# Steady-state and transient analysis of a diffusion-reaction process

# Table of Contents

# 1. Introduction

In chemistry, a reaction is a transformation from a set of chemical species into another. One type of reaction is known as a diffusion reaction. This type of reaction has dependence on time, like most reactions, with the addition of spatial dependence. These reactions can generally be described mathematically by parabolic differential equations. For this analysis, a set of reactions were to have their steady and transient characteristics studied through their respective set of parabolic differential equations in order to determine potential yields of reaction products.

Four reactions were observed taking place between the species A, B, D, U, and P. Species A and B were the reactants added to the reactor, D and P were side products of the reactions, and U was the desired product. The reactions are given as:

$$A+2B\xrightarrow{k_1}D\ , \qquad A\xrightarrow{k_2}P\ , \qquad B\xrightarrow{k_3}U\ , \qquad U\xrightarrow{k_4}B\ ;$$

Each reaction has it's own reaction rate constant defined by a "k" variable. These reactions were to take place in a one dimensional reactor. The reaction zone extends from r = 0 to r = L, where 0 is the beginning of the reactor and L is a distance away at the end of the reactor. To the left of r = 0 is a film through which species A and B are deposited. The rate at which the two are deposited from the film to the reactor is different for both species. However, the concentration of each species is kept at a fixed constant of $C_{AF}$ and $C_{BF}$ at the point where they enter the film. The various concentrations of species A, B, and U ($C_A$, $C_B$, $C_U$) vary according to position and time within the reactor. The change in concentrations, obtained from mass balances of each species, are described by the parabolic partial differential equations (PDE):

$$\frac{\partial C_A}{\partial t}=\overline{D}\frac{\partial^2 C_A}{\partial r^2}-k_1 C_A C_B^2-k_2 C_A$$

$$\frac{\partial C_B}{\partial t}=\overline{D}\frac{\partial^2 C_B}{\partial r^2}-2k_1 C_A C_B^2-k_3 C_B+k_4 C_U$$

$$\frac{\partial C_U}{\partial t}=\overline{D}\frac{\partial^2 C_U}{\partial r^2}+k_3 C_B-k_4 C_U$$

All species were assumed to have the same diffusion coefficient $\overline{D}$. Non-dimensionalization was carried out on these three PDE's using the dimensionless variables:

$$y_A=\frac{C_A}{C_{AF}},\ y_B=\frac{C_B}{C_{AF}},\ y_U=\frac{C_U}{C_{AF}},$$

$$\beta=\frac{C_{BF}}{C_{AF}},\ \tau=k_1 C_{AF}^2 t\ ,\ D=\frac{\overline{D}}{k_1 C_{AF}^2 L^2},$$

$$x=\frac{r}{L},\ \gamma=\frac{k_2}{k_1 C_{AF}^2},\ \delta=\frac{k_3}{k_1 C_{AF}^2},\ \zeta=\frac{k_4}{k_1 C_{AF}^2},$$

These variables substituted into the first PDE yield:

$$\frac{\partial C_A}{\partial t} = \overline{D}\frac{\partial^2 C_A}{\partial r^2} - k_1 C_A C_B^2 - k_2 C_A$$

$$\Rightarrow \frac{\partial y_A}{\partial \tau}\left(k_1 C_{AF}^3\right) = \frac{\left(k_1 C_{AF}^3\right)L^2}{L^2} D\frac{\partial^2 y_A}{\partial x^2} - \left(k_1 C_{AF}^3\right) y_A y_B^2 - \left(k_1 C_{AF}^3\right)\gamma\, y_A$$

$$\Rightarrow \frac{\partial y_A}{\partial \tau} = D\frac{\partial^2 y_A}{\partial x^2} - y_A y_B^2 - \gamma y_A$$

The same procedures are followed to obtain the non-dimensionalized system of PDE's:

$$\frac{\partial y_A}{\partial \tau} = D\frac{\partial^2 y_A}{\partial x^2} - y_A y_B^2 - \gamma y_A$$

$$\frac{\partial y_B}{\partial \tau} = D\frac{\partial^2 y_B}{\partial x^2} - 2 y_A y_B^2 - \delta\, y_B + \zeta\, y_U$$

$$\frac{\partial y_U}{\partial \tau} = D\frac{\partial^2 y_U}{\partial x^2} + \delta\, y_B - \zeta\, y_U$$

