AN EVALUATION OF POLYMER MEMBRANES FOR THE GASEOUS SEPARATION OF NO_X AND SO_2 FROM FLUE GAS

by

Brian Nelson

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

A thesis submitted to the Faculty and Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Chemical and Petroleum-Refining Engineering).

Golden, Colorado

Date 12/9/97

Signed: Brian Nelson

Brian Nelson

Approved: Robt Knecht

Dr. Robert Knecht
Thesis Advisor

Golden, Colorado

Date 12/9/97

Robert A. Baldwin

Dr. Robert Baldwin
Professor and Head,
Department of Chemical Engineering
and Petroleum Refining

ABSTRACT

A gas separation membrane pilot-plant was assembled at the Colorado School of Mines for experimental research in NO_x and SO_2 flue gas treatment at the Rocky Flats Environmental Technology Site. Since the pilot-plant was not available to researchers, the gas separation membrane system was modeled, and computer programs were written to simulate the pilot-plant's performance. The purpose of this study was to identify new membrane materials for NO_2 and SO_2 separation, evaluate the pilot-plant's membrane separation over a range of system variables, and investigate design concepts for increasing the pilot-plant's performance.

Of all the membrane materials identified for the separation of NO_2 and SO_2, PDMS (currently installed in the pilot-plant) and PEBAX have the best properties. At simulated temperatures and approximated experimental conditions, PDMS is shown to separate more SO_2 than PEBAX, but PEBAX produces a much smaller stage cut.

Single-stage performance charts show the separation of NO_2 and SO_2 over a range of system variables. By drastically reducing the permeate pressure or utilizing a multiple stage system, a single-stage membrane systems performance can be enhanced. Reducing the permeate pressure is shown to greatly increase SO_2 and NO_2 separation and reduce stage cut. A three-stage system does not increase separation, but it does decrease the stage cut.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

My sincerest gratitude goes to Dr. Robert Knecht for his guidance, support, and advise during this project. Thanks also goes to the members of the thesis committee, Dr. J. Douglas Way and Dr. John Dorgan, for their open ears during the past two years. Additionally, I would like to thank my family and friends (especially those at 1006 13th St.) for putting up with me during the last two years.

This work was made possible by a research grant from the Colorado Advanced Materials Institute (CAMI).

Chapter 1

INTRODUCTION

The contributions of sulfur dioxide (SO_2) and oxides of nitrogen (NO_x) to acid rain and photochemical smog threaten human health and the environment. Controlling SO_2 and NO_x in combustion flue gas streams has been a topic of research for decades. Conventional methods of control may target SO_2, NO_x, or both. These methods include combustion modification, fuel desulfurization, wet scrubbing, adsorption, selective catalytic reduction, selective non-catalytic reduction. These operations, however, require substantial capital investment and energy usage [1, 2]. Reducing costs and expanding treatment options is an incentive to reduce the total flow of exhaust gas into a stream of concentrated SO_2 and NO_x. Gas separation membranes have been proposed for this task [1, 3-7].

A hybrid process pilot-plant was constructed by faculty, students, and personnel at the Colorado School of Mines (CSM) for the treatment of SO_2 and NO_x. The pilot-plant was designed specifically for the effluent gas from an experimental process, microwave solidification of mixed inorganic wastes, at the Rocky Flats Environmental Technology Site (RFETS). Brief start-up tests focused on the pilot-plant's membrane separation, but these tests produced little usable data (see Section 3.2)[8].

Since the microwave solidification operations at RFETS have ceased and the pilot-plant has been unavailable, a new research concept is required to evaluate the hybrid process as a viable treatment concept for SO_2 and NO_x at RFETS or elsewhere. This research focuses on the membrane separation process of the pilot-plant. The objectives of this research are:

1. To identify polymer membrane materials for use in SO_2 and NO_x separations from flue gas.
2. To develop computer models for the prediction of multi-component gas separations over single-stage polymer membrane separators.
3. To use the computer models in evaluating the pilot-plant's performance.
4. To use the computer models to investigate methods for increasing the pilot-plant's performance.

Chapter 2

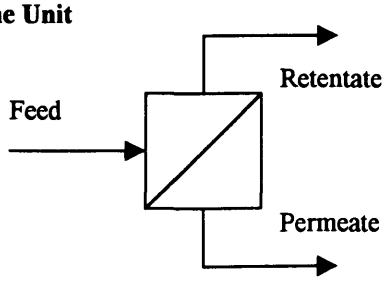
REVIEW AND DISCUSSION OF PERTINENT LITERATURE

The following literature review focuses on the membrane separation of NO_x and SO_2. The concepts presented in Section 2.1 are useful in understanding Section 2.2 as well as Section 5.1, which explains the usage of single-stage membrane models.

2.1 Membranes: An Overview

A membrane is often defined as "a selective barrier between two phases"[9]. Due to its simplicity, no moving parts, light weight, and small volume requirements, it is an attractive choice for a separation unit operation. A typical membrane process is illustrated in Figure 2-1 and characterized by the three flow streams: feed, retentate, and permeate.

Figure 2-1
Single-Stage Membrane Unit



A wide variety of membrane processes exist. Transport may be through pores or a dense film. The selective material may be a liquid or solid phase. The mechanism of transport may be active or passive, and the separated phases may be liquid, gas, or both. Although some of the concepts presented in this section are applicable to all membrane operations, this work focuses on the use of nonporous rubbery polymers for gaseous separations.

Permeability (Q) is a material property that expresses the molar flux (J) of a chemical species (i) passing through a membrane normalized by the membrane thickness (d) and partial pressure difference (ΔP_i). Permeability is dependent upon environmental conditions such as pressure, temperature, and concentration of gaseous species. Even a membrane material's history or previous exposure to such environmental conditions may affect the permeability.

$$Q_i = J_i d / \Delta P_i \qquad \textbf{Equation 2-1}$$

Permeability through a dense membrane is understood by the solution-diffusion model, which is described by a sequence of events. A chemical species absorbs at the surface of a membrane where a high concentration exists. The molecules diffuse through the membrane down a concentration gradient. Then the molecules desorb on the low concentration side. The magnitude of permeability is defined as the product of the diffusivity (D) and the solubility (S).

$$Q_i = D_i S_i \qquad \textbf{Equation 2-2}$$

Diffusivity, a kinetic property, is a measure of how fast a molecular species can move through the membrane material once it has sorbed to the surface. Solubility, a thermodynamic property, is a measure of the quantity of a chemical species within the membrane material. Environmental conditions affect a polymers diffusivity and solubility. Therefore, permeability is not a constant.

At sufficiently low pressures (and a given set of conditions) the solubility is nearly a constant, and the concentration of a chemical species within a polymer can be modeled by using Henry's Law. Henry's Law states that the equilibrium concentration (C) within a material exposed to a gas is equal to the product of the partial pressure of the gas (P) and a constant, the Henry's Law constant or solubility (S).

$$C_i = S_i P_i$$ **Equation 2-3**

Transport through a thin nonporous polymer is one-dimensional. Fick's Law for one-dimensional transport in Cartesian coordinates says that the flux of a chemical species is equal to a constant (diffusivity, D) times the negative of the concentration gradient.

$$J_i = -D_i \frac{dC_i}{dz}$$ **Equation 2-4**

One can derive Equation 2-5 from Equations 2-2 to 2-4. In the following equation P_h and P_l represent the respective pressures on the high and low pressure sides of the membrane. The mole fraction of a gas on the high and low pressure sides of a membrane are represented by x and y, respectively. The membrane thickness is represented by d.

$$J_i = -\frac{Q_i}{d}(x_i P_h - y_i P_l)$$ **Equation 2-5**

The flow rate of a gaseous species through a membrane can be thought of as the amount of gas leaving the feed side (-F) or the amount of gas entering the permeate side (G). It is calculated by multiplying the flux by the membrane area (A).

$$-F_i = \frac{Q_i A}{d}(x_i P_h - y_i P_l)$$ **Equation 2-6**

The selectivity (α) of a membrane is a measure of how well the membrane will separate a pair of gases, i and j. This quantity, along with the magnitude of permeability itself, is strongly considered when comparing membrane materials for a specific separation. It is simply a ratio of permeabilities.

$$\alpha_{ij} = Q_i / Q_j$$ **Equation 2-7**

By using the solution-diffusion definition of permeability in Equation 2-2, selectivity can be described by two physically different parts: diffusivity selectivity ($\alpha_{D,ij}$)and solubility selectivity ($\alpha_{S,ij}$).

$$\alpha_{ij} = \frac{D_i}{D_j}\frac{S_i}{S_j} = \alpha_{D,ij}\alpha_{S,ij}$$ **Equation 2-8**

2.2 The State of a Polymer and the Chemical Species in Flue Gas

A brief explanation of a polymer's physical state is important in the understanding of diffusivity selectivity and solubility selectivity. Polymers have a glass transition temperature (T_g) beneath which they are considered glassy and above which they are considered rubbery. On a molecular level, rubbery polymer molecules tend to move around a lot but are held together by intermolecular entanglements. They can easily shift a section of their molecular backbone to let a permeating molecule pass. Glassy polymer molecules, however, are relatively locked in place. A permeating molecule must be small enough to pass through existing gaps between molecular backbones or gaps created by small scale motions. In general, rubbery polymers exhibit larger permeabilities than glassy polymers.

The means by which the two states of polymeric membrane materials separate a pair of gases is quite different. Since large molecules have difficulty passing through a rigid glassy polymer matrix, glassy polymers characteristically have higher diffusivity selectivities than rubbery polymers. Rubbery polymers separate gases almost exclusively based on solubility selectivity.

A good measure of a gaseous species solubility within any polymer is its critical temperature. One should note that this property is independent of the membrane material itself. A pure substance cannot exist as a liquid above its critical temperature. The substance is resistant to aggregating or to intimate contact with any other substance. Therefore, a substance above its critical temperature is less likely to sorb on a membrane's surface than a substance below its critical temperature. As a rule of thumb, the higher the

critical temperature the higher the solubility, and in the case of a rubbery polymer, the higher the permeability.

The following table lists the physical properties of the most significant chemicals found in flue gas [10]. The volume percentages within the flue stream represent that of a coal-fired power plant [3]. Based on the critical temperature the following compounds: water (H_2O), SO_2, NO_2, and carbon dioxide (CO_2) are likely to be the most permeable materials through a rubbery membrane. Likewise, nitric oxide (NO) should have a low permeation rate according to its critical temperature. Since about nine times more nitric oxide exists in a flue gas stream than nitrogen dioxide [3], there is an inherent problem in using a single rubbery polymeric membrane to separate significant amounts of SO_2, NO_2, and NO from the other gaseous species.

Table 2-1

Components of Flue Gas

Gas	% volume of stream	MW	Tc (C)	Tb (C)	Size (A)
N_2	68-76	28.013	-147.1	-195.8	3.64
O_2	3-9	31.9988	-118.8	-183.0	3.46
CO_2	8-15	44.01	31.1	-78.5 (sub)	3.3
H_2O	5-12	18.015	374.15	100	2.65
SO_2	0.1-0.4	64.06	157.2	-10	3.6
NO	<0.1	30.01	-94.0	-151.8	3.17
NO_2	≈NO/9	46.01	157.8	21.15	
CO	?	28.01	-139.0	-191.5	3.76

2.3 Screening Polymeric Membrane Materials for SO_2 and NO_2

The scientific literature contains a number of papers on the separation of SO_2 with polymeric materials. Very little information, however, has been published concerning the permeability of NO or NO_2 through dense polymer membranes. Therefore, this section will only report the results of SO_2 membrane research. These membranes are speculated to be useful for a simultaneous separation of NO_2.

The table in Appendix A summarizes much of the data found on the subject of polymeric membrane materials for the separation of SO_2, N_2, O_2, and/or CO_2.

In most papers, polymers have been identified due to their high solubility of SO_2. Most of these polymers are glassy at room temperature, but the best ones tend to be rubbery. A number of researchers have been successful with increasing the solubility and permeability of SO_2 by adding a filler or modifying the polymeric backbone [4-6, 11-14]. Since all the membrane materials studied are solubility selective, the reviewed literature and the data compiled in Appendix A allow a few generalizations.

1. The permeability of SO_2 increases non-linearly with pressure or SO_2 partial pressure. This is due to an increase in solubility with pressure. Henry's Law is not followed at relatively high pressures.
2. The diffusivity of SO_2 increases exponentially with temperature, but the solubility of SO_2 decreases with temperature. Therefore, the permeability of SO_2 may increase or decrease with temperature. Permeability, solubility, and diffusivity have been shown to follow Arrhenius type relationships for several polymeric materials [4, 6, 7, 15, 16].
3. The selectivity of SO_2 over N_2 may be higher for a mixed gas than a pure gas [6, 7, 13].
4. The addition of a filler or humidity can increase the permeability of SO_2 [3, 4, 6, 12-15].

Of the polymers identified in Appendix A, the most outstanding material is PEBAX (a polyether block polyamide copolymer of unknown proportions). PDMS has the highest reported permeability of SO_2, but has a SO_2/N_2 selectivity of 50 [17]. The SO_2 permeance of PDMS, however, is less than PEBAX (thickness was not reported) [3]. PEBAX has a selectivity of 700.

2.4 Membrane System Optimization

Chemical processes are characterized by a number of system variables. A system response is a measurable result or output of a process, and it is dependent upon changes in the system variables. Optimization is the process of finding the values of system variables producing the most favorable system responses. In a membrane system, the system response may be the stage-cut (the fraction of permeate flow rate to feed flow rate), percentage of a species separation, profitability, and/or cost.

Only two authors discuss the optimization of membrane separation systems in the literature [18-21]. Both discuss the cost optimization of separating CO_2 from natural gas using thin film polymer membranes.

Natural gas may not be transported in pipelines with a CO_2 content of more than 2% [18, 20]. Any membrane, selective toward CO_2, is capable of reducing the CO_2 content, but some methane (valuable product) will always be lost in the permeate stream. In other words, as more CO_2 is separated from the feed gas, more CH_4 permeates the membrane, and the permeate flow rate increases. Therefore, the optimization problem is to minimize the cost of the membrane system (capital costs and operating costs) and the expenses associated with losing methane in the permeate stream.

The separation of CO_2 from natural gas is similar to the separation of NO_x and SO_2 from flue gas. In both cases, one wishes to separate the pollutant from the feed gas to a specified limit, while minimizing the volume of feed gas lost to the permeate stream. For the CO_2 separation, the maximum CO_2 concentration in the permeate stream (the minimum permeate flow rate) will minimize methane losses. For the NO_x and SO_2 separation, the maximum pollutant concentrations in the permeate stream (the minimum permeate flow rate) will minimize treatment costs.

Although a single-stage membrane system is simple and requires the least capital investment, it may not be the most cost effective. The performance of any single-stage membrane system is limited by the properties of the membrane and the system variable constraints. If more than one membrane stage is utilized, however, the performance can be enhanced. A number of membrane configurations are possible, and each configuration has many system variables. This makes the optimization of a membrane system for a particular separation a difficult task.

Both authors studied the following system configurations.

Figure 2-2

Single-Stage Membrane Configuration

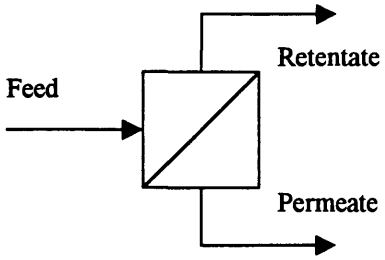


Figure 2-3

Two-Stage Membrane Configuration

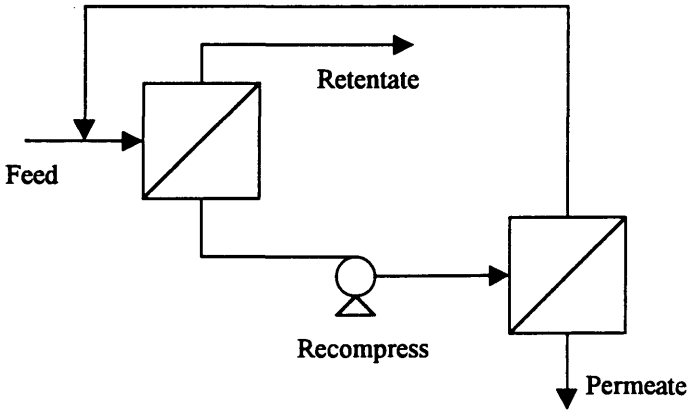
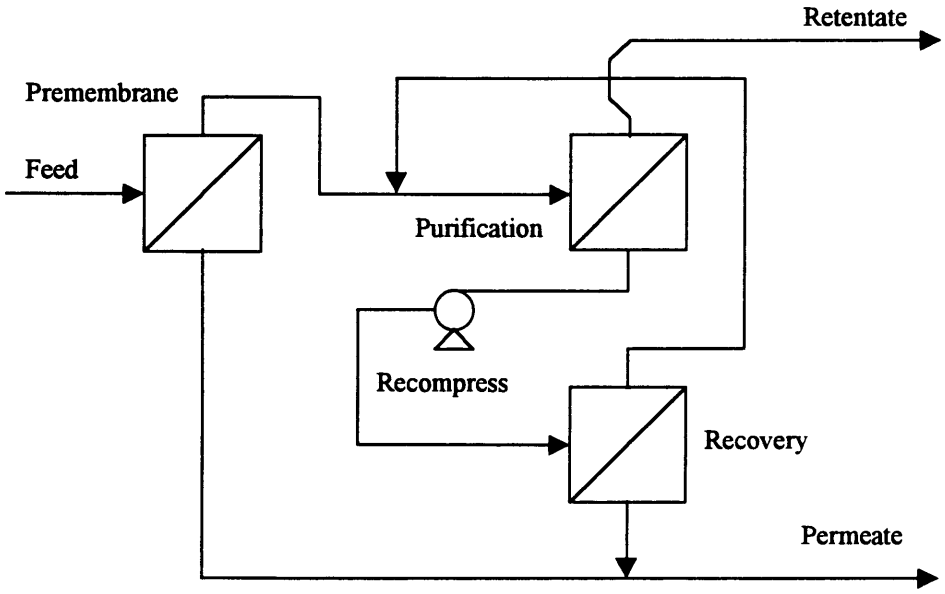


Figure 2-4

Three-Stage Membrane Configuration



Spillman showed that to achieve the same separation (2% CO_2 in the retentate with a minimized CH_4 loss), the membrane area requirement for the three-stage configuration is much less than the one or two stage configurations [21]. He concluded that an optimized three-stage configuration was the most cost effective.

Bhide and Stern performed a more detailed analysis. They investigated the use of recycling a ratio of the permeate and retentate streams from the last stage of each configuration [18]. They concluded that the above configurations, without recycle, were optimal for the CO_2 separation. They also concluded that depending upon the feed mole fraction of CO_2, either the two or three-stage configuration is the most cost effective.

The one and three-stage configurations are investigated in this work.

Chapter 3

PROCESS DESCRIPTION

The pilot-plant consists of two major processes. A membrane is used to separate and concentrate the NO_2 and SO_2 from a flue gas into a low volume permeate stream, and a glass column is used to treat the concentrated permeate stream with a dry sorbent. Each process is modular and can easily be replaced to investigate a variety of membrane and sorbent materials. Figure 3.1 is a diagram of the existing system, and Tables 3.1 and 3.2 provide the specifications for the major equipment and instruments. Further discussion of the pilot-plant's components and operation are located in Section 3.1, Section 3.2, and in previous thesis work [22].

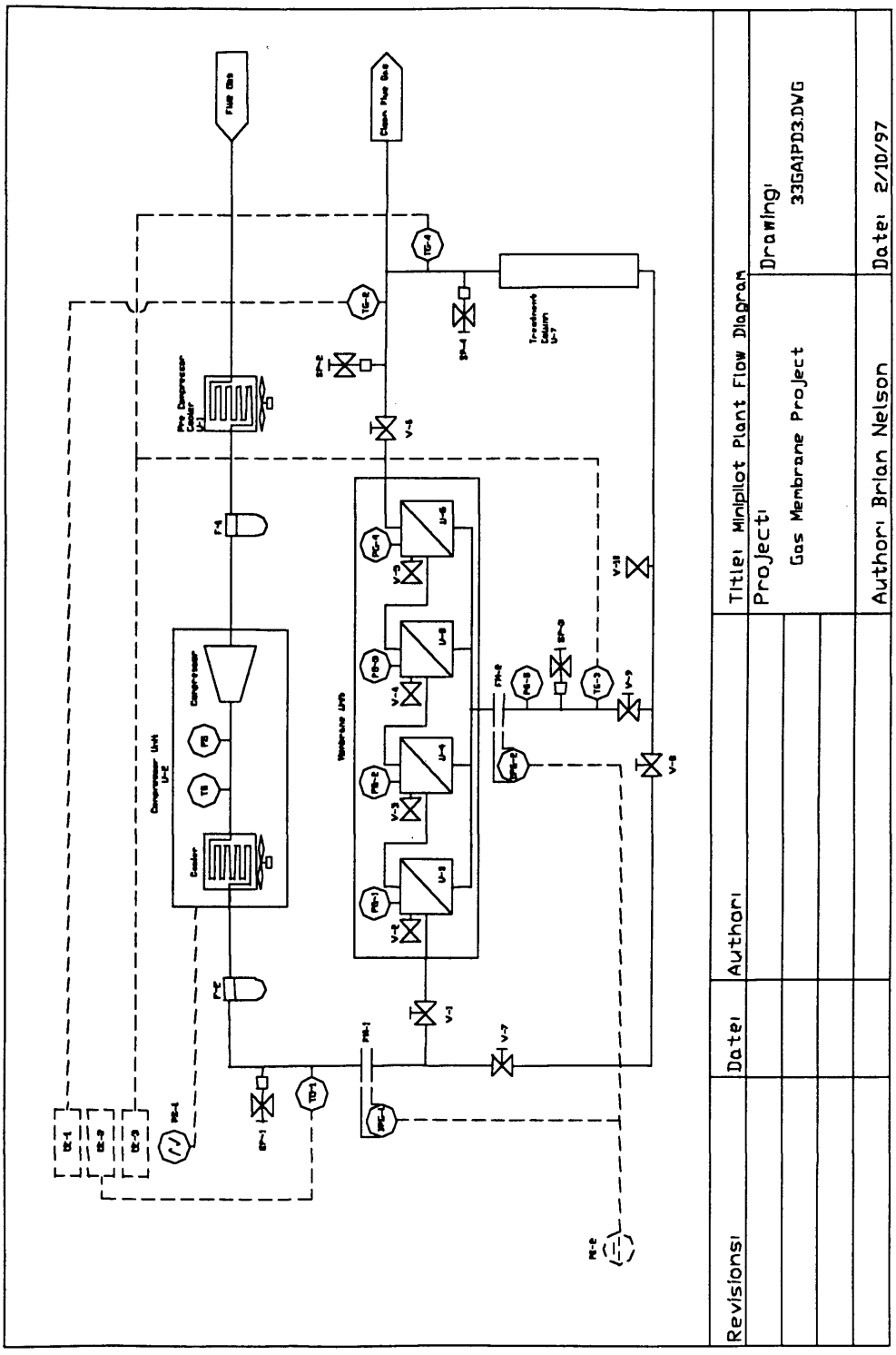


Figure 3.1
Pilot Plant Flow Diagram

Table 3-1

Pilot-plant Equipment List

Label	Equipment	Comments
	Piping	1 in. black steel pipe
U-1	Air Cooler	Ultra Air, Model UAA-240, 240 scfm max, ½ hp fans
U-2	Compressor Unit	Palatek compressor and air cooler, Model 40D, 40 hp, 166 acfm @ 125 psig, cooler 106,300 Btu/hr
PS-1	Compressor Power Source	3/60/460 volt
U-3 to U-6	Pressure Vessels and Membrane	Steel Fab, cylindrical vessels 0.25" thick, 18"O.D. and 29.25"long, rated to 60 psig @ 200°F, containing 100 sqft of PDMS membrane each
F-1	Water Filter	Ultra Air, Model UF18
F-2	Oil Filter	Ultra Air, Model UAC-150P-03
SP-1 to SP-4	Sample Ports	Whitey Ball Valve, Model SS-42S4
V-1, V-7 to V-9	Ball Valve	Apollo, Model 73-100
V-6	Globe Valve	Rockwell International, Model A105QT
V-2 to V-5, V-10	Pressure Relief Valve	Nupro, Model 55-HCA1-3, 3-50 psi
U-7	Glass Column and Fittings	Allen Scientific, 5ft length, 6in. diameter

Table 3-2

Pilot-Plant Instrument List

Label	Instrument	Comments
DPG-1, DPG-2	Differential Pressure Meter	Rosemount, Model 1151 DP-4E12-M2B3, 0-100 in. water
FM-1	Orifice Plate and Flange Union	Daniel, Calculation #95-4031, 1.049 in. bore, 120 scfm max, 1 in. 30RT carbon steel flange
FM-2	Orifice Plate and Flange Union	Daniel, Calculation #95-4032, 0.7511 in. bore, 100 scfm max, 1 in. 30RT carbon steel flange
TG	Temperature Gauge	Attached to Palatek Compressor, 130°F-250°F
PG	Pressure Gauge	Attached to Palatek Compressor, 0-200psig
PG-1 to PG-4	Pressure Gauge	NOSHOK, Model 40-400-100 (0-100psig)
PG-5	Pressure Gauge	NOT YET INSTALLED (0-20 psig)
PS-2	Power Supply	ATM Zagreb, Model 49.15.401, 24 VDC 100mA output, 115 AC input
TG-1 to TG-4	Thermocouple	Omega, Model CF-090-T-2-60-2, 0-200°C, 0-90 psig
EE-1, EE-2	Alarm Board	Omega, Model DP461T
EE-3	MultiChannel Alarm Board	Omega, Model DP462T

3.1 System Variables for Pilot-plant Membrane Separation

This section describes the pilot-plant's system variables and operating limits relevant to membrane separation. Non-numerical variables include flow pattern, membrane system configuration, and membrane material. Numerical variables include the following: membrane thickness, moisture, feed concentrations, permeate pressure, permeate side pressure changes, feed pressure, feed side pressure drop, feed flow rate, membrane area, and temperature. This information is used to model the pilot-plant's membrane performance. Tables 3.3 and 3.4, at the end of this section, summarize this information.

The pilot-plant is currently set up as four series single-stage cocurrent (see Section 5.1.1) membrane modules on the retentate stream. The pilot-plant can easily be modified to a counter-current flow pattern, but the crosscurrent flow pattern is also considered since it is industrially important. Since cascade systems have reported favorable results in the literature [18-21], the three-stage configuration is also considered.

The pilot-plant contains plate and frame membranes made of PDMS, which are easily removed or replaced. Assuming plate and frame membranes of PEBAX are available, PEBAX is also evaluated.

The thinnest reasonable membrane thickness is always preferred. These values are fixed to the manufacturers' specifications. The PDMS membrane is 0.0025 cm thick [17]. PEBAX has no reported thickness, but Baker et al. reported the permeance (permeability without the thickness adjustment) of several flue gas species [3].

Flue gas moisture has been experimentally demonstrated to enhance the selectivity of SO_2 over nitrogen [3] without reducing the SO_2 permeation rate. The combination of SO_2 and moisture in a gaseous stream, however, causes corrosion concerns for steel piping [22], so the pilot-plant was designed to reduce moisture levels as much as possible. Daily changes in the weather vary compressor inlet air temperature, pressure, humidity levels, and the cooling air used in the post compressor cooler. A water filter was installed on the pilot-plant after the compression stage, which keeps the moisture levels consistently low during operation. Cooling flue gas and removing moisture probably removes some SO_2 and NO_2. The moisture content of the flue gas within the membrane units is assumed negligible.

A typical flue gas contains a mixture of the following gaseous species: nitrogen, oxygen, carbon dioxide, carbon monoxide, water, sulfur dioxide, nitric oxide, nitrogen oxide, and products of incomplete combustion. Table 2.1 shows the approximate concentrations of each pollutant in a flue gas. In this study carbon monoxide, water, nitric oxide, and products of incomplete combustion are neglected. Nitrogen, oxygen, and carbon dioxide concentrations are held at approximately 0.8, 0.08, and 0.12 mole fraction,

respectively. Sulfur dioxide and nitrogen oxide concentrations are held at 5000 ppm and 400 ppm respectively.

The permeate side pressure is approximated as atmospheric pressure in the Denver area, 12 psia, and permeate side pressure changes are assumed to be negligible.

The compressor is capable of supplying ambient air at 125 psig. The start-up tests at RFETS show a compressor supply of only 45 psig [8]. The membrane manufacturer recommends the pressure differential remain below 70 psi. The pressure relief valves, V-2 to V-5, are set to release pressures over 50 psig, and the pressure vessels are rated for 60 psig. A previous study recommended the pilot-plant's feed pressure lie between 30 psig and 50 psig [23]. Assuming a compressed gas can be regulated to any pressure below 125 psig and higher rated pressure vessels are installed, the membrane performance is simulated at feed pressures 30 psig, 50 psig, and 70 psig.

Initial tests on the pilot-plant show a feed side pressure drop of about 5 psi per 100 sqft of membrane [8]. The pilot-plant was not evaluated over a range of many system variables, and the possibility of a leak is significant. Although the modeling programs are capable of including a feed side pressure drop, it is not addressed. This creates a slight over estimate concerning the membrane performance.

The compressor is capable of supplying approximately 126 scfm, and the pilot-plant contains 400 sqft of membrane area. Holding all other variables constant, a ratio of feed flow rate to membrane area always produces the same simulated results. Assuming the compressed gas can be split, a portion of the compressed gas is fed to the pilot-plant, and a portion of the compressed gas is vented, the membrane performance is evaluated over a range of feed flow rate to membrane area ratios. During simulations, the membrane area is held at 100 sqft, and the feed flow rate is varied.

A thin film rubbery polymer cannot withstand the high temperatures of a combustion flue gas. A typical combustion flue gas has temperatures around 300°F [2,3]. In order for the PDMS membrane to maintain integrity, temperatures must be held below 212°F [24]. The permeability of polymeric materials increases with temperature due to thermal expansion. At the same time, however, the molecular discrimination (selectivity) of the membrane decreases. All available permeability data is only valid near room temperature. Therefore, the pilot-plant performance simulations are valid at approximately 25°C.

Table 3-3

Non-Numerical Variables

Membrane Materials	Flow Patterns	System Configuration
PDMS PEBAX	cocurrent countercurrent crosscurrent	Single-Stage Three-Stage Configuration

Table 3-4

Numerical Variables

System Variable	Set Value or Range
Moisture	0
N_2 concentration	0.8 mole fraction
O_2 concentration	≈0.08 mole fraction
CO_2 concentration	0.12 mole fraction
SO_2 concentration	5000 ppm
NO_2 concentration	400 ppm
Membrane Thickness	PDMS = 0.0025 cm PEBAX = N/A
Membrane Area	100 sqft
Temperature	25°C
Feed Flow Rate	0.05-1000 scfm
Feed Pressure	42, 62, 82 psia
Feed Side ΔP	0
Permeate Pressure	12 psia

3.2 Problems During Initial Testing

Testing of the pilot-plant at RFETS began and ended on November 22, 1995. Data from these tests prove unreliable, and a proper material balance can not be calculated on the system.

The major problem encountered during startup was the inability to accurately measure flow rates without restricting flow. In order to calculate the flow rate of a compressible fluid through a calibrated orifice meter (FM-1 and FM-2), one must measure the fluid temperature, pressure drop across the orifice plate, and the absolute pressure either upstream or downstream of the orifice plate. During the testing, pressure differential meters, DPG-1 and DPG-2 (see Figure 3.1), did not function since the power supply (PS-2) had not yet been installed. Additionally, the permeate-side pressure gauge, PG-5, does not yet exist on the pilot-plant.

Flow rates were measured using one dry gas meter. The dry gas meter was attached to the retentate and permeate lines at different times. The back-pressure created by the flow meter probably changes the flow dynamics of the entire system.

Additionally, the gas analyzer was connected to only one sample port at a time. If feed concentrations were constant, this would not have been a problem, but feed concentrations varied quickly and widely during the 4 hour test [8].

For these reasons, the data from November 22, 1995 is not analyzed in this report.

Chapter 4

LABORATORY EXPERIMENTATION

The permeability of a gas through a polymer is not constant. Permeabilities of N_2, O_2, CO_2, and SO_2 through a number of materials are reported to vary with temperature and partial pressure [3-7, 13-15, 25]. A bench scale apparatus was assembled to evaluate material permeabilities of N_2, O_2, CO_2, SO_2, and NO_2 over a range of conditions. These tests, however, require accurate gas analysis, and gas analysis instrumentation was not available until September of 1997. No permeabilities are found, but tests were run to document the use of the bench scale apparatus for potential future experimenters.

Tests were run using permeate flow rate as the measured output. A factorial experiment was run on PDMS membrane material (supplied by Membrane Products Corporation, Albany, NY) with temperature, SO_2 concentration, and NO_2 concentration as variables. These tests show that the presence of SO_2, up to 5000 ppm, and the presence of NO_2, up to 400 ppm, have no measurable affect on the permeabilities of N_2, O_2, or CO_2 for a feed pressure of 50 psia, a feed flow rate of 1 slpm, a membrane area of 11.6 sqcm, and a temperature range of 20-50°C.

Another experiment shows that the total permeance of the PDMS membrane material increases linearly and nearly doubles between 15 and 85°C.

4.1 Bench Scale Apparatus

A diagram of the bench scale apparatus appears in Figure 4-1, and Table 4-1 describes the individual equipment. Since Scott Specialty Gas would not mix SO_2 and NO_2 in the same tank, and SO_2 and NO_2 have relatively low vapor pressures (approximately 3.2 atm and 1 atm at room temperature respectively [26]), three gas tanks and mass flow controllers are used to mix the feed gas. The system is capable of supplying a feed flow of 5 slpm with concentrations up to 4041 ppm SO_2 and up to 405 ppm

NO_2. For a feed flow rate of 1 slpm, this system provides up to 20205 ppm SO_2 and up to 2025 ppm NO_2. Between a 1 and 5 slpm feed flow rate, the feed will always have greater than 6.4% CO_2.

Figure 4-1

Bench Scale Apparatus for Evaluating Membrane Materials

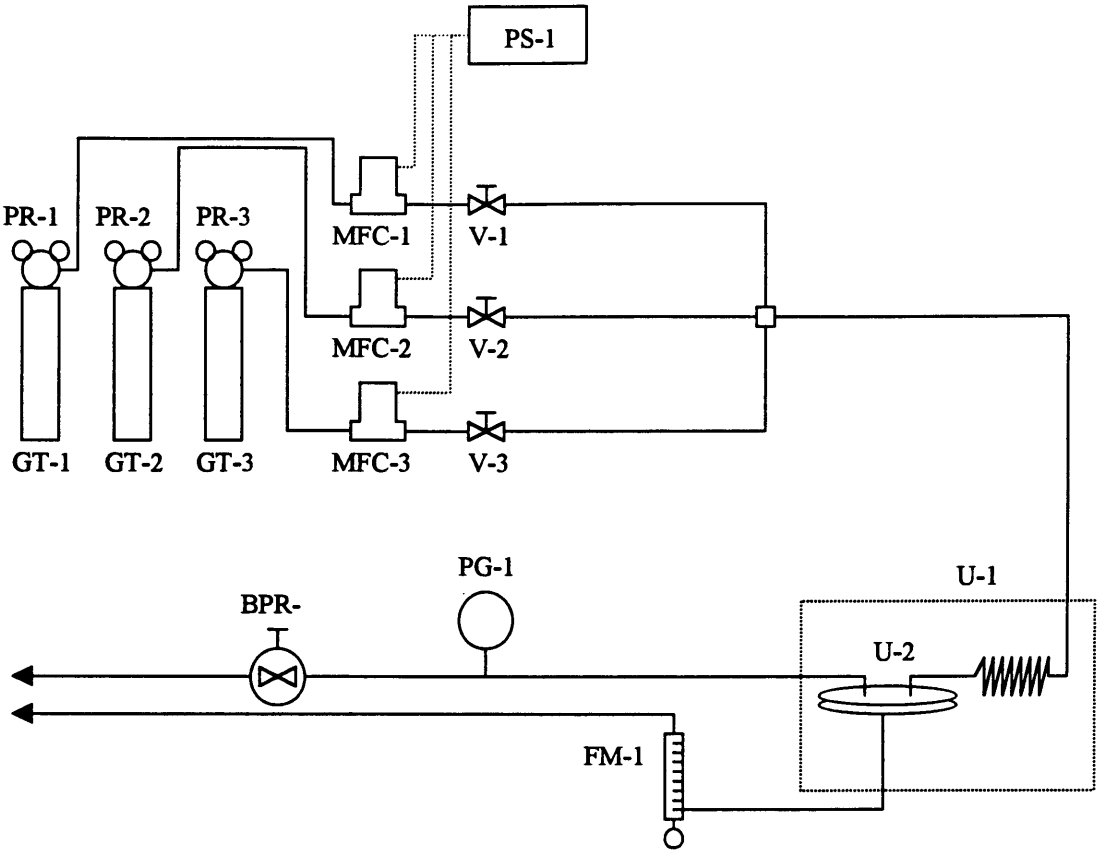


Table 4-1

Equipment for the Bench Scale Apparatus

Label	Equipment	Comments
	1/8 inch stainless steel tubing	
GT-1	Gas Tank	Scott Specialty Gas, Certified Master Gas, 9.9% CO_2 and balance of air, 2015 psia
GT-2	Gas Tank	Scott Specialty Gas, Certified Master Gas, 8.98% SO_2 and balance of air, 215 psia
GT-3	Gas Tank	Scott Specialty Gas, 1.5% NO_2 and balance of air, 950 psia
PR-1	Pressure Regulator	Matheson, model 3104-580, 0-2000 psig, 0-100 psig
PR-2	Pressure Regulator	Scott, model 5113A180, 0-3000 psig, 0-100 psig
PR-3	Pressure Regulator	GO, model 103173 ss316L, 0-1000 psig, 0-100 psig
MFC-1	Mass Flow Controller	Tylan General, model FC-260V, Calibrated for 10% CO_2 and balance Air, 5 slmp max
MFC-2	Mass Flow Controller	Tylan General, model FC-260KZ, Calibrated for 9% SO_2 and balance Air, 225 sccm max
MFC-3	Mass Flow Controller	Tylan General, model FC-260V, Calibrated for 1.5% NO_2 and balance Air, 135 sccm max
PS-1	Power Source	Tylan General, model RO-28-115-25ext
V-1, V-2, V-3	Plug Valves	Whitey, model ss4152
U-1	Temperature Bath	Poly Science, model 8205
U-2	Membrane Cell	Millipore, model xx4404700
FM-1	Bubble Flow Meter	50ml buret
PG-1	Test Gauge	0-60 psig
BPR-1	Back Pressure Regulator	GO, model A11115e14, 0-50 psig

A temperature bath (U-1) is used to control the membrane and feed gas temperature, and membrane cell (U-2) is used to safely mount membrane materials under pressure. A test gauge (PG-1) and back-pressure regulator (BPR-1) are used to control the feed pressure, and a bubble meter (FM-1) is used to measure the permeate flow.

