**Homework 7: Temporary Solution**

Ben Wang and Mark Styczynski

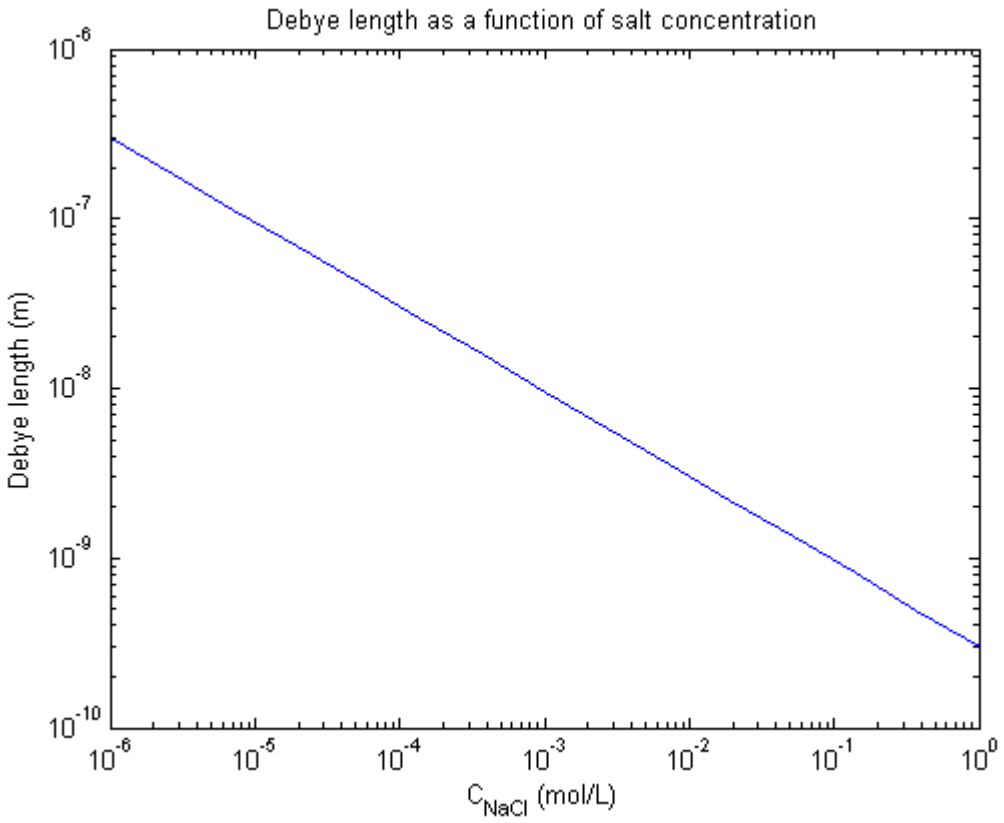
What is below is a temporary solution file for homework 7… it does not have details or derivations, but will serve to show you one approach to the problems and some of the expected output. A complete solution will be posted sometime soon, but probably not before the exam, and thus this temporary solution.

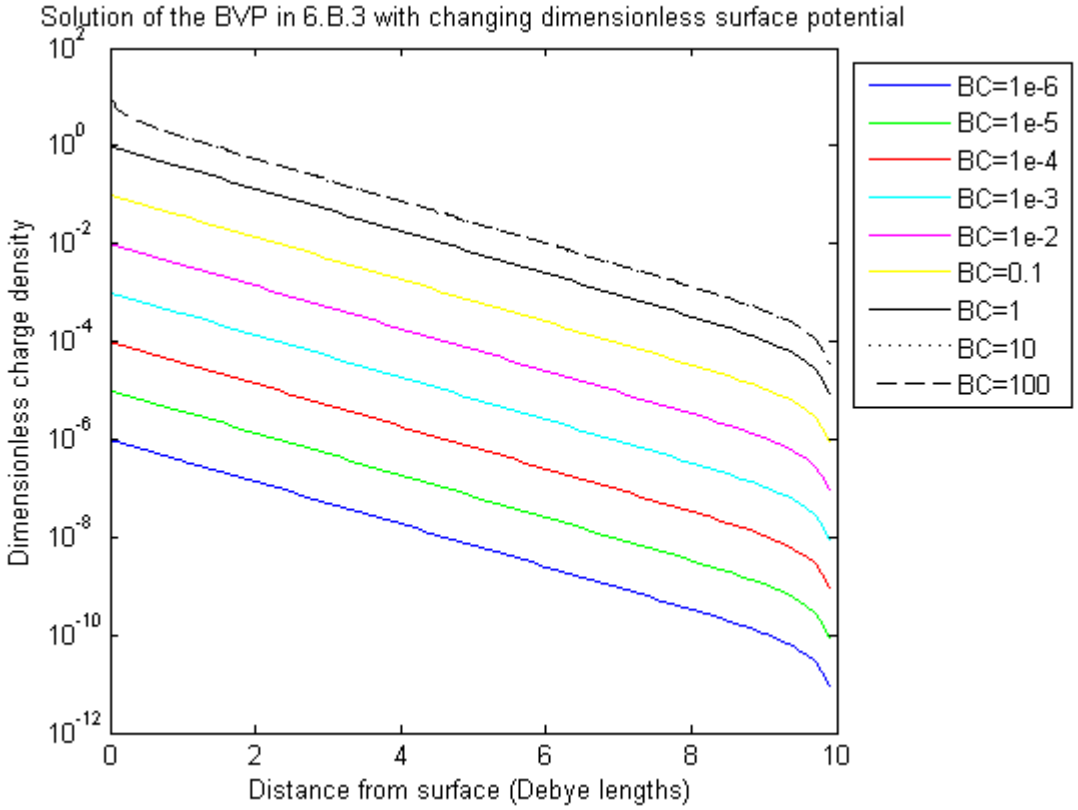
These are not a replacement for the final solution… these are a work in progress, and errors will be corrected later. The final solution, which is to come, will have precedence in any and all cases over this version. This version has not yet been subjected to the, uh, rigorous scrutiny that the final version always receives. Errors in this quickly-made solution do not mean that errors in your results will be ignored.

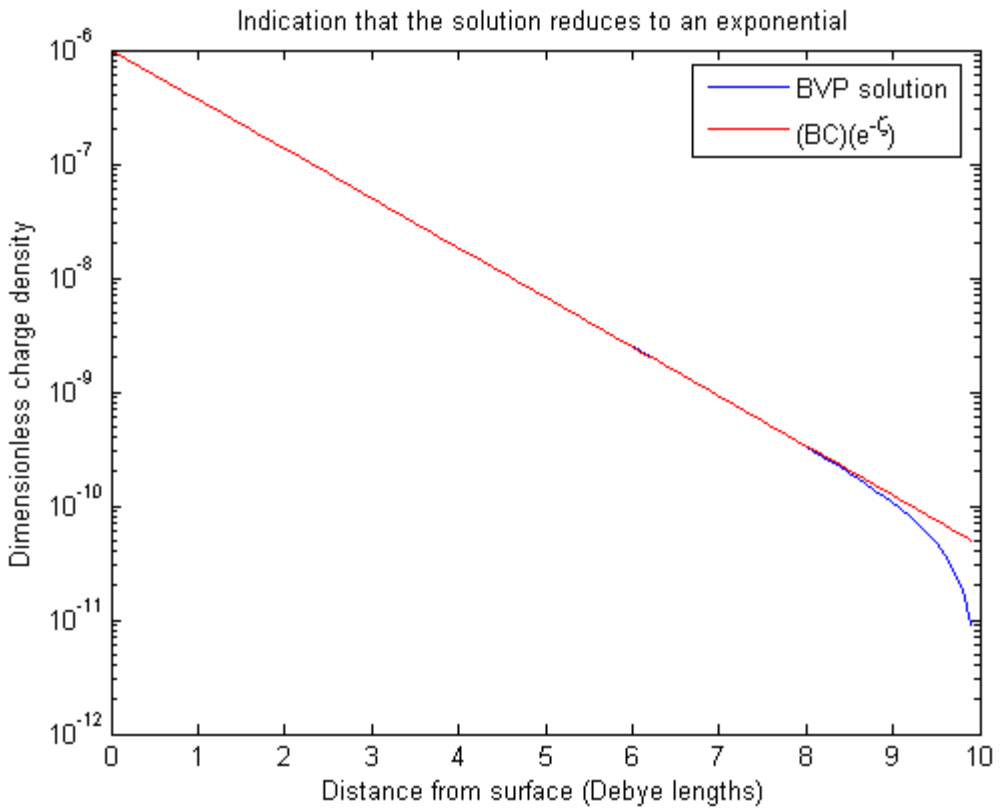
*1. 6.B.3*

Using the dimensionless equation he gives and applying the boundary conditions, we get the graphs below. Note that since most of the change in the function value is occurring near the surface, it is ideal to have non-uniform grid spacing allowed for in your function and to put a lot of points near the surface so that you get an accurate representation of your function with fewer total points (and thus less computational overhead).

The plot of our BVP solution is at our first dimensionless phi, which is 1e-6. We see that all the way to the end, it closely follows an exponential curve. We expect there to be some deviation at the end, since the exponential curve will never reach 0 (until inifinity), and yet we force our function to reach 0 at some finite infinity.


Debye length as a function of salt concentration
Debye length (m)
$10^{-6}$
$10^{-7}$
$10^{-8}$
$10^{-9}$
$10^{-10}$
$10^{-6}$
$10^{-5}$
$10^{-4}$
$10^{-3}$
$10^{-2}$
$10^{-1}$
$10^{0}$
$C_{NaCl}$ (mol/L)


Solution of the BVP in 6.B.3 with changing dimensionless surface potential
Dimensionless charge density
$10^{2}$
$10^{0}$
$10^{-2}$
$10^{-4}$
$10^{-6}$
$10^{-8}$
$10^{-10}$
$10^{-12}$
0
2
4
6
8
10
Distance from surface (Debye lengths)
BC=1e-6
BC=1e-5
BC=1e-4
BC=1e-3
BC=1e-2
BC=0.1
BC=1
BC=10
BC=100




```
clear all; %close all;
figure(1);
cSalt = logspace(-6,0,15)*1e3; % mol/m^3
cPlot = cSalt/1e3;
e0 = 8.854e-12; % Coulombs^2/(Joule * meters)
er = 78;
kb = 1.381e-23; % J/K
T = 25 + 273; % K
qe = 1.602e-19;  %Coulombs
Nav = 6.022e23;   % particles/mol
debye = (e0*er*kb*T./(qe^2*Nav*2*cSalt)).^(.5);
figure(1)
loglog(cPlot,debye);
title('Debye length as a function of salt concentration');
xlabel('C_N_a_C_l (mol/L)');
ylabel('Debye length (m)');
fineRes = 1000;
BC1 = logspace(-6,2,9);
% Set some finite infinity.
infLength = 10;
% Put lots of points near the surface.
points1 = linspace(0,1,fineRes+1);
points1 = points1(2:end);
points2 = linspace(1,infLength,(infLength-1)*10 + 1);
points2 = points2(2:end-1);
zeta = [points1 points2];
guess2 = zeros(size(points2));
options = optimset('Jacobian','on');
% Solve out equation using some reasonable initial guess.
```

```
for i=1:length(BC1),
    guess1 = interp1([0 1],[BC1(i) .25*BC1(i)],points1);
    phiGuess = [guess1 guess2];
    phi(i,:) =
fsolve(@(x)ta_BVP(x,zeta,infLength,BC1(i)),phiGuess,options);
end
figure(2)
plotArray=['b- ';'g- ';'r- ';'c- ';'m- ';'y- ';'k- ';'k: ';'k--'];
for i=1:length(BC1),
    semilogy(zeta,phi(i,:),plotArray(i,:))
    hold on
end
xlabel('Distance from surface (Debye lengths)');
ylabel('Dimensionless charge density');
title('Solution of the BVP in 6.B.3 with changing dimensionless surface
potential');
legend('BC=1e-6','BC=1e-5','BC=1e-4','BC=1e-3',...
    'BC=1e-2','BC=0.1','BC=1','BC=10','BC=100', ...
    'Location','BestOutside')
figure(3)
i = 1;
semilogy(zeta,phi(i,:))
hold on
semilogy(zeta,BC1(i)*exp(-zeta),'r')
title('Indication that the solution reduces to an exponential');
xlabel('Distance from surface (Debye lengths)');
ylabel('Dimensionless charge density');
legend('BVP solution','(BC)(e^-^\zeta)')
```