The new boundary conditions for this system, where, ε, η, and θ were constants describing how quickly species were removed from the reactor, became:

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

$$\frac{\partial y_A}{\partial x}(L,t) = -\varepsilon\, y_A; \frac{\partial y_B}{\partial x}(L,t) = -\eta\, y_B; \frac{\partial y_U}{\partial x}(L,t) = -\theta\, y_U$$

The first portion of this analysis was to obtain the profiles of the reaction at steady-state. At steady-state, the $\frac{\partial y}{\partial \tau}$ terms in the parabolic PDE's become 0. This leaves a system of 3 second-order differential equations. Each of the equations was discretized over the spatial dimension followed by applying newton's method. Central finite discretization was used as the discretization scheme, while multivariate newton's method was used to set up a linear system. The linear system was then solved using an LUsolver function that uses LU decomposition. Since the linear system always had a banded form, LU decomposition seemed appropriate to solve the system. This was done for three separate cases(A, B, and C). For each case, the constants were different and defined respectively by:

Case A) $\delta = 0, \zeta = 0, D = 0.1, \beta = 1.5, \gamma = 0.05, \varepsilon = 0.0, \eta = 0.0, \theta = 0.0$

Case B) $\delta = 0.05, \zeta = 0, D = 0.1, \beta = 1.5, \gamma = 0.02, \varepsilon = 0.1, \eta = 0.05, \theta = 0.1$

Case C) $\delta = 0.05, \zeta = 0.03, D = 0.1, \beta = 1.5, \gamma = 0.02, \varepsilon = 0.1, \eta = 0.05, \theta = 0.1$

The second problem was solving for the profiles with respect to time. Again, central finite discretization was applied to the equations. However, unlike the previous problem, the $\partial y \div \partial \tau$ terms were not zero. Discretization led to a system of 3xN first-order ODE's in time. These ODE's were integrated over their dimensionless time using explicit Euler method. Integration was ceased when they reached 99% of the steady-state values obtained from the first portion of the analysis.

# 2. Steady-State Analysis

## Nodal Rationale

Centered finite difference approximations were used to solve the PDE's of both parts of this analysis. As a result, a number of nodal points along the reactor had to be used to solve the equations. One of the problems associated with picking the number of nodes is that having too few or too many can increase the errors of the obtained approximations. For this, an error of 1.5e-7 was deemed as a fair error incurred by each use of centered finite differentiation. The method makes use of higher order derivate to approximate the values of a function. It is expressed by:

$$\frac{d^2 y}{dx^2} = \frac{y_{i+1} - 2\,y_i + y_{i-1}}{\Delta x^2} + E(f, \Delta x)$$

Where E is the errors that arise from both the rounding errors of each approximation as well as the truncation errors. It follows that the errors can be reduced to and described as:

$$E(f, \Delta x) = \frac{e - 2e + e}{\Delta x^2} - \frac{2\Delta x^2}{4!}\frac{d^4 y}{dx^4}$$

An assumption made here was that the rounding error for each node was approximately the same value, e. Another assumption was made that the absolute value of the fourth order derivative of y was going to be less than 1 (Note: This assumption was made on the basis that since the concentrations were made dimensionless by dividing by the initial concentration of species A outside the film they were essential a ratio of final and initial concentrations. Therefore, it could not be greater than one. Later realizations made it apparent that this can only really hold true for species A and not B since their deposition rates, $C_{AF}$, and $C_{BF}$ values were different.) With the assumption that the fourth order term is less than or equal to one, the absolute value of the error becomes:

$$\left|E(f, \Delta x)\right| = \frac{4\,e}{\Delta x^2} + \frac{\Delta x^2}{12}$$

To find the minimum value of this function, i.e. the minimum overall error, the derivative must be taken with respect to the step size and set to zero. The step size can then be solved for:

$$0=\frac{-8e}{\Delta x^3}+\frac{\Delta x}{6}\Rightarrow \Delta x^3=48\,e\Rightarrow \Delta x=\sqrt[4]{48\,e}=\sqrt[4]{48\,(1.5\text{e-}7)}=0.0518$$

.With the step size found, the number of appropriate nodes can be found.

$$\frac{1-0}{N-1}=0.0518\Rightarrow 1+0.0518=0.0518\,N\Rightarrow N=20.30$$

The optimal number of nodes was chosen to be 20 as it would have the lowest overall error contributed by the centered finite difference approximations with each node contributing a local error of 1.5e-7.

