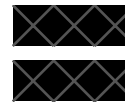




NUMERIC MODELLING OF LIGHT ACTIVATED TiO_2 MEMBRANE FILTERS

Niels Hakkert
Max Kwak



CHEMICAL ENGINEERING

Under supervision of:
Prof.Dr.Ir. R.G.H. Lammertink

2 May 2022

Contents

1. Mass Transfer Model	1
1.1. Derivation of Balances	1
1.2. Boundary Conditions.....	2
2. Membrane model function.....	4
2.1. Choice of ODE Solver	4
2.2. ODE Solver Function	4
3. Membrane Model Variations and fitting	7
3.1. General considerations	7
3.2. MMV1: Choosing an experiment to fit a value of K and AC for	9
Fitting	9
Comparing.....	11
Sensitivity analysis	12
3.3. MMV2: Fitting a value for k and AC for each experiment	14
3.4. MMV3: Fitting 1 value for k and AC to all experiments.....	15
4. Overview of results	16
Conclusion & discussion.....	22
Possible improvements.....	22
Bibliography	23

1. Mass Transfer Model

1.1. Derivation of Balances

The model of the photocatalytic membrane is split into two distinct regions:

Region 1: Liquid phase with only advection and diffusion.

$$(1) \quad u_1 \frac{dc_1}{dx} - D_1 \frac{d^2c_1}{dx^2} = 0 \quad \text{for } (0 < x < L1)$$

Region 2: Membrane phase which also takes the reaction into account.

$$(2) \quad u_2 \frac{dc_2}{dx} - D_1 \frac{d^2c_2}{dx^2} + kc_2 = 0 \quad \text{for } (0 < x < L2)$$

In these balances L1 is the length of the boundary layer and L2 is the thickness of the membrane. Both phases start at $x = 0$ due to the solver being able to concurrently solve both these balances over the same x mesh, which is useful when both regions are normalized to their length as done later in the derivation. Consider that k (1/s) can be represented as $k'S$ where k' (m/s) is the surface reaction rate and S (1/m) is the specific surface area of the membrane. Additionally, the following relations are considered in the shift from bulk liquid to the membrane phase assuming a constant cross-sectional area:

$$(3) \quad u_2 = \frac{u_1}{\varepsilon}$$

Where u is the velocity of the fluid and ε is the porosity of the membrane.

$$(4) \quad c_2 = c_1 \alpha$$

Here it is assumed that the concentration on the membrane side of the interface is equal to c_1 multiplied by a retention factor α , which can be determined experimentally.

$$(5) \quad D_2 = \varepsilon D_1 \left(1 + \left(\frac{u_2 r}{D_1} \right)^2 / 48 \right)$$

Which is based on Taylor dispersion through pores of radius r and effect of porosity on the diffusion. However, since this model is used with low flow velocities the second term is assumed to be negligible which simplifies equation 6 to:

$$(6) \quad D_2 = \varepsilon D_1$$

Taking these assumptions into account, the balances are made dimensionless by normalizing both phases to their respective lengths ($x = X * L$) and the concentration to the incoming concentration ($c = C * c_0$) the following equations are obtained:

$$(7) \quad Pe_1 \frac{dC_1}{dX} - \frac{d^2C_1}{dX^2} = 0 \quad \text{for } (0 < X < 1)$$

$$(8) \quad Pe_2 \frac{dC_2}{dX} - \frac{d^2C_2}{dX^2} + \frac{k'SL_2^2}{D_2} C_2 = 0 \quad \text{for } (0 < X < 1)$$

Considering k' is a depth dependent reaction rate based on light absorption by the photocatalyst k' can be expressed as followed:

$$(9) \quad k' = k'_0 \exp(-X * L_2 * AC)$$

Where AC ($1/m$) is the absorption coefficient of the photocatalyst, and the characteristic absorption length is defined as $1/AC$.

1.2. Boundary Conditions

The inlet boundary condition can simply be set to a constant concentration, and since the concentration is normalized on the inlet concentration the first boundary condition is defined as follows:

$$(10) \quad C_1 = 1 \quad \text{at } (X_{\text{fluid}} = 0)$$

The second boundary condition arises when you consider that at any plane in the reactor the mass transfer rate must be equal to the feed rate, this results in the following two equations at the interface between the bulk fluid and the membrane:

$$(11) \quad u_1 c_0 = u_1 C_1 - D_1 \frac{dc_1}{dx_{\text{fluid}}} \quad \text{at } (x_{\text{fluid}} = L_1)$$

$$(12) \quad u_1 c_0 = u_2 C_2 - D_2 \frac{dc_2}{dx_{\text{mem}}} \quad \text{at } (x_{\text{mem}} = 0)$$

Making both these equations dimensionless results in:

$$(13) \quad 1 = C_1 - \frac{1}{Pe_1} \frac{dC_1}{dX_{\text{fluid}}} \quad \text{at } (X_{\text{fluid}} = 1)$$

$$(14) \quad \varepsilon = C_2 - \frac{1}{Pe_2} \frac{dC_2}{dX_{\text{mem}}} \quad \text{at } (X_{\text{mem}} = 0)$$

Equating these two formulae to each other results in:

$$(15) \quad \varepsilon \left(C_1 - \frac{1}{Pe_1} \frac{dC_1}{dX_{\text{fluid}}} \right) = C_2 - \frac{1}{Pe_2} \frac{dC_2}{dX_{\text{mem}}} \quad \text{at } (X_{\text{fluid}} = 1 \text{ and } X_{\text{mem}} = 0)$$

Similarly, this can be done if you want to include the surface reaction term in the boundary condition. This term is then added to the second balance:

$$(16) \quad u_1 c_0 = u_2 C_2 - D_2 \frac{dc_2}{dx_{\text{mem}}} + k' c_2 (1 - \varepsilon) \quad \text{at } (x_{\text{mem}} = 0)$$

When made dimensionless using equation (3) and (6), and equated to the first balance:

$$(17) \varepsilon \left(C_1 - \frac{1}{Pe_1} \frac{dC_1}{dX_{fluid}} \right) = C_2 - \frac{1}{Pe_2} \frac{dC_2}{dX_{mem}} + \frac{k'}{u_2} C_2 (1 - \varepsilon)$$

at ($X_{fluid} = 1$ and $X_{mem} = 0$)

The third boundary condition also occurs at this interface but is a concentration (dis)continuity, this was also described in equation (4) and is written as:

$$(18) C_2 = C_1 \alpha \quad \text{at } (X_{fluid} = 1 \text{ and } X_{mem} = 0)$$

The last boundary condition at the outlet is simply a steady state declaration. As Danckwerts formulated [1]: if dc/dx were positive at the end of the membrane, the concentration would pass a minimum somewhere before rising again, and if dc/dx were negative, the concentration would be higher at the end of the membrane than earlier in the membrane. Both situations seem unlikely, so intuitively the dc/dx should be set to 0:

$$(19) \frac{dC}{dX_{mem}} = 0 \quad \text{at } (X_{mem} = 1)$$

2. Membrane model function

2.1. Choice of ODE Solver

For this model it was decided to use the bvp5c solver. The ODE solver in MATLAB that were considered were ode45 and its variants, and bvp4c/bvp5c. ode45 and its variants can only solve initial value problems and are thus not relevant for this model. This left the choice between bvp4c and bvp5c there are a few small differences between these two, but since bvp5c solves the algebraic equations directly, whereas bvp4c uses analytical condensation it was decided to use bvp5c.

2.2. ODE Solver Function

The ODE Solver Function is a function called by the control script, which is explained in more detail in the next heading. When the control script calls this function it passes the following variables to the solver:

```
%Input constants:

%F1 = Flowrate (g/h)
%L1 = Boundary layer length/thickness (m)
%L2 = Membrane length/thickness (m)
%rad = Membrane radius (m)
%P = Porosity of the membrane
%k = Surface reaction rate (m/s)
%alpha = Concentration ratio at boundary
%D1 = Diffusion coefficient in the liquid (m2/s)
%AC = Absorption coefficient (1/m)
%S = Specific surface area of the catalyst (m2/m3)
%plots = a boolean to indicate whether plots will be outputted or not
%feedback = a boolean to indicate whether stats are outputted from the bvpsolver
%boundaryreaction = a boolean to indicate whether the reaction flux at the boundary will be taken into
account in the flux balance
```

First some other constants used in the ODE solver are calculated:

```
%Conversion of input constants

Area = (rad^2)*pi;
u1 = (F1*(1e-6/3600))/Area;
u2 = u1/P;
D2 = D1*P;
Pe_1 = u1*L1/D1;
Pe_2 = u2*L2/D2;

%Surface area membrane (m^2)
%Velocity pre-membrane (m/s)
%Velocity in membrane (m/s) Assuming cylindrical pores
%Diffusion coefficient membrane (m2/s)
%Peclet number before membrane
%Peclet number in membrane
```