With the addition of three 3-way valves, on the feed, retentate, and permeate, this system can easily be adjusted to collect gases in a sample bag or evacuated gas bomb. Although collecting gas from the feed and retentate is not necessary, it is good practice to validate the performance of the mass flow controllers and gas analyzer.

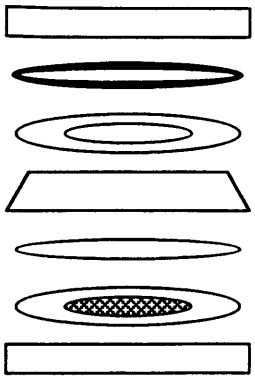
4.2 Procedure

The following procedure is used to gather experimental data.

1. Cut square (5 cm per side) piece of membrane material and measure its thickness with a micrometer (Ames 220-2 serial 56212M).
2. Mount membrane material in the membrane cell as shown in Figure 4-2.
3. Attach membrane cell to bench scale apparatus, submerse in temperature bath, and bring to temperature.
4. Open pressure regulators, turn on mass flow controllers, and adjust back pressure regulator.
5. Measure atmospheric pressure.
6. Wait 10 minutes and measure permeate flow rate with bubble meter and stop watch.

Figure 4-2

Mount Membrane Material in Membrane Cell



Cell Top

Viton 41 mm O-Ring

Viton Gasket with 38.4 mm Hole

Membrane Material

Whatman 540 Porous Filter Paper

Steel Support with 38.4 mm Diameter of Porous Surface

Cell Bottom

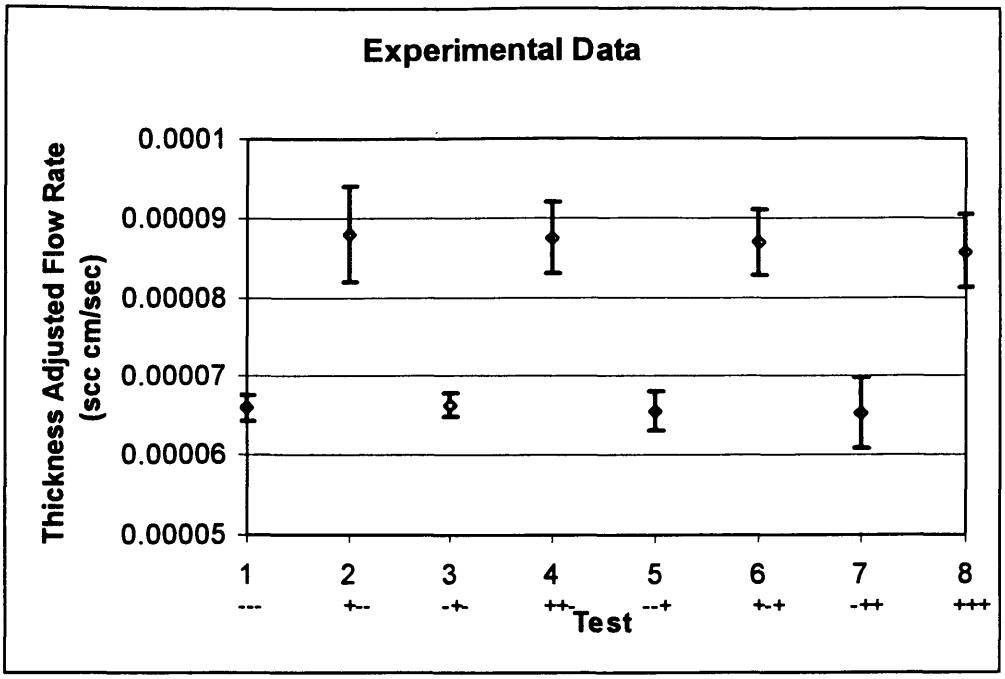
4.3 Experimental Data

A factorial design is a quick experimental method for determining the statistically significant variables and variable interactions of any process [27]. Temperature, SO_2 concentration, and NO_2 concentration are tested for PDMS while keeping feed pressure (38 psig), permeate pressure (approximately 12.0 psia), feed flow rate (1 slpm), and membrane area (11.6 sqcm) constant. Eight experiments are required to test three variables at two levels. A new piece of membrane material is used for each series of eight randomly ordered experiments. Figure 4-3 shows all of the experimental results. Positive (+) and negative (-) symbols are used to identify each experiment. The first symbol represents the temperature (+ = 50°C and - = 20°C). The second symbol represents the SO_2 concentration (+ = 5000 ppm and - = 0 ppm), and the third symbol represents the NO_2 concentration (+ = 400 ppm and - = 0 ppm).

Since the thickness of each membrane is different, the measured flow rate (in standard cubic centimeters per second) is multiplied by the thickness. The reported units are referred to as the thickness adjusted flow rate.

Figure 4-3

Experimental Data



A summary of the experimental results is in Table 4-2. The results of identical experiments from each of the three series are averaged and the variance is calculated. A pooled standard deviation of 3.98E-06 is calculated from these variances. The pooled standard deviation represents the amount of error introduced by the experimenter, equipment, and instrumentation. The effect is a measure of how each variable and variable interaction affects the thickness adjusted flow rate. The pooled standard deviation is larger than the absolute value of all the effects except the temperature, which means all other variables and interactions are attributed to experimental noise [27]. Consequently, only the temperature affects the overall permeance over the range of variables tested. Since the most prevalent species in the permeate flow rate are N_2, O_2, and CO_2, one can conclude that the effect of SO_2 and NO_2 concentration is less than the pooled standard deviation. Therefore, when using this membrane material (over the range of variables tested) in a membrane separation system, the stage-cut is not expected to change significantly as the concentrations of SO_2 and NO_2 change.

Another set of tests was conducted to determine the change in the total permeance over a range of temperatures. Figure 4-4 shows the change in the adjusted permeate flow rate versus temperature. As one

can see, the effect is mostly linear, and the adjusted permeate flow rate nearly doubles from 15 to 85°C. Therefore, when using this membrane material (over the range of variables tested) in a membrane separation system, the stage-cut will vary significantly with a change in temperature.

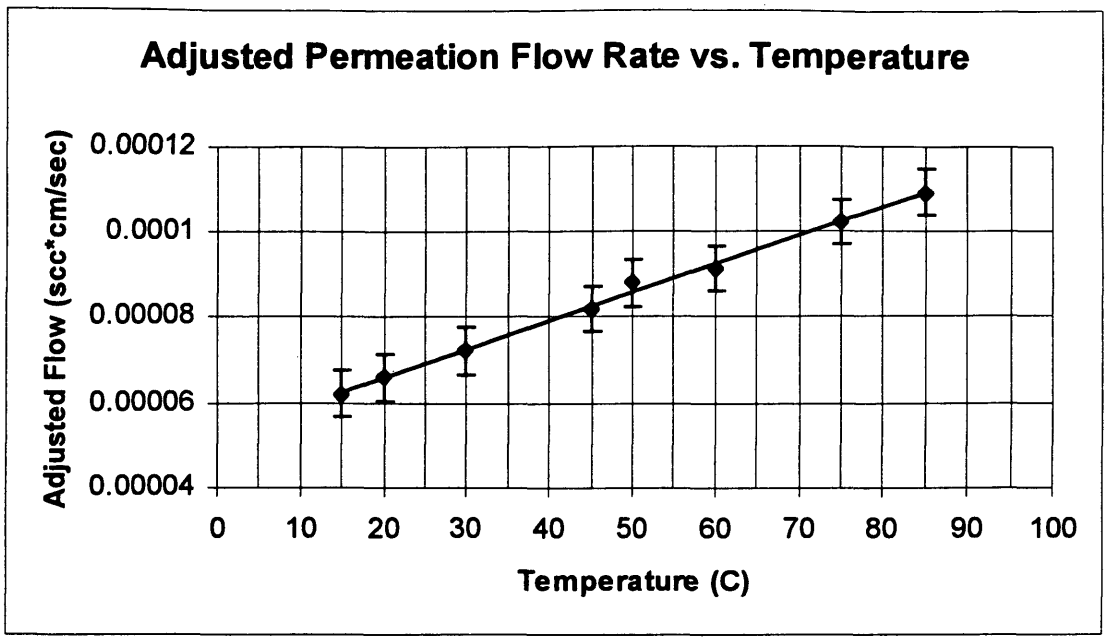
Table 4-2

Factorial Design Results

Symbols	Variable or Interaction	Average	Variance	Effect
---	**average**	6.60E-05	2.66E-12	7.64E-05
+--	**Temp.**	8.80E-05	3.58E-11	2.12E-05
-+-	**SO_2**	6.64E-05	2.15E-12	-3.66E-07
++-	**Temp. - SO_2**	8.74E-05	2.12E-11	-4.34E-07
--+	**NO_2**	6.56E-05	6.14E-12	-1.09E-06
+-+	**Temp. - NO_2**	8.68E-05	1.78E-11	-3.19E-07
-++	**SO_2 - NO_2**	6.53E-05	2.01E-11	-3.08E-07
+++	**Temp. - SO_2 - NO_2**	8.58E-05	2.11E-11	4.98E-08

Figure 4-4

Adjusted Permeate Flow Rate vs. Temperature



Chapter 5

MODEL DEVELOPMENT AND MATHEMATICA

Beginning in the late 1940s, a number of papers have discussed the modeling of two component, three component, or multi-component gas separation over a single-stage permeator at steady-state. [28-47] All of the listed papers are based on the Fick's Law mechanism of transport (Equation 2-6 or something similar). Modeling approaches cover a number of membrane flow patterns and module configurations, which may be very specific. This chapter outlines the modeling equations and solution methods of Shindo et al. [38]. This chapter also discusses the use of Mathematica 3.0 for calculating modeling objectives not discussed by Shindo.

5.1 Single-stage Models

Depending upon the desired output values, the modeling equations can be thought of as a solution to a problem. These are two of the most common problems:

Problem 1. Find the stage-cut (θ, the fraction of permeate flow rate to feed flow rate) and concentrations in the retentate (x_{o1}, x_{o2},..., x_{on}) and permeate (y_1, y_2,..., y_n), given the inlet concentrations (x_{i1}, x_{i2},..., x_{in}), permeabilities (Q_1, Q_2,..., Q_n), pressures (P_h and P_l), membrane thickness (d), and membrane area (A_t).

Problem 2. Find the membrane area (A_t) and concentrations in the retentate (x_{o1}, x_{o2},..., x_{on}) and permeate (y_1, y_2,..., y_n), given inlet concentrations (x_{i1}, x_{i2},..., x_{in}), permeabilities (Q_1, Q_2,..., Q_n), pressures (P_h and P_l), membrane thickness (d), and stage-cut (θ) [38].

Modeling equations may exist as differential equations or integrals (solved numerically) or algebraic equations (solved iteratively).

The paper by Shindo et al. [38] derives ordinary differential equations for the calculation of five different membrane flow patterns (cocurrent, countercurrent, crosscurrent, perfect mixing on the permeate side, perfect mixing on both sides). Their models are not specific to any module configuration (like spiral wound or hollow fibers), and are derived with the following assumptions:

1. All components in the feed stream are permeable.
2. The ideal gas law is valid, and standard volume is valid in keeping track of material balances.
3. Membrane thickness is constant.
4. The membrane area on the feed and permeate sides are equal.
5. Membrane operation is isothermal.
6. Perfect plug flow conditions occur where necessary.
7. Diffusion along the flow path is insignificant compared to bulk flow.
8. Fick's Law is valid.
9. The permeability of each gas component is a constant.
10. The pressure changes of feed and permeate gas streams are negligible.

The following ordinary differential equations are a solution to Problem 1 for a single-stage membrane module with cocurrent, crosscurrent, and countercurrent flow patterns (For Problem 2, a solution method different from Shindo's is discussed in Section 5.2.). These equations are taken directly from Shindo's paper. The cocurrent configuration is the easiest to model and quickest to run on a computer, but it is rarely used in practice. The crosscurrent and countercurrent configurations are much more valid for practical use applications. The crosscurrent configuration closely resembles a spiral wound membrane module, and the countercurrent configuration is often used in hollow fiber modules (either tube side or shell side feed). The equations include an overall material balance (Equation 5-1), a species material balance (Equation 5-2), and the constraint that the sum of mole fractions in any stream is equal to unity (Equations 5-3 and 5-4). In Equations 5-1 to 5-4 the following symbols are used: molar flow through the membrane (F), membrane area (A), membrane thickness (d), permeability (Q), partial pressure (P), and mole fractions (x,y). Due to the constraint that the sum of mole fractions equals unity, one species is not directly accounted for in a differential equation.

$$-\frac{dF}{dA} = \frac{1}{d}\sum_{k=1}^{n} Q_k (x_k P_h - y_k P_l) \qquad \textbf{Equation 5-1}$$

$$-\frac{dx_i}{dA} = \frac{1}{Fd}\left[Q_i (x_i P_h - y_i P_l) - x_i \sum_{k=1}^{n} Q_k (x_k P_h - y_k P_l) \right] \qquad \textbf{Equation 5-2}$$

$$x_n = 1 - \sum_{k=1}^{n-1} x_k \qquad \textbf{Equation 5-3}$$

$$y_n = 1 - \sum_{k=1}^{n-1} y_k \qquad \textbf{Equation 5-4}$$

The following sections provide additional equations, to use in conjunction with the above equations, which are specific to each membrane flow pattern. These sections also include a brief explanation of the solution methods and Mathematica [48, 49] functions necessary in carrying out the simulation calculations. The code for programs Co5 (cocurrent model for 5 gaseous species), Counter5 (countercurrent model for 5 gaseous species), and Cross5 (crosscurrent model for 5 gaseous species) contain a detailed explanation of applicable Mathematica functions. They are located in Appendix B, C, and D. For a complete explanation of the derivation of these models refer to the paper by Shindo.

Shindo recommends using a Runga-Kutta-Gill numerical method to estimate a solution. Mathematica's NDSolve command is used to find a solution to the set of ordinary differential equations. The NDSolve command, however, contains several different numerical methods (including both 4^{th} and 5^{th} order Runga-Kutta methods and variable step sizing routines) for calculating a solution. NDSolve may run through the calculation several times before meeting its precision requirements. Since the NDSolve code is complex (about 500 pages long [49]), further reference to the numerical solution method is the NDSolve command.

Shindo's models include the use of dimensionless numbers. Although it is not pointed out by Shindo, the use of the dimensionless numbers he defines are the only reason assumptions 9 and 10 are necessary. Although some numerical precision is sacrificed without the use of dimensionless numbers, answers are quite consistent without them. Without the use of dimensionless numbers, feed pressure, permeate pressure, and permeabilities do not have to be constants. These values can be written as a function of other system variables and can be recalculated at each step of the numerical procedure. The models in Appendix B, C, and D account for a feed side pressure drop relative to the exposed membrane area. The models can easily be adapted to utilize changes in permeability relative to the system pressures, temperature, or concentrations.

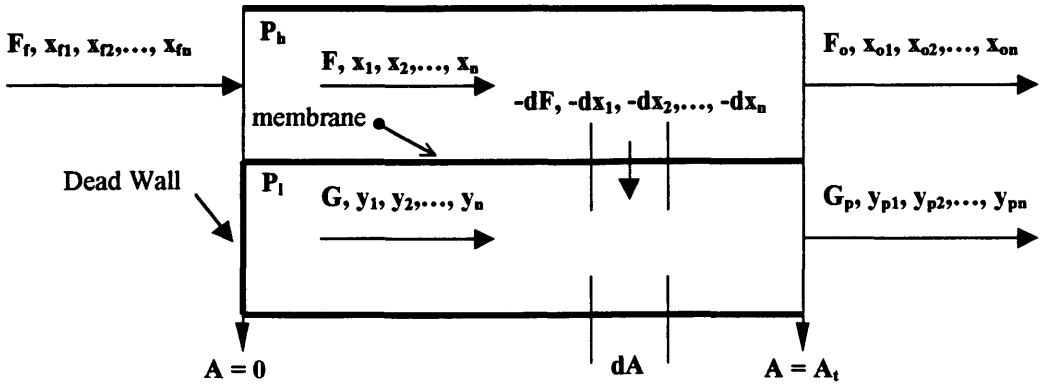
A comparison between the results of the models developed in this work and the results published by Shindo appear in Appendix E.

5.1.1 Cocurrent Flow Pattern

Figure 5-1 shows a diagram of a cocurrent membrane flow pattern, which labels several of the variables used in Equations 5-1 and 5-2 as well as the following equations.

Figure 5-1

Cocurrent Flow Pattern



These equations are used in conjunction with Equations 5-1 to 5-4, and keep track of mole fractions during the numerical solution to ordinary differential equations 5-1 and 5-2.

$$G = F - F_f \qquad \textbf{Equation 5-5}$$

The permeate side mole fraction can be calculated at any point along the membrane, except where G=0 or A=0, through a material balance.

$$y_i = \frac{x_{i,f}F_f - x_i F}{F_f - F}, \quad where\ G \neq 0\ for\ (i = 1,2,...,n-1) \qquad \textbf{Equation 5-6}$$

Where G=0 (A=0), the dead wall on the permeate side (see Figure 5-1), the mole fractions are calculated with the following equations.

$$\sum_{k=1}^{n} \frac{x_k Q_k / Q_i}{P_l / P_h\left[(Q_k / Q_i) - 1\right] + (x_i / y_i)} = 1, \quad where\ G = 0 \qquad \textbf{Equation 5-7}$$

$$y_j = \frac{x_j Q_j / Q_i}{P_l / P_h\left[(Q_j / Q_i) - 1\right] + (x_i / y_i)}, \ where\ G = 0\ and\ j \neq i\ or\ n \qquad \textbf{Equation 5-8}$$

Equation 5-8 is found through solving Equation 5-6 in the limit as $F \rightarrow F_f$ ($A \rightarrow 0$) by using L'Hospital's rule, and Equation 5-7 is found by applying Equation 5-4 to Equation 5-8. The value of y_i in Equation 5-7 can not be isolated, but it can be solved using an iterative procedure such as Newton's method. Mathematica's FindRoot command uses Newton's method [49], which requires a reasonable initial guess for y_i.

Equations 5-7 and 5-8 allow for the calculation of permeate side mole fractions where there is no bulk flow along the permeate side. The permeate side mole fractions and partial pressure differences at any

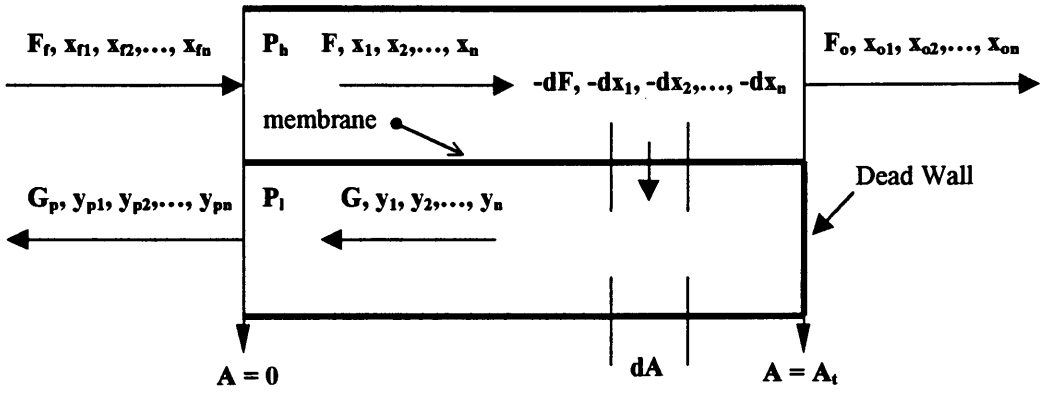
point along a membrane, where there is no bulk flow, is basically induced by the system pressures, permeabilities, and feed side mole fractions.

5.1.2 Counter-Current Flow Pattern

Figure 5-2 shows a diagram of a counter-current membrane flow pattern, which labels several of the variables used in Equations 5-1 and 5-2 as well as the following equations.

Figure 5-2

Countercurrent Flow Pattern



These equations are used in conjunction with Equations 5-1 to 5-4, and keep track of mole fractions during the numerical solution to ordinary differential Equations 5-1 and 5-2.

$$G = F - F_o \qquad \textbf{Equation 5-9}$$

The permeate side mole fraction can be calculated at any point along the membrane, except where G=0 ($A=A_t$), through a material balance.

$$y_i = \frac{x_i F - x_{i,o} F_o}{F - F_o}, \quad where\ G \neq 0\ for\ i = (1,...,n-1) \qquad \textbf{Equation 5-10}$$

Even though the dead wall is located at $A=A_t$, where G=0, and Equations 5-7 and 5-8 are derived from Equation 5-6, which is specific to the cocurrent configuration, Equations 5-7 and 5-8 are valid for the counter-current flow configuration. The exact same equations arise when applying L'Hospital's rule in the limit of $F \rightarrow F_o$ to Equation 5-10.

The complication in finding the numerical solution to the countercurrent flow pattern is that the permeate flow rate and retentate mole fractions are unknown. Therefore, permeate mole fractions at the dead wall, at A=At, are unknown, and partial pressure differences are also unknown along the entire membrane area. This solution must be found through trial and error. The retentate mole fractions and stage-cut are guessed, and the inlet concentrations and flow rate are calculated by running through the numerical solution procedure from $A=A_t$ to A=0. If the calculated flow rate and inlet concentrations are sufficiently close to the given values, the problem is solved, otherwise, a new guess is made. Two papers briefly discuss a solution method for the countercurrent model, but neither paper provides any specifics. Shindo recommends starting with the output from a cross flow model (see next section) and then using Powell's nonlinear optimization method. Thundyil et al. [47] also recommends starting with the output from the cross flow model and then using the golden section method of constrained optimization.

Perhaps the simplest approach to finding a solution is through an iterative procedure. Starting with the output from the cocurrent flow model, permeate flow rate and retentate concentrations, the next guess for permeate flow rate and retentate concentrations can be calculated through an identity and a material balance.

$$G_{p,next} = F_{f,k} - (F_{f,given} - G_{p,k}) \qquad \textbf{Equation 5-11}$$

$$x_{o,i,next} = \frac{(F_{f,given} x_{f,i,given}) - y_{p,i,k} G_{p,next}}{(F_{f,given} - G_{p,next})}$$ **Equation 5-12**

In the above equations, k represents the number of iterations. To prevent the iterative procedure from diverging, each newly calculated value of $x_{o,i,next}$ or $G_{p,next}$ is averaged with the previously used value ($G_{p,k}$ in Equation 5-13 and $x_{o,i,k}$ in Equation 5-14). The weighting factor, ϕ, is between zero and one and allows for the calculation of a weighted average. Each equation may have a different value for ϕ. The lower the value of ϕ, the faster each new value will change, but there is a greater chance of divergence. The higher the value of ϕ, however, the less each new value changes, and it will take longer for the procedure to converge on a solution.

$$G_{p,k+1} = (1-\phi) G_{p,next} + \phi G_{p,k}$$ **Equation 5-13**

$$x_{o,i,k+1} = (1-\phi) x_{o,i,next} + \phi x_{o,i,k}$$ **Equation 5-14**

This method works reasonably well, but problems arise when extreme conditions (a very high separation percentage, a very high stage cut, or a membrane with a very high selectivity) are defined for a separation.

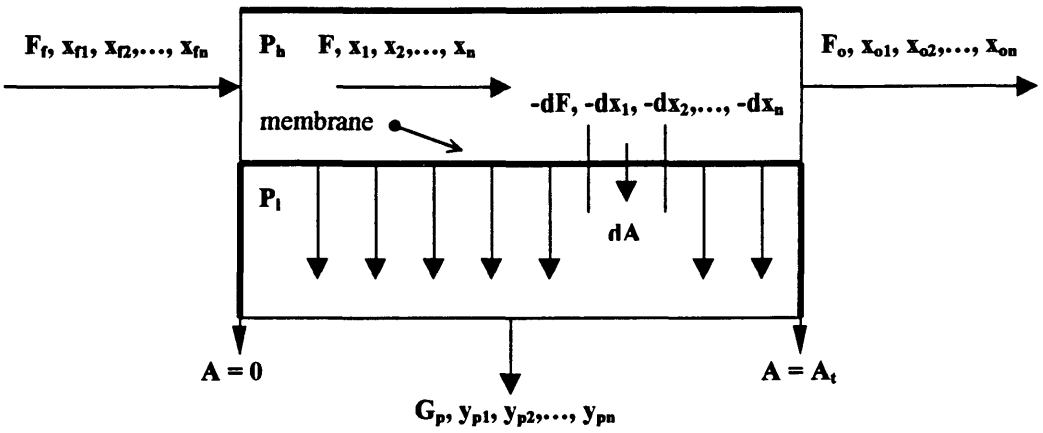
The NDSolve function in Mathematica will not carry out the numerical procedure backwards, where the independent variable, A, goes from a high value, A_t, to a low value, 0. Therefore, a coordinate transformation was made on equations 5-1 and 5-2, where $Z=A_t-A$.

5.1.3 Crosscurrent Flow Pattern

Figure 5-3 shows a diagram of a crosscurrent membrane flow pattern, which labels several of the variables used in Equations 5-1 and 5-2 as well as the following equations.

Figure 5-3

Crosscurrent Flow Pattern



There is no accumulation along the permeate side of the membrane. Consequently, there is no equivalent to equations 5-5 or 5-9. We can, however, write an equation for the material balance around the whole system. This equation is valid for all flow patterns.

$$F_f x_{f,i} = F_o x_{o,i} + G_p y_{p,i} \qquad \textbf{Equation 5-15}$$

Since there is no accumulation or mixing on the permeate side of a crosscurrent membrane, the permeate mole fractions are calculated at each step of the numerical solution procedure using equations 5-7 and 5-8. Equation 5-5 is valid for the crosscurrent flow pattern when $F=F_o$ and $G=G_p$.

5.2 Other Modeling Objectives and Mathematica

When Mathematica solves the NDSolve function the solution is also a function, referred to as an InterpolatingFunction [49]. An InterpolatingFunction can be renamed something like fint[area] (the feed side flow rate as a function of membrane area) or x1int[area] (the species 1 feed side mole fraction as a function of membrane area). The InterpolatingFunction stores a table of values for fint[area] or x1int[area], then interpolates this table to find an approximate solution when area is specified.

Although it's not the most efficient method to solve Problem 2, explained in the previous section, the InterpolatingFunction can be used to find the required membrane area for a specified separation. In other words, we can find the membrane area (A_t) and output concentrations in the retentate ($x_{o1}, x_{o2},\ldots, x_{on}$) and permeate ($y_1, y_2,\ldots, y_n$), given inlet concentrations ($x_{i1}, x_{i2},\ldots, x_{in}$), permeabilities ($Q_1, Q_2,\ldots, Q_n$), pressures ($P_h$ and P_l), membrane thickness (d), and stage-cut (θ). The stage-cut can be replaced by: a desired retentate mole fraction of a particular species or the total percentage of separation of a particular species. To do this we:

1. Run a program of desired flow pattern with a membrane area larger than necessary for the desired stage-cut, retentate mole fraction, or percentage separation.
2. Specify a function to minimize.

$$stage\ cut - \frac{feedflow - f\,\mathrm{int}[area]}{feedflow}$$

$$retentate\ mole\ fraction - x_i\,\mathrm{int}[area]$$

$$percentage\ separation - \frac{feedflow \times x_{f,i} - f\,\mathrm{int}[area] \times x_i\,\mathrm{int}[area]}{feedflow \times x_{f,i}}$$

3. Use the Mathematica's FindMinimum command to determine what value for area gives the minimum value.
4. Run the program again using the area which minimizes the desired equation.

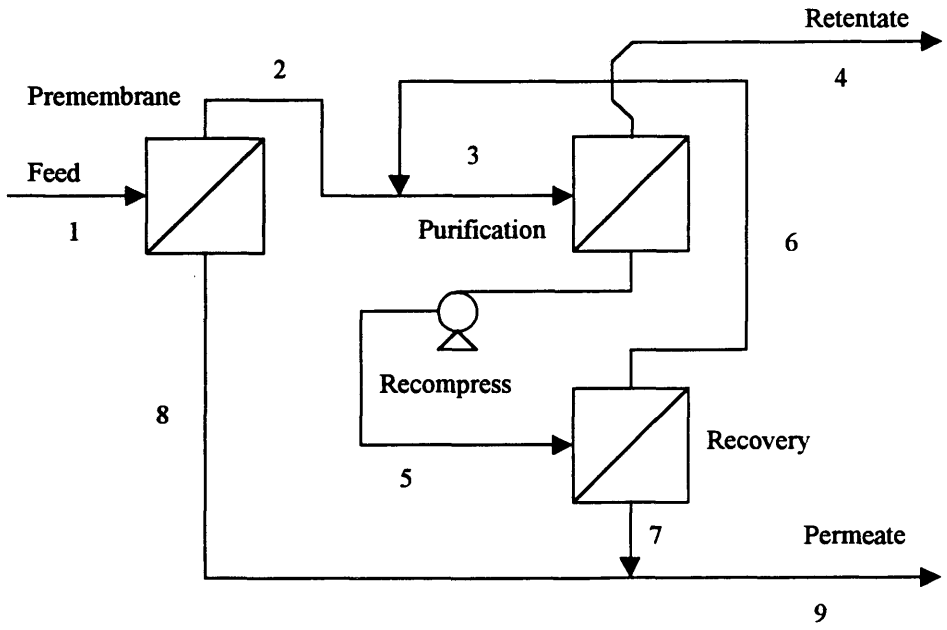
The code and an example of each procedure is in Appendix G.

5.3 Three-Stage Configuration

The three-stage configuration was discussed in Section 2.4. Figure 5-4 shows the three-stage configuration. This system has been recommended by two authors [18-21] for the separation of CO_2 from natural gas. Each membrane stage (premembrane, purification, or recovery) may contain different flow patterns, membrane materials, membrane area, feed pressures, or permeate pressures. In this analysis, however, only the membrane area of each cocurrent stage is varied.

Figure 5-4

Three-Stage Membrane Configuration



Using one of the models in Appendix B, C, or D, a solution is found iteratively. The models must first be modified so that the input from "PART 1 – INPUT CONSTANTS AND VARIABLES" is erased and placed in the function designation brackets (preceding PART 1) followed by an underscore. For example, the old function designation looked like this:

```
co5[{pfeed_,pdrop_,pperm_,feedflow_,totalarea_}] := ( etc.
```

The new function designation will look something like this:

```
co5[{totalarea_, pfeed_, pperm_, feedflow_,
                  x1i_, x2i_, x3i_, x4i_, x5i_}] := ( etc.
```

This allows the output from one function to be used as the input to the next function. Also, the "Print" statements in co5, cross5, or counter5 should be turned off by enclosing them within a parenthesis-star (*like this*). The calculation procedure proceeds as follows:

1. The values for stream 1 are given, and streams 2 and 8 are calculated.
2. Stream 6 is an unknown at this point, so stream 3 is set equal to stream 2. Streams 4 and 5 are calculated for the first time.
3. Stream 5 is compressed to the original feed pressure and streams 6 and 7 area calculated.
4. Stream 6 is combined with stream 3, and the concentrations and flow rate of stream 3 are tested to see if they have changed according to a convergence criteria. If the criteria is met, all streams are calculated and printed. If the criteria is not met, then steps 3 and 4 are repeated until the convergence criteria is met.

Appendix H contains the code for the co5 program. As long as the function name is changed (co5 to cross5 or counter5), the code is the same for any of the programs for 5 species. For any other number of species, the lists (enclosed in curvy brackets {}) may have to be modified.

Chapter 6

RESULTS AND DISCUSSION

As discussed in Chapter 3, the pilot-plant contains 400 sqft of PDMS membrane and was originally designed for a feed flow rate of approximately 100 scfm and a feed pressure up to 62 psia [23]. The 400 sqft of membrane are placed in a series of four 100 sqft modules with a cocurrent flow pattern. Using the base case concentrations and separation conditions (see Section 3.1 or Figure 6-1 below), Figure 6-1 shows output from the cocurrent Mathematica model. Since the membrane area is divided into four equal areas of cocurrent flow pattern, the results are slightly better (similar to a crosscurrent flow pattern) than if they were one unit of 400 sqft in cocurrent flow pattern.

Even though the results are slightly better than a true cocurrent flow pattern and no pressure drop is taken into consideration, the output shows the pilot-plant will only separate approximately 14.3 percent SO_2 and 12.9 percent NO_2 with a 3.37 percent stage cut.

To completely evaluate the pilot-plant's performance, the modeling programs are run over a range of variables. Section 6.1 summarizes these results in the form of performance charts.

Sections 6.2 and 6.3 discuss the simulation results of attempting to enhance membrane performance by increasing the partial pressure difference of gas components or using a cascade configuration.

Figure 6-1

Pilot-plant Performance at Design Conditions

****

SEPARATION CONDITIONS AND PERMEABILITIES

feed pressure (psia) = 62

feed side pressure drop (psi) = 0

permeate pressure (psia) = 12

membrane area (sqft) = 400

membrane thickness (in) = 0.000984

species 1 = nitrogen

species 2 = oxygen

species 3 = carbon dioxide

species 4 = dioxide sulfur

species 5 = dioxide nitrogen

permeability of 1 (barrers) = 250

permeability of 2 (barrers) = 500

permeability of 3 (barrers) = 2700

permeability of 4 (barrers) = 12500

permeability of 5 (barrers) = 6350

FEED FLOW RATE AND MOLE FRACTIONS

feed flow (scfm) = 100

x1i = 0.8

x2i = 0.00746

x3i = 0.12

x4i = 0.005

x5i = 0.0004

RETENTATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 96.6307%

retentate flow (scfm) = 96.6307

x1 = 0.80952

x2 = 0.0741856

x3 = 0.111498

x4 = 0.00443632

x5 = 0.000360511

PERMEATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 3.36933%

permeate flow (scfm) = 3.36933

y1 = 0.526983

y2 = 0.0864836

y3 = 0.363834

y4 = 0.0211662

y5 = 0.00153254

SEPARATION RESULTS

SO2 % Separation = 14.2632

NO2 % Separation = 12.9091

Permeate % of feed flow = 3.36933

6.1 Single-Stage Separation Results

Performance charts are made to show the simulated separation results of a single-stage membrane system over a range of variables. Figures 6-2 through 6-7 show the stage cut, separation of SO_2, and separation of NO_2 according to the variables outlined in Chapter 3. Figures 6-2, 6-3, and 6-4 show the cocurrent, crosscurrent, and countercurrent results for PDMS respectively, while Figures 6-5, 6-6, and 6-7 show the performance of PEBAX (NO_2 permeability data is not available).

Since permeability data is for room temperature, these charts are only valid near 25°C. Chapter 3 specifies pollutant concentrations of 5000 ppm SO_2 and 400 ppm NO_2. Since pollutant concentrations are so low, they do not significantly affect the relative feed concentrations of N_2, O_2, or CO_2. As long as pollutant concentrations are relatively low, the relative partial pressure difference (driving force) of all species is the same along the membrane. Therefore, Figures 6-2 to 6-7 are valid for all pollutant concentrations where pollutant concentrations are at least an order of magnitude less than the major species. As pollutant concentrations increase, however, separation percentage and stage cut gradually increase.

Figure 6-2 is the only performance chart applicable to the pilot-plant's current configuration. Even though the pilot-plant consists of four separate cocurrent modules in series, this figure closely estimates its performance over a range of variables. For example, a ratio of feed flow rate to membrane area of 0.25 (100 scfm/400 sqft) and a feed pressure of 62 psia shows the same separation results on Figure 6-2 as the output in Figure 6-1.

All of the performance charts show the same general trends. The percentage separation of SO_2 and NO_2 increase with a decrease in the ratio of feed flow rate to membrane area. At very low feed flow rate to membrane area ratios, the stage cut rapidly increases.

Using a high percentage separation and a low stage cut as a basis for desirability, the performance of the countercurrent flow pattern is most desirable, the cocurrent flow pattern is least desirable, and the crosscurrent flow pattern is somewhere in the middle. For example, at a feed flow rate to membrane area ratio of 0.05 and a feed pressure of 62 psia, the PDMS and PEBAX membrane produce separations shown in Table 6-1. The rank in desirability of each flow pattern is consistent for any feed pressure or feed flow rate to membrane area ratio, since the flow pattern determines the driving force (partial pressure difference) along the membrane.

Upon comparing the separation results of PDMS and PEBAX, one notices that PDMS consistently separates more SO_2, and it has a higher stage cut. The observation that PDMS separates more SO_2 than

PEBAX is surprising, since PEBAX has a much higher SO_2 permeance. (The SO_2 permeances are 35,000,000 and 5,000,000 (Barrer/cm) for PEBAX and PDMS respectively.) This result is explainable since PEBAX has a much higher SO_2/N_2 selectivity than PDMS (50 and 700 respectively). The SO_2 driving force across the membrane is much lower for PEBAX than PDMS.

Figures 6-8 and 6-9 show a comparison of driving forces for the cocurrent flow pattern. Both curves show the same general trend of decreasing driving force, but the driving force over the PDMS membrane is consistently an order of magnitude greater than the PEBAX membrane. Appendix F provides the code for creating such figures in Mathematica.

A common but somewhat outdated industrial standard for a SO_2 treatment system is 90 percent separation [2]. The performance charts show that a 90 percent single-stage separation is possible, but the feed flow rate to membrane area ratio is very low, and an large amount of membrane is required. For a 62 psia feed pressure, the feed flow rate to membrane area ratio is less than 0.05 scfm/sqft. That means the pilot plant is capable of 90 percent SO_2 separation for a 20 scfm feed flow rate.

