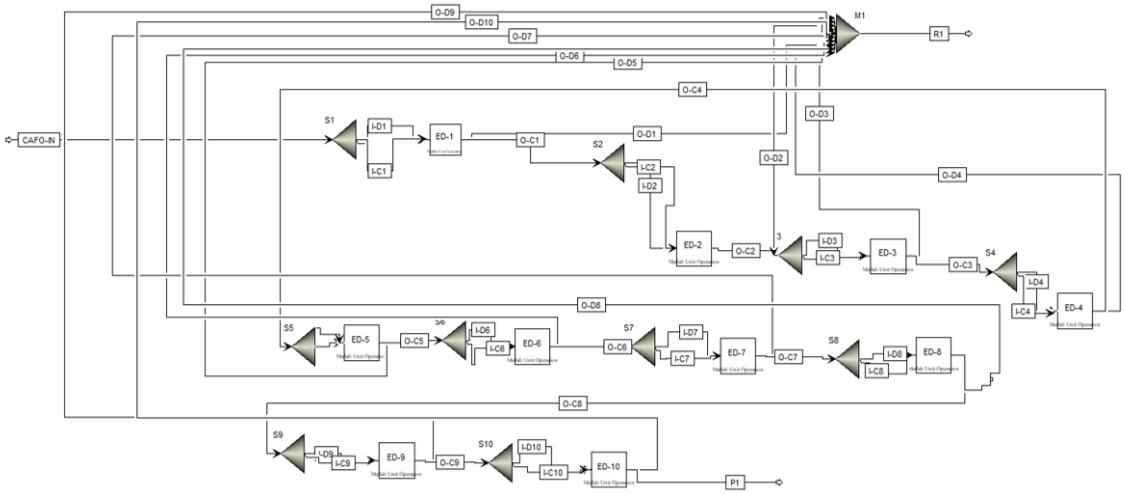


Figure 5.14: ED cascade AspenPlus flowsheet with *MATLAB* CAPE-OPEN ED unit operation. System specifications: 100V for each ED UO, 500 cell pairs (0.2V per cell pairs), 16.5A avg calculated current, 0.025 A/m2 average calculated current density, 0.5 m ED UO length, 0.15 ED UO width, 155-micron channel thickness.

a process simulation environment (e.g. AspenPlus). Using that custom unit operation, a preliminary process design was created. This yielded a higher amount of recycle-quality water compared to the base case presented in section 5.3. Additionally, a 5.6 wt% NH_4-N fertilizer product can be delivered at a power cost that is about 4 times less than the conventional means of producing ammonia. From these results, the next graduate student to work on this project will have a solid foundation to write a publication on an extension of this process design, or possibly take it one step further to run some initial process optimization techniques.

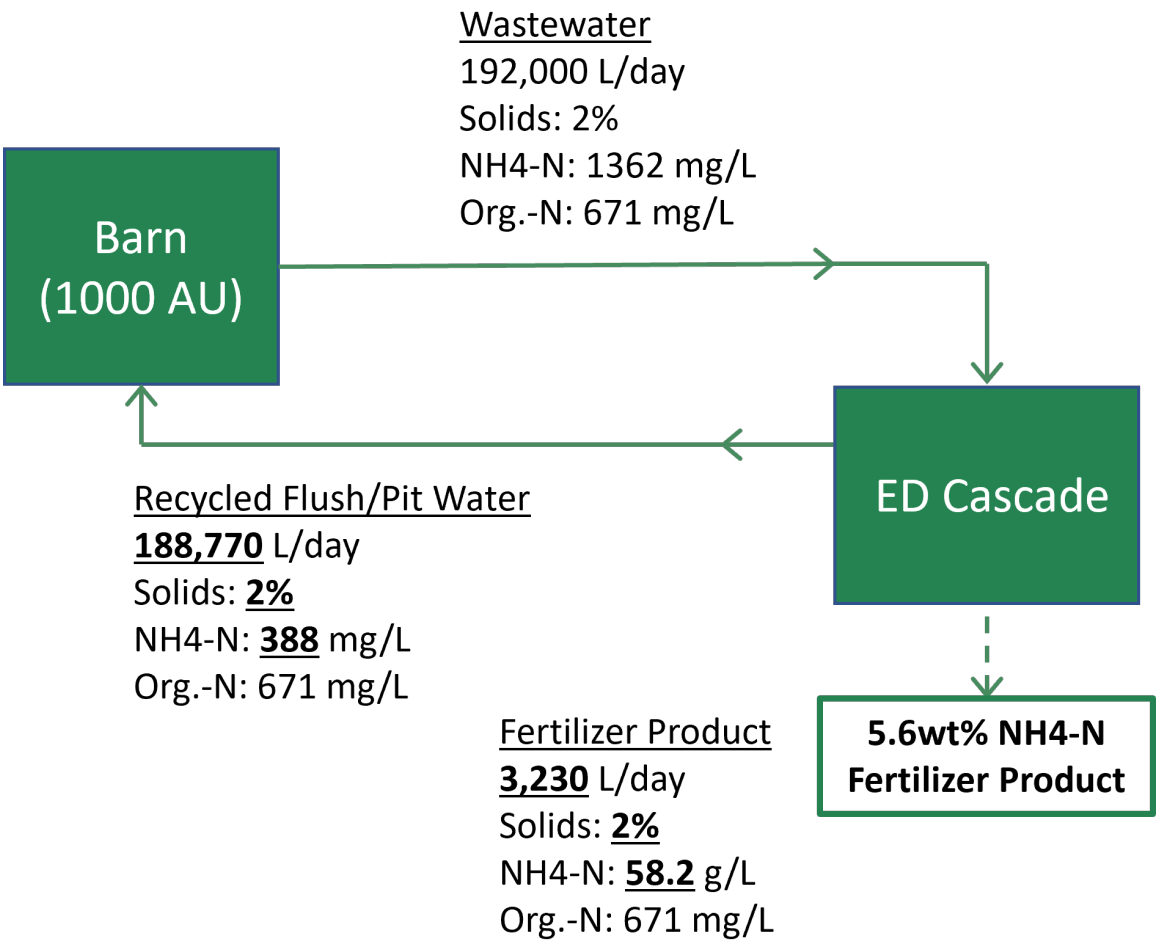


Figure 5.15: Base case flowsheet with ED system added in place of the lagoon and spray crops. Note that 3.18 tonnes/day 5.6 wt% product (0.15 tonnes per day NH4-N) is produced.

CHAPTER 6
CONCLUSIONS AND FUTURE OUTLOOK

6.1 Conclusions

This dissertation addresses the main problem of lowering the entry barrier to modeling and simulation of industrial membrane processes when confronted with complex mixture streams. Specifically, the numerical methods, sorption models, diffusion models, streamlined model implementations, software, and process designs will enable, for the first time, the ability for practicing engineers and/or experimental researchers to rigorously implement these processes that utilize complex mixtures within process simulation environments. In doing so, industrial membrane processes can be properly and fairly compared to the previously identified traditional separation routes to test for economic, operational, and energetic viability for applications that are of industrial relevance. Exemplar applications in which complex mixtures and industrial membrane processes have been listed to be: seawater desalination, brackish groundwater treatment, zero liquid discharge processes, xylene isomer separation, catalyst recovery, biorefinery separations, hydrocarbons from crude oil, alkenes from alkanes, uranium from seawater, rare-earth metals from ores, nitrogen/phosphorous nutrient recover from WWTP/CAFO streams, and carbon capture from atmospheric/industrial streams [2, 8, 141].

6.2 Summary of Novel Research Capabilities

6.2.1 Local Membrane Transport

In chapter 2, improved numerical methods were presented for the membrane local flux problem when confronted with complex mixtures. Specifically, our proposed shooting algorithm was identified as the most accurate, efficient, and robust for all test cases. How-

ever, for certain applications, the average coupling approximation was shown to be highly competitive and sometimes preferred since its implementation results in a simpler system of nonlinear algebraic equations. In contrast, the proposed shooting algorithm must solve DAEs in the worst case, so its implementation is not so straightforward. As an exemplar application to rigorously test these developed numerical methods, organic solvent reverse osmosis through pressure-based industrial membrane processes using glassy asymmetric polymer membranes was studied (permeating complex organic liquid feeds with up to nine components in non-dilute conditions). Regardless, the numeral methods can be modified for different applications based on: different thermodynamic sorption models, custom diffusion models, various concentration representations based on moles or adsorbed species concentration, mixed-matrix membrane materials, and downstream mixing rules (as seen in pervaporation, highly diffusive back-mixing Phase III support layers, and possible temperature differences as seen in membrane distillation).

In section 2.2, the *local flux problem* was rigorously derived and presented in a way that has never been seen in the literature (i.e. Figure 2.2 shows the exact number of knowns and unknowns in the system, and how every piece fits together when assuming a three-step sorption-diffusion mechanism). The key point to recognize is that this modelling framework is predictive in that all that needs to be known are the inlet conditions and all necessary system conditions/parameters. Then, section 2.3 presents the existing simulation methods in great detail such that they can be recreated in a simple manner. Again, throughout the literature on membrane modeling and simulation, there has rarely been a solid write-up of such numerical methods outside of general numerical method textbooks. When applied to pressure-based membrane modeling and simulation, these numerical method implementations will save academic and industrial researchers hours of time and effort. Next, section 2.4 proposes a novel shooting algorithm, a simpler finite differences implementation, and two initialization strategies for proper good initial guesses to the nonlinear equation solver. These initialization strategies were shown to improve robustness for all numerical

methods. Again, scientific literature in the space of membrane modeling and simulations focuses more on the modeling aspect without many details on the simulation and implementation of the model. This dissertation bridges that gap. The results and conclusions show that the proposed shooting algorithm is the most efficient, accurate, and robust for all examined test cases.

6.2.2 Extended Membrane Modeling

In chapter 3, two different modeling problems were presented and solved. In the first problem, section 3.2 presents extended thermodynamic sorption and diffusion capabilities when applied to organic solvent reverse osmosis through glassy polymers. The first challenge identified was the strong thermodynamic interactions between the organic solvent mixture species. Then, the second challenge (consequential from the first) was the way these solvents plasticized the membrane to a rubbery state that caused a transition in the observed permeation landscape. To bridge the gap from theoretical predictions to experimental observations when using state-of-the-art sorprion and diffusion models, three new membrane permeation concepts were constructed. The first concept had to do with the thermodynamic soprion that took place; specifically, how the complex mixture species partitioned from a bulk-feed mixture into the membrane matrix. This model was coined the Flory-Huggins-Langmuir sorption model, which was shown to better describe how the membrane phase interacted with the organic solvents. This concept of two different sorption mechanisms with a multicomponent Flory-Huggins lattice theory coupled with a glassy polymer void filling sorption type was never before seen in the literature. Through a comprehensive set of complex mixture experiments and preliminary multicomponent sorption data, this model was shown as promising to describe organic solvent and glassy polymer membrane interactions and was parameterized solely using single component data.

The second concept that was introduced was based on the observed swelling of the glassy polymer membranes through solvent plasticization which required a different ap-

proach to how the membrane diffusion was theorized. The idea was based on an exponential-type diffusion rate-law that was simple to parameterize based on a known swollen polymer state in the single component case. Consequently, this was easily extended for multicomponent mixture swollen polymer states. The last concept proposed was an even simpler approach to describing this swollen polymer diffusion landscape. The idea was based on an averaging of the diffusion coefficients such that a "cohort" type diffusion was coined to be happening in this polymer state. The simple explanation is that in this state, the membrane loses all diffusion selectivity, and the only selectivity observed is from thermodynamic partitioning. This idea was recently extended and shown to be true in certain cases of OSRO [68]. Applying and implementing these concepts for three complex mixtures permeating through two different polymer membranes, Figure 3.6 shows a profusion of comparisons between state-of-the-art models, and shows how the extended modeling framework outperforms in all cases.

For the final problem shown in section 3.3, the transport modeling framework was extended to be capable of simulating complex organic liquid feeds of up to hundreds of components. Prior to any modifications, the simulations were estimated to converge a solution in over year. After reconstructing the underlying thermodynamic sorption model and accompanying derivative to be represented in matrix-vector notation rather than nested summations, the model could be evaluated in n^2 operations. Whereas before, the model had to be evaluated in n^3 operations by writing multiple nested *FOR* loops. While part of the speed-up was due to vectorization in an interpreted programming language such as *MATLAB*, there was also a compacting of the Flory-Huggins fugacity model that took place to enable the model to be evaluated in n less computations (compare (Equation 3.1) with (Equation 3.14)) by reducing the number of multiplications required as well as the amount of nested summations that needed to be evaluated. This enabled a novel data-driven machine learning approach to crude oil membrane permeation predictions that is further presented in Lee et al. [69].

6.2.3 Pressure-based Industrial Membrane Software

From chapter 4, a *MATLAB*-based software package *asyMemSim*, was established for open-source use. The code allows for simulation of all numerical methods, membrane sorption models, membrane diffusion models, cross-diffusional coupling models, and initialization strategies presented in this dissertation. Within this software package, various pieces of code allow for running multiple mixture simulations at once, running many single component permeation predictions, fitting single component diffusivities based on single component data, running custom complex mixture simulations, and a centralized databank to store any number of model parameters, polymer parameters, and component/polymer properties. This code has been used by many students within Dr. Ryan Lively's research group with relative ease. The code is available at *https://github.com/transport-modeling/asyMemSim* for use by any academic researcher. Practicing chemical engineers can also use this software after reading through the accompanying *README* modeling document, and chapter 4. In addition to this, since the goal of this dissertation is to enable industrial membrane process simulation within process simulation environments, a CAPE-OPEN compliant custom unit operation is available for use in the AspenPlus chemical process simulator. This custom unit op is untested in other CAPE-OPEN compliant process simulation environments, but theoretically should work exactly the same. The implementation is made possible through AmsterCHEM's *MATLAB* CAPE-OPEN Unit Operation (*https://www.amsterchem.com/matlabunitop.html*). The custom unit operations can be found within the previously provide GitHub link. While there are many functionalities and code additions to address, this is a first step in the right direction.

6.2.4 Electrified Industrial Membrane Processes

From chapter 5, a promising application of nitrogen and phosphorous nutrient recovery from CAFOs and WWTPs was presented. Additionally, a modeling and simulation framework was derived for a number of operation modes to have a toolkit of different codes

for lab-scale simulation, control strategy implementation, and overall process design. All codes are open-source and available for use in *MATLAB*. As a necessary extension, the steady-state code is available as a custom unit operation within the AspenPlus process simulation environment. This allowed for a preliminary process design that is presented in subsection 5.5.2. Moreover, a proper CAFO base case was constructed using relevant literature, and simple control strategies were investigated for realistic process disturbances using the dynamic operation ED code. With these contributions, a step forward has been taken for elucidating a modular nutrient recovery process that can be directly deployed on CAFOs to create a fertilizer product, while still treating large quantities of wastewater for reuse on-site.

6.3 Future Outlook

Considering the future of the research presented in this dissertation, the flame is far from dying. Each chapter has its respective possibilities for future work. For chapter 2, the natural extension of solving the local flux problem is to extend it to the global scale modeling. This will allow for modeling and simulation of full-scale pressure-based membrane modules utilizing complex mixtures. The challenge lies within approximating the local problem in such a way that it can easily be extended to the material, energy, and momentum balances that describe the transport through a global membrane module (see subsection 1.3.1 and Appendix C for details of global modules). This is to avoid solving a system of partial differential and algebraic equations such that the problem can be solved within a process simulation environment. Alternatively, one can approximate the global module transport problem by discretizing it into segments, and solve the full local transport problem using the proposed shooting algorithm in subsection 2.4.2. Determining which way is more efficient, accurate, and reliable is an open question. Moreover, for the local transport problem, there are still many different tweaks to the implementation that can enable speed-ups to the code such that it is faster for higher component systems. One of those is writing the DAE

system in Figure 2.2 in terms of a fugacity gradient, and not making the simplification to the volume fraction gradient. From there, the thermodynamic matrix, $\mathbf{\Gamma}$, is circumvented (i.e. less computations are required). However, with this implementation, there will be a $\frac{1}{\mathbf{f}}$ multiplied by the fugacity gradient term that can cause solver instabilities for certain values of fugacity. Again, the relative trade-offs in efficiency, accuracy, and robustness is an open question. Another implementation tweak is supplying the Jacobian to the integrator and/or the outer solver. This can also possibly increase the three qualities listed previously. Extending the local transport problem presented here, and testing alternative model formulations is another possible future application. Specifically, those based in mole fractions and adsorbed species concentrations may be explored to expand the possible applications of the software, *asyMemSim*, while preserving the work done on the local flux problem numerical method development.

From chapter 3, there is still a need for applying the Flory-Huggins-Langmuir fugacity model and Vignes diffusional coupling model to large component systems. From section 3.3, only the cohort diffusion and Flory-Huggins models were assumed. While the experimental data matched fairly well, these alternative models are anticipated to still be applicable when the polymer system is not in a swollen state such that the cohort style diffusion occurs, and the diffusion selectivity is retained. Another natural extension for the work presented in section 3.2 is more rigorous testing of the transport framework and FH-LM fugacity model for many more OSRO systems. The FH-LM model may also be applied to rigid adsorption systems with type-II isotherm behavior. Progressing towards additional high-component simulations ($n > 1000$) is also a possible avenue of research since there are still numerical tweaks that can be applied an explored to make the high component simulations converge more efficiently, accurately, and reliably. Taking advantage of GPU parallelization is another great extension of the code to enable complex mixture permeation simulations of thousands of components. Lastly, exploring a pseudocomponent structure oriented lumping framework extension for membrane permeation simulations would also

enable high component simulations. This is similar to the grouped component models that are already used in high component distillation simulations. Utilizing some sort of parameterization based on penetrant-polymer interactions and molecule shape/size could be a useful basis for such a structure-oriented lumped component model.

In chapter 4, there are many custom implementations and tweaks that can be applied (as described in section 4.5). As far as using the code within a process simulation environment, there are many contributions to be found from process design/optimization/control of ORSO membrane cascades to replicate an oil refinery (or bio-oil) with glassy polymer membranes. All the experimental work done thus far from chapter 2 has been focused on the lab-scale local transport problem. However, the next logical step is to take these parameters and results to see how the performance scales and compares with traditional separation methods. The conclusion from that work may be some sort of hybrid process that can be retrofitted within existing refineries. The possibilities are endless in that regard.

Finally, chapter 5 is the most open-ended work presented in this dissertation. Since the work presented is mainly preliminary, the foundations are there for a solid publication. Specifically, the process design of electrified industrial membrane processes for nutrient recovery from wastewater is something that is missing from the literature. Additionally, the modeling and simulation framework presented was a simple single salt model. Extensions for multicomponent complex mixtures will be required when moving towards more realistic process streams from WWTPs and CAFOs. Possible models are the Nernt-Plank and Maxwell-Stefan. These models account for the thermodynamic and diffusional coupling between species at the cost of being more computationally complex. The balance between matching experiments with the level of computational complexity to include in the process design is a trade-off with an open question. Another extension of this work is to take the lab-scale ED simulators and attempt to fit parameters to more realistic salt species (since the current membrane parameters are based on KCl). Even though KCl and NH_4Cl are both monovalent salts on their own, there will most likely be different transport

behaviors when in contact with multiple species within a complex mixture. There are many experimental collaborations within CASFER to be made, and Dr. Marta Hatzell at Georgia Tech is a great start. Even at the other partner institutions, there are other unit operations such as electrolysis, biologically-assisted struvite production, and magnetic adsorption that should also have custom unit operations and implementations to test the correct treatment train to yield the optimal fertilizer product. Pretreatment filtration and reverse osmosis is another avenue to test feasibility to decrease total solids and water content to be fed to the ED modules. Additional base case scenarios for different CAFO and WWTP operations will also be another research avenue to explore to see how each nutrient recovery process design and control strategies change as a function of the various scenarios.

With all these different branches of research to explore, I hope this dissertation will provide a stepping stone for someone to stand on, and progress in their own contribution(s) to the web of science. Please feel free to reach out to me with any questions or ideas that I could possibly help with. No matter what, we are all in this together to progress the human condition forward in a positive direction.

Appendices

APPENDIX A
IMPROVED NUMERICAL METHODS FOR LOCAL TRANSPORT SUPPLEMENTAL INFORMATION

This chapter provides all the necessary supplemental information for chapter 2 and references to the paper titled "Improved numerical methods for simulating complex mixture transport across asymmetric polymer membranes using a Maxwell–Stefan model" [40].

