STEADY-STATE AND TRANSIENT ANALYSIS OF A DIFFUSION-REACTING PROCESS

Donnie Lee
UID: 604766203
ChE 109 Final Project

Table of Content

Introduction

Reaction-diffusion system models are simulations to predict the change in space and time of the concentration of one or more chemical substances. This MATLAB project studies the steady-state and transient characteristics which takes place in a membrane reactor. The chemical activity follows the mechanism below:

$$\begin{array}{rcl} A + 2B & \longrightarrow & D \\ A & \longrightarrow & P \\ B & \longrightarrow & U \\ U & \longrightarrow & B \end{array}$$

Reactants A and B enter the reactor through the left membrane at fixed concentrations of C_{AF} and C_{BF} respectively. Within the reactor, the chemicals interact according to the following system of ODE:

$$\begin{aligned} \frac{\partial C_A}{\partial t} &= \bar{D}\frac{\partial^2 C_A}{\partial r^2} - k_1 C_A C_B^2 - k_2 C_A \\ \frac{\partial C_B}{\partial t} &= \bar{D}\frac{\partial^2 C_B}{\partial r^2} - 2k_1 C_A C_B^2 - k_3 C_B + k_4 C_U \\ \frac{\partial C_U}{\partial t} &= \bar{D}\frac{\partial^2 C_U}{\partial r^2} + k_3 C_B - k_4 C_U \end{aligned}$$

Where C_A: concentration of species *A*, C_B: concentration of species *B*, C_U: concentration of species *U*, k_1: rate constant for the first reaction, k_2: rate constant for the second reaction, k_3: rate constant for the third reaction, k_4: rate constant for the fourth reaction, $\bar{D}$: diffusion coefficient, *r*: distance, *t*: time.

These relationships were derived under the assumptions that the diffusion coefficients for all the species are constant and equal, and that the molecular diffusion insider the one-dimensional reaction zone follows Fick's law. The equations describing the change in C_A, C_B and C_U can be redefined using dimensionless variables and parameters as such:

$$
\begin{aligned}
\frac{\partial y_A}{\partial \tau} &= D\frac{\partial^2 y_A}{\partial x^2} - y_A y_B^2 - \gamma y_A \\
\frac{\partial y_B}{\partial \tau} &= D\frac{\partial^2 y_B}{\partial x^2} - 2y_A y_B^2 - \delta y_B + \zeta y_U \\
\frac{\partial y_U}{\partial \tau} &= D\frac{\partial^2 y_U}{\partial x^2} + \delta y_B - \zeta y_U
\end{aligned}
$$

where the dimensionless variables and parameters represent the following:

$$
\begin{aligned}
&y_A = \frac{C_A}{C_{AF}}, \quad y_B = \frac{C_B}{C_{AF}}, \quad y_U = \frac{C_U}{C_{AF}}, \\
&\beta = \frac{C_{BF}}{C_{AF}}, \quad \tau = k_1 C_{AF}^2 t, \quad D = \frac{\bar{D}}{k_1 C_{AF}^2 L^2} \\
&x = \frac{r}{L}, \quad \gamma = \frac{k_2}{k_1 C_{AF}^2}, \quad \delta = \frac{k_3}{k_1 C_{AF}^2}, \quad \zeta = \frac{k_4}{k_1 C_{AF}^2}
\end{aligned}
$$

The following boundary conditions are also provided:

$$\frac{\partial y_A}{\partial x}(0,t) = -(1 - y_A), \ \frac{\partial y_B}{\partial x}(0,t) = -(\beta - y_B), \ y_U(0,t) = 0 \ \text{ at } x = 0, \text{ and}$$

$$\frac{\partial y_A}{\partial x}(1,t) = -\epsilon y_A, \ \frac{\partial y_B}{\partial x}(1,t) = -\eta {y_B}^2, \ \frac{\partial y_U}{\partial x}(1,t) = -\theta y_U \ \text{ at } x = 1.$$

Modeling Steady State

In the first part of the project, the steady-state characteristics of the system is examined for three different cases where constants δ, ζ, γ, ϵ, η, *and* θ *are varied in the following manner:*

a. $\delta = 0$, $\zeta = 0$, $D = 0.1$, $\beta = 1.5$, $\gamma = 0.05$, $\epsilon = 0.0$, $\eta = 0.0$, $\theta = 0.0$.

b. $\delta = 0.05$, $\zeta = 0.0$, $D = 0.1$, $\beta = 1.5$, $\gamma = 0.02$, $\epsilon = 0.1$, $\eta = 0.05$, $\theta = 0.1$.

c. $\delta = 0.05$, $\zeta = 0.03$, $D = 0.1$, $\beta = 1.5$, $\gamma = 0.02$, $\epsilon = 0.1$, $\eta = 0.05$, $\theta = 0.1$.