Figure 6-2

Cocurrent Flow Pattern, PDMS Membrane Performance Chart for Low SO_2 and NO_2 Pollutant Concentrations. (Permeate pressure = 12 psia, Membrane Thickness = 0.000984 in., Mole Fractions of Other gases: N_2 = 0.8, O_2≈0.08, and CO_2=0.12)

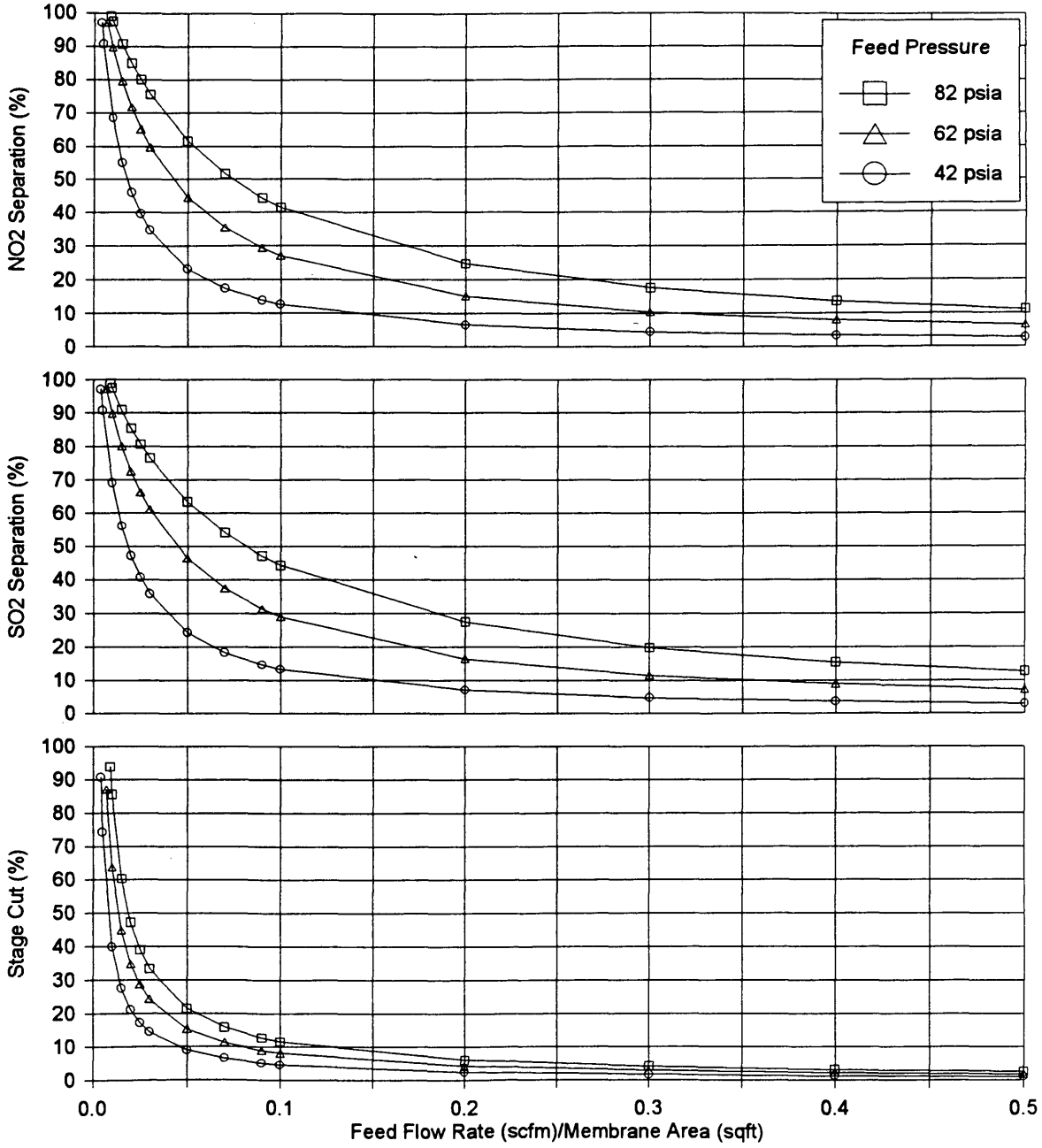


Figure 6-3

Crosscurrent Flow Pattern, PDMS Membrane Performance Chart for Low SO_2 and NO_2 Pollutant Concentrations. (Permeate pressure = 12 psia, Membrane Thickness = 0.000984 in., Mole Fractions of Other gases: N_2 = 0.8, O_2≈0.08, and CO_2=0.12)

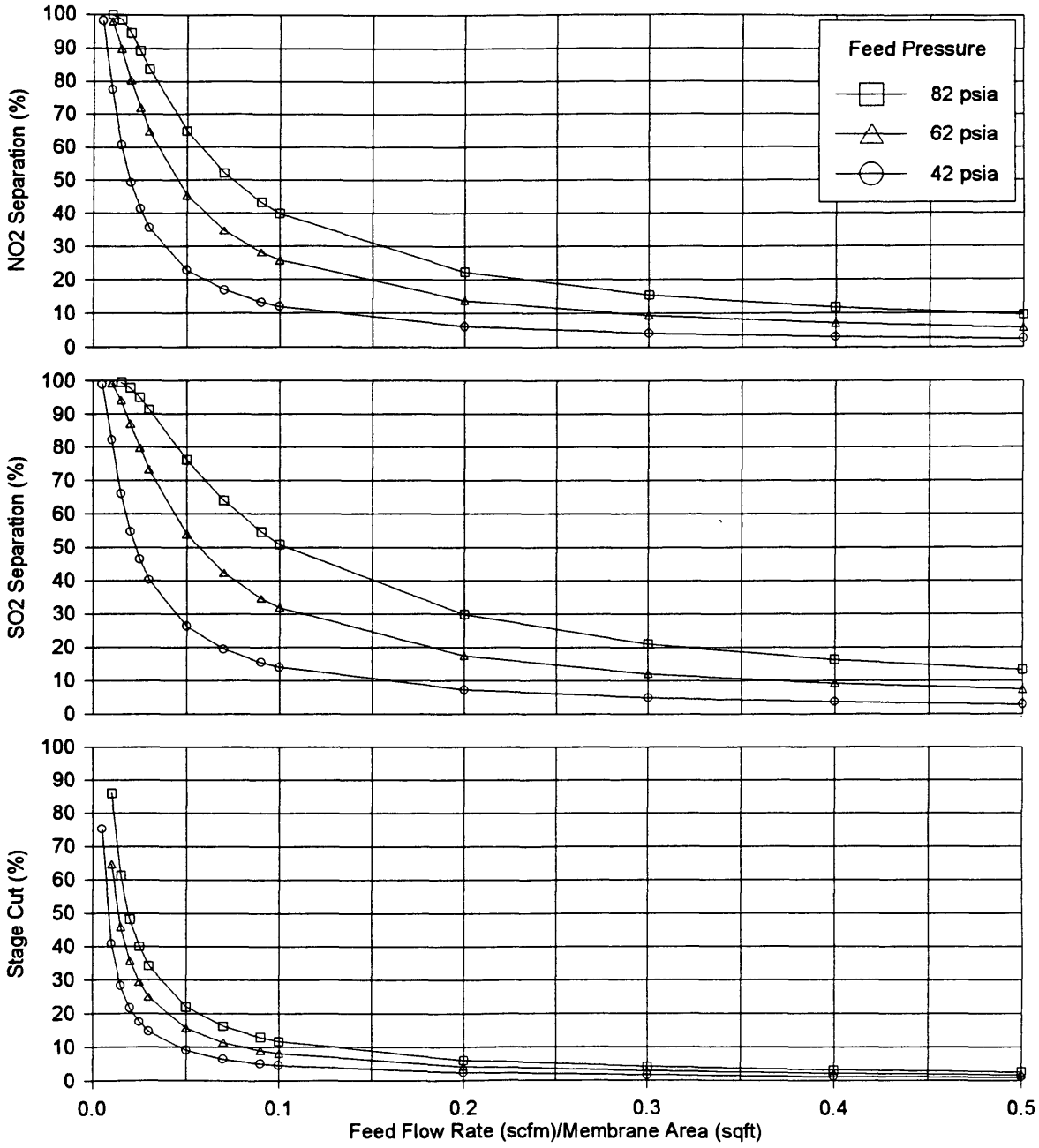


Figure 6-4

Countercurrent Flow Pattern, PDMS Membrane Performance Chart for Low SO_2 and NO_2 Pollutant Concentrations. (Permeate pressure = 12 psia, Membrane Thickness = 0.000984 in., Mole Fractions of Other gases: N_2 = 0.8, O_2≈0.08, and CO_2=0.12)

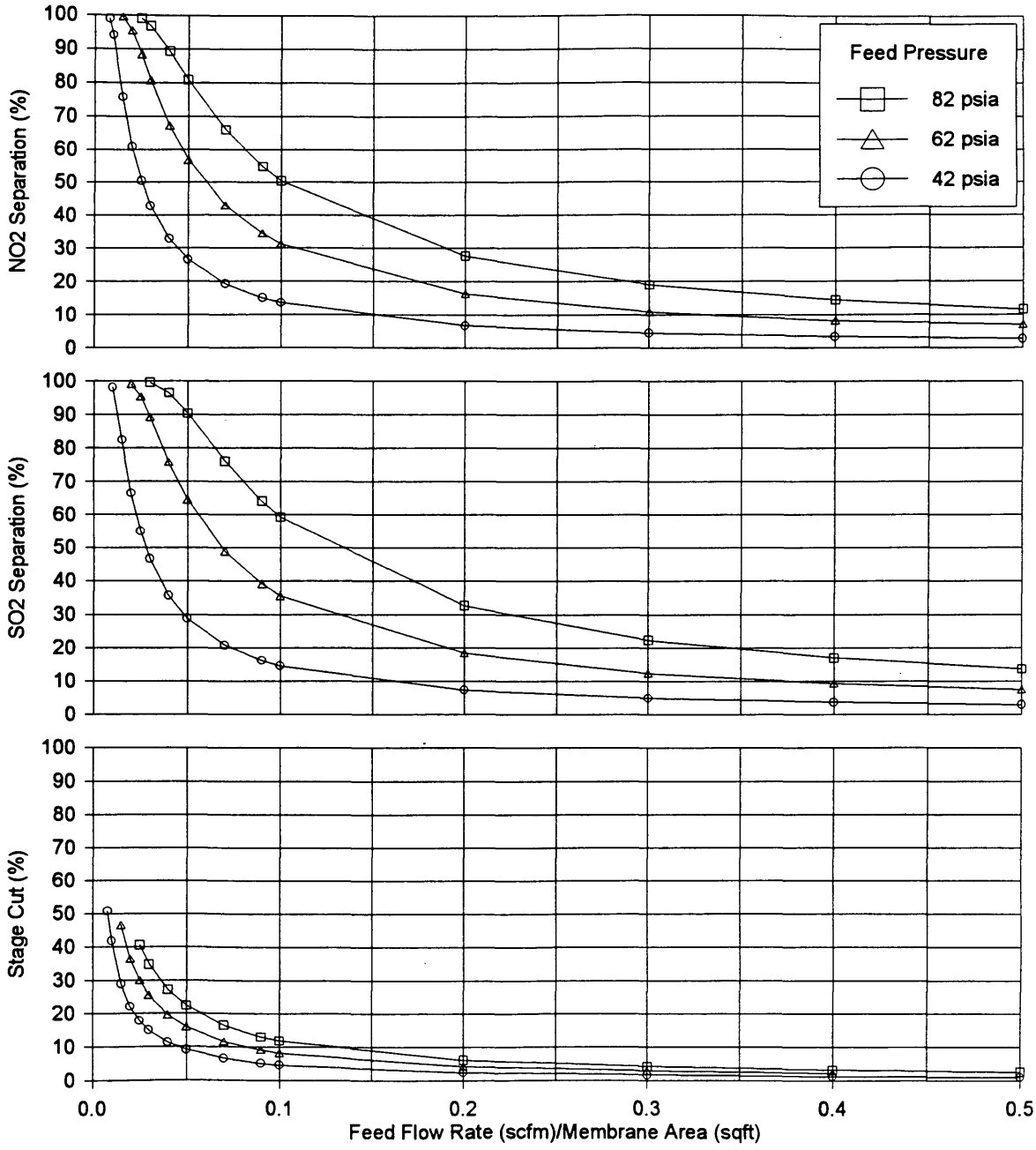


Figure 6-5

Cocurrent Flow Pattern, PEBAX Membrane Performance Chart for Low SO_2 and NO_2 Pollutant Concentrations. (Permeate pressure = 12 psia, Membrane Thickness = unknown, Mole Fractions of Other gases: N_2 = 0.8, O_2≈0.08, and CO_2=0.12)

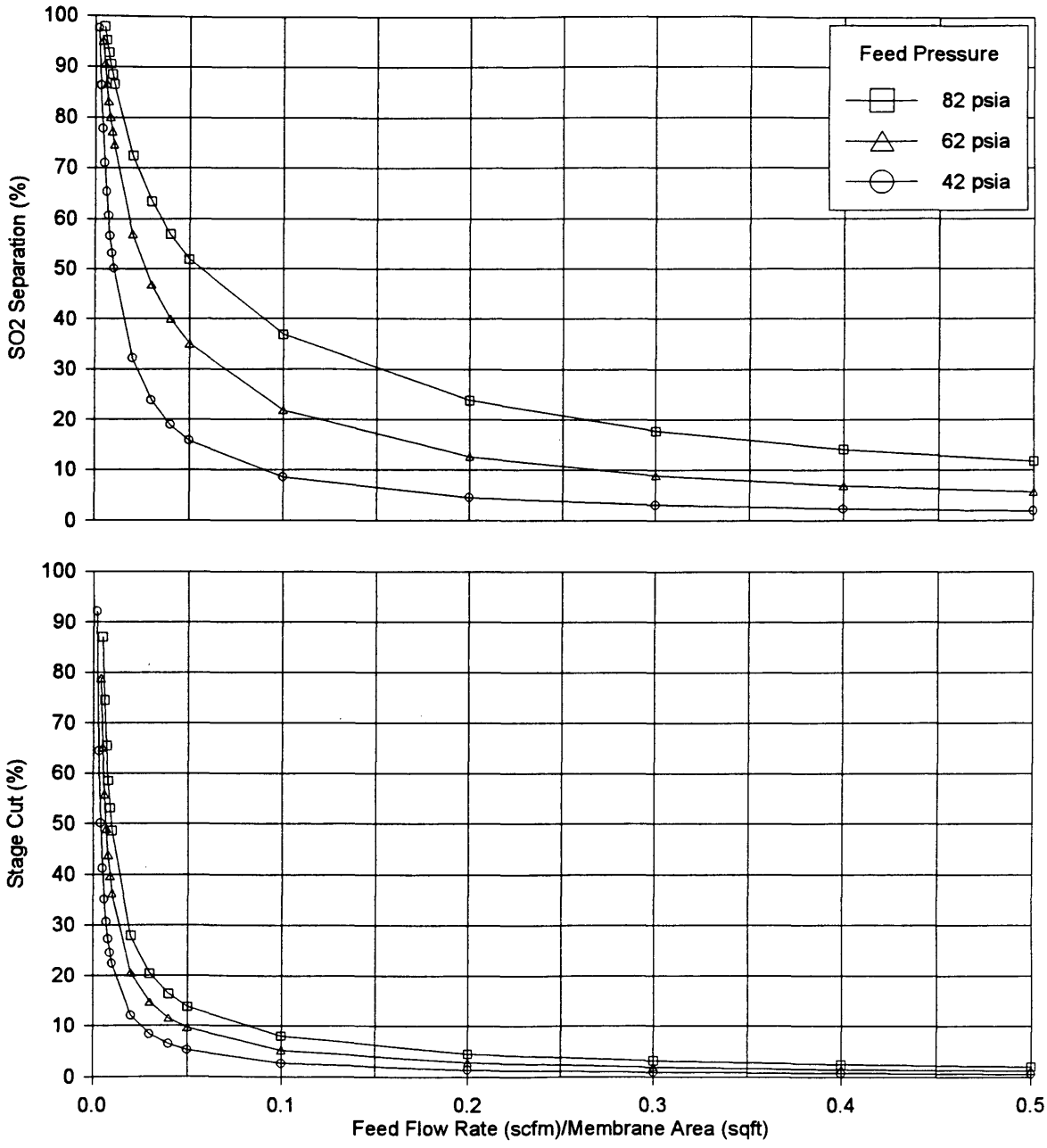


Figure 6-6

Crosscurrent Flow Pattern, PEBAX Membrane Performance Chart for Low SO_2 and NO_2 Pollutant Concentrations. (Permeate pressure = 12 psia, Membrane Thickness = unknown, Mole Fractions of Other gases: N_2 = 0.8, $O_2 \approx 0.08$, and CO_2=0.12)

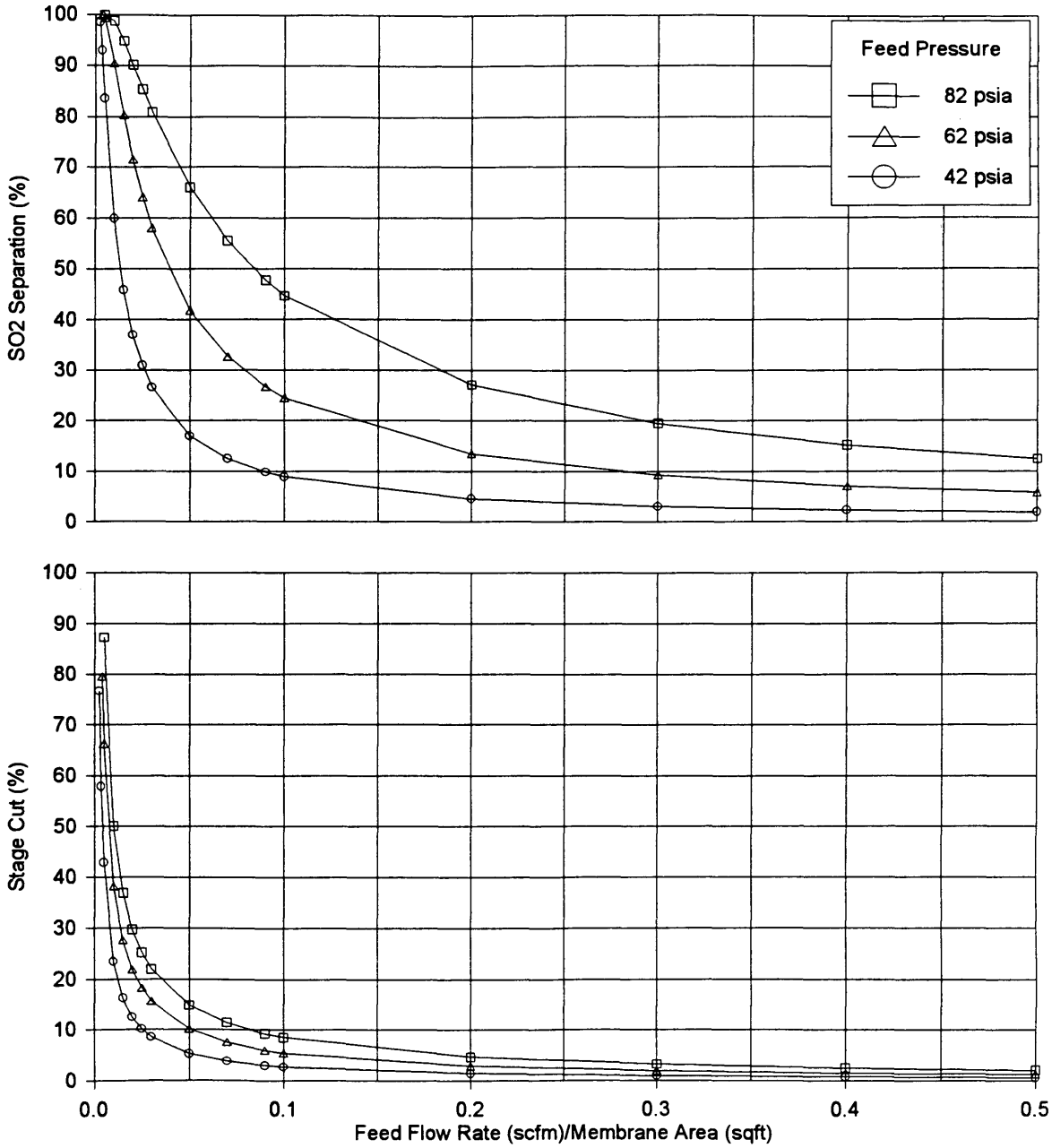


Figure 6-7

Countercurrent Flow Pattern, PEBAX Membrane Performance Chart for Low SO_2 and NO_2 Pollutant Concentrations. (Permeate pressure = 12 psia, Membrane Thickness = unknown, Mole Fractions of Other gases: N_2 = 0.8, O_2≈0.08, and CO_2=0.12)

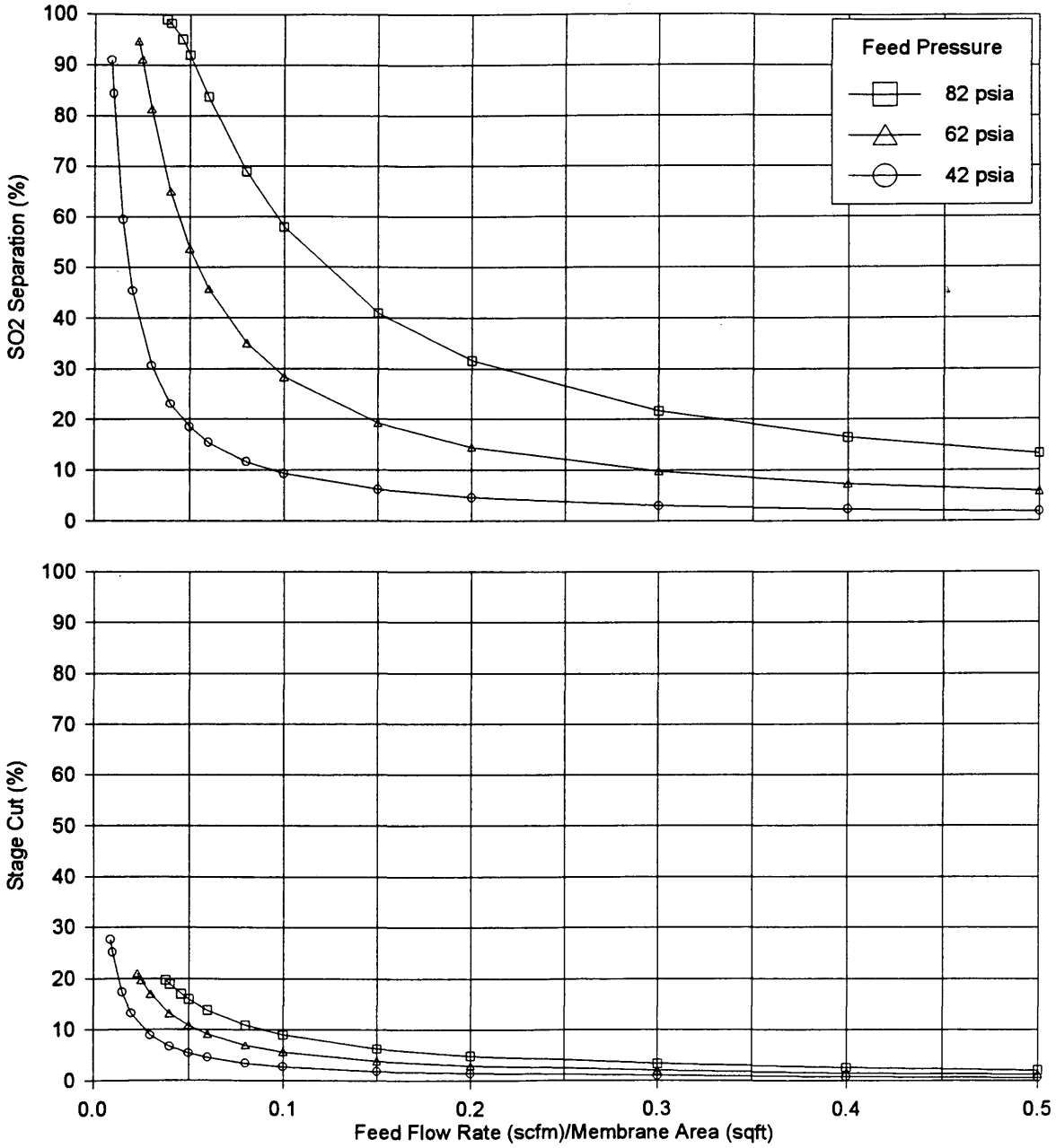


Table 6-1

Comparison of Single-stage Flow Pattern Separations and Membrane Material at 62 psia and 0.05 scfm per sqft

Single-Stage Flow Pattern	Material	SO_2 % Separation	NO_2 % Separation	Stage Cut %
Cocurrent	PDMS	47	45	16
Crosscurrent	PDMS	54	46	16
Countercurrent	PDMS	65	58	16
Cocurrent	PEBAX	35	-	10
Crosscurrent	PEBAX	42	-	10
Countercurrent	PEBAX	54	-	10

Figure 6-8

Driving Force Across a PEBAX Membrane for 62 psia Feed Pressure and 0.05 scfm/sqft.

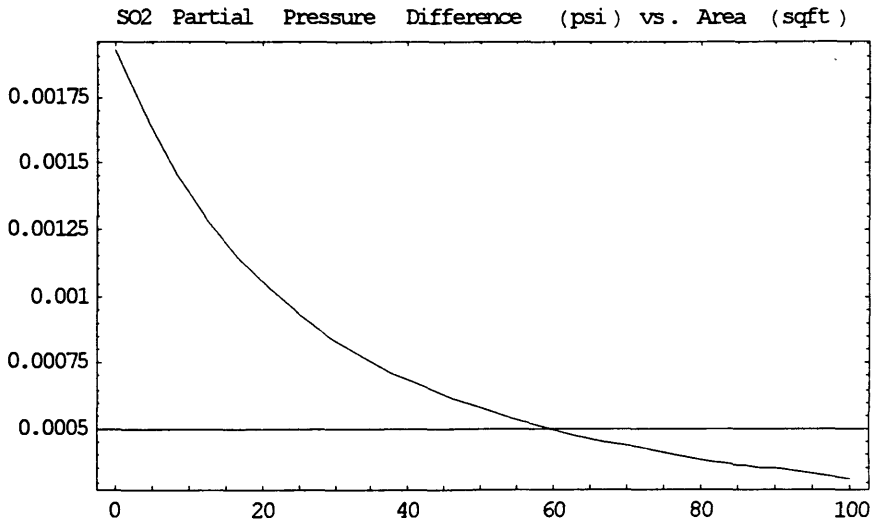
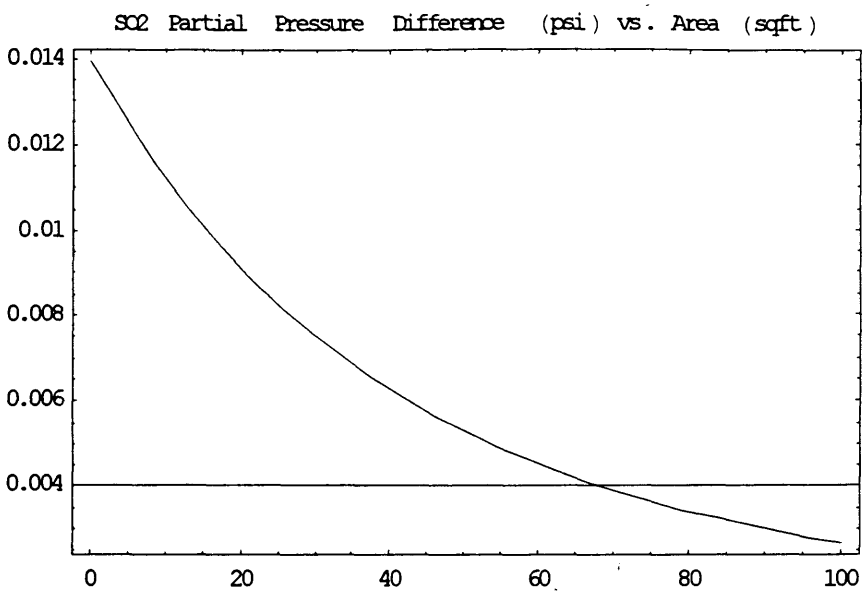


Figure 6-9

Driving Force Across a PDMS Membrane for 62 psia Feed Pressure and 0.05 scfm/sqft.



6.2 Increasing Membrane Efficiency, Increasing the Driving Force

Single-stage gas separation membranes do not provide the most efficient means for separating SO_2 or NO_2. Several researchers have investigated methods for increasing the driving force of SO_2 and/or NO_x by sweeping the permeate side of the membrane with a liquid solution [50-55] or steam [3]. This approach is effectively making the permeate pollutant partial pressure much less than for a gas separation. Similarly, if the permeate pressure is set to zero the separation limitations are observed for each membrane material, and each membrane flow pattern produces the same result.

Figures 6-10 and 6-11 show performance charts for PDMS and PEBAX with the permeate pressure set close to zero. As one can see, this greatly increases the separation performance of each membrane material. PEBAX, with a higher SO_2 permeance than PDMS, separates 90 percent SO_2 at 62 psia for a feed flow rate to membrane area ratio of 1.0 scfm/sqft.

Figure 6-10

Maximized Driving Force, PDMS Membrane Performance Chart for Low SO_2 and NO_2 Pollutant Concentrations. (Permeate pressure = 12 psia, Membrane Thickness = 0.000984 in., Mole Fractions of Other gases: N_2 = 0.8, O_2≈0.08, and CO_2=0.12)

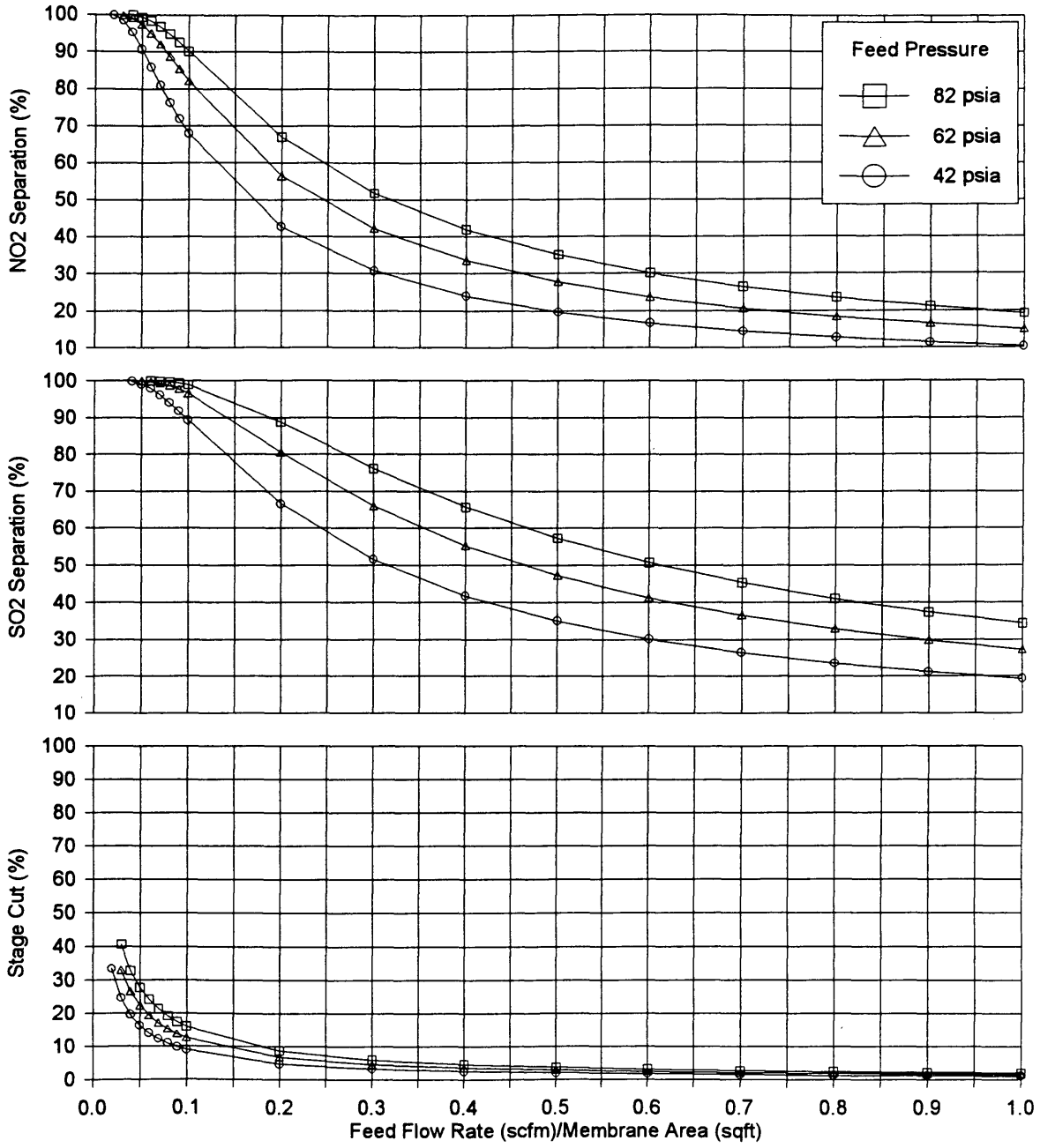
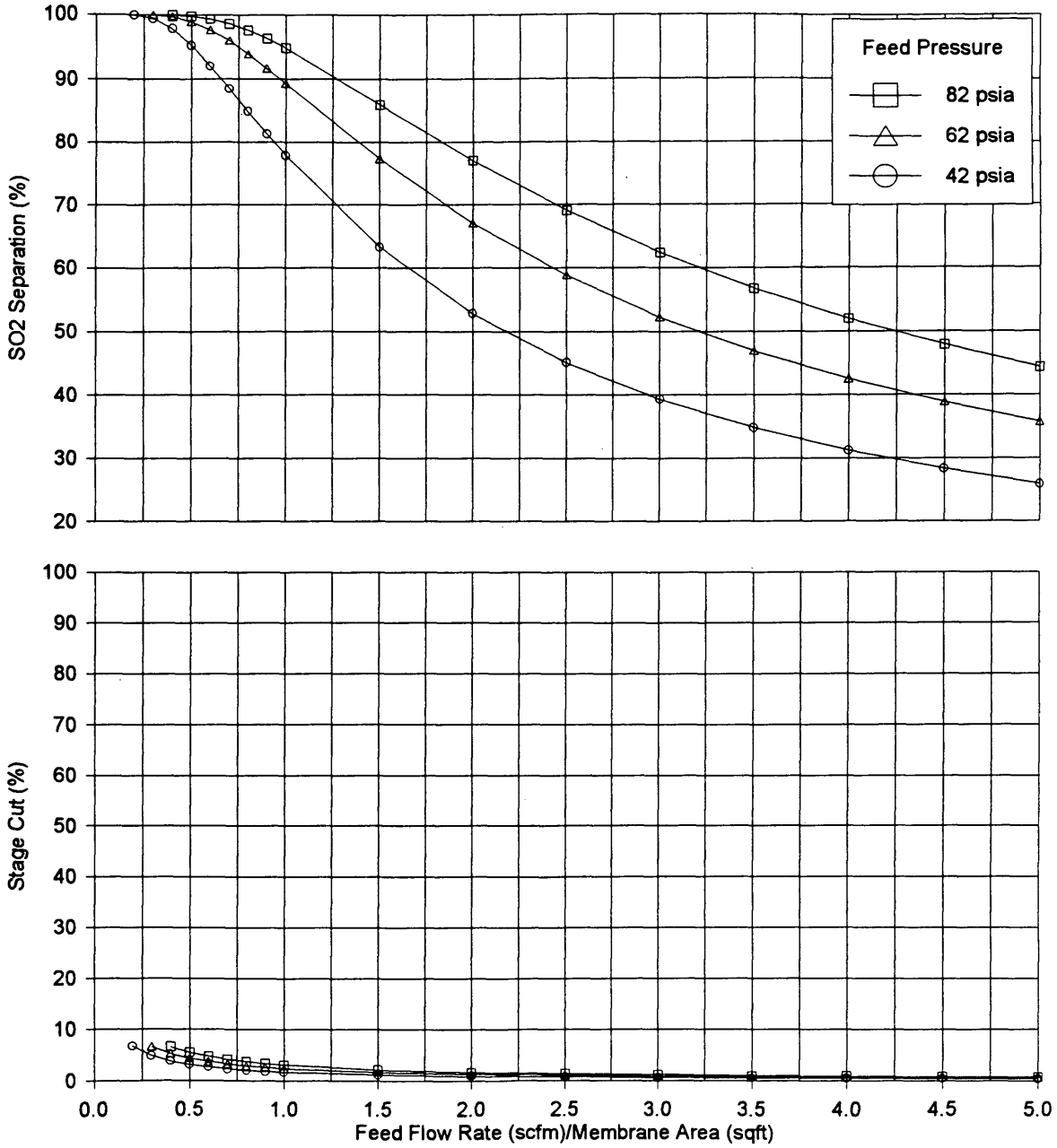


Figure 6-11

Maximized Driving Force, PEBAX Membrane Performance Chart for Low SO_2 and NO_2 Pollutant Concentrations. (Permeate pressure = 12 psia, Membrane Thickness = unknown, Mole Fractions of Other gases: N_2 = 0.8, O_2≈0.08, and CO_2=0.12)



6.3 Decreasing Stage Cut, The Three-Stage Configuration

Spillman [20, 21] and Bhide [18, 19] showed how a three-stage membrane configuration can reduce the cost of separating CO_2 from natural gas. Specifically, they show that a three-stage system (discussed in Section 2.4) reduces the amount methane lost to the permeate.

Since this is not an economic analysis, this section is meant to demonstrate the benefits of the three-stage system and the quantity of data involved.

PDMS and PEBAX membranes are evaluated for a cocurrent flow pattern with a 62 psia feed pressure. The feed flow rate to total membrane area ratio is held at 0.05 scfm/sqft (5scfm/100sqft), but since the total membrane area is divided into three-stages, a number of calculations are made to completely evaluate this system. Figures 6-12 and 6-13 and Figures 6-15 and 6-16 provide two different views of the same data. Although these figures are difficult to accurately read, they show the general trends in stage cut, SO_2 separation, and NO_2 separation over a range of membrane area combinations. These figures are similar to the performance charts in Section 6-1, but they show a surface plot for one feed pressure, 62 psia, and one feed flow rate, 5scfm. In each plot, *area1* represents the pre-membrane stage (see Figure 2-4), *area2* represents the purification stage, and *area3* represents the recovery stage. *Area3* is defined since the total area is bound to 100 sqft.