## Computational Accuracy

The accuracy of the results depended on several factors. The main contributing factors were the methods employed to compute the solutions. There were three different methods for the first portion that had an impact on the accuracy of the results. They were central finite difference approximation, multivariate newton's, and LU decomposition. The accuracy of the difference approximations were discussed briefly in the previous section. Newton's method, however, has no easy way to judge what the actual error is(short of having the actual root that is being found). The error is highly dependent on when you have stopped the repeated iterations. So for the analysis, the difference threshold was chosen to be the same as the error expected from each node. This would ensure that the error would be kept low, but not so low that it had a significant impact on computation time.

Lastly, LU decomposition was used to solve the linear algebraic system that was obtained from the implementation of newton's method. This method contributed its own errors, but were neglected in a fashion similar to those of newton's method as they were more difficult to be gauged. However, of more concern was likely the cost of finding the solution to the system which required a number of arithmetic operation on the order of $O(2N^3/3)$, where N is the size of the A matrix containing the differential equations . For the system of three ODE's with 20 nodes each, the Jacobian had a size of 60. The errors incurred from this step are most likely the highest. Other methods such as Jacobi or Gauss-Seidel would also likely work, while resulting in fewer operations. Since the analysis was not based on obtaining the solution with the fewest number of operations, LU decomposition was chosen in favor of the latter two for the sake of simplicity.

Centered finite difference approximations were the biggest controllable contributors to the errors associated with the solutions found. The program was also tested with different methods to compute the solution to the algebraic system. The results were the same; helping confirm the idea that the LU decomposition had a negligible impact on the errors. With newtons having no quantifiable way of finding the actual error without knowing the root, its errors were deemed negligible as well, on the assumption that a sufficient guess and termination criteria would generate approximately the same result. The global error incurred by the central difference was on the order of $\Delta x^2$. This put it on the order of 0.0025 for the 20 nodes used. So the errors in the solution were accurate to about 0.0001 give or take 0.0025.