For steady state processes, the concentrations are not a function of time. Therefore, $\frac{dy_A}{d\tau} = \frac{dy_B}{d\tau} = \frac{dy_C}{d\tau} = 0$. With the time-dependent differentials equal to zero, the multivariable

Newton's Method can be used with the centered finite difference approximation method of y_A, y_B, and y_U. In order to apply the approximation, the x domain is discretized into nodes ranging from 1 to N corresponding to x = 0 to x =1. x varies from 0 to 1 because it is the dimensionless ratio between the distance along the membrane reactor over the max length of the reaction zone. The general equation for the approximations of the second-order differential is:

$$\frac{\partial^2 y_i}{\partial x^2} = \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2}$$

For y_A and y_B, the second-order approximation applies for i =1 to i = N since one fictitious node is added on both ends of the discretized domain to estimate the initial values at i = 1 and i =N with first order boundary conditions. This brings the number of variables to N+2. The general equation for the approximations of the first-order differential is:

$$\frac{\partial y_i}{\partial x} = \frac{y_{i+1} - y_{i-1}}{2\Delta x}$$

Therefore, the following is true for the boundary conditions of y_A and y_B:

$$\frac{dy_A}{dx}(0,t) = -(1 - y_A) \Rightarrow \frac{y_{A2} - y_{A0}}{2\Delta x} = -\left(1 - y_{A1}\right) \Rightarrow y_{A0} = 2\Delta x\left(1 - y_{A1}\right) + y_{A2}$$

$$\frac{dy_B}{dx}(0,t) = -(\beta - y_B) \Rightarrow \frac{y_{B2} - y_{B0}}{2\Delta x} = -\left(\beta - y_{B1}\right) \Rightarrow y_{B0} = 2\Delta x\left(\beta - y_{B1}\right) + y_{B2}$$

$$\frac{dy_A}{dx}(1,t) = -\varepsilon y_A \Rightarrow \frac{y_{A2} - y_{A0}}{2\Delta x} = -\varepsilon y_{A1} \Rightarrow y_{A0} = 2\Delta x\left(\varepsilon y_{A1}\right) + y_{A2}$$

$$\frac{dy_B}{dx}(1,t) = -\eta y^2{}_B \Rightarrow \frac{y_{B2} - y_{B0}}{2\Delta x} = -\eta y^2{}_{B1} \Rightarrow y_{B0} = 2\Delta x\left(\eta y^2{}_{B1}\right) + y_{B2}$$

For y_U, the boundary condition at x=0 is given as $y_U\left(0,t\right) = 0$. Therefore, only one fictitious node after x = N is needed to solve N+1 variables. The second-order approximation of y_U applies for i =2 to i = N since i =1 is a known boundary condition. The boundary conditions of y_U is as follows,

$$y_U(0,t) = 0 \Rightarrow y_{U1} = 0$$

$$\frac{dy_U}{dx}(1,t) = -\theta y_U \Rightarrow \frac{y_{U2} - y_{U0}}{2\Delta x} = -\theta y_{U1} \Rightarrow y_{U0} = 2\Delta x\left(\theta y_{U1}\right) + y_{U2}$$

With the ODE converted into algebraic equations, the Newton's Method can be used to iterate the solution at each node.

$$y^{m+1} = y^m - J^{-1}F$$

where J being the Jacobian of the system of equations, y^m signifies the concentration of the current iteration of Newton's Method, and F is the system of equations evaluated at y^m.

100 nodes where used to compute the finite difference method. For the MATLAB script will continue to iterate until all the values of F is less than 0.00001. If this condition is met, the y^{m+1} will be final result of the Newton Method iteration for each concentration with respect to x.

Modeling Transient State

The second part of the project analyzes the behavior of the diffusion reaction process through time. The Explicit Euler Method can be used in conjugation with the F presented in the steady-state introduction to model the changing concentration at different instances of time. The general formula for Explicit Euler Method:

$$y^{j+1} = y^j + \Delta t \cdot F^j$$

Where F^j is the approximation of the concentration profile at the time iteration j, Δt is the time step, y^j is the concentration profile at the time iteration j and y^{j+1} is the concentration profile at the next time iteration.

The initial conditions at t = 0 where given as the following:

$$y_A(x,0) = y_B(x,0) = y_U(x,0) = 0$$

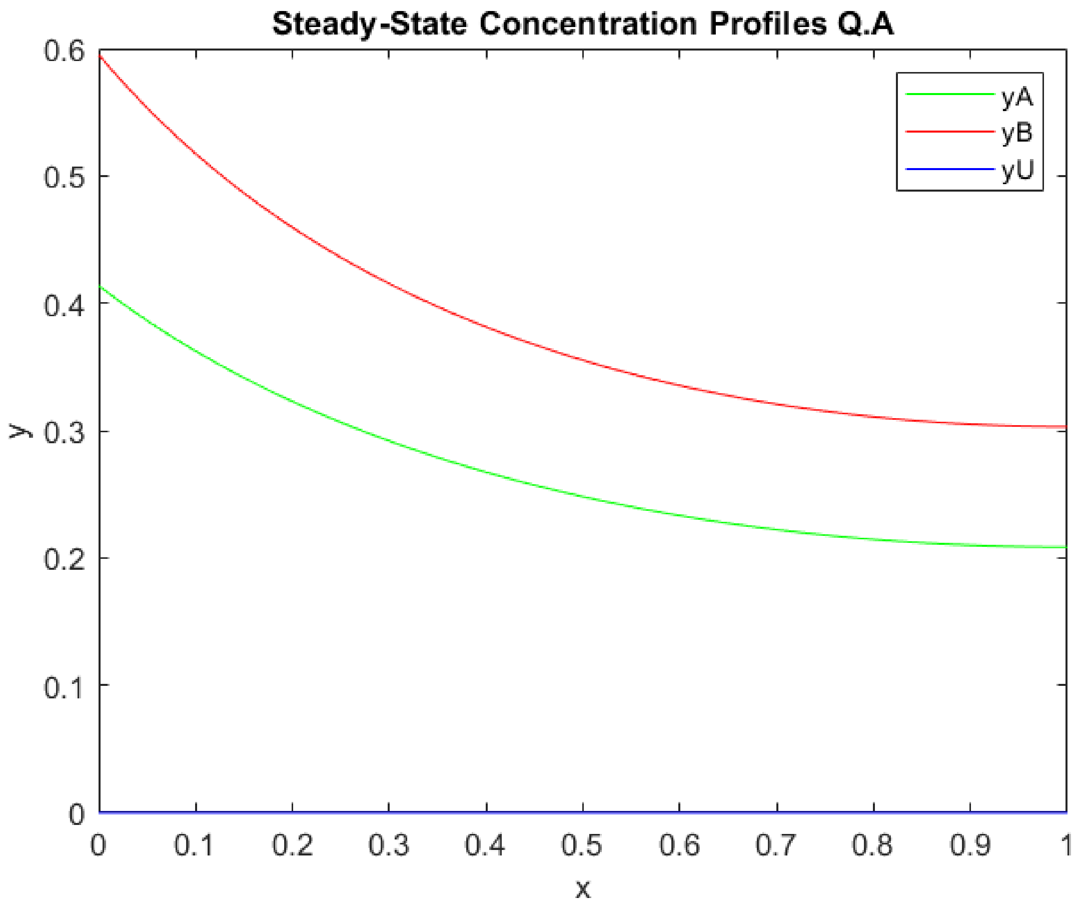
The constants modeled for the transient profile is the same as case c in the steady state profile. Therefore, the transient profile can be assumed to reach steady state when the difference between y^j is less than 1% of y^m. The MATLAB script will terminate the iteration to plot for the transient profile when the transient profile is deemed to have reached steady state. The code will generate a new concentration profile on the same graph every 20000 time steps.