Now the ODE solver generates a x mesh from 0 to 1, where both the fluid phase and the membrane phase are solved concurrently and initializes the ODE function on this mesh. The initial guesses are: $C1 = 1$ at $X_{\text{fluid}} = 0$, $dC1/dX = 0$ at $X_{\text{fluid}} = 0$, $C2 = 1$ at $X_{\text{mem}} = 0$ and $dC2/dX = 0$ at $X_{\text{mem}} = 0$. Then depending on the feedback boolean, the ODE is solved with or without stats being outputted:

```
%Define the xmesh and its data point density the BVPsolver will use
xmesh = linspace(0,1,101);

%initialize a solution on the xmesh (c = 1 on both the start of the liquid and membrane and dc/dx is 0
on both)
solinit = bvpinit(xmesh, @guess);

%bvp5c to solve this boundary value problem

sol = bvp5c(@bvpfcn, @bcfcn, solinit, options);
```

The differential equations used to solve this boundary value problem are described in the function below. Here the concentration is expressed in an array of y where y(1) is C1, y(2) = dC1/dX, y(3) = C2 and y(4) = dC2/dX. The function then expresses the dC/dX and d^2C/dX^2 of both concentrations in an array of length 4 where the array dydx(1) describes dC1/dX, dydx(2) describes d^2C1/dX^2 , dydx(3) and dydx(4) do the same but now for the C2. The function used for the boundary conditions is also an array of length 4 where each element is calculated to be zero. So as shown earlier in equation 11 the first boundary condition is $C1 = 1$ at $X_{\text{fluid}} = 0$, which is expressed here as $ya(1) - 1 = 0$. Here, ya is an array of length 4 which describes both the inlet side of the fluid and the membrane and ya(1) and ya(3) describe the C of both phases. And ya(2) and ya(4) describe the dC/dX of both phases. Similarly, yb is an array of length 4 which describes the concentration in the outlet of both the fluid and the membrane phase.

```
%Define the differential equations of this system

function dydx = bvpfcn(x,y)
    dydx = [y(2)                %C1' = dC1/dX
            Pe_1*y(2)           %C1'' = Pe_1*dC1/dX
            y(4)                %C2' = dC2/dX
            Pe_2*y(4)+(k*exp(-x*L2*AC))*S*L2^2/D2*y(3)];
    %^C2'' = Pe_2*y(4)+((k*exp(-X*L2/LDL)*S*L2^2)/D2)*C2
end

%Define the boundary conditions of this system depending on boundaryreaction flag
function res = bcfcn(ya,yb)
    if boundaryreaction == false
        res = [ya(1) - 1                %Inlet concentration = 1
                P*(yb(1)-(1/Pe_1)*yb(2)) - (ya(3)-(1/Pe_2)*ya(4)) %Flux continuity at membrane surface
                ya(3)-yb(1)*alpha        %Concentration continuity at membrane surface
                yb(4)];                  %dC2/dX = 0 at the end of the membrane
    else
        res = [ya(1) - 1                %Inlet concentration = 1
                P*(yb(1)-(1/Pe_1)*yb(2)) - (ya(3)-(1/Pe_2)*ya(4)+(k/u2)*ya(3)*(1-P)) %Flux continuity at membrane surface (including reaction)
                ya(3)-yb(1)*alpha        %Concentration continuity at membrane surface
                yb(4)];                  %dC2/dX = 0 at the end of the membrane
    end
end
```

Lastly, parts of the solution are saved to the array `solpoints` which will be returned to the caller of the function and depending on the “plots” boolean a plot of the concentration distribution over the system is made.

```
%These solutions are returned to the script that calls this function
solpoints = [sol.y(1,1),sol.y(1,end),sol.y(3,1),sol.y(3,end)];

%Check for plots flag and output plots if 1
if plots == true
    figure('Name' , 'Danckwerts'); hold on; grid on;
    plot(sol.x, sol.y(1,:), 'blue',Linewidth=3) %plot of the concentration profile fluid
    plot([sol.x(end),sol.x(end)], [sol.y(1,length(sol.y)),sol.y(3,1)], '--
g',Linewidth=3) %plot of the concentration gap (if present)
    plot((sol.x+1), sol.y(3,:), 'red',Linewidth=3) %plot of the concentration profile membrane
    legend(' C in Boundary layer', ' Concentration gap', ' C in Membrane', 'location', 'SouthWest')
    xlabel({'Dimensionless axial coordinate', 'X = x/L1 for 0<X<1 and X = x/L2 for 1<X<2'});
    ylabel({'Dimensionless concentration'; 'C = c/c_i_n'});
    title(['Concentration profile at F1 = ' num2str(F1) ' mL/h'])
    ylim([0 1]); xlim([0 2])
    h = gca; set(h, 'fontsize', 24, 'FontName', 'GillSansMT');
    hold off
end
```

The solution array is created by firstly taking the ingoing concentration (`sol.y(1,1)`) at the beginning of the system and the concentration at the end of region 1 (`sol.y(1,end)`), being the concentration at the liquid-membrane interface on the liquid side. Next to this, the concentration at the entrance of the membrane (`sol.y(3,1)`), being the concentration at the liquid-membrane interface on the membrane side, and the concentration at the end of the membrane (`sol.y(3,end)`), or the outgoing concentration are taken.

3. Membrane Model Variations and fitting

In this chapter, the three Membrane Model Variables (MMV) scripts are considered and elaborated upon. A lot of parts are already explained within the script using comments, so not everything is elaborated fully. For the first time running the scripts, it is recommended to not touch any of the script control settings, as they are set to yield the plots the quickest, using previously determined values, without re-fitting every value of AC and k0, as this might take a considerable amount of time (5-10 minutes for MMV2 and MMV3), without yielding any new results if the data has not changed.

3.1. General considerations

There are some parts of the scripts that are present in each of the MMV scripts, which will be explained first. Using `DefaultFigureWindowState` docks all figures to the editor window. All figures can then be undocked at once and kept together, which is especially helpful when plotting a lot of figures. The global variables are then declared, which can be found distributed over different lines to easily distinguish them and see where they are used.

The script control is one of the most important parts of each script. It allows us to tune the output of the script to what is desired at that point. If you want to generate one plot with a picked value for the reaction rate (k) and possibly other constants, `basecaseonly` can be set to `true`. This will then only plot the outgoing concentration as a function of the flow rate for the given parameters. If you want to compare the base case to a set of values from one of the datasets as well, `comparebasecase` can be left on, but it can also be turned off if desired.

“plots” gives the option to plot the profiles within the system for the different flowrates. This option should be enabled with care, as it should only be used if no data is being fit. Otherwise, it will return all profiles at every flowrate for every iteration of the fit, leading to an unmanageable number of figures. Ideally, it should only be used if `basecaseonly` can be set to `true`.

```
clear all; close all; clc;
set(0,'DefaultLegendAutoUpdate','off')           % Prevents random "Data" entries to plots
set(0,'DefaultFigureWindowState','docked')       % Keeps figures together
%set(0,'DefaultFigureWindowState','normal')

global alpha rad D1 P tau L1 boundaryreaction Fmeasurements S % General globals used in all scripts
global L2_fit tofit                                           % Globals specific for this script
global itK Ktest AC                                           % Globals required for the fit of k
global itAC ACtest AnsmindvK mk                               % Globals required for the fit of AC

%% script control
basecaseonly = false ;           % Do not fit any datapoints: just makes plots for the set variables
comparebasecase = false ;       % Compare set parameters manually to chosen measurement (see below)
plots = false ;                 % Plot individual membrane graph for each flowrate
feedback = false ;              % Show bvp solver statistics of MembraneModelFunction.m
boundaryreaction = false ;      % Include boundary reaction flux in flux boundary condition
Kfit = false ;                  % Determine the fit for K using LSQ
ACfit = false ;                 % Determine the fit for AC using LSQ (after fitting K)

Measurements = {'d19_5' 'd11_6' 'd4_4' 'd3_44' 'd2_6' 'd1_5' 'd1_14' 'd0_56'};
Measurement = 'd19_5' ;        % Chosen measurement - can be changed as desired

% The number of datapoints in the model can be adjusted if desired
Fcomparison = [1:.25:10];      % Values of the flow rate to plot in comparing graphs
%Fcomparison = [1 2 4 6 8 10]; % Default to match measurement points
```

The measurements are then indicated using the membrane thickness that was used for that experiment. Also, the number of datapoints for which the model should be plotted in the different graphs can be adjusted using `Fcomparison`. After this, the variables and constants are inserted in 2 sections. The constants can be varied as well if it is desired to see what the effects of these values are, but these are considered constant for the current system.