A.1 Derivation of Chemical Potential Expression

This section is meant to derive a common equation written in membrane literature (see (Equation 2.4) and (Equation 2.5)) [19]. After only reading it in the final form without derivation in [19], understanding its origin will be a good reference below. There are two parts to this derivation. The first will be deriving the relationship between activity and fugacity. The second will be the derivation of the Poynting correction factor from differences in chemical potential reference states. This textbook was used as reference [142]. For sake of simplicity, the superscript phase y notation will be disregarded since throughout these derivations we only work with one phase. Staring with the first derivation, we write the definition of activity in terms of chemical potentials and Gibbs free energy at some T,P as (defining the reference state be pure component, i.e. $\mu_i^\circ = G_i(T,P)$):

$$RT\ln(a_i) = \mu_i - \mu_i^\circ(T,P) = \mu_i - G_i(T,P) \tag{A.1}$$

Next, the definition for the mixture fugacity can be written as the deviation from a reference state that is an pure component ideal gas at the same T, P:

$$\mu_i - \mu_i^{\circ,IG} = RT\ln\left(\frac{f_i}{P}\right) \tag{A.2}$$

The same can also be written for the pure component chemical potential, $\mu^{\circ}(T,P)$, reference state as:

$$\mu_i^{\circ} - \mu_i^{\circ,IG} = RT\ln\left(\frac{f_i^{\circ}}{P}\right) \tag{A.3}$$

Subtracting (Equation A.2) and (Equation A.3) then setting equal to the LHS of (Equation A.1) yields the final expression:

$$RT\ln(a_i) = RT\ln\left(\frac{f_i}{f_i^{\circ}}\right) \tag{A.4}$$

For the second derivation we again start from (Equation A.1) at some T, P but with an arbitrary pure component Gibbs free energy reference state at T, P^{+}:

$$RT\ln(a_i) = \mu_i - \mu_i^{\circ}(T,P) = \mu_i - G_i(T,P) + G_i(T,P^{+}) - G_i(T,P^{+}). \tag{A.5}$$

Then, one can integrate the fundamental property relation for Gibb's free energy $dG_i = V_i dP - S_i dT$:

$$G_i(T,P^{+}) - G_i(T,P) = \int_{T,P}^{T,P^{+}} V_i dP, \tag{A.6}$$

so we end up with (for incompressible liquid):

$$\mu_i = \mu_i^{\circ} + RT\ln(a_i) + V_i(P - P^{+}), \tag{A.7}$$

or in differential form (constant T):

$$d\mu_i = RTd\ln(a_i) + V_i dP. \tag{A.8}$$

A.2 Maxwell-Stefan Equation Analysis

The equations below were adapted from (S10)-(S12) of [64]. In this section we aim to justify why only n Maxwell-Stefan equations are needed. First we will present the initially force-balance derived Maxwell-Stefan model in terms of mole fractions and component

velocities as:

$$-\frac{d\mu_1}{dz} = \frac{RT}{Đ_{12}}x_2(u_1 - u_2) + \frac{RT}{Đ_{13}}x_3(u_1 - u_3) + \cdots + \frac{RT}{Đ_{1m}}x_m(u_1 - u_m), \tag{A.9}$$

$$-\frac{d\mu_2}{dz} = \frac{RT}{Đ_{21}}x_1(u_2 - u_1) + \frac{RT}{Đ_{23}}x_3(u_2 - u_3) + \cdots + \frac{RT}{Đ_{2m}}x_m(u_2 - u_m), \tag{A.10}$$

$$\vdots$$

$$-\frac{d\mu_n}{dz} = \frac{RT}{Đ_{n1}}x_1(u_1 - u_n) + \frac{RT}{Đ_{n2}}x_2(u_n - u_2) + \cdots + \frac{RT}{Đ_{nm}}x_m(u_n - u_m), \tag{A.11}$$

$$-\frac{d\mu_m}{dz} = \frac{RT}{Đ_{m1}}x_1(u_m - u_1) + \frac{RT}{Đ_{m3}}x_3(u_m - u_3) + \cdots + \frac{RT}{Đ_{mn}}x_n(u_m - u_n), \tag{A.12}$$

where $Đ_{ij}$ is the molar-based Maxwell-Stefan diffusivity for the ij component pair, and u_i is the velocity of component i.

Multiplying each Maxwell-Stefan equation by its respective mole fractions, x_i, the RHS of each equation can be added up to yield zero (noting that $Đ_{ij} = Đ_{ji}$). This is exactly the Gibbs-Duhem relationship,

$$x_1\frac{d\mu_1}{dz} + x_2\frac{d\mu_2}{dz} + \cdots + x_n\frac{d\mu_n}{dz} + x_m\frac{d\mu_m}{dz} = 0, \tag{A.13}$$

and as such, only the first 1 to n chemical potential gradients, $\frac{d\mu_i}{dz}$, are independent.

A.3 Thermodynamic Factor Matrix Calculations

This section provides the necessary derivative matrices, also called thermodynamic factors $\left(\Gamma_{ij} = \phi_i\frac{\partial \ln(f_i)}{\partial \phi_j} = \frac{\phi_i}{f_i}\frac{\partial f_i}{\partial \phi_j}\right)$, required for the transport framework outlined in subsection 2.2.3. Since there are three implemented sorption models, there are again three different derivations required for each model (Flory-Huggins, Dual-Mode, and Flory-Huggins-Langmuir).

A.3.1 Flory-Huggins

This derivative is fairly straightforward when starting from (Equation 3.17). Differentiating (Equation 3.17) gives

$$\boldsymbol{\Gamma}(\boldsymbol{\phi}_{1:n}) = \mathbf{I}_{n\times n} + \text{diag}(\boldsymbol{\phi}_{1:n})\left[\tilde{\mathbf{C}} - \mathbf{V}^{\circ}\boldsymbol{\phi}_{1:n}^{\text{T}}\left(\tilde{\mathbf{Q}}^{\text{T}} + \tilde{\mathbf{Q}}\right)\right]. \tag{A.14}$$

Where the required matricies are defined in (Equation 3.17). Using (Equation A.14), the full $\boldsymbol{\Gamma}(\boldsymbol{\phi}_{1:n})$ matrix can be evaluated in order n^2 operations.

A.3.2 Dual-Mode Sorption

The thermodynamic factor matrix is not a straightforward as the previous section. Starting from (Equation 3.6), note that there is no way to rearrange this equation to give fugacity explicitly as a function of the volume fractions. Thus, this must be treated as a general implicit fugacity model of the form $\mathbf{g}(\mathbf{f}, \boldsymbol{\phi}) = \mathbf{0}$. Specifically, the i^{th} component of $\mathbf{g}$ is given by

$$g_i(\mathbf{f}, \boldsymbol{\phi}_{1:n}) = \phi_i - (1 - \sum_{j=1}^{n}\phi_j)\left(k_i f_i + \frac{C_i^H b_i f_i}{1 + \sum_{k=1}^{n} b_k f_k}\right). \tag{A.15}$$

The function $\mathbf{f}(\boldsymbol{\phi}_{1:n})$ that maps a given set of volume fractions into the unique set of component fugacities in the corresponding mixture is defined implicitly by the condition

$$\boldsymbol{\lambda}(\boldsymbol{\phi}_{1:n}) \equiv \mathbf{g}(\mathbf{f}(\boldsymbol{\phi}_{1:n}), \boldsymbol{\phi}_{1:n}) = \mathbf{0}. \tag{A.16}$$

Since $\boldsymbol{\Gamma}$ involves the derivatives of this function, it must be obtained by implicit differentiation. This is done by first taking the total derivative of $\boldsymbol{\lambda}$ and setting it equal to zero (it must equal zero since $\boldsymbol{\lambda}(\boldsymbol{\phi}_{1:n})$ is constant by (Equation A.16)):

$$\frac{d\boldsymbol{\lambda}}{d\boldsymbol{\phi}_{1:n}} = \frac{\partial \mathbf{g}}{\partial \mathbf{f}}\frac{\partial \mathbf{f}}{\partial \boldsymbol{\phi}_{1:n}} + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\phi}_{1:n}} = \mathbf{0}. \tag{A.17}$$

Rearranging for the needed derivative of $\mathbf{f}(\boldsymbol{\phi}_{1:n})$ gives

$$\frac{\partial \mathbf{f}}{\partial \boldsymbol{\phi}_{1:n}} = -\left(\frac{\partial \mathbf{g}}{\partial \mathbf{f}}\right)^{-1} \frac{\partial \mathbf{g}}{\partial \boldsymbol{\phi}_{1:n}}. \tag{A.18}$$

The derivatives of $\mathbf{g}$ above should be taken as if $\mathbf{f}$ and $\boldsymbol{\phi}_{1:n}$ are independent variables. Moreover, note that ϕ_{n+1} was eliminated from (Equation A.15) using $\phi_{n+1} = 1 - \sum_{i=1}^{n} \phi_1$. Thus, both $\mathbf{g}$ and the implicit function $\mathbf{f}(\boldsymbol{\phi}_{1:n})$ defined by (Equation A.16) are functions of only the n independent volume fractions $\boldsymbol{\phi}_{1:n}$. This is necessary for obtaining correct derivatives via (Equation A.18).

Equation (Equation 3.6) can be vectorized as

$$\mathbf{g}(\mathbf{f}, \boldsymbol{\phi}_{1:n}) = \boldsymbol{\phi}_{1:n} - (1 - \mathbf{1}_n^{\mathrm{T}} \boldsymbol{\phi}_{1:n}) \left(\mathrm{diag}(\mathbf{k}) - (1 + \mathbf{b}^{\mathrm{T}}\mathbf{f})^{-1} \mathrm{diag}(\mathbf{c}^H)\mathrm{diag}(\mathbf{b})\right) \mathbf{f}, \tag{A.19}$$

where $\mathbf{k} = (k_1, \ldots, k_n)$, $\mathbf{c}^H = (C_1^H, \ldots, C_n^H)$, and $\mathbf{b} = (b_1, \ldots, b_n)$. Using (Equation A.19), the derivative matrices needed for (Equation A.18) can be obtained using standard matrix calculus rules:

$$\frac{\partial \mathbf{g}}{\partial \boldsymbol{\phi}_{1:n}} = \mathbf{I}_{n\times n} + \left(\mathrm{diag}(\mathbf{k}) + (1 + \mathbf{b}^{\mathrm{T}}\mathbf{f})^{-1} \mathrm{diag}(\mathbf{c}^H)\mathrm{diag}(\mathbf{b})\right) \mathbf{f}\mathbf{1}_n^{\mathrm{T}}, \tag{A.20}$$

$$\frac{\partial \mathbf{g}}{\partial \mathbf{f}} = -(1 - \mathbf{1}_n^{\mathrm{T}} \boldsymbol{\phi}_{1:n}) \left(\mathrm{diag}(\mathbf{k}) - \frac{\mathrm{diag}\left(\mathbf{c}^H\right) \mathrm{diag}\left(\mathbf{b}\right)}{(1 + \mathbf{b}^{\mathrm{T}}\mathbf{f})} \left[\left(1 + \mathbf{b}^{\mathrm{T}}\mathbf{f}\right)^{-1} \mathbf{f}\mathbf{b}^{\mathrm{T}} - \mathbf{I}_{n\times n}\right]\right). \tag{A.21}$$

The DMS thermodynamic factor matrix can now be computed as

$$\boldsymbol{\Gamma}(\boldsymbol{\phi}_{1:n}, \mathbf{f}) = \mathrm{diag}(\boldsymbol{\phi}_{1:n})\mathrm{diag}(\mathbf{f})^{-1} \frac{\partial \mathbf{f}}{\partial \boldsymbol{\phi}_{1:n}}. \tag{A.22}$$

A.3.3 Combined Flory-Huggins-Langmuir

To get the required derivatives for this model, the derivation will start from (Equation 3.17) and (Equation 3.9). As with the DMS model, this model cannot be rearranged to give fugacity explicitly as a function of $\boldsymbol{\phi}_{1:n}$, and is more complex still because it involves the intermediate variables $\boldsymbol{\phi}^{\text{FH}}_{1:n}$. Thus, it is an implicit model with $2n$ equations of the form

$$\mathbf{g}(\boldsymbol{\phi}_{1:n}, \boldsymbol{\phi}^{\text{FH}}_{1:n}, \mathbf{f}) = \begin{bmatrix} \mathbf{r}(\boldsymbol{\phi}_{1:n}, \boldsymbol{\phi}^{\text{FH}}_{1:n}, \mathbf{f}) \\ \mathbf{s}(\boldsymbol{\phi}_{1:n}, \boldsymbol{\phi}^{\text{FH}}_{1:n}, \mathbf{f}) \end{bmatrix} = \begin{bmatrix} \mathbf{0}_n \\ \mathbf{0}_n \end{bmatrix}, \tag{A.23}$$

where $\mathbf{r}$ and $\mathbf{s}$ represent (Equation 3.9) and (Equation 3.17), respectively. In matrix-vector form,

$$\mathbf{r}(\boldsymbol{\phi}_{1:n}, \boldsymbol{\phi}^{\text{FH}}_{1:n}, \mathbf{f}) = \boldsymbol{\phi}_{1:n} - \boldsymbol{\phi}^{\text{FH}}_{1:n} - (1 - \mathbf{1}_n^{\text{T}}\boldsymbol{\phi}_{1:n})(1 + \mathbf{b}^{\text{T}}\mathbf{f})^{-1}\text{diag}(\mathbf{c}^H)\text{diag}(\mathbf{b})\mathbf{f}, \tag{A.24}$$

$$\mathbf{s}(\boldsymbol{\phi}_{1:n}, \boldsymbol{\phi}^{\text{FH}}_{1:n}, \mathbf{f}) = \ln\left(\boldsymbol{\phi}^{\text{FH}}_{1:n}\right) - \ln\left(\frac{\mathbf{f}}{\mathbf{f}^\circ}\right) + \mathbf{p} + \tilde{\mathbf{C}}\boldsymbol{\phi}^{\text{FH}}_{1:n} - \mathbf{V}^\circ_{n+1}(\boldsymbol{\phi}^{\text{FH}}_{1:n})^{\text{T}}\tilde{\mathbf{Q}}\boldsymbol{\phi}^{\text{FH}}_{1:n}, \tag{A.25}$$

where the natural log and division are taken componentwise.

Given any $\boldsymbol{\phi}_{1:n}$, these equations can be solved simultaneously to obtain the corresponding $\boldsymbol{\phi}^{\text{FH}}_{1:n}$ and $\mathbf{f}$. Thus, the functions $\boldsymbol{\phi}^{\text{FH}}_{1:n}(\boldsymbol{\phi}_{1:n})$ and $\mathbf{f}(\boldsymbol{\phi}_{1:n})$ that map a given set of volume fractions into the unique set of Flory-Huggins volume fractions and component fugacities in the corresponding mixture are defined implicitly by the conditions

$$\boldsymbol{\Lambda}(\boldsymbol{\phi}_{1:n}) \equiv \mathbf{r}\left(\boldsymbol{\phi}_{1:n}, \mathbf{f}(\boldsymbol{\phi}_{1:n}), \boldsymbol{\phi}^{\text{FH}}_{1:n}(\boldsymbol{\phi}_{1:n})\right) = \mathbf{0}, \tag{A.26}$$

$$\boldsymbol{\Upsilon}(\boldsymbol{\phi}_{1:n}) \equiv \mathbf{s}\left(\boldsymbol{\phi}_{1:n}, \mathbf{f}(\boldsymbol{\phi}_{1:n}), \boldsymbol{\phi}^{\text{FH}}_{1:n}(\boldsymbol{\phi}_{1:n})\right) = \mathbf{0}. \tag{A.27}$$

Since $\boldsymbol{\Gamma}$ involves the derivatives of $\mathbf{f}(\boldsymbol{\phi}_{1:n})$, it must be obtained by implicit differentiation. This is done by first taking the total derivatives of $\boldsymbol{\Lambda}$ and $\boldsymbol{\Upsilon}$ and setting them equal to zero

(they must equal zero since they are constant by (Equation A.26) and (Equation A.27)):

$$\frac{d\boldsymbol{\Lambda}}{d\boldsymbol{\phi}_{1:n}} = \frac{\partial \mathbf{r}}{\partial \mathbf{f}}\frac{\partial \mathbf{f}}{\partial \boldsymbol{\phi}_{1:n}} + \frac{\partial \mathbf{r}}{\partial \boldsymbol{\phi}^{\text{FH}}_{1:n}}\frac{\partial \boldsymbol{\phi}^{\text{FH}}_{1:n}}{\partial \boldsymbol{\phi}_{1:n}} + \frac{\partial \mathbf{r}}{\partial \boldsymbol{\phi}_{1:n}} = \mathbf{0}, \tag{A.28}$$

$$\frac{d\boldsymbol{\Upsilon}}{d\boldsymbol{\phi}_{1:n}} = \frac{\partial \mathbf{s}}{\partial \mathbf{f}}\frac{\partial \mathbf{f}}{\partial \boldsymbol{\phi}_{1:n}} + \frac{\partial \mathbf{s}}{\partial \boldsymbol{\phi}^{\text{FH}}_{1:n}}\frac{\partial \boldsymbol{\phi}^{\text{FH}}_{1:n}}{\partial \boldsymbol{\phi}_{1:n}} + \frac{\partial \mathbf{s}}{\partial \boldsymbol{\phi}_{1:n}} = \mathbf{0}. \tag{A.29}$$

Noting that $\frac{\partial \mathbf{s}}{\partial \boldsymbol{\phi}_{1:n}} = \mathbf{0}$, (Equation A.29) can be solved for $\frac{\partial \boldsymbol{\phi}^{\text{FH}}_{1:n}}{\partial \boldsymbol{\phi}_{1:n}}$ as

$$\frac{\partial \boldsymbol{\phi}^{\text{FH}}_{1:n}}{\partial \boldsymbol{\phi}_{1:n}} = -\left(\frac{\partial \mathbf{s}}{\partial \boldsymbol{\phi}^{\text{FH}}_{1:n}}\right)^{-1}\frac{\partial \mathbf{s}}{\partial \mathbf{f}}\frac{\partial \mathbf{f}}{\partial \boldsymbol{\phi}_{1:n}}. \tag{A.30}$$

Substituting this into (Equation A.28) and solving for $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\phi}_{1:n}}$ then gives

$$\frac{\partial \mathbf{f}}{\partial \boldsymbol{\phi}_{1:n}} = -\left(\frac{\partial \mathbf{r}}{\partial \mathbf{f}} - \frac{\partial \mathbf{r}}{\partial \boldsymbol{\phi}^{\text{FH}}_{1:n}}\left(\frac{\partial \mathbf{s}}{\partial \boldsymbol{\phi}^{\text{FH}}_{1:n}}\right)^{-1}\frac{\partial \mathbf{s}}{\partial \mathbf{f}}\right)^{-1}\frac{\partial \mathbf{r}}{\partial \boldsymbol{\phi}_{1:n}}. \tag{A.31}$$

Finally, the thermodynamic factor matrix (Equation A.22) is given by

$$\boldsymbol{\Gamma}(\boldsymbol{\phi}_{1:n}, \mathbf{f}) = -\text{diag}(\boldsymbol{\phi}_{1:n})\text{diag}(\mathbf{f})^{-1}\left(\frac{\partial \mathbf{r}}{\partial \mathbf{f}} - \frac{\partial \mathbf{r}}{\partial \boldsymbol{\phi}^{\text{FH}}_{1:n}}\left(\frac{\partial \mathbf{s}}{\partial \boldsymbol{\phi}^{\text{FH}}_{1:n}}\right)^{-1}\frac{\partial \mathbf{s}}{\partial \mathbf{f}}\right)^{-1}\frac{\partial \mathbf{r}}{\partial \boldsymbol{\phi}_{1:n}}. \tag{A.32}$$

The derivatives of $\mathbf{r}$ and $\mathbf{s}$ above should be taken as if $\mathbf{f}$, $\boldsymbol{\phi}^{\text{FH}}_{1:n}$, and $\boldsymbol{\phi}_{1:n}$ are independent variables. These derivatives are readily derived from (Equation A.24) and (Equation A.25) using standard matrix calculus rules as

$$\frac{\partial \mathbf{s}}{\partial \mathbf{f}} = -\text{diag}\left(\mathbf{f}\right)^{-1}, \tag{A.33}$$

$$\frac{\partial \mathbf{s}}{\partial \boldsymbol{\phi}_{1:n}^{\text{FH}}} = \text{diag}\left(\boldsymbol{\phi}_{1:n}^{\text{FH}}\right)^{-1} + \tilde{\mathbf{C}} - \mathbf{V}^{\circ}(\boldsymbol{\phi}_{1:n}^{\text{FH}})^{\text{T}}\left(\tilde{\mathbf{Q}}^{\text{T}} + \tilde{\mathbf{Q}}\right), \tag{A.34}$$

$$\frac{\partial \mathbf{r}}{\partial \boldsymbol{\phi}_{1:n}^{\text{FH}}} = -\mathbf{I}_{n\times n}, \tag{A.35}$$

$$\frac{\partial \mathbf{r}}{\partial \boldsymbol{\phi}_{1:n}} = \mathbf{I}_{n\times n} + (1 + \mathbf{b}^{\text{T}}\mathbf{f})^{-1}\text{diag}(\mathbf{c}^{H})\text{diag}(\mathbf{b})\mathbf{f}\mathbf{1}_{n}^{\text{T}}, \tag{A.36}$$

$$\frac{\partial \mathbf{r}}{\partial \mathbf{f}} = \phi_m\left(\text{diag}(\mathbf{c}^{H})\text{diag}(\mathbf{b})\left[\left(1 + \mathbf{b}^{\text{T}}\mathbf{f}\right)^{-2}\mathbf{f}\mathbf{b}^{\text{T}} - \left(1 + \mathbf{b}^{\text{T}}\mathbf{f}\right)^{-1}\mathbf{I}_{n\times n}\right]\right), \tag{A.37}$$

where the matrices in (Equation A.34) are defined in (Equation 3.17).

A.4 Explicit Isotherm Simulation Method Simplifications

As mentioned in Section 3.1 of the main text, the implementation of the approximation methods and finite difference (FD) methods described in the main text can all be simplified when the membrane phase fugacity model is explicit in fugacity (simplifications of the shooting method in this case are discussed in the main text; see Figure 3). Recall that the general implicit membrane fugacity model takes the form $\mathbf{g}^{II}(\mathbf{f}^{II}, \boldsymbol{\phi}^{II}, T^{II}, P^{II}) = \mathbf{0}$. The model is said to be explicit in fugacity if there exists and algebraic rearrangement that isolates $\mathbf{f}^{II}$ on one side of the equation; i.e.,

$$\mathbf{f}^{II} = \tilde{\mathbf{g}}^{II}(\boldsymbol{\phi}_{1:n}^{II}, T^{II}, P^{II}). \tag{A.38}$$

Among the fugacity models used in this paper, only the Flory-Huggins (FH) model can be rearranged in this way, so only the simulations using the FH model benefited from the simplifications described in this section.