Figures 6-14 and 6-17 were made to compare the three-stage and single-stage configurations. Each figure shows horizontal lines, which represent the percentage of pollutant separation and stage cut for the single-stage configuration. The curved lines and jagged lines represent the numerous results for the three-stage configuration. The three-stage results were sorted in descending order of percentage of SO_2 separated. Each figure shows the percentage of pollutants separated is less than the single-stage configuration. The stage cut, however, is also less, but in some cases, by a wider margin. (This trade-off in stage cut to percentage separation yields the economic savings in the CO_2/natural gas separation.) The same qualitative observation is observed over a range of feed flow rate to membrane area ratios, feed pressures, and different flow patterns.

The data presented in this section is qualitative. The real advantage of the three-stage configuration is the trade-off of percentage separation for a lesser stage cut. This trade-off can only be objectively evaluated in an economic analysis.

Figure 6-12

View 1, Three-Stage Separation Performance Chart for PDMS for Low SO_2 and NO_2 Pollutant Concentrations (62 psia feed pressure, 12 psia permeate pressure, 5 scfm feed flow rate, 100 sqft total membrane area, 0.000984 in membrane thickness, and the following mole fractions: N_2=0.8, O_2≈0.08, and CO_2=0.12)

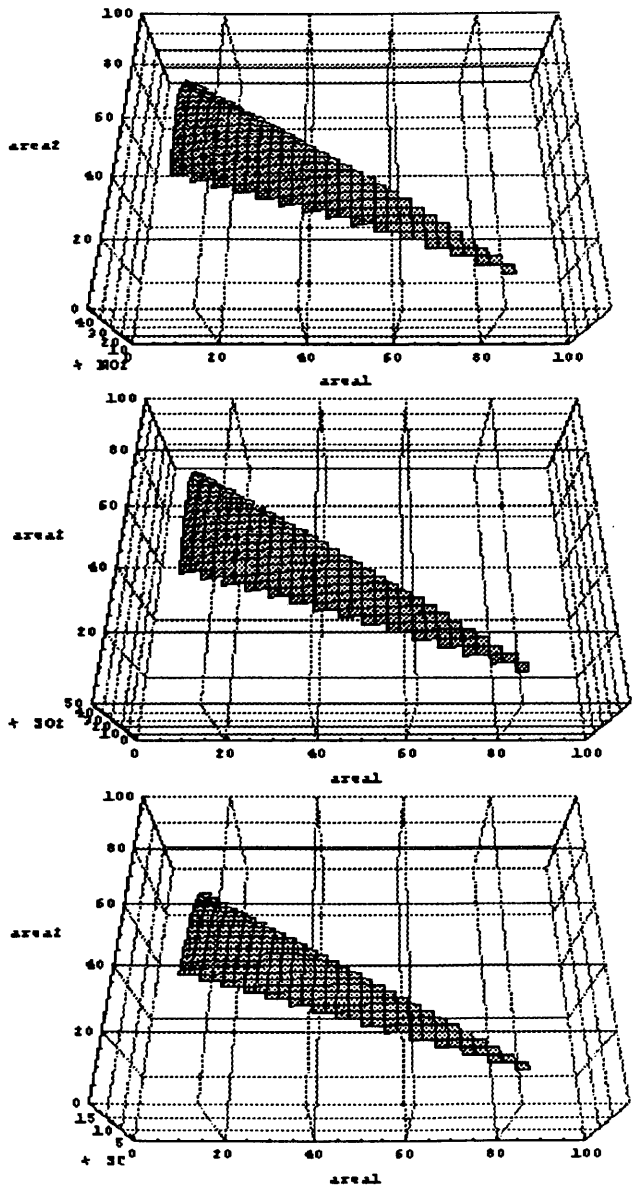


Figure 6-13

View 2, Three-Stage Separation Performance Chart for PDMS for Low SO_2 and NO_2 Pollutant Concentrations (62 psia feed pressure, 12 psia permeate pressure, 5 scfm feed flow rate, 100 sqft total membrane area, 0.000984 in membrane thickness, and the following mole fractions: N_2=0.8, O_2≈0.08, and CO_2=0.12)

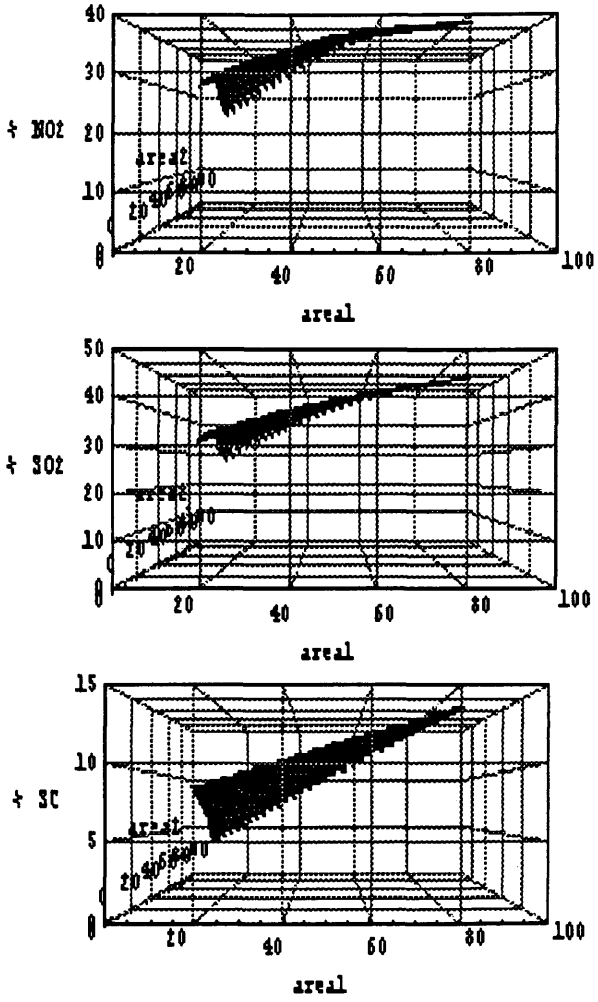


Figure 6-14

Comparison of PDMS Single-Stage Results with Three-Stage Results

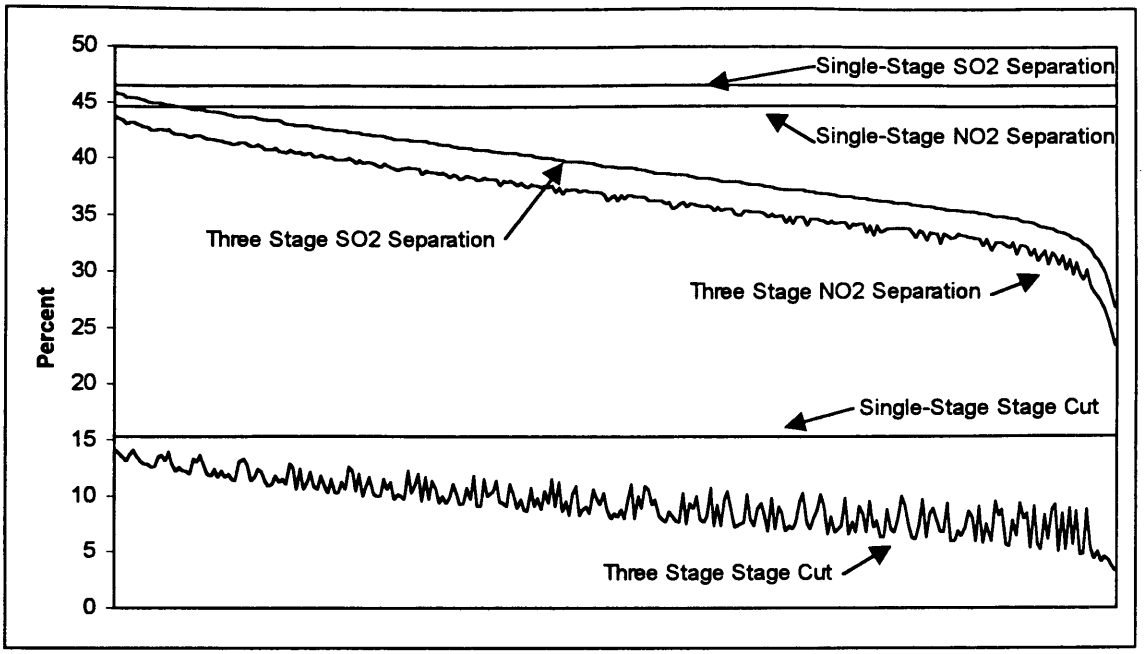


Figure 6-15

View 1, Three-Stage Separation Performance Chart for PEBAX for Low SO_2 Pollutant Concentrations (62 psia feed pressure, 12 psia permeate pressure, 5 scfm feed flow rate, 100 sqft total membrane area, and the following mole fractions: N_2=0.8, O_2≈0.08, and CO_2=0.12)

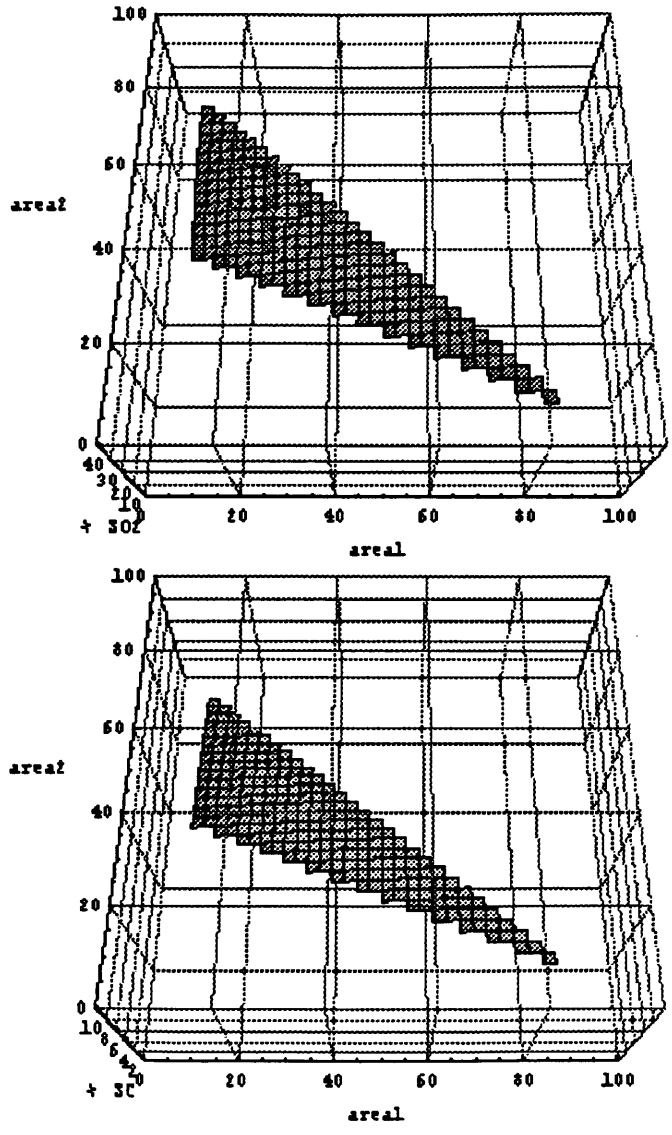


Figure 6-16

View 2, Three-Stage Separation Performance Chart for PEBAX for Low SO_2 Pollutant Concentrations (62 psia feed pressure, 12 psia permeate pressure, 5 scfm feed flow rate, 100 sqft total membrane area, and the following mole fractions: N_2=0.8, O_2≈0.08, and CO_2=0.12)

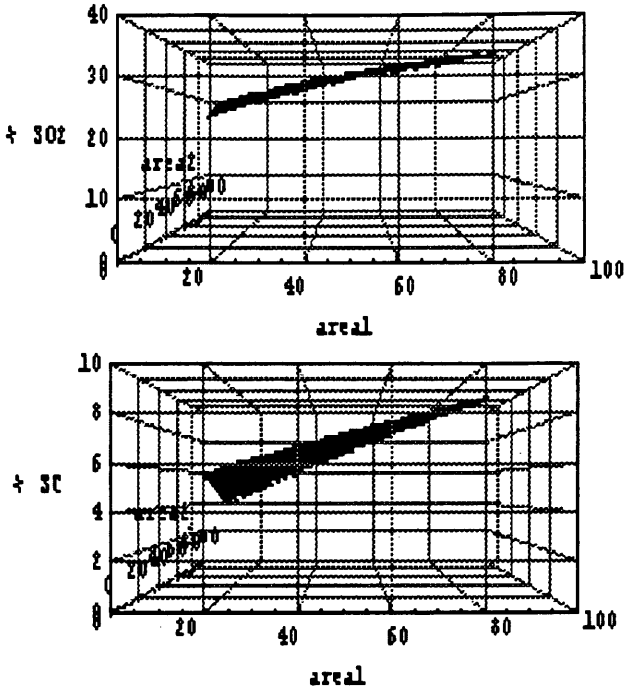
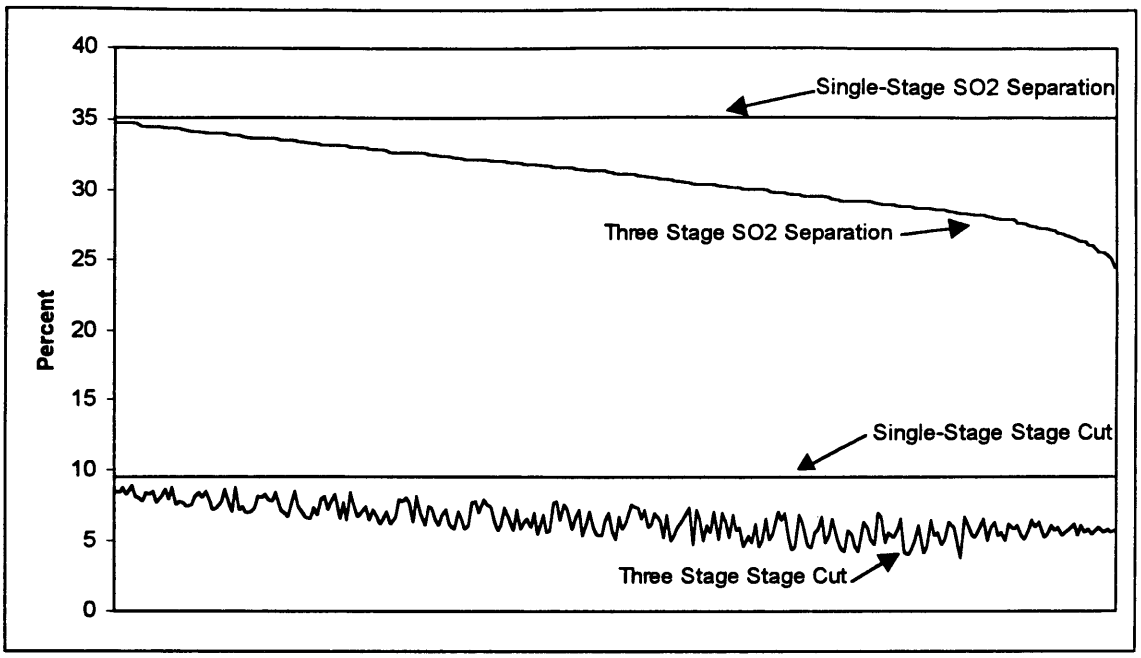


Figure 6-17

Comparison of PEBAX Single-Stage Results with Three-Stage Results



Chapter 7

CONCLUSIONS AND RECOMMENDATIONS

One of the objectives of this work was to develop computer models to simulate the single-stage membrane separation of a multi-component gaseous mixture. Upon applying these computer models to the separation of SO_2 and NO_2 from flue gas in an existing pilot-plant, a relatively low feed flow rate to membrane area ratio is required for a 90 percent separation of pollutants. The use of PEBAX rather than PDMS does not improve the separation percentage under the system constraints of the pilot-plant, but the stage cut is greatly reduced. A three-stage configuration, recommended in the literature for a similar separation, will not improve the feed flow rate to membrane area ratio, but it may reduce costs by decreasing the total stage cut. Enhancing the separation by decreasing the partial pressure of SO_2 and NO_2 in the permeate stream greatly improves separation.

PDMS is not selective toward NO, a major component in NO_x. Any unenhanced polymer membrane system is unlikely to separate NO, NO_2 and SO_2. A separation of NO_2 and SO_2, however, is currently possible with polymer membranes.

The following recommendations are made.

1. Only an economic analysis can determine the benefits of the hybrid treatment process concept. An economic analysis can also determine the benefits of using different membrane system configurations. Gather data concerning the costs of treating SO_2 and/or NO_2 as a function of concentration and flow rate, and perform economic analysis.
2. The results of this work are based on the system constraints of the pilot-plant. Investigate separation conditions for the countercurrent flow pattern at high feed pressures and low permeate pressures, and determine the feasibility of obtaining these pressures in practice.
3. The results of this work are based on the availability of published permeability data at specified temperatures. Use the bench scale apparatus discussed in Chapter 4 to confirm published results, and obtain permeability data over a range of temperatures.

4. The computer models have been confirmed by the results given in only one publication [38]. Operate the pilot-plant in cocurrent and countercurrent flow patterns to further confirm the validity of the models.
5. Investigate the practicality of using a sweep fluid that increases the SO_2 and/or NO_2 driving force.
6. The computer models written for this work are applicable to any multi-component membrane separation. During the early stages of this research, efforts were made to optimize the SO_2 and NO_2 separation for single-stage membrane systems. The optimization criteria, not based on economics, were rather ambiguous, and these efforts were abandoned. The material presented in Appendix I, J, and K may be useful in developing a comprehensive optimization scheme for any proposed membrane separation.

REFERENCES SITED

1. Walker, R.J., C.J. Drummond, and J.M. Ekmann, *Evaluation of Advanced Separation Techniques for Application to Flue Gas Cleanup Processes for the Simultaneous Removal of Sulfur Dioxide and Nitrogen Oxides*, . 1985, Pittsburgh Energy Technology Center.
2. Cooper, C.D. and F.C. Alley, *Air Pollution Control: A Design Approach*. 1st ed. 1986, Prospect Heights, IL: Waveland Press, Inc. 630.
3. Baker, R.W., *et al.*, *Development of a Membrane SOx/NOx System*, . 1988.
4. Li, J., *et al.*, *Selective Permeation of Sour Gases Through Polymetric Membranes Modified by Sulfolanes. II. Study of Selective Permeation of SO2 Through Polytrimethylsilyl Methyl Methacrylate and its Blend Membranes Containing 3-Methylsulfolane.* Journal of Applied Polymer Science, 1996. **61**: p. 2467-2470.
5. Peterson, E.S., *et al.*, *Mixed Gas Separation Properties of Phosphazene Polymer Membranes.* Separation Science and Technology, 1993. **28**(1-3): p. 423-440.
6. Kuehne, D.L. and S.K. Friedlander, *Selective Transport of Sulfur Dioxide through Polymer Membranes. 1. Polyacrylate and Cellulose Triacetate Single-Layer Membranes.* Ind. Eng. Chem. Process Des. Dev., 1980. **19**: p. 609-616.
7. Dytnerskii, Y.I., *et al.*, *SO2 Separation from Gaseous Mixtures by Membranes.* Journal of Membrane Science, 1989. **41**: p. 49-54.
8. Knecht, R.D., *Membrane-Based Mini-Pilot Plant Facility to Treat Off-Gas from the Microwave Solidification Process*, . 1996: Golden, CO.
9. Mulder, M., *Basic Principles of Membrane Technology*. 1991: Kluwer Academic Publishers. 363.
10. *CRC Handbook of Chemistry and Physics*. 70 ed, ed. R.C. Weast and D.R. Lide. 1989, Boca Raton, FL: CRC Press.
11. Imai, K., *et al.*, *Synthesis of Sulfoxide-Modified Cellulose Derivatives Designed for the Permselective Membrane of Sulfur Dioxide.* Bull. Chem. Soc. Japan, 1987. **60**: p. 753-757.
12. Imai, K., *et al.*, *Separability of So2 from SO2/N2 Mixtures through Suloxide-Modified Poly(Vinyl Alcohol) and Cellulose Membranes.* Journal of Applied Polymer Science, 1993. **48**: p. 1525-1529.

13. Zavaleta, R. and F.P. McCandless, *Selective Permeation Through Modified Polyvinylidene Fluoride Membranes.* Journal of Membrane Science, 1976. **1**: p. 333-353.

14. Seibel, D.R. and F.P. McCandless, *Separation of Sulphur Dioxide and Nitrogen by Permeation through a Sulfolane Plasticized Vinylidene Fluoride Film.* Ind. Eng. Chem. Process Des. Develop., 1974. **13**(1): p. 76-78.

15. Felder, R.M., R.D. Spence, and J.K. Ferrel, *Permeation of Sulfur Dioxide Through Polymers.* Journal of Chemical and Engineering Data, 1975. **20**(3): p. 235-242.

16. Frimpong, S., C.A. Plank, and W.L.S. Laukhuf, *Sorption and Transport of Sulfur Dioxide in Polycarbonate-Polyacrylate Blends.* The Chemical Engineering Journal, 1991. **47**: p. 63-73.

17. Products, M., *General Electric Permselective Membranes*, , G. Electric, Editor, General Electric: Schenectady, NY.

18. Bhide, B.D. and S.A. Stern, *Membrane Processes for the Removal of Acid Gases from Natural Gas. I. Process Configurations and Optimization of Operating Conditions.* Journal of Membrane Science, 1993. **81**: p. 209-237.

19. Bhide, B.D. and S.A. Stern, *Membrane Processes for the Removal of Acid Gases from Natural Gas. II. Effects of Operating Conditions, Economic Parameters, and Membrane Properties.* Journal of Membrane Science, 1993. **81**: p. 239-252.

20. Spillman, R.S., M.G. Barrett, and T.E. Cooley. *Gas Membrane Process Optimization.* in *AIChE National Spring Meeting*. 1988. New Orleans, LA: AIChE.

21. Spillman, R.W., *Economics of Gas Separation Membranes.* Chemical Engineering Progress, 1989. **85**(1): p. 41-62.

22. Strom, P., *Flue Sas Scrubbing: An Experimental and Pilot Plant Study*, in *Department of Chemical Engineering*. 1996, Colorado School of Mines: Golden, CO. p. 168.

23. Knecht, R.D., *Evaluating a Membrane-Based Process for Treating the Off-Gas from the Microwave Solidification System*, . 1995: Golden, CO.

24. McMannous, T., *Personal Communication About PDMS membrane*, . 1996, Mempro.

25. Davis, E.G. and M.L. Rooney, *Transport of Sulfur Dioxide in Polymers.* Kolloid-Zeitschrift & Zeitschrift Fur Polymere, 1971. **249**(1-2): p. 1043-1050.

26. Perry, R.H. and D. Green, eds. *Perry's Chemical Engineering Handbook*. 6th ed. . 1984, McGraw-Hill Inc.

27. Box, G.E.P., W.G. Hunter, and J.S. Hunter, *Statistics for Experimenters*. Wiley series of probability and mathematical statistics. 1978, New York, NY: John Wiley and Sons.

28. Weller, S. and W.A. Steiner, *Separation of Gases by Fractional Permeation through Membranes.* Journal of Applied Physics, 1950. **21**: p. 279.

29. Weller, S. and W.A. Steiner, *Engineering Aspects of Separation of Gases: Fractional Permeation through Membranes.* Chemical Engineering Progress, 1950. **46**: p. 585.

30. Huckins, H.E. and K. Kammermeyer, *The Separation of Gases by means of Porous Membranes.* Chemical Engineering Progress, 1953. **49**(6): p. 294-298.

31. Brubaker, D.W. and K. Kammermeyer, *Separation of Gases by Plastic Membranes.* Industrial and Engineering Chemistry, 1954. **46**(4): p. 733-739.

32. Walawender, W.P. and S.A. Stern, *Analysis of Membrane Separation Parameters. II: Countercurrent and Cocurrent Flow in a Single Permeation Process.* Separation Science, 1972. 7: p. 553.

33. Blaisdell, C.T. and K. Kammermeyer, *Counter-current and Co-current Gas Separation.* Chemical Engineering Science, 1973. **28**: p. 1249.

34. Antonsen, C.R., *et al.*, *Analysis of Gas Separation by Permeation in Hollow Fibers.* Ind. Eng. Chem. Process Des. Div., 1977. **16**: p. 463.

35. Pan, C.Y. and H.W. Habgood, *Gas Separation by Permeation: Part I. Calculation Methods and Parametric Analysis.* The Canadian Journal of Chemical Engineering, 1978. **56**(April): p. 197-209.

36. Hogsett, J.E. and W.H. Mazur, *Estimate Membrane System Area.* Hydrocarbon Processing, 1983(August): p. 52-54.

37. Boucif, N., S. Majumdar, and K. Sirkar, *Series Solutions for a Gas Permeator with Countercurrent and Cocurrent Flow.* Ind. Eng. Chem. Fundam., 1984. **23**: p. 470-480.

38. Shindo, Y., *et al.*, *Calculation Methods for Multicomponent Gas Separation by Permeation.* Separation Science and Technology, 1985. **20**(5,6): p. 445-259.

39. Saltonstall, C.W., *Calculation of the Membrane Area Required for Gas Separations.* Journal of Membrane Science, 1987. **32**: p. 185-193.

40. Chern, R.T., W.J. Koros, and P.S. Fedkiw, *Simulation of a Hollow-Fiber Gas Separator: The Effects of Process and Design Variables.* Ind. Eng. Chem. Process Des. Dev., 1985. **24**: p. 1015-1022.

41. Rautenbach, R. and W. Dahm, *Gas Permeation-Module Design and Arrangement.* Chemical Engineering Progress, 1987. **21**: p. 141.

42. Li, K., D.R. Acharya, and R. Hughes, *Mathematica Modelling of Multicomponent Membrane Permeators.* Journal of Membrane Science, 1990. **52**: p. 205-219.

43. Giglia, S., *et al.*, *Mathematical and Experimental Analysis of Gas Separation by Hollow Fiber Membranes.* Ind. Eng. Chem. Res., 1991. **30**: p. 1239-1248.

44. Kataoka, T., *et al.*, *Permeation Equations Developed for Prediction of Membrane Performance in Pervaporation, Vapor Permeation, and Reverse Osmosis Based on the Solution-Diffusion Model.* Journal of Chemical Engineering of Japan, 1991. **24**(3): p. 326-333.

45. Fattah, K.A., *et al.*, *A Nonideal Model for Analysis of Gas Sepatation Permeators.* Journal of Membrane Science, 1992. **65**: p. 247-257.

46. Pettersen, T. and K.M. Lien, *A New Robust Design Model for Gas Separating Membrane Modules, Based on Analogy with Counter-Current Heat Exchangers.* Computers Chem. Eng., 1994. **18**(5): p. 427-439.

47. Thundyil, M.J. and W.J. Koros, *Mathematical Modeling of Gas Separation Permeators - for Radial Crossflow, Countercurrent, and Cocurrent Hollow Fiber Membrane Modules.* Journal of Membrane Science, 1997. **125**: p. 2756-291.

48. Wolfram Research, I., *Mathematica 3.0*, . 1996: Champaign, IL.

49. Wolfram, S., *The Mathematica Book.* 3 ed. 1996: Wolfram Media Cambridge University Press. 1403.

50. *Membrane System Blocks SOx & NOx Emissions.* Chemical Engineering, 1996. **August**: p. 21.

51. Nii, S. and H. Takeuchi. *Gas Absorption with Membrane Permeation - Acid Gas Removal from Flue Gases by Permabsorption Method.* in *ICHEME Symposium Series No. 128.*

52. Nii, S. and H. Takeuchi, *Removal of CO2 and/or SO2 from Gas Streams by a Membrane Absorption Method.* Gas Separation and Purification, 1994. **8**(2): p. 107-114.

53. Chakma, A., *Separation of CO2 and SO2 from Flue Gas Streams by Liquid Membranes.* Energy Conversion and Management, 1995. **36**(6-9): p. 405-410.

54. Pakala, N.R., S. Varanasi, and S.E. LeBlanc, *Citrate-Based Contained Liquid Membranes for Flue Gas Desulfurization.* Ind. Eng. Chem. Res., 1993. **32**: p. 553-563.

55. Majumdar, S., *et al.*, *Simultaneous SO2/NO Separation from Flue Gas in a Contained Liquid Membrane Permeator.* Ind. Eng. Chem. Res., 1994. **33**: p. 667-675.

56. Kim, S.J., B.R. Min, and T.H. Lee, *A Study on Separation of N2 - SO2 Mixed Gas by Polymer Membranes.* Membrane Journal, 1992. **2**(2): p. 135-143.

57. Davis, E.G. and M.L. Rooney, *Transport of Sulfur Dioxide in Polymers.* Kolloid-Zeitschrift & Zeitschrift Fur Polymere, 1971. **249**: p. 1043-1050.

58. Pfromm, P.H. and W.J. Koros, *Sorption and Transport of Sulfur Dioxide in Polysulphone.* Macromolecules, 1993. **26**: p. 6141-6142.

59. Kuehne, D.L. and S.K. Friedlander, *Selective Transport of Sulfur Dioxide through Polymer Membranes. 2. Cellulose Triacetate/Polyacrylate Composite Membranes.* Ind. Eng. Chem. Process Des. Dev., 1980. **19**: p. 616-623.

60. Peters, M.S. and K.D. Timmerhaus, *Plant Design and Economics for Chemical Engineers.* 4th ed. 1991, New York: McGraw-Hill. 910.

61. Walters, F., *et al.*, *Sequential Simplex Optimization.* 1991, Boca Raton, FL: CRC Press.

APPENDIX

APPENDIX A – Polymers for SO_2 Separations

Table A-1

Polymers for SO_2 Separations

Polymer	Permeability (Barrers)				Selectivity SO2/N2	Testing Conditions
	SO2	N2	O2	CO2		
Polycarbonate K[7]	543.6	32.3	183.7		16.9	75%SO2 and 25% air, 22°C, 0.13 MPa gauge
TCP-T[7] Triazine-containing copolymer	2240	31.36	71.7		71.4	75%SO2 and 25% air, 22°C, 0.13 MPa gauge
Seragel-70[7]	6272	7.77	29.6		808	75%SO2 and 25% air, 22°C, 0.13 MPa gauge
Seragel-120[7]	4928	8.36	32.85		589	75%SO2 and 25% air, 22°C, 0.13 MPa gauge
PVTMS [7] Polyvinyltrimethyl silane	120.4	11.05	44.2		10.9	75%SO2 and 25% air, 22°C, 0.13 MPa gauge
Silar [7] Siloxane arylate	2635.8	105.4	221.31		25	75%SO2 and 25% air, 22°C, 0.13 MPa gauge
CA[56] Cellulose Acetate	3351	98.6			34	Mixed Gas (unknown conc.), 20°C, 0.15MPa gauge
TFC[56] Polysulfone thin film composite	5528	95.6			58	Mixed Gas (unknown conc.), 20°C, 0.15MPa gauge
FT-30[56] Polysulfone-aromatic polyamid copolymer	7314	65.7			112	Mixed Gas (unknown conc.), 20°C, 0.15MPa gauge
PE[57] Polyethylene	2090		309			Pure gas, 25°C
Nylon II[57] Polyamide	658		14			Pure gas, 25°C
PC[57] Polycarbonate	2240		154			Pure gas, 25°C
Polysulfone[58]	20.8		0.25		83.3	Pure gas, 35°C, 1 atm gauge

Polyvinylidine Fluoride With 18wt. % sulfolene[13]	250	0.09		3	2778	Pure gas, 25°C, SO2 at 30psig, N2 and CO2 at 100psig
Buna N Nylon[6]	800	1.8		39.7	444	Pure gas, 25°C, 1 atm gauge
PA[6] Polyacrylate on Dacron	1920	2.04		46.7	941	Pure gas, 25°C, 1 atm gauge
PA with 25% polyethylene glycol[6]	3210	3.22		113	997	Pure gas, 25°C, 1 atm gauge
PA with 50% polyethylene glycol[6]	5160	4.68		128	110	Pure gas, 25°C, 1 atm gauge
PA with 25% sulfolane[6]	2150	3.65		107	589	Pure gas, 25°C, 1 atm gauge
CA[6] Cellulose Acetate	270	0.2		5.6	1350	Pure gas, 25°C, 1 atm gauge
Cellulose acetate-butyrate[6]	720	1.5		39.2	480	Pure gas, 25°C, 1 atm gauge
CTA/PA [59] Cellulose triacetate/Polyacrylate Composite	311	2.96		42.2	105	Pure gas, 25°C, 1 atm gauge
PVA [12] Polyvinyl alcohol with CH3 substitution and 22.4 mol% sulfoxide	18.2	0.34			54	Mixed Gas (40%SO2 and 60%N2), 23°C, 1.5 atm gauge
PVA [12] Polyvinyl alcohol with C6H5 substitution and 11.5 mol% sulfoxide	16.5	0.21			78	Mixed Gas (40%SO2 and 60%N2), 23°C, 1.5 atm gauge
PVA [12] Polyvinyl alcohol with C6H5 substitution and 23.5 mol% sulfoxide	47	0.28			170	Mixed Gas (40%SO2 and 60%N2), 23°C, 1.5 atm gauge
PPOP[5] Poly[bis(phenoxy)phosphazene]	55, 177	0.17, 3.2			320,56	Mixed Gas (10%SO2 and 90% N2), (30°C, 80°C), 20psig
SO3-PPOP[5] Poly[bis(sulfoxidophenoxy)phosphazene]	42, 146	0.21, 2.2			200,65	Mixed Gas (10%SO2 and 90% N2), (30°C, 80°C), 20psig
m-F-PPOP[5] Poly[bis(metafluorophenoxy)phosphazene]	104, 348	0.28, 6.7			370,52	Mixed Gas (10%SO2 and 90% N2), (30°C, 80°C), 20psig

CH3-PPOP[5] Poly[bis(methyl-phenoxy)phosphazene]	163, 516	0.54, 51.6			300,10	Mixed Gas (10%SO2 and 90% N2), (30°C, 80°C), 20psig
PTMSMMA[4] Poly[trimethylsilylmethylmeth acrylate]	237	9.4		135	25.2	Pure Gas, 30°C, 40cmHg guage feed and vacuum permeate
PTMSMMA[4] Poly[trimethylsilylmethylmeth acrylate] with 16% 3-methylsulfolane additive	411	7.8		122	52.5	Pure Gas, 30°C, 40cmHg guage feed and vacuum permeate
Ethylcellulose[3]	5100000	150000			35	Permeance (thickness not given), Mixed Gas (0.3%SO2, 15%CO2, 84.7%N2), Room Temperature, 85-90cmHg feed and 1.5-1.9 cmHg permeate
Styrene copolymer (unknown copolymer)[3]	1200000	28000			43	Permeance (thickness not given), Mixed Gas (0.3%SO2, 15%CO2, 84.7%N2), Room Temperature, 85-90cmHg feed and 1.5-1.9 cmHg permeate
Polymethylpentene[3]	290000	19000			15	Permeance (thickness not given), Mixed Gas (0.3%SO2, 15%CO2, 84.7%N2), Room Temperature, 85-90cmHg feed and 1.5-1.9 cmHg permeate
PEBAX[3] Polyether block polyamide (refered to in article as Polyamide Copolymer grade 3)	3.5E(7), 3.5E(7), 6.9E(7)	5E(4), 9.2E(4), 2.8E(5)	1.1E(5), 2.1E(5), 6.2E(5)	2.8E(6), 3.5E(6), 6.7E(6)	700, 332, 250	Permeance (thickness not given), Mixed Gas (0.4%SO2, 10%CO2, 7%O2, 82.6%N2), (23°C, 37°C, 61°C), 2.5psig feed and 2.2 cmHg permeate
PDMS[17] Poly(dimethylsiloxane)	12500	250	500	2700	50	Pure Gas, Room Temperature

APPENDIX B – Co5

(*Brian Nelson*)

(*Single stage cocurrent membrane system for the multi-component separation of 5 gases*)

(*Based on method of Shindo, et al ['85] without dimensionless variables*)

(*INSTRUCTIONS*)

(*

1. This program is written as a function in Mathematica.
   In order to operate the function "co5[{}]", you must first click the cursor into the cell below (cells are designated by the brackets to the right.)---------->

2. Press Shift-Enter to enter the function into
   Mathematica memory.

3. The function is designed to place some or all of the
   following variables inside the function brackets [{}].
   The function is currently set up to input the following five values in the given order.

   a. feed pressure (psia)
   b. feed side pressure drop (psi)
   c. permeate pressure (psia)
   d. feed flow rate (scfm)
   e. membrane area (sq ft)

If you want the function to include more variables than listed directly above, simply remove the variable from PART 1 - INPUT CONSTANTS AND VARIABLES and put the variable, with the identical name, into the function designation (explained below) followed by an underscore, _. Other values, which can be

included in the function, are listed below. Note: When running the function you must always input the variable values of a function in the same order as the function is written. Also, if you make a change to the function you must re-enter it into the current memory by pressing Shift-Enter while the cursor is in the cell containing the function.

f. membrane thickness (inches)
g. species 1 permeability (barrer)
h. species 2 permeability (barrer)
i. species 3 permeability (barrer)
j. species 4 permeability (barrer)
k. species 5 permeability (barrer)
l. species 1 feed mole fraction
m. species 2 feed mole fraction
n. species 3 feed mole fraction
o. species 4 feed mole fraction
p. species 5 feed mole fraction

4. Definition of the following variables (Note: "..." stands for the species number 1, 2, 3, etc.):

a. permeability... = a permeability in barrers.
b. pfeed and pperm = feed and permeate pressures in psia.
c. pdrop = pressure drop on the feed side in psi
d. x...i = an initial feed mole fraction
e. area = variable for membrane area.
f. d = membrane thickness in inches.
g. feedflow = feed flow in scfm
h. perm... = a permeability in ft^3 (stp)*in/min/ft^2/delta psi.
i. gamma1 = a pressure ratio of permeate pressure to the initial feed pressure.
j. gamma2[area] = pressure ratio of permeate pressure to feed pressure as it changes.

k. totalarea = the total membrane area in sqft.

l. sol... = a solution or solution set

m. yi is explained in Part 3 below.

n. y...i = initial permeate mole fractions

o. x...[area] = a mole fraction at some point along the membrane

p. eq... = equation number...

q. f[area] = retentate flow rate at some point along the membrane.

r. ...int = integrated solution set, an InterpolatingFunction, of some value. note: you can find a value by entering ...int[area].

s. g = permeate flow rate in scfm.

t. ...perm = permeate mole fractions after exposure to the entire membrane surface.

u. species... = identification of gaseous species.

