asyMemLocal – Trans-membrane Flux Simulator

User Manual for asymmetric membrane local flux boundary-value problem solver. **Current support** for polymer membranes and applicable sorption/diffusion models

Dylan J. Weber and Dr. Joseph K. Scott

Version 1.1

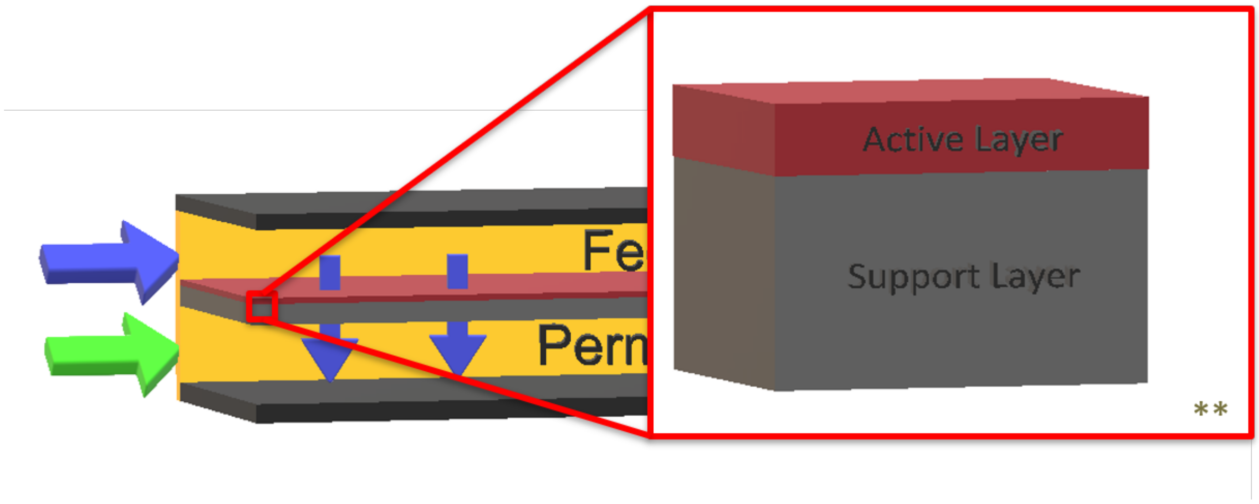
RAPID Center for Process Modeling
Georgia Institute of Technology
Atlanta, GA, USA

March 17th, 2021

Contents

1 Problem Statement

The purpose of this code is to solve for any number of partial fluxes through an asymmetric membrane layer based on a given set of experimental, modeling, and numerical method inputs and specifications. See introduction section for technical figure and input specifications required, but the point of this software is to provide the user with a means to calculate local transport of complex mixtures across an asymmetric polymer (current support) membrane utilizing the rigorous Maxwell-Stefan transport modeling framework which considers diffusional and thermodynamic coupling. See simple figure below of the system being solved for (boxed in red) transmembrane partial fluxes:



Even though there is one fundamental 2-point boundary value problem described above to solve, there are four different scripts included to:

- run a single complex mixture simulation
- fit transport diffusivities based on single component permeation experiments
- loop through a set of pressures and sorption models to predict single component transport
- loop through sets of mixtures, cross-diffusion models, polymer swelling diffusion models, and other model assumptions to generate a large data set to compare different ways to simulate the membrane system

2 End User Statement

This document outlines how to numerically simulate the modeling framework outlined in published journal article *A Framework for Predicting the Fractionation of Complex Liquid Feeds via Polymer Membranes* by Mathis et. al. (https://doi.org/10.1016/j.memsci.2021.119767) Our research group provided the modeling and simulation half to the publication. A future publication on the numerical methods and detailed comparisons of when to use which is also in progress and will be submitted soon titled *Improved Numerical Methods for Simulating Complex Mixture Transport Across Asymmetric Polymer Membranes using a Maxwell-Stefan Model.*

This document presents how to use each provided scripts and assumes the user is acquainted with various numerical methods such as full-discritization and shooting algorithms. Note the full-discritization methods are more prone to tuning based on number of nodal points and initial guesses, so our novel shooting algorithm approach is used as the default method.

This document also assumes the user is acquainted with the Maxwell-Stefan modeling framework as presented in Mathias et. al. as well as a good reference for a modeling framework presented by Krishna titled *Describing mixture permeation across polymeric membranes by a combination of Maxwell-Stefan and Flory-Huggins models* (https://doi.org/10.1016/j.polymer.2016.09.051). A full modeling and numerical methods overview is presented in the accompanying document titled "MODEL_METHOD_DOC.pdf" and **please note this is unpublished material so please do not redistribute this code without forwarding the person(s) to us using this Google form (https://forms.gle/Q5E3TD7BY2hj5MwCA). This is so we can provide the most up to date code and allow the same statement to be made**.

3 Introduction

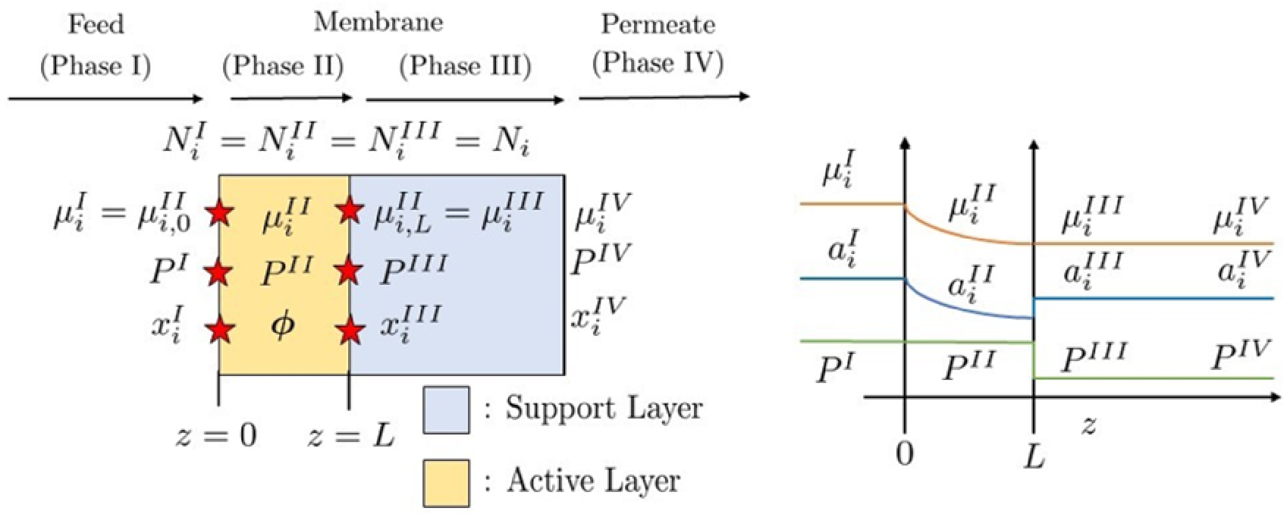
For the purposes of being able to understand what is actually being solved, first think about this local transport simulator from 3 main modeling steps as:

I. thermodynamic equilibrium between bulk feed phase (I) and the active layer membrane phase (II)

II. diffusion through the membrane phase (II)

III. thermodynamic equilibrium between active phase (II) and support layer phase (III)

and from the standpoint of inputs/parameters needed, see figures below first thinking about that active and local support layer in a much more quantitative sense



and in terms of equations for each local transport step (DoF = Degrees of Freedom)

Sorption-Diffusion Step I

$$\gamma_i^I = g_i^I(\mathbf{x}^I, \mathbf{s}, T, P^I),\ i = 1, 2, ..., n$$

$$a_{i,0}^{II} = \gamma_{i,0}^I x_i^I,\ i = 1, 2, ..., n$$

$$a_{i,0}^{II} = h_i^{II}(\boldsymbol{\phi}_0, \mathbf{s}, T, P^{II}),\ i = 1, 2, ..., n$$

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

Sorption-Diffusion Step II

$$\frac{d\boldsymbol{\phi}_{1:n}}{dz} = -\boldsymbol{\Gamma}^{-1}(\boldsymbol{\phi}, \mathbf{s})\mathbf{B}(\boldsymbol{\phi})\mathbf{x}^{III} N_{tot}^{V*}$$

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

Sorption-Diffusion Step III

$$\gamma_i^{III} = g_i^{III}(\mathbf{x}^{III}, \mathbf{s}, T, P^{III}),\ i = 1, 2, ..., n$$

$$a_{i,L}^{II} = \gamma_{i,L}^{III} x_i^{III} \eta,\ i = 1, 2, ..., n$$

$$a_{i,L}^{II} = h_i^{II}(\boldsymbol{\phi}_L, \mathbf{s}, T, P^{II}),\ i = 1, 2, ..., n$$

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

Known variables:

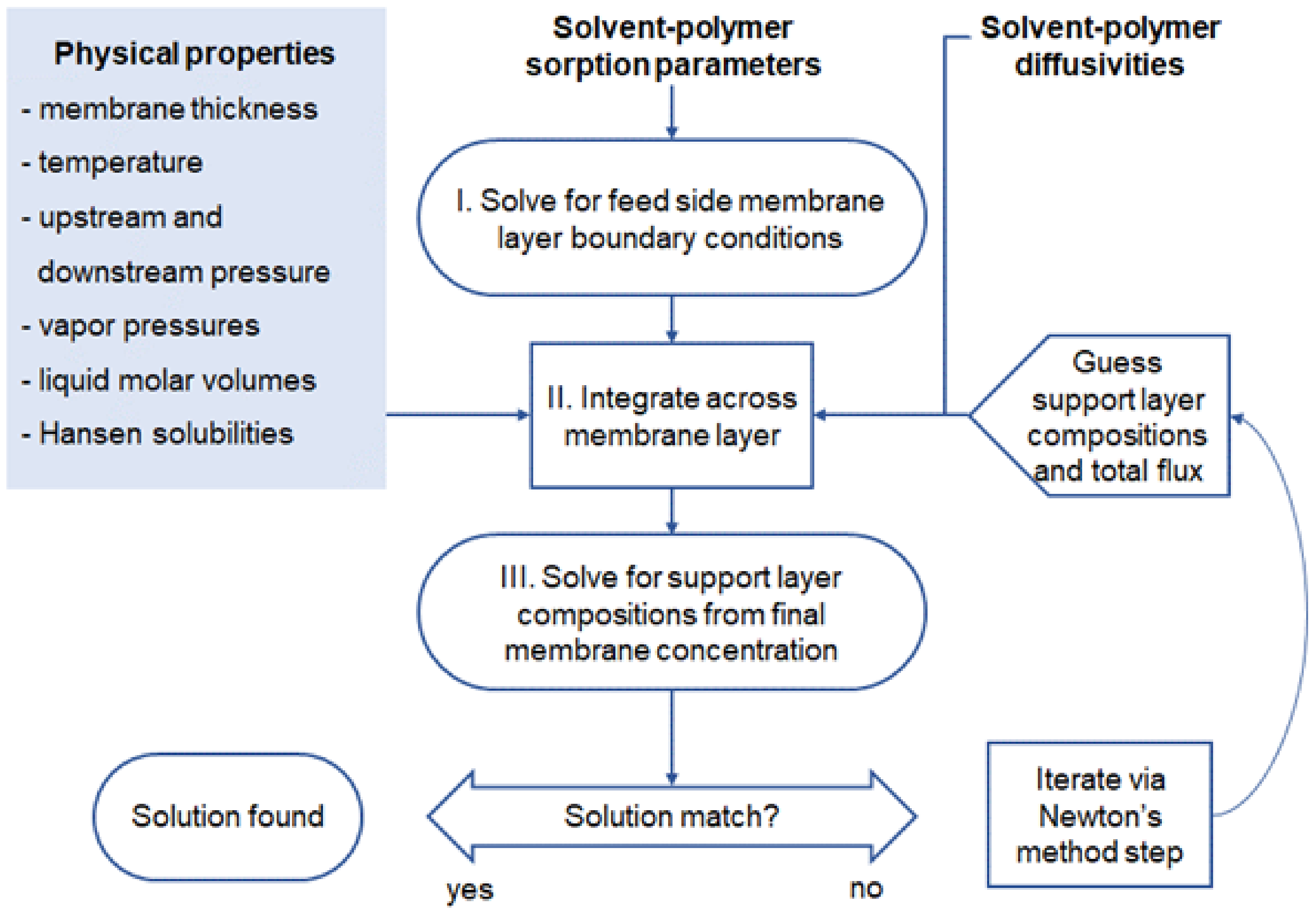
$$(\mathbf{x}^I, L, T, P^I, P^{II}, P^{III}, \mathbf{s}, \mathbf{Đ})$$

Unknown variables:

$$(\boldsymbol{\gamma}^I, \boldsymbol{\gamma}^{III}, \mathbf{a}_0^{II}, \mathbf{a}_L^{II}, \boldsymbol{\phi}_0, \boldsymbol{\phi}_L, \mathbf{x}^{III}, N_{tot}^{V*})$$