Recall that the complete set of equations describing the rigorous local flux model with no simplifications is given in Figure 2.2 of the main text. That figure groups the equations

into three blocks describing, respectively, phase equilibrium on the feed side of the membrane, transport across the active layer, and phase equilibrium on the permeate side. All of the approximation and FD methods described in the main text were presented in terms of the modifications they make to this set of equations, which all occur in Block 2. Also recall that, for all of those methods, the final numerical procedure involved first solving Block 1 independently, and then solving Blocks 2 and 3 simultaneously. The simplifications of these methods enabled by the explicit form (Equation A.38) involve using (Equation A.38) to completely eliminate the unknowns $\mathbf{f}^{II}$ from Blocks 2 and 3, resulting in a smaller set of equations to solve numerically.

To accomplish this, first note that, as a consequence of having an explicit fugacity model, the corresponding thermodynamic factor matrix $\mathbf{\Gamma}(\boldsymbol{\phi}_{1:n}^{II}, \mathbf{f}^{II})$ can be written as a function of $\boldsymbol{\phi}_{1:n}^{II}$ only. This is because

$$\begin{aligned}\mathbf{\Gamma}(\boldsymbol{\phi}_{1:n}^{II}) &= \text{diag}(\boldsymbol{\phi}_{1:n}^{II})\text{diag}(\mathbf{f}^{II})^{-1}\frac{\partial \mathbf{f}^{II}}{\partial \boldsymbol{\phi}_{1:n}^{II}}, \\ &= \text{diag}(\boldsymbol{\phi}_{1:n}^{II})\text{diag}(\tilde{\mathbf{g}}^{II}(\boldsymbol{\phi}_{1:n}^{II}, T^{II}, P^{II}))^{-1}\frac{\partial \tilde{\mathbf{g}}^{II}}{\partial \boldsymbol{\phi}_{1:n}^{II}}(\boldsymbol{\phi}_{1:n}^{II}, T^{II}, P^{II}).\end{aligned} \tag{A.39}$$

This implies that the first equation in Block 2 no longer depends on $\mathbf{f}^{II}$, and hence both the unknowns $\mathbf{f}^{II}$ and the equations $\mathbf{g}^{II} = \mathbf{0}$ can be completely eliminated from Block 2. To completely eliminate $\mathbf{f}^{II}$, the value at the permeate side, $\mathbf{f}^{II}$, must also be eliminated from Block 3 by directly substituting $\tilde{\mathbf{g}}^{II}(\boldsymbol{\phi}_{L,1:n}^{II}, T^{II}, P^{II})$. If the bulk permeate is assumed to be ideal ($\boldsymbol{\gamma}^{III} = \mathbf{1}$), then the first equation in Block 3 is also not needed and the second equation can be used to completely eliminate the variables $\mathbf{x}^{III}$ as well.

Since each of the approximation and FD methods handle the Block 2 equations differently, the following subsections show the final form of these equations for each method after incorporating the explicit fugacity model simplifications outlined above.

A.4.1 Fick's Law Approximation

$$\mathbf{0} = \mathbf{h}(\boldsymbol{\phi}_L, \mathbf{x}^{III}, N_{\text{tot}}) \tag{A.40}$$

$$= \begin{bmatrix} \text{diag}(\mathbf{V}^{\circ})\mathbf{x}^{III} N_{\text{tot}} + \text{diag}(\mathbf{D}_{\text{Fick}}) \frac{\boldsymbol{\phi}_{1:n,L} - \boldsymbol{\phi}_{1:n,0}}{L} \\ 1 - \sum_{i=1}^{n+1} \phi_{i,L} \end{bmatrix}, \tag{A.41}$$

where $\mathbf{D}_{\text{Fick}} = (D^V_{1m,\text{Fick}}, D^V_{2m,\text{Fick}}, \ldots, D^V_{nm,\text{Fick}})$

A.4.2 Volume Fraction-form Average Coupling Approximation

$$\mathbf{0} = \mathbf{h}(\boldsymbol{\phi}_L, \mathbf{x}^{III}, N_{\text{tot}}) \tag{A.42}$$

$$= \begin{bmatrix} \text{diag}(\mathbf{V}^{\circ})\mathbf{x}^{III} N_{\text{tot}} + \mathbf{B}^{-1}(\boldsymbol{\phi}_{\text{avg}})\boldsymbol{\Gamma}(\boldsymbol{\phi}_{\text{avg}}) \frac{\boldsymbol{\phi}_{1:n,L} - \boldsymbol{\phi}_{1:n,0}}{L} \\ 1 - \sum_{i=1}^{n+1} \phi_{i,L} \end{bmatrix}. \tag{A.43}$$

A.4.3 Fugacity Form Average Coupling Approximation

$$\mathbf{0} = \mathbf{h}(\boldsymbol{\phi}_L, \mathbf{x}^{III}, N_{\text{tot}}) \tag{A.44}$$

$$\equiv \begin{bmatrix} \frac{\text{diag}(\boldsymbol{\phi}_{1:n,\text{avg}}) \ln\left(\frac{\tilde{\mathbf{g}}^{II}(\boldsymbol{\phi}_L)}{\tilde{\mathbf{g}}^{II}(\boldsymbol{\phi}_0)}\right)}{L} + \mathbf{B}(\boldsymbol{\phi}_{\text{avg}})\text{diag}(\mathbf{V}^{\circ})\mathbf{x}^{III} N_{\text{tot}} \\ 1 - \sum_{i=1}^{n+1} \phi_{i,L} \end{bmatrix}.$$

A.4.4 Fugacity Form Finite Difference Method

The equations for a generic node $0 < s < S$ are

$$\mathbf{0} = \mathbf{h}_s(\boldsymbol{\phi}_{s-1}, \boldsymbol{\phi}_s, \boldsymbol{\phi}_{s+1}, \mathbf{x}^{III}, N_{\text{tot}}) \tag{A.45}$$
$$\equiv \begin{bmatrix} \frac{\text{diag}(\boldsymbol{\phi}_{1:n,s}) \ln\left(\frac{\tilde{\mathbf{g}}^{II}(\boldsymbol{\phi}_{s+1}))}{\tilde{\mathbf{g}}^{II}(\boldsymbol{\phi}_{s-1}))}\right)}{z_{s+1}-z_{s-1}} + \mathbf{B}(\boldsymbol{\phi}_s)\text{diag}(\mathbf{V}^\circ)\mathbf{x}^{III}N_{\text{tot}} \\ 1 - \sum_{i=1}^{n+1} \phi_{i,s} \end{bmatrix}.$$

A.4.5 Volume-Fraction Form Finite Difference Method

The equations for a generic node $0 < s < S$ are

$$\mathbf{0} = \mathbf{h}_s(\boldsymbol{\phi}_{s-1}, \boldsymbol{\phi}_s, \boldsymbol{\phi}_{s+1}, \mathbf{x}^{III}, N_{\text{tot}}) \tag{A.46}$$
$$\equiv \begin{bmatrix} \boldsymbol{\Gamma}(\boldsymbol{\phi}_s)\frac{(\boldsymbol{\phi}_{1:n,s+1}-\boldsymbol{\phi}_{1:n,s-1})}{z_{s+1}-z_{s-1}} + \mathbf{B}(\boldsymbol{\phi}_s)\text{diag}(\mathbf{V}^\circ)\mathbf{x}^{III}N_{\text{tot}} \\ 1 - \sum_{i=1}^{n+1} \phi_{i,s} \end{bmatrix}.$$

A.5 Thermodynamic Factor Matrix Comparison

Figure A.1, discussed in Section 5.4 of the main text, shows that $\boldsymbol{\Gamma}$ has significant off-diagonal entries when evaluated at $z = 0$ for the 5 and 9 component test cases. The implicit fugacity model has larger off-diagonal elements compared to the explicit model and contains negative values.

A.6 Solving the Local Flux Problem with Fixed Permeate Composition

As discussed in subsection 2.2.1 of the main text, the use of Pan's relationship may not always be appropriate in the formulation of the local flux problem. In this section, we provide details on an alternative formulation in which Pan's relationship is omitted and we instead assume that $\mathbf{x}^{III} = \mathbf{x}^{IV}$ is given. While Pan's relationship assumes that the bulk permeate stream does not back-mix into the membrane support layer, and hence $\mathbf{x}^{III}$ is determined entirely by the component fluxes through the active layer, the formulation discussed here

Thermodynamic Matrix, Γ, at z=0

	1	2	3	4	5	6	7	8	9
1	1.1389	0.1485	0.0823	0.1676	0.1003	0.1366	0.1330	0.1162	0.1062
2	0.1697	1.1990	0.0862	0.2069	0.1500	0.2043	0.1704	0.1459	0.1528
3	0.0143	0.0123	1.0125	0.0176	0.0046	0.0074	0.0124	0.0104	0.0048
4	0.0369	0.0401	0.0234	1.0446	0.0304	0.0392	0.0365	0.0329	0.0316
5	0.2064	0.2597	0.0713	0.2685	1.1916	0.2765	0.2143	0.1764	0.1921
6	0.0358	0.0458	0.0139	0.0454	0.0363	1.0503	0.0383	0.0315	0.0354
7	0.0161	0.0179	0.0091	0.0196	0.0130	0.0177	1.0159	0.0139	0.0136
8	0.0094	0.0102	0.0054	0.0116	0.0073	0.0097	0.0093	1.0083	0.0079
9	0.0059	0.0071	0.0023	0.0075	0.0053	0.0072	0.0061	0.0053	1.0055

	1	2	3	4	5
1	1.120	-0.032	0.064	0.076	-0.001
2	-0.099	0.823	-0.116	-0.108	-0.100
3	0.046	-0.088	1.003	0.013	-0.049
4	0.096	-0.080	0.034	1.050	-0.047
5	0.186	0.089	0.133	0.138	1.088

Figure A.1: Thermodynamic coupling matrix, $\mathbf{\Gamma}$, evaluated at $z = 0$ for the explicit fugacity model nine component mixture through SBAD-1, and the implicit fugacity model five component mixture through PIM-1.

instead assumes that back-mixing occurs so completely that $\mathbf{x}^{III}$ is determined entirely by the fixed permeate composition $\mathbf{x}^{IV}$. Figure A.2 shows the complete set of equations and unknowns for this formulation. In the corresponding figure assuming Pan's relationship in the main text, the unknown component volumetric fluxes $\mathbf{N}^V$ were eliminated from the MS equations using Pan's relationship in the form

$$\mathbf{N}^V = \text{diag}(\mathbf{V}^\circ)\mathbf{x}^{III}N_{\text{tot}},$$

and both $\mathbf{x}^{III}$ and N_{tot} were unknowns. In contrast, in Figure A.2, $\mathbf{x}^{III}$ is known and, since Pan's relationship no longer holds, the unknown $\mathbf{N}^V$ is retained in the MS equations. In sum, the $n + 1$ unknowns $\mathbf{x}^{III}$ and N_{tot} are swapped for the n unknowns $\mathbf{N}^V$. This results in the loss of one degree of freedom, but the system is made square again by eliminating the equation $\sum_i x_i^{III} = 1$ from the third block.

We now discuss the impacts of these change on the numerical methods discussed in the main text. The most significant impact is on the Fick's law and average-coupling approx-

Phase Equilibrium at $z = 0$

$$\mathbf{g}^{\circ}(\mathbf{f}^{\circ}, T, P^{II}) = \mathbf{0}$$

$$\mathbf{g}^{I}(\boldsymbol{\gamma}^{I}, \mathbf{x}^{I}, T, P^{I}) = \mathbf{0}$$

$$\mathbf{g}^{II}(\mathbf{f}_0^{II}, \boldsymbol{\phi}_0, T, P^{II}) = \mathbf{0}$$

$$f_{i,0}^{II} = \gamma_{i,0}^{I} x_i^{I} f_i^{\circ}, \quad i = 1, \ldots, n$$

$$\sum_{j=1}^{n+1} \phi_{j,0} = 1$$

Active Layer Diffusion

$$\frac{d\boldsymbol{\phi}_{1:n}}{dz} = -\boldsymbol{\Gamma}^{-1}(\boldsymbol{\phi}, \mathbf{f}^{II})\mathbf{B}(\boldsymbol{\phi})\mathbf{N}^{V}$$

$$\frac{d\phi_m}{dz} = -\sum_{j=1}^{n} \frac{d\phi_j}{dz}$$

$$\mathbf{g}^{II}(\mathbf{f}^{II}, \boldsymbol{\phi}, T, P^{II}) = \mathbf{0}$$

Phase Equilibrium at $z = L$

$$\mathbf{g}^{III}(\boldsymbol{\gamma}^{III}, \mathbf{x}^{III}, T, P^{III}) = \mathbf{0}$$

$$f_{i,L}^{II} = \gamma_{i,L}^{III} x_i^{III} f_i^{\circ} \exp\left[\frac{-V_i^{\circ}(P^{II} - P^{III})}{RT}\right],$$

$$i = 1, \ldots, n$$

Known variables: $(\mathbf{x}^{I}, \mathbf{x}^{III}, T, P^{I}, P^{II}, P^{III})$

Unknown variables: $(\mathbf{f}^{\circ}, \boldsymbol{\gamma}^{I}, \boldsymbol{\gamma}^{III}, \mathbf{f}_0^{II}, \mathbf{f}_L^{II}, \boldsymbol{\phi}_0, \boldsymbol{\phi}_L, \mathbf{N}^{V})$

DoF: $6n + 2(n+1)$ equations $- [6n + 2(n+1)]$ unknowns
= 0 DoF

Figure A.2: Full set of equations and degrees of freedom analysis for the local transport problem when assuming a fixed permeate (Phase IV) composition.

imation methods. Specifically, fixing the value of $\mathbf{x}^{III}$ enables then permeate-side phase equilibrium problem to be solved independently for ϕ_L and $\mathbf{f}_L^{II}$ in exactly the same way that the feed side is handled in the main text. Then, with ϕ and $\mathbf{f}^{II}$ known on both sides of the active layer, it becomes straightforward to solve for $\mathbf{N}^V$ explicitly, without the need for an iterative procedure (see, e.g., [39]). In consequence, all three approximation methods become more efficient and significantly more robust (although the permeate side phase equilibrium problem could still fail to converge in principle). However, the main drawback of these methods is their inaccuracy relative to the solution of the full MS model, which is unchanged.

The changes to the FD and shooting approaches are much less significant. For the FD methods, the governing equations are still discretized over S nodes and solved simultaneously as before. The structure of the shooting algorithm, which is nearly identical to the original, is shown in Figure A.3. While is it is true that fixing $\mathbf{x}^{III}$ enables then permeate-side phase equilibrium problem to be solved independently for these methods as well, this does not result in a significant simplification of the overall solution procedures because it remains necessary to solve the implicit membrane phase fugacity model at every intermediate point inside the active layer. The only potentially significant simplification is that $\boldsymbol{\gamma}^{III}$ can now be computed in advance, as reflected by this calculation lying outside of the main loop in Figure A.3. Based on these observations, we expect that our conclusions in the main text about the accuracy, efficiency, and robustness of these approaches are largely applicable to this alternative formulation of the problem as well.

A.7 Index of the DAE System in the Local Flux Problem

In this section, we justify the claim in the main text that the DAE system in the second block of Figure 2.2 is index 1. The algebraic equations in that system are the implicit fugacity

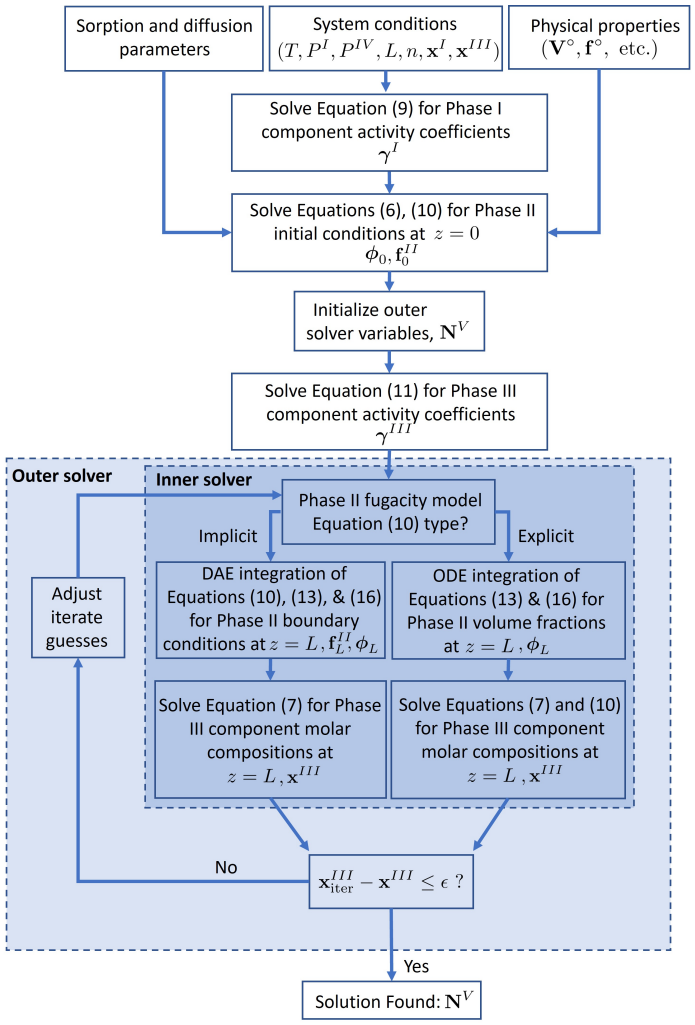


Figure A.3: Shooting algorithm for the local flux problem in Figure A.2 assuming a fixed permeate (Phase IV) composition. Different outer solvers may use slightly different termination criteria.

model, which must hold at every point in the membrane:

$$\mathbf{g}^{II}(\mathbf{f}^{II}(z), \boldsymbol{\phi}_{1:n}(z), T, P^{II}) = \mathbf{0}, \quad \forall z \in [0, L]. \tag{A.47}$$

By definition, this system is index 1 if we can derive an equivalent set of ODEs describing how $\mathbf{f}^{II}$ changes with respect to z by differentiating (Equation A.47) one time.

In the following derivations, we assume that $\mathbf{g}^{II}$ has been written as a function of $\boldsymbol{\phi}_{1:n}$ rather than $\boldsymbol{\phi}$ by using the fact that the volume fractions sum to one to explicitly eliminate the dependent membrane phase volume fraction, ϕ_m. Taking the total derivative with respect to z (and dropping the function arguments for brevity) results in

$$\frac{\partial \mathbf{g}^{II}}{\partial \mathbf{f}^{II}} \frac{\partial \mathbf{f}^{II}}{\partial z} + \frac{\partial \mathbf{g}^{II}}{\partial \boldsymbol{\phi}_{1:n}} \frac{\partial \boldsymbol{\phi}_{1:n}}{\partial z} = \mathbf{0}. \tag{A.48}$$

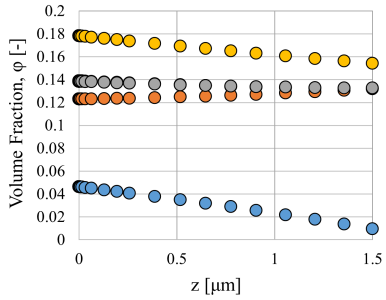
Solving for $\frac{\partial \mathbf{f}^{II}}{\partial z}$, we obtain the following ODEs for $\mathbf{f}^{II}$:

$$\frac{\partial \mathbf{f}^{II}}{\partial z} = -\left(\frac{\partial \mathbf{g}^{II}}{\partial \mathbf{f}^{II}}\right)^{-1} \frac{\partial \mathbf{g}^{II}}{\partial \boldsymbol{\phi}_{1:n}} \frac{\partial \boldsymbol{\phi}_{1:n}}{\partial z}. \tag{A.49}$$

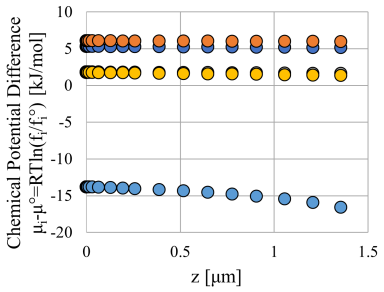
Therefore, the system is index 1 provided that the matrix $\frac{\partial \mathbf{g}^{II}}{\partial \mathbf{f}^{II}}$ is full rank everywhere along the solution trajectory. This of course depends on the function $\mathbf{g}^{II}$ and could fail in principle. However, this is exactly the condition that ensures that the fugacity model can be solved for $\mathbf{f}^{II}$ given any value of $\boldsymbol{\phi}_{1:n}$, which is required of any meaningful model (note that the inverse of $\frac{\partial \mathbf{g}^{II}}{\partial \mathbf{f}^{II}}$ was also used in the derivation of $\boldsymbol{\Gamma}$ for the FH, dual mode, and FHLM models).

A.8 Predicted Volume Fraction Profiles and Fluxes for the Approximation Methods

Figure A.6 shows the volume fraction profiles predicted by each approximation method for the five and nine component test cases with implicit fugacity model. For comparison, Figure A.4 and Figure A.5 show the correct profiles obtained using a high-order numerical

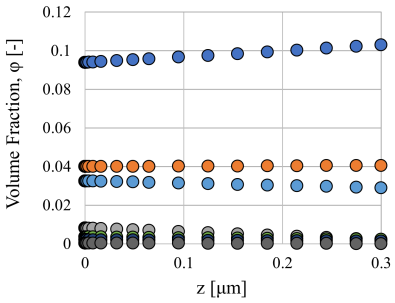


(a) Volume Fraction Profiles

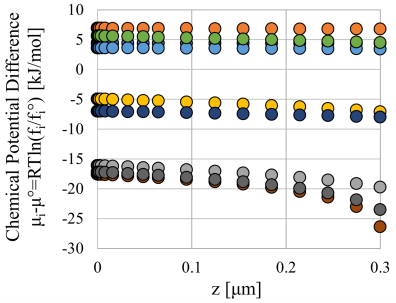


(b) Chemical Potential Profiles

Figure A.4: Volume fraction and chemical potential profiles through the membrane active layer for the five-component mixture with the Flory-Huggins-Langmuir fugacity model. Each line represents a different component. Some components overlap and the membrane phase volume fraction, ϕ_m, is excluded from this figure.