Results and Discussion

Steady State Analysis

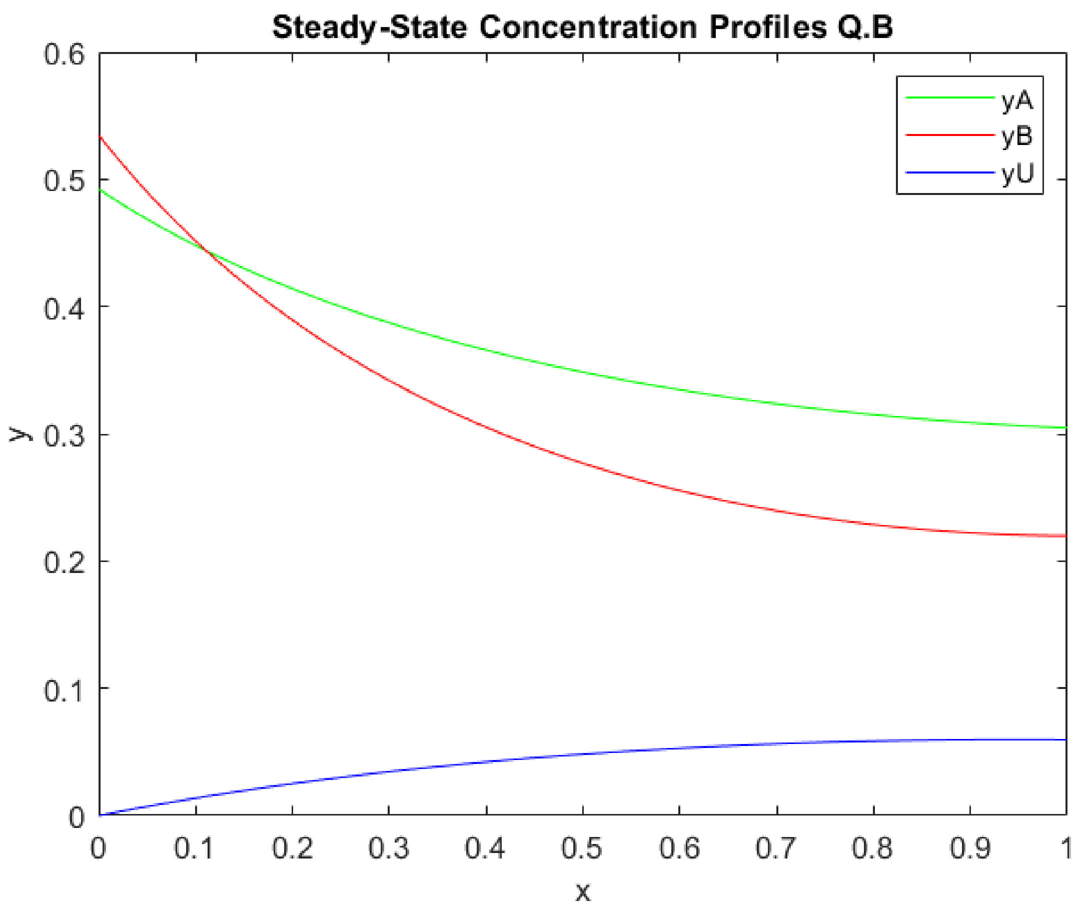
100 nodes were used to accurately calculate the behavior of concentration throughout the reaction zone to ensure a smooth accurate simulation. If more nodes were used, the simulation would have slowed down without much improvement in accuracy. When testing the script with 10 nodes, the simulation plotted visibly pointed line segments instead of a smooth curve. Not having enough nodes may change the approximation results significantly.