DoF: $4n + 3(n+1)$ equations
$-4n + 3(n+1)$ unknowns
$=$ **0 DoF**

finally the overall solution algorithm as

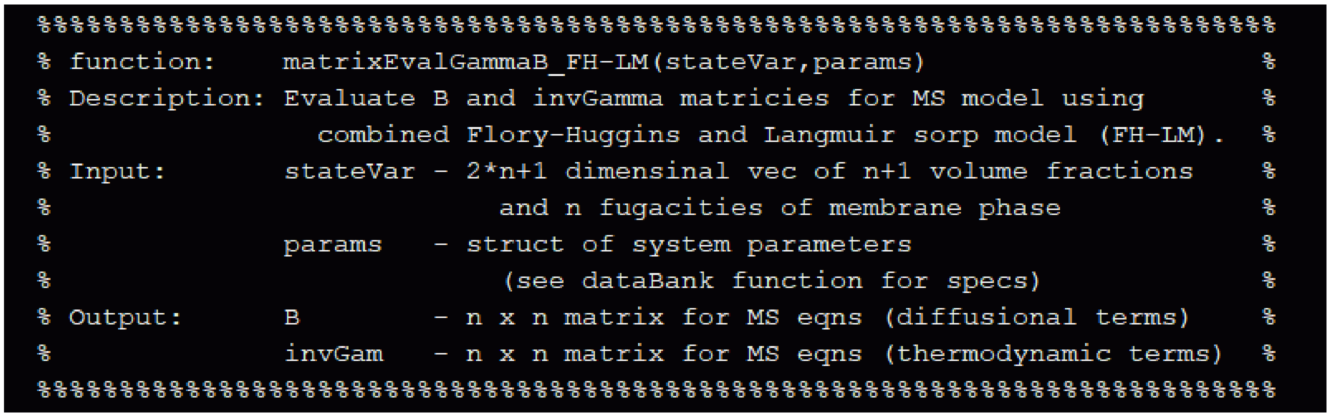

Physical properties
- membrane thickness
- temperature
- upstream and downstream pressure
- vapor pressures
- liquid molar volumes
- Hansen solubilities
Solvent-polymer sorption parameters
Solvent-polymer diffusivities
I. Solve for feed side membrane layer boundary conditions
II. Integrate across membrane layer
Guess support layer compositions and total flux
III. Solve for support layer compositions from final membrane concentration
Solution found
Solution match?
yes
no
Iterate via Newton's method step

Then see pages 2-3 in "MODEL_METHOD_DOC.pdf" for nomenclature and detailed derivations. And see pages 11-12 in that same document for the detailed nonlinear solver equations to test if the iterate guess and iterate found solutions match.

For the remainder of this user manual, we will discuss:

- the main parameter data bank function which is used for all the membrane sorption, diffusion, component physical parameters, and membrane physical parameters
- the main numerical method specification code section and what each means
- the diffusivity fitting script
- the single mixture simulation script
- the single component permeation prediction script
- the multiple mixture/sorption/diffusion model permeation prediction script
- future work

And lastly within every function a preamble of the format

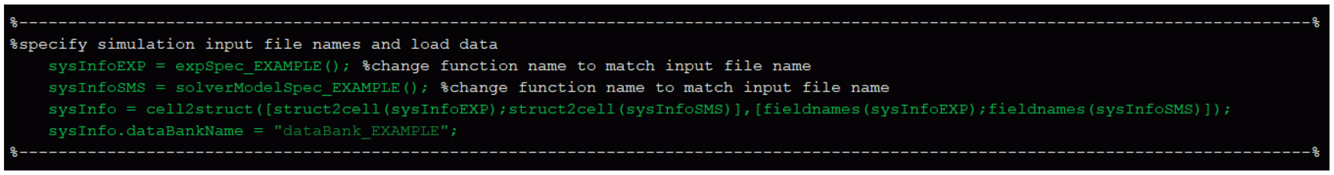


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function:    matrixEvalGammaB_FH-LM(stateVar,params)                     %
% Description: Evaluate B and invGamma matricies for MS model using        %
%                combined Flory-Huggins and Langmuir sorp model (FH-LM).   %
% Input:       stateVar - 2*n+1 dimensinal vec of n+1 volume fractions     %
%                         and n fugacities of membrane phase               %
%              params   - struct of system parameters                      %
%                          (see dataBank function for specs)               %
% Output:      B        - n x n matrix for MS eqns (diffusional terms)     %
%              invGam   - n x n matrix for MS eqns (thermodynamic terms)   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

is present to help the user understand the function name, description, inputs, and outputs.

4 Inputs

The next subsection detail the main three files required. Note the general way to specify each of these files for all included scripts at the top of the file as:



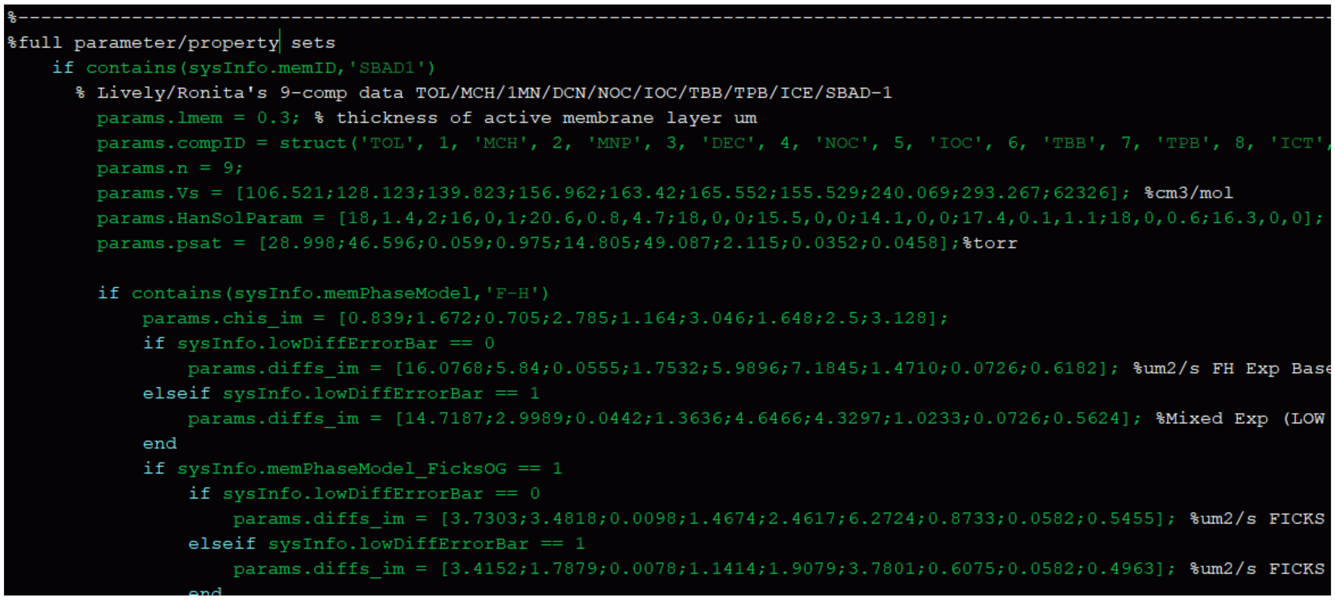
```
%-----------------------------------------------------------------------------------------------------------------------------%
%specify simulation input file names and load data
    sysInfoEXP = expSpec_EXAMPLE(); %change function name to match input file name
    sysInfoSMS = solverModelSpec_EXAMPLE(); %change function name to match input file name
    sysInfo = cell2struct([struct2cell(sysInfoEXP);struct2cell(sysInfoSMS)],[fieldnames(sysInfoEXP);fieldnames(sysInfoSMS)]);
    sysInfo.dataBankName = "dataBank_EXAMPLE";
%-----------------------------------------------------------------------------------------------------------------------------%
```

Where "expSpec_EXAMPLE" , "solverModelSpec_EXAMPLE" , "dataBank_EXAMPLE" can be used as EXAMPLES throughout all scripts. For custom files, the example files can be copied as a template and then renamed in as "exeSpec_DATA1" , etc. or however the template files got renamed; and then again simply change the name from EXAMPLE to DATA1 or whichever name was chosen in each script as needed.

4.1 Parameter Database Function – dataBank.m

This function will be the database for all membrane and component parameters. There is functionality for custom polymers to be added by copy and pasting the "if...then" statement that checks for a certain polymer name as a template.