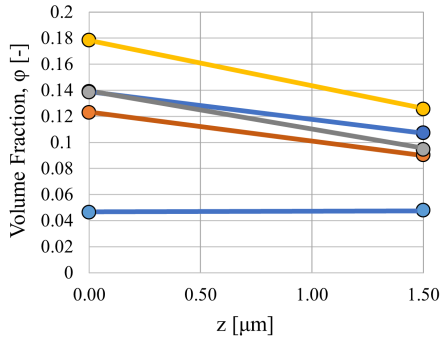
(a) Volume Fraction Profiles



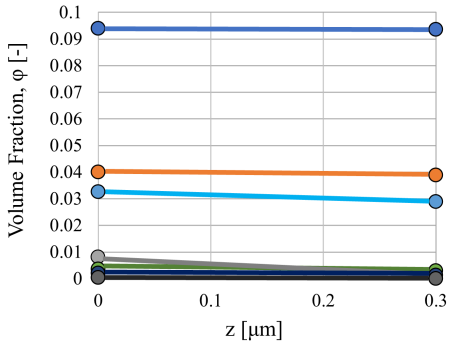
(b) Chemical Potential Difference Profiles

Figure A.5: Volume fraction and chemical potential profiles through the membrane active layer for the nine-component mixture with the Flory-Huggins-Langmuir fugacity model. Each line represents a different component. Some components overlap and the membrane phase volume fraction, ϕ_m, is excluded from this figure.

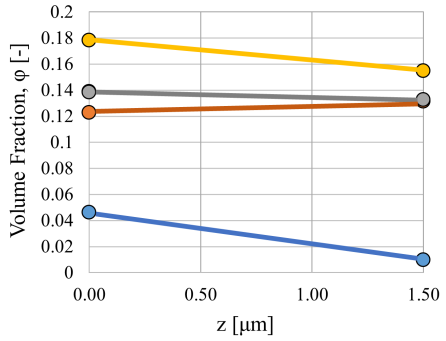
integrator with state-of-the-art error control, as in the inner-loop of the shooting algorithm. Similarly, Table Table A.1 shows the predicted and correct total molar fluxes for these test cases. As expected, the profiles and fluxes predicted using Fick's law exhibit large errors. Conversely, the ϕ-form and $\mathbf{f}$-form average approximation methods both predict profiles that are very close to correct, and the ϕ-form also produces relatively accurate fluxes.



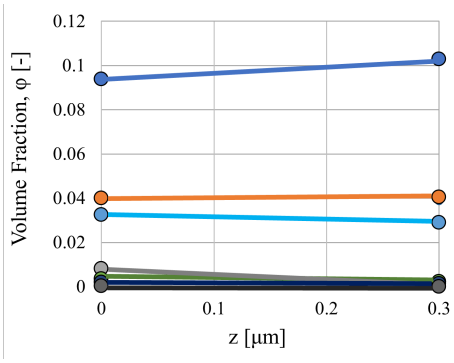
(a) Fick's law, five-component mixture



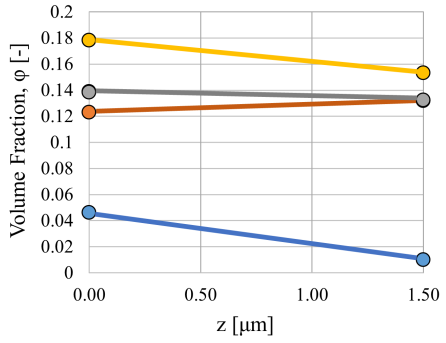
(b) Fick's law, nine-component mixture



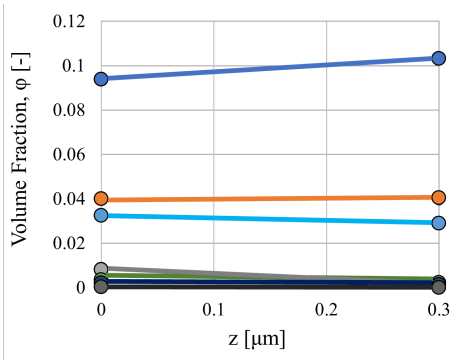
(c) ϕ-form average approximation, five-component mixture



(d) ϕ-form average approximation, nine-component mixture



(e) $\mathbf{f}$-form average approximation, five-component mixture



(f) $\mathbf{f}$-form average approximation, nine-component mixture

Figure A.6: Volume fraction profiles predicted by the approximation methods for five and nine component mixtures assuming the Flory-Huggins-Langmuir fugacity model. Each line represents a different component. Some components overlap and the membrane phase volume fraction, ϕ_m, is excluded from this figure.

Table A.1: Predicted total volumetric fluxes for the approximation methods for five and nine component test cases with implicit and explicit fugacity models.

Mixture and Fugacity Model	Total Volumetric Flux, N_{tot}^V, $\frac{\text{L}}{\text{m}^2\text{hr}}$			
	True Solution	Fick's Law	ϕ-form	$\mathbf{f}$-form
5C-FH	5.46	3.975	5.43	5.93
5C-FHLM	5.47	2.24	5.29	6.12
9C-FH	0.725	0.725	0.723	0.757
9C-FHLM	0.613	0.52	0.61	0.644

A.9 Fugacity Form Finite Difference Results with Fifty Nodes and Extended Time

Recalling the discussion near the end of subsection 2.5.5 in the main text, Figure A.7 shows the results of the $\mathbf{f}$-form FD method with $S = 50$ and a maximum solution time limit of 2.5 minutes. All points on the right-hand boundary of the figure correspond to simulations that reached the 2.5 minute limit without converging, but the data points are clipped at 100s to keep the axes consistent with the figures in the main text. Inspection of the iterates in these runs showed that, in all cases, there were variables with small negative values and/or imaginary parts caused by violations of the domains of the natural logarithms in this formulation. The other solution components were not far from the true solution, but the problematic components caused excessively slow convergence.

A.10 Impact of Solver Algorithm on Extended Robustness Comparisons

As discussed at the end of subsection 2.5.7 in the main text, Figure A.8 show that the robustness results are negligibly affected by the *fsolve* solver algorithm used.

A.11 Model Parameters

This section details all the parameters used throughout this study. Specifically, Table Table A.2 and Table A.3 give permeant physical property data. Then, Tables Table A.4–Table A.6 give all required fugacity model parameters. Lastly, Tables Table A.7 and Table A.8 give transport parameters for each permeant. All data was extracted from [14].

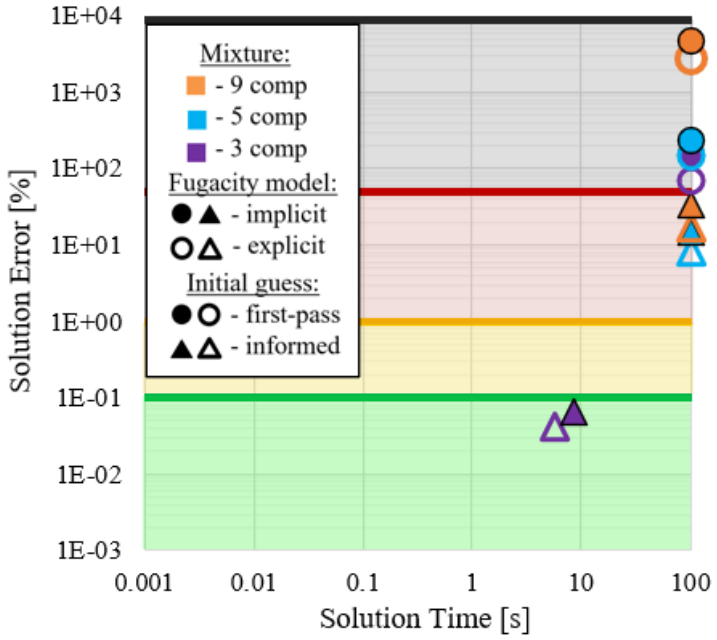
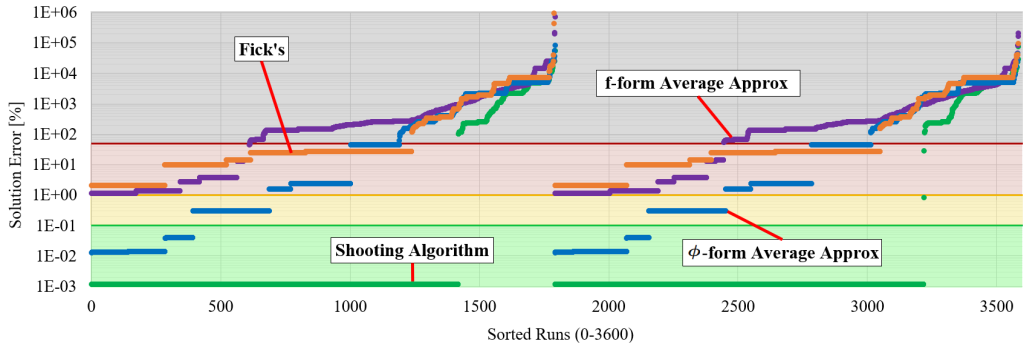


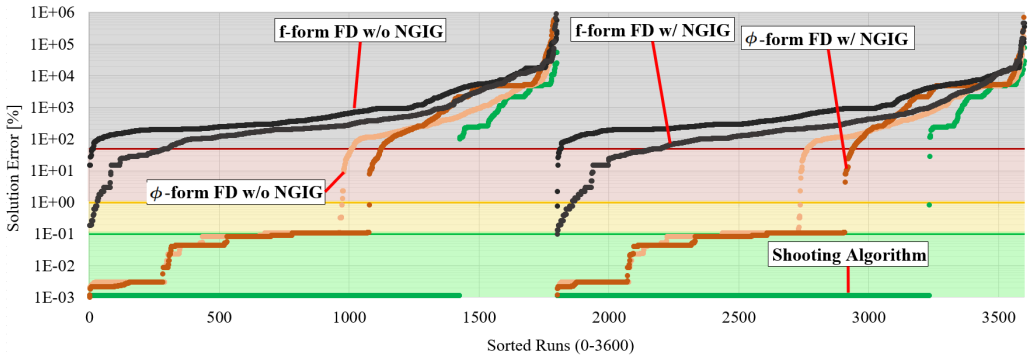
Figure A.7: Error (%) versus solution time (s) for the **f**-form finite difference method with $S = 50$. Note simulation time is 2.5 minutes and has been clipped at 100 seconds for the figure.

Table A.2: Physical properties of permeants.

Mixture Components	Vapor Pressure, P_i^{vap}	Molar Volume, V_i°
	torr	$\frac{\text{cm}^3}{\text{mol}}$
heptane	44.845	146.927
p-xylene	8.803	123.738
o-xylene	6.637	121.196
toluene	28.998	106.521
methylcyclohexane	46.596	128.123
1-methylnaphthalene	0.059	139.823
decalin	0.975	156.962
n-octane	14.805	163.420
iso-octane	49.087	165.552
tert-butlybenzene	2.115	155.529
1,3,5-triisopropylbenzene	0.035	240.069
iso-cetane	0.046	298.267



(a) Approximation Methods Robustness Comparison



(b) Finite Difference Methods Robustness Comparison

Figure A.8: Solution errors for all methods on 3600 test cases with random initial guesses. Runs 0–1800 are for the *trust-region-dogleg* solver algorithm and runs 1801–3600 are for the *Levenberg-Marquardt* solver algorithm. The results for each method are sorted in order of increasing error. Shaded regions correspond to the following error ranges: >50%, 1–50%, 0.1–1%, <0.1%. NGIG = nodal good initial guess. For the shooting algorithm, the solution error is often much less than 1E-3, but the values are clipped at 1E-3 for plotting.

Table A.3: Hansen solubility parameters of permeants [91]. Molecules in parentheses were determined as satisfactory substitutes for species that did not have recorded solubility parameters. These species include: decalin (average of isomers), *tert*-butlybenzene (n-isomer), 1,3,5-triisopropylbenzene (mesitylene), and *iso*-cetane (n-isomer).

Mixture Components	Dispersion, $\delta_{D.i}$	Polarity, $\delta_{P,i}$	Hydrogen Bonding, $\delta_{H,i}$
	MPa$^{1/2}$		
heptane	15.3	0	0
p-xylene	17.6	1	3.1
o-xylene	14.1	1	3.1
toluene	18	1.4	2
methylcyclohexane	16	0	1
1-methylnaphthalene	20.6	0.8	4.7
decalin	18.4	0	0
n-octane	15.5	0	0
iso-octane	14.1	0	0
tert-butlybenzene	17.4	0.1	1.1
1,3,5-triisopropylbenzene	18	0	0.6
iso-cetane	16.3	0	0

Table A.4: Flory-Huggins fugacity model parameters

Membrane Material	Mixture Components	χ_{im}
PIM-1	toluene	0.648
	heptane	0.826
	p-xylene	0.642
	o-xylene	0.560
	iso-cetane	0.698
SBAD-1	toluene	0.871
	methylcyclohexane	1.672
	1-methylnaphthalene	0.705
	decalin	2.783
	n-octane	1.163
	iso-octane	3.049
	tert-butlybenzene	1.648
	1,3,5-triisopropylbenzene	2.500
	iso-cetane	3.130

Table A.5: Dual mode sorption fugacity model parameters.

Membrane Material	Mixture Components	C_i^H	b_i	k_i
		$\frac{\text{m}^3 \text{ penetrant}}{\text{m}^3 \text{ polymer}}$	torr^{-1}	$\frac{\text{m}^3 \text{ penetrant}}{\text{m}^3 \text{ polymer torr}}$
PIM-1	toluene	0.770	0.0566	0.0443
	heptane	0.679	0.595	0.00401
	p-xylene	0.858	0.618	0.117
	o-xylene	0.844	0	0.538
	iso-cetane	0.812	146	14.3
SBAD-1	toluene	0.314	0.141	0.0138
	methylcyclohexane	0.0730	0.180	0.00121
	1-methylnaphthalene	0.405	0	18.5
	decalin	0.00759	0	0.0263
	n-octane	0.0510	0.0000195	0.0148
	iso-octane	0.0112	0.202	0.000158
	tert-butlybenzene	0.0158	0	0.0525
	1,3,5-triisopropylbenzene	0.0137	6.66	0.792
	iso-cetane	0.00801	67.6	0.258

Table A.6: Flory-Huggins-Langmuir fugacity model parameters.

Membrane Material	Mixture Components	C_i^H	b_i	χ_{im}
		$\frac{\text{m}^3 \text{ penetrant}}{\text{m}^3 \text{ polymer}}$	torr^{-1}	
PIM-1	toluene	0.770	0.590	0.726
	heptane	0.679	0.726	1.65
	p-xylene	0.858	2.48	0.724
	o-xylene	0.844	2.99	0.571
	iso-cetane	0.812	331	0.891
SBAD-1	toluene	0.314	0.472	1.09
	methylcyclohexane	0.0730	0.278	2.32
	1-methylnaphthalene	0.405	88.8	0.775
	decalin	0.00759	2.12	3.06
	n-octane	0.0510	0.01645	1.25
	iso-octane	0.0112	0.168	3.75
	tert-butlybenzene	0.0158	0.258	1.73
	1,3,5-triisopropylbenzene	0.0137	150	2.86
	iso-cetane	0.00801	110	3.64

Table A.7: Maxwell-Stefan diffusivities for each fugacity model.

Membrane Material	Mixture Components	$Đ_{im}^{V,\text{FH}}$	$Đ_{im}^{V,\text{DMS}}$	$Đ_{im}^{V,\text{FHLM}}$
		$\frac{\mu\text{m}^2}{\text{s}}$		
PIM-1	toluene	50.30	27.75	25.14
	heptane	184.24	119.67	119.06
	p-xylene	35.05	18.89	15.52
	o-xylene	16.21	5.61	6.85
	iso-cetane	2.81	1.17	1.17
SBAD-1	toluene	16.77	13.15	12.60
	methylcyclohexane	5.57	4.95	5.29
	1-methylnaphthalene	0.053	0.035	0.034
	decalin	1.66	1.57	1.78
	n-octane	5.71	6.25	6.37
	iso-octane	6.99	6.55	6.86
	tert-butlybenzene	1.40	1.34	1.488
	1,3,5-triisopropylbenzene	0.069	0.059	0.069
	iso-cetane	0.590	0.496	0.59

Table A.8: Fickian diffusivities for each fugacity model.

Membrane Material	Mixture Components	$D_{\text{Fick},im}^{V,\text{FH}}$	$D_{\text{Fick},im}^{V,\text{DMS}}$	$D_{\text{Fick},im}^{V,\text{FHLM}}$
		$\frac{\mu\text{m}^2}{\text{s}}$		
PIM-1	toluene	15.25	94.35	35.81
	heptane	72.18	876.70	925.62
	p-xylene	10.92	80.88	27.88
	o-xylene	4.46	26.67	8.51
	iso-cetane	1.12	5.08	3.55
SBAD-1	toluene	6.51	31.73	16.85
	methylcyclohexane	3.70	10.57	10.70
	1-methylnaphthalene	0.018	0.078	0.032
	decalin	1.43	1.62	1.89
	n-octane	2.98	7.72	3.68
	iso-octane	6.23	13.75	12.87
	tert-butlybenzene	0.93	1.50	1.06
	1,3,5-triisopropylbenzene	0.057	0.075	0.088
	iso-cetane	0.530	0.676	0.88

APPENDIX B

A FRAMEWORK FOR PREDICTING THE FRACTIONATION OF COMPLEX LIQUID FEEDS VIA POLYMER MEMBRANES SUPPLEMENTAL INFORMATION

This chapter provides all the necessary supplemental information for chapter 3 and references to the paper titled "A Framework for Predicting the Fractionation of Complex Liquid Feeds via Polymer Membranes" [14].

B.1 Derivation of Multicomponent Flory-Huggins

The Flory-Huggins lattice theory may be applied to a multicomponent system as [88]

$$\frac{\Delta G_m}{RT} = \frac{\Delta H_m - T\Delta S_m}{RT} = \sum_{i=1}^{n+1}\left(n_i^m \ln(\phi_i) + \sum_{j=i+1}^{n+1} \chi_{ij} n_i^m \phi_j\right), \tag{B.1}$$

where the $(n+1)^{\text{st}}$ component is the membrane component itself. From there, we may convert mole terms to volume fraction terms using the following relationship

$$\phi_i = \frac{V_i n_i^m}{\sum_{j=1}^{n+1} V_j \phi_j} \tag{B.2}$$

Fundamentally, χ_{ij} is dependent on the composition of the mixture. However, if composition dependence is assumed, then extensive multicomponent sorption experiments are required to determine the [89]. Additionally, the differentiation of (Equation B.1) will become much more involved. Here, we focus on a key approximation, which is the assumption that is concentration independent, thus enabling utilization of experimental measurements of this value. Substituting (Equation B.2) into (Equation B.1) and differentiating it

yields

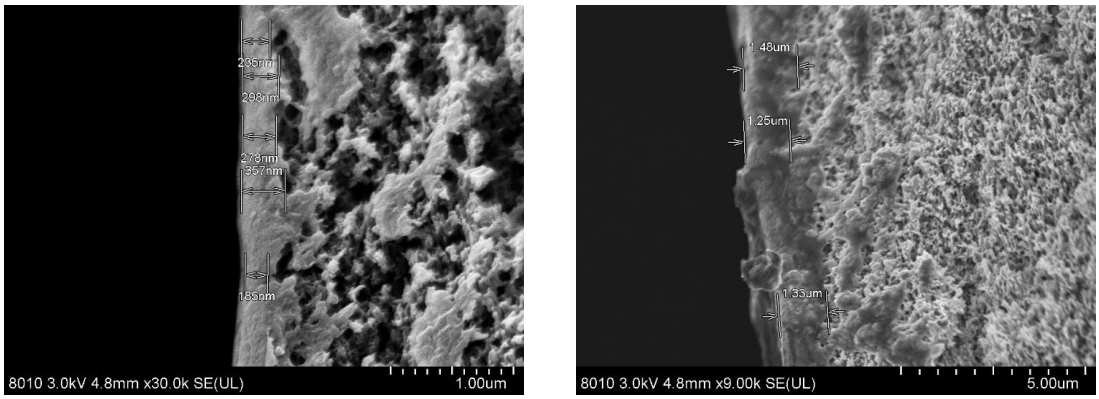
$$\begin{aligned}\ln(a_i) &= \ln\left(\frac{f_i}{f_i^\circ(T,P)}\right) = \frac{\Delta\mu_i}{RT} = \frac{\partial}{\partial n_i}\left(\frac{\Delta G_m}{RT}\right) \\ &= \ln(\phi_i) + 1 - \sum_{j=1}^{n+1}\frac{V_i^\circ}{V_j^\circ}\phi_j + \left(\sum_{j=1}^{i-1}\chi_{ji}\phi_j\frac{V_i^\circ}{V_j^\circ} + \sum_{j=i+1}^{n+1}\chi_{ij}\phi_j\right)(1-\phi_i) \\ &\quad - \sum_{\substack{j=1\\ j\neq i}}^{n}\sum_{\substack{k=j+1\\ k\neq i}}^{n+1}\chi_{jk}\frac{V_i^\circ}{V_j^\circ}\phi_j\phi_k.\end{aligned} \tag{B.3}$$

The above expression has been verified as correct by taking finite differences of (Equation B.1) and comparing the evaluated values to those given by (Equation B.3). The values matched within the order of magnitude of the finite differences ste*p*-size.