## Profile Analysis

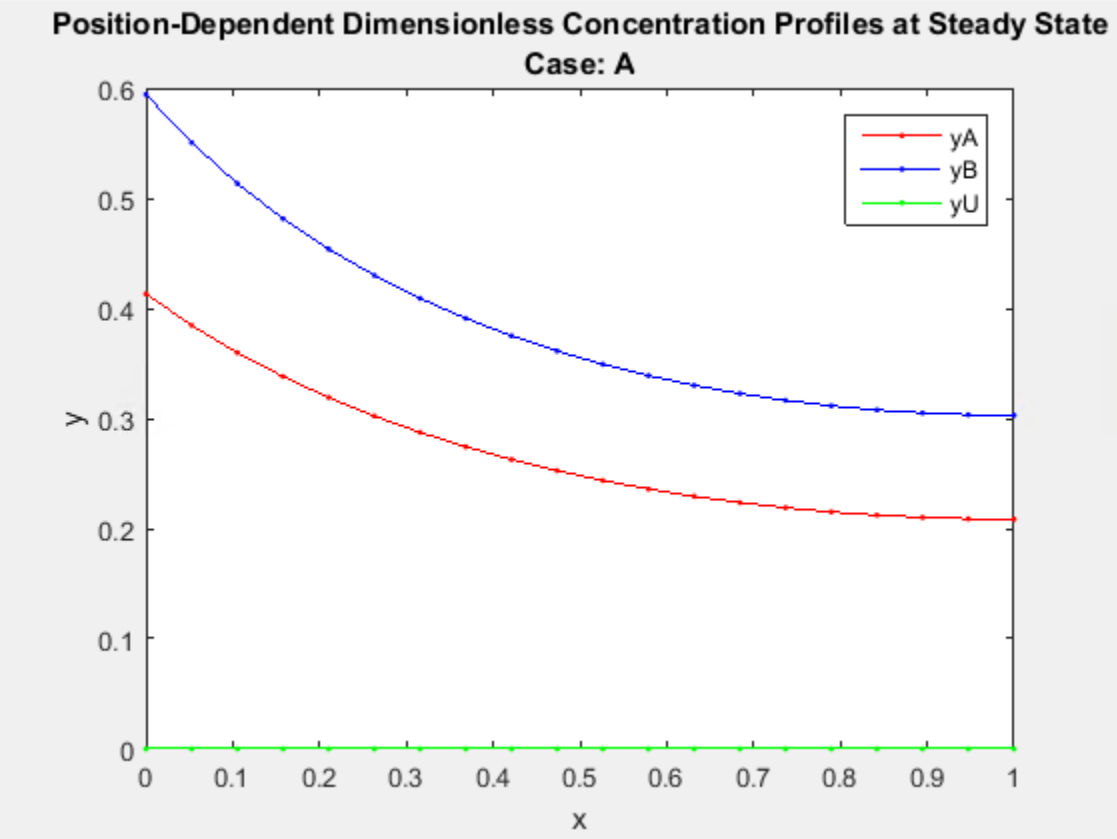


*Figure 1: steady state profiles Case A - (values listed in Appendix B)*

The program DesignProjectP1.m was used to obtain the profiles at steady state. For case A, there were a number of the constants that were zero. Two of them effected the equations, while three effected the boundary conditions. The first two constants, $\delta$ and $\zeta$, being zero essentially signified that the reaction constants $k_3$ and $k_4$ were zero. Due to the being zero, no reaction from B to U or U to B was taking place. It would be expected for U to not be produced. Figure 1 shows that there was no U present in any part of the reactor, which coincides with the aforementioned expectation. The boundary conditions also become 0 for both species A and B at x = 1. This implies that they will either reach a maximum or minimum at this location. Inspection of the figure reveals that both begin to plateau and do reach a minimum value at this point. The two reactions that are left both consume species A and B. The first reaction consumes 2 moles of B for every mole of A. This would lead someone to believe that species B would decrease faster. However, the second reaction consumes an additional mole of A. The main factor between the consumption of the two depends on the rate constants of each reaction. The graph suggests that the two are consumed equally by the combined reactions throughout most of the reaction zone. The exception being x = 0 where species B is consumed slightly faster than A.