See figure below for visual aid:



```
%----------------------------------------------------------------------------------------------------------------
%full parameter/property sets
    if contains(sysInfo.memID,'SBAD1')
      % Lively/Ronita's 9-comp data TOL/MCH/1MN/DCN/NOC/IOC/TBB/TPB/ICE/SBAD-1
        params.lmem = 0.3; % thickness of active membrane layer um
        params.compID = struct('TOL', 1, 'MCH', 2, 'MNP', 3, 'DEC', 4, 'NOC', 5, 'IOC', 6, 'TBB', 7, 'TPB', 8, 'ICT',
        params.n = 9;
        params.Vs = [106.521;128.123;139.823;156.962;163.42;165.552;155.529;240.069;293.267;62326]; %cm3/mol
        params.HanSolParam = [18,1.4,2;16,0,1;20.6,0.8,4.7;18,0,0;15.5,0,0;14.1,0,0;17.4,0.1,1.1;18,0,0.6;16.3,0,0];
        params.psat = [28.998;46.596;0.059;0.975;14.805;49.087;2.115;0.0352;0.0458];%torr

        if contains(sysInfo.memPhaseModel,'F-H')
            params.chis_im = [0.839;1.672;0.705;2.785;1.164;3.046;1.648;2.5;3.128];
            if sysInfo.lowDiffErrorBar == 0
                params.diffs_im = [16.0768;5.84;0.0555;1.7532;5.9896;7.1845;1.4710;0.0726;0.6182]; %um2/s FH Exp Base
            elseif sysInfo.lowDiffErrorBar == 1
                params.diffs_im = [14.7187;2.9989;0.0442;1.3636;4.6466;4.3297;1.0233;0.0726;0.5624]; %Mixed Exp (LOW
            end
            if sysInfo.memPhaseModel_FicksOG == 1
                if sysInfo.lowDiffErrorBar == 0
                    params.diffs_im = [3.7303;3.4818;0.0098;1.4674;2.4617;6.2724;0.8733;0.0582;0.5455]; %um2/s FICKS
                elseif sysInfo.lowDiffErrorBar == 1
                    params.diffs_im = [3.4152;1.7879;0.0078;1.1414;1.9079;3.7801;0.6075;0.0582;0.4963]; %um2/s FICKS
                end
```

Note the "dataBank.m" function is also separated by sorption model since for any given membrane, there will be different sportion parameters associated with each model. Then with each sorption model, there will also be different Maxwell-Stefan (MS) penetrant-membrane diffusivites to be fit for each model. The order of magnitudes are the same but their values will vary.

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

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

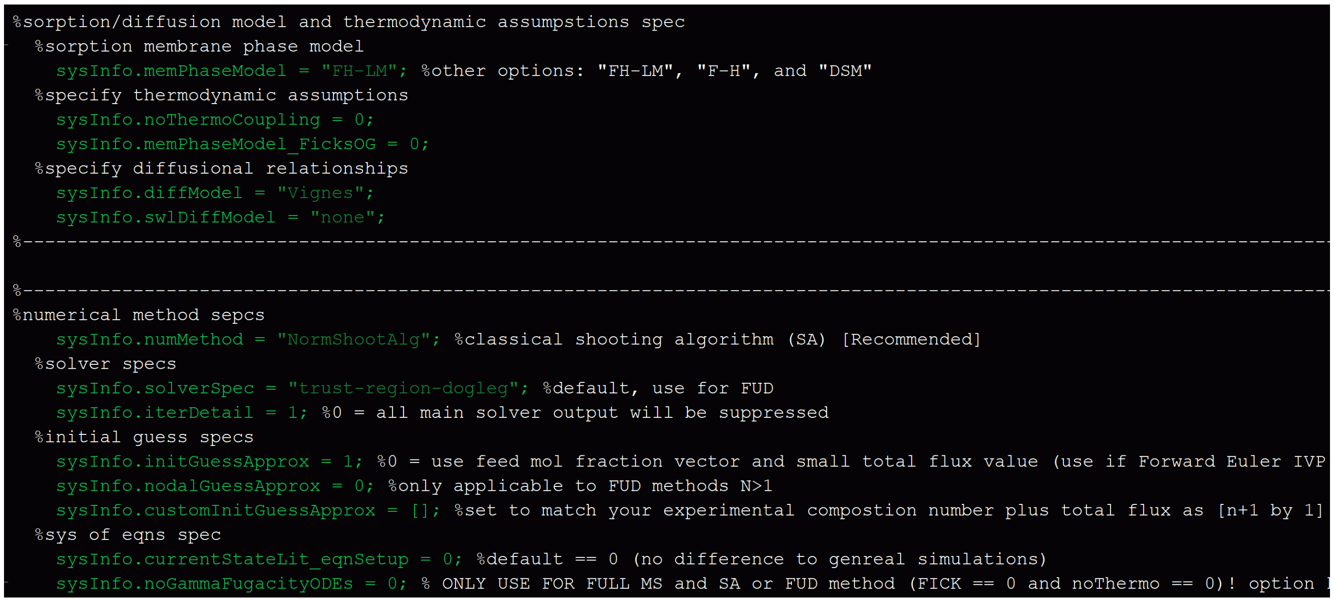
- params.lmem – thickness of active layer (phsae II), μm
- params.compID – struct defining component name and order of parameters to be defined in subsequent parameter set definitions
- params.n – total number of components which parameters are defined below for (equal to number of components listed in "params.compID")
- params.Vs – penetrant pure comonent molar volumes, cm^{3}mol^{-1}, and V_m is the membrane molar volume that is assumed to be equal to a large estimate $\bar{V}_m = 62,326$ cm^{3}mol^{-1} based on polymer weight average molecular weight and bulk density
- params.HanSolParam – Hansen solubility parameters (dispersion, intermolecular forces, and hydrogen bonding), note $= [\delta_{D1}, \delta_{P1}, \delta_{H1}; ...; \delta_{Dn}, \delta_{Pn}, \delta_{Hn}]$ where n is equal to params.n, MPa$^{0.5}$
- params.psat – permeant vapor pressures evaluated at system T, torr
- params.chis – **Flory-Huggins or combined Flory-Huggins-Langmuir sorption model** χ_{im} where m refers to the membrane component and $\chi_{ij} = 1$ are placeholders until relevant Hansen solubility parameter relationships are applied

- params.bs – **Dual-mode or Flory-Huggins-Langmuir sorption** Langmuir affinity constants, torr^{-1}
- params.Ch – **Dual-mode or Flory-Huggins-Langmuir sorption** volume based Langmuir free volume capacities, mol (cm^3 polymer)$^{-1}$
- params.ks – **Dual-mode sorption** Henry's type sorption parameter, (cm^3 solvent)(cm^3 polymer)$^{-1}$torr^{-1}
- params.diffs Maxwell-Stefan penetrant-membrane $Đ_{im}$ and $Đ_{ij} = 1$ as placeholder until cross-diffusivity relationship is applied or user-specified cross-diffsivity $Đ_{ij}$ are given, μm$^2 s^{-1}$; also note there are 4 sets of diffusivities for one given soprtion model (Normal MS, lowFluxError MS, Normal Fick's law, and lowFluxError Fick's law) and if doing error calc

With all these parameter pieces in place, one can move on to the numerical method specifications and then finally describing the system conditions, and mixture/polymer type to simulate the full local transport model.