---

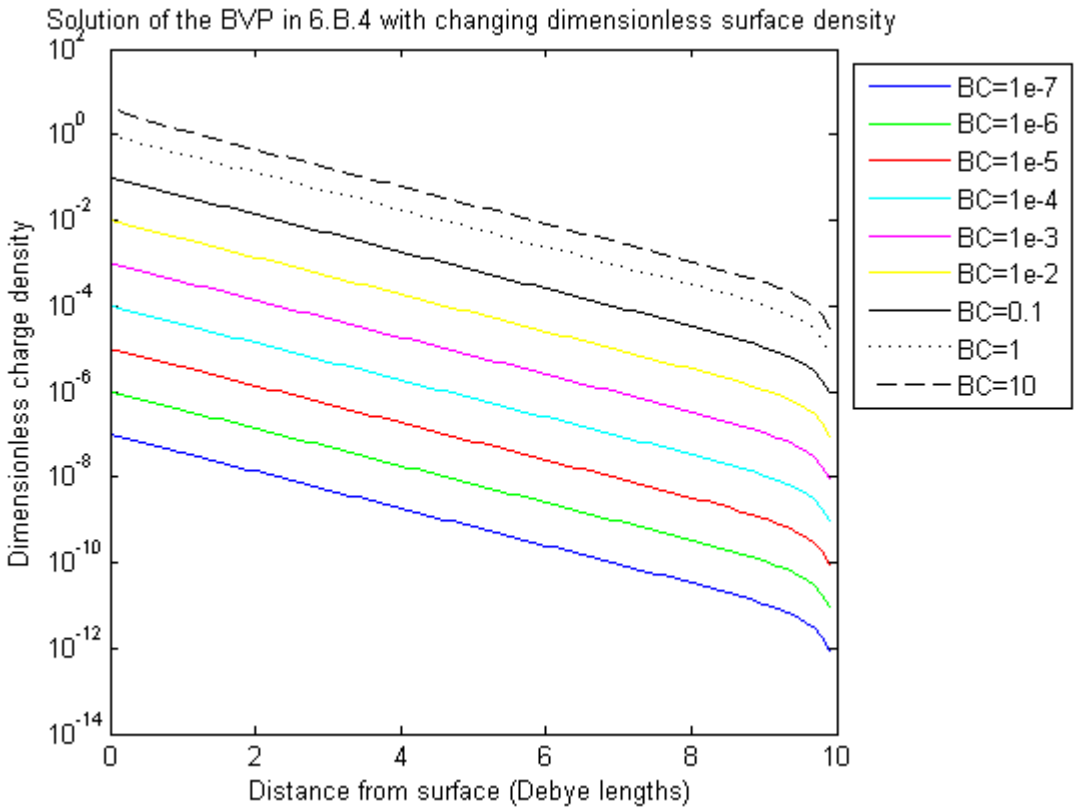
```
function [f, J]= ta_BVP(phi,zeta,infLength,BC1)

% I use "zetaPlusHalf" and "zetaMinusHalf" as placeholders so
%  that I'm not constantly writing tons of equations that will
%  otherwise have no meaning to me.  Beyond that, this is
%  just a finite differences implementation of the BVP, using
%  nonuniform grid spacing.
J=spalloc(length(phi),length(phi),3*length(phi));
zetaPlusHalf = (zeta(2)+zeta(1))/2;
zetaMinusHalf = zeta(1)/2;
zetaHalfDiff = zetaPlusHalf - zetaMinusHalf;
f(1) = -phi(2)/((zeta(2)-zeta(1))*zetaHalfDiff) ...
    + phi(1)/zetaHalfDiff*(1/(zeta(2)-zeta(1)) + 1/(zeta(1))) ...
    - BC1/(zeta(1)*zetaHalfDiff) ...
    + .5*exp(phi(1)) - .5*exp(-phi(1));
J(1,1) = 1/zetaHalfDiff*(1/(zeta(2)-zeta(1)) + 1/(zeta(1))) ...
    + .5*exp(phi(1))+.5*exp(-phi(1));
J(1,2) = -1/((zeta(2) - zeta(1))*zetaHalfDiff);
for i=2:(length(phi)-1),
    zetaPlusHalf = (zeta(i+1)+zeta(i))/2;
    zetaMinusHalf = (zeta(i)+zeta(i-1))/2;
    zetaHalfDiff = zetaPlusHalf - zetaMinusHalf;
    f(i) = -phi(i+1)/((zeta(i+1)-zeta(i))*zetaHalfDiff) ...
        + phi(i)/zetaHalfDiff*(1/(zeta(i+1)-zeta(i)) + 1/(zeta(i)-
zeta(i-1))) ...
        - phi(i-1)/((zeta(i)-zeta(i-1))*zetaHalfDiff) ...
```

```
        + .5*exp(phi(i)) - .5*exp(-phi(i));
    J(i,i) = 1/zetaHalfDiff*(1/(zeta(i+1)-zeta(i)) + 1/(zeta(i)-zeta(i-
1))) ...
        + .5*exp(phi(i)) + .5*exp(-phi(i));
    J(i,i-1) = -1/((zeta(i)-zeta(i-1))*zetaHalfDiff);
    J(i,i+1) = -1/((zeta(i+1)-zeta(i))*zetaHalfDiff);
end
zetaPlusHalf = (infLength+zeta(end))/2;
zetaMinusHalf = (zeta(end)+zeta(end-1))/2;
zetaHalfDiff = zetaPlusHalf - zetaMinusHalf;
f(length(phi)) = phi(end)/zetaHalfDiff*(1/(infLength-zeta(end)) + ...
                                        1/(zeta(end)-zeta(end-1))) ...
              - phi(end-1)/((zeta(end)-zeta(end-1))*zetaHalfDiff) ...
              +.5*exp(phi(end)) - .5*exp(-phi(end));
J(end,end) = 1/zetaHalfDiff*(1/(infLength-zeta(end)) + ...
                                        1/(zeta(end)-zeta(end-1))) ...
              + .5*exp(phi(end)) + .5*exp(-phi(end));
J(end,end-1) = -1/((zeta(end)-zeta(end-1))*zetaHalfDiff);
```

*2. 6.B.4*

Same thing… we'll just implement the different boundary condition. The interesting thing is that when you implement the boundary condition, it turns out to be a very simple Neumann BC, and when you actually code this and solve it, the picture looks identical to above, at least for small values of the BC. Now, does that make sense? In fact, since we said above that the graph looks like a simple exponential, and we know that the derivative of an exponential is equal to its value at that point, we realize that forcing an exponential to have a derivative at point 0 is the same as forcing to have that specific value. So, we are not surprised to see high similarity for dimensionless boundary conditions much less than one.




```
clear all; close all;
kb = 1.381e-23; % J/K
T = 25 + 273; % K
qe = 1.602e-19;  %Coulombs
fineRes = 100;
BC1 = logspace(-7,1,9);
infLength = 10;
points1 = linspace(0,1,fineRes+1);
points1 = points1(2:end);
points2 = linspace(1,infLength,(infLength-1)*10 + 1);
points2 = points2(2:end-1);
zeta = [points1 points2];
guess2 = zeros(size(points2));
LzeroP = -(zeta(1) + zeta(2))/(zeta(1)*zeta(2))
LoneP = zeta(2)/(zeta(1)*(zeta(2)-zeta(1)))
LtwoP = -zeta(1)/(zeta(2)*(zeta(2)-zeta(1)))
for i=1:length(BC1),
    guess1 = interp1([0 1],[BC1(i) .25*BC1(i)],points1);
    phiGuess = [guess1 guess2];
    phi(i,:) = fsolve(@(x)ta_BVP2(x,zeta,infLength,BC1(i)),phiGuess);
    phi0 = -BC1(i)/LzeroP - LoneP/LzeroP*phi(i,1) -
LtwoP/LzeroP*phi(i,2);
    phiF(i,:) = [phi0 phi(i,:)];
end
zeta = [0 zeta];
figure(1)
plotArray=['b- ';'g- ';'r- ';'c- ';'m- ';'y- ';'k- ';'k: ';'k--'];
for i=1:length(BC1),
    semilogy(zeta,phiF(i,:),plotArray(i,:))
```

```
    hold on
end
xlabel('Distance from surface (Debye lengths)');
ylabel('Dimensionless charge density');
title('Solution of the BVP in 6.B.4 with changing dimensionless surface
density');
legend('BC=1e-7','BC=1e-6','BC=1e-5','BC=1e-4','BC=1e-3',...
    'BC=1e-2','BC=0.1','BC=1','BC=10', ...
    'Location','BestOutside')
% We'll show BC - deriv at 0 to make sure that they're identical
for i=1:length(BC1)
    resid(i) = LzeroP*phiF(i,1)+LoneP*phiF(i,2)+LtwoP*phiF(i,3) +
BC1(i);
end
maxResidual = max(resid)
% Which is close enough to 0 for me
```

---

```
function f = ta_BVP2(phi,zeta,infLength,BC1)

LzeroP = -(zeta(1) + zeta(2))/(zeta(1)*zeta(2));
LoneP = zeta(2)/(zeta(1)*(zeta(2)-zeta(1)));
LtwoP = -zeta(1)/(zeta(2)*(zeta(2)-zeta(1)));
zetaPlusHalf = (zeta(2)+zeta(1))/2;
zetaMinusHalf = zeta(1)/2;
zetaHalfDiff = zetaPlusHalf - zetaMinusHalf;
f(1) = -phi(2)/((zeta(2)-zeta(1))*zetaHalfDiff) ...
    + phi(1)/zetaHalfDiff*(1/(zeta(2)-zeta(1)) + 1/(zeta(1))) ...
    - 1/(zeta(1)*zetaHalfDiff) ...
       *(-BC1/LzeroP - LoneP/LzeroP*phi(1) - LtwoP/LzeroP*phi(2)) ...
    + .5*exp(phi(1)) - .5*exp(-phi(1));
for i=2:(length(phi)-1),
    zetaPlusHalf = (zeta(i+1)+zeta(i))/2;
    zetaMinusHalf = (zeta(i)+zeta(i-1))/2;
    zetaHalfDiff = zetaPlusHalf - zetaMinusHalf;
    f(i) = -phi(i+1)/((zeta(i+1)-zeta(i))*zetaHalfDiff) ...
        + phi(i)/zetaHalfDiff*(1/(zeta(i+1)-zeta(i)) + 1/(zeta(i)-
zeta(i-1))) ...
        -phi(i-1)/((zeta(i)-zeta(i-1))*zetaHalfDiff) ...
        + .5*exp(phi(i)) - .5*exp(-phi(i));
end
zetaPlusHalf = (infLength+zeta(end))/2;
zetaMinusHalf = (zeta(end)+zeta(end-1))/2;
zetaHalfDiff = zetaPlusHalf - zetaMinusHalf;
f(length(phi)) = phi(end)/zetaHalfDiff*(1/(infLength-zeta(end)) + ...
                                        1/(zeta(end)-zeta(end-1))) ...
              - phi(end-1)/((zeta(end)-zeta(end-1))*zetaHalfDiff) ...
              +.5*exp(phi(end)) - .5*exp(-phi(end));
```