Case (a) with the constant values $\delta = 0$, $\zeta = 0$, $D = 0.1$, $\beta = 1.5$, $\gamma = 0.05$, $\epsilon = 0.0$, $\eta = 0.0$, $\theta = 0.0$ resulted in the following concentration profile:



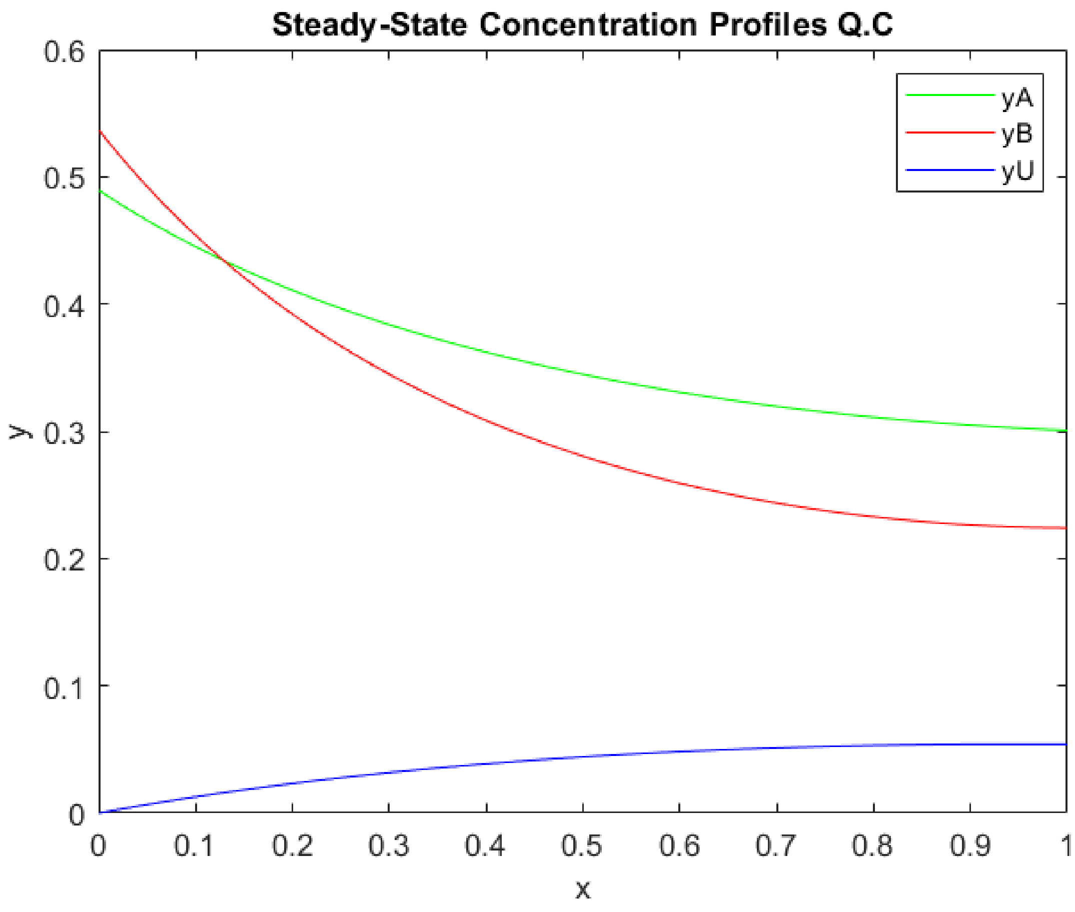
Since $\delta = 0$ and $\zeta = 0$, no concentration of y_U should be present in the reaction zone. The concentration profile confirms this expectation as the y_U remains zero for the entire x domain. This consistency between the simulation and theoretical results indicates that the simulation is accurate. Although y_U does not increase, y_A and y_B decreases in the reaction zone. This means that species A and B are consumed in the reaction to form D and P at steady-state.

Case (b) with the constant values $\delta = 0.05$, $\zeta = 0.0$, $D = 0.1$, $\beta = 1.5$, $\gamma = 0.02$, $\epsilon = 0.1$, $\eta = 0.05$, $\theta = 0.1$ resulted in the following concentration profile:



For case b, $\delta = 0.05$ but $\zeta = 0.0$. This theoretically means that species B is reacted to species U in the membrane reactor, but the reverse reaction does not occur. Therefore, y_B should decrease steeply to a lower concentration along the x domain while the y_U should increase compared the case a. The simulation results back up the assumptions. y_B decreased steeply below y_A while y_U increased from its initial boundary condition at x=0. As expected, the simulations show that y_A decreases slower compared to case a since constant γ decreased by 0.03. The changes in constants $\epsilon = 0.1$, $\eta = 0.05$, $\theta = 0.1$ changed the slope final concentration of y_A, y_B and y_U. Concentrations of all species is on the trend of increasing or decrease at x = L whereas in case a, the concentrations of all species should have plateaued near x = L. The simulations support this as the concentration is still changing near x = L for case b whereas for case a, the concentration is almost constant near x = L.

Case (c) with the constant values $\delta = 0.05$, $\zeta = 0.03$, $D = 0.1$, $\beta = 1.5$, $\gamma = 0.02$, $\epsilon = 0.1$, $\eta = 0.05$, $\theta = 0.1$ resulted in the following concentration profile:

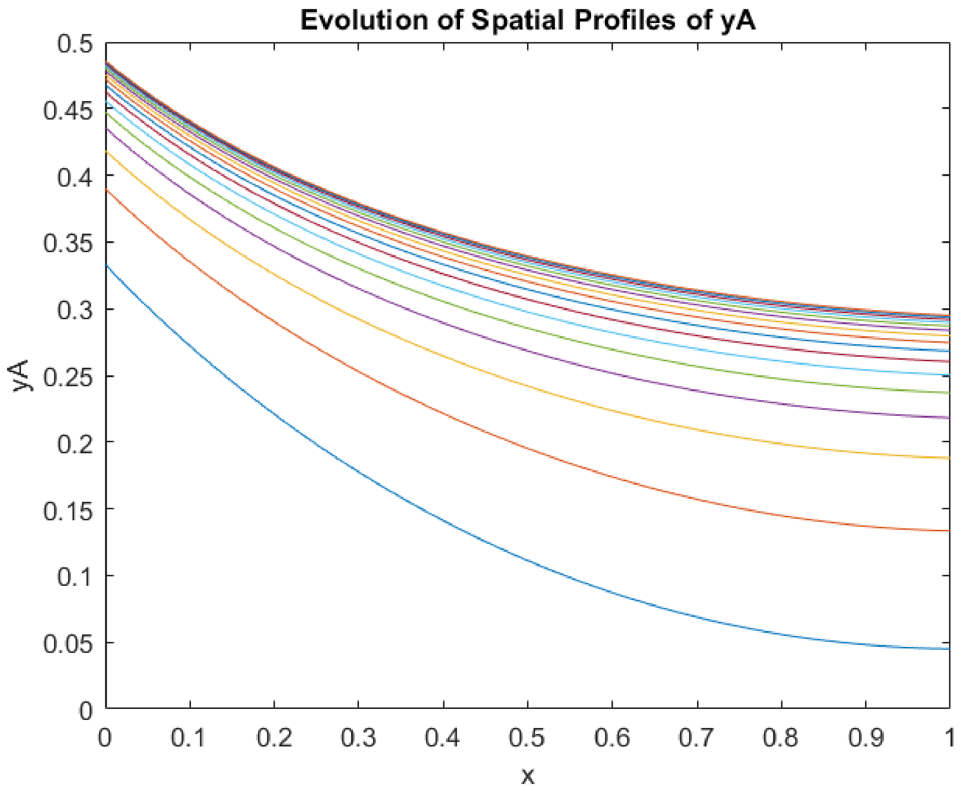


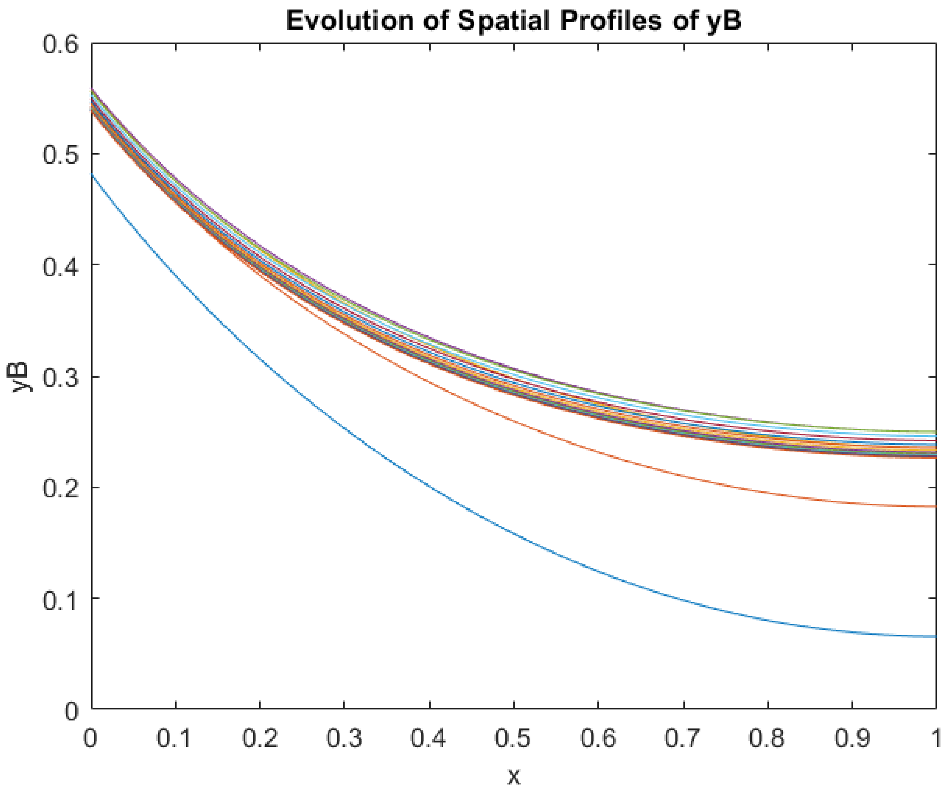
Case c is similar to case b except $\zeta = 0.03$. This indicates that the reverse reaction to convert U to B is present in the reaction mechanism. Therefore, y_B should decrease at a slower rate since it is replenished by the reverse reaction of species U. On the other hand, y_U should increase slower since some of it is converted to species B. As expected, case c simulations indicate y_B is higher at x = L as compared to case b while y_U is lower for case c than for case b.