B.2 Tables and Figures Cited in Manuscript

Table B.1: Physical properties of polymers and solvents at 25 °C and atmospheric pressure [143, 144, 145, 146]. Vapor pressure for 1-methylnaphthalene, decalin (50/50 mol% cis + trans mix), *tert*-butylbenzene, 1,3,5-triisopropylbenzene and *iso*-cetane were obtained from Aspen Plus using the PC-SAFT EoS.

	Density	**Vapor Pressure**	**Molar Mass**	**Molar Volume**
Component	**g/mL**	**torr**	**g/mol**	**cm3/mol**
toluene	0.865	28.998	92.141	106.521
methylcyclohexane	0.766	46.596	98.188	128.123
1-methylnaphthalene	1.017	0.059	142.2	139.823
decalin	0.881	0.975	138.253	156.962
n-octane	0.699	14.805	114.231	163.42
iso-octane	0.69	49.087	114.231	165.552
tert-butylbenzene	0.863	2.115	134.221	155.529
1,3,5-triisopropylbenzene	0.855	0.0352	204.356	240.069
iso-cetane	0.757	0.0458	226.446	293.267
heptane	0.682	44.854	100.204	146.927
p-xylene	0.858	8.803	106.167	123.738
o-xylene	0.876	6.637	106.167	121.196
PIM-1 (skeleton) =	1.4			
SBAD-1 (skeleton) =	1.29			



(a) SBAD-1 (b) PIM-1

Figure B.1: SEM images showing the approximate thickness of (a) SBAD-1 membrane film thickness of around 300 nm and (b) PIM-1 membrane film thickness of around 1.5 microns

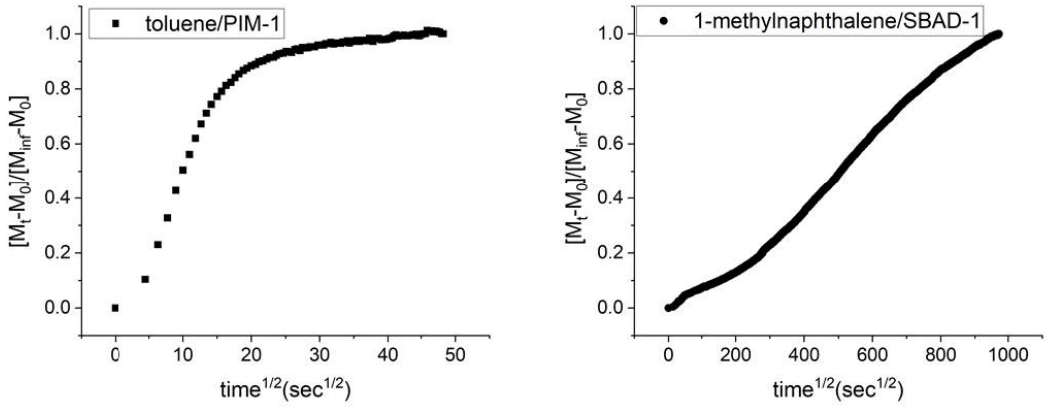


Figure B.2: Kinetic sorption of toluene in PIM-1 at toluene activity = 0.7 (left) and 1-methylnaphthalene in SBAD-1 at 1-methylnaphthalene activity = 0.7 (right).

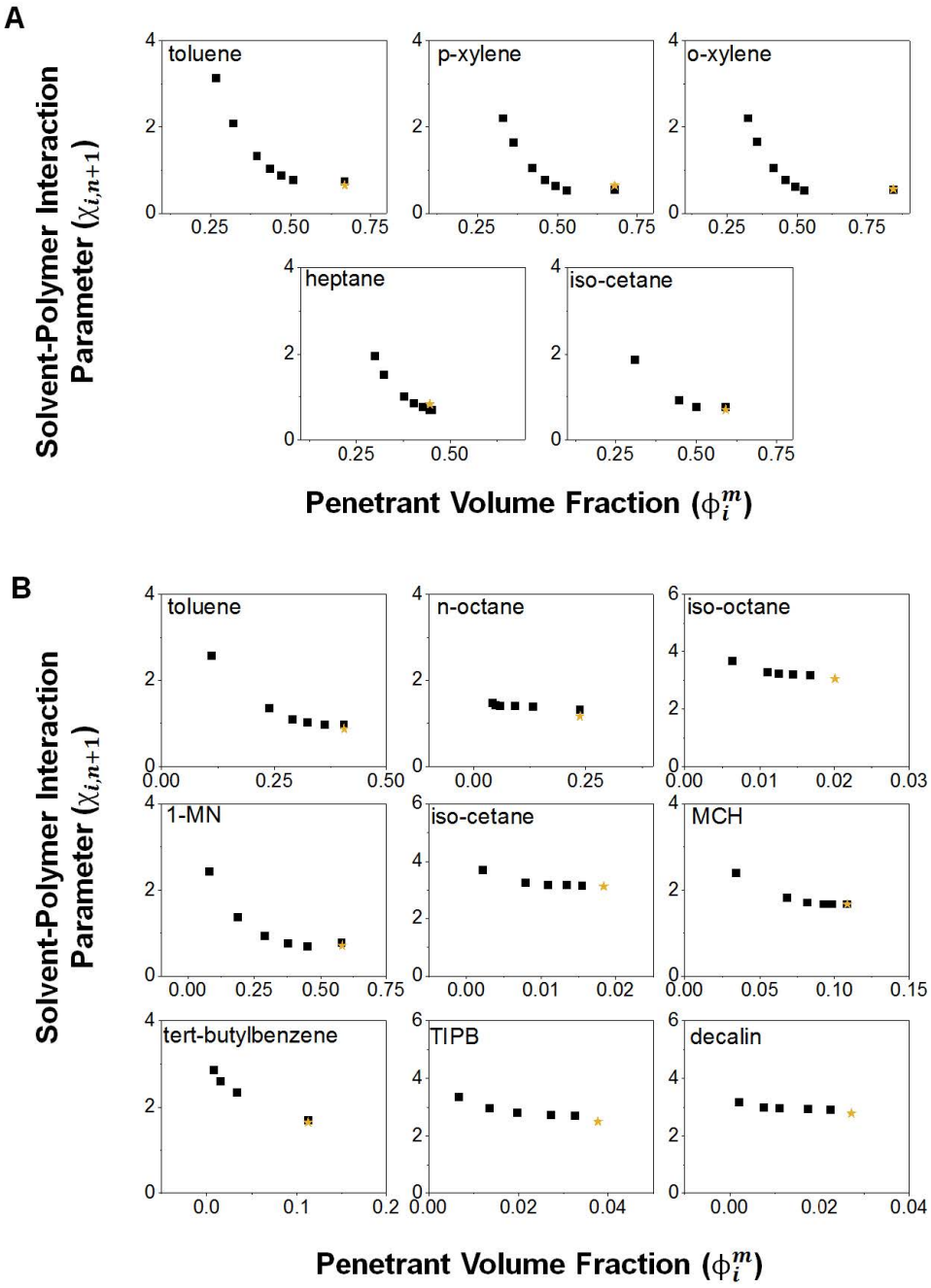


Figure B.3: Calculation of Flory-Huggins solvent-polymer interaction parameter χ_{im} using the FH model for composition-dependent interaction parameters, $\ln(a_i) = \ln(\phi_i) + (1 - \phi_i) - (1 - \phi_i)\frac{V_i}{V_m} + \chi_{im}(1 - \phi_i)^2 + \phi_i(1 - \phi_i)\frac{\partial \chi_{im}}{\partial \phi_i}$ [39], and measured sorption isotherms for PIM-1 (A) and SBAD-1 (B) in single penetrant systems. $V_m \gg V_i$ was assumed. Here, χ_{im} is not fixed at a constant value and is allowed to vary with activity of the penetrant.

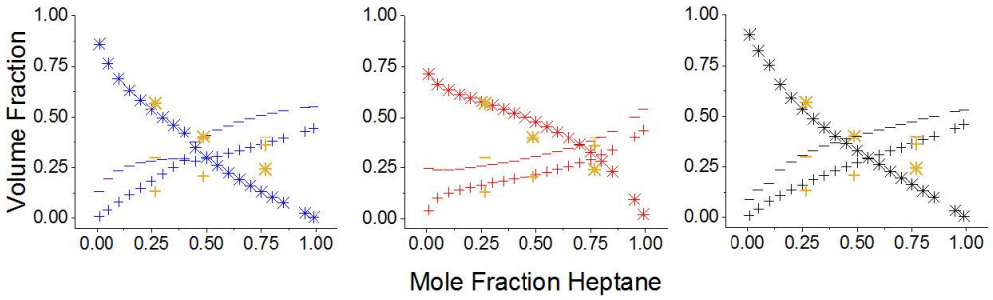


Figure B.4: Predicted multicomponent sorption of heptane/*o*-xylene mixtures in PIM-1 according to Flory-Huggins (left, blue), Dual-mode (middle, red) and Langmuir + Flory-Huggins (right, black) models compared with experimental measurements (yellow). Legend: heptane, + ; *o*-xylene, $*$; polymer, –

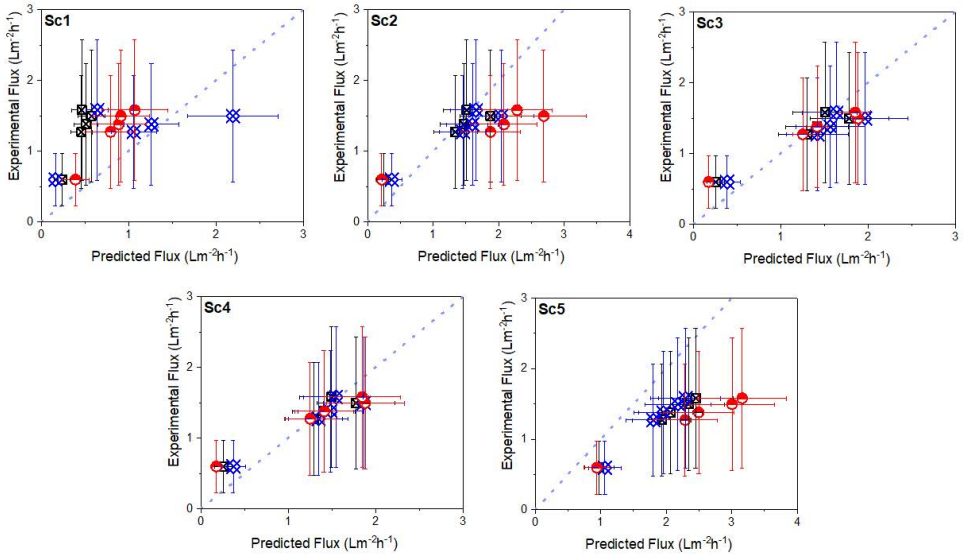


Figure B.5: Partial flux predictions for **Separation 1** via PIM-1 at varying combinations of sorption and diffusion assumptions. Markers indicate Dual-mode (red), Flory-Huggins (blue) and Langmuir + Flory-Huggins (black) sorption models. X-axis error bars are propagated from error in penetrant-polymer diffusivities (from unary flux measurements) and y-axis error bars are propagated from error in mixture separation flux measurements.

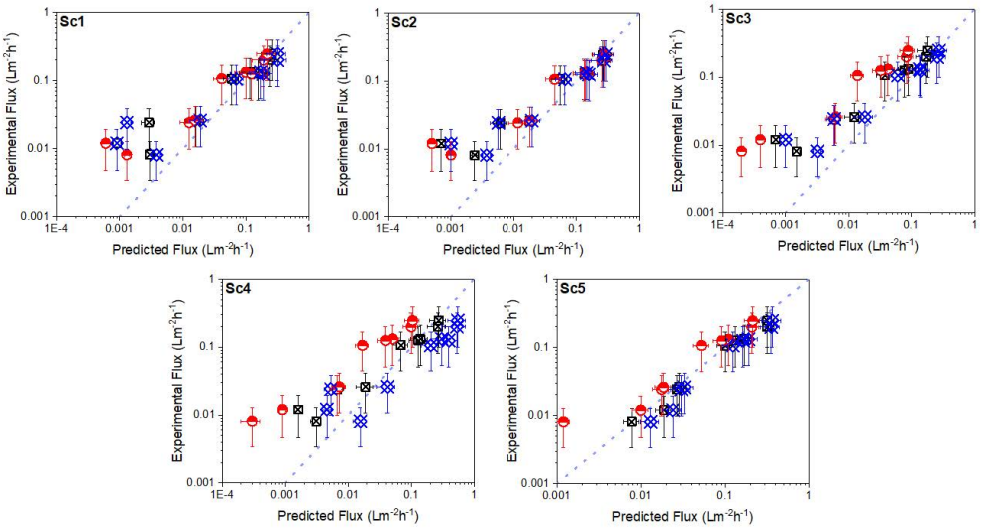


Figure B.6: Partial flux predictions for **Separation 2** via PIM-1 at varying combinations of sorption and diffusion assumptions. Markers indicate Dual-mode (red), Flory-Huggins (blue) and Langmuir + Flory-Huggins (black) sorption models. X-axis error bars are propagated from error in penetrant-polymer diffusivities (from unary flux measurements) and y-axis error bars are propagated from error in mixture separation flux measurements.

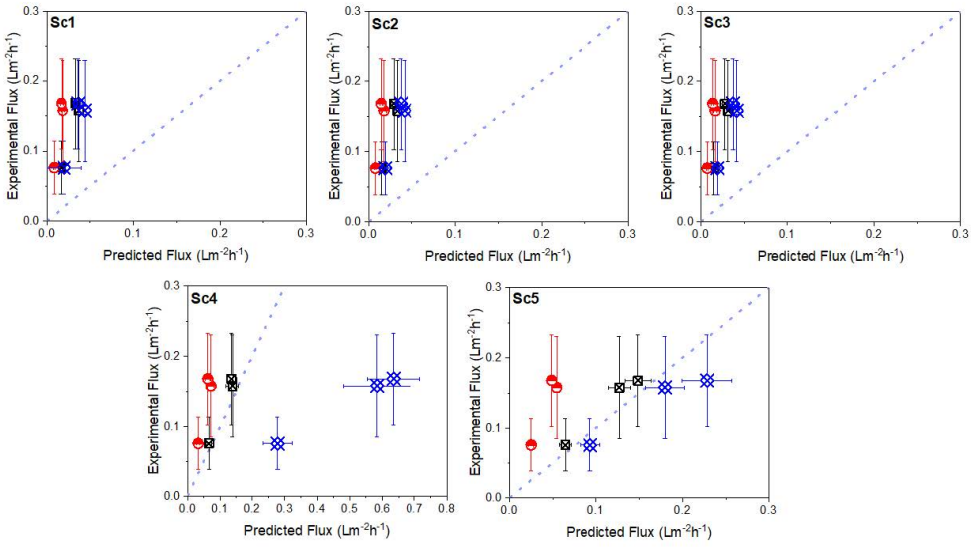


Figure B.7: Partial flux predictions for **Separation 3** via PIM-1 at varying combinations of sorption and diffusion assumptions. Markers indicate Dual-mode (red), Flory-Huggins (blue) and Flory-Huggins + Langmuir (black) sorption models. X-axis error bars are propagated from error in penetrant-polymer diffusivities (from unary flux measurements) and y-axis error bars are propagated from error in mixture separation flux measurements.

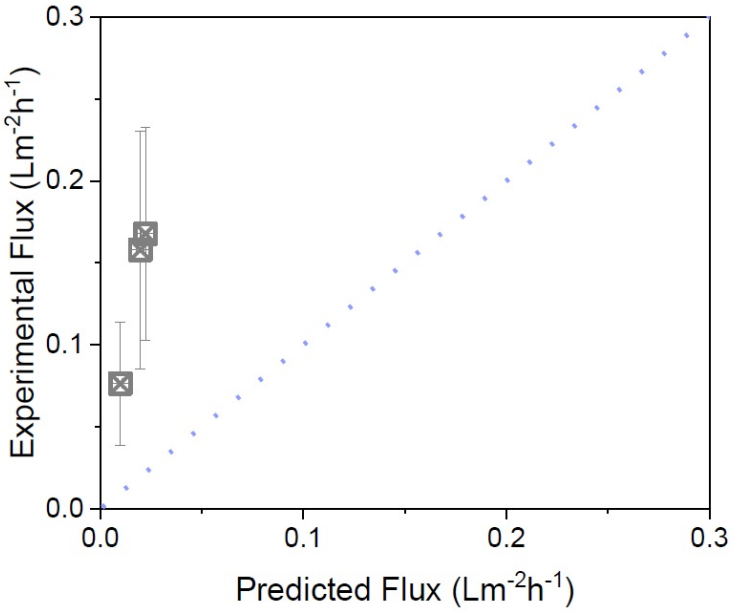


Figure B.8: Separation 3 (via SBAD-1) partial fluxes predicted using FH-LM and cross-92 diffusivities ($Đ_{ij}$) fit to match permeate compositions.

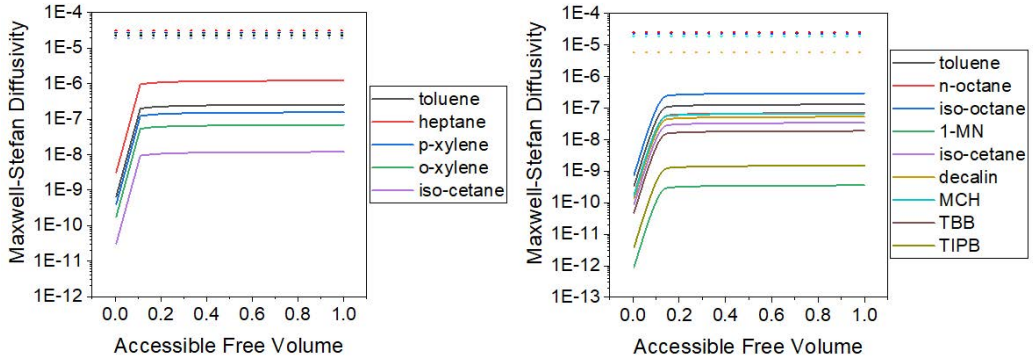


Figure B.9: Free-volume theory-based prediction of diffusivity, $Đ^{V}_{im}$, as it varies with accessible 96 free volume indicated by solid lines for PIM-1 (left) and SBAD-1 (right) assuming $B = 0.03$. 97 Dotted lines are the self-diffusivities of molecules (excluding 1-methylnaphthalene, *tert*-98 butylbenzene and 1,3,5-triisopropylbenzene) and represent an upper limit on diffusivity in the polymers [98].

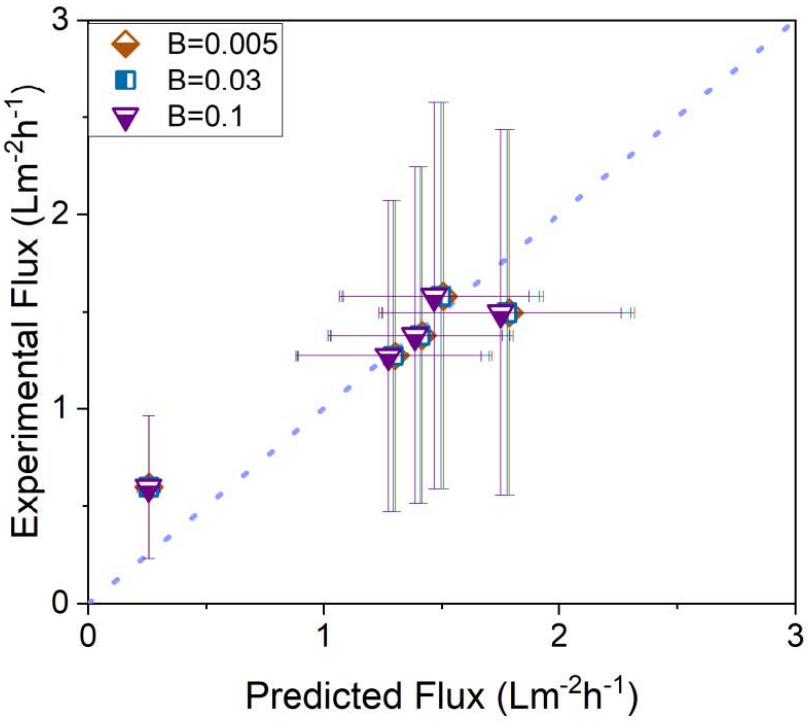


Figure B.10: **Separation 1** (via PIM-1) predicted using FH-LM, Vignes diffusion coupling, and free volume theory with varying B = i) 0.005, ii) 0.03 and iii) 0.1.

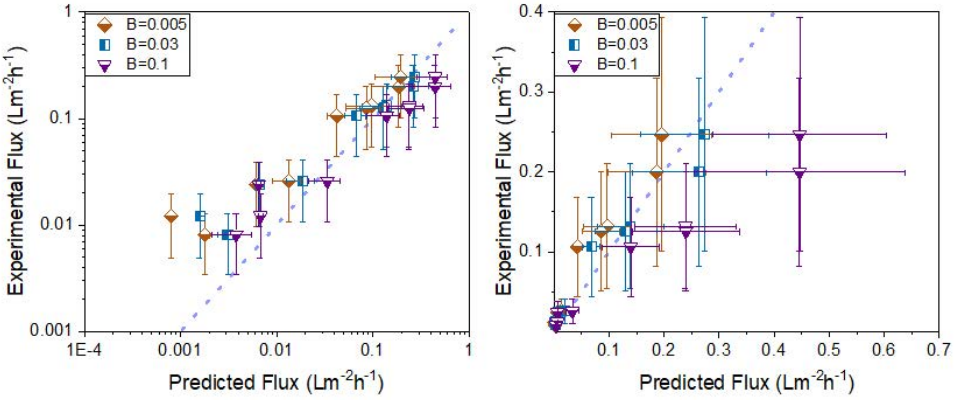


Figure B.11: Separation 2 (via SBAD-1) predicted using FH-LM, Vignes diffusion coupling and free volume theory with varying B = i) 0.005, ii) 0.03 and iii) 0.1. Left plot has log-log scale while right plot in linear.