*3. 6.C.5*

The calculation of extraction was just a simple double integral of all of the AB and A in the liquid, since we assume that AB is not volatile. Normalized to the length of the film, this gives us an average extraction per unit area of surface. These numbers may vary

significantly based on the number of grid points; a range of acceptable values will be established later, but here is one set of values that is likely to be acceptable, in mol/$m^2$:

**Dirichlet boundary conditions (surface concentration of AB and B are 0):**

withExtraction =

1.8624e-003

withoutExtraction =

1.8624e-003

**Neumann boundary conditions (no flux at surface for AB, B):**

withExtraction =

1.8646e-003

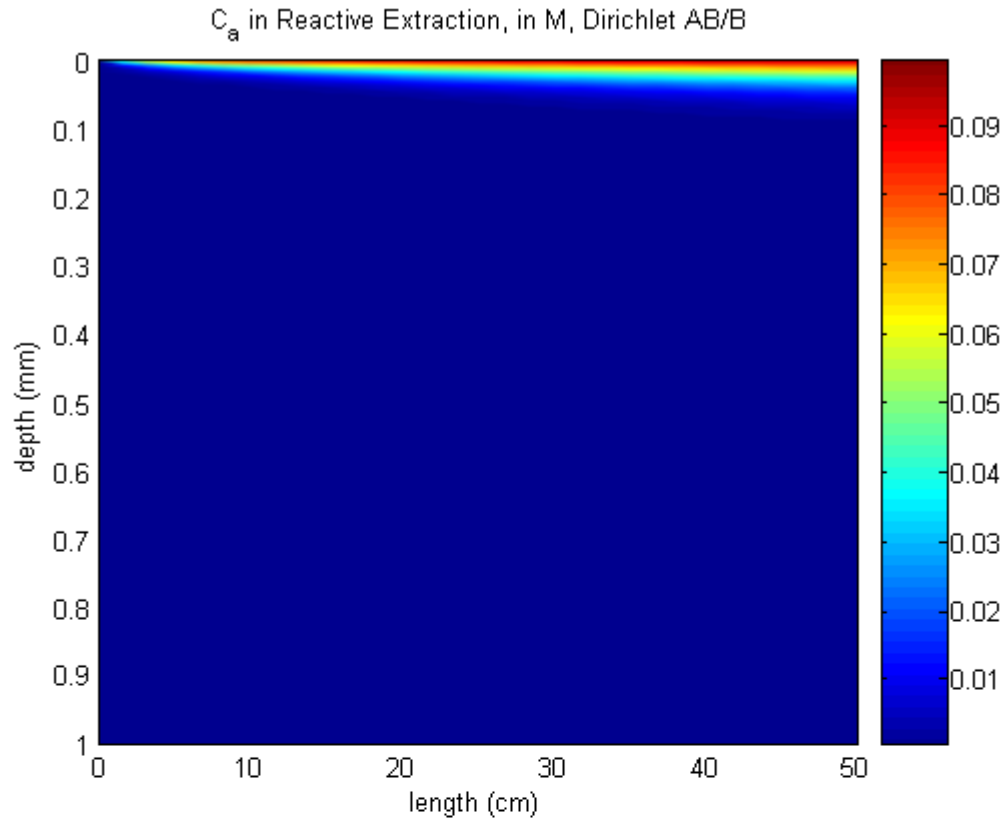
withoutExtraction =

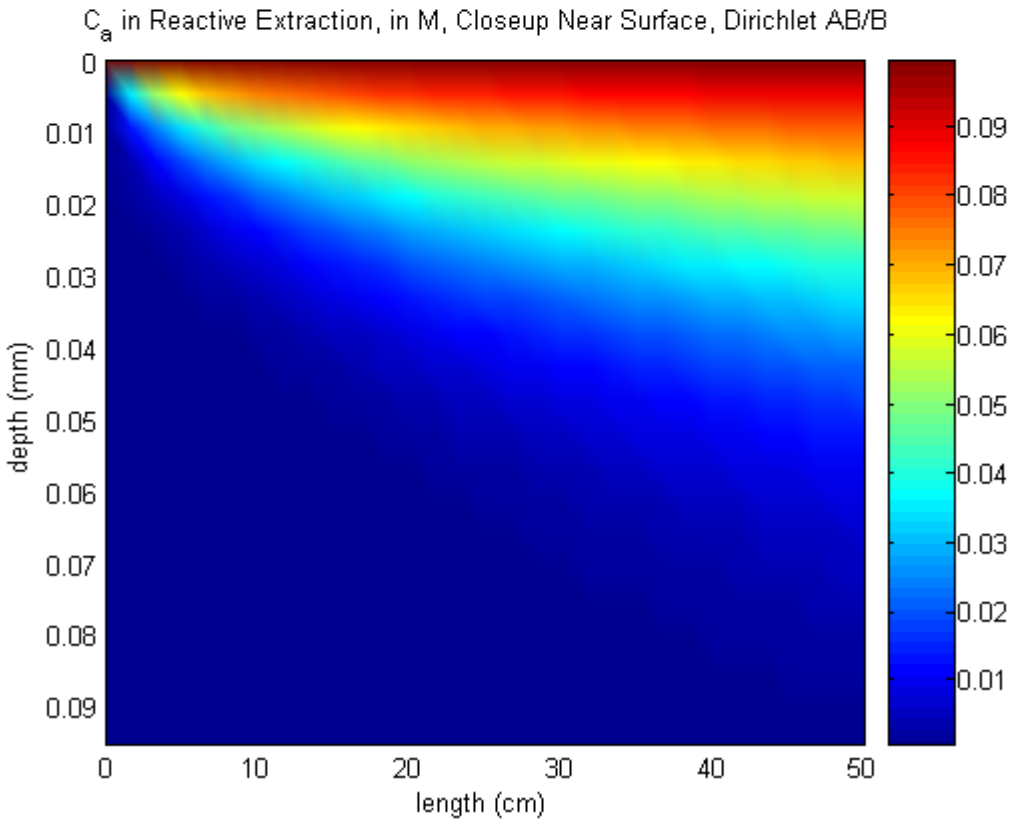
1.8624e-003

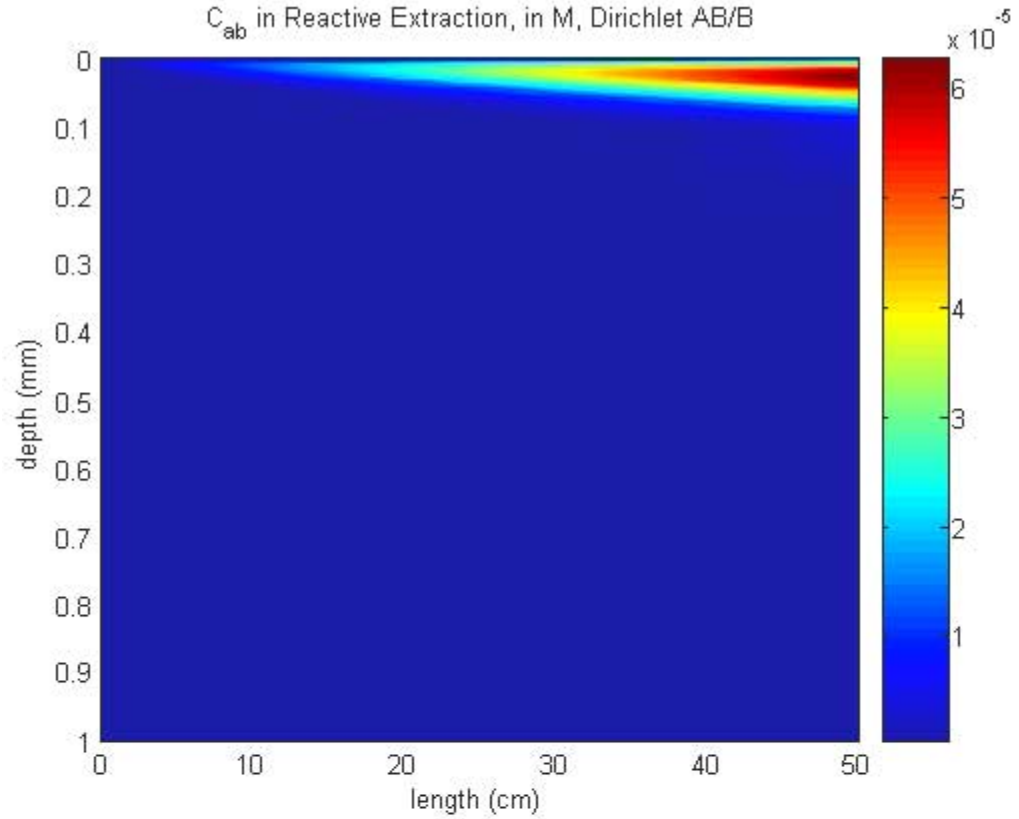
So it is acceptable for the Dirichlet boundary condition to see no relative increase in extraction… this is because much of that AB (and B, for that matter) has evaporated off into the air. We'd need to model that as well in order to get a fuller picture of what has happened. As Dr. Beers pointed out in class, a way to inelegantly force things to look like we'd like is to apply the Neumann condition for AB and B… in this case, we see an increase in extractive potential. We note that it is small, though this may be expected because the reaction is relatively slow (less than 0.1 M/s) and the fluid is flowing fast (0.8 m/s) over a short span (0.5 m). If you were to crank up the value of the reaction constant k, you'd see a significant improvement of extraction for the reactive extraction.