For the importing of the experimental data that was provided, all values for the membrane thickness should be defined beforehand. In the datasheet, the measurements for which only a part of the full flowrate range was measured, should be removed, and all datapoints should be put next to the flowrate. This can also be seen in the `data.xlsx` document that was sent along with the code. If done correctly, all experimental data will be put into one cell structure, which is used throughout the rest of the script. The commented `Exp_Data{n}` in the loop can be uncommented, with which the import of the data can be followed. If everything goes correctly, the output should match the datasets in the excel sheet.

```
% Importing Experimental data
Data = readmatrix('data.xlsx','range','A1:K91');
% All_L2 = [19.5 11.6 4.4 3.44 2.6 1.5 1.14 0.56 0.194]*1e-6;
All_L2 = [19.5 11.6 4.4 3.44 2.6 1.5 1.14 0.56]*1e-6; % 0.194]*1e-6;
nmeasurements = length(All_L2);

for n = 1:nmeasurements
    idx1 = 10*n-10;
    nrepeats(n) = find(isnan(Data(5+idx1,:)),1)-1; % Number of (experimental repeats+1) for 1 membrane
    Exp_Data{n}=Data([5:10]+idx1,[2:nrepeats(n)]);
    %Exp_Data{n}
end
```

The next step is the comparison with the base case, which has been left in each script to check for any (in)equality between the results using the different scripts. This is also the first time the `MembraneModelFunction` is called, using the parameters as described in the section regarding the *Membrane model function*. Here, a for loop is used to compute the concentration profiles for the different flowrates. By changing `Fcomparison` at the beginning of the script it can be defined what the desired range and number of datapoints is for which the calculations should be made. Note that while more datapoints would yield a smoother curve, taking too many datapoints greatly increases computation time. Similarly, by changing the value of `Measurement` to one of the possible values in the above-mentioned array, an experiment can be selected for which the base case must be compared. The remainder of the base case section is used to plot the resulting data, in- or excluding a comparison with the selected experimental data, depending on the chosen settings. In the final lines of the base case section, a check is also put in place to terminate further execution of the script if the `basecaseonly` boolean has been set to true.

```
% Base case: Calculation of concentration as function of flow rate for given K AC alpha values
for n = 1:length(Fcomparison)
    Cpoints(n,:) =
        MembraneModelFunction(Fcomparison(n),L1,L2,rad,P,k,alpha,D1,AC,S,plots,feedback,boundaryreaction);
end
```

3.2. MMV1: Choosing an experiment to fit a value of K and AC for

As the differences between the three scripts are not very large, the entire MMV script will be explained for the most general case of the three, after which solely the differences will be described.

The first script, MMV1__fit_1_for_chosen_exp.m, describes the fitting of one value for k and one for AC for 1 set of experiments corresponding to a specific membrane. As a basis for this, the different datasets corresponding to a total of 8 measurement sets, all using a unique membrane were used. Using this script, one of those 8 sets can be chosen, and an optimum value for k and AC are determined, specifically for the chosen dataset.

Fitting

In order to determine the optimal fit for the datapoints, a least squares regression analysis was applied to the datapoints. For this, the sum of squares about the regression was determined according to formula XXX below:

$$SS_{res} = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Where $Y_i - \hat{Y}_i$ is the distance of the data (Y_i) to the fitted value (\hat{Y}_i).

This sum of squares is defined in both the function balanceK(k2), which is shown below, and balanceAC(AC2), these are similar in function but the first varies the surface reaction rate k0 until a minimum has been found while the other balance varies the AC after an optimal k0 has been found.

```
function f = balanceK(k2)
global alpha rad D1 P L1 boundaryreaction Fmeasurements
global L2_fit tofit
global itK Ktest AC

itK = [itK+ 1];
Ktest = [Ktest, k2];
Ktestpoints = zeros(length(Fmeasurements),4);
disp(['Iteration number ' num2str(itK) ', Value for K: ' num2str(k2)])
for n = 1:length(Fmeasurements)
    Ktestpoints(n,:) =
MembraneModelFunction(Fmeasurements(n),L1,L2_fit,rad,P,k2,alpha,D1,AC,S,false,false,boundaryreaction);
end
Ymodel = Ktestpoints(:,4);
SumOfSquares = sum((Ymodel-tofit).^2);
f = abs(SumOfSquares);
end
```

The calculation is similar to the base case, as the MembraneModelFunction is called in the same way. The only difference here, is that the obtained datapoints for the outgoing membrane concentration are compared to the measurement data using SumOfSquares. Also, the number of iterations is being tracked and displayed along with the current fit value, so the progress of the solver can be seen.

In the balances, the plots and feedback Boolean are set to false because it is not desirable to output 6 graphs and the feedback for every iteration of the fit for every flowrate that is considered.

The balances are minimized according to the boolean Kfit and ACfit, which can be manually set in the script control. Consider that AC cannot be fitted on its own since it relies on the result of Kfit. Since both balances are minimized in a similar way only the k0 fit is shown in this report:

```
%% Optimization K fit
if Kfit == true

itK = 0; Ktest = [];
% options = optimset('Display','iter','TolFun',1e-6,'TolX',1e-6,'MaxFunEvals',1000);
% Can be used as alternative to display iteration results
options = optimset('Display','none','TolFun',1e-9,'TolX',1e-9,'MaxFunEvals',150);

L2_fit = All_L2(idx2)
tofit = Exp_Data{idx2}

AnsminddevK = fminsearchbnd(@balanceK,k,0,+inf,options); % searches for the K giving the lowest deviation
from the datapoints
disp(['LSQ fit value for k: ' num2str(AnsminddevK)])

for n = 1:length(Fcomparison) % Compare the found optimal value of k with experimental data
    Cpoints(n,:) =
    MembraneModelFunction(Fcomparison(n),L1,L2_fit,rad,P,AnsminddevK,alpha,D1,AC,S,plots,feedback,boundaryreaction);
end

...
```

In the assignment of the AnsminddevK variable `fminsearchbnd` is called with some variables. First, a function handle is added to the balance function, which indicates that `fminsearchbnd` will try and minimize the final value of this function. As the output value of this function is the total sum of squares, this function will aim to find the minimum sum of squares. The second variable indicates that the first value that will be calculated is equal to `k`, which is the value manually entered in the beginning of the script. It can be seen as the initial guess for the value of, in this case, `k`. The next 2 variables set the bounds in which a minimum must be found. These bounds go from 0 to positive infinity for the determination of the value for `k`, and from 25% below to 25% above the guessed value of `AC`. Lastly, the solver options, considering the tolerance for finding the minimum of the sum of squares and the maximum number of iterations allowed, are entered. For the `AC` fit the variables are as follows: `@balanceAC, AC, AC*(1-maxdev_AC), AC*(1+maxdev_AC), options`, where `maxdev_AC` indicates the maximum fraction that `AC` is allowed to deviate. With an initial set deviation of 25%, the value for `maxdev_AC` becomes 0.25. The solver options only differ in tolerance, which is set to `1e-2` for the `AC` fit.

The `fminsearchbnd` function was obtained from the MATLAB file exchange [2]. This function allows you to enter lower and upper bounds for finding a minimum of a function, which is not possible to directly do with MATLAB's standard `fminsearch` function. By clever manipulation of the entered function, it uses `fminsearch` to search for a value within the entered bounds.

Comparing

After an optimized K has been found, it is inserted back into the model once more, and relevant concentrations are determined for every flowrate used in the experiments. This is carried out in a similar way to the base case, but here, the optimal value of k (AnsmindvK) is passed on to the function instead of the guessed value.

```
...  
  
for n = 1:length(Fcomparison) % Compare the found optimal value of k with experimental data  
    Cpoints(n,:) =  
    MembraneModelFunction(Fcomparison(n),L1,L2_fit,rad,P,AnsmindvK,alpha,D1,AC,S,plots,feedback,boundaryreaction);  
End  
  
...
```

Finally, the outlet concentrations per flowrate for both the experiment and the model are plotted in the same plot for comparison. For clarity, some of the variables are added to the plot in the form of text, listing the values of k_0 (fit value), α (chosen value), L_2 (value depending on chosen measurement), and AC (initial value before ACfit, fit value in the second part of the script). Then, depending on the ACfit boolean, this whole process is repeated for the absorption coefficient by using the determined optimal k and varying the AC within the bounds to find another minimal sum of squares.

```
...  
  
% plotting comparison between data and experimental results  
figure(); hold on; grid on  
plot(Fcomparison,Cpoints(:,4),'.-b',Markersize=35,LineWidth=3);  
  
for m = 1:nrepeats(idx2)-1  
    plot(Fmeasurements,Exp_Data{idx2}{:,m),'--k',Marker=mstring(m),Markersize=16,LineWidth=3);  
end  
  
legend([' Model',legendset],'fontsize', 20, 'Location', 'southeast');  
h = gca; set(h,'fontsize',24,'FontName','GillSansMT');  
ylim([0 1]); ylabel({'Dimensionless concentration [C = c/c_i_n]'});  
xlim([1 10]); xlabel('Flow rate [mL/h]');  
title('Comparing modeled Cout with experiments');  
text(1.1,.9925,['k0 = ' num2str(AnsmindvK)];['alpha = ' num2str(alpha)];['L2 = '  
num2str(L2)];['AC = ' num2str(AC)]},'fontsize', 20,'HorizontalAlignment', 'left','VerticalAlignment',  
'top');  
  
end
```