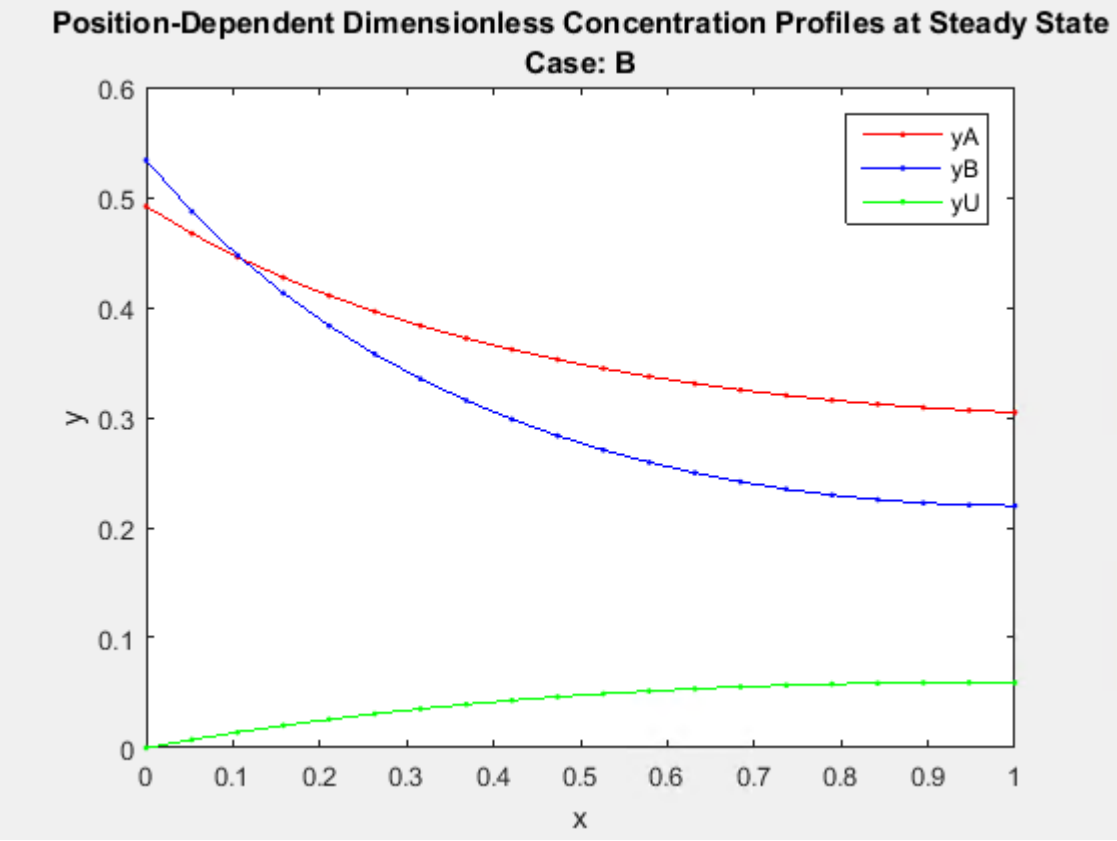


*Figure 2: steady state profiles Case B - (values listed in Appendix B)*

Case B, depicted by figure 2, had non-zero values for all constants except for $\zeta$. This constant corresponded to the $k_4$ rate constant. With it being zero, there was no reaction taking place that converted U back into B. However, unlike the previous case, $\delta$ was nonzero so the rate constant $k_3$ was also nonzero. This indicates that there must have been conversion of species B to species U. The graph agrees with this assessment as the profile of species U increases toward the end of the reactor. The profile of species B also significantly decreases much more quickly and to a lower level than the previous case, further proving that species B is now being converted to species U. Another significant change from the previous case are the nonzero boundary conditions at x = 1. This implies that the species are being removed at the end of the reactor. Although the profiles seem to be reaching a minimum, the boundary conditions are not zero at x = 1, so they are not guaranteed to be a local minima or maxima as in the previous case.