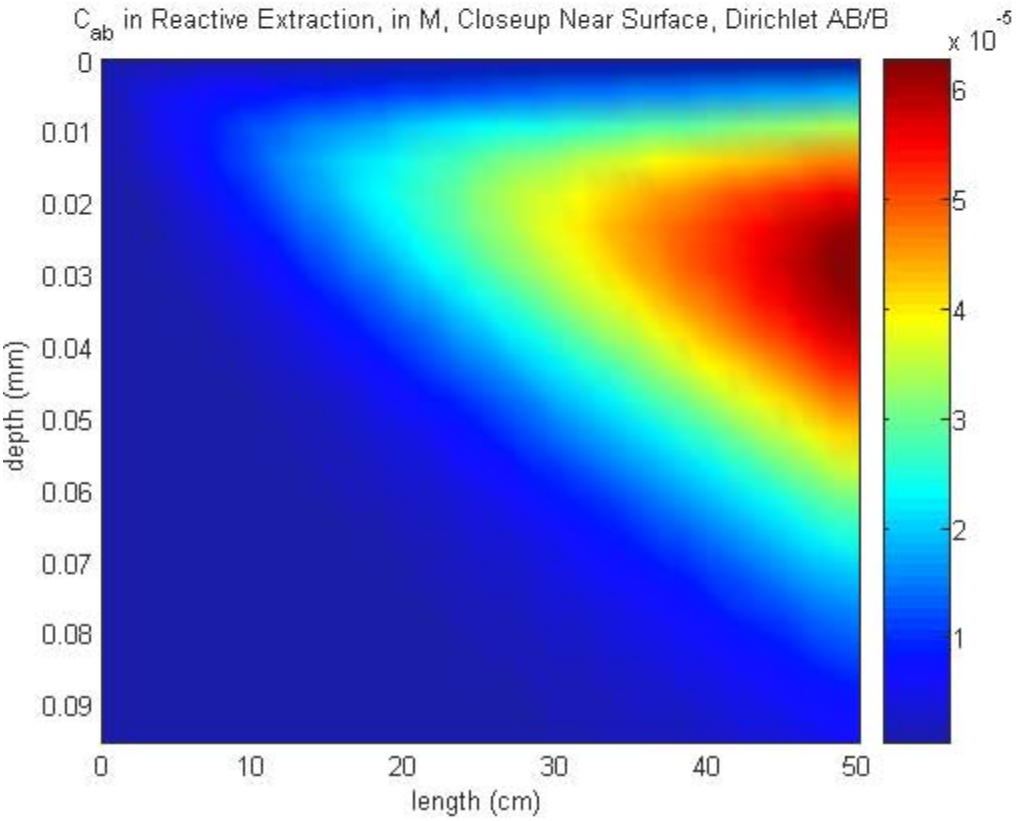
And here are plots and code… first for Dirichlet boundary conditions, then for von Neumann boundary conditions.

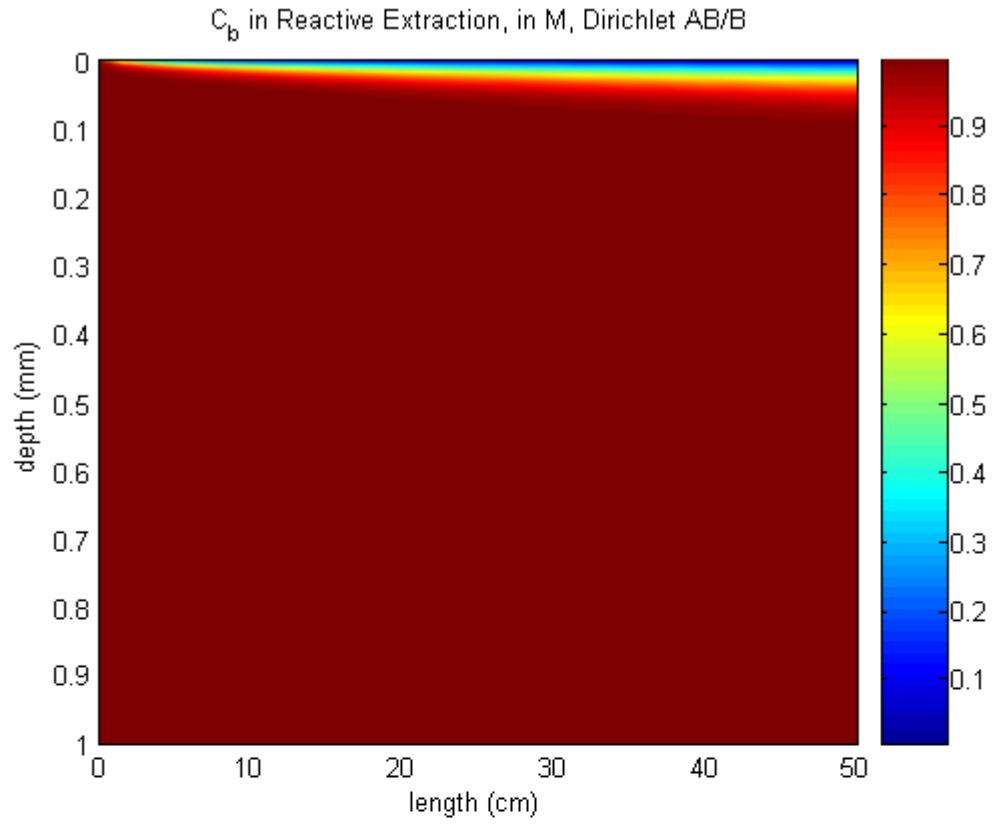
**Results for Dirichlet conditions on AB and B:**

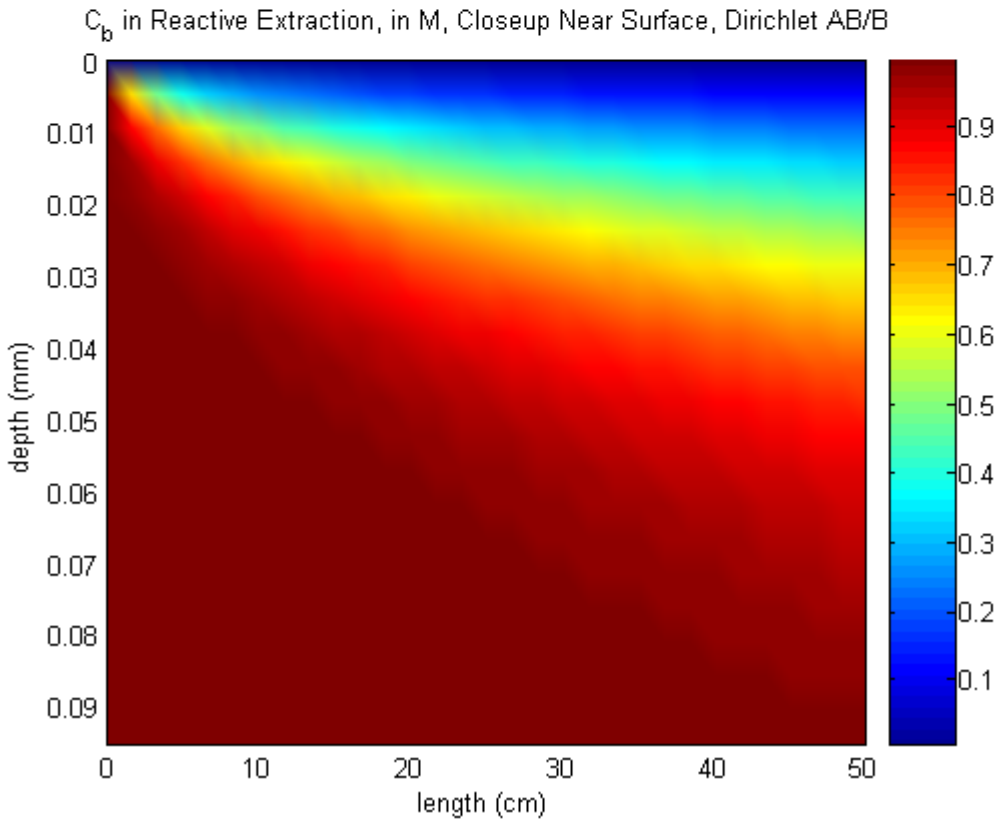


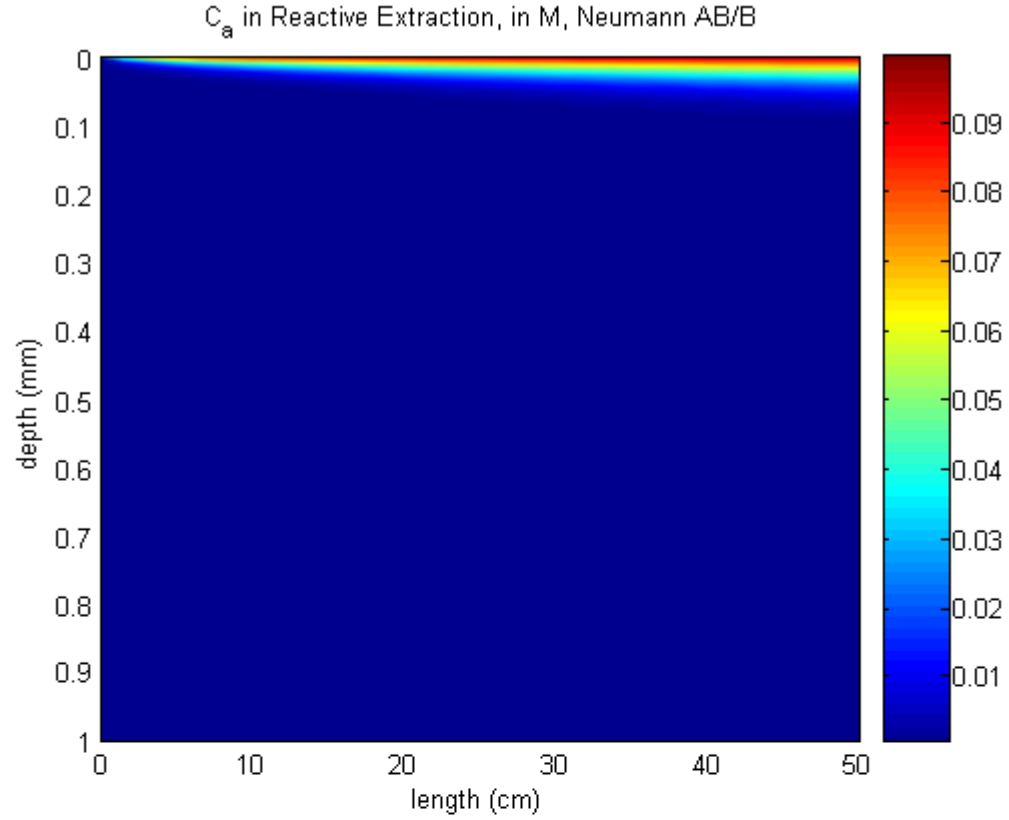

$C_{ab}$ in Reactive Extraction, in M, Dirichlet AB/B
x 10^-5
depth (mm)
length (cm)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
1
2
3
4
5
6