B.3 Raw Data

Table B.2: Experimentally measured sorption isotherms of hydrocarbons in PIM-1 at 25°C.

Activity	Uptake	Error	Uptake	Error	Uptake	Error
p/psat	g solvent / g polymer					
	toluene		*p*-xylene		*o*-xylene	
0.05	0.225	0.00272	0.302	0.00607	0.301	0.00583
0.1	0.294	0.00431	0.348	0.00453	0.347	0.00647
0.25	0.403	0.00592	0.445	0.00336	0.445	0.00709
0.4	0.476	0.00696	0.526	0.00822	0.528	0.00929
0.55	0.547	0.00746	0.595	0.00494	0.605	0.00613
0.7	0.636	0.00757	0.685	0.00403	0.692	0.00644
1	1.248	0.00926	1.297	0.00416	3.339	0.02220
	heptane		*iso*-cetane			
0.05	0.209	0.00238				
0.1	0.235	0.00483	0.244	0.01326		
0.25	0.297	0.00539				
0.4	0.331	0.00589	0.439	0.00917		
0.55	0.363	0.00866				
0.7	0.402	0.00972	0.546	0.05514		
1	0.392	0.03200	0.785	-		

Table B.3: Experimentally measured sorption isotherms of hydrocarbons in SBAD-1 at 25°C.

Activity	Uptake	Error	Uptake	Error	Uptake	Error
p/psat	g solvent / g polymer					
	toluene		n-octane		*iso*-octane	
0.1	0.08464	0.00371	0.024290	0.00608	0.00342	0.00052
0.3	0.21028	0.00315	0.027619	0.00340	0.00597	0.00067
0.5	0.27565	0.00407	0.034537	0.00522	0.00681	0.00057
0.7	0.32198	0.00363	0.054731	0.01057	0.00784	0.00089
0.85	0.37918	0.00709	0.082895	0.01146	0.00911	0.00137
1	0.51419	-	0.167621	0.01367	-	-
	1-methylnaphthalene		*iso*-cetane		methylcyclohexane	
0.1	0.06908	-	0.001312	0.00016	0.02109	0.00227
0.3	0.18240	0.06233	0.004700	0.00165	0.04335	0.00301
0.5	0.31957	0.08424	0.006487	0.00258	0.05283	0.00353
0.7	0.47995	0.02836	0.007953	0.00308	0.06100	0.00225
0.85	0.65000	0.05000	0.009234	0.00362	0.06500	0.00744
1	1.09844	-	-	-	0.07236	0.00236
	tert-butylbenzene		1,3,5-triisopropylbenzene		decalin	
0.1	0.00549	0.00050	0.004568	0.00191	0.00145	0.00022
0.3	0.01060	0.00079	0.009098	0.00209	0.00518	0.00069
0.5	0.02363	0.00309	0.013380	0.00096	0.00764	0.00180
0.7	-	-	0.018540	0.00291	0.01203	0.00210
0.85	-	-	0.022217	0.00501	0.01574	0.00208
1	0.08468	-	-	-	-	-

Table B.4: Flory-Huggins sorption isotherm parameters of hydrocarbons in PIM-1 and SBAD-1 at 25°C. Values for *iso*-octane, decalin, 1,3,5-triisopropylbenzene and *iso*-cetane are estimates. Upper bound and lower bound predictions are calculated from experimental error.

		χ_{im}		
		Average	Upper bound	Lower bound
PIM-1	toluene	0.648	0.647	0.649
	heptane	0.826	0.806	0.849
	p-xylene	0.642	0.641	0.642
	o-xylene	0.56	0.559	0.56
	iso-cetane	0.698	-	-
SBAD-1	toluene	0.871	0.827	0.93
	iso-octane	3.049	2.949	3.241
	n-octane	1.163	1.129	1.202
	methylcyclohexane	1.672	1.652	1.693
	decalin	2.783	2.667	2.86
	tert-butylbenzene	1.648	-	-
	1-methylnaphthalene	0.705	0.678	0.743
	1,3,5-triisopropylbenzene	2.5	2.339	2.765
	iso-cetane	3.13	2.912	3.61

Table B.5: Average Dual-mode sorption isotherm parameters of hydrocarbons in PIM-1 and SBAD-1 at 25°C. Values for *iso*-octane, decalin, 1,3,5-triisopropylbenzene and *iso*-cetane are estimates. Upper bound and lower bound predictions are calculated from experimental isotherm error.

	Average			
		$C_i^H \left(\frac{\text{cc solvent}}{\text{cc polymer}}\right)$	(torr^{-1})	$k_i \left(\frac{\text{cc solvent}}{\text{cc polymer}}\right)$
PIM-1	toluene	7.70E-01	5.66E-02	4.43E-02
	heptane	6.79E-01	5.95E-01	4.01E-03
	p-xylene	8.58E-01	6.18E-01	1.17E-01
	o-xylene	8.44E-01	0.00E+00	5.38E-01
	iso-cetane	8.12E-01	1.46E+02	1.43E+01
SBAD-1	toluene	3.14E-01	1.41E-01	1.38E-02
	iso-octane	1.12E-02	2.02E-01	1.58E-04
	n-octane	5.10E-02	1.90E-05	1.48E-02
	methylcyclohexane	7.30E-02	1.80E-01	1.21E-03
	decalin (average of isomers)	7.59E-03	0.00E+00	2.63E-02
	tert-butylbenzene	1.58E-02	0.00E+00	5.25E-02
	1-methylnaphthalene	4.05E-01	0.00E+00	1.85E+01
	1,3,5-triisopropylbenzene	1.37E-02	6.66E+01	7.92E-01
	iso-cetane	8.01E-03	6.76E+01	2.58E-01

Table B.6: Upper bound Dual-mode sorption isotherm parameters of hydrocarbons in PIM-1 and SBAD-1 at 25°C. Values for *iso*-octane, decalin, 1,3,5-triisopropylbenzene and *iso*-cetane are estimates. Upper bound and lower bound predictions are calculated from experimental isotherm error.

	Upper Bound			
		C_i^H $\left(\frac{\text{cc solvent}}{\text{cc polymer}}\right)$	(torr^{-1})	k_i $\left(\frac{\text{cc solvent}}{\text{cc polymer}}\right)$
PIM-1	toluene	7.82E-01	6.56E-02	4.33E-02
	heptane	6.92E-01	5.65E-01	4.79E-03
	p-xylene	8.72E-01	6.48E-01	1.15E-01
	o-xylene	8.59E-01	0.00E+00	5.42E-01
	iso-cetane	8.29E-01	1.66E+02	1.45E+01
SBAD-1	toluene	3.18E-01	8.75E-02	1.69E-02
	iso-octane	1.24E-02	2.11E-01	1.78E-04
	n-octane	5.72E-02	1.31E-05	1.65E-02
	methylcyclohexane	7.81E-02	1.91E-01	1.26E-03
	decalin (average of isomers)	8.60E-03	0.00E+00	3.05E-02
	tert-butylbenzene	1.70E-02	0.00E+00	5.34E-02
	1-methylnaphthalene	5.12E-01	0.00E+00	2.12E+01
	1,3,5-triisopropylbenzene	1.69E-02	6.49E+01	9.24E-01
	iso-cetane	1.08E-02	5.79E+01	3.77E-01

Table B.7: Lowe bound bound Dual-mode sorption isotherm parameters of hydrocarbons in PIM-1 and SBAD-1 at 25°C. Values for *iso*-octane, decalin, 1,3,5-triisopropylbenzene and *iso*-cetane are estimates. Upper bound and lower bound predictions are calculated from experimental isotherm error.

	Lower Bound			
		C_i^H $\left(\frac{\text{cc solvent}}{\text{cc polymer}}\right)$	(torr^{-1})	k_i $\left(\frac{\text{cc solvent}}{\text{cc polymer}}\right)$
PIM-1	toluene	7.59E-01	4.58E-02	4.59E-02
	heptane	6.67E-01	6.26E-01	3.24E-03
	p-xylene	8.45E-01	5.86E-01	1.19E-01
	o-xylene	8.29E-01	0.00E+00	5.33E-01
	iso-cetane	7.95E-01	1.25E+02	1.41E+01
SBAD-1	toluene	3.09E-01	1.97E-01	1.13E-02
	iso-octane	9.91E-03	2.12E-01	1.32E-04
	n-octane	4.47E-02	0.00E+00	1.31E-02
	methylcyclohexane	6.79E-02	1.68E-01	1.17E-03
	decalin (average of isomers)	6.58E-03	0.00E+00	2.21E-02
	tert-butylbenzene	1.47E-02	0.00E+00	5.15E-02
	1-methylnaphthalene	2.98E-01	0.00E+00	1.65E+01
	1,3,5-triisopropylbenzene	1.06E-02	6.82E+01	6.62E-01
	iso-cetane	5.20E-03	9.41E+01	1.40E-01

Table B.8: Average Flory-Huggins-Langmuir sorption isotherm parameters of hydrocarbons in PIM-1 and SBAD-1 at 25°C. Values for *iso*-octane, decalin, 1,3,5-triisopropylbenzene and *iso*-cetane are estimates. Upper bound and lower bound predictions are calculated from experimental isotherm error.

	Average			
		$C_i^H \left(\frac{\text{cc solvent}}{\text{cc polymer}}\right)$	(torr^{-1})	χ_{im}
PIM-1	toluene	7.70E-01	5.90E-01	7.26E-01
	heptane	6.79E-01	7.26E-01	1.65E+00
	p-xylene	8.58E-01	2.48E+00	7.24E-01
	o-xylene	8.44E-01	2.99E+00	5.71E-01
	iso-cetane	8.12E-01	3.31E+02	8.91E-01
SBAD-1	toluene	3.14E-01	4.72E-01	1.09E+00
	iso-octane	1.12E-02	1.68E-01	3.75E+00
	n-octane	5.10E-02	1.64E-02	1.25E+00
	methylcyclohexane	7.30E-02	2.78E-01	2.32E+00
	decalin (average of isomers)	7.59E-03	2.12E+00	3.06E+00
	tert-butylbenzene	1.58E-02	2.58E-01	1.73E+00
	1-methylnaphthalene	4.05E-01	8.88E+01	7.75E-01
	1,3,5-triisopropylbenzene	1.37E-02	1.50E+02	2.86E+00
	iso-cetane	8.01E-03	1.11E+02	3.64E+00

Table B.9: Upper bound Flory-Huggins-Langmuir sorption isotherm parameters of hydrocarbons in PIM-1 and SBAD-1 at 25°C. Values for *iso*-octane, decalin, 1,3,5-triisopropylbenzene and *iso*-cetane are estimates. Upper bound and lower bound predictions are calculated from experimental isotherm error.

	Upper Bound			
		$C_i^H \left(\frac{\text{cc solvent}}{\text{cc polymer}}\right)$	(torr^{-1})	χ_{im}
PIM-1	toluene	7.82E-01	5.92E-01	7.25E-01
	heptane	6.92E-01	6.98E-01	1.44E+00
	p-xylene	8.72E-01	2.46E+00	7.25E-01
	o-xylene	8.59E-01	2.98E+00	5.70E-01
	iso-cetane	8.29E-01	3.82E+02	8.99E-01
SBAD-1	toluene	3.18E-01	4.15E-01	9.84E-01
	iso-octane	1.24E-02	1.73E-01	3.65E+00
	n-octane	5.72E-02	4.03E-02	1.21E+00
	methylcyclohexane	7.81E-02	3.16E-01	2.34E+00
	decalin (average of isomers)	8.60E-03	2.38E+00	2.93E+00
	tert-butylbenzene	1.70E-02	3.99E-01	1.74E+00
	1-methylnaphthalene	5.12E-01	7.98E+01	7.46E-01
	1,3,5-triisopropylbenzene	1.69E-02	1.32E+02	2.69E+00
	iso-cetane	1.08E-02	1.43E+02	3.44E+00

Table B.10: Lower bound Flory-Huggins-Langmuir sorption isotherm parameters of hydrocarbons in PIM-1 and SBAD-1 at 25°C. Values for *iso*-octane, decalin, 1,3,5-triisopropylbenzene and *iso*-cetane are estimates. Upper bound and lower bound predictions are calculated from experimental isotherm error.

	Lower Bound			
		$C_i^H \left(\frac{\text{cc solvent}}{\text{cc polymer}}\right)$	(torr^{-1})	χ_{im}
PIM-1	toluene	7.59E-01	5.87E-01	7.26E-01
	heptane	6.67E-01	7.64E-01	2.04E+00
	p-xylene	8.45E-01	2.51E+00	7.23E-01
	o-xylene	8.29E-01	2.99E+00	5.71E-01
	iso-cetane	7.95E-01	2.85E+02	8.83E-01
SBAD-1	toluene	3.09E-01	5.61E-01	1.25E+00
	iso-octane	9.91E-03	2.27E-01	4.09E+00
	n-octane	4.47E-02	0.00E+00	1.29E+00
	methylcyclohexane	6.79E-02	2.41E-01	2.31E+00
	decalin (average of isomers)	6.58E-03	1.24E+00	3.12E+00
	tert-butylbenzene	1.47E-02	1.45E-01	1.72E+00
	1-methylnaphthalene	2.98E-01	1.07E+02	8.13E-01
	1,3,5-triisopropylbenzene	1.06E-02	2.34E+02	3.17E+00
	iso-cetane	5.20E-03	1.38E+02	4.21E+00

Table B.11: Experimentally measured liquid hydrocarbon fluxes in PIM-1 at 22°C.

Δp	Flux	Error	Flux	Error	Flux	Error
bar	Lm^{-2}hr^{-1}					
	toluene		*o*-xylene		*p*-xylene	
20	10.03847	2.58284	4.74276	0.8665	7.88225	1.983
30	17.08978	6.27083	7.31397	2.27264	12.75304	4.66493
40	24.37002	9.4645	10.54603	3.5375	18.61956	6.91919
50	28.73756	10.65754	11.60172	3.69922	21.47525	8.23786
60	34.09515	12.86672	14.00018	4.55125	24.73986	9.19547
	heptane		*iso*-cetane			
20	27.69568	5.20394	0.88812	0.40465		
30	40.81357	14.00933	1.3965	0.66492		
40	55.87734	19.42812	1.85738	1.00806		
50	69.42885	24.97179	2.30408	1.17764		
60	83.63427	30.17899	2.72396	1.38315		

Table B.12: Experimentally measured liquid hydrocarbon fluxes in SBAD-1 at 22°C.

Δp	Flux	Error	Flux	Error	Flux	Error
bar	$Lm^{-2}hr^{-1}$					
	toluene		n-octane		*iso*-octane	
20	8.76	0.74	2.23	0.5	0.087	0.025
30	14.89	1.99	3.55	0.79	0.302	0.12
40	17.26	5.28	4.76	0.94	0.419	0.176
50	24.06	2.52	6.01	1	0.537	0.233
60	27.53	3.09	7.5	1.71	0.627	0.3
	1-methylnaphthalene		*iso*-cetane		decalin	
20	0.054	0.011	0.0266	0.0024	0.063	0.014
30	0.072	0.008	0.0377	0.0021	0.104	0.017
40	0.1	0.01	0.0499	0.0016	0.13	0.016
50	0.108	0.007	0.0626	0.003	0.173	0.016
60	0.121	0.008	0.0762	0.0037	0.186	0.015
	1,3,5-triisopropylbenzene		*tert*-butylbenzene		methylcyclohexane	
20	0.0054	-	0.23	0.07	0.74	0.36
30	0.0132	-	0.32	0.09	1.16	0.46
40	0.0155	-	0.44	0.11	1.51	0.6
50	0.0201	-	0.53	0.12	2.41	0.92
60	0.0233	-	0.66	0.16	2.37	0.93

Table B.13: Calculated FH-LM Maxwell-Stefan diffusivities, $Đ_{im}^{V}$, of liquid hydrocarbons in PIM-1 and SBAD-1 at unit activity, 22°C. For Maxwell-Stefan diffusivities calculated based on other fugacity models, see Table A.7. For Fickian diffusivities, see Table A.8.

PIM-1		
	Diffusivity (cm2/s)	Error
toluene	2.51E-07	6.46E-08
heptane	1.19E-06	2.14E-07
p-xylene	1.55E-07	3.90E-08
o-xylene	6.85E-08	1.26E-08
iso-cetane	1.17E-08	5.36E-09
SBAD-1		
	Diffusivity (cm2/s)	Error
toluene	1.27E-07	1.07E-08
iso-octane	6.97E-08	2.76E-08
n-octane	6.44E-08	1.44E-08
methylcyclohexane	5.35E-08	2.60E-08
decalin	1.83E-08	4.08E-09
tert-butylbenzene	1.50E-08	4.56E-09
1-methylnaphthalene	3.52E-10	7.20E-11
1,3,5-triisopropylbenzene	7.03E-10	0.00E+00
iso-cetane	6.98E-09	6.32E-10

Table B.14: Experimentally measured ternary sorption versus predictions of toluene, *p*-xylene and heptane in PIM-1 at 22°C with a bulk fluid containing 0.321 wt% toluene, 0.314 wt% *p*-xylene and 0.365 wt% heptane. Note that $\phi_{\text{total}} = 1 - \phi_m$ and ϕ_{tolulene}, $\phi_{p\text{-xylene}}$. Additionally, ϕ_{heptane} are only on the basis of sorped species (i.e. $\phi_{\text{tolulene}} + \phi_{p\text{-xylene}} + \phi_{\text{heptane}} = 1$).

	Exp	FH	DMS	FH-LM
ϕ_{total}	0.916 ± 0.128	0.557 ± 0.013	0.754 + 0.006	0.824 ± 0.075
ϕ_{tolulene}	0.431 ± 0.038	0.317 ± 0.002	0.308 ± 0.007	0.315 ± 0.009
$\phi_{p\text{-xylene}}$	0.333 ± 0.044	0.304 ± 0.002	0.260 ± 0.002	0.359 ± 0.007
ϕ_{heptane}	0.236 ± 0.083	0.379 ± 0.003	0.432 ± 0.004	0.326 ± 0.013

Table B.15: Experimentally measured binary sorption of heptane and *o*-xylene in PIM-1 at 22°C.

	Experimental	FH	DMS	FH-LM
Bulk fluid: heptane = 0.2548 mol%, *o*-xylene = 0.7452 mol%				
ϕ_{polymer}	0.299 ± 0.011	0.281 ± 0.003	0.258 ± 0.002	0.303 ± 0.038
ϕ_{heptane}	0.132 ± 0.036	0.182 ± 0.002	0.164 ± 0.003	0.161 ± 0.016
$\phi_{p\text{-xylene}}$	0.570 ± 0.025	0.539 ± 0.002	0.578 ± 0.001	0.537 ± 0.022
Bulk fluid: heptane = 0.4722 mol%, *o*-xylene = 0.5278 mol%				
ϕ_{polymer}	0.391 ± 0.013	0.411 ± 0.013	0.304 ± 0.004	0.403 ± 0.035
ϕ_{heptane}	0.206 ± 0.026	0.291 ± 0.008	0.217 ± 0.005	0.268 ± 0.022
$\phi_{p\text{-xylene}}$	0.403 ± 0.013	0.299 ± 0.004	0.479 ± 0.001	0.329 ± 0.013
Bulk fluid: heptane = 0.7578 mol%, *o*-xylene = 0.2422 mol%				
ϕ_{polymer}	0.399 ± 0.066	0.508 ± 0.014	0.382 ± 0.006	0.474 ± 0.028
ϕ_{heptane}	0.360 ± 0.039	0.362 ± 0.012	0.291 ± 0.008	0.364 ± 0.024
$\phi_{p\text{-xylene}}$	0.241 ± 0.027	0.130 ± 0.002	0.327 ± 0.003	0.327 ± 0.005

Table B.16: Predicted partial fluxes ($Lm^{-2}h^{-1}$) for hydrocarbon molecules according to Scenario 1 where Fick's law transport is assumed.

	Exp	FH	DMS	FH-LM
Separation 1 via PIM-1 at a transmembrane pressure of 30 bar				
toluene	1.3791	0.7216	1.9818	0.6207
heptane	1.4961	1.1141	2.6668	0.7516
p-xylene	1.2720	0.6223	1.8671	0.4961
o-xylene	1.5808	0.4226	2.4642	0.4874
iso-cetane	0.5973	0.1166	0.2668	0.2611
Separation 2 via SBAD-1 at a transmembrane pressure of 40 bar				
toluene	0.1319	0.1145	0.2036	0.0968
methylcyclohexane	0.1996	0.2178	0.3333	0.2016
1-methylnaphthalene	0.0241	0.0006	0.0097	0.0013
decalin	0.1063	0.0532	0.0306	0.0357
n-octane	0.2466	0.2248	0.4451	0.2487
iso-octane	0.1254	0.1324	0.2381	0.1258
tert-butylbenzene	0.0259	0.0137	0.0181	0.0108
1,3,5-triisopropylbenzene	0.0120	0.0008	0.0003	0.0005
iso-cetane	0.0081	0.0036	0.0011	0.0022
Separation 3 via SBAD-1 at a transmembrane pressure of 30 bar				
toluene	0.0764	0.0187	0.0186	0.0155
iso-octane	0.1579	0.0411	0.0410	0.0401
iso-cetane	0.1678	0.0353	0.0196	0.0303

Table B.17: Predicted partial fluxes ($Lm^{-2}h^{-1}$) for hydrocarbon molecules according to Scenario 2 where Maxwell-Stefan transport without diffusion coupling is assumed.