5. Keeping the window with the entered "co5[{}]" function open, enter into a new window (by clicking File-New from the pull-down menu) the function with the desired variables, as given in the following example or as the function has been changed, and press Shift-Enter.

Example: co5[{65,0,12,50,100}]

*)

(*A function is created in Matiematica when Mathematica sees the following form:

function[variable1_,variable2_, etc...] := (task)

The function name can be anything continuous and should not start with a capital letter (so they are not confused with Mathematica functions). Variables are enclosed in brackets, [], and immediately followed by an underscore, _. Please note, in the following function the variables are written in a list inside the brackets. A list is denoted by curvy brackets {}. The := notation tells Mathematica to delay the calculation until the function is called upon. The task or set of equations or procedures is enclosed in parentheses (). Each line of code within the parentheses must end with a semicolon. Mathematica will not stop reading a function until all parentheses are closed. *)

```
co5[{pfeed_,pdrop_,pperm_,feedflow_,totalarea_}] := (

(*Make sure other functions do not affect the current calculation*)
Clear[area,eq1,eq2,eq3,eq4,eq5,sol1,sol2];


(*PART 1 -- INPUT CONSTANTS AND VARIABLES*)
(*Enter an identification for each species.
Make the species with the highest product of x..i and permeability species 1.*)
species1 = nitrogen;
species2 = oxygen;
species3 = carbon dioxide;
species4 = sulfur dioxide;
species5 = nitrogen dioxide;

(*Enter permeabilities in barrers
10^(-10)*cc(stp)*cm/cm^2/sec/delta cmHg.
Make the species with the highest product of x..i and permeability species 1.*)
permeability1  = 250;
permeability2  = 500;
permeability3 = 2700;
permeability4 = 12500;
permeability5 = 6350;

(*Enter feed mole fractions.
Make the species with the highest product of x...i and permeability species 1.
Mole fractions must add to 1.  Since oxygen, x2i, is taken up with a change in carbon dioxide (x3i), sulfur
dioxide (x4i), and nitrogen dioxide (x5i), it is used as the final difference.*)
x1i = 0.8;

x3i = 0.12;
```

```
x4i = 0.005;
x5i = 0.0004;

x2i = 1-x1i-x3i-x4i-x5i;


(*Enter membrane thickness in inches*)
d = 0.000984;


(*PART 2 -- CALCULATE INTERMEDIATE VALUES*)
(*Convert permeability from barrers to
ft^3(stp)*in/min/ft^2/delta psi*)
perm1 = permeability1 * 4.0077*10^(-10);
perm2 = permeability2 * 4.0077*10^(-10);
perm3 = permeability3 * 4.0077*10^(-10);
perm4 = permeability4 * 4.0077*10^(-10);
perm5 = permeability5 * 4.0077*10^(-10);

(*Ratio of permeate pressure to inlet feed pressure*)
gamma1 = pperm/pfeed;


(*PART 3 -- SOLVE FOR INITIAL PERMEATE MOLE FRACTIONS*)
(*The following FindRoot function numerically solves for the initial concentration of species 1 on the
permeate side of the membrane. One can solve for the initial concentration of any of the species, but the
numerical method for solving the given equation converges most frequently for the species with the highest
product of x...i and permeability.

The following FindRoot function solves for yi, the initial permeate concentration of species 1, with an
initial guess of 0.5. If this value does not converge between 0 or 1, a failure message will be returned. If
this value does not converge before 20 iterations, a failure message will be returned.
```

The notation "y1i = yi /. sol1[[1]]" tells Mathematica to assign y1i the value of yi from the equation sol1. The [[1]] notation tells Mathematica to assign the first solution to sol1 it finds. Note that more than one solution is some times found. *)

```
(*The following equation is equivalent to Equation 5-7*)
sol1 = FindRoot[
(x1i*perm1/perm1)/(gamma1*(perm1/perm1-1)+x1i/yi)+
(x2i*perm2/perm1)/(gamma1*(perm2/perm1-1)+x1i/yi)+
(x3i*perm3/perm1)/(gamma1*(perm3/perm1-1)+x1i/yi)+
(x4i*perm4/perm1)/(gamma1*(perm4/perm1-1)+x1i/yi)+
(x5i*perm5/perm1)/(gamma1*(perm5/perm1-1)+x1i/yi)==1,
{yi,0.5,0,1}, MaxIterations->20];

(*The following equations are equivalent to Equation 5-8*)
y1i = yi /. sol1[[1]];
y2i = x2i*perm2/perm1/(gamma1*(perm2/perm1-1)+x1i/y1i);
y3i = x3i*perm3/perm1/(gamma1*(perm3/perm1-1)+x1i/y1i);
y4i = x4i*perm4/perm1/(gamma1*(perm4/perm1-1)+x1i/y1i);
y5i = x5i*perm5/perm1/(gamma1*(perm5/perm1-1)+x1i/y1i);

(*PART 4 -- USE NUMERICAL DIFFERENTIAL EQUATION SOLVER*)
(*note: x5 = (1-x1-x2-x3-x4) substituted below*)
(*note: y5 = (1-y1-y2-y3-y4) substituted below*)
(*The system of equations does not directly solve for species 5.*)

(*
```

The If statements are used to assign the permeate side mole fractions. In Part 3 we calculated the initial values so these are assigned when the membrane area, "area", is zero. At all other locations along the membrane the permeate side mole fractions, y, can be calculated through a material balance.

In order for the NDSolve function to work it must be given an equal number of first order ordinary differential equations and initial conditions. An example of a first order differential equation is "f'[area]",

which denotes the derivative of f with respect to area. An example of an initial condition is "f[0]", which means the value of f when area is equal to 0.

The value gamma2[area] is used to continually calculate the pressure on the feed side as it drops linearly from inlet to outlet.*)

```
(*The following equations are equivalent to Equation 5-6*)
y1  = If[area==0, y1i,
(feedflow*x1i-f[area]*x1[area])/(feedflow-f[area])];
y2  = If[area==0, y2i,
(feedflow*x2i-f[area]*x2[area])/(feedflow-f[area])];
y3 = If[area==0, y3i,
(feedflow*x3i-f[area]*x3[area])/(feedflow-f[area])];
y4 = If[area==0, y4i,
(feedflow*x4i-f[area]*x4[area])/(feedflow-f[area])];

gamma2[area_] := pperm/(pfeed-(pdrop)/totalarea*area);


(*The following equation is equivalent to Equation 5-1*)
eq1 = -pperm/gamma2[area]/d*
(perm1*(x1[area]-gamma2[area]*y1) +
perm2*(x2[area]-gamma2[area]*y2) +
perm3*(x3[area]-gamma2[area]*y3) +
perm4*(x4[area]-gamma2[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma2[area]*(1-y1-y2-y3-y4)));

(*The following equations are equivalent to Equation 5-2*)
eq2 = -pperm/gamma2[area]/d*perm1/f[area]*
(x1[area]-gamma2[area]*y1) +
pperm/gamma2[area]/d*x1[area]/f[area]*
(perm1*(x1[area]-gamma2[area]*y1) +
```

```
perm2*(x2[area]-gamma2[area]*y2) +
perm3*(x3[area]-gamma2[area]*y3) +
perm4*(x4[area]-gamma2[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma2[area]*(1-y1-y2-y3-y4)));

eq3 = -pperm/gamma2[area]/d*perm2/f[area]*
(x2[area]-gamma2[area]*y2) +
pperm/gamma2[area]/d*x2[area]/f[area]*
(perm1*(x1[area]-gamma2[area]*y1) +
perm2*(x2[area]-gamma2[area]*y2) +
perm3*(x3[area]-gamma2[area]*y3) +
perm4*(x4[area]-gamma2[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma2[area]*(1-y1-y2-y3-y4)));

eq4 = -pperm/gamma2[area]/d*perm3/f[area]*
(x3[area]-gamma2[area]*y3) +
pperm/gamma2[area]/d*x3[area]/f[area]*
(perm1*(x1[area]-gamma2[area]*y1) +
perm2*(x2[area]-gamma2[area]*y2) +
perm3*(x3[area]-gamma2[area]*y3) +
perm4*(x4[area]-gamma2[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma2[area]*(1-y1-y2-y3-y4)));

eq5 = -pperm/gamma2[area]/d*perm4/f[area]*
(x4[area]-gamma2[area]*y4) +
pperm/gamma2[area]/d*x4[area]/f[area]*
(perm1*(x1[area]-gamma2[area]*y1) +
perm2*(x2[area]-gamma2[area]*y2) +
perm3*(x3[area]-gamma2[area]*y3) +
perm4*(x4[area]-gamma2[area]*y4) +
```

```
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma2[area]*(1-y1-y2-y3-y4)));

sol2 = NDSolve[
{f'[area]==eq1, x1'[area]==eq2, x2'[area]==eq3,
x3'[area]==eq4, x4'[area]==eq5, f[0]==feedflow,
x1[0]==x1i, x2[0]==x2i, x3[0]==x3i, x4[0]==x4i},
{f,x1,x2,x3,x4},{area,0,totalarea}];


(*PART 5 -- PRODUCE RESULTS*)
fint  = f  /. sol2[[1]];
x1int = x1 /. sol2[[1]];
x2int = x2 /. sol2[[1]];
x3int = x3 /. sol2[[1]];
x4int = x4 /. sol2[[1]];

(*The following equation is equivalent to Equation 5-5*)
g = feedflow-fint[totalarea];

x5int = 1-x1int[totalarea]-x2int[totalarea]-
x3int[totalarea]-x4int[totalarea];

(*The following equations are equivalent to Equation 5-6*)
y1perm  =
(feedflow*x1i-fint[totalarea]*x1int[totalarea])/g;
y2perm  =
(feedflow*x2i-fint[totalarea]*x2int[totalarea])/g;
y3perm =
(feedflow*x3i-fint[totalarea]*x3int[totalarea])/g;
y4perm =
(feedflow*x4i-fint[totalarea]*x4int[totalarea])/g;
y5perm =
```

```
(feedflow*x5i-fint[totalarea]*x5int)/g;


Print[" "];
Print["************************************************"];
Print["SEPARATION CONDITIONS AND PERMEABILITIES"];
Print["feed pressure (psia)        = ", pfeed];
Print["feed side pressure drop (psi) = ", pdrop];
Print["permeate pressure (psia)    = ", pperm];
Print["membrane area (sqft)        = ", totalarea];
Print["membrane thickness (in)     = ", d];
Print["species 1 = ", species1];
Print["species 2 = ", species2];
Print["species 3 = ", species3];
Print["species 4 = ", species4];
Print["species 5 = ", species5];
Print["permeability of 1 (barrers) = ", permeability1];
Print["permeability of 2 (barrers) = ", permeability2];
Print["permeability of 3 (barrers) = ", permeability3];
Print["permeability of 4 (barrers) = ", permeability4];
Print["permeability of 5 (barrers) = ", permeability5];
Print["   "];
Print["FEED FLOW RATE AND MOLE FRACTIONS"];
Print["feed flow (scfm) = ", feedflow];
Print["x1i = ", x1i];
Print["x2i = ", x2i];
Print["x3i = ", x3i];
Print["x4i = ", x4i];
Print["x5i = ", x5i];
Print[" "];
Print["RETENTATE FLOW RATE AND MOLE FRACTIONS"];
Print["% of feed flow = ", fint[totalarea]/feedflow*100, "%"];
Print["retentate flow (scfm) = ", fint[totalarea]];
```

```
Print["x1 = ", x1int[totalarea]];
Print["x2 = ", x2int[totalarea]];
Print["x3 = ", x3int[totalarea]];
Print["x4 = ", x4int[totalarea]];
Print["x5 = ", 1-x1int[totalarea]-x2int[totalarea]-
x3int[totalarea]-x4int[totalarea]];
Print[" "];
Print["PERMEATE FLOW RATE AND MOLE FRACTIONS"];
Print["% of feed flow = ", g/feedflow*100, "%"];
Print["permeate flow (scfm) = ", g];
Print["y1 = ", y1perm];
Print["y2 = ", y2perm];
Print["y3 = ", y3perm];
Print["y4 = ", y4perm];
Print["y5 = ", y5perm];
Print[" "];
Print["SEPARATION RESULTS"];
Print["SO2 % Separation = ",
(feedflow*x4i-x4int[totalarea]*fint[totalarea])/
(feedflow*x4i)*100];
Print["NO2 % Separation = ",
(feedflow*x5i-x5int*fint[totalarea])/
(feedflow*x5i)*100];
Print["Permeate % of feed flow = ",(g/feedflow)*100];
Print["************************************************"];

(*end of function co5*)
)
```

APPENDIX C – Counter5

(*Brian Nelson*)
(*Single stage counter-current membrane system for the
multicomponent separation of 5 gases*)
(*Based on method of Shindo, et al ['85] without dimensionless variables*)

(*IMPORTANT*)
(*This program may not work for your specific separation. Solving for a countercurrent membrane separation is an iterative procedure, which is sometimes unstable. The values of chi, rho, and phi (see PART 6) may need to be changed. A completely different method for determining these values may also be necessary.*)

(*INSTRUCTIONS*)
(*

1. This program is written as a function in Mathematica. In order to operate the function "counter5[{}]", you must first click into the cell below (cells are designated by the brackets to the right.)--------->

2. Press Shift-Enter to enter the function into Mathematica memory.

3. The function is designed to place some or all of the following variables inside the function brackets [{}]. The function is currently set up to input the following five values in the given order.

   a. feed pressure (psia)
   b. feed side pressure drop (psi)

c. permeate pressure (psia)
d. feed flow rate (scfm)
e. membrane area (sq ft)

If you want the function to include more variables than listed directly above, simply remove the variable from PART 1 - INPUT CONSTANTS AND VARIABLES and put the variable, with the identical name, into the function designation (explained below) followed by an underscore, _. Other values, which can be included in the function, are listed below. Note: When running the function you must always input the variable values of a function in the same order as the function is written. Also, if you make a change to the function you must re-enter it into the current memory by pressing Shift-Enter while the cursor is in the cell containing the function.

f. membrane thickness (inches)
g. species 1 permeability (barrer)
h. species 2 permeability (barrer)
i. species 3 permeability (barrer)
j. species 4 permeability (barrer)
k. species 5 permeability (barrer)
l. species 1 feed mole fraction
m. species 2 feed mole fraction
n. species 3 feed mole fraction
o. species 4 feed mole fraction
p. species 5 feed mole fraction

4. Definition of the following variables (Note: "..." stands for the species number 1, 2, 3, etc.):
   a. permeability... = a permeability in barrers.
   b. pfeed and pperm = feed and permeate pressures in psia.
   c. pdrop = pressure drop on the feed side in psi
   d. x...i = an initial feed mole fraction
   e. area = variable for membrane area
   f. d = membrane thickness in inches.

g. feedflow = feed flow in scfm

h. perm... = a permeability in
ft^3 (stp)*in/min/ft^2/delta psi.

i. gamma1[area] = a pressure ratio of permeate pressure to the feed pressure as it changes.

j. gamma2 = pressure ratio of permeate pressure to
feed pressure at exit.

k. totalarea = the total membrane area in sqft.

l. sol... = a solution or solution set

m. yo is explained in Part 4 below.

n. y...o = initial permeate mole fractions at the feed side exit.

o. x...[area] = a mole fraction at some point
along the membrane.

p. eq... = equation number...

q. f[area] = retentate flow rate at some point along
the membrane.

r. ...int = integrated solution set, an InterpolatingFunction, of some value.
note: you can find a value by entering ...int[area].

s. g = permeate flow rate in scfm.

t. ...perm = permeate mole fractions after exposure to the entire membrane surface.

u. x...o = a retentate mole fraction.

v. ...compare = values used in While loop to test for convergence.

w. chi,phi,rho = ratio for calculating a weighted
average for the next iteration.

x. ...new = values calculated for the next iteration.

y. iterations = counts the number of iterations
required to achieve convergence.

z. species... = identification of the gaseous species.

5. Keeping the window with the entered "counter5[]" function open, enter into a new window (by clicking File-New in the pull-down menu) the function with the desired variables, as given in the following example or as the function has been changed, and press Shift-Enter.

Example: counter5[{65,0,12,50,100}]

```
*)

(*A function is created in Matematica when Mathematica sees the following form:
        function[variable1_,variable2_, etc...] := (task)
The function name can be anything continuous and should not start with a capital letter (so they are not
confused with Mathematica functions). Variables are enclosed in brackets, [], and immediately followed
by an underscore, _. Please note, in the following function the variables are written in a list inside the
brackets. A list is denoted by curvy brackets {}. The := notation tells Mathematica to delay the calculation
until the function is called upon. The task or set of equations or procedures is enclosed in parenthases ().
Each line of code within the parentheses must end with a semicolon. Mathematica will not stop reading a
function until all parenthases are closed. *)


counter5[{pfeed_,pdrop_,pperm_,feedflow_,totalarea_}] := (


(*PART 1 -- INPUT CONSTANTS AND VARIABLES*)
(*Enter an identification for each species.
Make the species with the highest product of x..o and permeability species 1.*)
species1 = nitrogen;
species2 = sulfur dioxide;
species3 = nitrogen dioxide;
species4 = carbon dioxide;
species5 = oxygen;


(*Enter permeabilities in barrers
10^(-10)*cc(stp)*cm/cm^2/sec/delta cmHg.
Make the species with the highest product of x..o and permeability species 1.*)
permeability1 = 250;
permeability2 = 12500;
permeability3 = 6350;
permeability4 = 2700;
```

```
permeability5 = 500;

(*Enter feed mole fractions. These will not be used during
NDSolve calculation, but they will be used as a comparison to calculated output for further iterations.
Make the species with the highest product of x...o and permeability species 1.
Mole fractions must add to 1. Since oxygen, x5i, is taken up with a change in carbon dioxide (x4i), sulfur
dioxide (x2i), and nitrogen dioxide (x3i), it is used as the final difference *)
x1i = 0.8;
x2i = 0.005;
x3i = 0.0004;
x4i = 0.12;
x5i = 1-x1i-x2i-x3i-x4i;

(*enter membrane thickness in inches*)
d = 0.000984;

(*PART 2 -- CALCULATE INTERMEDIATE VALUES*)
(*Convert permeability from barrers to
ft^3(stp)*in/min/ft^2/delta psi*)
perm1 = permeability1 * 4.0077*10^(-10);
perm2 = permeability2 * 4.0077*10^(-10);
perm3 = permeability3 * 4.0077*10^(-10);
perm4 = permeability4 * 4.0077*10^(-10);
perm5 = permeability5 * 4.0077*10^(-10);


(*The following code is for a cocurrent membrane system.
The retentate mole fractions and permeate flow rate will
be used as an initial guess for the counter-current solution.*)

(*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%*)
(*COCURRENT PART 1 -- VALUES ALREADY INPUT*)
```

```
Clear[gamma1,gamma2,area,eq1,eq2,eq3,eq4,eq5,sol1,sol2];

(*Ratio of permeate pressure to inlet feed pressure*)
gamma1 = pperm/pfeed;

(*COCURRENT PART 2 -- SOLVE FOR INITIAL PERMEATE MOLE FRACTIONS*)
(*see cocurrent program for explaination of code*)

sol1 = FindRoot[
(x1i*perm1/perm1)/(gamma1*(perm1/perm1-1)+x1i/yi)+
(x2i*perm2/perm1)/(gamma1*(perm2/perm1-1)+x1i/yi)+
(x3i*perm3/perm1)/(gamma1*(perm3/perm1-1)+x1i/yi)+
(x4i*perm4/perm1)/(gamma1*(perm4/perm1-1)+x1i/yi)+
(x5i*perm5/perm1)/(gamma1*(perm5/perm1-1)+x1i/yi)==1,
{yi,0.5,0,1}, MaxIterations->20];

y1i = yi /. sol1[[1]];
y2i = x2i*perm2/perm1/(gamma1*(perm2/perm1-1)+x1i/y1i);
y3i = x3i*perm3/perm1/(gamma1*(perm3/perm1-1)+x1i/y1i);
y4i = x4i*perm4/perm1/(gamma1*(perm4/perm1-1)+x1i/y1i);
y5i = x5i*perm5/perm1/(gamma1*(perm5/perm1-1)+x1i/y1i);


(*COCURRENT PART 3 -- USE NUMERICAL DIFFERENTIAL EQUATION SOLVER*)
(*see cocurrent program for explaination of code*)

y1  = If[area==0, y1i,
(feedflow*x1i-f[area]*x1[area])/(feedflow-f[area])];
y2 = If[area==0, y2i,
(feedflow*x2i-f[area]*x2[area])/(feedflow-f[area])];
y3 = If[area==0, y3i,
(feedflow*x3i-f[area]*x3[area])/(feedflow-f[area])];
```

```
y4 = If[area==0, y4i,
(feedflow*x4i-f[area]*x4[area])/(feedflow-f[area])];

gamma2[area_] := pperm/(pfeed-(pdrop)/totalarea*area);

eq1 = -pperm/gamma2[area]/d*
(perm1*(x1[area]-gamma2[area]*y1) +
perm2*(x2[area]-gamma2[area]*y2) +
perm3*(x3[area]-gamma2[area]*y3) +
perm4*(x4[area]-gamma2[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma2[area]*(1-y1-y2-y3-y4)));

eq2 = -pperm/gamma2[area]/d*perm1/f[area]*
(x1[area]-gamma2[area]*y1) +
pperm/gamma2[area]/d*x1[area]/f[area]*
(perm1*(x1[area]-gamma2[area]*y1) +
perm2*(x2[area]-gamma2[area]*y2) +
perm3*(x3[area]-gamma2[area]*y3) +
perm4*(x4[area]-gamma2[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma2[area]*(1-y1-y2-y3-y4)));

eq3 = -pperm/gamma2[area]/d*perm2/f[area]*
(x2[area]-gamma2[area]*y2) +
pperm/gamma2[area]/d*x2[area]/f[area]*
(perm1*(x1[area]-gamma2[area]*y1) +
perm2*(x2[area]-gamma2[area]*y2) +
perm3*(x3[area]-gamma2[area]*y3) +
perm4*(x4[area]-gamma2[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma2[area]*(1-y1-y2-y3-y4)));
```

```
eq4 = -pperm/gamma2[area]/d*perm3/f[area]*
(x3[area]-gamma2[area]*y3) +
pperm/gamma2[area]/d*x3[area]/f[area]*
(perm1*(x1[area]-gamma2[area]*y1) +
perm2*(x2[area]-gamma2[area]*y2) +
perm3*(x3[area]-gamma2[area]*y3) +
perm4*(x4[area]-gamma2[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma2[area]*(1-y1-y2-y3-y4)));

eq5 = -pperm/gamma2[area]/d*perm4/f[area]*
(x4[area]-gamma2[area]*y4) +
pperm/gamma2[area]/d*x4[area]/f[area]*
(perm1*(x1[area]-gamma2[area]*y1) +
perm2*(x2[area]-gamma2[area]*y2) +
perm3*(x3[area]-gamma2[area]*y3) +
perm4*(x4[area]-gamma2[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma2[area]*(1-y1-y2-y3-y4)));

sol2 = NDSolve[
{f'[area]==eq1, x1'[area]==eq2, x2'[area]==eq3,
x3'[area]==eq4, x4'[area]==eq5, f[0]==feedflow,
x1[0]==x1i, x2[0]==x2i, x3[0]==x3i, x4[0]==x4i},
{f,x1,x2,x3,x4},{area,0,totalarea}];


(*COCURRENT PART 4 -- PRODUCE RESULTS*)
fint  = f  /. sol2[[1]];
x1int = x1 /. sol2[[1]];
x2int = x2 /. sol2[[1]];
x3int = x3 /. sol2[[1]];
x4int = x4 /. sol2[[1]];
```

```
g = feedflow-fint[totalarea];

Clear[gamma1,gamma2,area,eq1,eq2,eq3,eq4,eq5,sol1,sol2];

(*END COCURRENT CODE FOR INITIAL COUNTERCURRENT GUESS*)
(*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%*)


(*Assign initial guess for counter-current model*)
x1o = x1int[totalarea];
x2o = x2int[totalarea];
x3o = x3int[totalarea];
x4o = x4int[totalarea];
x5o = 1-x1int[totalarea]-x2int[totalarea]-
x3int[totalarea]-x4int[totalarea];
permflow = g;

(*Ratio of permeate pressure to outlet feed pressure*)
gamma2 = pperm/(pfeed-pdrop);



(*PART 3 -- INITIATE WHILE LOOP FOR ITERATIONS*)
(*Assign compare values so the While loop will initiate*)
fcompare = 0;
x1compare = 0;
x2compare = 0;
x3compare = 0;
x4compare = 0;
x5compare = 0;
```

```
iterations = 0;


(*Start While loop here*)
(*The calculation will end once the calculated values of inlet mole fractions and inlet flow rate are
sufficiently close to the given values*)
While[Abs[(fcompare-feedflow)/feedflow]>0.001 ||
Abs[(x1compare-x1i)/x1i]>0.001 ||
Abs[(x2compare-x2i)/x2i]>0.001 ||
Abs[(x3compare-x3i)/x3i]>0.001 ||
Abs[(x4compare-x4i)/x4i]>0.001 ||
Abs[(x5compare-x5i)/x5i]>0.001,


(*PART 4 -- SOLVE FOR PERMEATE MOLE FRACTIONS AT FEED SIDE EXIT*)
(*The following FindRoot function numerically solves for the initial concentration, at the feed side exit, of
species 1 on the permeate side of the membrane. One can solve for the this concentration for any of the
species, but the numerical method for solving the given equation converges most frequently for the species
with the highest product of x... and permeability.

The following FindRoot function solves for yo, the initial permeate concentration, at the feed side exit, of
species 1, with an initial guess of 0.5. If this value does not converge between 0 or 1, a failure message
will be returned. If this value does not converge before 20 iterations, a failure message will be returned.

The notation "y1o = yo /. sol1[[1]]" tells Mathematica to assign y1o the value of yo from the equation sol1.
The [[1]] notation tells Mathematica to assign the first solution to sol1 it finds. Note that more than one
solution is some times found. *)

(*the following equation is equivalent to Equation 5-7*)
sol1 = FindRoot[
(x1o*perm1/perm1)/(gamma2*(perm1/perm1-1)+x1o/yo)+
(x2o*perm2/perm1)/(gamma2*(perm2/perm1-1)+x1o/yo)+
(x3o*perm3/perm1)/(gamma2*(perm3/perm1-1)+x1o/yo)+
```

```
(x4o*perm4/perm1)/(gamma2*(perm4/perm1-1)+x1o/yo)+
(x5o*perm5/perm1)/(gamma2*(perm5/perm1-1)+x1o/yo)==1,
{yo,0.1,0,1}, MaxIterations->20];

(*The following equations are equivalent to Equation 5-8*)
y1o = yo /. sol1[[1]];
y2o = x2o*perm2/perm1/(gamma2*(perm2/perm1-1)+x1o/y1o);
y3o = x3o*perm3/perm1/(gamma2*(perm3/perm1-1)+x1o/y1o);
y4o = x4o*perm4/perm1/(gamma2*(perm4/perm1-1)+x1o/y1o);
y5o = x5o*perm5/perm1/(gamma2*(perm5/perm1-1)+x1o/y1o);


(*PART 5 -- USE NUMERICAL DIFFERENTIAL EQUATION SOLVER*)
(*note: x5 = (1-x1-x2-x3-x4) substituted below*)
(*note: y5 = (1-y1-y2-y3-y4) substituted below*)
(*The system of equations does not directly solve for species 5.*)

(*
The If statements are used to assign the permeate side mole fractions. In Part 4 we calculated the initial
values, at the feed side exit, so these are assigned when the transformed membrane area, area, is 0. At all
other locations along the membrane the permeate side mole fractions, y, can be calculated through a
material balance.

In order for the NDSolve function to work it must be given an equal number of first order ordinary
differential equations and initial conditions. An example of a first order differential equation is "f'[area]",
which denotes the derivative of f with respect to area. An example of an initial condition is "f[0]", which
means the value of f when area is equal to 0.

The value gamma1[area] is used to continually calculate the pressure on the feed side as it increases
linearly from outlet to inlet*)
```

```
(*NOTE: This solution requires a coordinate transformation since Mathematica will not numerically solve
differential equations, using the NDSolve function, when going from a large to a small number. Therefore,
let area(transformed) = totalarea - area  *)

(*the following equations are equivalent to Equation 5-10*)
y1  = If[area==0, y1o,
(x1[area]*f[area]-x1o*(feedflow-permflow))/
(f[area]-(feedflow-permflow))];

y2  = If[area==0, y2o,
(x2[area]*f[area]-x2o*(feedflow-permflow))/
(f[area]-(feedflow-permflow))];

y3  = If[area==0, y3o,
(x3[area]*f[area]-x3o*(feedflow-permflow))/
(f[area]-(feedflow-permflow))];

y4  = If[area==0, y4o,
(x4[area]*f[area]-x4o*(feedflow-permflow))/
(f[area]-(feedflow-permflow))];


gamma1[area_] := pperm/((pfeed-pdrop)+(pdrop)/totalarea*area);

(*The following equation is equivalent to Equation 5-1*)
eq1 = -pperm/gamma1[area]/d*
(perm1*(x1[area]-gamma1[area]*y1) +
perm2*(x2[area]-gamma1[area]*y2) +
perm3*(x3[area]-gamma1[area]*y3) +
perm4*(x4[area]-gamma1[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma1[area]*(1-y1-y2-y3-y4)));
```

```
(*the following equations are equivalent to Equation 5-2*)
eq2 = -pperm/gamma1[area]/d*perm1/f[area]*
(x1[area]-gamma1[area]*y1) +
pperm/gamma1[area]/d*x1[area]/f[area]*
(perm1*(x1[area]-gamma1[area]*y1) +
perm2*(x2[area]-gamma1[area]*y2) +
perm3*(x3[area]-gamma1[area]*y3) +
perm4*(x4[area]-gamma1[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma1[area]*(1-y1-y2-y3-y4)));

eq3 = -pperm/gamma1[area]/d*perm2/f[area]*
(x2[area]-gamma1[area]*y2) +
pperm/gamma1[area]/d*x2[area]/f[area]*
(perm1*(x1[area]-gamma1[area]*y1) +
perm2*(x2[area]-gamma1[area]*y2) +
perm3*(x3[area]-gamma1[area]*y3) +
perm4*(x4[area]-gamma1[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma1[area]*(1-y1-y2-y3-y4)));

eq4 = -pperm/gamma1[area]/d*perm3/f[area]*
(x3[area]-gamma1[area]*y3) +
pperm/gamma1[area]/d*x3[area]/f[area]*
(perm1*(x1[area]-gamma1[area]*y1) +
perm2*(x2[area]-gamma1[area]*y2) +
perm3*(x3[area]-gamma1[area]*y3) +
perm4*(x4[area]-gamma1[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma1[area]*(1-y1-y2-y3-y4)));

eq5 = -pperm/gamma1[area]/d*perm4/f[area]*
(x4[area]-gamma1[area]*y4) +
```

```
pperm/gamma1[area]/d*x4[area]/f[area]*
(perm1*(x1[area]-gamma1[area]*y1) +
perm2*(x2[area]-gamma1[area]*y2) +
perm3*(x3[area]-gamma1[area]*y3) +
perm4*(x4[area]-gamma1[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma1[area]*(1-y1-y2-y3-y4)));


sol2 = NDSolve[{f'[area]==-eq1, x1'[area]==-eq2,
x2'[area]==-eq3, x3'[area]==-eq4, x4'[area]==-eq5,
f[0]==feedflow-permflow, x1[0]==x1o, x2[0]==x2o, x3[0]==x3o,
x4[0]==x4o},
{f,x1,x2,x3,x4},{area,0,totalarea}];


(*PART 6 -- PRODUCE INTERNAL RESULTS AND VALUES FOR
NEXT ITERATION*)
fint  =  f /. sol2[[1]];
x1int = x1 /. sol2[[1]];
x2int = x2 /. sol2[[1]];
x3int = x3 /. sol2[[1]];
x4int = x4 /. sol2[[1]];


(*Re-Calculate values of the final permeate flow rate and permeate mole fractions.  These values were first
calculated by the cocurrent model above.*)

(*the following equation is equivalent to Equation 5-11*)
g = fint[totalarea]-(feedflow-permflow);

(*the following equations are equivalent to Equation 5-12*)
```

```
y1perm = (x1int[totalarea]*fint[totalarea]-x1o*(feedflow-permflow))/(fint[totalarea]-(feedflow-
permflow));

y2perm = (x2int[totalarea]*fint[totalarea]-x2o*(feedflow-permflow))/(fint[totalarea]-(feedflow-
permflow));

y3perm = (x3int[totalarea]*fint[totalarea]-x3o*(feedflow-permflow))/(fint[totalarea]-(feedflow-
permflow));

y4perm = (x4int[totalarea]*fint[totalarea]-x4o*(feedflow-permflow))/(fint[totalarea]-(feedflow-
permflow));


(*Obtain values for next iteration and assign. The next guess for the outlet mole fractions and permflow
are calculated through an overall material balance and individual material balances. Values for these
balances are obtained from the previous iteration as well as given values. The next value is calculated
through a weighted average between the previous value and the newly calculated value. Chi, phi, and rho
are the weighting on the previous value. The larger chi, phi, or rho is the faster the new values will change,
but this may also result in instabilities and a solution may not result. One should note that the species with
the highest permeability tends to move the fastest and often requires the most dampening.*)

iterations = 1 + iterations;

If[feedflow/totalarea>0.09,{chi=0.5,rho=0.6,phi=0.6},
If[feedflow/totalarea>0.07,{chi=0.6,rho=0.7,phi=0.8},
If[feedflow/totalarea>0.06,{chi=0.7,rho=0.8,phi=0.9},
If[feedflow/totalarea>0.05,{chi=0.8,rho=0.9,phi=0.95},
If[feedflow/totalarea>0.039,{chi=0.8,rho=0.95,phi=0.99},
If[feedflow/totalarea>0.035,{chi=0.8,rho=0.97,phi=0.995},
If[feedflow/totalarea>0.032,{chi=0.9,rho=0.99,phi=0.999},
If[feedflow/totalarea>0.031,{chi=0.9,rho=0.99,phi=0.9995},
{chi=0.95,rho=0.99,phi=0.99995}]]]]]]];
```

```
(*The following equations are equivalent to Equations 5-13 and 5-14*)
x1onew = chi*x1o + (1-chi)*((x1i*feedflow-y1perm*g)/(feedflow-g));
x2onew = phi*x2o + (1-phi)*((x2i*feedflow-y2perm*g)/(feedflow-g));
x3onew = rho*x3o + (1-rho)*((x3i*feedflow-y3perm*g)/(feedflow-g));
x4onew = chi*x4o + (1-chi)*((x4i*feedflow-y4perm*g)/(feedflow-g));
x5onew = 1-x1onew-x2onew-x3onew-x4onew;
permflownew = chi*permflow + (1-chi)*g;

(*set the new values*)
permflow = permflownew;
x1o = x1onew;
x2o = x2onew;
x3o = x3onew;
x4o = x4onew;
x5o = x5onew;

(*
(*temporary print statements*)
Print["iterations = ", iterations];
Print["CALCULATED VALUES FOR FEED FLOW RATE AND MOLE FRACTIONS"];
Print["feed flow (stp) = ", fint[totalarea]];
Print["x1i calc. = ", x1int[totalarea]];
Print["x2i calc. = ", x2int[totalarea]];
Print["x3i calc. = ", x3int[totalarea]];
Print["x4i calc. = ", x4int[totalarea]];
Print["x5i calc. = ", 1-x1int[totalarea]-x2int[totalarea]-x3int[totalarea]-x4int[totalarea]];
Print[" "];
Print["ENTERED STAGE CUT AND RETENTATE MOLE FRACTIONS"];
Print["guessed permeate flow (stp) = ", permflow];
Print["retentate flow (stp) = ", feedflow-permflow];
Print["x1o = ", x1o];
Print["x2o = ", x2o];
Print["x3o = ", x3o];
```

```
Print["x4o = ", x4o];
Print["x5o = ", x5o];
Print[" "];
*)


(*Assign variable values for fint[totalarea], x1int[totalarea], x2int[totalarea], x3int[totalarea] so they can be
compared in While loop *)

fcompare  = fint[totalarea];
x1compare = x1int[totalarea];
x2compare = x2int[totalarea];
x3compare = x3int[totalarea];
x4compare = x4int[totalarea];
x5compare = 1-x1compare-x2compare-x3compare-x4compare;

(*End of While loop*)
];


(*PART 7 -- PRINT RESULTS*)
Print["iterations = ", iterations];
Print["SEPARATION CONDITIONS AND PERMEABILITIES"];
Print["feed pressure (psia)          = ", pfeed];
Print["feed side pressure drop (psia)  = ", pdrop];
Print["permeate pressure (psia)       = ", pperm];
Print["membrane area (sqft)          = ", totalarea];
Print["membrane thickness (in)        = ", d];
Print["species 1 = ", species1];
Print["species 2 = ", species2];
Print["species 3 = ", species3];
Print["species 4 = ", species4];
Print["species 5 = ", species5];
```

```
Print["permeability of 1 (barrers) = ", permeability1];
Print["permeability of 2 (barrers) = ", permeability2];
Print["permeability of 3 (barrers) = ", permeability3];
Print["permeability of 4 (barrers) = ", permeability4];
Print["permeability of 5 (barrers) = ", permeability5];
Print[" "];
Print["ACTUAL VALUES FOR FEED FLOW RATE AND MOLE FRACTIONS"];
Print["feed flow (stp) = ", feedflow];
Print["x1i actual = ", x1i];
Print["x2i actual = ", x2i];
Print["x3i actual = ", x3i];
Print["x4i actual = ", x4i];
Print["x5i actual = ", x5i];
Print[" "];
Print["CALCULATED VALUES FOR FEED FLOW RATE AND MOLE FRACTIONS"];
Print["feed flow (stp) = ", fint[totalarea]];
Print["x1i calc. = ", x1int[totalarea]];
Print["x2i calc. = ", x2int[totalarea]];
Print["x3i calc. = ", x3int[totalarea]];
Print["x4i calc. = ", x4int[totalarea]];
Print["x5i calc. = ", 1-x1int[totalarea]-x2int[totalarea]-
x3int[totalarea]-x4int[totalarea]];
Print[" "];
Print["ENTERED STAGE CUT AND RETENTATE MOLE FRACTIONS"];
Print["guessed permeate flow (stp) = ", permflow];
Print["retentate flow (stp) = ", feedflow-permflow];
Print["% of feed flow = ", (feedflow-permflow)/feedflow*100, "%"];
Print["x1o = ", x1o];
Print["x2o = ", x2o];
Print["x3o = ", x3o];
Print["x4o = ", x4o];
Print["x5o = ", x5o];
Print[" "];
```

```
Print["PERMEATE FLOW RATE AND MOLE FRACTIONS"];
Print["calculated permeate flow (stp) = ", g];
Print["% of feed flow = ", g/feedflow*100, "%"];
Print["y1 = ", y1perm];
Print["y2 = ", y2perm];
Print["y3 = ", y3perm];
Print["y4 = ", y4perm];
Print["y5 = ", 1-y1perm-y2perm-y3perm-y4perm];
Print[" "];
Print["SEPARATION RESULTS"];
Print["SO2 % Separation = ",
(x2i*feedflow-x2o*(feedflow-permflow))/(feedflow*x2i)*100];
Print["NO2 % Separation = ",
(x3i*feedflow-x3o*(feedflow-permflow))/(feedflow*x3i)*100];
Print["Permeate % of feed flow = ", g/feedflow*100];


(*end function counter5*)
)
```

APPENDIX D – Cross5

(*Brian Nelson*)
(*Single stage crosscurrent membrane system for the multi-component separation of 5 gases*)
(*Based on method of Shindo, et al ['85] without dimensionless variables*)

(*INSTRUCTIONS*)
(*

1. This program is written as a function in Mathematica. In order to operate the function "cross5[{}]", you must first click the cursor into the cell below (cells are designated by the brackets to the right.)------->

2. Press Shift-Enter to enter the function into Mathematica memory.

3. The function is designed to place some or all of the following variables inside the function brackets [{}]. The function is currently set up to input the following five values in the given order.

    a. feed pressure (psia)
    b. feed side pressure drop (psi)
    c. permeate pressure (psia)
    d. feed flow rate (scfm)
    e. membrane area (sq ft)

If you want the function to include more variables than listed directly above, simply remove the variable from PART 1 - INPUT CONSTANTS AND VARIABLES and put the variable, with the identical name, into the function designation (explained below) followed by an underscore, _. Other values, which can be included in the function, are listed below. _Note: When running the function you must always input the variable values of a function in the same order as the function is written. Also, if you make a change to the

function you must re-enter it into the current memory by pressing Shift-Enter while the cursor is in the cell containing the function.

f. membrane thickness (inches)
g. species 1 permeability (barrer)
h. species 2 permeability (barrer)
i. species 3 permeability (barrer)
j. species 4 permeability (barrer)
k. species 5 permeability (barrer)
l. species 1 feed mole fraction
m. species 2 feed mole fraction
n. species 3 feed mole fraction
o. species 4 feed mole fraction
p. species 5 feed mole fraction

4. Definition of the following variables (Note: "..." stands for the species number 1, 2, 3, etc.):
   a. permeability... = a permeability in barrers.
   b. pfeed and pperm = feed and permeate pressures in psia.
   c. pdrop = pressure drop on the feed side in psi.
   d. x...i = an initial feed mole fraction.
   e. area = variable for membrane area.
   f. d = membrane thickness in inches.
   g. feedflow = feed flow in scfm
   h. perm... = a permeability in ft^3 (stp)*in/min/ft^2/delta psi.
   i. gamma1[area] = pressure ratio of permeate pressure to feed pressure as it changes.
   j. totalarea = the total membrane area in sqft.
   k. sol... = a solution or solution set.
   l. y is explained in Part 3 below.
   m. y...[] = permeate mole fractions as they change.
   o. x...[area] = a mole fraction at some point.

along the membrane.

p. eq... = equation number...

q. f[area] = retentate flow rate at some point along the membrane.

r. ...int = integrated solution set, an InterpolatingFunction, of some value. note: you can find a value by entering ...int[area].

s. g = permeate flow rate in scfm.

t. ...perm = permeate mole fractions after exposure to the entire membrane surface.

u. species... = identification of gaseous species.