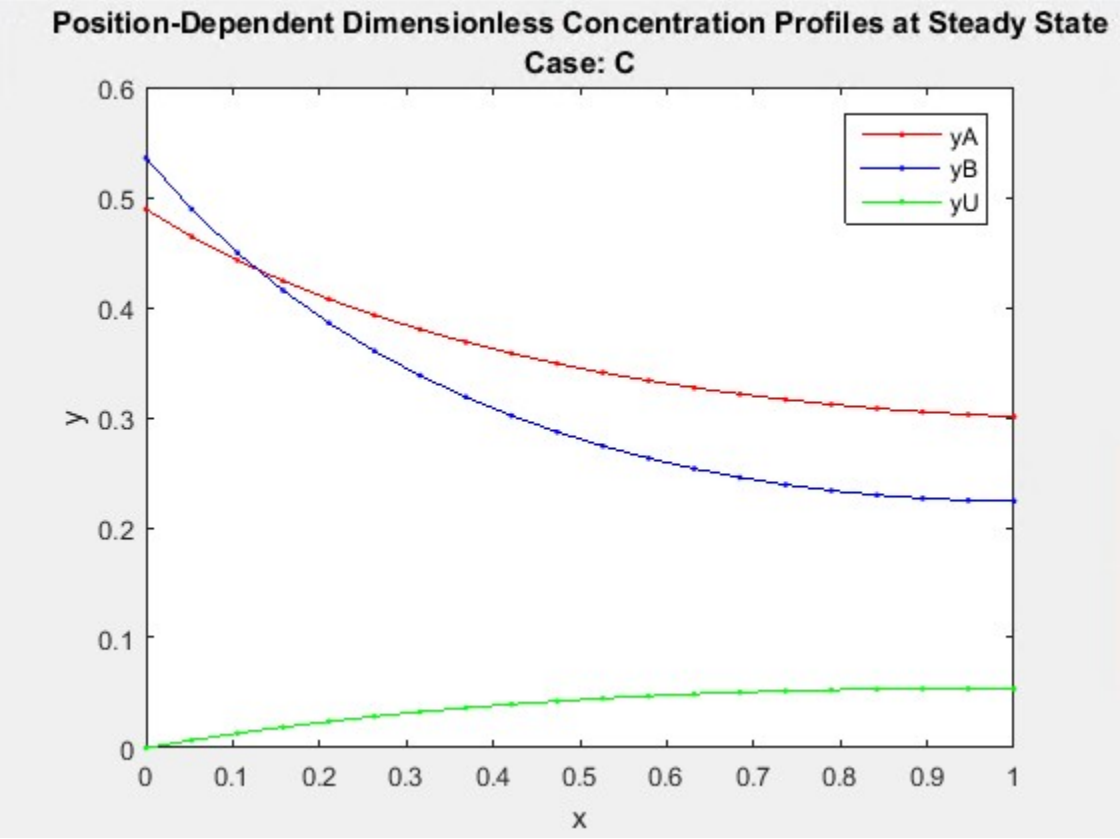


*Figure 3: steady state profiles Case C - (values listed in Appendix B)*

The final case, case C, was almost exactly the same as case B. The only difference was the nonzero value of the $\zeta$ variable. The fourth reaction, which converted species U to species B, was now occurring within the reactor. Comparison of the two graphs of case B and case C seems as though the two are exactly the same. Despite this fact, the numerical values show a slight difference from case B in the species profiles within the reactor. A quick look at Appendix B, which contains said values, reveals that the profiles for case B and C at x = 1 for $y_A$, $y_B$, and $y_U$ are 0.3046, 0.2199, 0.0591 and 0.3008, 0.2241, and 0.0536, respectively. For case C, the profiles were all lower numerically than case B. This was likely caused by species U being converted back into species B, which then participated in the first reaction. Since the first reaction requires species A, it would explain why the profile of species A decreases as well. A larger rate constant of reaction four, and by extension variable $\zeta$, may have caused the yield of desired product U to decrease and would have a heavy impact on the productivity of the reactor.

# 3. Transient-State Analysis

## Numerical Integration

Integration in time for this system involved applying centered finite differences to the system of PDE's. This yielded a system of nodal ODE's in time that could be integrated with respect to time. Integration over time allows the approximate values of the nodes to be found at a certain time. The method employed here was Explicit-Euler method. Similar to LU decomposition, this method was chosen due to its simple scheme. Although there are other methods that could yield more accurate solutions such as Implicit-Euler, Euler Predictor-Corrector, and Runge-Kutta, the amount of accuracy that they would contribute seemed to be outweighed by the amount of extra time that would be required to modify the methods to work for this model. Since the ultimate goal was analyzing the reactor species profile evolutions, Explicit-Euler was essentially chosen for its moderate accuracy and easy implementation.

Explicit-Euler is not without its faults. Like other numerical integration methods, there is a certain amount of error that is incurred, which effects the accuracy of the computed solution. Each time step contributes to the overall error of the method. This error is known as the local error. For the intents of this analysis, the focus was placed instead in the global error of the method. Explicit-Euler has a global error on the order of $h^2$, where h is the size of the time step. Integration was to be carried out until values were found that were 99% of the values obtained during steady state analysis. Essentially, integration was to be carried out until steady-state was reached. For the sake of consistency, an error close to the error of the steady state values, approximately 0.0025, was desired. The time step was chosen to be 0.01 as it put the error from Explicit-Euler on the order of 1e-4. As a result, it would not be too far off from the error that occurred in finding the approximate steady-state values from the previous section.

## Transitional Relationship

As numerical integration was carried out with DesignProjectP2.m, the node values of each species were plotted on their own respective graph (see figures 4, 5, 6). The resulting graphs show a rough depiction of how the profiles change over time. Since the reactants are deposited into the reactor at $x = 0$, it should come as no surprise that that is the location where all three species begin to increase as time progresses forward. The graphs can be viewed as contour maps for the concentration profiles. The profiles approach a "limit" over time. The "limit" being the steady-state profiles. As time goes on the profiles form those seen in the previous section for Case C. After that time has been reach the profiles remained approximately the same. There may be slight changes and fluctuations, but they will keep approaching steady state to the point that one time frame is indistinguishable from the next time frame. This point was found to be approximately $\tau = 30.35$, as shown in the figures below. At this time, the profiles become virtually identical to those obtained in the previous section.