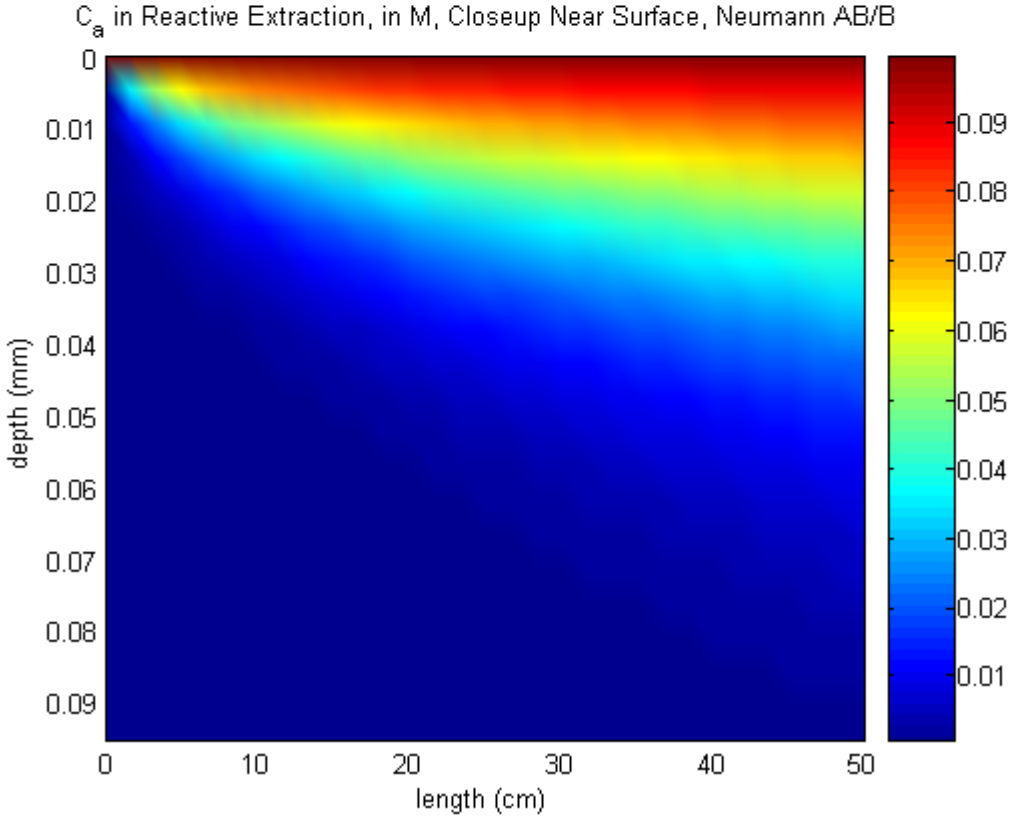

$C_{ab}$ in Reactive Extraction, in M, Closeup Near Surface, Dirichlet AB/B
x 10^-5
depth (mm)
length (cm)
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0
10
20
30
40
50
1
2
3
4
5
6

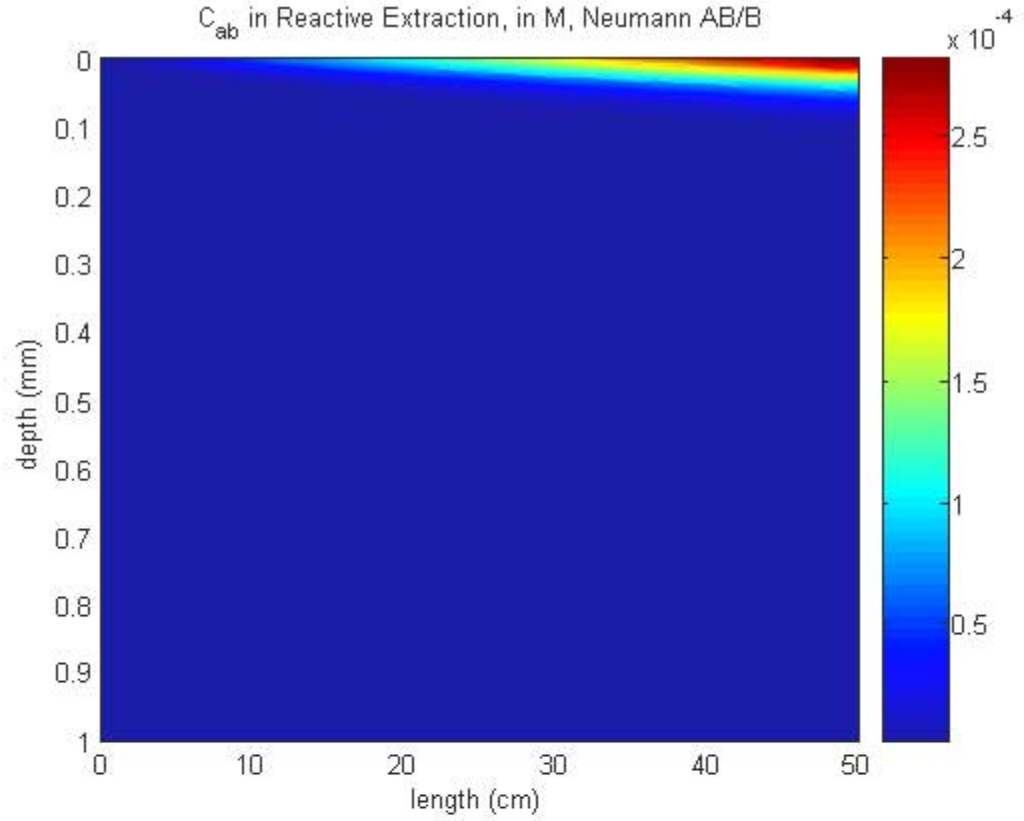

$C_b$ in Reactive Extraction, in M, Dirichlet AB/B
depth (mm)
length (cm)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

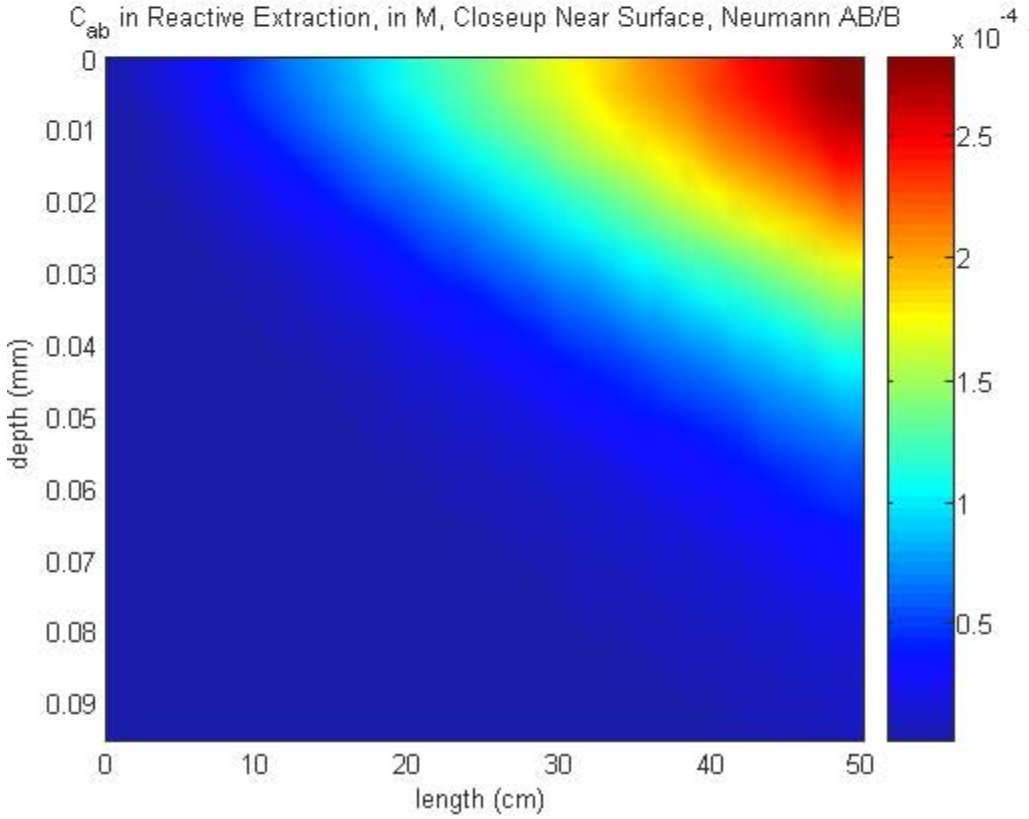

$C_b$ in Reactive Extraction, in M, Closeup Near Surface, Dirichlet AB/B
depth (mm)
length (cm)
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0
10
20
30
40
50
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

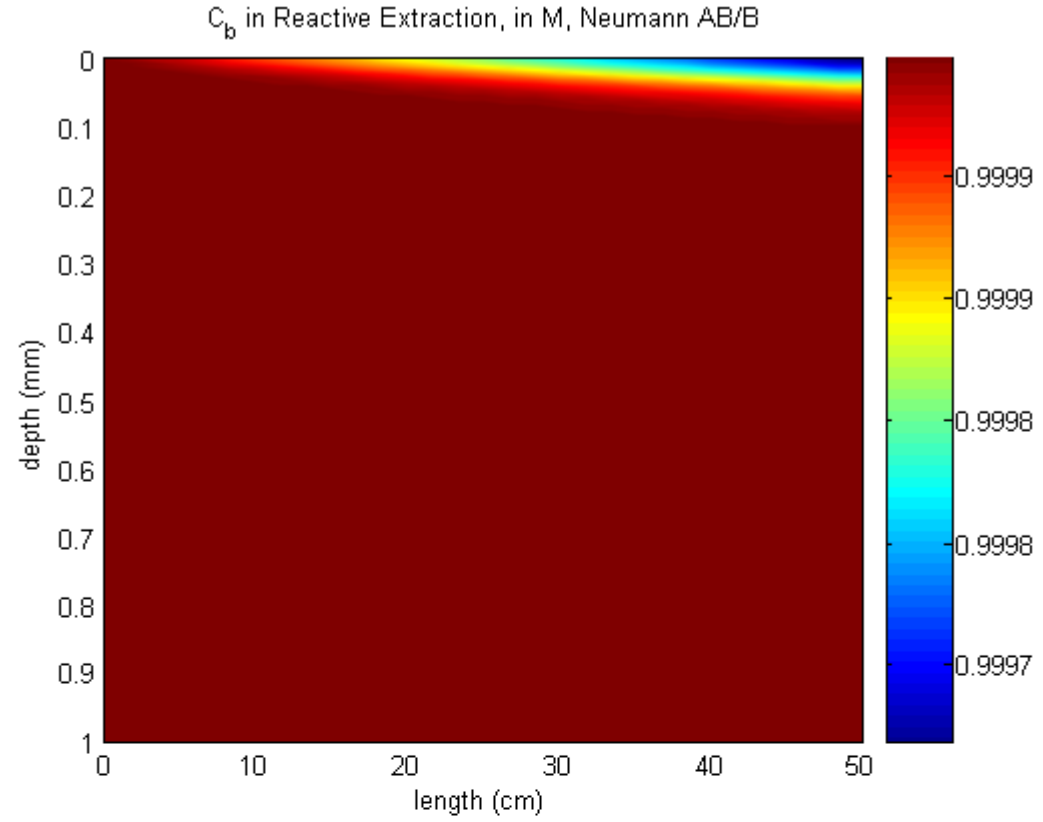
**Results for Neumann conditions on AB and B:**