Sensitivity analysis

As part of MMV1, a sensitivity analysis for the reaction rate constant (k_0) and the absorption coefficient (AC) were added. With this, the effects of changing k and AC can be plotted, when set to true at the beginning of the script. The range for which k and AC were varied was chosen arbitrarily, so as to cover the largest possible range of outgoing concentrations. Next to this, the porosity was set to 0.45, which is the porosity used in the experimental membranes. The membrane thickness was chosen to be equal to 2.5 μm but can be adjusted freely if desired.

```
% Sensitivity analysis of k and AC
if Sensitivity == true
    Fcomparison = [1:.25:10]; % Values of the flow rate to plot in comparing graphs
    kspace = [1e-3 1e-2 0.05 0.1 0.5 1 10]; % Surface reaction rate (m/s)
    ACspace = logspace(4,8,5); % Absorption coefficient (1/m)

    L2 = 2.5E-6;
    % Sensitivity of k
    legendset_Sk = [];
    for m = 1:length(kspace)
        for n = 1:length(Fcomparison)
            Cpoints_Sk{m}(n,:) =
                MembraneModelFunction(Fcomparison(n),L1,L2,rad,P,kspace(m),alpha,D1,AC,S,false,false,boundaryreaction);
        end
        legendset_Sk = [legendset_Sk, {' k = ' num2str(kspace(m))}]];
    end
end
```

For the sensitivity analysis of k , AC was set to its default value, and vice versa. The resulting data was plotted and can be found in the figures below. In *Figure 1*, the sensitivity analysis for k_0 can be found. Here, it can be seen that for a value of 10, the outgoing concentration remains (almost fully) equal to 0, independent of the flow rate. At a value of $1\text{e-}3$, the outgoing concentration tends to reach 1 at higher flowrates. These differences are to be expected, since the outgoing concentration should remain close to the ingoing concentration for very low reaction rates, while the outgoing concentration should tend to 0 for very high reaction rates.

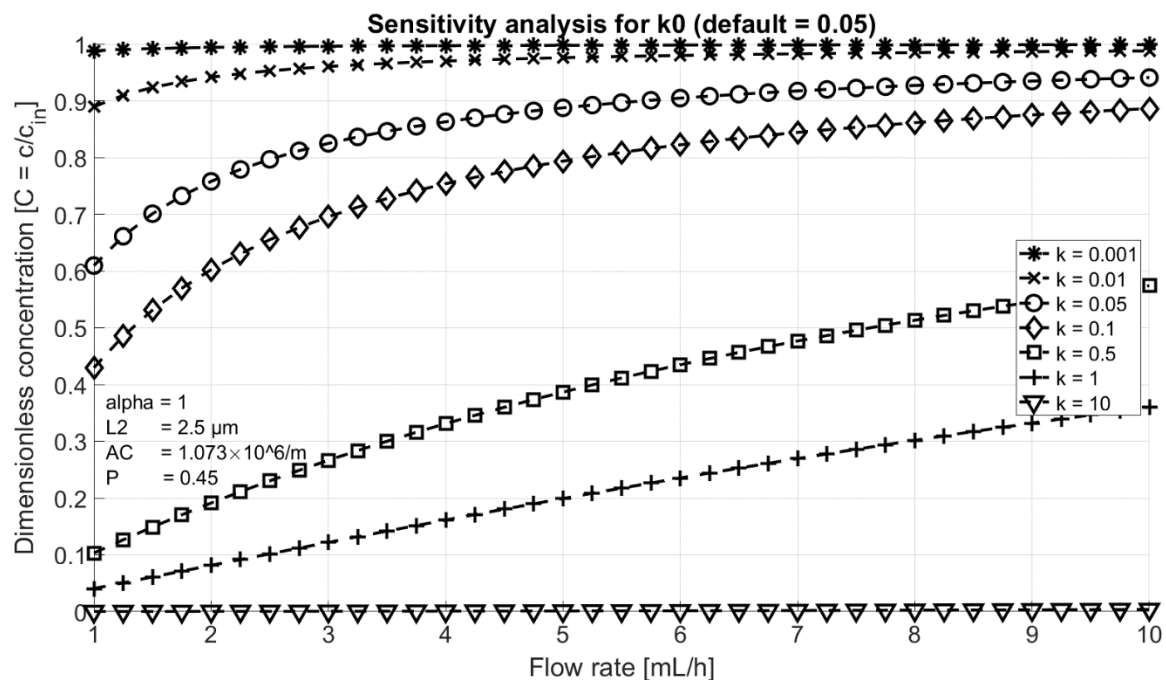


Figure 1: Sensitivity analysis for the reaction rate constant (porosity = 0.45)

In *Figure 2*, the sensitivity analysis for AC is shown. The absorption coefficient describes the decrease of light intensity over the membrane thickness due to absorption of light by the catalyst. At lower values of the absorption coefficient, the characteristic absorption length is longer, due to their inverse relation. This means the light is not absorbed quickly in the membrane, and the reaction rate does not degrade quickly, allowing for a lower outgoing concentration to be reached. At very high values of the absorption coefficient, all light is absorbed very quickly, and the reaction rate rapidly decreases over the length of the membrane, yielding higher outgoing concentrations. This then leads to an outgoing concentration of $0.98 \cdot C_{in}$, increasing further at higher flowrates, while this would normally yield an outgoing concentration between 0.4 and $0.9 \cdot C_{in}$, at the same reaction rate.

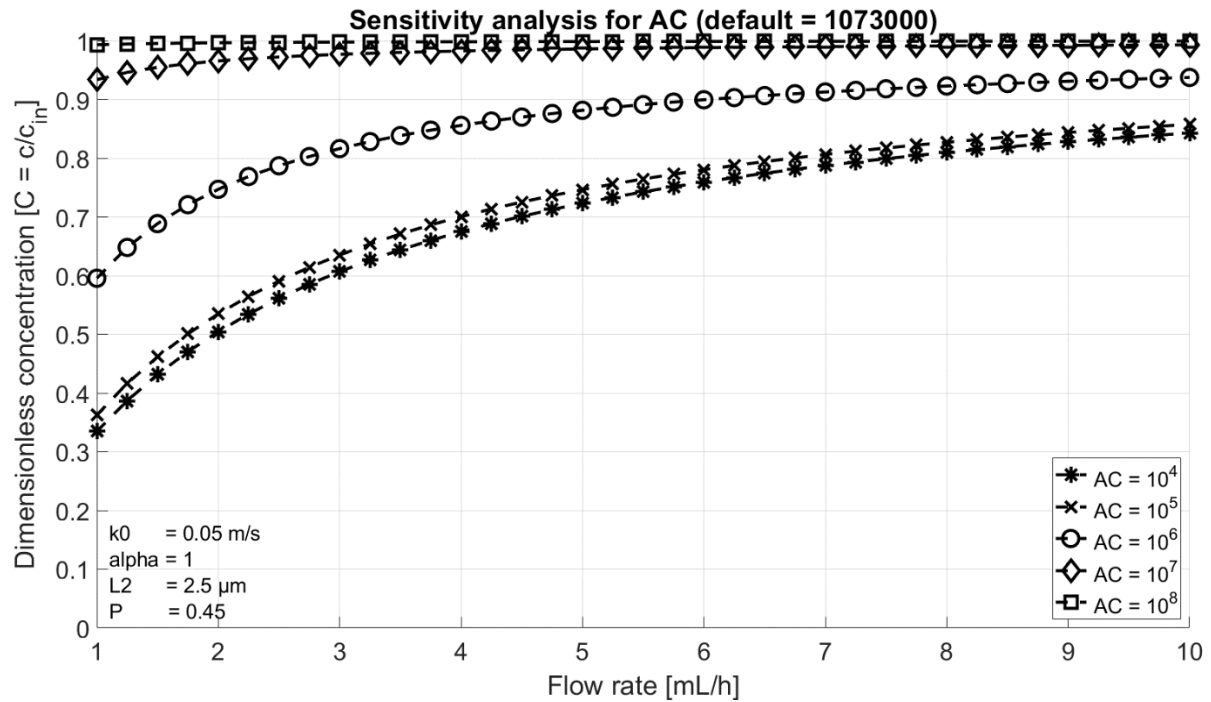


Figure 2: Sensitivity analysis for the absorption coefficient (porosity = 0.45)

3.3. MMV2: Fitting a value for k and AC for each experiment

The second script, MMV2_fit_x_for_all_exp.m, is quite similar to MMV1, with the addition that this script will determine an optimal value for k and AC for each of the measurements found in the data file.