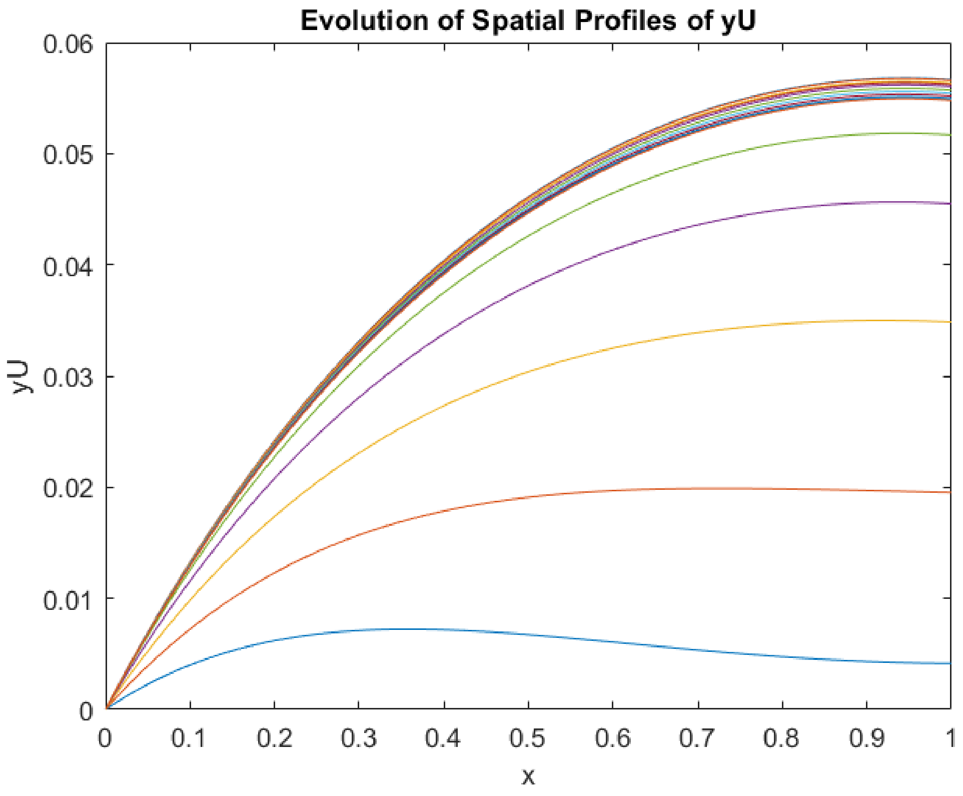
Since the simulations confirm the theoretical diffusion reaction predictions, the accuracy of the compute results is high.

Time-Dependent Behavior Analysis

Case (c) with the constant values $\delta = 0.05$, $\zeta = 0.03$, $D = 0.1$, $\beta = 1.5$, $\gamma = 0.02$, $\epsilon = 0.1$, $\eta = 0.05$, $\theta = 0.1$ resulted in the following concentration profiles throughout time,


Evolution of Spatial Profiles of yA
yA
x
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Evolution of Spatial Profiles of yB
yB
x
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



For a transient profile near t = 0, it is expected that the reaction is driven favorably towards none existing products in accordance to Le’Chatelier's principle. Due to the abundance in y_A and y_B near t = 0, the concentrations of y_A and y_B depleted quickly close to the initial rate of concentration change as specified in the boundary condition at x = 0. As time passed, the concentration of all species began to reach equilibrium at a slower pace with each time step as expected by Le’Chatelier's principle. Therefore, as the system approaches steady-state, the concentration profiles for each time step being to converge. The final time step plotted for the transient profile is near identical to the steady state profile simulated for case c indicating that the compute results are accurate. The behavior of the transient profile near t = 0 matches the behavior expected with equilibrium is greatly disturbed as represented by steep declines in abundant y_A and y_B.

Appendix

FinalProject.m contains all the code for the iteration schemes. The external scripts are used as functions to be called upon for the centered finite difference approximation method equations.