4.2 Numerical Method and Sorption/Diffusion Model Specifications – solverMethodSpec.m

This section deals with the more technical specifications of how to solve the local flux complex mixture modeling framework presented in the accompanying document "MODEL_AND_METHOD_DOC.pdf". Here is the initial code interface presented below that is an accompanying input file.



```
%sorption/diffusion model and thermodynamic assumpstions spec
  %sorption membrane phase model
    sysInfo.memPhaseModel = "FH-LM"; %other options: "FH-LM", "F-H", and "DSM"
  %specify thermodynamic assumptions
    sysInfo.noThermoCoupling = 0;
    sysInfo.memPhaseModel_FicksOG = 0;
  %specify diffusional relationships
    sysInfo.diffModel = "Vignes";
    sysInfo.swlDiffModel = "none";
%-----------------------------------------------------------------------------------------------------------------------

%-----------------------------------------------------------------------------------------------------------------------
%numerical method sepcs
    sysInfo.numMethod = "NormShootAlg"; %classical shooting algorithm (SA) [Recommended]
  %solver specs
    sysInfo.solverSpec = "trust-region-dogleg"; %default, use for FUD
    sysInfo.iterDetail = 1; %0 = all main solver output will be suppressed
  %initial guess specs
    sysInfo.initGuessApprox = 1; %0 = use feed mol fraction vector and small total flux value (use if Forward Euler IVP
    sysInfo.nodalGuessApprox = 0; %only applicable to FUD methods N>1
    sysInfo.customInitGuessApprox = []; %set to match your experimental compostion number plus total flux as [n+1 by 1]
  %sys of eqns spec
    sysInfo.currentStateLit_eqnSetup = 0; %default == 0 (no difference to genreal simulations)
    sysInfo.noGammaFugacityODEs = 0; % ONLY USE FOR FULL MS and SA or FUD method (FICK == 0 and noThermo == 0)! option
```

First the sorption and diffusion models can be specified.

The following choices are for "sorption membrane phase model" where **sysInfo.memPhaseModel** can equal ("F-H", "DSM", "FH-LM") which stand for classical Flory-Huggins, Dual-mode sorption, or our novel combined Flory-Huggins-Langmuir. Model derivations can be found in "MODEl_AND_METHOD_DOC.pdf". Note for some scripts a vector **sysInfo.modSpec** will be specified instead to test different sorption models at once. See diffusivity fitting script and single component prediction script EXAMPLE inputs for direct use.

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

Lastly are the specifications of the membrane swelling diffusivity model where **sysInfo.swlDiffModel** can equal ("none", "FFV", "Avg-Diff") where "none" uses constant diffusivities, "FFV" uses the fractional free volume novel diffusion model presented in the only Mathias et. al. paper recerended in the "End User Statement" section. Then "Avg-Diff" utilizes the novel sorption-vection average cohort style diffuson presented in the same paper referenced above.

Looking at the methods section in the supplemental document provided, one can specify which method they would like to to simulate the local flux transport where **sysInfo.numMethod** can equal ("NormShootAlg", "FullDis", and "MultShootAlg"). See included model and method document for extended details but know the first is the single shooting point algorithm, second is the full discritization style solve, and the last is a multiple shooting point numerical method.

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

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

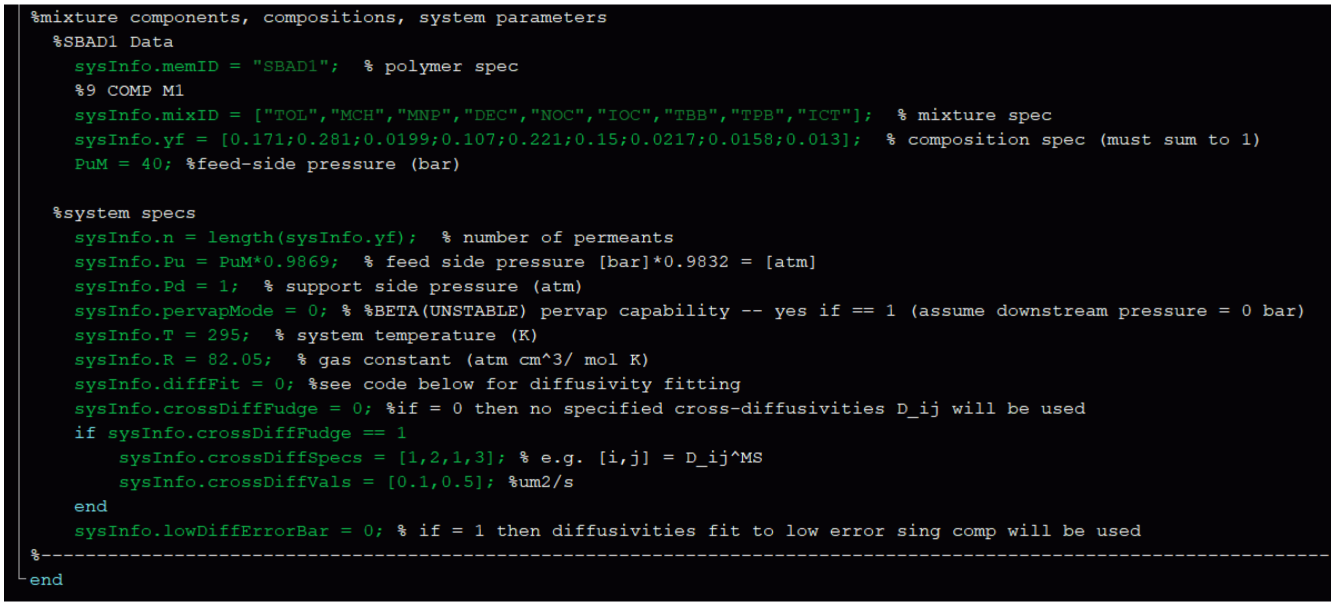
The **recommended method** is our shooting algorithm but for a quick estimate that is probably within less than 20% error in most cases(full publication will detail differences), full discritization is recommended. For strong diffusional coupling ($Đ_{ij} << 1$) and high component number ($n \geq 9$), multiple shooting point algorithm is recommended with number of shooting points starting at 5 and increasing number if needed.