5. Keeping the window with the entered "cross5[]" function open, enter into a new window (by clicking File-New from the pull-down menu) the function with the desired variables, as given in the following example or as the function has been changed, and press Shift-Enter.

Example: cross5[{65,0,12,50,100}]

*)

(*A function is created in Mathematica when Mathematica sees the following form:

function[variable1_,variable2_, etc...] := (task)

The function name can be anything continuous and should not start with a capital letter (so they are not confused with Mathematica functions). Variables are enclosed in brackets, [], and immediately followed by an underscore, _. Please note, in the following function the variables are written in a list inside the brackets. A list is denoted by curvy brackets {}. The := notation tells Mathematica to delay the calculation until the function is called upon. The task or set of equations or procedures is enclosed in parentheses (). Each line of code within the parentheses must end with a semicolon. Mathematica will not stop reading a function until all parentheses are closed. *)

```
cross5[{pfeed_,pdrop_,pperm_,feedflow_,totalarea_}] := (

(*Make sure other functions do not affect the current calculation*)
Clear[area,eq1,eq2,eq3,eq4,eq5,sol1,sol2];
```

```
(*PART 1 -- INPUT CONSTANTS AND VARIABLES*)
(*Enter an identification for each species.
Make the species with the highest product of x..i and permeability species 1.*)
species1 = nitrogen;
species2 = oxygen;
species3 = carbon dioxide;
species4 = sulfur dioxide;
species5 = nitrogen dioxide;


(*Enter permeabilities in barrers
10^(-10)*cc(stp)*cm/cm^2/sec/delta cmHg.
Make the species with the highest product of x..i and permeability species 1.*)
permeability1  = 250;
permeability2  = 500;
permeability3 = 2700;
permeability4 = 12500;
permeability5 = 6350;

(*Enter feed mole fractions.  Make the species with the highest product of x...i and permeability species1.
Mole fractions must add to 1.  Since oxygen, x2i, is taken up with a change in carbon dioxide (x3i), sulfur
dioxide (x4i), and nitrogen dioxide (x5i), it is used as the final difference *)
x1i = 0.8;
x3i = 0.12;
x4i = 0.005;
x5i = 0 0004;

x2i = 1-x1i-x3i-x4i-x5i;


(*Enter membrane thickness in inches*)
d = 0.000984;
```

```
(*PART 2 -- CALCULATE INTERMEDIATE VALUES*)
(*Convert permeability from barrers to
ft^3(stp)*in/min/ft^2/delta psi*)
perm1 = permeability1 * 4.0077*10^(-10);
perm2 = permeability2 * 4.0077*10^(-10);
perm3 = permeability3 * 4.0077*10^(-10);
perm4 = permeability4 * 4.0077*10^(-10);
perm5 = permeability5 * 4.0077*10^(-10);


(*PART 3 -- USE NUMERICAL DIFFERENTIAL EQUATION SOLVER*)
(*
note: x5 = (1-x1-x2-X3-x4) substituted below
note: y5 = (1-y1-y2-Y3-y4) substituted below
The system of equations does not directly solve for species 5.
```

In order for the NDSolve function to work it must be given an equal number of first order ordinary differential equations and initial conditions. An example of a first order differential equation is "f'[area]", which denotes the derivative of f with respect to area. An example of an initial condition is "f[0]", which means the value of f when area is equal to 0.

Both the pressure ratio (gamma1[]) and the permeate mole fractions are calculated continually along the membrane.

The Module function tells Mathematica to treat the specified variable, {y}, to be a local variable. The value, y, is only defined inside the Module function.

The FindRoot function numerically solves for the concentration of species 1 on the permeate side of the membrane. One can solve for the concentration of any of the species, but the numerical method for solving the given equation converges most frequently for the species with the highest product of x... and permeability.

The following FindRoot function solves for y, the permeate concentration of species 1, with an initial guess of 0.5. If this value does not converge between 0 or 1, a failure message will be returned. If this value does not converge before 20 iterations, a failure message will be returned.

The notation "Module[{y}, y /. FindRoot[..." tells Mathematica to assign y1[...] the value of y. The use of ?NumberQ checks values input to y1[...] to make sure they are real numbers. For some unknown reason this program will not run without ?NumberQ .*)

```
gamma1[area_] := pperm/(pfeed-(pdrop)/totalarea*area);

(*The following equation is equivalent to Equation 5-7*)
y1[x1area_?NumberQ,x2area_?NumberQ,x3area_?NumberQ,
x4area_?NumberQ,farea_?NumberQ,area_?NumberQ,gamma1area_?NumberQ] :=
Module[{y}, y /. FindRoot[
(x1area*perm1/perm1)/(gamma1area*(perm1/perm1-1)+x1area/y)+
(x2area*perm2/perm1)/(gamma1area*(perm2/perm1-1)+x1area/y)+
(x3area*perm3/perm1)/(gamma1area*(perm3/perm1-1)+x1area/y)+
(x4area*perm4/perm1)/(gamma1area*(perm4/perm1-1)+x1area/y)+
((1-x1area-x2area-x3area-x4area)*perm5/perm1)/
(gamma1area*(perm5/perm1-1)+x1area/y)==1,{y,0.5,0,1}]];

(*The following equations are equivalent to Equation 5-8*)
y2 := x2[area]*perm2/perm1/
(gamma1[area]*(perm2/perm1-1)+x1[area]/
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]);

y3 := x3[area]*perm3/perm1/
(gamma1[area]*(perm3/perm1-1)+x1[area]/
y1[x1[area],x2[area],x3[area],x4[area],
```

```
f[area],area,gamma1[area]]);

y4 := x4[area]*perm4/perm1/
(gamma1[area]*(perm4/perm1-1)+x1[area]/
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]);


(*The following equation is equivalent to Equation 5-1*)
eq1 := -pperm/gamma1[area]/d*(perm1*(x1[area]-gamma1[area]*
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]) +
perm2*(x2[area]-gamma1[area]*y2) +
perm3*(x3[area]-gamma1[area]*y3) +
perm4*(x4[area]-gamma1[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma1[area]*(1-
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]-y2-y3-y4)));
(*The following equations are equivalent to Equation 5-2*)
eq2 := -pperm/gamma1[area]/d*perm1/f[area]*
(x1[area]-gamma1[area]*
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]) +
pperm/gamma1[area]/d*x1[area]/f[area]*
(perm1*(x1[area]-gamma1[area]*
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]) +
perm2*(x2[area]-gamma1[area]*y2) +
perm3*(x3[area]-gamma1[area]*y3) +
perm4*(x4[area]-gamma1[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma1[area]*(1-
```

```
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]-y2-y3-y4)));

eq3 := -pperm/gamma1[area]/d*perm2/f[area]*
(x2[area]-gamma1[area]*y2) +
pperm/gamma1[area]/d*x2[area]/f[area]*
(perm1*(x1[area]-gamma1[area]*
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]) +
perm2*(x2[area]-gamma1[area]*y2) +
perm3*(x3[area]-gamma1[area]*y3) +
perm4*(x4[area]-gamma1[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma1[area]*(1-
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]-y2-y3-y4)));

eq4 := -pperm/gamma1[area]/d*perm3/f[area]*
(x3[area]-gamma1[area]*y3) +
pperm/gamma1[area]/d*x3[area]/f[area]*
(perm1*(x1[area]-gamma1[area]*
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]) +
perm2*(x2[area]-gamma1[area]*y2) +
perm3*(x3[area]-gamma1[area]*y3) +
perm4*(x4[area]-gamma1[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma1[area]*(1-
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]-y2-y3-y4)));

eq5 := -pperm/gamma1[area]/d*perm4/f[area]*
(x4[area]-gamma1[area]*y4) +
```

```
pperm/gamma1[area]/d*x4[area]/f[area]*
(perm1*(x1[area]-gamma1[area]*
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]) +
perm2*(x2[area]-gamma1[area]*y2) +
perm3*(x3[area]-gamma1[area]*y3) +
perm4*(x4[area]-gamma1[area]*y4) +
perm5*((1-x1[area]-x2[area]-x3[area]-x4[area]) -
gamma1[area]*(1-
y1[x1[area],x2[area],x3[area],x4[area],
f[area],area,gamma1[area]]-y2-y3-y4)));

sol2 = NDSolve[
{f'[area]==eq1,
x1'[area]==eq2,
x2'[area]==eq3,
x3'[area]==eq4,
x4'[area]==eq5,
f[0]==feedflow, x1[0]==x1i, x2[0]==x2i,
x3[0]==x3i, x4[0]==x4i},
{f,x1,x2,x3,x4},{area,0,totalarea}];


(*PART 4 -- PRODUCE RESULTS*)
fint  = f  /. sol2[[1]];
x1int = x1 /. sol2[[1]];
x2int = x2 /. sol2[[1]];
x3int = x3 /. sol2[[1]];
x4int = x4 /. sol2[[1]];

(*the following equation is equivalent to Equation 5-5 where F=F_o and G=G_p*)
g = feedflow-fint[totalarea];
```

```
x5int = 1-x1int[totalarea]-x2int[totalarea]-
x3int[totalarea]-x4int[totalarea];

(*The following equations are equivalent to Equation 5-15*)
y1perm =
(feedflow*x1i-fint[totalarea]*x1int[totalarea])/g;
y2perm =
(feedflow*x2i-fint[totalarea]*x2int[totalarea])/g;
y3perm =
(feedflow*x3i-fint[totalarea]*x3int[totalarea])/g;
y4perm =
(feedflow*x4i-fint[totalarea]*x4int[totalarea])/g;
y5perm =
(feedflow*x5i-fint[totalarea]*x5int)/g;


Print[" "];
Print["**************************************************"];
Print["SEPARATION CONDITIONS AND PERMEABILITIES"];
Print["feed pressure (psia)        = ", pfeed];
Print["feed side pressure drop (psi) = ", pdrop];
Print["permeate pressure (psia)     = ", pperm];
Print["membrane area (sqft)         = ", totalarea];
Print["membrane thickness (in)      = ", d];
Print["species 1 = ", species1];
Print["species 2 = ", species2];
Print["species 3 = ", species3];
Print["species 4 = ", species4];
Print["species 5 = ", species5];
Print["permeability of 1 (barrers) = ", permeability1];
Print["permeability of 2 (barrers) = ", permeability2];
Print["permeability of 3 (barrers) = ", permeability3];
Print["permeability of 4 (barrers) = ", permeability4];
```

```
Print["permeability of 5 (barrers) = ", permeability5];
Print["   "];
Print["FEED FLOW RATE AND MOLE FRACTIONS"];
Print["feed flow (scfm) = ", feedflow];
Print["x1i = ", x1i];
Print["x2i = ", x2i];
Print["x3i = ", x3i];
Print["x4i = ", x4i];
Print["x5i = ", x5i];
Print[" "];
Print["RETENTATE FLOW RATE AND MOLE FRACTIONS"];
Print["retentate flow (scfm) = ", fint[totalarea]];
Print["% of feed flow = ", fint[totalarea]/feedflow*100, "%"];
Print["x1 = ", x1int[totalarea]];
Print["x2 = ", x2int[totalarea]];
Print["x3 = ", x3int[totalarea]];
Print["x4 = ", x4int[totalarea]];
Print["x5 = ", 1-x1int[totalarea]-x2int[totalarea]-
x3int[totalarea]-x4int[totalarea]];
Print[" "];
Print["PERMEATE FLOW RATE AND MOLE FRACTIONS"];
Print["permeate flow (scfm) = ", g];
Print["% of feed flow = ", g/feedflow*100, "%"];
Print["y1 = ", y1perm];
Print["y2 = ", y2perm];
Print["y3 = ", y3perm];
Print["y4 = ", y4perm];
Print["y5 = ", y5perm];
Print[" "];
Print["SEPARATION RESULTS"];
Print["SO2 % Separation = ",
(feedflow*x4i-x4int[totalarea]*fint[totalarea])/
(feedflow*x4i)*100];
```

```
Print["NO2 % Separation = ",
(feedflow*x5i-x5int*fint[totalarea])/
(feedflow*x5i)*100];
Print["Permeate % of feed flow = ",(g/feedflow)*100];

Print["************************************************"];

(*end of function cross5*)
)
```

APPENDIX E - Compare Programs with Shindo's Results

This section is an attempt to validate the code of the co, counter, and cross programs by comparing the results provided by Shindo [38] (for 3 and 5 component gas separations) to the output of co3, co5, counter3, counter5, cross3, and cross5. The largest percentage difference for either co3 or co5 and the reported results is 0.4%. The following table and paragraphs provide the conversions from the units used by Shindo to the units used in this report.

Table D-1

Conversion of Permeabilities

Membrane Material	Species	mol/(s*m*Pa)	Barrer
Polyethylene at 50°C	NH_3	36.9E(-15)	110.0
	H_2	11.7E(-15)	35.0
	N_2	2.41E(-15)	7.18
Microporous Glass at 25°C	H_2	48.0E(-12)	143040
	CH_4	19.1E(-12)	56918
	CO	14.0E(-12)	41720
	N_2	13.8E(-12)	41124
	CO_2	14.8E(-12)	44104

Shindo demonstrates the differences between membrane flow patterns by choosing 2 separation scenarios. Since Shindo uses dimensionless units, he does not need to completely define the membrane system in order to calculate results. Instead, he chooses two dimensionless values, the ratio of permeate pressure to feed pressure and a value representing dimensionless membrane area (st).

$$st = \frac{(membrane\ area)(feed\ pressure)(highest\ permeability\ in\ separation)}{(feed\ flowrate)(membrane\ thickness)}$$

The first scenario is a three species separation of ammonia, hydrogen, and nitrogen using polyethylene with feed mole fractions of 0.45, 0.25, and 0.30 respectively. The pressure ratio is set to 0.13 and the dimensionless area is set to 1.0 (approximately 1.5 sqft for the second calculation). In order to define the same membrane system, the following values were used: feed pressure = 1.0 psia, permeate pressure = 0.13 psia, membrane area = 1 sqft (approximately 1.5 sqft for the second calculation), membrane thickness = 1 in., and feed flow rate = 4.416E(-8) scfm.

Table D-2

Comparison of co3 to Shindo

Perm conc.	co3	st = 1.0 Shindo	% difference	co3	st = 1.4963 Shindo	% difference
NH_3	0.7298	0.7300	0.03	0.6921	0.6923	0.03
H_2	0.2071	0.2067	0.19	0.2306	0.2303	0.13
N_2	0.0631	0.0632	0.16	0.0772	0.0774	0.26
Stage Cut	0.3699	0.3702	0.08	0.4998	0.5000	0.04
Perm conc.	cross3	st = 1.0 Shindo	% difference	cross3	st = 1.4740 Shindo	% difference
NH_3	0.7336	0.7338	0.03	0.7002	0.7003	0.01
H_2	0.2039	0.2035	0.20	0.2244	0.2241	0.13
N_2	0.0625	0.0627	0.32	0.0755	0.0756	0.13
Stage Cut	0.3723	0.3726	0.08	0.4998	0.5000	0.04
Perm conc.	counter3	st = 1.0 Shindo	% difference	counter3	st = 1.4603 Shindo	% difference
NH_3	0.7366	0.7368	0.03	0.7057	0.7054	0.04
H_2	0.2013	0.2010	0.15	0.2200	0.2200	0.00
N_2	0.0621	0.0622	0.16	0.0743	0.0746	0.40
Stage Cut	0.3741	0.3745	0.11	0.5002	0.5000	0.04

The second scenario is a five species separation of hydrogen, methane, carbon monoxide, nitrogen, and carbon dioxide using a microporous glass with feed mole fractions of 0.3, 0.1, 0.25, 0.15, and 0.2 respectively. The pressure ratio is set to 0.1 and the dimensionless area is set to 1.0 (approximately 1.5 sqft for the second calculation). In order to define the same membrane system, the following values were used: feed pressure = 1.0 psia, permeate pressure = 0.1 psia, membrane area = 1 sqft (approximately 1.2 sqft for the second calculation), membrane thickness = 1 in., and feed flow rate = 5.743E(-5) scfm.

Table D-3

Comparison of co5 with Shindo

Perm conc.	co5	st = 1.0 Shindo	% difference	co5	st = 1.244 Shindo	% difference
H_2	0.4665	0.4662	0.06	0.4444	0.4441	0.07
CH_4	0.0917	0.0917	0.00	0.0942	0.0942	0.00
CO	0.1819	0.1821	0.11	0.1902	0.1904	0.11
N_2	0.1079	0.1081	0.19	0.1129	0.1131	0.18
CO_2	0.1519	0.1518	0.07	0.1584	0.1582	0.13
Stage Cut	0.4104	0.4112	0.19	0.4991	0.5000	0.18
Perm conc.	cross5	st = 1.0 Shindo	% difference	cross5	st = 1.236 Shindo	% difference
H_2	0.4710	0.4707	0.06	0.4505	0.4502	0.07
CH_4	0.0910	0.0910	0.00	0.0933	0.0933	0.00
CO	0.1804	0.1806	0.11	0.1880	0.1882	0.11
N_2	0.1070	0.1072	0.19	0.1116	0.1118	0.18
CO_2	0.1506	0.1505	0.07	0.1566	0.1565	0.06
Stage Cut	0.4123	0.4131	0.19	0.4989	0.5000	0.22
Perm conc.	counter5	st = 1.0 Shindo	% difference	conter5	st = 1.231 Shindo	% difference
H_2	0.4744	0.4742	0.04	0.4548	0.4544	0.09
CH_4	0.0904	0.0905	0.11	0.0926	0.0927	0.11
CO	0.1791	0.1793	0.11	0.1865	0.1867	0.11
N_2	0.1063	0.1065	0.19	0.1107	0.1109	0.18
CO_2	0.1498	0.1495	0.20	0.1554	0.1553	0.06
Stage Cut	0.4138	0.4146	0.19	0.4991	0.5000	0.18

APPENDIX F – Plotting Values as a Function of Membrane Area

The following example shows how different values (retentate and permeate flow rate, retentate and permeate molar flow rates, retentate and permeate mole fractions, and partial pressure differences) can be plotted in Mathematica as a function of membrane area. The example is for a PDMS membrane with a countercurrent flow pattern under the following conditions: 82 psia feed pressure, 12 psia permeate pressure, 5 scfm feed flow rate, 100 sqft of membrane area, and the following feed mole fractions: N_2=0.8, O_2=0.0746, CO_2=0.12, SO_2=0.005, and NO_2=0.0004.

The example is divided into two parts. The first part is the input and output of the function "counter5". The input of the second part instructs Mathematica to create the plots explained above, and the plots comrise the output of the second part.

Part 1 - input

counter5[{82,0,12,5,100}]

Part 1 - output

iterations = 67

SEPARATION CONDITIONS AND PERMEABILITIES

feed pressure (psia) = 82

feed side pressure drop (psia) = 0

permeate pressure (psia) = 12

membrane area (sqft) = 100

membrane thickness (in) = 0.000984

species 1 = nitrogen

species 2 = dioxide sulfur

species 3 = dioxide nitrogen

species 4 = carbon dioxide

species 5 = oxygen

species 1 permeability (barrers) = 250

species 2 permeability (barrers) = 12500

species 3 permeability (barrers) = 6350

species 4 permeability (barrers) = 2700

species 5 permeability (barrers) = 500

ACTUAL VALUES FOR FEED FLOW RATE AND MOLE FRACTIONS"

feed flow (stp) = 5

x1i actual = 0.8

x2i actual = 0.005

x3i actual = 0.0004

x4i actual = 0.12

x5i actual = 0.0746

CALCULATED VALUES FOR FEED FLOW RATE AND MOLE FRACTIONS"

feed flow (stp) = 4.99998`

x1i calc. = 0.800003

x2i calc. = 0.00500494

x3i calc. = 0.000399997

x4i calc. = 0.12

x5i calc. = 0.0745922

ENTERED STAGE CUT AND RETENTATE MOLE FRACTIONS

guessed permeate flow (stp) = 1.13665504084895441

retentate flow (stp) = 3.86334495915104536`

% of feed flow = 77.2669%

x1o = 0.873128

x2o = 0.000598918

x3o =0.0000973152

x4o = 0.0553665

x5o = 0.0708091

PERMEATE FLOW RATE AND MOLE FRACTIONS

calculated permeate flow (stp) = 1.13664

% of feed flow = 22.7327%

y1 = 0.551452

y2 = 0.0199805

y3 = 0.00142879

y4 = 0.339685

y5 = 0.0874535

SEPARATION RESULTS

SO2 % Separation = 90.7447

NO2 % Separation = 81.2019

Permeate % of feed flow = 22.7327

****

Part 2 - input

```
pfeed = 82;
pdrop = 0;
pperm = 12;
feedflow = 5;
totalarea = 100;


(*Plot retentate and permeate side flow rates as a function
of membrane area*)
Plot[{fint[totalarea-area],fint[totalarea-area]-fint[0]},
        {area,0,totalarea},Frame->True, AxesOrigin->None,
        PlotLabel->"Permeate and Retentate Flow Rate (scfm) vs. Area (sqft)"];


(*Compare flow rate of each species on the
retentate and permeate side*)
```

```
Plot[{x1int[totalarea-area]*fint[totalarea-area],
        (x1int[totalarea-area]*fint[totalarea-area]-
        x1o*(feedflow-permflow))},
        {area,0,totalarea},Frame->True, AxesOrigin->None,
        PlotLabel->"Nitrogen Permeate and Retentate Flow Rate (scfm) vs. Area (sqft)"];

Plot[{x2int[totalarea-area]*fint[totalarea-area],
        (x2int[totalarea-area]*fint[totalarea-area]-
        x2o*(feedflow-permflow))},
        {area,0,totalarea},Frame->True, AxesOrigin->None,
        PlotLabel->"Sulfur Dioxide Permeate and Retentate Flow Rate (scfm) vs. Area (sqft)"];

Plot[{x3int[totalarea-area]*fint[totalarea-area],
        (x3int[totalarea-area]*fint[totalarea-area]-
        x3o*(feedflow-permflow))},
        {area,0,totalarea},Frame->True, AxesOrigin->None,
        PlotLabel->"Nitrogen Dioxide Permeate and Retentate Flow Rate (scfm) vs. Area (sqft)"];

Plot[{x4int[totalarea-area]*fint[totalarea-area],
        (x4int[totalarea-area]*fint[totalarea-area]-
        x4o*(feedflow-permflow))},
        {area,0,totalarea},Frame->True, AxesOrigin->None,
        PlotLabel->"Carbon Dioxide Permeate and Retentate Flow Rate (scfm) vs. Area (sqft)"];

Plot[{(1-x1int[totalarea-area]-x2int[totalarea-area]-
        x3int[totalarea-area]-x4int[totalarea-area])*
        fint[totalarea-area],
        (1-x1int[totalarea-area]-x2int[totalarea-area]-
        x3int[totalarea-area]-x4int[totalarea-area])*
        fint[totalarea-area]-x5o*(feedflow-permflow)},
        {area,0,totalarea},Frame->True, AxesOrigin->None,
        PlotLabel->"Oxygen Permeate and Retentate Flow Rate (scfm) vs. Area (sqft)"];
```

```
(*Plot retentate mole fractions as a function of membrane area*)
Plot[{x1int[totalarea-area],x2int[totalarea-area],
        1-x1int[totalarea-area]-x2int[totalarea-area]},
        {area,0,totalarea}, Frame->True, AxesOrigin->None,
        PlotLabel->"Retentate Mole Fractions vs. Area (sqft)"];

Plot[x1int[totalarea-area], {area,0,totalarea},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Nitrogen Retentate Mole Fraction vs. Area (sqft)"];

Plot[x2int[totalarea-area], {area,0,totalarea},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Sulfur Dioxide Retentate Mole Fraction vs. Area (sqft)"];

Plot[x3int[totalarea-area],{area,0,totalarea},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Nitrogen Dioxide Retentate Mole Fraction vs. Area (sqft)"];

Plot[x4int[totalarea-area],{area,0,totalarea},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Carbon Dioxide Retentate Mole Fraction vs. Area (sqft)"];

Plot[1-x1int[totalarea-area]-x2int[totalarea-area]-
        x3int[totalarea-area]-x4int[totalarea-area],
        {area,0,totalarea},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Oxygen Retentate Mole Fraction vs. Area (sqft)"];



(*Plot permeate mole fractions as a function of membrane area*)
```

```
Plot[{
        (x1int[totalarea-area]*fint[totalarea-area]-
        x1o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        (x2int[totalarea-area]*fint[totalarea-area]-
        x2o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        (x3int[totalarea-area]*fint[totalarea-area]-
        x3o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        (x4int[totalarea-area]*fint[totalarea-area]-
        x4o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        (1-(x1int[totalarea-area]*fint[totalarea-area]-
        x1o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow))-
        (x2int[totalarea-area]*fint[totalarea-area]-
        x2o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow))-
        (x3int[totalarea-area]*fint[totalarea-area]-
        x3o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow))-
        (x4int[totalarea-area]*fint[totalarea-area]-
        x4o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)))},
        {area,0,totalarea-2}, Frame->True, AxesOrigin->None,
        PlotLabel->"Permeate Mole Fractions vs. Area (sqft)",
        PlotRange->All];

Plot[(x1int[totalarea-area]*fint[totalarea-area]-
        x1o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        {area,0,totalarea-2},
```

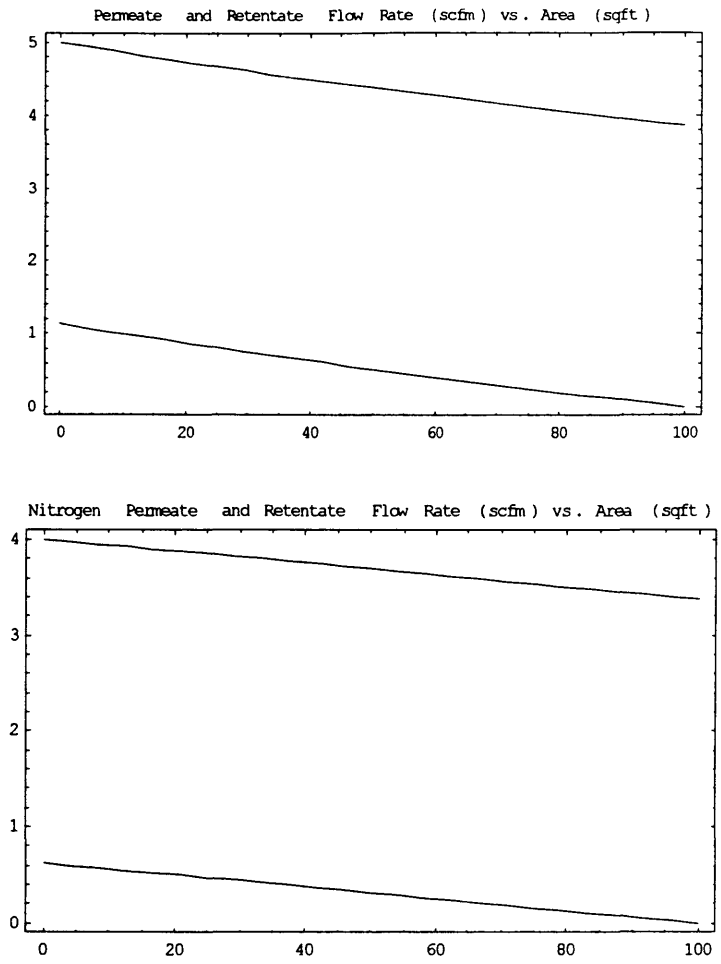
```
        Frame->True, AxesOrigin->None,
        PlotLabel->"Nitrogen Permeate Mole Fraction vs. Area (sqft)",
        PlotRange->All];

Plot[(x2int[totalarea-area]*fint[totalarea-area]-
        x2o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        {area,0,totalarea-2},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Sulfur Dioxide Permeate Mole Fraction vs. Area (sqft)",
        PlotRange->All];

Plot[(x3int[totalarea-area]*fint[totalarea-area]-
        x3o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        {area,0,totalarea-2},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Nitrogen Dioxide Permeate Mole Fraction vs. Area (sqft)",
        PlotRange->All];

Plot[(x4int[totalarea-area]*fint[totalarea-area]-
        x4o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        {area,0,totalarea-2},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Carbon Dioxide Permeate Mole Fraction vs. Area (sqft)",
        PlotRange->All];

Plot[(1-(x1int[totalarea-area]*fint[totalarea-area]-
        x1o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow))-
        (x2int[totalarea-area]*fint[totalarea-area]-
        x2o*(feedflow-permflow))/(fint[totalarea-area]-
```

```
        (feedflow-permflow))-
        (x3int[totalarea-area]*fint[totalarea-area]-
        x3o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow))-
        (x4int[totalarea-area]*fint[totalarea-area]-
        x4o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow))),
        {area,0,totalarea-2},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Oxygen Permeate Mole Fraction vs. Area (sqft)",
        PlotRange->All];


(*Plot driving force as a function of membrane area*)
(*Driving force = pfeed*x - pperm*y*)
Plot[pperm/gamma1[area]*x1int[totalarea-area]-pperm*
        (x1int[totalarea-area]*fint[totalarea-area]-
        x1o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        {area,0,totalarea-2},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Nitrogen Partial Pressure Difference (psi) vs. Area (sqft)"];

Plot[pperm/gamma1[area]*x2int[totalarea-area]-pperm*
        (x2int[totalarea-area]*fint[totalarea-area]-
        x2o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        {area,0,totalarea-2},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Sulfur Dioxide Partial Pressure Difference (psi) vs. Area (sqft)"];

Plot[pperm/gamma1[area]*x3int[totalarea-area]-pperm*
        (x3int[totalarea-area]*fint[totalarea-area]-
```

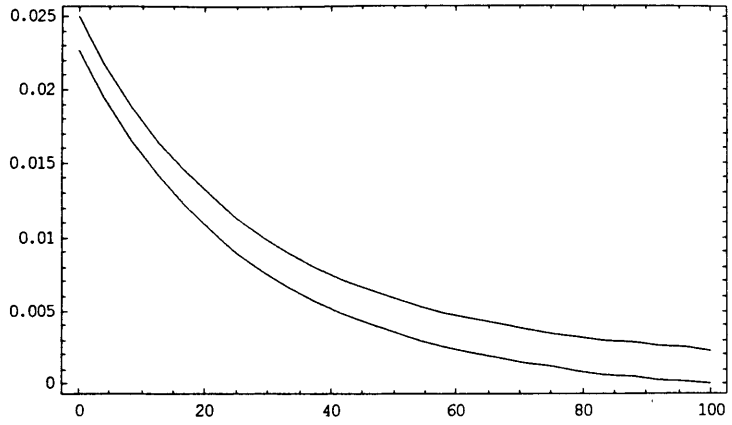
```
        x3o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        {area,0,totalarea-2},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Nitrogen Dioxide Partial Pressure Difference (psi) vs. Area (sqft)"];

Plot[pperm/gamma1[area]*x4int[totalarea-area]-pperm*
        (x4int[totalarea-area]*fint[totalarea-area]-
        x4o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow)),
        {area,0,totalarea-2},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Carbon Dioxide Partial Pressure Difference (psi) vs. Area (sqft)"];

Plot[pperm/gamma1[area]*(1-x1int[totalarea-area]-
        x2int[totalarea-area]-
        x3int[totalarea-area]-x4int[totalarea-area])-
        pperm*(1-(x1int[totalarea-area]*fint[totalarea-area]-
        x1o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow))-
        (x2int[totalarea-area]*fint[totalarea-area]-
        x2o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow))-
        (x3int[totalarea-area]*fint[totalarea-area]-
        x3o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow))-
        (x4int[totalarea-area]*fint[totalarea-area]-
        x4o*(feedflow-permflow))/(fint[totalarea-area]-
        (feedflow-permflow))),
        {area,0,totalarea-2},
        Frame->True, AxesOrigin->None,
        PlotLabel->"Oxygen Partial Pressure Difference (psi) vs. Area (sqft)"];
```

Figure F-1

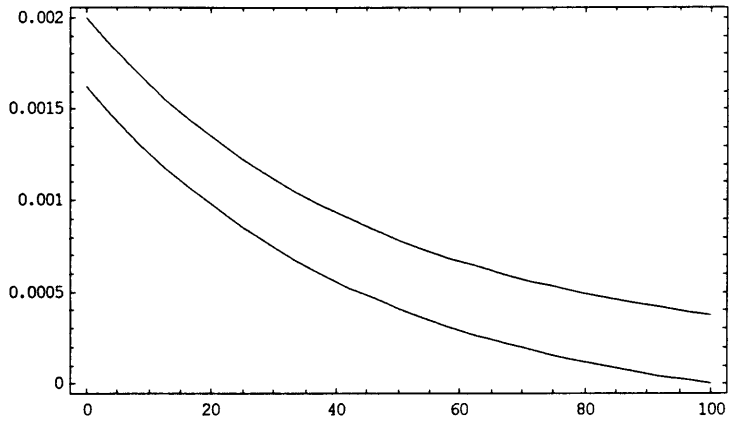
Part 2 – output, Plotting Values as a Function of Membrane Area



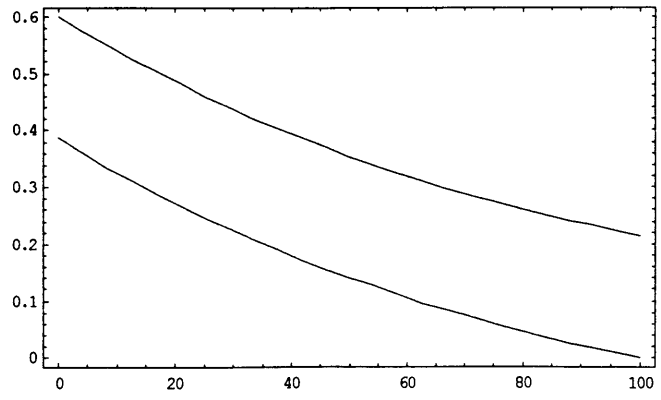
Sulfur Dioxide Permeate and Retentate Flow Rate (scfm)
vs. Area (sqft)