The main difference found in the code, is the presence of an additional for statement around a major part of both Optimization sections, as shown below. This will iterate the fitting function for all measurements, generating an array with the optimal values, taking the respective length in combination with the measurement data for each membrane. Again, like with the previous file, once the final value for k has been found, it will be used to calculate the modelled datapoints once more, after which it can be compared with the measurement data in a plotted figure (for each respective measurement). For the fit of AC, the similar piece of code is used like before, where the fitted values for k are now passed along to the function, and an optimum for AC is determined.

```
if Kfit == true && justplotfinal == false

itK = 0; Ktest = [];
% options = optimset('Display','iter','TolFun',1e-6,'TolX',1e-6,'MaxFunEvals',1000);
% Can be used as alternative to display iteration results
options = optimset('Display','none','TolFun',1e-9,'TolX',1e-9,'MaxFunEvals',150);

for mk = 1:length(All_L2)
    L2_fit = All_L2(mk);
    tofit = Exp_Data{mk}
    AnsmindK(mk) = fminsearchbnd(@balanceK,k,0,+inf,options); % searches for the K giving the lowest
                                                             deviation from the datapoints
    disp(['LSQ fit value for k: ' num2str(AnsmindK(mk))])

    % Compare the found optimal value of k with experimental data & plotting comparison
    % Similar to what has been shown before, and therefore omitted to keep the shown code to a minimum
end
end
```

If the script has been run at least once, the resulting fitted values can be inserted into the script in the form of an array, after which justplotfinal can be used to skip the fitting process and plotting of the results from k and k+AC separately. It will then only plot the final figures with all fitted parameters. This is greatly recommended to use if the script has already been run before and no data has changed, as it can take quite a while before all fitting has completed.

```
justplotfinal    = true ;           % Can be used if Kfit and ACfit have previously been determined and
                                     fittings are turned off
```

```
%% Additional processing unique to this script

if justplotfinal == true
    AnsmindK = [ ] % Paste previously determined array of values here for easy future plotting
    AnsmindAC= [ ]

for mk = 1:length(All_L2)
    L2_fit = All_L2(mk);
    tofit = Exp_Data{mk}

    % Compare the found optimal value of k with experimental data & plotting comparison
    % Similar to what has been shown before, and therefore omitted to keep the shown code to a minimum
end
end
```


Finally, a figure is plotted, comparing the fitted values for k for each experiment, as they do not seem to be constant. Using this figure, it can be checked if a relation between k and the membrane thickness can be found.

```
%% Relation between k and F
figure(); hold on; grid on
plot(All_L2([3:end]),AnsindevK([3:end]), '-*k',Markersize=16,LineWidth=3)
legend(['Data'], 'fontsize', 20, 'Location', 'southeast');
h = gca; set(h, 'fontsize', 24, 'FontName', 'GillSansMT');
ylabel({'Fitted reaction constant (k0)'});
xlabel('Membrane thickness (L\2)');
title('Relation between membrane thickness and reaction constant');
```

3.4. MMV3: Fitting 1 value for k and AC to all experiments

The third model variation script, MMV3_fit_1_for_all_exp.m, is identical to MMV2, but now the for loop considering all measurements has been placed within the `fminsearch` balance, instead of around the balance in the optimization parts. With this, the total sum of squares is determined for each iteration by taking the total sum of all individual sum of squares values for each experimental dataset. This allows for the determination of the least squares fit of a single value of k and AC holding for the total dataset of all experiments.

Like with script 2, the section allowing to generate just the final plots is also added here, so if the script has been run once, the (now singular values) for k and AC can be filled in, after which `justplotfinal` can be set to true.

```
function f = balanceK(k2)
global alpha rad D1 P L1 boundaryreaction Fmeasurements % General globals used in all scripts
global All_L2 Exp_Data % Globals specific for this script
global itK Ktest AC % Globals required for the fit of k

Fmeasurements = [1 2 4 6 8 10];
itK = [itK+ 1];
Ktest = [Ktest, k2];
Ktestpoints = zeros(length(Fmeasurements),4);
disp(['Iteration number ' num2str(itK) ', Value for K: ' num2str(k2)])

for m = 1:length(All_L2)
    L2_fit = All_L2(m);
    for n = 1:length(Fmeasurements)
        Ktestpoints(n,:) =
MembraneModelFunction(Fmeasurements(n),L1,L2_fit,rad,P,k2,alpha,D1,AC,S,false,false,boundaryreaction);
    end
    Ymodel = Ktestpoints(:,4);
    SumOfSquares(m) = sum((Ymodel-Exp_Data{m}).^2);
end
fullsum = sum(SumOfSquares)
f = abs(fullsum);
end
```

4. Overview of results

For all following model results the following parameters were used:

```
%% Variables (set values for base case)

Fmeasurements = [1 2 4 6 8 10];           % g/h (assumed to be equal to mL/h)
alpha = 1;                                % Concentration ratio at the boundary
AC = 1.073E6;                              % Absorption coefficient (1/m)
maxdev_AC = 0.25;                          % Maximum deviation from literary AC when fitting

%% constants
rad = 0.002;                               % Radius membrane (m)
D1 = 5.7e-10;                              % Diffusion coefficient pre-membrane (m2/s)
P = 0.45;                                  % Porosity
L1 = 200e-6;                               % Thickness of the mass transfer boundary layer (m)
S = 700;                                   % Specific surface area of the catalyst (m2/m3)
```

First, a single membrane is considered by using the MMV1 script. Here the plots are set to true so the concentration profile per flowrate can be seen (Figure 3). Kfit and ACfit are set to true to make the model fit the experimental values, which is shown in (Figure 4).

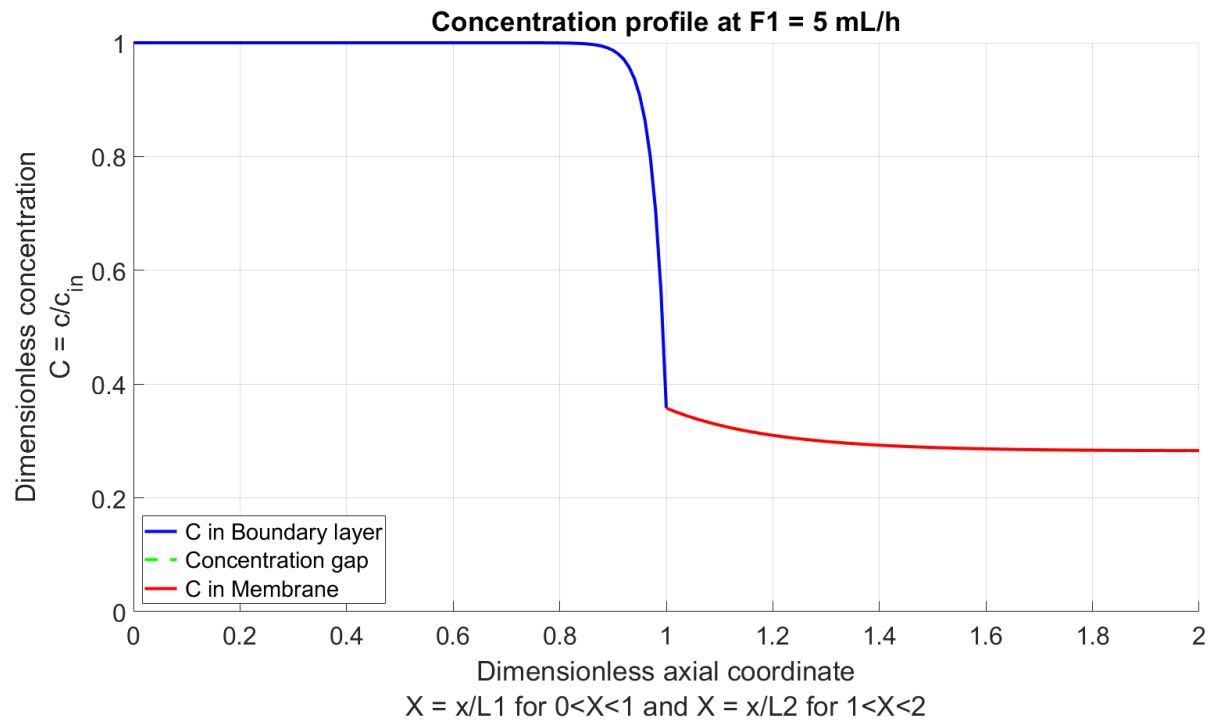


Figure 3: Concentration profile of a modelled membrane
 with $k_0 = 0.19911$ m/s, $AC = 1.073e6$ m⁻¹, $L2 = 4.4$ μ m, $P = 0.45$ and $F1 = 5$ mL/h.