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

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

4.3 Experimental Specifications – expSpec.m

The last required input file is the most pertinent to the actual experiment or industrial process stream one is trying to simulate looks something like this:



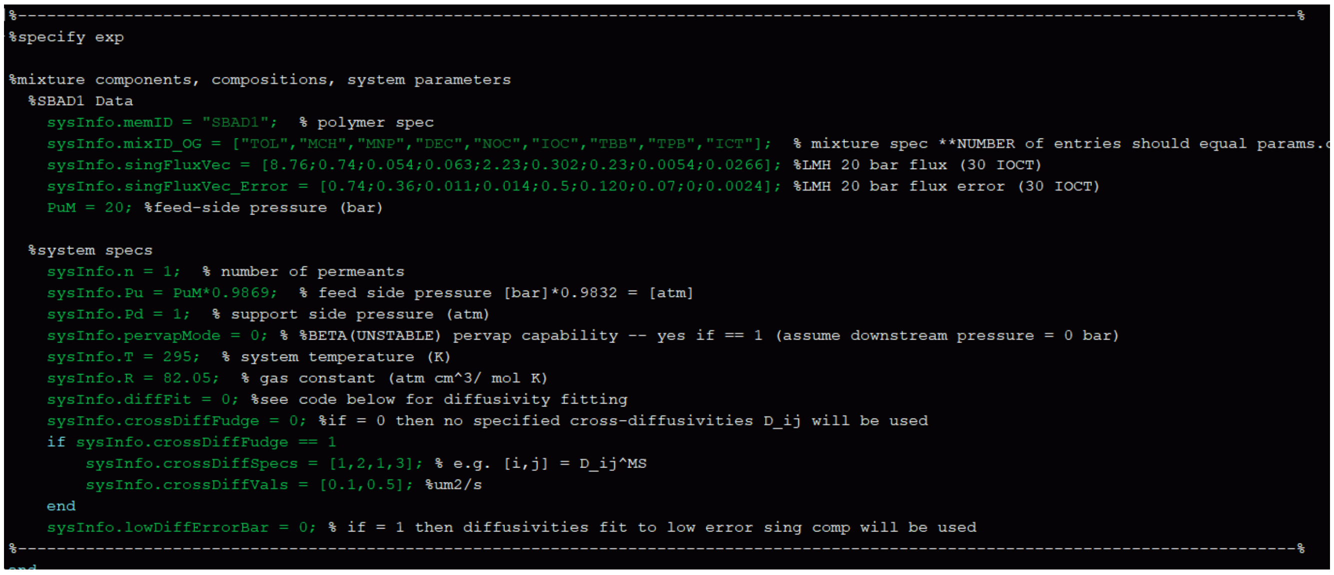
```
%mixture components, compositions, system parameters
  %SBAD1 Data
    sysInfo.memID = "SBAD1";  % polymer spec
    %9 COMP M1
    sysInfo.mixID = ["TOL","MCH","MNP","DEC","NOC","IOC","TBB","TPB","ICT"];  % mixture spec
    sysInfo.yf = [0.171;0.281;0.0199;0.107;0.221;0.15;0.0217;0.0158;0.013]; 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

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

Where the lower "system specs" section is usually constant and self explanitory across the different scripts, the top section will change and is to be explained in the later sections.

5 Diffusivity Fitting Script – diffFit_asyMemLocal.m

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



```
%--------------------------------------------------------------------------------------------------------------------%
%specify exp

%mixture components, compositions, system parameters
  %SBAD1 Data
    sysInfo.memID = "SBAD1";  % polymer spec
    sysInfo.mixID_OG = ["TOL","MCH","MNP","DEC","NOC","IOC","TBB","TPB","ICT"];  % mixture spec **NUMBER of entries should equal params.
    sysInfo.singFluxVec = [8.76;0.74;0.054;0.063;2.23;0.302;0.23;0.0054;0.0266]; %LMH 20 bar flux (30 IOCT)
    sysInfo.singFluxVec_Error = [0.74;0.36;0.011;0.014;0.5;0.120;0.07;0;0.0024]; %LMH 20 bar flux error (30 IOCT)
    PuM = 20; %feed-side pressure (bar)

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

Again the "singFluxVec" and "singFluxVec_error" are to be the exact size and component order and the defined "sysInfo.mixID_OG" is set to.

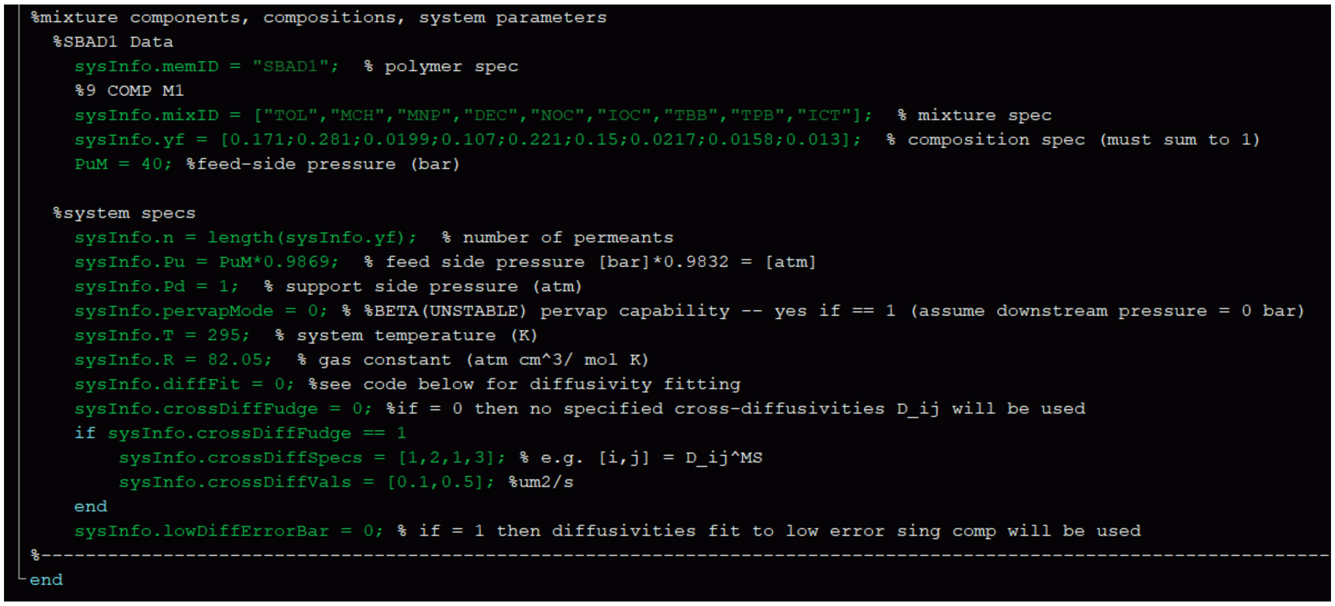
As far as the solverModelSpec.m, the inputs are the same when described in section 4.2. THe only difference is if only fitting one model, then "modSpec" is as defined above. For all available sorption models use "modSpec = [1 2 3]". This is in contrast to only defining one sorption model to use for the other scripts.

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

6 Single Mixture Local Flux Simulation Script – singMixSim_asyMemLocal.m

This script is used when one wants to look at a single mixture local flux membrane permeation simulation. Since this script is too long to fit the overall script in one figure, each section will be went through.

The input file "expSpec.m" looks like:



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

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

deals with defining the single mixture (or if "sysInfo.n = 1" then single component) to be simulated.