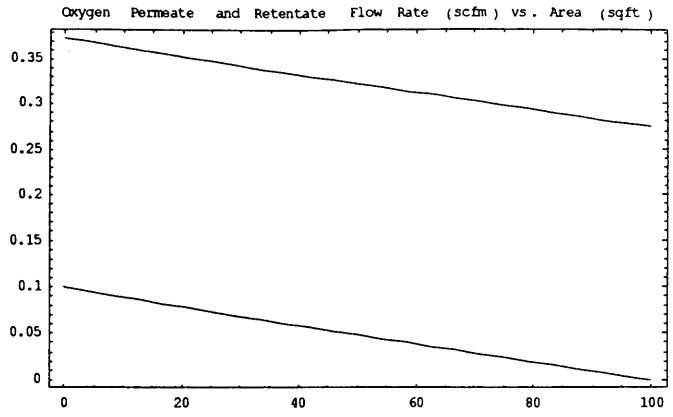


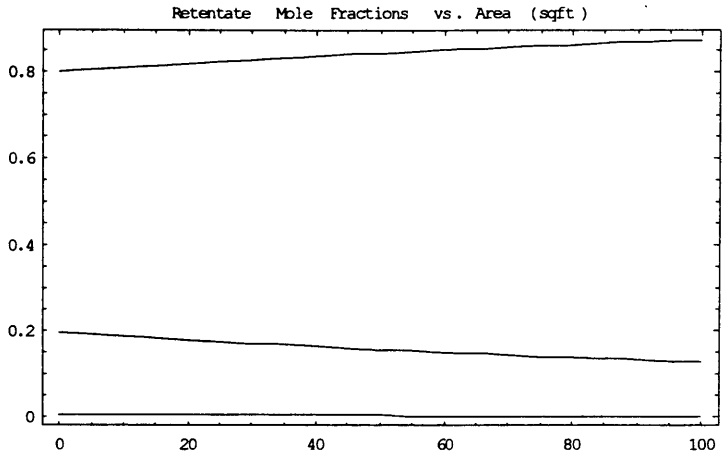
Nitrogen Dioxide Permeate and Retentate Flow Rate (scfm)
vs. Area (sqft)

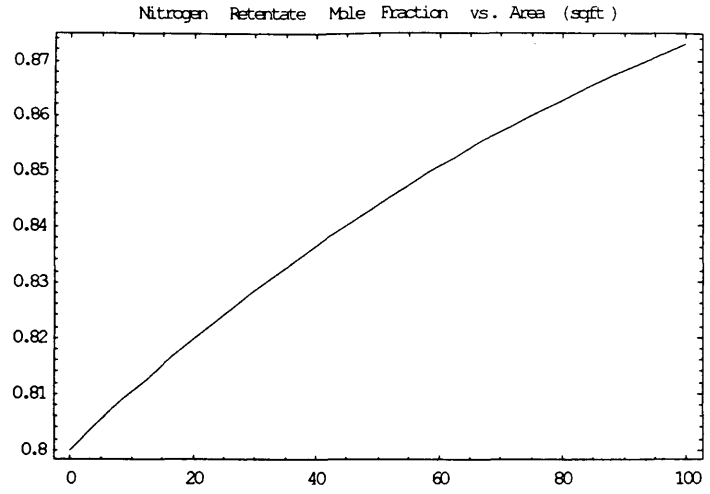


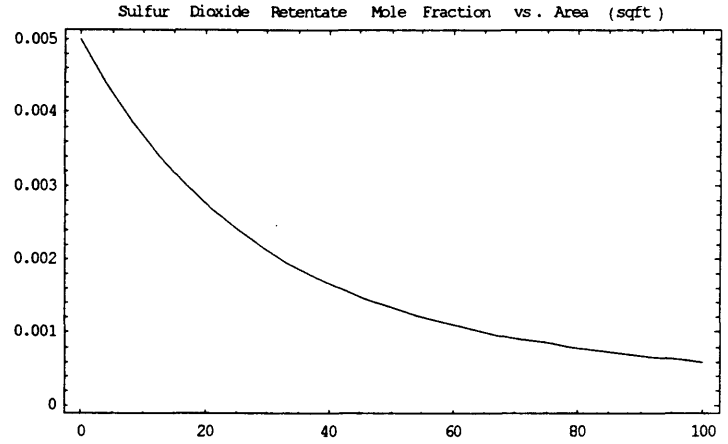
Carbon Dioxide Permeate and Retentate Flow Rate (scfm)
vs. Area (sqft)

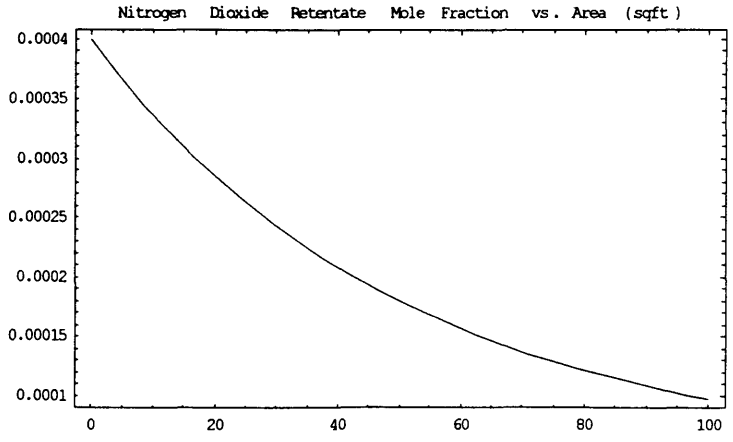


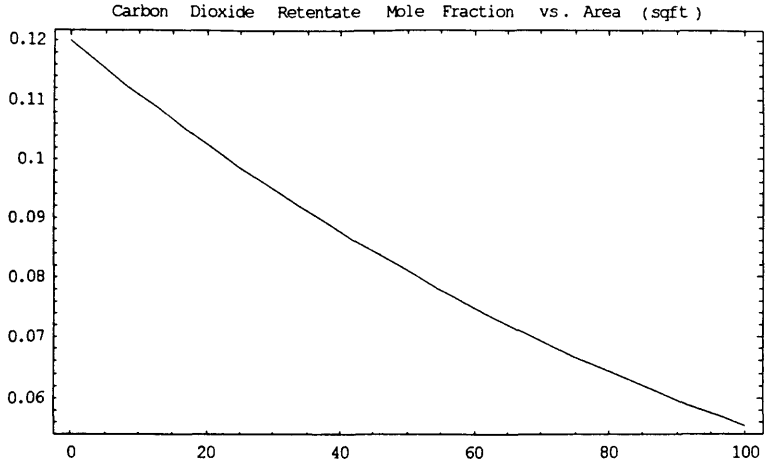

Oxygen Permeate and Retentate Flow Rate (scfm) vs. Area (sqft)
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
20
40
60
80
100

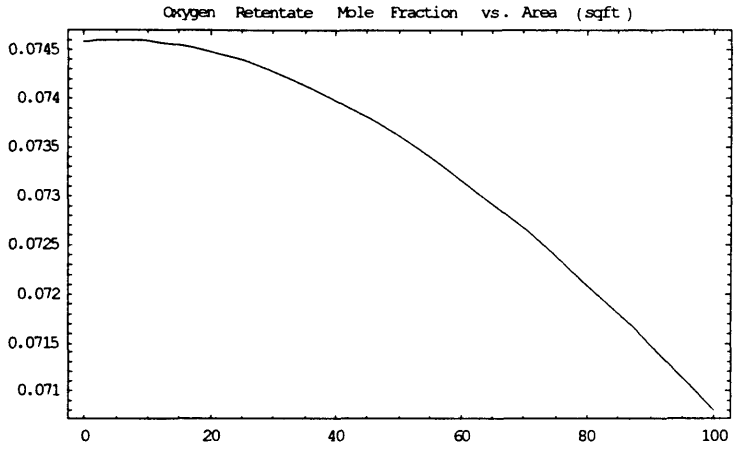

Retentate Mole Fractions vs. Area (sqft)
0.8
0.6
0.4
0.2
0
0
20
40
60
80
100

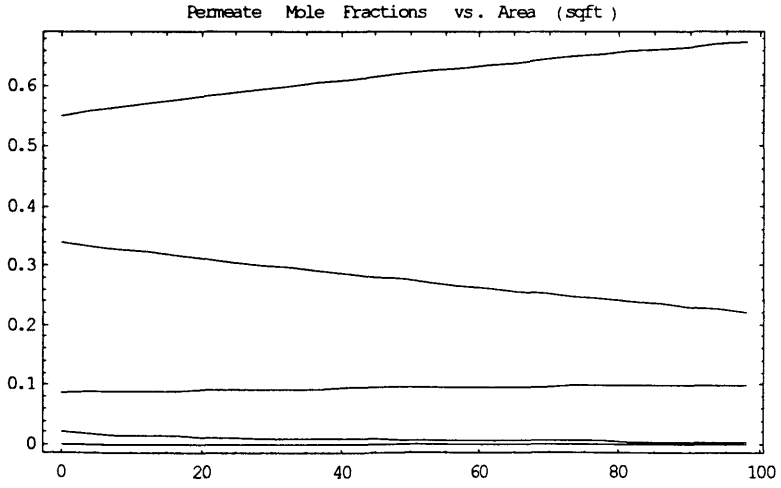

Nitrogen Retentate Mole Fraction vs. Area (sqft)
0.87
0.86
0.85
0.84
0.83
0.82
0.81
0.8
0
20
40
60
80
100

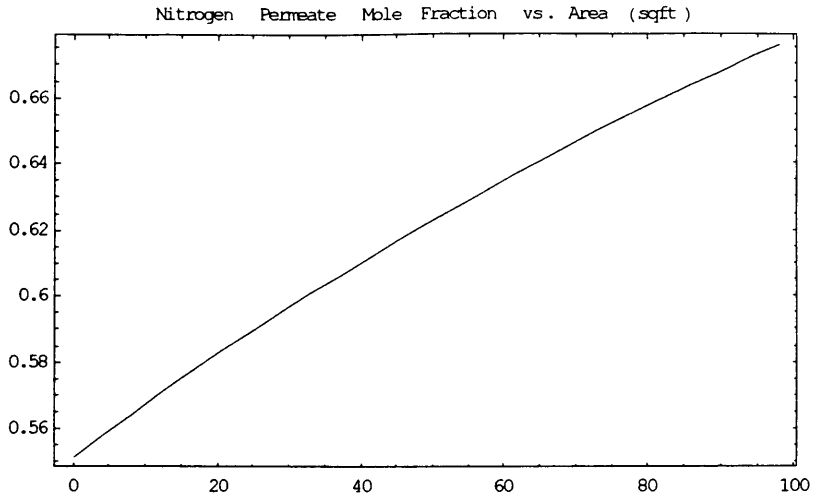

Sulfur Dioxide Retentate Mole Fraction vs. Area (sqft)
0.005
0.004
0.003
0.002
0.001
0
0
20
40
60
80
100

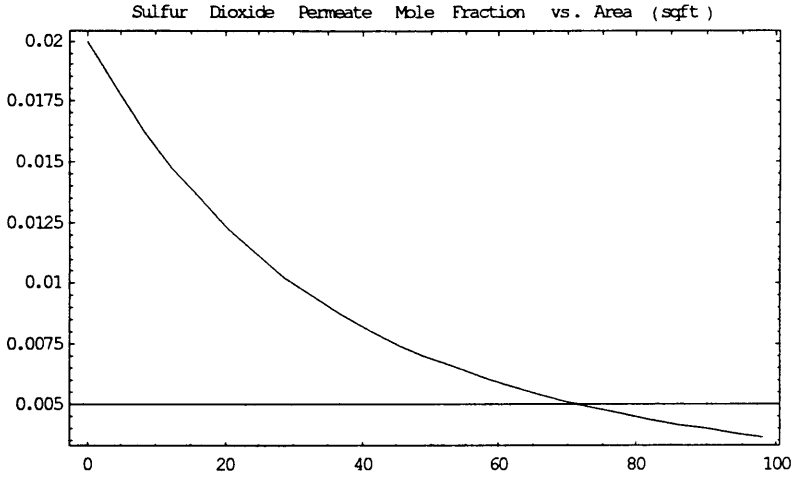

Nitrogen Dioxide Retentate Mole Fraction vs. Area (sqft)
0.0004
0.00035
0.0003
0.00025
0.0002
0.00015
0.0001
0
20
40
60
80
100

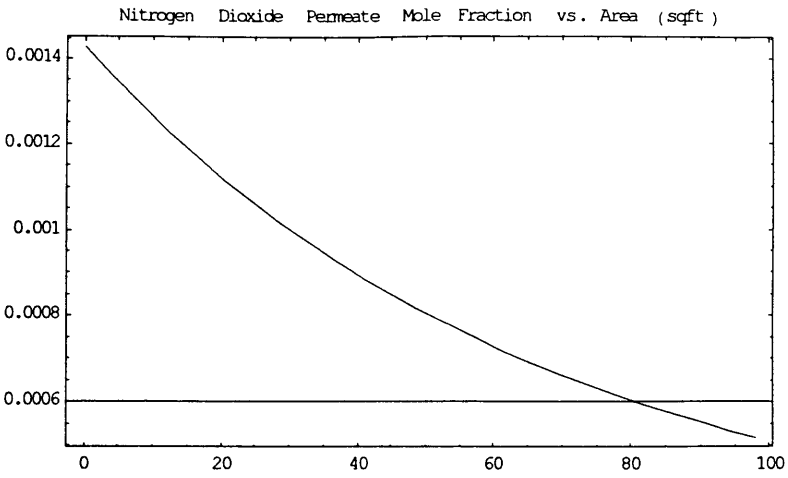


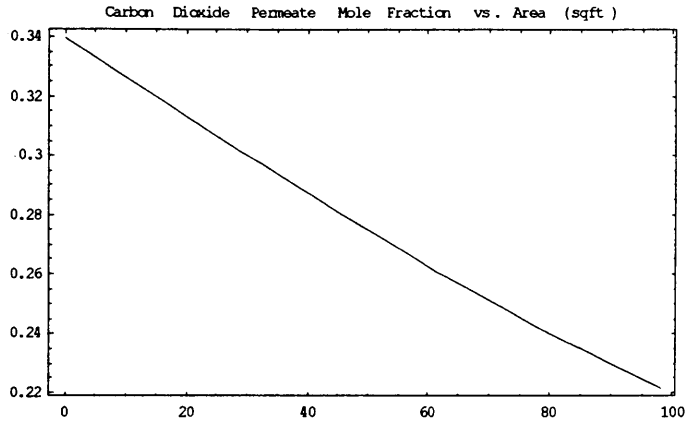


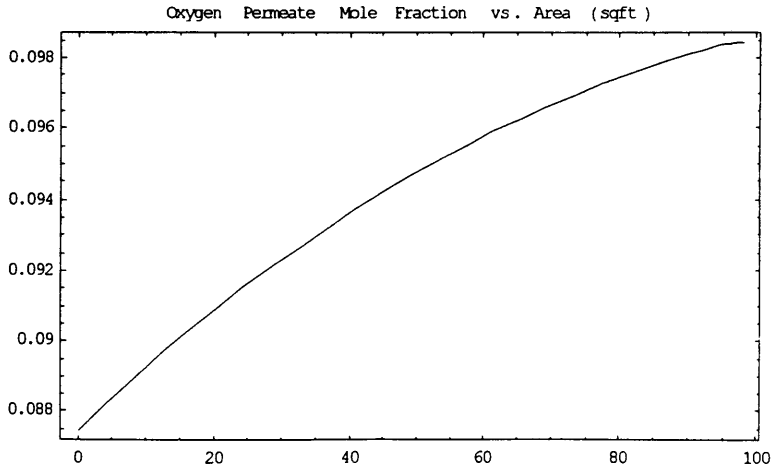


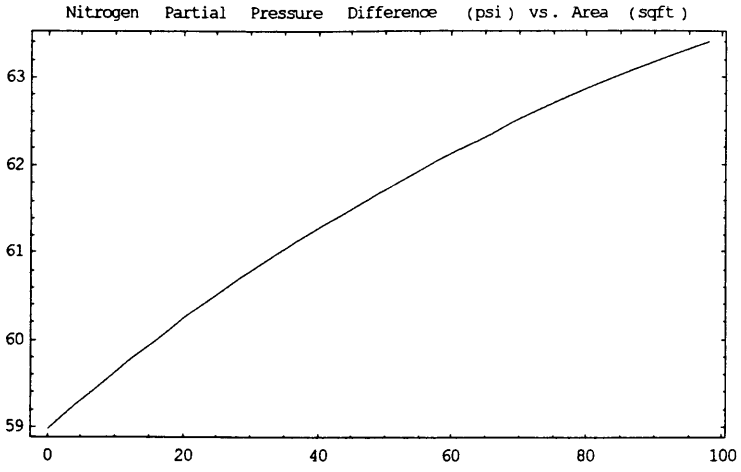

Nitrogen Permeate Mole Fraction vs. Area (sqft)
0.66
0.64
0.62
0.6
0.58
0.56
0
20
40
60
80
100

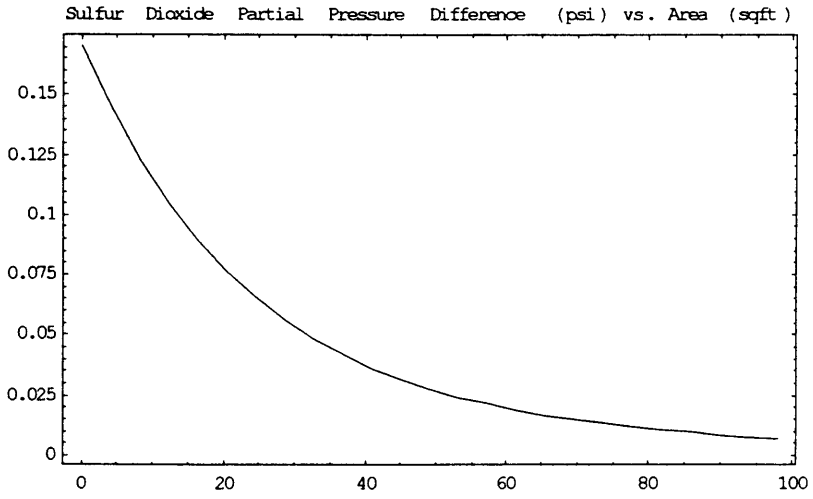

Sulfur Dioxide Permeate Mole Fraction vs. Area (sqft)
0.02
0.0175
0.015
0.0125
0.01
0.0075
0.005
0
20
40
60
80
100

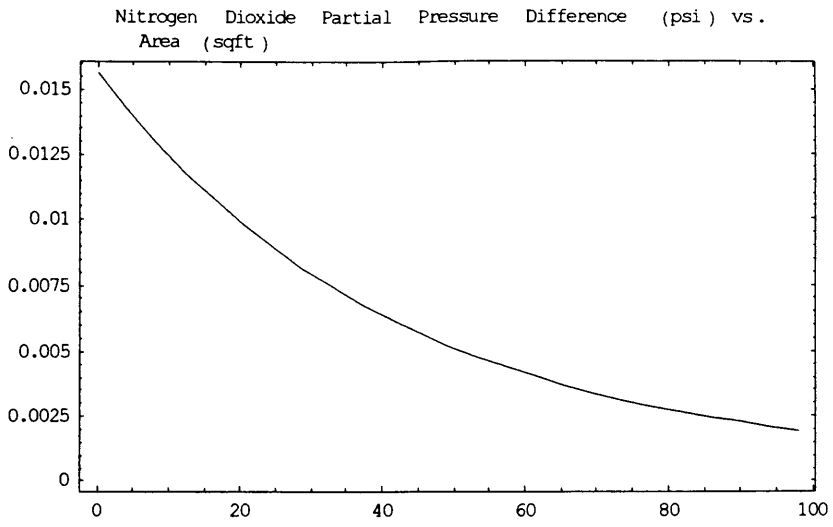

Nitrogen Dioxide Permeate Mole Fraction vs. Area (sqft)
0.0014
0.0012
0.001
0.0008
0.0006
0
20
40
60
80
100

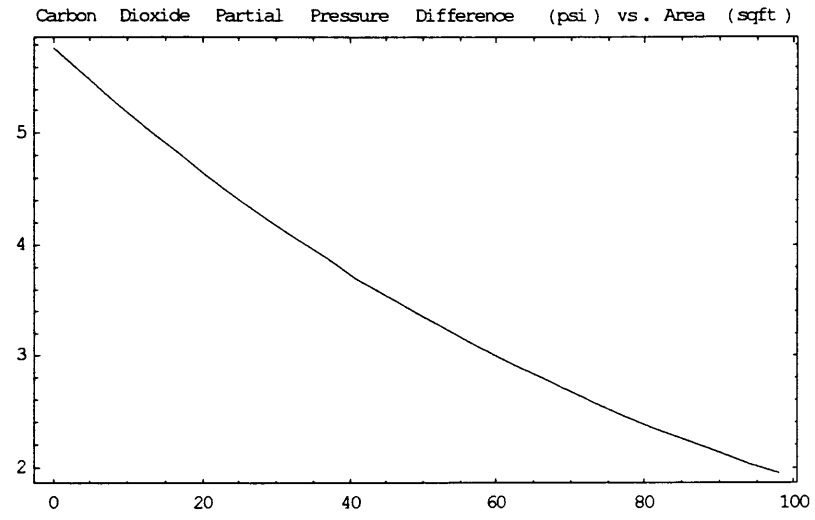

Carbon Dioxide Permeate Mole Fraction vs. Area (sqft)
0.34
0.32
0.3
0.28
0.26
0.24
0.22
0
20
40
60
80
100

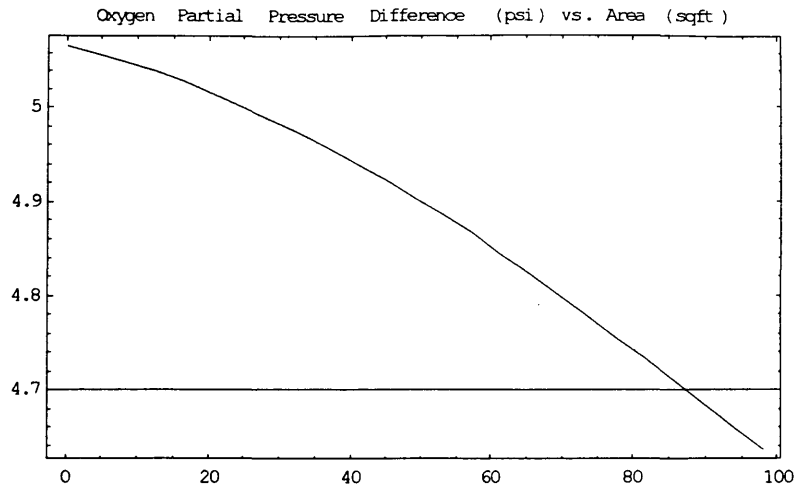

Oxygen Permeate Mole Fraction vs. Area (sqft)
0.098
0.096
0.094
0.092
0.09
0.088
0
20
40
60
80
100


Nitrogen Partial Pressure Difference (psi) vs. Area (sqft)
63
62
61
60
59
0
20
40
60
80
100


Sulfur Dioxide Partial Pressure Difference (psi) vs. Area (sqft)
0.15
0.125
0.1
0.075
0.05
0.025
0
0
20
40
60
80
100


Nitrogen Dioxide Partial Pressure Difference (psi) vs. Area (sqft)
0.015
0.0125
0.01
0.0075
0.005
0.0025
0
0
20
40
60
80
100


Carbon Dioxide Partial Pressure Difference (psi) vs. Area (sqft)
5
4
3
2
0
20
40
60
80
100


Oxygen Partial Pressure Difference (psi) vs. Area (sqft)
5
4.9
4.8
4.7
0
20
40
60
80
100

APPENDIX G – Finding Membrane Area for a Specific Separation

This appendix contains three examples of finding the required membrane area for a specific separation. For each example: the membrane area is over estimated, the program "co5" is run, the necessary membrane area is found, and the program "co5" is run a second time. The output of each "co5" run is produced. The second output contains the necessary membrane area and results. Each example is for the room temperature separation of N_2, O_2, CO_2, NO_2, and SO_2 with a PDMS membrane for a 25 scfm feed flow rate, 82 psia feed pressure, 12 psia permeate pressure, and the following mole fractions: N_2=0.8, O_2=0.0746, CO_2=0.12, SO_2=0.005, and NO_2=0.0004.

Example 1

Demonstration of finding the required membrane area for a specified stage-cut. Find the membrane area required to produce a 35% stage cut.

The required area is approximately 873 sqft.

Input – Example 1

```
stagecut = 0.35;
feedflow = 25;
totalarea = 1000;
co5[{82,0,12,feedflow,totalarea}]
sol4 = FindMinimum[
        Abs[stagecut-(feedflow*fint[area])/feedflow],
        {area,100,500}];
areafound = area /. sol4[[2]];
co5[{82,0,12,feedflow,areafound}]
```

Output – Example 1

****

SEPARATION CONDITIONS AND PERMEABILITIES

feed pressure (psia) = 82

feed side pressure drop (psi) = 0

permeate pressure (psia) = 12

membrane area (sqft) = 1000

membrane thickness (in) = 0.000984

species 1 = nitrogen

species 2 = oxygen

species 3 = carbon dioxide

species 4 = dioxide sulfur

species 5 = dioxide nitrogen

permeability of 1 (barrers) = 250

permeability of 2 (barrers) = 500

permeability of 3 (barrers) = 2700

permeability of 4 (barrers) = 12500

permeability of 5 (barrers) = 6350

FEED FLOW RATE AND MOLE FRACTIONS

feed flow (scfm) = 25

x1i = 0.8

x2i = 0.0746

x3i = 0.12

x4i = 0.005

x5i = 0.0004

RETENTATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 60.7313%

retentate flow (scfm) = 15.1828

x1 = 0.892942

x2 = 0.061358

x3 = 0.0439962

x4 = 0.0015733

x5 = 0.000130506

PERMEATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 39.2687%

permeate flow (scfm) = 9.81717

y1 = 0.65626

y2 = 0.0950796

y3 = 0.237544

y4 = 0.0102996

y5 = 0.000816789

SEPARATION RESULTS

SO2 % Separation = 80.8903

NO2 % Separation = 80.1856

Permeate % of feed flow = 39.2687

****

****

SEPARATION CONDITIONS AND PERMEABILITIES

feed pressure (psia) = 82

feed side pressure drop (psi) = 0

permeate pressure (psia) = 12

membrane area (sqft) = 872.948

membrane thickness (in) = 0.000984

species 1 = nitrogen

species 2 = oxygen

species 3 = carbon dioxide

species 4 = dioxide sulfur

species 5 = dioxide nitrogen

permeability of 1 (barrers) = 250

permeability of 2 (barrers) = 500

permeability of 3 (barrers) = 2700

permeability of 4 (barrers) = 12500

permeability of 5 (barrers) = 6350

FEED FLOW RATE AND MOLE FRACTIONS

feed flow (scfm) = 25

x1i = 0.8

x2i = 0.0746

x3i = 0.12

x4i = 0.005

x5i = 0.0004

RETENTATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 65.0002%

retentate flow (scfm) = 16.2501

x1 = 0.886182

x2 = 0.0636764

x3 = 0.0482933

x4 = 0.00170628

x5 = 0.000141933

PERMEATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 34.9998%

permeate flow (scfm) = 8.74994

y1 = 0.639946

y2 = 0.0948869

y3 = 0.253171

y4 = 0.011117

y5 = 0.000879272

SEPARATION RESULTS

SO2 % Separation = 77.8183

NO2 % Separation = 76.9358

Permeate % of feed flow = 34.9998

****

Example 2

Demonstration of finding the required membrane area for a specified retentate concentration.

Find the required membrane area to obtain 0.002 mole fraction of sulfur dioxide from 0.005 mole fraction.

The required membrane area is approximately 660 sqft.

Input – Example 2

```
x4retentate = 0.002;
feedflow = 25;
totalarea = 1000;
co5[{82,0,12,feedflow,totalarea}]
sol5 = FindMinimum[Abs[x4retentate-x4int[area]],
        {area,20,25}];
areafound = area /. sol5[[2]];
co5[{82,0,12,feedflow,areafound}]
```

Output – Example 2

```
**************************************************
SEPARATION CONDITIONS AND PERMEABILITIES
feed pressure (psia)        = 82
feed side pressure drop (psi) = 0
permeate pressure (psia)    = 12
membrane area (sqft)        = 1000
membrane thickness (in)     = 0.000984
species 1 = nitrogen
species 2 = oxygen
species 3 = carbon dioxide
species 4 = dioxide sulfur
species 5 = dioxide nitrogen
permeability of 1 (barrers) = 250
permeability of 2 (barrers) = 500
```

permeability of 3 (barrers) = 2700

permeability of 4 (barrers) = 12500

permeability of 5 (barrers) = 6350

FEED FLOW RATE AND MOLE FRACTIONS

feed flow (scfm) = 25

x1i = 0.8

x2i = 0.0746

x3i = 0.12

x4i = 0.005

x5i = 0.0004

RETENTATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 60.7313%

retentate flow (scfm) = 15.1828

x1 = 0.892942

x2 = 0.061358

x3 = 0.0439962

x4 = 0.0015733

x5 = 0.000130506

PERMEATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 39.2687%

permeate flow (scfm) = 9.81717

y1 = 0.65626

y2 = 0.0950796

y3 = 0.237544

y4 = 0.0102996

y5 = 0.000816789

SEPARATION RESULTS

SO2 % Separation = 80.8903

NO2 % Separation = 80.1856

Permeate % of feed flow = 39.2687

****

****

SEPARATION CONDITIONS AND PERMEABILITIES

feed pressure (psia) = 82

feed side pressure drop (psi) = 0

permeate pressure (psia) = 12

membrane area (sqft) = 659.514

membrane thickness (in) = 0.000984

species 1 = nitrogen

species 2 = oxygen

species 3 = carbon dioxide

species 4 = dioxide sulfur

species 5 = dioxide nitrogen

permeability of 1 (barrers) = 250

permeability of 2 (barrers) = 500

permeability of 3 (barrers) = 2700

permeability of 4 (barrers) = 12500

permeability of 5 (barrers) = 6350

FEED FLOW RATE AND MOLE FRACTIONS

feed flow (scfm) = 25

x1i = 0.8

x2i = 0.0746

x3i = 0.12

x4i = 0.005

x5i = 0.0004

RETENTATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 72.4346%

retentate flow (scfm) = 18.1087

x1 = 0.872806

x2 = 0.0673189

x3 = 0.0577081

x4 = 0.00199987

x5 = 0.000167294

PERMEATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 27.5654%

permeate flow (scfm) = 6.89134

y1 = 0.608685

y2 = 0.0937328

y3 = 0.283687

y4 = 0.0128836

y5 = 0.00101149

SEPARATION RESULTS

SO2 % Separation = 71.0281

NO2 % Separation = 69.7053

Permeate % of feed flow = 27.5654

Example 3

Demonstration of finding the required membrane area for a specified percentage separation. Find the required membrane area to remove 50% of the sulfur dioxide from the feed.

The required membrane area is approximately 303 sqft.

Input – Example 3

```
x4percentremoved = 50;
feedflow = 25;
totalarea = 1000;
co5[{82,0,12,feedflow,totalarea}]
sol6 = FindMinimum[
        Abs[x4percentremoved-
        (feedflow*x4i-x4int[area]*fint[area])/
        (feedflow*x4i)*100],
        {area,100,500}];
areafound = area /. sol6[[2]];
co5[{82,0,12,feedflow,areafound}]
```

Output – Example 3

```
**************************************************
SEPARATION CONDITIONS AND PERMEABILITIES
feed pressure (psia)        = 82
feed side pressure drop (psi) = 0
permeate pressure (psia)    = 12
membrane area (sqft)        = 1000
membrane thickness (in)     = 0.000984
species 1 = nitrogen
species 2 = oxygen
species 3 = carbon dioxide
```