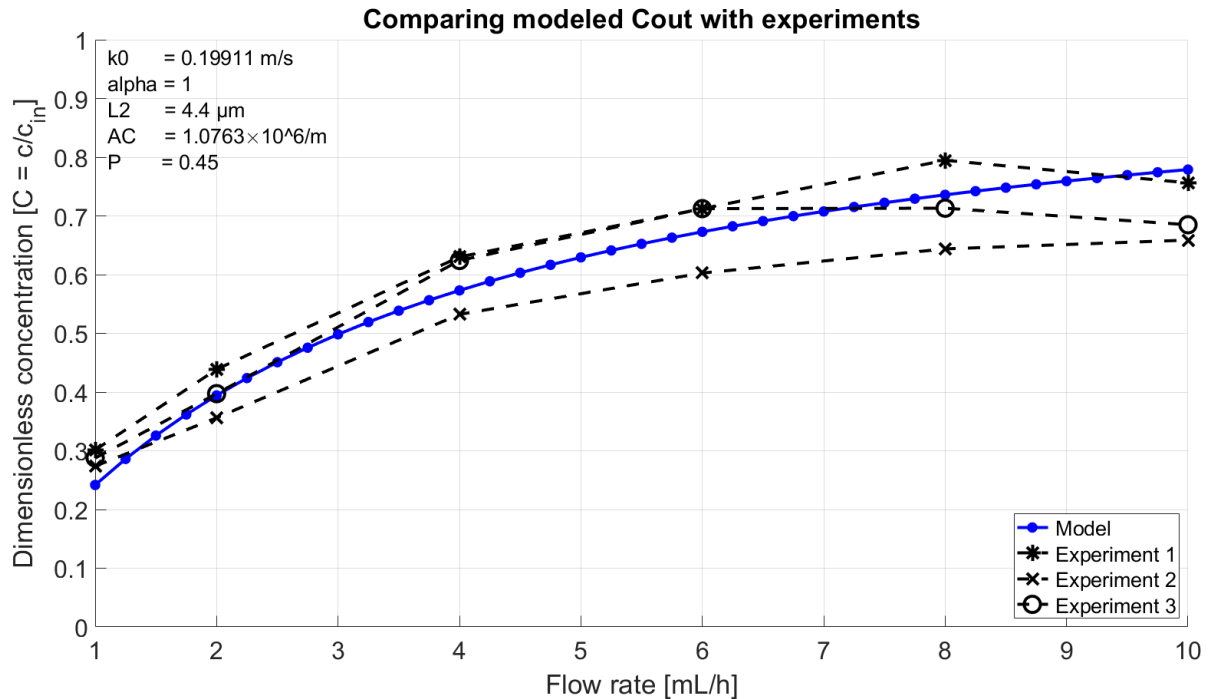


Figure 4: Model results plotted over experimental data from a membrane with a thickness of $4.4 \text{ } \mu\text{m}$.
(without surface reaction boundary flux)

Next, all models are fitted twice to an optimal k_0 and AC using MMV2 at a porosity of 0.45. The first fit is done without taking the boundary reaction flux into account and can be seen in Figure 5. In the second fit this boundary reaction used in the solution and can be seen in Figure 6. In both these figures the values of k_0 are plotted at each membrane thickness they were fitted at.

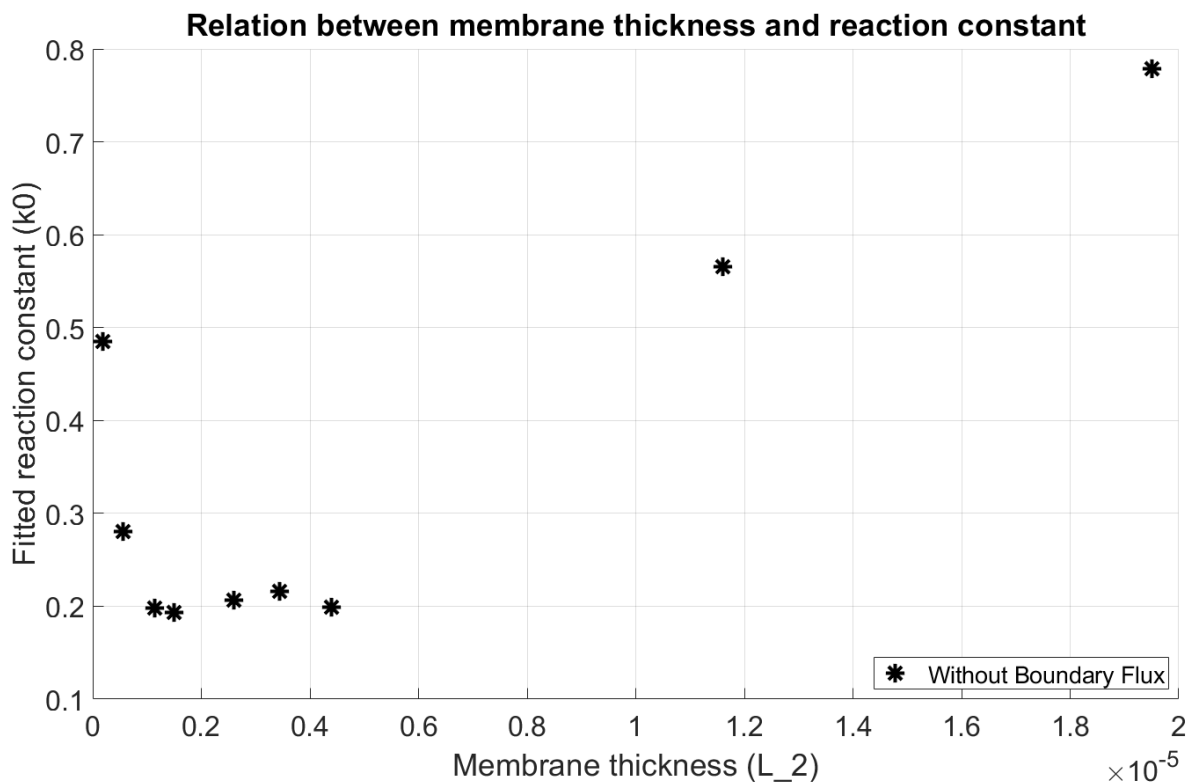


Figure 5: fitted k_0 per membrane thickness plotted against membrane thickness (without surface reaction boundary flux).

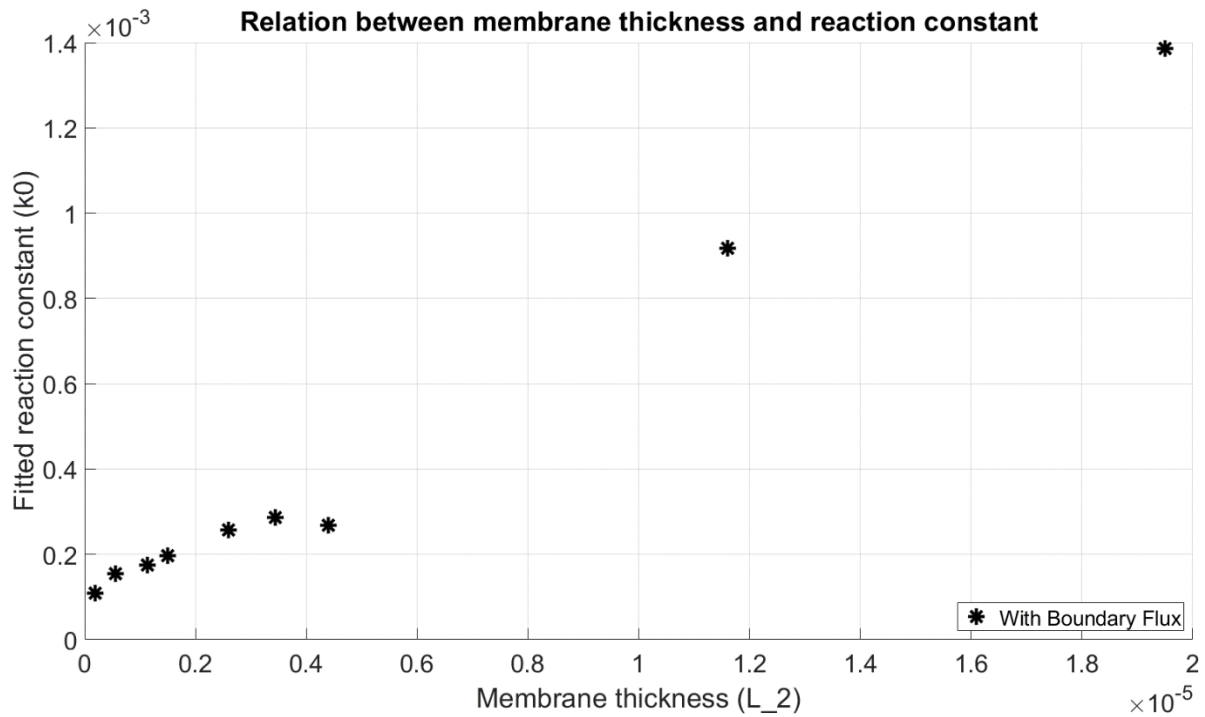


Figure 6: fitted k_0 per membrane thickness plotted against membrane thickness (with surface reaction boundary flux).