FinalProject.m

```
clear all; clc;

%Define Step Size
nodes = 100; h = 1/(nodes-1);
% Define Constants
d = 0.0; Z = 0.0; D = 0.1; B = 1.5; Y = 0.05; E = 0.0; H = 0.00; theta = 0.0;
% Define Errror
error = 1.0;

% Intialize matrixes for Jacobian and F
J_matrix = zeros(3*nodes-1, 3*nodes-1);
F_vector = zeros(3*nodes-1, 1);
% Define Y which will host all 3 concentrations in an array.
Y_current = zeros(3*nodes-1,1);
Y_next = zeros(3*nodes-1,1);
% Define X - distance from r = 0 to r = 1 discreted into 100 points
X = linspace(0,1,nodes);


%Iterate multivariable Newton's Method until it reaches appropriate error
% Y_M+1 = Y_M - J(Y_M)^-1 * F(Y_M)
while (error > 0.1)
    %Define Jacobian and F for each iteration
    %yA
    J_matrix(1,1) = -2*D*(1+h)/h^2 - Y_current(nodes+1,1)^2 - Y;
    J_matrix(1,2) = 2*D/h^2;
    J_matrix(nodes+1,1) = Y_current(nodes+1,1)^2;
    F_vector(1,1) =
A_first_function(Y_current(1,1),Y_current(2,1),Y_current(nodes+1,1));
    for i = 2:nodes-1
        J_matrix(i,i-1)= D/h^2;
        J_matrix(i,i) = -2*D/h^2 - Y_current(i+nodes,1)^2 - Y;
        J_matrix(i,i+1) = D/h^2;
        J_matrix(i+nodes, i) = -2*Y_current(i+nodes,1)^2;
        F_vector(i,1) =
A_middle_function(Y_current(i+1,1),Y_current(i,1),Y_current(i-
1,1),Y_current(i+nodes,1));
    end
    J_matrix(nodes, nodes-1) = 2*D/h^2;
    J_matrix(nodes, nodes) = -2*D*(1+E*h)/h^2 - Y_current(2*nodes,1)^2 - Y;
    J_matrix(2*nodes, nodes) = -2*Y_current(2*nodes,1)^2;
    F_vector(nodes, 1) = A_last_function(Y_current(nodes-
1,1),Y_current(nodes,1), Y_current(2*nodes,1));
    %yB
    J_matrix(1,nodes+1) = -2*Y_current(1,1)*Y_current(nodes+1,1);
    J_matrix(nodes+1, nodes+1) = -2*D*(1+h)/h^2 -
4*Y_current(1,1)*Y_current(nodes+1,1) - d;
```

```
    J_matrix(nodes+1,nodes+2) = 2*D/h^2;
    F_vector(nodes+1,1) =
B_first_function(Y_current(1,1),Y_current(nodes+1,1),Y_current(nodes+2,1),0);
    for i = (nodes+2):(2*nodes-1)
        J_matrix(i-nodes,i) = -2*Y_current(i-nodes,1)*Y_current(i,1);
        J_matrix(i,i-1) = D/h^2;
        J_matrix(i,i) = -2*D/h^2 - 4*Y_current(i-nodes,1)*Y_current(i,1) - d;
        J_matrix(i,i+1) = D/h^2;
        J_matrix(i+nodes-1,i) = d;
        F_vector(i,1) = B_middle_function(Y_current(i-
nodes,1),Y_current(i+1,1),Y_current(i,1),Y_current(i-
1,1),Y_current(i+nodes,1));
    end
    J_matrix(nodes,2*nodes) = 2*Y_current(nodes,1)*Y_current(2*nodes,1);
    J_matrix(2*nodes,2*nodes-1) = 2*D/h^2;
    J_matrix(2*nodes,2*nodes) = D*(-2-4*h*H*Y_current(2*nodes,1))/h^2 -
4*Y_current(nodes,1)*Y_current(2*nodes) - d;
    J_matrix(3*nodes-1,2*nodes) = d;
    F_vector(2*nodes,1) =
B_last_function(Y_current(nodes,1),Y_current(2*nodes-
1,1),Y_current(2*nodes,1), Y_current(3*nodes-1,1));
    %yU
    J_matrix(nodes+2,2*nodes+1) = Z;
    J_matrix(2*nodes+1,2*nodes+1) = -2*D/h^2 - Z;
    J_matrix(2*nodes+1,2*nodes+2) = D/h^2;
    F_vector(2*nodes+1,1) =
U_middle_function(Y_current(nodes+1,1),Y_current(2*nodes+2,1),Y_current(2*nod
es+1,1),0);
    for i = (2*nodes+2):(3*nodes-2)
        J_matrix(i-nodes+1,i) = Z;
        J_matrix(i,i-1) = D/h^2;
        J_matrix(i,i) = -2*D/h^2 - Z;
        J_matrix(i,i+1) = D/h^2;
        F_vector(i,1) = U_middle_function(Y_current(i-
nodes,1),Y_current(i+1,1),Y_current(i,1),Y_current(i-1,1));
    end
    J_matrix(2*nodes,3*nodes-1) = Z;
    J_matrix(3*nodes-1,3*nodes-2) = 2*D/h^2;
    J_matrix(3*nodes-1,3*nodes-1)= -2*D*(1+h*theta)/h^2 - Z;
    F_vector(3*nodes-1,1) =
U_last_function(Y_current(2*nodes,1),Y_current(3*nodes-
2,1),Y_current(3*nodes-1,1));
    %Calculate new ym1
    Y_next = Y_current - inv(J_matrix)*F_vector;
    %Determine if we should keep iterating
    error = 0;
    for j = 1:3*nodes-1
       if F_vector(j,1)^2 > 0.0001
           error = 1;
       end
    end
    %Update values
    Y_current = Y_next;
end
%Create matrices to hold concentration values for each species
yA(:,1) = Y_current(1:nodes,1);
yB(:,1) = Y_current(nodes+1:2*nodes,1);
```

```
yU(2:nodes,1) = Y_current(2*nodes+1:3*nodes-1,1);
yU(1,1) = 0;
%Plot Graphs
figure(1);
plot(X, yA, 'g');
hold on
plot(X, yB, 'r');
plot(X, yU, 'b');
xlabel('x');
ylabel('y');
title('Steady-State Concentration Profiles Q.A');
legend('yA', 'yB', 'yU');

% Transient State with Explict Euler Method
delta_t = 0.0001;

% Define Y which will host all 3 concentrations in an array.
Yt_current = zeros(3*nodes-1,1);
Yt_next = zeros(3*nodes-1,1);

err = 1;
count = 0;

while err > 0.01
    %Use Explict Euler to determine new y_transient values
    for i = 1:3*nodes-1
       if i == 1
           Yt_next(1,1) = Yt_current(1,1) +
delta_t*A_first_function(Yt_current(1,1),Yt_current(2,1),Yt_current(nodes+1,1
));
       elseif i < nodes
           Yt_next(i,1) = Yt_current(i,1) +
delta_t*A_middle_function(Yt_current(i+1,1),Yt_current(i,1),Yt_current(i-
1,1),Yt_current(i+nodes,1));
       elseif i == nodes
           Yt_next(i,1) = Yt_current(i,1) +
delta_t*A_last_function(Yt_current(i-1,1),Yt_current(i,1),Yt_current(2*i,1));
       elseif i == nodes+1
           Yt_next(i,1) = Yt_current(i,1) +
delta_t*B_first_function(Yt_current(1,1),Yt_current(i,1),Yt_current(i+1,1),0)
;
       elseif i < 2*nodes
           Yt_next(i,1) = Yt_current(i,1) +
delta_t*B_middle_function(Yt_current(i-
nodes,1),Yt_current(i+1,1),Yt_current(i,1),Yt_current(i-
1,1),Yt_current(i+nodes,1));
       elseif i == 2*nodes
           Yt_next(i,1) = Yt_current(i,1) +
delta_t*B_last_function(Yt_current(nodes,1),Yt_current(i-
1,1),Yt_current(i,1), Yt_current(3*nodes-1,1));
       elseif i == 2*nodes+1
           Yt_next(i,1) = Yt_current(i,1) +
delta_t*U_middle_function(Yt_current(i-
nodes,1),Yt_current(i+1,1),Yt_current(i,1),0);
       elseif i < 3*nodes-1
```

```
            Yt_next(i,1) = Yt_current(i,1) +
delta_t*U_middle_function(Yt_current(i-
nodes,1),Yt_current(i+1,1),Yt_current(i,1),Yt_current(i-1,1));
        elseif i == 3*nodes-1
            Yt_next(i,1) = Yt_current(i,1) +
delta_t*U_last_function(Yt_current(2*nodes,1),Yt_current(i-
1,1),Yt_current(i,1));
        end
    end
    %Determine error for this iteration
    err = norm(abs(Yt_next-Y_current))/norm(Y_current);
    %Plot curves
    count = count + 1;
    % Plot every 20000 iterations
    if mod(count,20000) == 0
        yAt(:,1) = Yt_next(1:nodes,1);
        yBt(:,1) = Yt_next(nodes+1:2*nodes,1);
        yUt(1,1) = 0;
        yUt(2:nodes,1) = Yt_next(2*nodes+1:3*nodes-1,1);

        figure(2);
        plot(X,yAt);
        hold on;
        figure(3);
        plot(X,yBt);
        hold on;
        figure(4);
        yUt(1,1) = 0;
        plot(X,yUt);
        hold on;
    end
    Yt_current = Yt_next;
end
t = delta_t*count;

figure(2);
xlabel('x');
ylabel('yA');
title('Evolution of Spatial Profiles of yA');

figure(3);
xlabel('x');
ylabel('yB');
title('Evolution of Spatial Profiles of yB');

figure(4);
xlabel('x');
ylabel('yU');
title('Evolution of Spatial Profiles of yU');

fprintf('Time elapsed to convergence: %f \n', t);
```