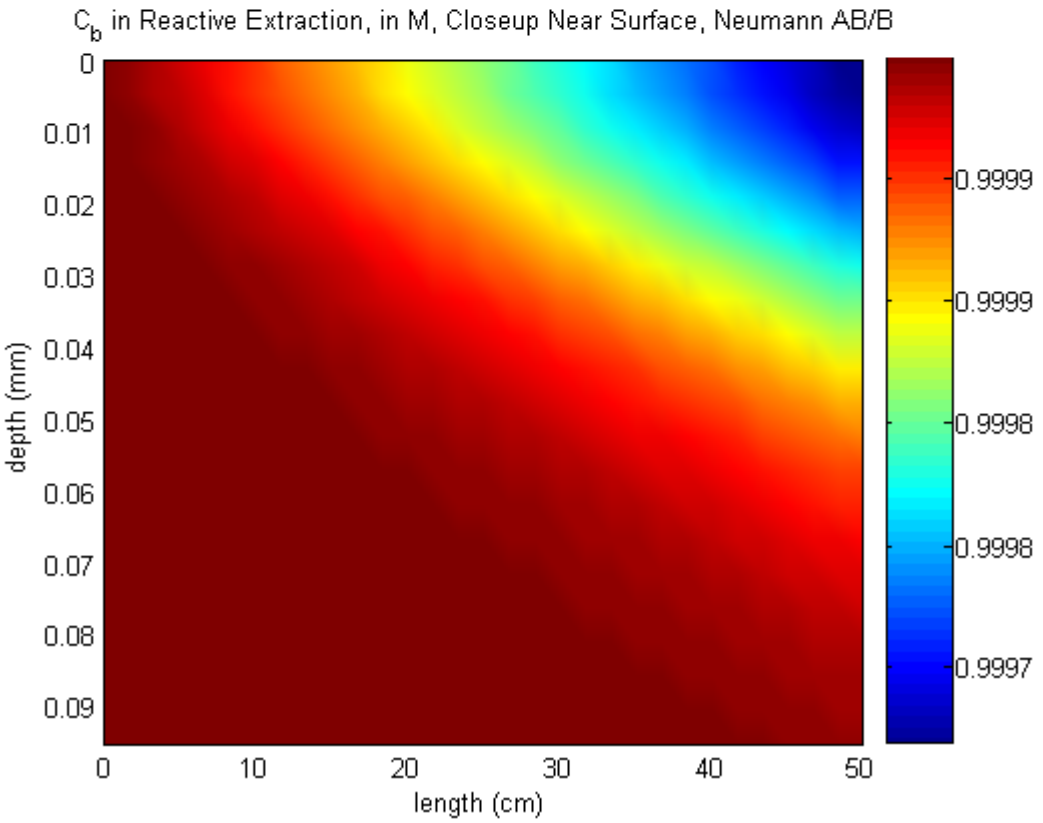



$C_{ab}$ in Reactive Extraction, in M, Neumann AB/B
x 10^-4
depth (mm)
length (cm)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
2.5
2
1.5
1
0.5


$C_{ab}$ in Reactive Extraction, in M, Closeup Near Surface, Neumann AB/B
x 10^-4
depth (mm)
length (cm)
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0
10
20
30
40
50
2.5
2
1.5
1
0.5


$C_b$ in Reactive Extraction, in M, Neumann AB/B
depth (mm)
length (cm)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
0.9999
0.9999
0.9998
0.9998
0.9997


$C_b$ in Reactive Extraction, in M, Closeup Near Surface, Neumann AB/B
depth (mm)
length (cm)
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0
10
20
30
40
50
0.9999
0.9999
0.9998
0.9998
0.9997

**Code to get Dirichlet answers:**

```
% Problem 6.C.5

clear all; close all;

% Lots of parameter setting.
% M is in length direction, N is in depth direction
param.M = 29;
% Most of the activity in N will be near the surface, so put
%  more points there.
surfaceRes = 20;
deepRes = 9;
param.N = surfaceRes + deepRes;
param.L = .5; % meters
param.b = 1e-3; % meters
param.dz = param.L/(param.M + 1);
param.dy = param.b/(param.N + 1);
param.theta = 80;  % degrees
param.rho = 1e3;  % kg/m^3
param.mu = .001; % kg/(m*s), or 1 cP
param.g = 9.8; % m/s^2
% Note, can use atm here because HaPa always come together.
param.Ha = 1e6;  %  mol/(m^3 * atm)
param.Pa = 1e-4;  % atm
param.Da = 1e-9;   % m^2/s
param.Db = 1e-9;  % m^2/s
param.Dab = 1e-9;  % m^2/s
param.Cb0 = 1e3;  % mol/m^3
param.k = 1e-5;  % m^3/(mol * s)
param.Kcap = 1e6;  %  m^3/mol
surfaceY = linspace(0,param.b/10,surfaceRes+2);
deepY = linspace(param.b/10,param.b,deepRes+1);
param.y = [surfaceY(1:end-1) deepY]; % includes endpoints
% These parameters make vz be in m/s, and everything ought to
%  be consistent.

% Need a reasonable initial guess
zPlot = linspace(0,param.L,param.M+2);
zManip = zPlot(2:end-1);
yPlot = param.y;
yManip = param.y(2:end-1);
% Linear change seems reasonable enough for me.
CaGuess = interp1([0 param.b],[param.Ha*param.Pa 0],yManip);
CbGuess = interp1([0 param.b],[0 param.Cb0],yManip);
CabGuess = interp1([0 param.b],[0 param.Cb0],yManip);
% We pack/stack our values.
uniformCguess(1:3:3*param.N) = CaGuess;
uniformCguess(2:3:3*param.N) = CbGuess;
uniformCguess(3:3:3*param.N) = CabGuess;
Cguess = [];
% This is the total guess vector.
for i=1:param.M,
    Cguess = [Cguess uniformCguess*(param.M-i)/param.M];
```

```
end
% Now we run fsolve
options = optimset('Jacobian','on');
totProf = fsolve(@(x)ta_filmReacWJacUnevenY(x,param),Cguess,options);
% We unpack our values.
Ca = totProf(1:3:end);
Cb = totProf(2:3:end);
Cab = totProf(3:3:end);
% Make them into matrices suitable for plotting.
Ca = reshape(Ca,param.N,param.M);
Cb = reshape(Cb,param.N,param.M);
Cab = reshape(Cab,param.N,param.M);
% Fill in boundary values
Ca = [zeros(param.N,1) Ca];
Ca = [Ca (4/3*Ca(:,end) - 1/3*Ca(:,end-1))];
Ca = [ones(1,param.M+2)*param.Ha*param.Pa; Ca];
Ca = [Ca; (4/3*Ca(end,:) - 1/3*Ca(end-1,:))];
Cb = [param.Cb0*ones(param.N,1) Cb];
Cb = [Cb (4/3*Cb(:,end) - 1/3*Cb(:,end-1))];
Cb = [zeros(1,param.M+2); Cb];
Cb = [Cb; (4/3*Cb(end,:) - 1/3*Cb(end-1,:))];
Cab = [zeros(param.N,1) Cab];
Cab = [Cab (4/3*Cab(:,end) - 1/3*Cab(:,end-1))];
Cab = [zeros(1,param.M+2); Cab];
Cab = [Cab; (4/3*Cab(end,:) - 1/3*Cab(end-1,:))];
Castore = Ca;
Cbstore = Cb;
Cabstore = Cab;
% And we plot.
figure(1);
pcolor(1e2*zPlot,1e3*yPlot,1e-3*Ca);
shading('interp');
colorbar;
axis ij;
title('C_a in Reactive Extraction, in M, Dirichlet AB/B')
xlabel('length (cm)')
ylabel('depth (mm)')
figure(2);
pcolor(1e2*zPlot,1e3*yPlot,1e-3*Cb);
shading('interp');
colorbar;
axis ij;
title('C_b in Reactive Extraction, in M, Dirichlet AB/B')
xlabel('length (cm)')
ylabel('depth (mm)')
figure(3);
pcolor(1e2*zPlot,1e3*yPlot,1e-3*Cab);
shading('interp');
colorbar;
axis ij;
title('C_a_b in Reactive Extraction, in M, Dirichlet AB/B')
xlabel('length (cm)')
ylabel('depth (mm)')
figure(4);
pcolor(1e2*zPlot,1e3*yPlot(1:surfaceRes+1),1e-3*Ca(1:surfaceRes+1,:))
shading('interp');
colorbar;
```

```
axis ij;
title('C_a in Reactive Extraction, in M, Closeup Near Surface,
Dirichlet AB/B')
xlabel('length (cm)')
ylabel('depth (mm)')
figure(5);
pcolor(1e2*zPlot,1e3*yPlot(1:surfaceRes+1),1e-3*Cb(1:surfaceRes+1,:))
shading('interp');
colorbar;
axis ij;
title('C_b in Reactive Extraction, in M, Closeup Near Surface,
Dirichlet AB/B')
xlabel('length (cm)')
ylabel('depth (mm)')
figure(6);
pcolor(1e2*zPlot,1e3*yPlot(1:surfaceRes+1),1e-3*Cab(1:surfaceRes+1,:))
shading('interp');
colorbar;
axis ij;
title('C_a_b in Reactive Extraction, in M, Closeup Near Surface,
Dirichlet AB/B')
xlabel('length (cm)')
ylabel('depth (mm)')

% Find extraction: simple integration of what's in the flowing
%  fluid... this is reasonable because we assume AB is not
%  volatile.
trapz(zPlot,trapz(yPlot,Ca,1))/param.L
trapz(zPlot,trapz(yPlot,Cab,1))/param.L
(trapz(zPlot,trapz(yPlot,Ca,1))+trapz(zPlot,trapz(yPlot,Cab,1)))/param.
L

% Now change k and find extraction
param.k = 0;

CaGuess = interp1([0 param.b],[param.Ha*param.Pa 0],yManip);
CbGuess = interp1([0 param.b],[0 param.Cb0],yManip);
CabGuess = zeros(size(yManip));
uniformCguess(1:3:3*param.N) = CaGuess;
uniformCguess(2:3:3*param.N) = CbGuess;
uniformCguess(3:3:3*param.N) = CabGuess;
Cguess = [];
for i=1:param.M,
    Cguess = [Cguess uniformCguess*(param.M-i)/param.M];
end
totProf = fsolve(@(x)ta_filmReacWJacUnevenY(x,param),Cguess,options);
Ca = totProf(1:3:end);
Cb = totProf(2:3:end);
Cab = totProf(3:3:end);
Ca = reshape(Ca,param.N,param.M);
Cb = reshape(Cb,param.N,param.M);
Cab = reshape(Cab,param.N,param.M);
% Fill in boundary values
Ca = [zeros(param.N,1) Ca];
Ca = [Ca (4/3*Ca(:,end) - 1/3*Ca(:,end-1))];
Ca = [ones(1,param.M+2)*param.Ha*param.Pa; Ca];
```

```
Ca = [Ca; (4/3*Ca(end,:) - 1/3*Ca(end-1,:))];
Cb = [param.Cb0*ones(param.N,1) Cb];
Cb = [Cb (4/3*Cb(:,end) - 1/3*Cb(:,end-1))];
Cb = [zeros(1,param.M+2); Cb];
Cb = [Cb; (4/3*Cb(end,:) - 1/3*Cb(end-1,:))];
Cab = [zeros(param.N,1) Cab];
Cab = [Cab (4/3*Cab(:,end) - 1/3*Cab(:,end-1))];
Cab = [zeros(1,param.M+2); Cab];
Cab = [Cab; (4/3*Cab(end,:) - 1/3*Cab(end-1,:))];

trapz(zPlot,trapz(yPlot,Ca,1))/param.L
trapz(zPlot,trapz(yPlot,Cab,1))/param.L
(trapz(zPlot,trapz(yPlot,Ca,1))+trapz(zPlot,trapz(yPlot,Cab,1)))/param.
L
```