In *Figure 5* and *Figure 6*, both plots have a similarities, where the first 6 thicknesses ranging from $\sim 0.2 \mu\text{m}$ to $\sim 4.5 \mu\text{m}$ are around the same range and both plots also have a jump in k_0 for the last 2 thicknesses at $11.6 \mu\text{m}$ and $19.5 \mu\text{m}$. For the non-surface reaction flux model there is an unexpected jump in k_0 values compared to the 4 following thicknesses, where the k_0 seems to plateau as is expected since thickness should not affect the k_0 much. In the surface reaction flux model, the k_0 values seem to have a slight positive correlation with the membrane thickness. But these kinds of correlations are difficult to confirm with only 2 to 4 measurements per thickness. As stated earlier, for both models there is a large jump in k_0 for the 2 thickest membranes. Possible reasons for this are elaborated further in the discussion. By comparing both figures it is also visible that while considering the surface reaction flux in the boundary condition does not change the relation between thicknesses much, it does impact the value of k_0 quite significantly. If only the first 6 thicknesses are considered, their average k_0 value differ with about a factor of 1000.

Finally, using MMV3, a common k_0 and AC is determined for all experiments. Since the membranes of $11.6 \mu\text{m}$ and $19.5 \mu\text{m}$ differ greatly from the other 6 thicknesses as discussed previously, it was decided to only look at the 6 thinner membranes for this fit. Below in *Figure 7* and *Figure 8*, two different membranes with their respective experimental data are plotted with this common k_0 and AC. It can be seen that they both fit quite nicely. Similarly to in *Figure 7* and *Figure 8*, a common value for k_0 and AC was determined for the experiments of the membranes of $11.6 \mu\text{m}$ and $19.5 \mu\text{m}$ was determined as well. With these values, the comparison between the model and experimental data can be found in *Figure 9* and *Figure 10*.

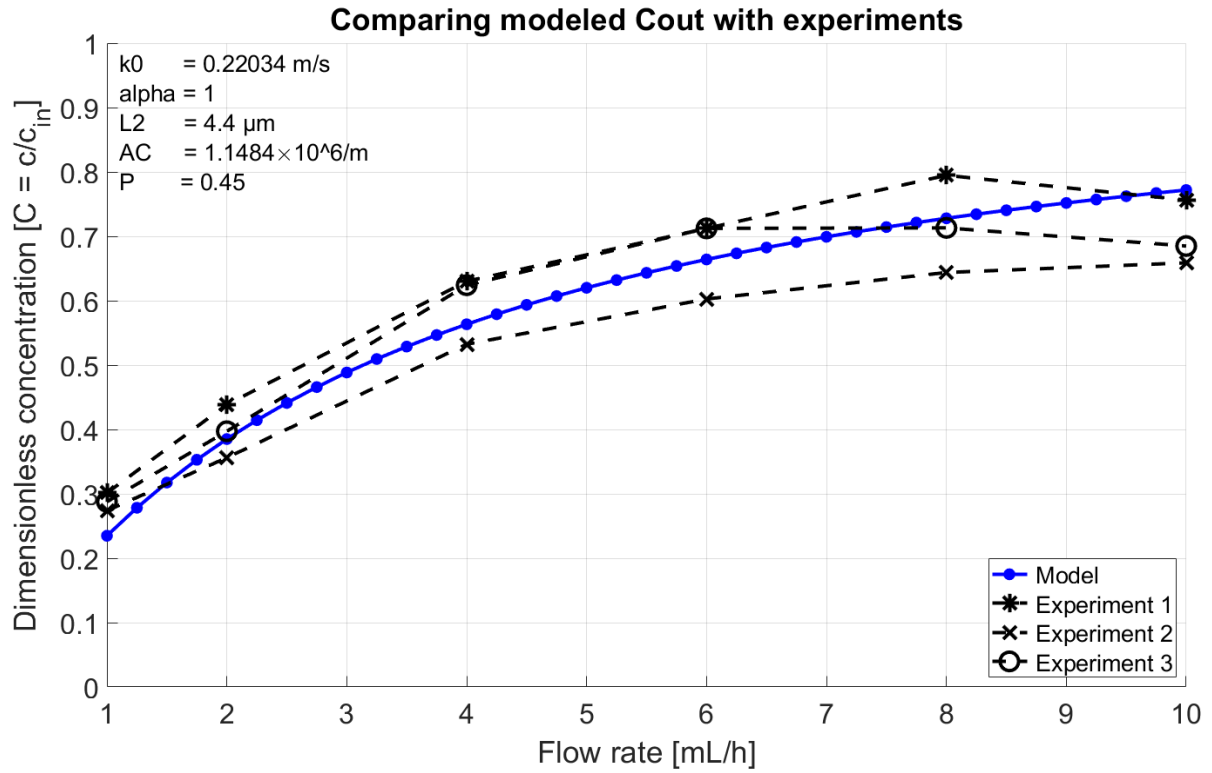


Figure 7: Comparing the experimental data of a membrane with $L_2 = 4.4 \text{ } \mu\text{m}$ with the overall fitted k_0 and AC

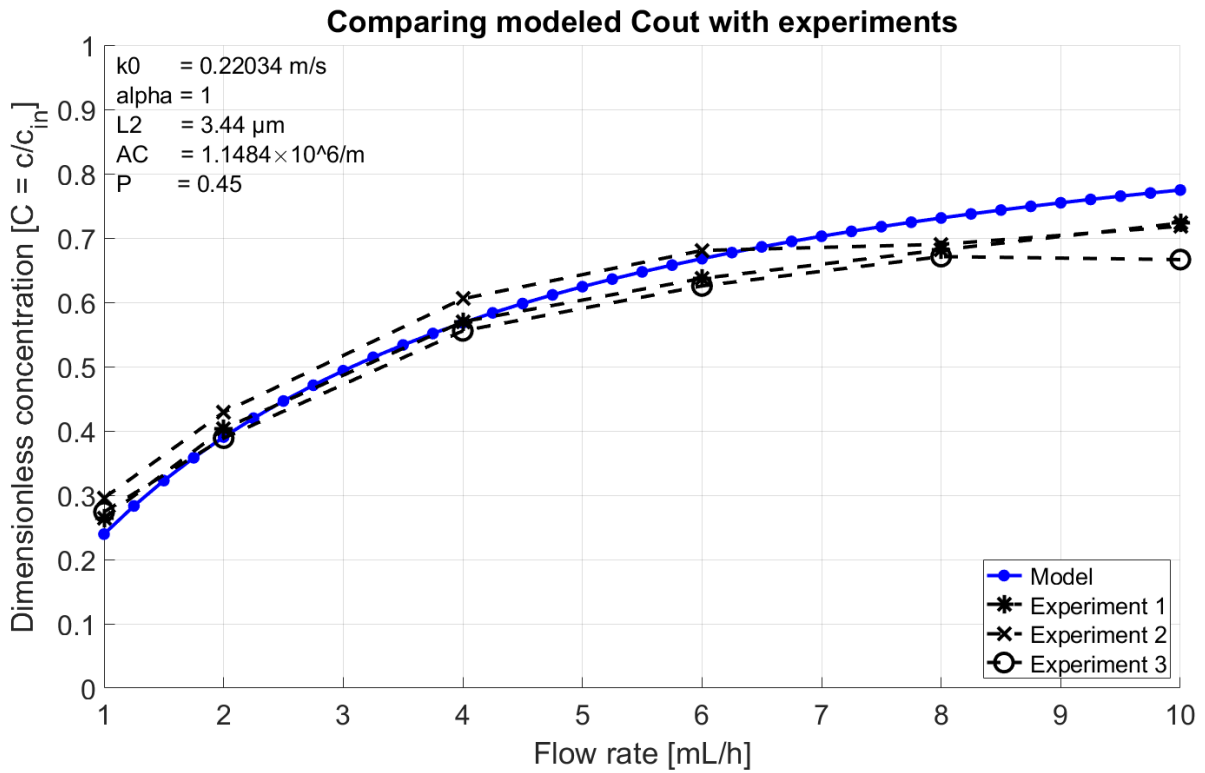


Figure 8: Comparing the experimental data of a membrane with $L_2 = 3.44 \text{ } \mu\text{m}$ with the overall fitted k_0 and AC