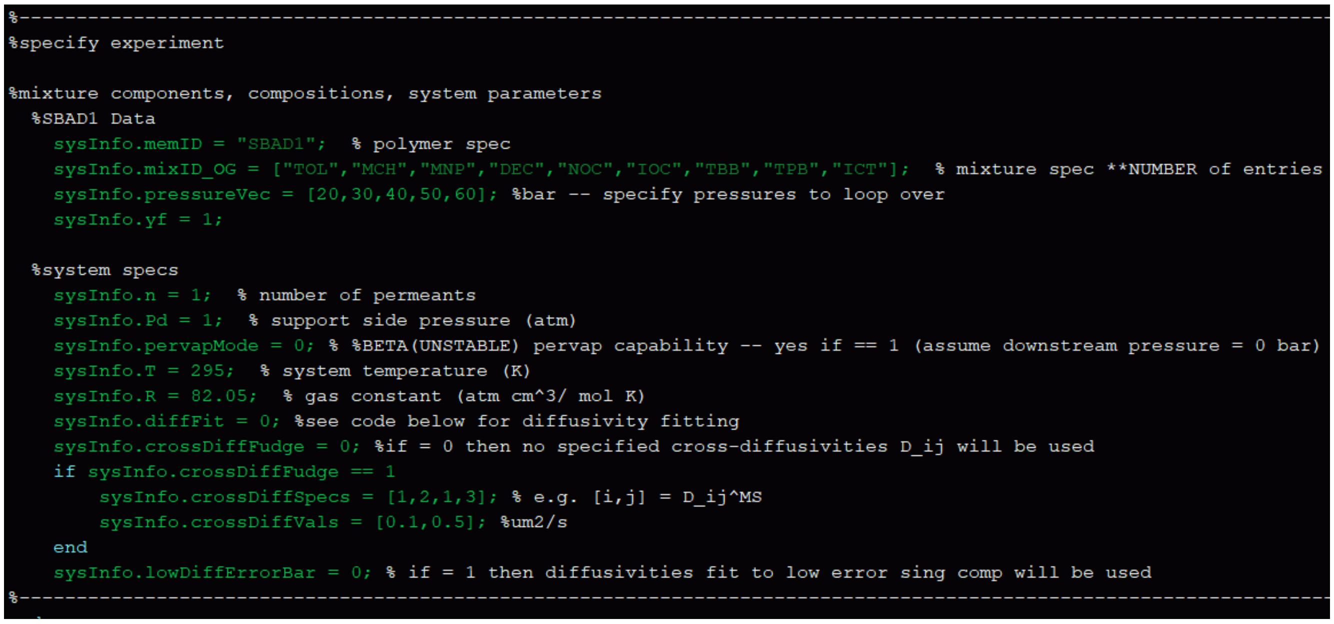
Then the method and modeling input file looks the same as shown in section 4.2.

The main outputs are the localCompFlux and partialFlux vectors that contain the phase III molar compositions and total volumetric flux through the membrane (in liters per square meter per hour or LMH) and then the partial fluxes of each permeant are also given in the same units.

7 Single Component Mixture Permeation Prediction Script – singCompLoop_asyMemLocal.m

This script is a simple loop of the single component permeation simulation defined above for different sorption models and various pressures to asses prediction capabilities of models from single component permeation experimental diffusivity fits.