---

```
function [f,jac] = ta_filmReacWJacUnevenY(Cvec,param)

% Expect param to have param.M = number of points in z (plate)
%  direction and param.N = number of points in y (depth) direction
% and values of dy and dz

% Assume all things in param are in consistent units

Ca = Cvec(1:3:end);
Cb = Cvec(2:3:end);
Cab = Cvec(3:3:end);
M = param.M;
N = param.N;
numPoints = M*N;
if (length(Cvec) ~= numPoints*3),
    return;
end
dy = param.dy;
dz = param.dz;
Da = param.Da;
Db = param.Db;
Dab = param.Dab;
k = param.k;
Kcap = param.Kcap;
Ha = param.Ha;
Pa = param.Pa;
Cb0 = param.Cb0;
jac = spalloc(3*numPoints,3*numPoints,7*numPoints);
y = param.y;  % assumed to have the endpoints in it



for i=1:M,
    for j=1:N,
        index = ta_getIndex(i,j,param);
        jacIndex = ta_getJacIndex(i,j,param);
        % add 1 for a, 2 for b, 3 for ab
        vz = ta_getVz(j,param);
        yj = j+1; % The "j" for the y index
```

```
        halfDiff = (y(yj+1) - y(yj-1))/2;
        fa(index) = -vz*Ca(index)/dz ...
            - Da*Ca(index)/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) -
y(yj-1))) ...
            - Da*2*Ca(index)/dz^2 - k*(Ca(index)*Cb(index) - ...
                                       1/Kcap*Cab(index));
        jac(jacIndex+1,jacIndex+1) = -vz/dz - ...
            Da/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) - y(yj-1))) -
Da*2/dz^2 ...
            - k*Cb(index);
        jac(jacIndex+1,jacIndex+2) = -k*Ca(index);
        jac(jacIndex+1,jacIndex+3) = -1/Kcap;
        fb(index) = -vz*Cb(index)/dz ...
            - Db*Cb(index)/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) -
y(yj-1))) ...
            - Db*2*Cb(index)/dz^2 - k*(Ca(index)*Cb(index) - ...
                                       1/Kcap*Cab(index));
        jac(jacIndex+2,jacIndex+1) = -k*Cb(index);
        jac(jacIndex+2,jacIndex+2) = -vz/dz - ...
            Db/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) - y(yj-1))) -
Da*2/dz^2 ...
            - k*Ca(index);
        jac(jacIndex+2,jacIndex+3) = 1/Kcap;
        fab(index) = -vz*Cab(index)/dz ...
            - Dab*Cab(index)/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) -
y(yj-1))) ... ...
            - Dab*2*Cab(index)/dz^2 + k*(Ca(index)*Cb(index) - ...
                                       1/Kcap*Cab(index));
        jac(jacIndex+3,jacIndex+1) = k*Cb(index);
        jac(jacIndex+3,jacIndex+2) = k*Ca(index);
        jac(jacIndex+3,jacIndex+3) = -vz/dz ...
            -Dab/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) - y(yj-1))) -
Dab*2/dz^2 ...
            - 1/Kcap;
        if (i==1),
            fb(index) = fb(index) + vz*Cb0/dz + Db*Cb0/dz^2;
        else
            indexIminus = ta_getIndex(i-1,j,param);
            jacIndexIminus = ta_getJacIndex(i-1,j,param);
            fa(index) = fa(index) + vz*Ca(indexIminus)/dz ...
                + Da*Ca(indexIminus)/dz^2;
            jac(jacIndex+1,jacIndexIminus+1) = vz/dz + Da/dz^2;
            fb(index) = fb(index) + vz*Cb(indexIminus)/dz ...
                + Db*Cb(indexIminus)/dz^2;
            jac(jacIndex+2,jacIndexIminus+2) = vz/dz + Db/dz^2;
            fab(index) = fab(index) + vz*Cab(indexIminus)/dz ...
                + Dab*Cab(indexIminus)/dz^2;
            jac(jacIndex+3,jacIndexIminus+3) = vz/dz + Dab/dz^2;
        end
        if (i==M),
            CaIplus = 4/3*Ca(index)-1/3*Ca(indexIminus);
            CbIplus = 4/3*Cb(index)-1/3*Cb(indexIminus);
            CabIplus = 4/3*Cab(index)-1/3*Cab(indexIminus);
            fa(index) = fa(index) + Da*CaIplus/dz^2;
            jac(jacIndex+1,jacIndex+1) = jac(jacIndex+1,jacIndex+1) ...
                + Da/dz^2*(4/3);
```

```
            jac(jacIndex+1,jacIndexIminus+1) =
jac(jacIndex+1,jacIndexIminus+1) ...
                + Da/dz^2*(-1/3);
            fb(index) = fb(index) + Db*CbIplus/dz^2;
            jac(jacIndex+2,jacIndex+2) = jac(jacIndex+2,jacIndex+2) ...
                + Db/dz^2*(4/3);
            jac(jacIndex+2,jacIndexIminus+2) =
jac(jacIndex+2,jacIndexIminus+2) ...
                + Db/dz^2*(-1/3);
            fab(index) = fab(index) + Dab*CabIplus/dz^2;
            jac(jacIndex+3,jacIndex+3) = jac(jacIndex+3,jacIndex+3) ...
                + Dab/dz^2*(4/3);
            jac(jacIndex+3,jacIndexIminus+3) =
jac(jacIndex+3,jacIndexIminus+3) ...
                + Dab/dz^2*(-1/3);
        else
            indexIplus = ta_getIndex(i+1,j,param);
            jacIndexIplus = ta_getJacIndex(i+1,j,param);
            fa(index) = fa(index) + Da*Ca(indexIplus)/dz^2;
            jac(jacIndex+1,jacIndexIplus+1) = Da/dz^2;
            fb(index) = fb(index) + Db*Cb(indexIplus)/dz^2;
            jac(jacIndex+2,jacIndexIplus+2) = Db/dz^2;
            fab(index) = fab(index) + Dab*Cab(indexIplus)/dz^2;
            jac(jacIndex+3,jacIndexIplus+3) = Dab/dz^2;
        end
        if (j==1),
            fa(index) = fa(index) + Da*Ha*Pa/halfDiff/(y(yj)-y(yj-1));
        else
            indexJminus = ta_getIndex(i,j-1,param);
            jacIndexJminus = ta_getJacIndex(i,j-1,param);
            fa(index) = fa(index) + Da*Ca(indexJminus)/halfDiff/(y(yj)-
y(yj-1));
            jac(jacIndex+1,jacIndexJminus+1) = Da/halfDiff/(y(yj)-y(yj-
1));
            fb(index) = fb(index) + Db*Cb(indexJminus)/halfDiff/(y(yj)-
y(yj-1));
            jac(jacIndex+2,jacIndexJminus+2) = Db/halfDiff/(y(yj)-y(yj-
1));
            fab(index) = fab(index) +
Dab*Cab(indexJminus)/halfDiff/(y(yj)-y(yj-1));
            jac(jacIndex+3,jacIndexJminus+3) = Dab/halfDiff/(y(yj)-
y(yj-1));
        end
        if (j==N),
            CaJplus = 4/3*Ca(index) - 1/3*Ca(indexJminus);
            CbJplus = 4/3*Cb(index) - 1/3*Cb(indexJminus);
            CabJplus = 4/3*Cab(index) - 1/3*Cab(indexJminus);
            fa(index) = fa(index) + Da*CaJplus/halfDiff/(y(yj+1)-
y(yj));
            jac(jacIndex+1,jacIndex+1) = jac(jacIndex+1,jacIndex+1) ...
                + Da/halfDiff/(y(yj+1)-y(yj))*(4/3);
            jac(jacIndex+1,jacIndexJminus+1) =
jac(jacIndex+1,jacIndexJminus+1) ...
                + Da/halfDiff/(y(yj+1)-y(yj))*(-1/3);
            fb(index) = fb(index) + Db*CbJplus/halfDiff/(y(yj+1)-
y(yj));
            jac(jacIndex+2,jacIndex+2) = jac(jacIndex+2,jacIndex+2) ...
```

```
                + Db/halfDiff/(y(yj+1)-y(yj))*(4/3);
            jac(jacIndex+2,jacIndexJminus+2) =
jac(jacIndex+2,jacIndexJminus+2) ...
                + Db/halfDiff/(y(yj+1)-y(yj))*(-1/3);
            fab(index) = fab(index) + Dab*CabJplus/halfDiff/(y(yj+1)-
y(yj));
            jac(jacIndex+3,jacIndex+3) = jac(jacIndex+3,jacIndex+3) ...
                + Dab/halfDiff/(y(yj+1)-y(yj))*(4/3);
            jac(jacIndex+3,jacIndexJminus+3) =
jac(jacIndex+3,jacIndexJminus+3) ...
                + Dab/halfDiff/(y(yj+1)-y(yj))*(-1/3);
        else
            indexJplus = ta_getIndex(i,j+1,param);
            jacIndexJplus = ta_getJacIndex(i,j+1,param);
            fa(index) = fa(index) +
Da*Ca(indexJplus)/halfDiff/(y(yj+1)-y(yj));
            jac(jacIndex+1,jacIndexJplus+1) = Da/halfDiff/(y(yj+1)-
y(yj));
            fb(index) = fb(index) +
Db*Cb(indexJplus)/halfDiff/(y(yj+1)-y(yj));
            jac(jacIndex+2,jacIndexJplus+2) = Db/halfDiff/(y(yj+1)-
y(yj));
            fab(index) = fab(index) +
Dab*Cab(indexJplus)/halfDiff/(y(yj+1)-y(yj));
            jac(jacIndex+3,jacIndexJplus+3) = Dab/halfDiff/(y(yj+1)-
y(yj));
        end
    end
end
f(1:3:3*numPoints) = fa;
f(2:3:3*numPoints) = fb;
f(3:3:3*numPoints) = fab;
```

---

```
function index = ta_getIndex(i,j,param)

% param has param.M which is number of i points and param.N
%  which is number of j points

% If we assume that N < M, it makes sense to do it this way to
%  reduce the amount of fill-in (band is only N away instead
%  of M).

index = (i-1)*param.N + j;
```

---

```
function index = ta_getJacIndex(i,j,param)


% param has param.M which is number of i points and param.N
%  which is number of j points

% If we assume that N < M, it makes sense to do it this way to
%  reduce the amount of fill-in (band is only N away instead
%  of M).

index = 3*(i-1)*param.N + (j-1)*3;
```

```
function vz = ta_getVz(j,param)

% param has param.N, the number of points in j direction
% So we want there to actually be N+1 points in our
%  math here, because we want the last point to be the plate,
%  but the plate isn't in our finite difference.  We want
%  n points, but none of them are actually the plate.

% Since our equation calls for 1-(h/H)^2, we realize this is
%  equivalent to 1-(j/(N+1))^2, where at j=0 we'd have the full
%  speed and at j=N+1 we'd have no-slip.

% Param also needs to have theta (angle in degrees with
%  respect to gravity), rho (density of liquid), mu (viscosity
%  of liquid), g (acceleration due to gravity), and b (thickness
%  of film), all in the same units.

thetaRad = param.theta/180*pi;
vz = param.rho*param.g*cos(thetaRad)*(param.b)^2/2/param.mu ...
    * (1 - (j/(param.N+1))^2);
```