A_first_function.m

```
function iteration = A_first_function(yA1, yA2, yB1)
%Define Step Size
```

```
nodes = 100; h = 1/(nodes-1);
% Define Constants
d = 0.0; Z = 0.0; D = 0.1; B = 1.5; Y = 0.05; E = 0.0; H = 0.00; theta = 0.0;

iteration = D*(2*yA2 - 2*yA1*(1+h) + 2*h)/h^2 - yA1*(yB1)^2 - Y*yA1;

end
```

A_middle_function.m

```
function iteration = A_middle_function(yA1, yA2, yA3, yB2)
%Define Step Size
nodes = 100; h = 1/(nodes-1);
% Define Constants
d = 0.0; Z = 0.0; D = 0.1; B = 1.5; Y = 0.05; E = 0.0; H = 0.00; theta = 0.0;

iteration = D*(yA1 - 2*yA2 +yA3)/h^2 - yA2*(yB2)^2 - Y*yA2;

end
```

A_last_function.m

```
function iteration = A_last_function(yAN1, yAN, yBN)
%Define Step Size
nodes = 100; h = 1/(nodes-1);
% Define Constants
d = 0.0; Z = 0.0; D = 0.1; B = 1.5; Y = 0.05; E = 0.0; H = 0.00; theta = 0.0;

iteration = D*(2*yAN1 - 2*yAN*(1+h*E))/h^2 - yAN*(yBN)^2 - Y*yAN;

end
```

B_first_function.m

```
function iteration = B_first_function(yA1, yB1, yB2, yU1)
%Define Step Size
nodes = 100; h = 1/(nodes-1);
% Define Constants
d = 0.0; Z = 0.0; D = 0.1; B = 1.5; Y = 0.05; E = 0.0; H = 0.00; theta = 0.0;

iteration = D*(2*yB2 - 2*yB1*(1+h) +2*h*B)/h^2 - 2*yA1*(yB1)^2 - d*yB1 +
Z*yU1;

end
```

B_middle_function.m

```
function iteration = B_middle_function(yA2, yB1, yB2, yB3, yU2)
%Define Step Size
nodes = 100; h = 1/(nodes-1);
% Define Constants
d = 0.0; Z = 0.0; D = 0.1; B = 1.5; Y = 0.05; E = 0.0; H = 0.00; theta = 0.0;

iteration = D*(yB1 - 2*yB2 + yB3)/h^2 - 2*yA2*(yB2)^2 - d*yB2 + Z*yU2;
```

```
end
```

B_last_function.m

```
function iteration = B_last_function(yAN, yBN1, yBN, yUN)
%Define Step Size
nodes = 100; h = 1/(nodes-1);
% Define Constants
d = 0.0; Z = 0.0; D = 0.1; B = 1.5; Y = 0.05; E = 0.0; H = 0.00; theta = 0.0;

iteration = D*(2*yBN1 - 2*yBN - 2*h*H*(yBN^2))/h^2 - 2*yAN*(yBN)^2 - d*yBN +
Z*yUN;

end
```

U_middle_function.m

```
function iteration = U_middle_function(yB2, yU1, yU2, yU3)
%Define Step Size
nodes = 100; h = 1/(nodes-1);
% Define Constants
d = 0.0; Z = 0.0; D = 0.1; B = 1.5; Y = 0.05; E = 0.0; H = 0.00; theta = 0.0;

iteration = D*(yU1 - 2*yU2 +yU3)/h^2 + d*yB2 - Z*yU2;

end
```

U_last_function.m

```
function iteration = U_last_function(yBN, yUN1, yUN)
%Define Step Size
nodes = 100; h = 1/(nodes-1);
% Define Constants
d = 0.0; Z = 0.0; D = 0.1; B = 1.5; Y = 0.05; E = 0.0; H = 0.00; theta = 0.0;

iteration = D*(2*yUN1 - 2*yUN*(1+h*theta))/h^2 + d*yBN - Z*yUN;

end
```