	Exp	FH	DMS	FH-LM
Separation 1 via PIM-1 at a transmembrane pressure of 30 bar				
toluene	1.3791	1.4761	2.0036	1.9376
heptane	1.4961	1.7463	2.3983	2.8581
p-xylene	1.2720	1.3030	1.8258	1.8218
o-xylene	1.5808	1.4736	2.0341	1.8835
iso-cetane	0.5973	0.2202	0.0742	0.0783
Separation 2 via SBAD-1 at a transmembrane pressure of 40 bar				
toluene	0.1319	0.1179	0.0145	0.1412
methylcyclohexane	0.1996	0.2206	0.0292	0.2660
1-methylnaphthalene	0.0241	0.0028	0.0028	0.0023
decalin	0.1063	0.0543	0.0137	0.0336
n-octane	0.2466	0.2343	0.0306	0.3143
iso-octane	0.1254	0.1254	0.0209	0.1090
tert-butylbenzene	0.0259	0.0158	0.0028	0.0144
1,3,5-triisopropylbenzene	0.0120	0.0009	0.0015	0.0005
iso-cetane	0.0081	0.0033	0.0038	0.0014
Separation 3 via SBAD-1 at a transmembrane pressure of 30 bar				
toluene	0.0764	0.0187	0.0151	0.0145
iso-octane	0.1569	0.0406	0.0334	0.0354
iso-cetane	0.1678	0.0362	0.0148	0.0222

Table B.18: Predicted partial fluxes ($Lm^{-2}h^{-1}$) for hydrocarbon molecules according to Scenario 3 where Maxwell-Stefan transport with Vignes diffusion coupling is assumed.

	Exp	FH	DMS	FH-LM
Separation 1 via PIM-1 at a transmembrane pressure of 30 bar				
toluene	1.3791	1.4493	0.5657	1.5199
heptane	1.4961	1.7108	0.3111	2.0929
p-xylene	1.2720	1.2832	0.5287	1.5693
o-xylene	1.5808	1.4626	0.7193	1.7248
iso-cetane	0.5973	0.2950	0.0576	0.1972
Separation 2 via SBAD-1 at a transmembrane pressure of 40 bar				
toluene	0.1319	0.1138	0.1373	0.1204
methylcyclohexane	0.1996	0.2130	0.1983	0.2292
1-methylnaphthalene	0.0241	0.0028	0.0056	0.0025
decalin	0.1063	0.0525	0.0088	0.0291
n-octane	0.2466	0.2270	0.2907	0.2703
iso-octane	0.1254	0.1182	0.1392	0.0814
tert-butylbenzene	0.0259	0.0153	0.0114	0.0125
1,3,5-triisopropylbenzene	0.0120	0.0009	0.0003	0.0005
iso-cetane	0.0081	0.0032	0.0006	0.0012
Separation 3 via SBAD-1 at a transmembrane pressure of 30 bar				
toluene	0.0764	0.0185	0.0145	0.0138
iso-octane	0.1579	0.0401	0.0319	0.0334
iso-cetane	0.1678	0.0360	0.0141	0.0215

Table B.19: Predicted partial fluxes ($Lm^{-2}h^{-1}$) for hydrocarbon molecules according to Scenario 4 where Maxwell-Stefan transport with Vignes diffusion coupling and a diffusivity dependence on polymer swelling is assumed.

	Exp	FH	DMS	FH-LM
Separation 1 via PIM-1 at a transmembrane pressure of 30 bar				
toluene	1.3791	1.3490	0.5637	1.5075
heptane	1.4961	1.5940	0.3100	2.0793
p-xylene	1.2720	1.1943	0.5268	1.5563
o-xylene	1.5808	1.3568	0.7165	1.7068
iso-cetane	0.5973	0.2765	0.0576	0.1963
Separation 2 via SBAD-1 at a transmembrane pressure of 40 bar				
toluene	0.1319	0.2387	0.2116	0.1821
methylcyclohexane	0.1996	0.4166	0.2770	0.3328
1-methylnaphthalene	0.0241	0.0020	0.0057	0.0024
decalin	0.1063	0.1669	0.0108	0.0615
n-octane	0.2466	0.4367	0.4171	0.3747
iso-octane	0.1254	0.3137	0.2783	0.1576
tert-butylbenzene	0.0259	0.0331	0.0167	0.0192
1,3,5-triisopropylbenzene	0.0120	0.0036	0.0008	0.0012
iso-cetane	0.0081	0.0229	0.0013	0.0036
Separation 3 via SBAD-1 at a transmembrane pressure of 30 bar				
toluene	0.0764	0.2201	0.0691	0.0590
iso-octane	0.1579	0.4702	0.1497	0.1357
iso-cetane	0.1678	0.5109	0.0743	0.1061

Table B.20: Predicted partial fluxes ($Lm^{-2}h^{-1}$) for hydrocarbon molecules according to Scenario 5 where Maxwell-Stefan transport an average diffusivity of all molecules is assumed.

	Exp	FH	DMS	FH-LM
Separation 1 via PIM-1 at a transmembrane pressure of 30 bar				
toluene	1.3791	2.5377	2.8841	2.9000
heptane	1.4961	2.7358	1.9286	3.3457
p-xylene	1.2720	2.3094	2.7630	2.9371
o-xylene	1.5808	2.9313	3.7122	3.6870
iso-cetane	0.5973	0.9903	1.1648	1.1562
Separation 2 via SBAD-1 at a transmembrane pressure of 40 bar				
toluene	0.1319	0.1286	0.1870	0.1296
methylcyclohexane	0.1996	0.2399	0.2510	0.2420
1-methylnaphthalene	0.0241	0.0200	0.0324	0.0223
decalin	0.1063	0.0868	0.0235	0.0583
n-octane	0.2466	0.2557	0.3264	0.2604
iso-octane	0.1254	0.1139	0.2473	0.0737
tert-butylbenzene	0.0259	0.0225	0.0284	0.0215
1,3,5-triisopropylbenzene	0.0120	0.0196	0.0169	0.0173
iso-cetane	0.0081	0.0141	0.0063	0.0077
Separation 3 via SBAD-1 at a transmembrane pressure of 30 bar				
toluene	0.0764	0.1051	0.0980	0.0665
iso-octane	0.1579	0.1944	0.2059	0.1228
iso-cetane	0.1678	0.2802	0.1269	0.1766

APPENDIX C

GLOBAL MODULE MODELING AND SIMULATION

This section will discuss existing methods for modeling and simulation of the global transport through an industrial membrane module. This problem combines feed channel model, permeate channel model, and local membrane transport model discussed in the previous section to generate a partial differential algebraic equation (PDAE) system (see Figure C.1 for visual of the system and dimensions). Many papers on modeling and simulation of the global transport through the membrane module discretize material, momentum, and energy balances in the feed and permeate channels to converge a set of algebraic equations [147] [148] [149] [150].

Of many ways to solve different membrane module geometries, the literature is sparse in using the Maxwell-Stefan framework. Most often, a simple local flux model like equation (Equation 1.3) is employed. By employing this, the coupling between each species is inherently not captured.

The first existing method that applies Maxwell-Stefan is by Virales et al. [151]. They use finite differences for the feed and permeate channels. Then they use orthogonal collocation throughout the membrane layer to converge the algebraic equations using gPROMS.

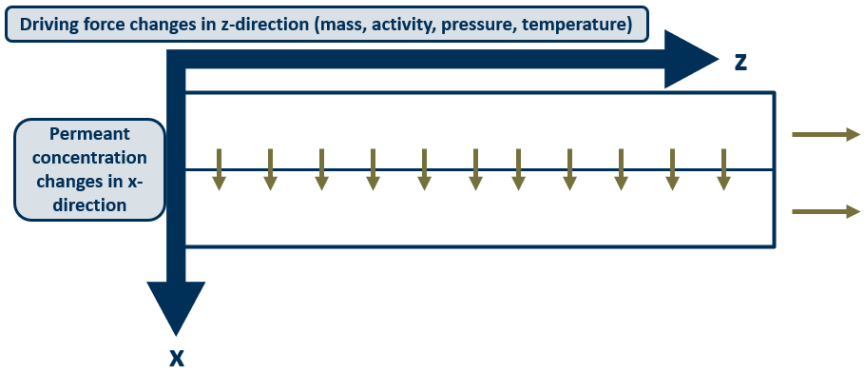


Figure C.1: Representation of dimensions and driving for changes within flat-plate membrane module.

Although, the full solution procedure is not specified nor is the code available for general-purpose use. The publication also lists assumptions including negligible pressure-drop ideal gas behavior in bulk channels. Both of which are not genrealizable for systems with complex mixtures and appreciable momentum loses. Another publication coded the PDAEs in gPROMs as well, but gave no indication of how to simulate the presented model equations [38]. Lastly, Graaf et al. assumed well mixed volumes through the feed and permeate channels [152]. This reduces the global membrane transport problem to solving the local transport fluxes and multiplying it by the total module membrane area to get channel flow rate changes.

Even when assuming a simple local transport model, the global transport is still a challenging two-point boundary value problem. The governing and constitutive equations for material, momentum, and energy balances (including boundary conditions) are presented below.

For the feed channel, the PDAEs system in Figure C.1 can be expressed as $\forall x \in (0, L), \forall z \in (0, P)$ [153]

Axial material balance on component i, momentum balance, and energy balance

$$\frac{\partial F_i^f}{\partial z} = -N_i(x, z), \quad \frac{\partial \left(\rho^m (v^f)^2\right)}{\partial z} = -\frac{\partial P^f}{\partial z} - f_v^f, \quad \frac{\partial e^f}{\partial z} = q^f, \tag{C.1}$$

Definition of molar flux of component i, heat flux, frictional pressure loss

$$F_i^f = c_i^f v^f - D_i \frac{\partial c_i^f}{\partial z}, e^f = \rho v^f H - k^c \frac{\partial T^f}{\partial z}, f_v^z = \frac{\kappa}{K_F} v^z, \tag{C.2}$$

Boundary conditions

$$(c_i^f v^f)|_{z=0} = A^f m_i^f, \quad P^f|_{z=0} = P_o, \quad T^f|_{z=0} = T_o, \tag{C.3}$$

where ρ is density, k^c is thermal conductivity, A^f is cross-sectional area of the feed channel,

v^f is axial velocity in feed channel, κ is viscosity, K_F is pressure loss parameter, and other variable have been defined but superscript f means for feed channel.

For the permeate channel, the equations are exact except superscript f goes to p, and flux sign should be flipped in the material balances. As stated above, the equations are usually discritized to take advantage of a "succession of states" methodology [147]. This numerical approach works great for simple flux relationships, however in highly nonlinear systems when using complex mixtures are subject to the same pitfalls for the 2-point boundary value problem of complex mixture local transport outlined in section 2.2. These pitfalls arise when dealing with high component numbers, strong cross-diffusional coupling, and non-ideal thermodynamics.

Given the state of the literature on global module modeling using Maxwell-Stefan as the local transport model, the main challenge is that a generalizable global membrane transport numerical method using complex mixtures has not been demonstrated in the literature. This presents a critical need to fulfill in order to have a complex mixture industrial membrane module unit operation within commercial process simulation environments. Again, this dissertation does not contribute any solutions, but will list possible future directions in section 6.3.

REFERENCES

[1] Antonino Campione, Andrea Cipollina, I. David L. Bogle, Luigi Gurreri, Alessandro Tamburini, Michele Tedesco, and Giorgio Micale. A hierarchical model for novel schemes of electrodialysis desalination. *Desalination*, 465:79–93, 2019.

[2] David S. Sholl and Ryan P. Lively. Seven chemical separations to change the world. *Nature*, 532(7600):435–437, Apr 2016.

[3] National Academies of Sciences Engineering and Medicine. *A Research Agenda for Transforming Separation Science*. The National Academies Press, Washington, DC, 2019.

[4] Apostolis A. Koutinas, Anestis Vlysidis, Daniel Pleissner, Nikolaos Kopsahelis, Isabel Lopez Garcia, Ioannis K. Kookos, Seraphim Papanikolaou, Tsz Him Kwan, and Carol Sze Ki Lin. Valorization of industrial waste and by-product streams via fermentation for the production of chemicals and biopolymers. *Chemical Society Reviews*, 43(8):2587, 2014.

[5] Roger A. Sheldon. Green solvents for sustainable organic synthesis: state of the art. *Green Chem.*, 7:267–278, 2005.

[6] Coby J. Clarke, Wei-Chien Tu, Oliver Levers, Andreas Bröhl, and Jason P. Hallett. Green and sustainable solvents in chemical processes. *Chemical Reviews*, 118(2):747–800, 2018. PMID: 29300087.

[7] Alan G. Marshall and Ryan P. Rodgers. Petroleomics: The next grand challenge for chemical analysis. *Accounts of Chemical Research*, 37(1):53–59, 2004. PMID: 14730994.

[8] David S. Sholl and Ryan P. Lively. Exemplar mixtures for studying complex mixture effects in practical chemical separations. *JACS Au*, 2(2):322–327, Feb 2022.

[9] J P Chastain, J J Camberato, J E Albrecht, and J Adams. Swine manure production and nutrient content. *S. C. Confin. Anim. Manure Manag. Certif. Program Clemson Univ. SC*, pages 1–17, 1999.

[10] Geert Boeije, Ronald Corstanje, André Rottiers, and Diederik Schowanek. Adaptation of the cas test system and synthetic sewage for biological nutrient removal: Part i: Development of a new synthetic sewage. *Chemosphere*, 38(4):699–709, 1999.

[11] Kepa Ruiz-Mirazo and Alvaro Moreno. Basic autonomy as a fundamental step in the synthesis of life. *Artificial Life*, 10(3):235–259, 07 2004.

[12] R. Baker. *Membrane Transport Theory*, chapter 2, pages 15–96. John Wiley & Sons, Ltd, 2012.

[13] S. Robinson, R. Jubin, and B. Choate. Materials for separation technologies. energy and emission reduction opportunities. Technical report, Research Org.: Oak Ridge National Lab. (ORNL), Oak Ridge, TN (United States), United States, 2005.

[14] Ronita Mathias, Dylan J. Weber, Kirstie A. Thompson, Bennett D. Marshall, M.G. Finn, Joseph K. Scott, and Ryan P. Lively. Framework for predicting the fractionation of complex liquid feeds via polymer membranes. *Journal of Membrane Science*, 640:119767, 2021.

[15] R. Baker. *Membranes and Modules*, chapter 3, pages 97–178. John Wiley & Sons, Ltd, 2012.

[16] Patricia Luis. Chapter 1 - introduction. In Patricia Luis, editor, *Fundamental Modelling of Membrane Systems*, pages 1–23. Elsevier, 2018.

[17] C. Roos, H. Y. Jang an D. Weber, and R. Lively. Matching analysis of mixed matrix membranes for organic solvent reverse osmosis. *Industrial & Engineering Chemistry Research*, Submitted, August 2021.

[18] Endre Nagy. 1 - on mass transport through a membrane layer. In Endre Nagy, editor, *Basic Equations of the Mass Transport through a Membrane Layer*, pages 1–34. Elsevier, Oxford, 2012.

[19] J.G. Wijmans and R.W. Baker. The solution-diffusion model: a review. *Journal of Membrane Science*, 107(1):1–21, 1995.

[20] D. R. Paul. The role of membrane pressure in reverse osmosis. *Journal of Applied Polymer Science*, 16(3):771–782, 1972.

[21] Matteo Minelli, Karel Friess, Ondřej Vopička, and Maria Grazia De Angelis. Modeling gas and vapor sorption in a polymer of intrinsic microporosity (pim-1). *Fluid Phase Equilibria*, 347:35–44, 2013.

[22] A. L. Myers and J. M. Prausnitz. Thermodynamics of mixed-gas adsorption. *AIChE Journal*, 11(1):121–127, 1965.

[23] A. Kapoor, J. A. Ritter, and Ralph T. Yang. An extended Langmuir model for adsorption of gas mixtures on heterogeneous surfaces. *Langmuir*, 6(3):660–664, Mar 1990.

[24] Krista S. Walton and David S. Sholl. Predicting multicomponent adsorption: 50 years of the ideal adsorbed solution theory. *AIChE Journal*, 61(9):2757–2762, 2015.

[25] A. Erto, A. Lancia, and D. Musmarra. A real adsorbed solution theory model for competitive multicomponent liquid adsorption onto granular activated carbon. *Microporous and Mesoporous Materials*, 154:45–50, 2012. Special Issue: Characterisation of Porous Solids IX.

[26] Paul J. Flory. Thermodynamics of high polymer solutions. *The Journal of Chemical Physics*, 9(8):660–660, 1941.

[27] P T P Aryanti, D Ariono, A N Hakim, and I G Wenten. Flory-Huggins based model to determine thermodynamic property of polymeric membrane solution. *Journal of Physics: Conference Series*, 1090:012074, September 2018.

[28] W.R. Vieth, J.M. Howell, and J.H. Hsieh. Dual sorption theory. *Journal of Membrane Science*, 1:177–220, January 1976.

[29] Nicolas Kruse, Yuliya Schießer, Norman Reger-Wagner, Hannes Richter, Ingolf Voigt, Gerd Braun, and Jens-Uwe Repke. High pressure adsorption, permeation and swelling of carbon membranes – measurements and modelling at up to 20 MPa. *Journal of Membrane Science*, 544:12–17, December 2017.

[30] Timothy V. Bartholomew and Meagan S. Mauter. Computational framework for modeling membrane processes without process and solution property simplifications. *Journal of Membrane Science*, 573:682–693, 2019.

[31] R. Krishna and J.A Wesselingh. Review article number 50-the Maxwell-Stefan approach to mass transfer. *Chemical Engineering Science*, 52(6):861–911, 3 1997. J eview AR 94.

[32] Ruslan M. Marutovsky and Martin Bulow. Sorption kinetics of multi-component gaseous and liquid mixtures on porous sorbents. *Gas Separation & Purification*, 1(2):66–76, December 1987.

[33] Oleg Medvedev. *Diffusion Coefficients in Multicomponent Mixtures*. PhD thesis, Technical University of Denmark, 2005.

[34] J.P. Brun, C. Larchet, R. Melet, and G. Bulvestre. Modelling of the pervaporation of

binary mixtures through moderately swelling, non-reacting membranes. *Journal of Membrane Science*, 23(3):257–283, May 1985.

[35] C. Y. Pan. Gas separation by permeators with high-flux asymmetric membranes. *AIChE Journal*, 29(4):545–552, 1983.

[36] Lisa Hesse, Jovana Mićović, Patrick Schmidt, Andrzej Górak, and Gabriele Sadowski. Modelling of organic-solvent flux through a polyimide membrane. *Journal of Membrane Science*, 428:554 – 561, 2013.

[37] P. Izák, L. Bartovska, K. Friess, M. Sıek, and P. Uchytil. Description of binary liquid mixtures transport through non-porous membrane by modified Maxwell–Stefan equations. *Journal of Membrane Science*, 214(2):293 – 309, 2003.

[38] Nitish Mittal, Peng Bai, Adam Kelloway, J. Ilja Siepmann, Prodromos Daoutidis, and Michael Tsapatsis. A mathematical model for zeolite membrane module performance and its use for techno-economic evaluation of improved energy efficiency hybrid membrane-distillation processes for butane isomer separations. *Journal of Membrane Science*, 520:434 – 449, 2016.

[39] Rajamani Krishna. Describing mixture permeation across polymeric membranes by a combination of Maxwell-Stefan and Flory-Huggins models. *Polymer*, 103:124–131, 2016.

[40] Dylan J. Weber, Ronita Mathias, Ryan P. Lively, and Joseph K. Scott. Improved numerical methods for simulating complex mixture transport across asymmetric polymer membranes using a Maxwell–Stefan model. *Journal of Membrane Science*, 687:121995, 2023.

[41] Robert F. DeJaco, Kenneth Loprete, Kenneth Pennisi, Sudip Majumdar, J. Ilja Siepmann, Prodromos Daoutidis, Hannah Murnen, and Michael Tsapatsis. Modeling

and simulation of gas separations with spiral-wound membranes. *AIChE Journal*, 66(8):e16274, 2020.

[42] Roda Bounaceur, Etienne Berger, Marc Pfister, Alvaro Andres Ramirez Santos, and Eric Favre. Rigorous variable permeability modelling and process simulation for the design of polymeric membrane gas separation units: MEMSIC simulation tool. *Journal of Membrane Science*, 523:77–91, 2017.

[43] Ravichand Kancherla, Shaik Nazia, Swayampakula Kalyani, and Sundergopal Sridhar. Modeling and simulation for design and analysis of membrane-based separation processes. *Computers & Chemical Engineering*, 148:107258, 2021.

[44] Dimitar Peshev and Andrew G. Livingston. Osn designer, a tool for predicting organic solvent nanofiltration technology performance using aspen one, matlab and cape open. *Chemical Engineering Science*, 104:975–987, 2013.

[45] Rubaba Mohammadi, Walter Tang, and Mika Sillanpää. A systematic review and statistical analysis of nutrient recovery from municipal wastewater by electrodialysis. *Desalination*, 498:114626, 2021.

[46] R. Krishna. Problems and pitfalls in the use of the Fick formulation for intraparticle diffusion. *Chemical Engineering Science*, 48(5):845–861, 1993.

[47] C.-C. CHEN, M. Gorensek, J. Scott, and I. Palou-Rivera. Process modeling of intensified chemical processes. *Chemical Engineering Progress Magazine*, 116(3), March 2020.

[48] A. Heintz and W. Stephan. A generalized solution—diffusion model of the pervaporation process through composite membranes part ii. concentration polarization, coupled diffusion and the influence of the porous support layer. *Journal of Membrane Science*, 89(1):153 – 169, 1994.

[49] Lisa Hesse, Shahbaz Naeem, and Gabriele Sadowski. Voc sorption in glassy polyimides—measurements and modeling. *Journal of Membrane Science*, 415-416:596–607, 2012.