**Code to get Neumann answers:**

Only need to change boundary values in filmReac function.

```
function [f,jac] = ta_filmReacWJacUnevenY(Cvec,param)

% Expect param to have param.M = number of points in z (plate)
%  direction and param.N = number of points in y (depth) direction
% and values of dy and dz

% Assume all things in param are in consistent units

Ca = Cvec(1:3:end);
Cb = Cvec(2:3:end);
Cab = Cvec(3:3:end);
M = param.M;
N = param.N;
numPoints = M*N;
if (length(Cvec) ~= numPoints*3),
    return;
end
dy = param.dy;
dz = param.dz;
Da = param.Da;
Db = param.Db;
Dab = param.Dab;
k = param.k;
Kcap = param.Kcap;
Ha = param.Ha;
```

```
Pa = param.Pa;
Cb0 = param.Cb0;
jac = spalloc(3*numPoints,3*numPoints,7*numPoints);
y = param.y;  % assumed to have the endpoints in it



for i=1:M,
    for j=1:N,
        index = ta_getIndex(i,j,param);
        jacIndex = ta_getJacIndex(i,j,param);
        % add 1 for a, 2 for b, 3 for ab
        vz = ta_getVz(j,param);
        yj = j+1; % The "j" for the y index
        halfDiff = (y(yj+1) - y(yj-1))/2;
        fa(index) = -vz*Ca(index)/dz ...
            - Da*Ca(index)/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) -
y(yj-1))) ...
            - Da*2*Ca(index)/dz^2 - k*(Ca(index)*Cb(index) - ...
                                       1/Kcap*Cab(index));
        jac(jacIndex+1,jacIndex+1) = -vz/dz - ...
            Da/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) - y(yj-1))) -
Da*2/dz^2 ...
            - k*Cb(index);
        jac(jacIndex+1,jacIndex+2) = -k*Ca(index);
        jac(jacIndex+1,jacIndex+3) = -1/Kcap;
        fb(index) = -vz*Cb(index)/dz ...
            - Db*Cb(index)/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) -
y(yj-1))) ...
            - Db*2*Cb(index)/dz^2 - k*(Ca(index)*Cb(index) - ...
                                       1/Kcap*Cab(index));
        jac(jacIndex+2,jacIndex+1) = -k*Cb(index);
        jac(jacIndex+2,jacIndex+2) = -vz/dz - ...
            Db/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) - y(yj-1))) -
Da*2/dz^2 ...
            - k*Ca(index);
        jac(jacIndex+2,jacIndex+3) = 1/Kcap;
        fab(index) = -vz*Cab(index)/dz ...
            - Dab*Cab(index)/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) -
y(yj-1))) ... ...
            - Dab*2*Cab(index)/dz^2 + k*(Ca(index)*Cb(index) - ...
                                       1/Kcap*Cab(index));
        jac(jacIndex+3,jacIndex+1) = k*Cb(index);
        jac(jacIndex+3,jacIndex+2) = k*Ca(index);
        jac(jacIndex+3,jacIndex+3) = -vz/dz ...
            -Dab/halfDiff*(1/(y(yj+1) - y(yj)) + 1/(y(yj) - y(yj-1))) -
Dab*2/dz^2 ...
            - 1/Kcap;
        if (i==1),
            fb(index) = fb(index) + vz*Cb0/dz + Db*Cb0/dz^2;
        else
            indexIminus = ta_getIndex(i-1,j,param);
            jacIndexIminus = ta_getJacIndex(i-1,j,param);
            fa(index) = fa(index) + vz*Ca(indexIminus)/dz ...
                + Da*Ca(indexIminus)/dz^2;
            jac(jacIndex+1,jacIndexIminus+1) = vz/dz + Da/dz^2;
```

```
            fb(index) = fb(index) + vz*Cb(indexIminus)/dz ...
                + Db*Cb(indexIminus)/dz^2;
            jac(jacIndex+2,jacIndexIminus+2) = vz/dz + Db/dz^2;
            fab(index) = fab(index) + vz*Cab(indexIminus)/dz ...
                + Dab*Cab(indexIminus)/dz^2;
            jac(jacIndex+3,jacIndexIminus+3) = vz/dz + Dab/dz^2;
        end
        if (i==M),
            CaIplus = 4/3*Ca(index)-1/3*Ca(indexIminus);
            CbIplus = 4/3*Cb(index)-1/3*Cb(indexIminus);
            CabIplus = 4/3*Cab(index)-1/3*Cab(indexIminus);
            fa(index) = fa(index) + Da*CaIplus/dz^2;
            jac(jacIndex+1,jacIndex+1) = jac(jacIndex+1,jacIndex+1) ...
                + Da/dz^2*(4/3);
            jac(jacIndex+1,jacIndexIminus+1) =
jac(jacIndex+1,jacIndexIminus+1) ...
                + Da/dz^2*(-1/3);
            fb(index) = fb(index) + Db*CbIplus/dz^2;
            jac(jacIndex+2,jacIndex+2) = jac(jacIndex+2,jacIndex+2) ...
                + Db/dz^2*(4/3);
            jac(jacIndex+2,jacIndexIminus+2) =
jac(jacIndex+2,jacIndexIminus+2) ...
                + Db/dz^2*(-1/3);
            fab(index) = fab(index) + Dab*CabIplus/dz^2;
            jac(jacIndex+3,jacIndex+3) = jac(jacIndex+3,jacIndex+3) ...
                + Dab/dz^2*(4/3);
            jac(jacIndex+3,jacIndexIminus+3) =
jac(jacIndex+3,jacIndexIminus+3) ...
                + Dab/dz^2*(-1/3);
        else
            indexIplus = ta_getIndex(i+1,j,param);
            jacIndexIplus = ta_getJacIndex(i+1,j,param);
            fa(index) = fa(index) + Da*Ca(indexIplus)/dz^2;
            jac(jacIndex+1,jacIndexIplus+1) = Da/dz^2;
            fb(index) = fb(index) + Db*Cb(indexIplus)/dz^2;
            jac(jacIndex+2,jacIndexIplus+2) = Db/dz^2;
            fab(index) = fab(index) + Dab*Cab(indexIplus)/dz^2;
            jac(jacIndex+3,jacIndexIplus+3) = Dab/dz^2;
        end
        if (j==1),
            indexJplus = ta_getIndex(i,j+1,param);
            jacIndexJplus = ta_getJacIndex(i,j+1,param);
            CbJminus = 4/3*Cb(index) - 1/3*Cb(indexJplus);
            CabJminus = 4/3*Cab(index) - 1/3*Cab(indexJplus);

            fa(index) = fa(index) + Da*Ha*Pa/halfDiff/(y(yj)-y(yj-1));

            fb(index) = fb(index) + Db*CbJminus/halfDiff/(y(yj)-y(yj-
1));
            jac(jacIndex+2,jacIndex+2) = jac(jacIndex+2,jacIndex+2) ...
                + Db/halfDiff/(y(yj)-y(yj-1))*(4/3);
            jac(jacIndex+2,jacIndexJplus+2) =
jac(jacIndex+2,jacIndexJplus+2) ...
                + Db/halfDiff/(y(yj)-y(yj-1))*(-1/3);
            fab(index) = fab(index) + Dab*CabJminus/halfDiff/(y(yj)-
y(yj-1));
```

```
            jac(jacIndex+3,jacIndex+3) = jac(jacIndex+3,jacIndex+3) ...
                + Dab/halfDiff/(y(yj)-y(yj-1))*(4/3);
            jac(jacIndex+3,jacIndexJplus+3) =
jac(jacIndex+3,jacIndexJplus+3) ...
                + Dab/halfDiff/(y(yj)-y(yj-1))*(-1/3);
        else
            indexJminus = ta_getIndex(i,j-1,param);
            jacIndexJminus = ta_getJacIndex(i,j-1,param);
            fa(index) = fa(index) + Da*Ca(indexJminus)/halfDiff/(y(yj)-
y(yj-1));
            jac(jacIndex+1,jacIndexJminus+1) = Da/halfDiff/(y(yj)-y(yj-
1));
            fb(index) = fb(index) + Db*Cb(indexJminus)/halfDiff/(y(yj)-
y(yj-1));
            jac(jacIndex+2,jacIndexJminus+2) = Db/halfDiff/(y(yj)-y(yj-
1));
            fab(index) = fab(index) +
Dab*Cab(indexJminus)/halfDiff/(y(yj)-y(yj-1));
            jac(jacIndex+3,jacIndexJminus+3) = Dab/halfDiff/(y(yj)-
y(yj-1));
        end
        if (j==N),
            CaJplus = 4/3*Ca(index) - 1/3*Ca(indexJminus);
            CbJplus = 4/3*Cb(index) - 1/3*Cb(indexJminus);
            CabJplus = 4/3*Cab(index) - 1/3*Cab(indexJminus);
            fa(index) = fa(index) + Da*CaJplus/halfDiff/(y(yj+1)-
y(yj));
            jac(jacIndex+1,jacIndex+1) = jac(jacIndex+1,jacIndex+1) ...
                + Da/halfDiff/(y(yj+1)-y(yj))*(4/3);
            jac(jacIndex+1,jacIndexJminus+1) =
jac(jacIndex+1,jacIndexJminus+1) ...
                + Da/halfDiff/(y(yj+1)-y(yj))*(-1/3);
            fb(index) = fb(index) + Db*CbJplus/halfDiff/(y(yj+1)-
y(yj));
            jac(jacIndex+2,jacIndex+2) = jac(jacIndex+2,jacIndex+2) ...
                + Db/halfDiff/(y(yj+1)-y(yj))*(4/3);
            jac(jacIndex+2,jacIndexJminus+2) =
jac(jacIndex+2,jacIndexJminus+2) ...
                + Db/halfDiff/(y(yj+1)-y(yj))*(-1/3);
            fab(index) = fab(index) + Dab*CabJplus/halfDiff/(y(yj+1)-
y(yj));
            jac(jacIndex+3,jacIndex+3) = jac(jacIndex+3,jacIndex+3) ...
                + Dab/halfDiff/(y(yj+1)-y(yj))*(4/3);
            jac(jacIndex+3,jacIndexJminus+3) =
jac(jacIndex+3,jacIndexJminus+3) ...
                + Dab/halfDiff/(y(yj+1)-y(yj))*(-1/3);
        else
            indexJplus = ta_getIndex(i,j+1,param);
            jacIndexJplus = ta_getJacIndex(i,j+1,param);
            fa(index) = fa(index) +
Da*Ca(indexJplus)/halfDiff/(y(yj+1)-y(yj));
            jac(jacIndex+1,jacIndexJplus+1) = Da/halfDiff/(y(yj+1)-
y(yj));
            fb(index) = fb(index) +
Db*Cb(indexJplus)/halfDiff/(y(yj+1)-y(yj));
            jac(jacIndex+2,jacIndexJplus+2) = Db/halfDiff/(y(yj+1)-
y(yj));
```

```
            fab(index) = fab(index) +
Dab*Cab(indexJplus)/halfDiff/(y(yj+1)-y(yj));
            jac(jacIndex+3,jacIndexJplus+3) = Dab/halfDiff/(y(yj+1)-
y(yj));
        end
    end
end
f(1:3:3*numPoints) = fa;
f(2:3:3*numPoints) = fb;
f(3:3:3*numPoints) = fab;
```