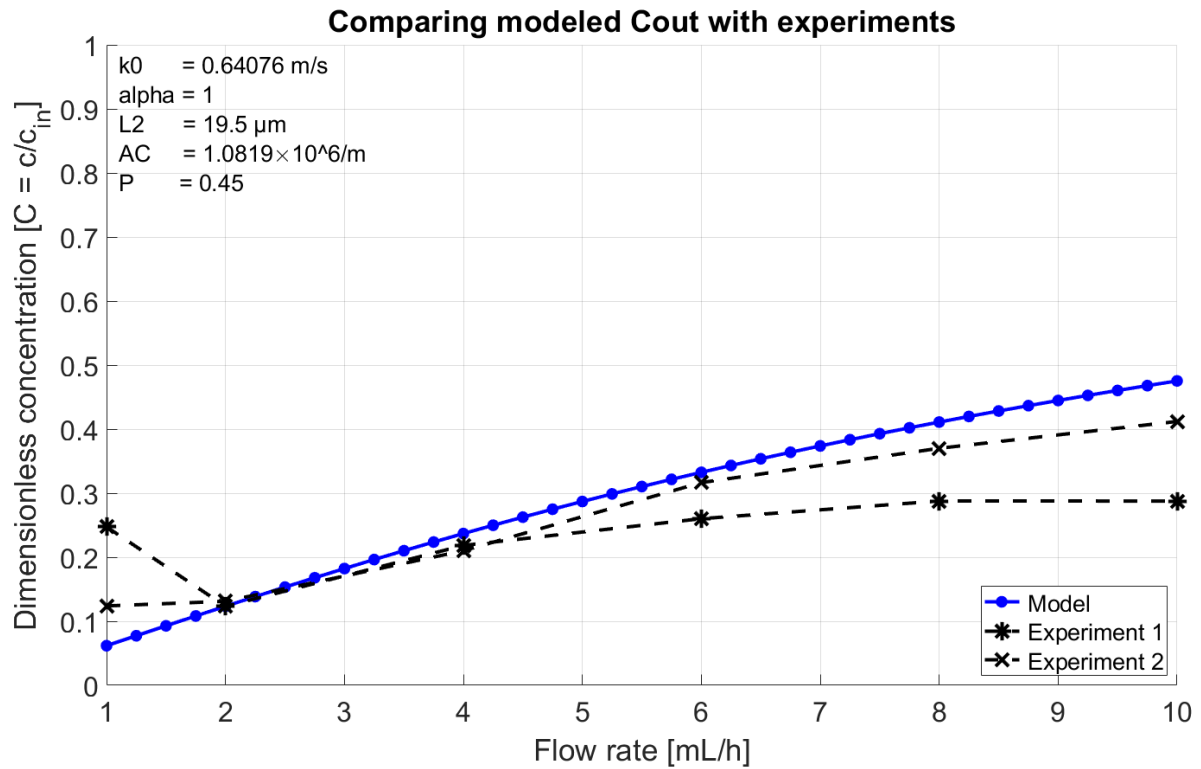


Figure 9: Comparing the experimental data of a membrane with $L_2 = 19.5 \text{ } \mu\text{m}$ with the overall fitted k_0 and AC

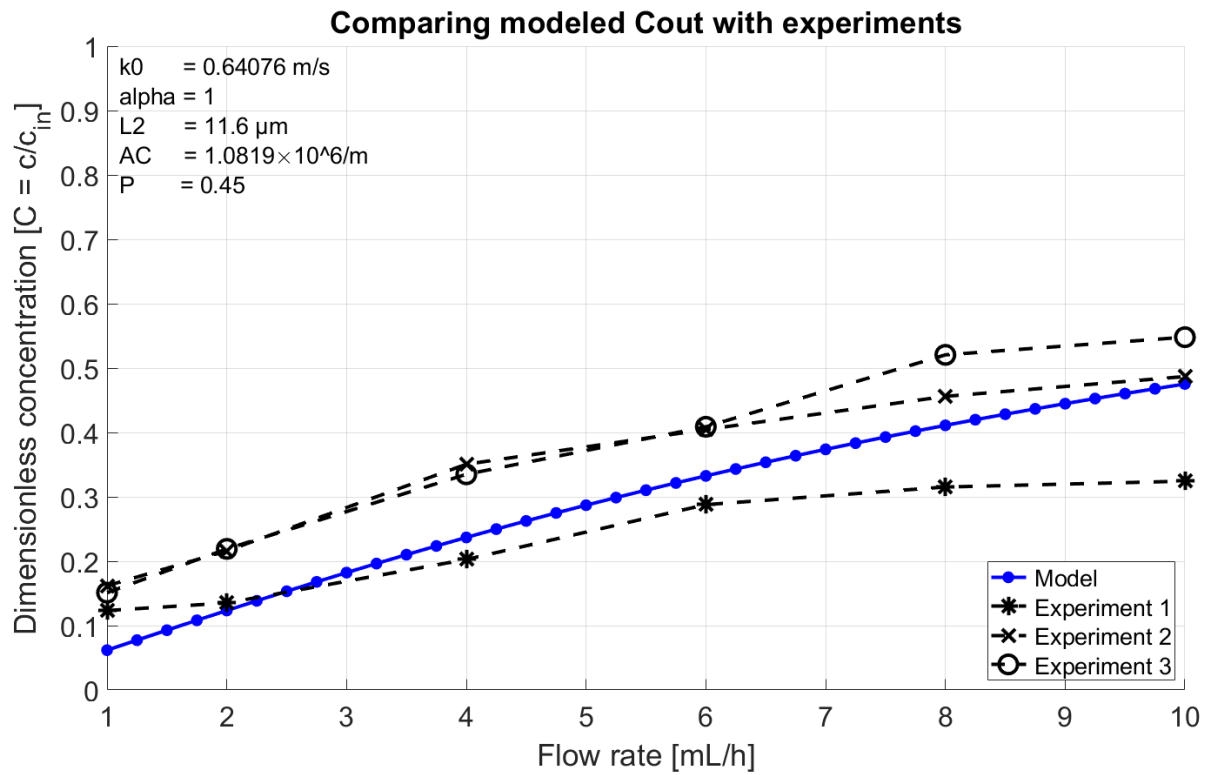


Figure 10: Comparing the experimental data of a membrane with $L_2 = 11.6 \text{ } \mu\text{m}$ with the overall fitted k_0 and AC

All individual computed values for k_0 and AC, for both the system without consideration of the boundary flux and the system with the consideration of the boundary flux, have been summarized in *Table 1*, below. Here, the overall fit values for k_0 and AC, considering all datasets, considering the 6 thin membranes, and considering just the 2 thicker membranes can be found as well.

Table 1: Overview of results

Membrane thickness (μm)	k_0 (m/s)	AC (1/m) *1e6	k_0 (m/s) *1e-3 (boundary flux)	AC (1/m) *1e6 (boundary flux)
19.5	0.7783	1.0831	1.3872	1.3412
11.6	0.5656	1.0808	0.9182	1.3412
4.4	0.1991	1.0763	0.2674	1.3412
3.44	0.2157	1.0766	0.2855	1.3410
2.6	0.2064	1.0757	0.2580	1.3405
1.5	0.1938	1.0731	0.1976	1.3412
1.14	0.1981	1.0730	0.1749	1.2308
0.56	0.2806	1.0730	0.1544	1.0940
0.194	0.4849	1.0730	0.1088	1.0730
All (MMV3)	0.2981	1.0185	0.3056	0.8048
Low L2 (MMV3)	0.2203	1.1482	0.2068	0.8048
High L2 (MMV3)	0.6408	1.0819	1.0758	1.3412

Conclusion & discussion

Looking at the sensitivity analysis, the model behaves as expected. By varying the k_0 the outgoing concentration over all flowrates changes accordingly. Similarly, when raising the AC the outgoing concentration increases due to more light absorption resulting in shallower catalyst activation. Inversely, the outgoing concentration lowers when lowering AC.

The determination of a common k_0 and AC seemed to work quite well for the regular model. A value between the two extreme values of the k_0 and AC datasets were found, and the 6 thinner membranes were considered more by the LSQ since they take up a larger percentage of the datapoints.

The fitting of individual k_0 values resulted in large jump in k_0 for the membranes with a thickness of 19.5 μm compared to the 6 thinner membranes. Additionally, there seems to be a slight positive correlation between membrane thickness and k_0 for the surface reaction boundary-flux model. It is expected that k_0 would remain constant since the surface reactivity of the catalyst should not change depending on the thickness of the membrane it is deposited on. Variation in the coating or structure of the membranes is a likely explanation for the apparent change in the reaction constant. It could also be the case that the surface of thicker membranes is more accessible than for the thinner membranes resulting in a higher apparent value for k_0 .

Possible improvements

For the derivation of the model, it was assumed that the membrane only has cylindrical pores. This is useful for the derivation of the ODE's and the boundary conditions since this imposes a simple velocity relation between pre-membrane and intra-membrane. While this assumption will probably not make a significant difference in results from reality, it is worth considering that this assumption does not describe reality, where the pores deviate in shape and flow path. In future modelling this could be approximated more closely by considering i.e. tortuosity in the derivation.

Additionally, it was assumed that the light on the membrane was evenly distributed over the whole surface and is thus able to penetrate the membrane consistently over the entire surface. This could either be compensated by making the radius of the model membrane smaller or by scripting this uneven distribution into the model.

For the fit, only the sum of squares about the regression was determined. This could possibly be improved by also considering the sum of squares due to regression and calculating the total sum of squares about the mean. With this, and a proper analysis of variance, more concrete conclusions could be drawn on the correctness and goodness of the fit, as it has now subjectively been judged by comparing the appearance of the model with relation to the datapoints.

Bibliography

- [1] P.V. Danckwerts, Continuous flow systems: Distribution of residence times, Chemical Engineering Science, Volume 2, Issue 1, 1953, Pages 1-13, ISSN 0009-2509, [https://doi.org/10.1016/0009-2509\(53\)80001-1](https://doi.org/10.1016/0009-2509(53)80001-1).
- [2] John D'Errico (2022). fminsearchbnd, fminsearchcon (<https://www.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd-fminsearchcon>), MATLAB Central File Exchange. Retrieved January 25, 2022.