[50] Cludio P. Ribeiro, Benny D. Freeman, and Donald R. Paul. Modeling of multicomponent mass transfer across polymer films using a thermodynamically consistent formulation of the Maxwell–Stefan equations in terms of volume fractions. *Polymer*, 52(18):3970–3983, 2011.

[51] Ross Taylor. Coupled heat and mass transfer in multicomponent systems: Solution of the Maxwell-Stefan equations. *Letters in Heat and Mass Transfer*, 8(5):405–416, 1981.

[52] R. R. Sijabat, M. T. de Groot, S. Moshtarikhah, and J. van der Schaaf. Maxwell–Stefan model of multicomponent ion transport inside a monolayer nafion membrane for intensified chlor-alkali electrolysis. *Journal of Applied Electrochemistry*, 49(4):353–368, Apr 2019.

[53] Jan-Baptist Loos, Peter Verheijen, and J.A. Moulijn. Numerical simulation of the generalized Maxwell-Stefan model for multicomponent diffusion in microporous sorbents. *Collection of Czechoslovak Chemical Communications*, 57:687–697, 01 1992.

[54] Francesco Fornasiero, John M. Prausnitz, and Clayton J. Radke. Multicomponent diffusion in highly asymmetric systems. an extended Maxwell-Stefan model for starkly different-sized, segment-accessible chain molecules. *Macromolecules*, 38(4):1364–1370, 2005.

[55] D.R Paul. Reformulation of the solution-diffusion theory of reverse osmosis. *Journal of Membrane Science*, 241(2):371–386, 2004.

[56] Nicolas Kruse, Yuliya Schießer, Norman Reger-Wagner, Hannes Richter, Ingolf Voigt, Gerd Braun, and Jens-Uwe Repke. High pressure adsorption, permeation and swelling of carbon membranes – measurements and modelling at up to 20mpa. *Journal of Membrane Science*, 544:12–17, 2017.

[57] W.C. CONNER. The contribution of surface diffusion to transport in nanoporous solids. In Wm. Curtis Conner and Jacques Fraissard, editors, *Fluid Transport in Nanoporous Materials*, pages 195–210, Dordrecht, 2006. Springer Netherlands.

[58] Rajamani Krishna. A Maxwell-Stefan-Glueckauf description of transient mixture uptake in microporous adsorbents. *Separation and Purification Technology*, 191:392–399, 2018.

[59] Y. Bard. *Nonlinear Parameter Estimation*. Academic Press, New York, April 1974.

[60] Lorenz T. Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers & Chemical Engineering*, 8:243–247, 1984.

[61] Hans Georg Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17:1603–1608, 1984.

[62] Claas Michalik, Ralf Hannemann, and Wolfgang Marquardt. Incremental single shooting—a robust method for the estimation of parameters in dynamical systems. *Computers & Chemical Engineering*, 33(7):1298–1305, 2009.

[63] Maqsud R. Chowdhury and Jeffrey R. McCutcheon. Elucidating the impact of temperature gradients across membranes during forward osmosis: Coupling heat and mass transfer models for better prediction of real osmotic systems. *Journal of Membrane Science*, 553:189–199, 2018.

[64] Rajamani Krishna. Highlighting thermodynamic coupling effects in alcohol/water pervaporation across polymeric membranes. *ACS Omega*, 4(12):15255–15264, 2019. PMID: 31552372.

[65] L So Darken. Diffusion, mobility and their interrelation through free energy in binary metallic systems. *Trans. Aime*, 175:184–201, 1948.

[66] Uri M. Ascher and Linda R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. Society for Industrial and Applied Mathematics, 3600 University City Science Center Philadelphia, PA, United States, 1998.

[67] Patrizia Marchetti and Andrew G. Livingston. Predictive membrane transport models for organic solvent nanofiltration: How complex do we need to be? *Journal of Membrane Science*, 476:530–553, 2015.

[68] Bennett D. Marshall, Joshua W. Allen, and Ryan P. Lively. A model for the separation of complex liquid mixtures with glassy polymer membranes: A thermodynamic perspective. *Journal of Membrane Science*, 647:120316, 2022.

[69] Young Joo Lee, Lihua Chen, Janhavi Nistane, Hye Youn Jang, Dylan J. Weber, Joseph K. Scott, Neel D. Rangnekar, Bennett D. Marshall, Wenjun Li, J. R. Johnson, Nicholas C. Bruno, M. G. Finn, Rampi Ramprasad, and Ryan P. Lively. Data-driven predictions of complex organic mixture permeation in polymer membranes. *Nature Communications*, 14(1):4931, Aug 2023.

[70] J Gaynor. Monthly energy review, march 2021. Technical report, Department of Energy, Washington, DC (USA). Energy Information Administration, 2021.

[71] Laura E Black and Heather A Boucher. Process for separating alkylaromatics from aromatic solvents and the separation of the alkylaromatic isomers using membranes, February 18 1986. US Patent 4,571,444.

[72] Lloyd S. White and Craig R. Wildemuth. Aromatics enrichment in refinery streams using hyperfiltration. *Industrial & Engineering Chemistry Research*, 45(26):9136–9143, 2006.

[73] S Rosenbaum and O Cotton. Steady-state distribution of water in cellulose acetate membrane. *Journal of Polymer Science Part A-1: Polymer Chemistry*, 7(1):101–109, 1969.

[74] DR Paul and JD Paciotti. Driving force for hydraulic and pervaporative transport in homogeneous membranes. *Journal of Polymer Science: Polymer Physics Edition*, 13(6):1201–1214, 1975.

[75] Michele Galizia and Kelly P Bye. Advances in organic solvent nanofiltration rely on physical chemistry and polymer chemistry. *Frontiers in chemistry*, 6:511, 2018.

[76] Kelly P Bye and Michele Galizia. Fundamental origin of flux non-linearity in organic solvent nanofiltration: formulation of a thermodynamic/diffusion framework. *Journal of Membrane Science*, 603:118020, 2020.

[77] William Mickols, Zhaohuan Mai, and Bart van der Bruggen. Effect of pressure and temperature on solvent transport across nanofiltration and reverse osmosis membranes: An activity-derived transport model. *Desalination*, 501:114905, 2021.

[78] Pedro Silva, Shejiao Han, and Andrew G. Livingston. Solvent transport in organic solvent nanofiltration membranes. *Journal of Membrane Science*, 262(1):49–59, 2005.

[79] D. R. Paul and O. M. Ebra-Lima. Pressure-induced diffusion of organic liquids through highly swollen polymer membranes. *Journal of Applied Polymer Science*, 14(9):2201–2224, 1970.

[80] Lloyd S White. Transport properties of a polyimide solvent resistant nanofiltration membrane. *Journal of Membrane Science*, 205(1):191–202, 2002.

[81] M.H. Abdellah, C.A. Scholes, B.D. Freeman, L. Liu, and S.E. Kentish. Transport of terpenes through composite pdms/pan solvent resistant nanofiltration membranes. *Separation and Purification Technology*, 207:470–476, 2018.

[82] Edwin Habeych, Atze Jan van der Goot, and Remko Boom. Prediction of permeation fluxes of small volatile components through starch-based films. *Carbohydrate Polymers*, 68(3):528–536, 2007.

[83] Ahmadreza Raisi, Abdolreza Aroujalian, and Tahereh Kaghazchi. A predictive mass transfer model for aroma compounds recovery by pervaporation. *Journal of Food Engineering*, 95(2):305 – 312, 2009.

[84] Stefanie Postel, Saskia Wessel, Timm Keil, Patrick Eiselt, and Matthias Wessling. Multicomponent mass transport in organic solvent nanofiltration with solvent mixtures. *Journal of Membrane Science*, 466:361–369, 2014.

[85] Cláudio P. Ribeiro, Benny D. Freeman, and Donald R. Paul. Modeling of multicomponent mass transfer across polymer films using a thermodynamically consistent formulation of the Maxwell–Stefan equations in terms of volume fractions. *Polymer*, 52(18):3970–3983, August 2011.

[86] Peter M. Budd, Bader S. Ghanem, Saad Makhseed, Neil B. McKeown, Kadhum J. Msayib, and Carin E. Tattershall. Polymers of intrinsic microporosity (pims): robust, solution-processable, organic nanoporous materials. *Chem. Commun.*, pages 230–231, 2004.

[87] Kirstie A. Thompson, Ronita Mathias, Daeok Kim, Jihoon Kim, Neel Rangnekar, J. R. Johnson, Scott J. Hoy, Irene Bechis, Andrew Tarzia, Kim E. Jelfs, Benjamin A.

McCool, Andrew G. Livingston, Ryan P. Lively, and M. G. Finn. N-aryl-linked spirocyclic polymers for membrane separations of complex hydrocarbon mixtures. *Science*, 369(6501):310–315, 2020.

[88] Paul J. Flory. *Principles of polymer chemistry*. George Fisher Baker non-resident lectureship in chemistry at Cornell University. Cornell University Press Ithaca, N.Y., Ithaca, N.Y., 1953.

[89] Tzu-Huai Yang and Shingjiang Jessie Lue. Modeling sorption behavior for ethanol/water mixtures in a cross-linked polydimethylsiloxane membrane using the Flory-Huggins equation. *Journal of Macromolecular Science, Part B*, 52(7):1009–1029, 2013.

[90] Charles M Hansen. The three dimensional solubility parameter. *Danish Technical: Copenhagen*, 14, 1967.

[91] Charles M Hansen. *Hansen solubility parameters: a user's handbook*. CRC press, 2007.

[92] Ferruccio Doghieri and Giulio C. Sarti. Nonequilibrium lattice fluids: A predictive model for the solubility in glassy polymers. *Macromolecules*, 29(24):7885–7896, 1996.

[93] D. R. Paul and W. J. Koros. Effect of partially immobilizing sorption on permeability and the diffusion time lag. *Journal of Polymer Science: Polymer Physics Edition*, 14(4):675–685, 1976.

[94] RM Barrer, JA Barrie, and J Slater. Sorption and diffusion in ethyl cellulose. part iii. comparison between ethyl cellulose and rubber. *Journal of Polymer Science*, 27(115):177–197, 1958.

[95] L. Ansaloni and L. Deng. 7 - advances in polymer-inorganic hybrids as membrane materials. In P.M. Visakh, Gordana Markovic, and Daniel Pasquini, editors, *Recent Developments in Polymer Macro, Micro and Nano Blends*, pages 163–206. Woodhead Publishing, 2017.

[96] Shilpa Damle and W. J. Koros. "sorp-vection": An unusual membrane-based separation. *AIChE Journal*, 51(5):1396–1405, 2005.

[97] Neil B. McKeown and Peter M. Budd. Polymers of intrinsic microporosity (pims): organic materials for membrane separations, heterogeneous catalysis and hydrogen storage. *Chem. Soc. Rev.*, 35:675–683, 2006.

[98] Octavio Suárez-Iglesias, Ignacio Medina, María de los Ángeles Sanz, Consuelo Pizarro, and Julio L. Bueno. Self-diffusion in molecular fluids and noble gases: Available data. *Journal of Chemical & Engineering Data*, 60(10):2757–2817, 2015.

[99] transport modeling. transport-modeling/asymemsim: Initial release, June 2023.

[100] James N. Galloway, John D. Aber, Jan Willem Erisman, Sybil P. Seitzinger, Robert W. Howarth, Ellis B. Cowling, and B. Jack Cosby. The Nitrogen Cascade. *BioScience*, 53(4):341–356, 04 2003.

[101] Marc Ribaudo, Jorge Delgado, LeRoy Hansen, Michael Livingston, Roberto Mosheim, and James Williamson. Nitrogen in agricultural systems: Implications for conservation policy. Technical report, U.S. Dept. of Agriculture, Econ. Res. Serv., United States, 2011.

[102] Man Lang, Ping Li, and Xiaoyuan Yan. Runoff concentration and load of nitrogen and phosphorus from a residential area in an intensive agricultural watershed. *Science of The Total Environment*, 458-460:238–245, 2013.

[103] Luisa Barrera and Rohini Bala Chandran. Harnessing photoelectrochemistry for wastewater nitrate treatment coupled with resource recovery. *ACS Sustainable Chemistry & Engineering*, 9(10):3688–3701, 2021.

[104] R. Fernández-González, M.A. Martín-Lara, G. Blázquez, G. Tenorio, and M. Calero. Hydrolyzed olive cake as novel adsorbent for copper removal from fertilizer industry wastewater. *Journal of Cleaner Production*, 268:121935, 2020.

[105] Simone Visigalli, Andrea Turolla, Giacomo Bellandi, Micol Bellucci, Elisa Clagnan, Lorenzo Brusetti, Mingsheng Jia, Roberto Di Cosmo, Glauco Menin, Martina Bargna, Giovanni Bergna, and Roberto Canziani. Autotrophic nitrogen removal for decentralized treatment of ammonia-rich industrial textile wastewater: process assessment, stabilization and modelling. *Environmental Science and Pollution Research*, 28(34):46643–46654, Sep 2021.

[106] Scott A Bradford, Eran Segal, Wei Zheng, Qiquan Wang, and Stephen R Hutchins. Reuse of concentrated animal feeding operation wastewater on agricultural lands. *J Environ Qual*, 37(5 Suppl):S97–S115, September 2008.

[107] Yang Deng and James D. Englehardt. Electrochemical oxidation for landfill leachate treatment. *Waste Management*, 27(3):380–388, 2007.

[108] C. Brienza, I. Sigurnjak, T. Meier, E. Michels, F. Adani, O. Schoumans, C. Vaneeckhaute, and E. Meers. Techno-economic assessment at full scale of a biogas refinery plant receiving nitrogen rich feedstock and producing renewable energy and biobased fertilisers. *Journal of Cleaner Production*, 308:127408, 2021.

[109] Francisco Corona, Dolores Hidalgo, Jesús María Martín-Marroquín, Juan Castro, Sergio Sanz-Bedate, and Gregorio Antolín. Study of the crystallisation reaction behaviour to obtain struvite. *Waste and Biomass Valorization*, 13(9):3767–3786, Sep 2022.

[110] Francisco Corona, Dolores Hidalgo, Jesús María Martín-Marroquín, Juan Castro, Sergio Sanz-Bedate, and Gregorio Antolín. Study of the Crystallisation Reaction Behaviour to Obtain Struvite. *Waste and Biomass Valorization*, 13(9):3767–3786, September 2022.

[111] J. Bousek, D. Scroccaro, Jan Sima, Norbert Weissenbacher, and W. Fuchs. Influence of the gas composition on the efficiency of ammonia stripping of biogas digestate. *Bioresource Technology*, 203:259–266, March 2016.

[112] Alice Limoli, Michela Langone, and Gianni Andreottola. Ammonia removal from raw manure digestate by means of a turbulent mixing stripping process. *Journal of Environmental Management*, 176:1–10, July 2016.

[113] A. Campione, L. Gurreri, M. Ciofalo, G. Micale, A. Tamburini, and A. Cipollina. Electrodialysis for water desalination: A critical assessment of recent developments on process fundamentals, models and applications. *Desalination*, 434:121–160, 2018. Reviews on Research and Development in Desalination.

[114] Lin Shi, Yuansheng Hu, Sihuang Xie, Guangxue Wu, Zhenhu Hu, and Xinmin Zhan. Recovery of nutrients and volatile fatty acids from pig manure hydrolysate using two-stage bipolar membrane electrodialysis. *Chemical Engineering Journal*, 334:134–142, 2018.

[115] M. Mondor, L. Masse, D. Ippersiel, F. Lamarche, and D.I. Massé. Use of electrodialysis and reverse osmosis for the recovery and concentration of ammonia from swine manure. *Bioresource Technology*, 99(15):7363–7368, 2008.

[116] Emma Thompson Brewster, Chirag M. Mehta, Jelena Radjenovic, and Damien J. Batstone. A mechanistic model for electrochemical nutrient recovery systems. *Water Research*, 94:176–186, 2016.

[117] Andrew J. Ward, Kimmo Arola, Emma Thompson Brewster, Chirag M. Mehta, and Damien J. Batstone. Nutrient recovery from wastewater through pilot scale electrodialysis. *Water Research*, 135:57–65, 2018.

[118] Bion's advanced waste treatment technology start-up yields positive early results. https://bionenviro.com/bions-advanced-waste-treatment-technology-start-up-yields-positive-early-results/.

[119] A. P. I. Imported. World's first combined biofertilizer-biogas facility using poultry waste opens. http://staging.p613436.webspaceconfig.de/worlds-first-combined-biofertilizer-biogas-facility-using-poultry-waste-opens/.

[120] Company. https://www.envirokure.com/company.

[121] Sistema.bio - the biodigester solution. https://sistema.bio/.

[122] P-recovery | CNP cycles. https://cnp-cycles.de/en/cycles/nutrients-cycle/p-recovery.

[123] Pima county, AZ, tres rios WRF biosolids system upgrades – part four: NuReSys nutrient recovery system - schwing bioset. https://www.schwingbioset.com/pima-county-az-tres-rios-wrf-biosolids-system-upgrades-part-four-nuresys-nutrient-recovery-system-2/.

[124] Municipalities, industries, more. https://www.ostara.com/nutrient-recovery/municipalities-industries-more/.

[125] Frameworks. Projects. https://n2applied.com/casestudies/.

[126] Trammo and ReMo energy sign MOU - development of a low-carbon NH3 & exclusive offtake of green NH3. https://www.trammo.com/post/trammo-and-remo-energy-sign-mou-development-of-a-low-carbon-nh3-exclusive-offtake-of-green-nh3.

[127] Turning cow manure into clean water and dry fertilizer. https://www.farmprogress.com/commentary/turning-cow-manure-into-clean-water-and-dry-fertilizer-.

[128] About digested organics | company overview. https://digestedorganics.com/about-us/#numbers.

[129] Organics oceania| about ammonia. https://organicsoceania.com.au/about-ammonia/.

[130] Byosis - case studies. https://www.byosis.com/case-studies.

[131] Ammonia stripping | hansa engineering. https://hansa-engineering.se/en/solutions/ammonia-stripping/.

[132] Air strippers | heil process equipment. https://heilprocessequipment.com/products/air-strippers.

[133] Ammonia strippers - indusco environmental services. https://www.induscoenviro.com/air-strippers/air-strippers/.

[134] Packed tower air stripper - buy or rent an ammonia stripper. https://www.machengineering.com/ammonia-stripper/.

[135] About. https://www.nijhuisindustries.com/uk/about.

[136] Ammonia recovery | RVT. https://www.rvtpe.com/global/en/products/turnkey-units/ammonia-recovery.

[137] NH3 stripping tower for ammonia removal. https://task.be/en/wastewater-treatment/nh3-stripping-towers/.

[138] About us. https://www.burnhamrng.com/about-us.

[139] Antonino Campione, Andrea Cipollina, Francesco Calise, Alessandro Tamburini, Mosè Galluzzo, and Giorgio Micale. Coupling electrodialysis desalination with

photovoltaic and wind energy systems for energy storage: Dynamic simulations and control strategy. *Energy Conversion and Management*, 216:112940, 2020.

[140] Lu Wang, Meikun Xia, Hong Wang, Kefeng Huang, Chenxi Qian, Christos T. Maravelias, and Geoffrey A. Ozin. Greening ammonia toward the solar ammonia refinery. *Joule*, 2(6):1055–1074, 2018.

[141] Ryan P. Lively and David S. Sholl. From water to organics in membrane separations. *Nature Materials*, 16(3):276–279, Mar 2017.

[142] Jefferson W. Tester and Michael Modell. *Thermodynamics and its applications*. Prentice Hall PTR, 3rd edition, 1996.

[143] Neil B. McKeown, Bader Gahnem, Kadhum J. Msayib, Peter M. Budd, Carin E. Tattershall, Khalid Mahmood, Siren Tan, David Book, Henrietta W. Langmi, and Allan Walton. Towards polymer-based hydrogen storage materials: Engineering ultramicroporous cavities within polymers of intrinsic microporosity. *Angewandte Chemie International Edition*, 45(11):1804–1807, 2006.

[144] Carl L Yaws. *The yaws handbook of physical properties for hydrocarbons and chemicals: physical properties for more than 54,000 organic and inorganic chemical compounds, Coverage for C1 to C100 Organics and Ac to Zr Inorganics*. Gulf Professional Publishing, 2015.

[145] Carl L Yaws. *The Yaws handbook of vapor pressure: Antoine coefficients*. Gulf Professional Publishing, 2015.

[146] Carl L Yaws. *Thermophysical properties of chemicals and hydrocarbons*. William Andrew, 2008.

[147] Mathews J. Thundyil and William J. Koros. Mathematical modeling of gas separa-

tion permeators — for radial crossflow, countercurrent, and cocurrent hollow fiber membrane modules. *Journal of Membrane Science*, 125(2):275–291, 1997.

[148] D. T. Coker, B. D. Freeman, and G. K. Fleming. Modeling multicomponent gas separation using hollow-fiber membrane contactors. *AIChE Journal*, 44(6):1289–1302, June 1998.

[149] A. Makaruk and M. Harasek. Numerical algorithm for modelling multicomponent multipermeator systems. *Journal of Membrane Science*, 344(1):258–265, 2009.

[150] Runhong Qi and Michael A. Henson. Approximate modeling of spiral-wound gas permeators. *Journal of Membrane Science*, 121(1):11–24, 1996.

[151] P. Vareltzis, E.S. Kikkinides, and M.C. Georgiadis. On the optimization of gas separation processes using zeolite membranes. *Chemical Engineering Research and Design*, 81(5):525–536, May 2003.

[152] Jolinde M. van de Graaf, Freek Kapteijn, and Jacob A. Moulijn. Modeling permeation of binary mixtures through zeolite membranes. *AIChE Journal*, 45(3):497–511, 1999.

[153] James Marriott and Eva Sørensen. A general approach to modelling membrane modules. *Chemical Engineering Science*, 58(22):4975–4990, 2003.