species 4 = dioxide sulfur

species 5 = dioxide nitrogen

permeability of 1 (barrers) = 250

permeability of 2 (barrers) = 500

permeability of 3 (barrers) = 2700

permeability of 4 (barrers) = 12500

permeability of 5 (barrers) = 6350

FEED FLOW RATE AND MOLE FRACTIONS

feed flow (scfm) = 25

x1i = 0.8

x2i = 0.0746

x3i = 0.12

x4i = 0.005

x5i = 0.0004

RETENTATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 60.7313%

retentate flow (scfm) = 15.1828

x1 = 0.892942

x2 = 0.061358

x3 = 0.0439962

x4 = 0.0015733

x5 = 0.000130506

PERMEATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 39.2687%

permeate flow (scfm) = 9.81717

y1 = 0.65626

y2 = 0.0950796

y3 = 0.237544

y4 = 0.0102996

y5 = 0.000816789

SEPARATION RESULTS

SO2 % Separation = 80.8903

NO2 % Separation = 80.1856

Permeate % of feed flow = 39.2687

****

****

SEPARATION CONDITIONS AND PERMEABILITIES

feed pressure (psia) = 82

feed side pressure drop (psi) = 0

permeate pressure (psia) = 12

membrane area (sqft) = 302.687

membrane thickness (in) = 0.000984

species 1 = nitrogen

species 2 = oxygen

species 3 = carbon dioxide

species 4 = dioxide sulfur

species 5 = dioxide nitrogen

permeability of 1 (barrers) = 250

permeability of 2 (barrers) = 500

permeability of 3 (barrers) = 2700

permeability of 4 (barrers) = 12500

permeability of 5 (barrers) = 6350

FEED FLOW RATE AND MOLE FRACTIONS

feed flow (scfm) = 25

x1i = 0.8

x2i = 0.0746

x3i = 0.12

x4i = 0.005

x5i = 0.0004

RETENTATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 86.0943%

retentate flow (scfm) = 21.5236

x1 = 0.841433

x2 = 0.0722905

x3 = 0.0831284

x4 = 0.00290377

x5 = 0.000244286

PERMEATE FLOW RATE AND MOLE FRACTIONS

% of feed flow = 13.9057%

permeate flow (scfm) = 3.47642

y1 = 0.543475

y2 = 0.0888986

y3 = 0.348284

y4 = 0.0179784

y5 = 0.00136408

SEPARATION RESULTS

SO2 % Separation = 50.0004

NO2 % Separation = 47.421

Permeate % of feed flow = 13.9057

****

APPENDIX H – Code for Three Stage Membrane System

```
(*The following program is for a three-stage membrane separation system. This program is set to use the
co5 function. The co5 function has the following input form:

co5[{totalarea, pfeed, pperm, feedflow, x1i, x2i, x3i, x4i, x5i}]

The co5 function has the followin output form:
{retentate flow, x1o,x2o,x3o,x4o,x5o,permeate flow,y1,y2,y3,y4,y5}
*)

co53s[area1_,area2_,feedflow_] := (

areagiven = 100;
area3 = areagiven-area1-area2;

If[(area1+area2)>95 || area3>area2, Goto[toomucharea]];

pfeed = 62;
pperm = 12;

(*Input the mole fractions of the feed stream*)
x1i = 0.8;

x3i = 0.12;
x4i = 0.001;
x5i = 0.0001;

x2i = 1-x1i-x3i-x4i-x5i;
```

```
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
(*premembrane stage calculation*)
stream1 = {area1,pfeed,pperm,feedflow,x1i,x2i,x3i,x4i,x5i};

output1 = co5[stream1];

(*stream2 is the retentate from the premembrane stage*)
stream2 =
{area2,pfeed,pperm,output1[[1]],output1[[2]],output1[[3]],output1[[4]],output1[[5]],output1[[6]]};

(*set stream 3 equal to stream 2 for the first iteration*)
stream3 = stream2;

iterations=0;

comparestream = {0,0,0,0,0,0,0,0,0};

(*@@@@@@@@@@@@@@@@@@WhileLoop@@@@@@@@@@@@@@@@@@*)
While[
Abs[(comparestream[[4]]-stream3[[4]])/stream3[[4]]] > 0.000001 ||
Abs[(comparestream[[5]]-stream3[[5]])/stream3[[5]]] > 0.000001 ||
Abs[(comparestream[[6]]-stream3[[6]])/stream3[[6]]] > 0.000001 ||
Abs[(comparestream[[7]]-stream3[[7]])/stream3[[7]]] > 0.000001 ||
Abs[(comparestream[[8]]-stream3[[8]])/stream3[[8]]] > 0.000001 ||
Abs[(comparestream[[9]]-stream3[[9]])/stream3[[9]]] > 0.000001,

iterations=iterations+1;

(*make sure you don't get caught in an endless loop*)
If[iterations>20,Goto[toomucharea]];

(*If the system has NOT converged then set stream3 = comparestream*)
```

```
If[comparestream != {0,0,0,0,0,0,0,0,0}, stream3 = comparestream];

(*purification stage calculation*)
output2 = co5[stream3];

(*stream5 is the permeate of the purification stage.
note: it is assumed to be repressurized*)
stream5 =
{area3,pfeed,pperm,output2[[7]],output2[[8]],output2[[9]],output2[[10]],output2[[11]],output2[[12]]};

(*recovery stage calculation*)
output3 = co5[stream5];

(*Check to see if an answer is converged upon or if you get gobbleegook*)
If[NumberQ[output3[[7]]],,Goto[toomucharea]];

(*combine stream6 and stream2 for the next value of stream3*)
comparestream = {area2,pfeed,pperm,output1[[1]]+output3[[1]],
(output1[[1]]*output1[[2]]+output3[[1]]*output3[[2]])/
(output1[[1]]+output3[[1]]),
(output1[[1]]*output1[[3]]+output3[[1]]*output3[[3]])/
(output1[[1]]+output3[[1]]),
(output1[[1]]*output1[[4]]+output3[[1]]*output3[[4]])/
(output1[[1]]+output3[[1]]),
(output1[[1]]*output1[[5]]+output3[[1]]*output3[[5]])/
(output1[[1]]+output3[[1]]),
(output1[[1]]*output1[[6]]+output3[[1]]*output3[[6]])/
(output1[[1]]+output3[[1]])};

(*end of While loop*)
];

(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
```

```
(*assign values for each stream*)

(*stream4 is the retentate of the purification stage and the retentate of the entire system*)
stream4 = {0,0,0,output2[[1]],output2[[2]],output2[[3]],output2[[4]],output2[[5]],output2[[6]]};

(*stream6 is the retentate of the recovery stage.
note: it is combined with stream2 in the purification stage.*)
stream6 =
{area2,pfeed,pperm,output3[[1]],output3[[2]],output3[[3]],output3[[4]],output3[[5]],output3[[6]]};

(*stream7 is the permeate of the recovery stage.
note: it will be combined with stream8 once the program converges.*)
stream7 = {0,0,0,output3[[7]],output3[[8]],output3[[9]],output3[[10]],output3[[11]],output3[[12]]};

(*stream8 is the permeate from the premembrane stage*)
stream8 = {0,0,0,output1[[7]],output1[[8]],output1[[9]],output1[[10]],output1[[11]],output1[[12]]};


(*stream9 is the overall permeate stream of the system and is a combination of stream7 and stream8*)
stream9 = {0,0,0,output1[[7]]+output3[[7]],
(output1[[7]]*output1[[8]]+output3[[7]]*output3[[8]])/
(output1[[7]]+output3[[7]]),
(output1[[7]]*output1[[9]]+output3[[7]]*output3[[9]])/
(output1[[7]]+output3[[7]]),
(output1[[7]]*output1[[10]]+output3[[7]]*output3[[10]])/
(output1[[7]]+output3[[7]]),
(output1[[7]]*output1[[11]]+output3[[7]]*output3[[11]])/
(output1[[7]]+output3[[7]]),
(output1[[7]]*output1[[12]]+output3[[7]]*output3[[12]])/
(output1[[7]]+output3[[7]])};
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)

(*Print Results*)
```

```
Print[" "];
Print["@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@"];
Print["SEPARATION CONDITIONS AND PERMEABILITIES"];
Print["feed pressure (psia)        = ", pfeed];
Print["feed side pressure drop (psi) = ", pdrop];
Print["permeate pressure (psia)     = ", pperm];
Print["module 1 membrane area (sqft) = ", area1];
Print["module 2 membrane area (sqft) = ", area2];
Print["module 3 membrane area (sqft) = ", area3];
Print["membrane thickness (in)      = ", d];
Print["permeability of 1 (barrers) = ", permeability1];
Print["permeability of 2 (barrers) = ", permeability2];
Print["permeability of 3 (barrers) = ", permeability3];
Print["permeability of 4 (barrers) = ", permeability4];
Print["permeability of 5 (barrers) = ", permeability5];
Print["   "];
Print["FEED FLOW RATE AND MOLE FRACTIONS"];
Print["feed flow (scfm) = ", feedflow];
Print["x1i = ", x1i];
Print["x2i = ", x2i];
Print["x3i = ", x3i];
Print["x4i = ", x4i];
Print["x5i = ", x5i];
Print[" "];
Print["RETENTATE FLOW RATE AND MOLE FRACTIONS"];
Print["% of feed flow = ", stream4[[4]]/feedflow*100, "%"];
Print["retentate flow (scfm) = ", stream4[[4]]];
Print["x1 = ", stream4[[5]]];
Print["x2 = ", stream4[[6]]];
Print["x3 = ", stream4[[7]]];
Print["x4 = ", stream4[[8]]];
Print["x5 = ", stream4[[9]]];
Print[" "];
```

```
Print["PERMEATE FLOW RATE AND MOLE FRACTIONS"];
Print["% of feed flow = ", stream9[[4]]/feedflow*100, "%"];
Print["permeate flow (scfm) = ", stream9[[4]]];
Print["y1 = ", stream9[[5]]];
Print["y2 = ", stream9[[6]]];
Print["y3 = ", stream9[[7]]];
Print["y4 = ", stream9[[8]]];
Print["y5 = ", stream9[[9]]];
Print[" "];
Print["SEPARATION RESULTS"];
Print["SO2 % Separation = ", (feedflow*x4i-stream4[[4]]*stream4[[8]])/(feedflow*x4i)*100];
Print["NO2 % Separation = ",(feedflow*x5i-stream4[[4]]*stream4[[9]])/(feedflow*x5i)*100];
Print["Permeate % of feed flow = ",(stream9[[4]]/feedflow)*100];
Print[" "];
(*stream flow rates and concentrations*)
Print["STREAM 1"];
Print["flow rate (scfm) = ", stream1[[4]]];
Print["x1 = ", stream1[[5]]];
Print["x2 = ", stream1[[6]]];
Print["x3 = ", stream1[[7]]];
Print["x4 = ", stream1[[8]]];
Print["x5 = ", stream1[[9]]];
Print[" "];
Print["STREAM 2"];
Print["flow rate (scfm) = ", stream2[[4]]];
Print["x1 = ", stream2[[5]]];
Print["x2 = ", stream2[[6]]];
Print["x3 = ", stream2[[7]]];
Print["x4 = ", stream2[[8]]];
Print["x5 = ", stream2[[9]]];
Print[" "];
Print["STREAM 3"];
Print["flow rate (scfm) = ", stream3[[4]]];
```

```
Print["x1 = ", stream3[[5]]];
Print["x2 = ", stream3[[6]]];
Print["x3 = ", stream3[[7]]];
Print["x4 = ", stream3[[8]]];
Print["x5 = ", stream3[[9]]];
Print[" "];
Print["STREAM 4"];
Print["flow rate (scfm) = ", stream4[[4]]];
Print["x1 = ", stream4[[5]]];
Print["x2 = ", stream4[[6]]];
Print["x3 = ", stream4[[7]]];
Print["x4 = ", stream4[[8]]];
Print["x5 = ", stream4[[9]]];
Print[" "];
Print["STREAM 5"];
Print["flow rate (scfm) = ", stream5[[4]]];
Print["x1 = ", stream5[[5]]];
Print["x2 = ", stream5[[6]]];
Print["x3 = ", stream5[[7]]];
Print["x4 = ", stream5[[8]]];
Print["x5 = ", stream5[[9]]];
Print[" "];
Print["STREAM 6"];
Print["flow rate (scfm) = ", stream6[[4]]];
Print["x1 = ", stream6[[5]]];
Print["x2 = ", stream6[[6]]];
Print["x3 = ", stream6[[7]]];
Print["x4 = ", stream6[[8]]];
Print["x5 = ", stream6[[9]]];
Print[" "];
Print["STREAM 7"];
Print["flow rate (scfm) = ", stream7[[4]]];
Print["x1 = ", stream7[[5]]];
```

```
Print["x2 = ", stream7[[6]]];
Print["x3 = ", stream7[[7]]];
Print["x4 = ", stream7[[8]]];
Print["x5 = ", stream7[[9]]];
Print[" "];
Print["STREAM 8"];
Print["flow rate (scfm) = ", stream8[[4]]];
Print["x1 = ", stream8[[5]]];
Print["x2 = ", stream8[[6]]];
Print["x3 = ", stream8[[7]]];
Print["x4 = ", stream8[[8]]];
Print["x5 = ", stream8[[9]]];
Print[" "];
Print["STREAM 9"];
Print["flow rate (scfm) = ", stream9[[4]]];
Print["x1 = ", stream9[[5]]];
Print["x2 = ", stream9[[6]]];
Print["x3 = ", stream9[[7]]];
Print["x4 = ", stream9[[8]]];
Print["x5 = ", stream9[[9]]];
Print["@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@"];


Label[toomucharea];

(*end of function
)
```

APPENDIX I – The Simplex Method

Although more sophisticated optimization methods exist, sequential simplex optimization method is chosen due to its simplicity and ease of application. Not to be confused with linear programming simplex method [60], sequential simplex optimization is a sequential search method which can be used for nonlinear functions and experimental equipment. A brief description of the method follows. For more details, consult the reference [61].

The measurable result or output of an experimental process is referred to as the ***system response***. If the process is characterized by several ***system variables***, and these variables are purposely varied, the system response is expected to change. It is assumed that there exists only one set of values for these system variables that produce the most desirable system response. This set of values is called ***optimal conditions.*** An experimental process may have a broad but specified range for each system variable, but an optimal set of conditions will remain unknown unless it is sought.

The information from one experimental run is comprised of a set of system variables and the system response. Together this information is considered as ***one data point.*** Sequential simplex optimization uses a set of data points, called ***a simplex***, to recommend system variables for one new data point. The method then "throws out" the data point with the least desirable system response, and a new simplex is formed with the new data point. This process proceeds until data points with progressively more desirable responses can not be found.

Sequential simplex optimization operates by surrounding or enclosing a range of experimental conditions. In order to surround a region of experimental conditions, the simplex must have one more data point than there are system variables. In other words, if there are n system variables then there must be n+1 data points. For example, if two variables are to be optimized, then three data points are needed to enclose an area. If three variables are to be optimized, then four data points are required to enclose a volume, and if four variables are to be optimized, then 5 data points are required to enclose a four dimensional space.

The following algorithm describes the variable-size simplex method for four variables. In this method, the range of each system variable may increase or decrease from simplex to simplex. The table format is useful for simplex calculations and will be explained in the following algorithm.

Step 1.

Rank each data point within a simplex in decreasing order from most desirable to least desirable response. Label the best data point **A**, the next best data point **B**, etc...,and the worst data point **E**. Enter all data points in their respective rows on the worksheet.

Example of Step 1

Simplex Number	System Variables				System Response
	V1	V2	V3	V4	R
A	55	40	11	40	.95
B	95	40	11	40	.84
C	55	50	11	40	.77
D	55	40	14	40	.76
E	55	40	11	90	.42
Average (of A to D)					
N1 = Average + (Average - E)					
N2 = N1 + (Average - E)					
N3 = Average + (Average - E)/2					
N4 = E + (Average - E)/2					

Step 2.

Calculate system variable values for the new data point, **N1**. The variables of a data point can be thought of as a coordinate since it consists of several system variables (V1, V2, V3, or V4).

A = (V1A, V2A, V3A, V4A)

To calculated the new point one must first find the average coordinate of all the simplex data points except the worst one.

Average = (A+B+C+D)/4

From this average the coordinates of the worst point are subtracted, therefore creating a vector.

Difference = Average – E

The coordinates of the new data point are then calculated by adding the vector to the average.

N1 = Average + Difference

This process effectively moves the simplex away from the worst data point.

Example of Step 2

Simplex Number	System Variables				System Response
	V1	V2	V3	V4	R
A	55	40	11	90	.95
B	95	40	11	40	.84
C	55	50	11	40	.77
D	55	40	14	40	.76
E	55	40	11	40	.42
Average (of A to D)	65	42.5	11.75	52.5	
N1 = Average + (Average - E)	75	45	12.5	65	
N2 = N1 + (Average - E)					
N3 = Average + (Average - E)/2					
N4 = E + (Average - E)/2					

Step 3.

Evaluate the response for **N1**. This data point must now be compared with the best data point in the simplex, **A**, the worst data point in the simplex **E**, and the next to worst data point in the simplex **D**. (The following comparative notation is interpreted as follows: **D <= N1**, the response of data point **D** is equal to or worse than the response of data point **N1**)

A. If **D <= N1 <= A**, use simplex **A-B-C-D-N1**, and go to step 4.

B. If **N1** > **A**, calculate and evaluate the system variables and response for another new data point **N2**. The logic here is to move further in the direction of positive responses. This is done by adding the same vector from above to the coordinates of **N1**.

N2 = **N1** + **Difference**

a) If **N2** >= **A**, use simplex **A-B-C-D-N2**, and go to step 4.

b) If **N2** < **A**, use simplex **A-B-C-D-N1**, and go to step 4.

C. If **N1** < **D**, compare values of **N1** with **E**:

a) If **N1** >= **E**, then the movement was mildly successful and the simplex method needs a new point so the next simplex can move in a different direction. The new data point coordinates, **N3**, are now calculated by dividing the vector, calculated above, by two and added to the average. This point is effectively retracted half way between the average and **N1**.

N3 = Average + **Difference**/2

Use simplex **A-B-C-D-N3**, and go to step 4.

b) If **N1** < **E**, then the movement was not successful, but the next simplex still needs a new data point other than **E**. The coordinates of the new data point, **N4**, are now calculated by dividing the vector, calculated above, by two and added to the coordinates of the worst point **E**. This point is effectively retracted a quarter of the way between **E** and **N1**.

N4 = **E** + **Difference**/2

Use simplex **A-B-C-D-N4**, and go to step 4.

Example of Step 3-B-b

(Note: N1 will be selected since N2 is not as good as A.)

Simplex Number	System Variables				System Response
	V1	V2	V3	V4	R
A	55	40	11	90	.95
B	95	40	11	40	.84
C	55	50	11	40	.77
D	55	40	14	40	.76
E	55	40	11	40	.42
Average (of A to D)	65	42.5	11.75	52.5	
N1 = Average + (Average - E)	75	45	12.5	65	.97
N2 = N1 + (Average - E)	85	47.5	13.25	77.5	.88
N3 = Average + (Average - E)/2					
N4 = E + (Average - E)/2					

Step 4.

Never transfer the worst data point, **E**, to the next worksheet. Always transfer the current data points **A** to **D** to the next worksheet. Re-rank old data points **A** to **D** and the new data point (**N1**, **N2**, **N3**, or **N4**) on the new worksheet, and go to step 2.

Example of Step 4

The Next Simplex

Simplex Number	System Variables				System Response
	V1	V2	V3	V4	R
A	75	45	12.5	65	.97
B	55	40	11	90	.95
C	95	40	11	40	.84
D	55	50	11	40	.77
E	55	40	14	40	.76
Average (of A to D)					
N1 = Average + (Average - E)					
N2 = N1 + (Average - E)					
N3 = Average + (Average - E)/2					
N4 = E + (Average - E)/2					

An important issue in searching for an optimal set of system variables is the concept of local verses global optima. Imagine a three dimensional surface with several peaks and valleys (a system response surface). The axis in the direction of the peaks and valleys is the desirability, and the other axis represent two system variables. The location of the highest peak, highest desirability, is the global optimum point. Other peaks are local optimum points. Local optima represent the highest desirability in their immediate location. As one might imagine, finding a global optima in a system containing numerous local optima is a difficult task. This is especially true when there are a number of system configurations with more that just two system variables each.

This method assumes that the responses are continuous, and that there are no discontinuous maxima as system variables are changed.

The movement from simplex to simplex is dependent upon the certainty of response measurements. For this reason replication is necessary to determine an approximate deviation of response data. If the deviation is consistently large, the sequential simplex method may not work.

Convergence upon the optimal conditions is determined when the sequential simplex algorithm collapses upon an area where the new data points can not be ranked with any confidence.

If the sequential simplex method calculates one or more variable values that exceed specified boundaries, the calculated point is ranked as infinitely bad, and N3 or N4 is calculated.

DESIRABILITY - COMBINATION OF RESPONSE VARIABLES

The simplex method operates by replacing an undesirable data point with a more desirable one. If we are interested in more than one response, an objective method for evaluating and rating a combination of responses is needed.

Due to simplicity and ease of application, a linear desirability function is used to evaluate and rate sets of responses. A desirability function normalizes individual responses to a scale of 0 to 1, where 0 equates to the least desirable response and 1 equates to the most desirable response. Desirability can be calculated for an individual response as follows:

$$di = (Ri-Riu)/(Rid-Riu)$$

where:

di = desirability for an individual response

Ri = the value of an individual response

Riu = the value of the most undesirable response imaginable (0%)

Rid = the value of the most desirable response imaginable (100%)

The overall desirability is then determined by calculating the geometric mean of all the desirabilities for a data point. This value is used to rank the data points.

$$D = (d1d2...dn)^{(1/n)}$$

where: D = overall desirability

dn = an individual desirability from an individual response

n = the number of desirabilities of responses in a set of data

INITIAL SIMPLEX

When using a variable size sequential simplex algorithm (usage of N3 and N4), a large initial simplex is recommended [61]. The simplex should cover most of the variable space, and it is expected to shrink and converge near the optimal conditions.

A corner initial simplex is a method for calculating variable values so that they are orthogonal to the systems coordinates [61]. After determining the experimental range of each variable, a centered low

and high value is selected covering 80% of this range. The variable values of the first simplex then appear as follows:

Variables for the Initial Simplex

Vertex	Variable 1	Variable 2	Variable 3	Variable 4
1	low	low	low	low
2	high	low	low	low
3	low	high	low	low
4	low	low	high	low
5	low	low	low	high

For the above example the experimental range of each variable was:

V1: 50 to 100

V2: 38.75 to 41.25

V3: 10 to 15

V4: 33.75 to 96.25

Variables for the initial Simplex in the above example

Vertex	Variable 1	Variable 2	Variable 3	Variable 4
1	55	40	11	40
2	95	40	11	40
3	55	50	11	40
4	55	40	14	40
5	55	40	11	90

APPENDIX J – Simplex Method Code

(*Sequential Simplex Search Algorithm*)
(*The following code is written to optimize the output of the co5 program. The program is called opco5, and the input values are:

maximum possible feed pressure,
minimum possible feed pressure,
maximum possible permeate pressure,
minimum possible permeate pressure,
maximum possible feed feed flow rate,
minimum possible feed feed flow rate.

The program may be used for any single-stage program by replacing the co5 with another program name (like co3, counter5, or cross2). The program must optimizes the feed pressure, permeate pressure, and feed flow rate. The membrane area should be set to 100 sqft. The program designation must have the following format.

co5[{pfeed_,pperm_,feedflow_}] := (etc.

The output of the program must not contain any active print statements. The response (value to maximize) must be the last value calculated in the program.*)

(*PART 1 -- ENTER INITIAL SIMPLEX POINTS, NUMBER OF ITERATIONS,
AND VARIABLE CONSTRATINTS*)
opco5[pfeedmax_,pfeedmin_,ppermmax_,ppermmin_,
feedflowmax_,feedflowmin_] := (

```
(*Calculate the initial Simplex points as Matnematica lists "{}".
Note: They have the following form:
                                    hi,hi,hi
                                    lo,hi,hi
                                    hi,lo,hi
                                    hi,hi,lo
Where hi refers to 10% of the range less than the max, and
lo refers to 10% of the range more than the min.*)

begin1 = {pfeedmax-0.1*(pfeedmax-pfeedmin),
          ppermmin+0.1*(ppermmax-ppermmin),
          feedflowmin+0.1*(feedflowmax-feedflowmin)};
begin2 = {pfeedmin+0.1*(pfeedmax-pfeedmin),
          ppermmin+0.1*(ppermmax-ppermmin),
          feedflowmin+0.1*(feedflowmax-feedflowmin)};
begin3 = {pfeedmax-0.1*(pfeedmax-pfeedmin),
          ppermmax-0.1*(ppermmax-ppermmin),
          feedflowmin+0.1*(feedflowmax-feedflowmin)};
begin4 = {pfeedmax-0.1*(pfeedmax-pfeedmin),
          ppermmin+0.1*(ppermmax-ppermmin),
          feedflowmax-0.1*(feedflowmax-feedflowmin)};



(*PART 2 -- CALCULATE AND PRINT RESULTS OF INITIAL SIMPLEX POINTS*)
(*Calculate desirability of each "begin point" *)
desirebegin1 = co5[begin1];
desirebegin2 = co5[begin2];
desirebegin3 = co5[begin3];
desirebegin4 = co5[begin4];

(*Print the initial simplex*)
Print[begin1,desirebegin1];
```

```
Print[begin2,desirebegin2];
Print[begin3,desirebegin3];
Print[begin4,desirebegin4];
Print[" "];


(*Assign convergence comparison values so the While loop will
initiate. These are recalculated at the end of the loop.*)
comparepfeed = (begin1[[1]]+begin2[[1]]+begin3[[1]]+begin4[[1]])/4;
comparepperm = (begin1[[2]]+begin2[[2]]+begin3[[2]]+begin4[[2]])/4;
comparefeedflow =
                        (begin1[[3]]+begin2[[3]]+begin3[[3]]+begin4[[3]])/4;



(*PART 3 -- SORT AND ARRANGE SIMPLEX POINTS ACCORDING TO
DESIRABILITY. THIS IS DONE WITHIN THE DO[] LOOP FOR THE
INPUT NUMBER OF ITERATIONS. THE CONVERGENCE CRITERIA IS A CHANGE BETWEEN
THE VARIABLES OF TWO CONSECUTIVE DATA POINTS LESS THAN 0.1%*)

While[Abs[(begin1[[1]]-comparepfeed)/comparepfeed]>0.001 ||
        Abs[(begin2[[1]]-comparepfeed)/comparepfeed]>0.001 ||
        Abs[(begin3[[1]]-comparepfeed)/comparepfeed]>0.001 ||
        Abs[(begin4[[1]]-comparepfeed)/comparepfeed]>0.001 ||
        Abs[(begin1[[2]]-comparepperm)/comparepperm]>0.001 ||
        Abs[(begin2[[2]]-comparepperm)/comparepperm]>0.001 ||
        Abs[(begin3[[2]]-comparepperm)/comparepperm]>0.001 ||
        Abs[(begin4[[2]]-comparepperm)/comparepperm]>0.001 ||
        Abs[(begin1[[3]]-comparefeedflow)/comparefeedflow]>0.001 ||
        Abs[(begin2[[3]]-comparefeedflow)/comparefeedflow]>0.001 ||
        Abs[(begin3[[3]]-comparefeedflow)/comparefeedflow]>0.001 ||
        Abs[(begin4[[3]]-comparefeedflow)/comparefeedflow]>0.001,
```

```
(*Sort and assign begining arrays according to desirability*)
sorted = Reverse[Sort[{desirebegin1,desirebegin2,
        desirebegin3,desirebegin4}]];

If[sorted[[1]]==desirebegin1,
        {point1 = begin1 , desirepoint1 = desirebegin1},
        If[sorted[[1]]==desirebegin2,
        {point1 = begin2 , desirepoint1 = desirebegin2},
        If[sorted[[1]]==desirebegin3,
        {point1 = begin3 , desirepoint1 = desirebegin3},
        If[sorted[[1]]==desirebegin4,
        {point1 = begin4 , desirepoint1 = desirebegin4}]]]];

If[sorted[[2]]==desirebegin1,
        {point2 = begin1 , desirepoint2 = desirebegin1},
        If[sorted[[2]]==desirebegin2,
        {point2 = begin2 , desirepoint2 = desirebegin2},
        If[sorted[[2]]==desirebegin3,
        {point2 = begin3 , desirepoint2 = desirebegin3},
        If[sorted[[2]]==desirebegin4,
        {point2 = begin4 , desirepoint2 = desirebegin4}]]]];

If[sorted[[3]]==desirebegin1,
        {point3 = begin1 , desirepoint3 = desirebegin1},
        If[sorted[[3]]==desirebegin2,
        {point3 = begin2 , desirepoint3 = desirebegin2},
        If[sorted[[3]]==desirebegin3,
        {point3 = begin3 , desirepoint3 = desirebegin3},
        If[sorted[[3]]==desirebegin4,
        {point3 = begin4 , desirepoint3 = desirebegin4}]]]];
```

```
If[sorted[[4]]==desirebegin1,
        {point4 = begin1 , desirepoint4 = desirebegin1},
        If[sorted[[4]]==desirebegin2,
        {point4 = begin2 , desirepoint4 = desirebegin2},
        If[sorted[[4]]==desirebegin3,
        {point4 = begin3 , desirepoint4 = desirebegin3},
        If[sorted[[4]]==desirebegin4,
        {point4 = begin4 , desirepoint4 = desirebegin4}]]]];



(*PART 4 -- WITHIN THE DO[] LOOP, FIND THE NEXT POINT FOR THE
SIMPLEX.  SEE APPENDIX G FOR CALCULATIONS AND
CONSTRAINT COMPARISONS.*)
(*Desirability comparisons and calculations for next point*)
mean[u__] := Plus[u]/Length[{u}];
average =
        {mean[point1[[1]],point2[[1]],point3[[1]]],
        mean[point1[[2]],point2[[2]],point3[[2]]],
        mean[point1[[3]],point2[[3]],point3[[3]]]};

difference = average - point4;

newpoint1 = average + difference;



If[newpoint1[[1]]>pfeedmax || newpoint1[[1]]<pfeedmin ||
        newpoint1[[2]]>ppermmax || newpoint1[[2]]<ppermmin ||
        newpoint1[[3]]>feedflowmax || newpoint1[[3]]<feedflowmin,
        Goto[out5]];

newdesirepoint1 = co5[newpoint1];
```

```
If[newdesirepoint1<=desirepoint1 &&
        newdesirepoint1>=desirepoint3,
        Goto[out1], Goto[newpoint2]];

Label[out1];
newcurrent = newpoint1;
newdesirecurrent = newdesirepoint1;
Goto[done];

Label[newpoint2];
If[newdesirepoint1>desirepoint1, Goto[out2], Goto[newpoint3]];

Label[out2];
newpoint2 = newpoint1 + difference;
If[newpoint2[[1]]>pfeedmax || newpoint2[[1]]<pfeedmin ||
        newpoint2[[2]]>ppermmax || newpoint2[[2]]<ppermmin ||
        newpoint2[[3]]>feedflowmax || newpoint2[[3]]<feedflowmin,
        Goto[out1], Goto[out3]];

Label[out3];
newdesirepoint2 = co5[newpoint2];
If[newdesirepoint2>=newdesirepoint1, Goto[out4], Goto[out1]];

Label[out4];
newcurrent = newpoint2;
newdesirecurrent = newdesirepoint2;
Goto[done];

Label[newpoint3];
If[(newdesirepoint1>=desirepoint4 && newdesirepoint1<desirepoint3),
        Goto[out5], Goto[newpoint4]];
```

```
Label[out5];
newcurrent = average + difference/2;
If[newcurrent[[1]]>pfeedmax || newcurrent[[1]]<pfeedmin ||
        newcurrent[[2]]>ppermmax || newcurrent[[2]]<ppermmin ||
        newcurrent[[3]]>feedflowmax || newcurrent[[3]]<feedflowmin,
        Goto[newpoint4], Goto[out6]];

Label[out6];
newdesirecurrent = co5[newcurrent];
Goto[done];

Label[newpoint4];
newcurrent = point4 + difference/2;
newdesirecurrent = co5[newcurrent];

Label[done];



(*PART 5 -- PRINT THE NEW SIMPLEX DATA POINT AND RENAME ALL POINTS
BEFROE GOING TO THE TOP OF THE DO[] LOOP.*)
Print[N[newcurrent],N[newdesirecurrent]];
begin1 = point1;
begin2 = point2;
begin3 = point3;
begin4 = newcurrent;
desirebegin1 = desirepoint1;
desirebegin2 = desirepoint2;
desirebegin3 = desirepoint3;
desirebegin4 = newdesirecurrent;

comparepfeed =  (begin1[[1]]+begin2[[1]]+begin3[[1]]+begin4[[1]])/4;
comparepperm = (begin1[[2]]+begin2[[2]]+begin3[[2]]+begin4[[2]])/4;
```

```
comparefeedflow =
                                (begin1[[3]]+begin2[[3]]+begin3[[3]]+begin4[[3]])/4;
];


(*PART 6 -- PRINT FINAL SIMPLEX.  ALL VARIABLES SHOULD HAVE
CONVERGED, AND THESE POINTS SHOULD SURROUND AN OPTIMAL POSITION
WITHIN THE VARIABLE SPACE.  IF ALL VARIABLES HAVE NOT CONVERGED,
THEN RE-ENTER FINAL SIMPLEX POINTS AS INITIAL SIMPLEX.*)
Print[" "];
Print[N[begin1],N[desirepoint1]];
Print[N[begin2],N[desirepoint2]];
Print[N[begin3],N[desirepoint3]];
Print[N[newcurrent],N[newdesirecurrent]];

co5[{comparepfeed,comparepperm,comparefeedflow}];

pfeed = comparepfeed;
pperm = comparepperm;
feedflow = comparefeedflow;

y1perm  =
        (feedflow*x1i-fint[totalarea]*x1int[totalarea])/g;
y2perm  =
        (feedflow*x2i-fint[totalarea]*x2int[totalarea])/g;
y3perm =
        (feedflow*x3i-fint[totalarea]*x3int[totalarea])/g;
y4perm =
        (feedflow*x4i-fint[totalarea]*x4int[totalarea])/g;
y5perm =
        (feedflow*x5i-fint[totalarea]*x5int)/g;
```

```
Print[" "];
Print["**************************************"];
Print["SEPARATION CONDITIONS AND PERMEABILITIES"];
Print["feed pressure (psia)        = ", pfeed];
Print["feed side pressure drop (psi) = ", pdrop];
Print["permeate pressure (psia)    = ", pperm];
Print["membrane area (sqft)        = ", totalarea];
Print["membrane thickness (in)     = ", d];
Print["permeability of 1 (barrers) = ", permeability1];
Print["permeability of 2 (barrers) = ", permeability2];
Print["permeability of 3 (barrers) = ", permeability3];
Print["permeability of 4 (barrers) = ", permeability4];
Print["permeability of 5 (barrers) = ", permeability5];
Print["   "];
Print["FEED FLOW RATE AND MOLE FRACTIONS"];
Print["feed flow (scfm) = ", feedflow];
Print["x1i = ", x1i];
Print["x2i = ", x2i];
Print["x3i = ", x3i];
Print["x4i = ", x4i];
Print["x5i = ", x5i];
Print[" "];
Print["RETENTATE FLOW RATE AND MOLE FRACTIONS"];
Print["retentate flow (scfm) = ", fint[totalarea]];
Print["x1 = ", x1int[totalarea]];
Print["x2 = ", x2int[totalarea]];
Print["x3 = ", x3int[totalarea]];
Print["x4 = ", x4int[totalarea]];
Print["x5 = ", 1-x1int[totalarea]-x2int[totalarea]-
                                    x3int[totalarea]-x4int[totalarea]];
Print[" "];
Print["PERMEATE FLOW RATE AND MOLE FRACTIONS"];
Print["permeate flow (scfm) = ", g];
```

```
Print["y1 = ", y1perm];
Print["y2 = ", y2perm];
Print["y3 = ", y3perm];
Print["y4 = ", y4perm];
Print["y5 = ", y5perm];
Print[" "];
Print["OPTIMIZATION RESULTS"];
Print["SO2 % Separation = ",
        (feedflow*x4i-x4int[totalarea]*fint[totalarea])/
        (feedflow*x4i)*100];
Print["NO2 % Separation = ",
        (feedflow*x5i-x5int*fint[totalarea])/
        (feedflow*x5i)*100];
Print["Permeate % of feed flow = ",(g/feedflow)*100];
Print["SO2 % Separation Desirability = ",
        ((feedflow*x4i-x4int[totalarea]*fint[totalarea])/
        (feedflow*x4i)*100)/100];
Print["NO2 % Separation Desirability = ",
        ((feedflow*x5i-x5int*fint[totalarea])/
        (feedflow*x5i)*100)/100];
Print["Permeate % of Feedflow Desirability = ",
        ((g/feedflow)*100 - 100)/(0-100)];
Print[" "];
Print["Desirability = ",
        ((((feedflow*x4i-x4int[totalarea]*fint[totalarea])/
        (feedflow*x4i)*100)/100)*
        (((feedflow*x5i-x5int*fint[totalarea])/
        (feedflow*x5i)*100)/100)*
        (((g/feedflow)*100 - 100)/(0-100)))^(1/3)];


(*End of function opco5*)
)
```

APPENDIX K – Explanation of Simplex Output

The following is an example of the output produced by the program in Appendix H. The hypothetical input to the program is:

150 psia maximum possible feed pressure,

42 psia minimum possible feed pressure,

15 psia maximum possible permeate pressure,

8 psia minimum possible permeate pressure,

100 scfm maximum possible feed flow rate, and

3 scfm minimum possible feed flow rate.

The membrane area was set to 100 scfm.

Each line of output represents a data point, the system variables and response. The format is:

{feed pressure, permeate pressure, feed flow rate}system response.

The system response is calculated from the separation percentage of SO_2, separation percentage of NO_2, and stage cut.

$$system\ response = \left(\frac{SO2\%\ Separation}{100} \bullet \frac{NO2\%\ Separation}{100} \bullet \frac{\frac{Permeate\ \ Flow}{Feed\ \ Flow} \bullet 100 - 100}{0 - 100} \right)^{1/3}$$

The first 4 lines of output, following the "From In[31]:=" statement, is the initial simplex. The following lines represent the next data point calculated. The final 4 lines of output are the final simplex (collapsed on a region). Finally, a report of the separation at optimal conditions is printed.

One should note that the system did not converge on the lowest feed flow rate because there is eventually a trade-off between percentage of separation and an increase in stage cut.

In[31]:=
opco5[150,42,15,8,100,3]
From In[31]:=

{139.2, 8.7, 12.7}0.763201
{52.8, 8.7, 12.7}0.379743
{139.2, 14.3, 12.7}0.683422
{139.2, 8.7, 90.3}0.360919

{124.8, 9.63333, 51.5}0.422564
{93.6, 9.78889, 19.1667}0.51715
{124.4, 10.2815, 33.1778}0.506798
{123.8, 11.2537, 5.69444}0.771496
{113.833, 10.6034, 14.7657}0.638815
{144.183, 11.8252, 8.16435}0.772624
{133.992, 8.73943, 6.9294}0.798387
{131.388, 11.5591, 4.0441}0.768457
{136.596, 9.65305, 9.8147}0.772376
{145.485, 9.48196, 9.607}0.788134
{145.844, 10.378, 6.65247}0.792924
{140.569, 8.38711, 7.51226}0.804884
{142.81, 9.32507, 8.31919}0.794653
{135.763, 8.03681, 8.05419}0.802799
{139.792, 8.85643, 7.9089}0.798479
{143.425, 8.11414, 8.72083}0.805845
{139.856, 8.51789, 8.00233}0.801549
{139.951, 8.01009, 8.14248}0.806289
{146.867, 8.30408, 8.19619}0.80828
{144.837, 8.02059, 8.77362}0.807788
{144.804, 8.10648, 7.67062}0.809782

{148.278, 8.21053, 8.24898}0.809948
{148.463, 8.39347, 7.30357}0.809305
{147.497, 8.16958, 7.28592}0.810673
{146.058, 8.04656, 7.95097}0.810655
{149.751, 8.17797, 7.98663}0.811509
{147.259, 8.05221, 7.23337}0.811473
{149.225, 8.17659, 7.27747}0.811287
{149.993, 8.1016, 7.71239}0.812431
{148.778, 8.04459, 8.01079}0.812135
{148.383, 8.08013, 7.56832}0.811883
{148.702, 8.02418, 7.65244}0.812494
{149.932, 8.03345, 8.01543}0.81284
{149.925, 8.05732, 7.68473}0.812768
{149.283, 8.00667, 7.8201}0.812875
{149.207, 8.02833, 7.74626}0.812679
{149.966, 8.03455, 7.887}0.812936
{149.628, 8.00868, 8.0189}0.812897
{149.472, 8.00823, 7.85529}0.812937
{149.892, 8.02239, 7.97054}0.812952
{149.925, 8.03477, 7.78965}0.812949
{149.56, 8.00904, 7.85665}0.812972
{149.953, 8.02899, 7.88078}0.81298
{149.678, 8.00551, 8.01566}0.812953
{149.569, 8.00664, 7.86486}0.812993
{149.686, 8.0102, 7.94155}0.812975
{149.911, 8.02151, 7.9348}0.812994
{149.936, 8.02789, 7.84541}0.812996
{149.658, 8.00837, 7.8826}0.813013
{149.968, 8.02557, 7.89898}0.813008
{149.796, 8.0197, 7.81653}0.81301
{149.679, 8.00787, 7.88666}0.813025
{149.583, 8.00519, 7.84341}0.813022
{149.561, 8.00086, 7.89807}0.813022

{149.558, 8.00091, 7.86949}0.813035
{149.651, 8.00845, 7.83497}0.813031
{149.723, 8.00685, 7.90431}0.813046
{149.574, 8.00048, 7.83544}0.813062
{149.635, 8.0056, 7.85236}0.81304
{149.816, 8.01111, 7.85312}0.81308
{149.844, 8.00724, 7.88816}0.81311
{149.788, 8.00512, 7.76811}0.81314
{149.937, 8.0115, 7.83697}0.81314
{149.936, 8.00165, 7.78699}0.813235
{149.931, 8.00494, 7.70656}0.813208
{149.859, 8.0001, 7.71234}0.813217
{149.969, 8.00078, 7.71889}0.81326
{149.926, 8.00289, 7.72298}0.813224
{149.986, 8.00261, 7.75826}0.813253
{149.982, 8.00107, 7.77058}0.813263
{149.958, 8.00157, 7.76812}0.813248
{149.99, 8.00145, 7.73981}0.813265
{149.978, 8.00035, 7.73551}0.813268
{149.998, 8.00113, 7.77837}0.813269
{149.994, 8.00088, 7.73188}0.813271
{149.99, 8.00012, 7.75736}0.813275
{149.986, 8.00053, 7.74569}0.813271
{149.986, 8.0002, 7.72828}0.813273
{149.999, 8.00014, 7.72614}0.813279
{149.993, 8.00052, 7.73457}0.813274
{149.998, 8.00029, 7.7449}0.813278
{149.997, 8.00002, 7.74692}0.81328
{149.994, 8.00014, 7.74834}0.813277
{150., 8.00016, 7.73481}0.81328
{149.999, 8.00001, 7.73149}0.813281
{149.998, 8.00002, 7.74354}0.81328

{149.999, 8.00001, 7.73149}0.813281

{150., 8.00016, 7.73481}0.81328

{149.997, 8.00002, 7.74692}0.81328

{149.998, 8.00002, 7.74354}0.81328

****

SEPARATION CONDITIONS AND PERMEABILITIES

feed pressure (psia) = 149.998

feed side pressure drop (psi) = 0

permeate pressure (psia) = 8.00005

membrane area (sqft) = 100

membrane thickness (in) = 0.000984

permeability of 1 (barrers) = 250

permeability of 2 (barrers) = 500

permeability of 3 (barrers) = 2700

permeability of 4 (barrers) = 12500

permeability of 5 (barrers) = 6350

FEED FLOW RATE AND MOLE FRACTIONS

feed flow (scfm) = 7.73919

x1i = 0.8

x2i = 0.0746

x3i = 0.12

x4i = 0.005

x5i = 0.0004

RETENTATE FLOW RATE AND MOLE FRACTIONS

retentate flow (scfm) = 5.47791

x1 = 0.897964

x2 = 0.0675339

x3 = 0.0335606

x4 = 0.000865818

x5 = 0.0000756474

PERMEATE FLOW RATE AND MOLE FRACTIONS

permeate flow (scfm) = 2.26128

y1 = 0.562684

y2 = 0.0917174

y3 = 0.329398

y4 = 0.015015

y5 = 0.00118574

OPTIMIZATION RESULTS

SO2 % Separation = 87.7432

NO2 % Separation = 86.6139

Permeate % of feed flow = 29.2186

SO2 % Separation Desirability = 0.877432

NO2 % Separation Desirability = 0.866139

Permeate % of Feedflow Desirability = 0.707814

Desirability = 0.81328