Here is the "expSpec.m" file



```
%-----------------------------------------------------------------------------------------------------------
%specify experiment

%mixture components, compositions, system parameters
  %SBAD1 Data
    sysInfo.memID = "SBAD1";  % polymer spec
    sysInfo.mixID_OG = ["TOL","MCH","MNP","DEC","NOC","IOC","TBB","TPB","ICT"];  % mixture spec **NUMBER of entries
    sysInfo.pressureVec = [20,30,40,50,60]; %bar -- specify pressures to loop over
    sysInfo.yf = 1;

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

And the method and model spec file looks the same as the diffusivity fitting script method and model spec file.

Again the variable comments and structure should not need more explanation.

The outputs are matrices detailing predicted fluxes for each column to be at the given pressure defined from the **sysInfo.pressureVec** and then each component defined from the **sysInfo.mixID_OG** and **sysInfo.memID** string vector. Then the given matrix has n component partial fluxes for each sorption model. So if three sorption models are specified then the output matrices will contain $3n$ rows.

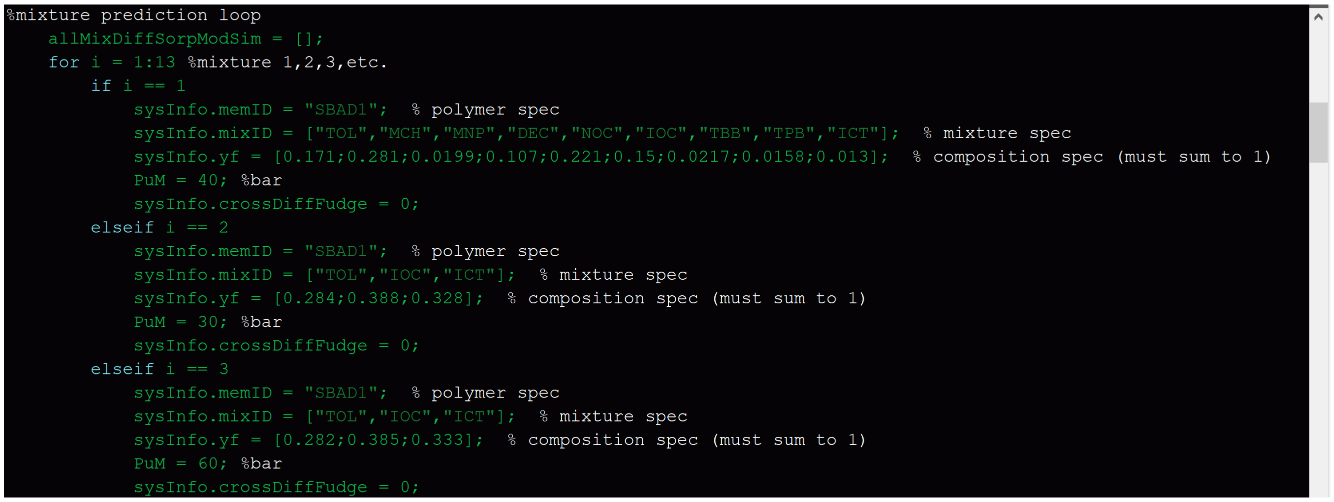
Included is an example experimental data and simulated data file titled "Example_Data.xlsx" for provided example input files to see how the output matrices look like if the data is separated out.

8 Multiple Mixture/Model Permeation Prediction Script – mixLoop_asyMemLocal.m

This script mimics the last loop presented but instead of varying the single component pressures, each defined mixture, sorption model, diffusion models, and thermodynamic assumptions are looped through to give a comprehensive modeling data. The inputs are also similar and less complicated compared to the others. Refer to examples for quick details.

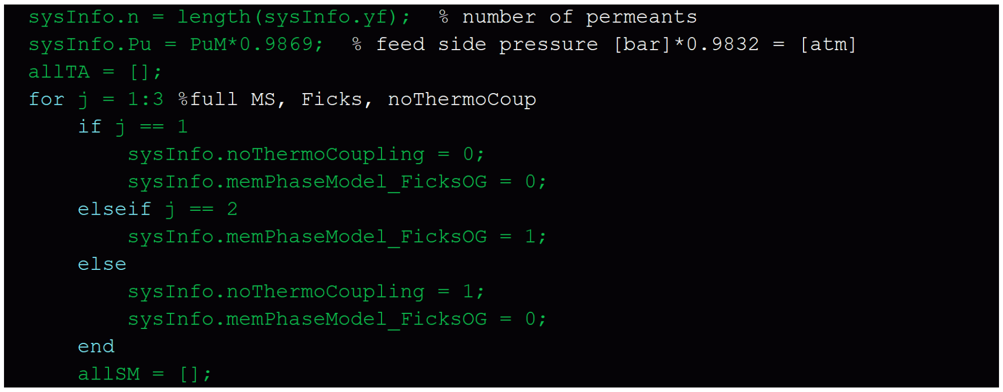
In order to loop over specific models, simply change the for loop parameters from say "1:12" to "[1 4 6 12]" or "2:6" or etc. to pick and choose.

Again this script is long so will be explained through "for...loop" sections. The first defines which mixtures to be looped over as:



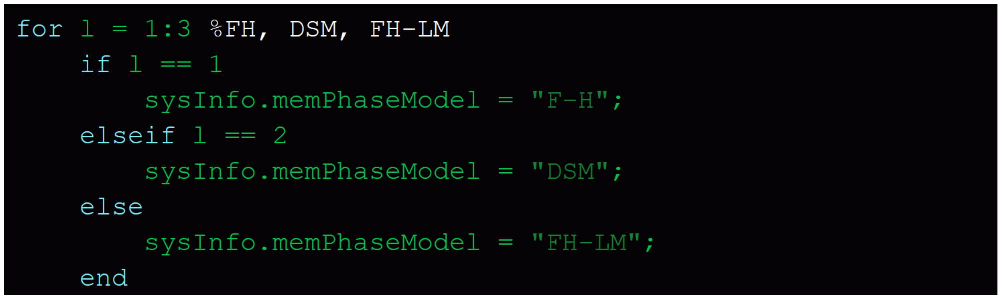
```
%mixture prediction loop
    allMixDiffSorpModSim = [];
    for i = 1:13 %mixture 1,2,3,etc.
        if i == 1
            sysInfo.memID = "SBAD1";  % polymer spec
            sysInfo.mixID = ["TOL","MCH","MNP","DEC","NOC","IOC","TBB","TPB","ICT"];  % mixture spec
            sysInfo.yf = [0.171;0.281;0.0199;0.107;0.221;0.15;0.0217;0.0158;0.013];  % composition spec (must sum to 1)
            PuM = 40; %bar
            sysInfo.crossDiffFudge = 0;
        elseif i == 2
            sysInfo.memID = "SBAD1";  % polymer spec
            sysInfo.mixID = ["TOL","IOC","ICT"];  % mixture spec
            sysInfo.yf = [0.284;0.388;0.328];  % composition spec (must sum to 1)
            PuM = 30; %bar
            sysInfo.crossDiffFudge = 0;
        elseif i == 3
            sysInfo.memID = "SBAD1";  % polymer spec
            sysInfo.mixID = ["TOL","IOC","ICT"];  % mixture spec
            sysInfo.yf = [0.282;0.385;0.333];  % composition spec (must sum to 1)
            PuM = 60; %bar
            sysInfo.crossDiffFudge = 0;
```

Then for the next loop, each thermodynamic assumption is looped over



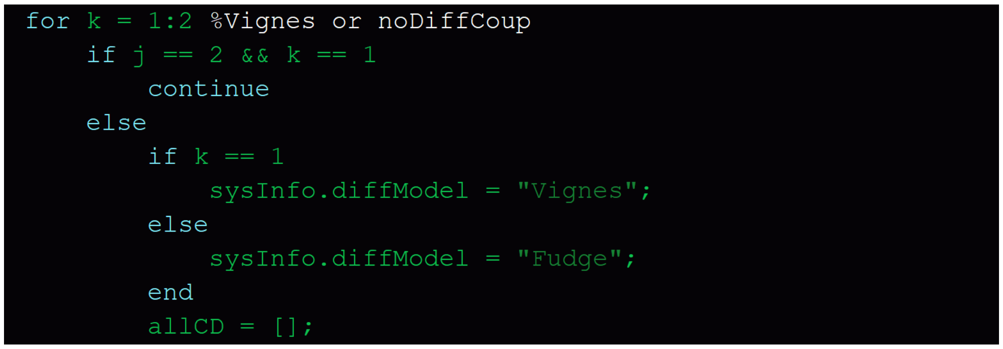
```
sysInfo.n = length(sysInfo.yf);  % number of permeants
sysInfo.Pu = PuM*0.9869;  % feed side pressure [bar]*0.9832 = [atm]
allTA = [];
for j = 1:3 %full MS, Ficks, noThermoCoup
    if j == 1
        sysInfo.noThermoCoupling = 0;
        sysInfo.memPhaseModel_FicksOG = 0;
    elseif j == 2
        sysInfo.memPhaseModel_FicksOG = 1;
    else
        sysInfo.noThermoCoupling = 1;
        sysInfo.memPhaseModel_FicksOG = 0;
    end
    allSM = [];
```

Then each sorption model is simulated



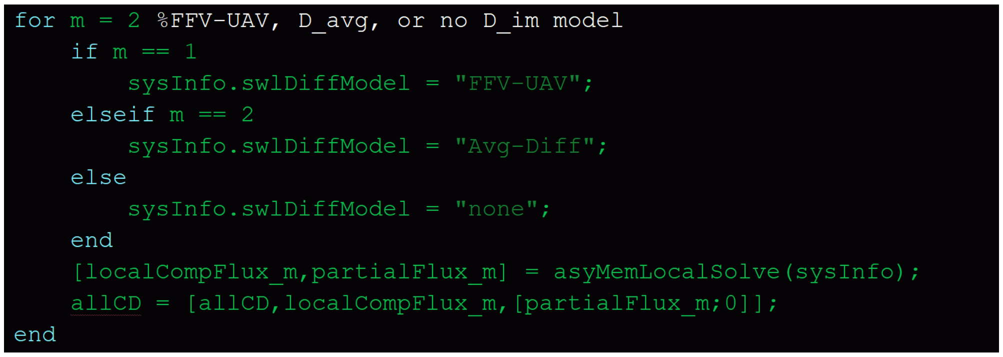
```
for l = 1:3 %FH, DSM, FH-LM
    if l == 1
        sysInfo.memPhaseModel = "F-H";
    elseif l == 2
        sysInfo.memPhaseModel = "DSM";
    else
        sysInfo.memPhaseModel = "FH-LM";
    end
```

And next, each cross-diffusion model is looped



```
for k = 1:2 %Vignes or noDiffCoup
    if j == 2 && k == 1
        continue
    else
        if k == 1
            sysInfo.diffModel = "Vignes";
        else
            sysInfo.diffModel = "Fudge";
        end
        allCD = [];
```

And finally, each membrane concentration dependent diffuusivity model is looped through.



```
for m = 2 %FFV-UAV, D_avg, or no D_im model
    if m == 1
        sysInfo.swlDiffModel = "FFV-UAV";
    elseif m == 2
        sysInfo.swlDiffModel = "Avg-Diff";
    else
        sysInfo.swlDiffModel = "none";
    end
    [localCompFlux_m,partialFlux_m] = asyMemLocalSolve(sysInfo);
    allCD = [allCD,localCompFlux_m,[partialFlux_m;0]];
end
```

The method and model spec file only need the method spec as input. See previous figures or EXAMPLE file for details.

Since the output is going to be a vary large matrix, I am including the organized simulated data to provide a template to figure out how to trace the for loops to what the data is actually representing. This table is what was used in our recent publication so it reflects current mixture loop specifications. See file "Example_Data_Error_Analysis.xlxs" and workbook titled "SIM DATA".