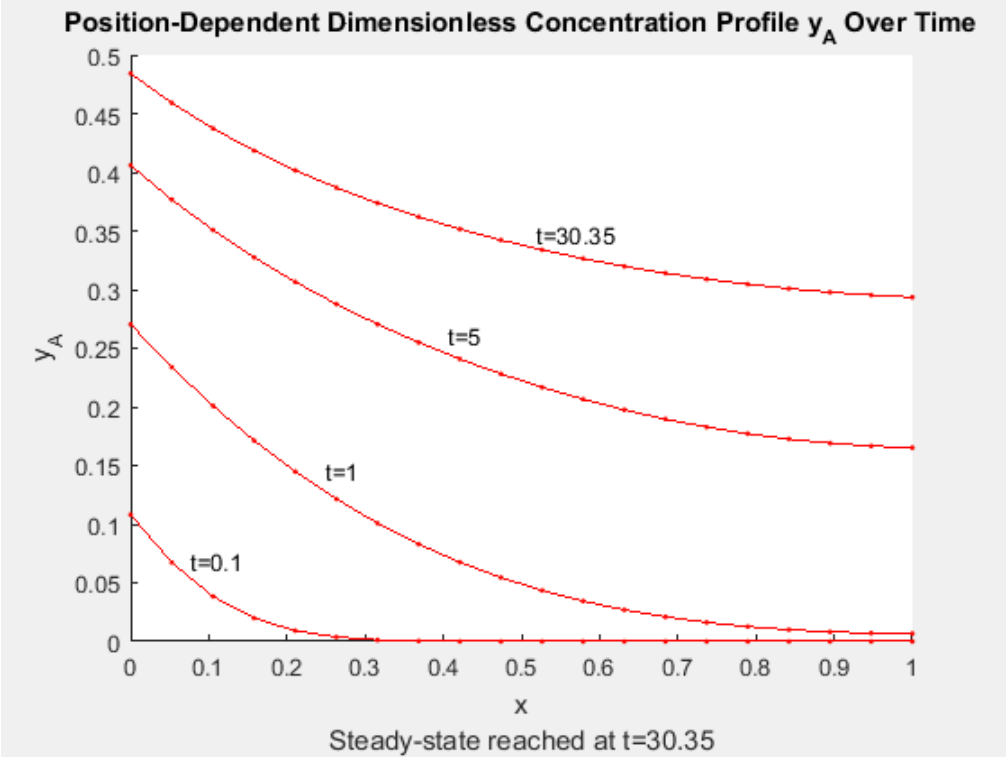


*Figure 4: profile $y_A$ over time*

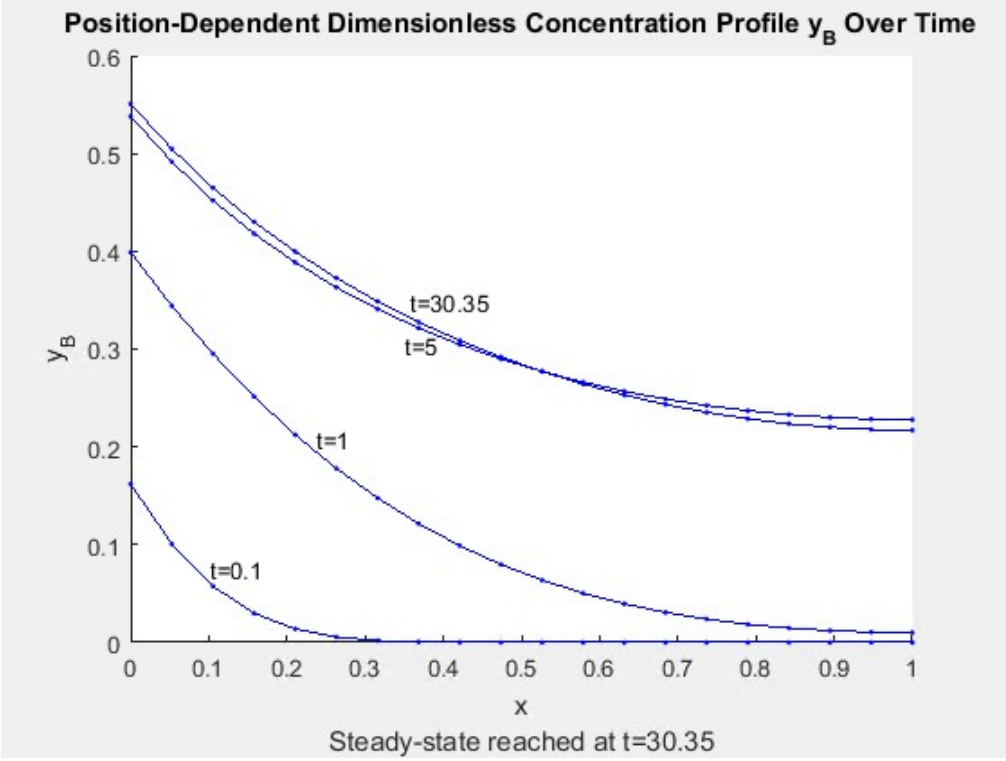


*Figure 5: profile $y_B$ over time*

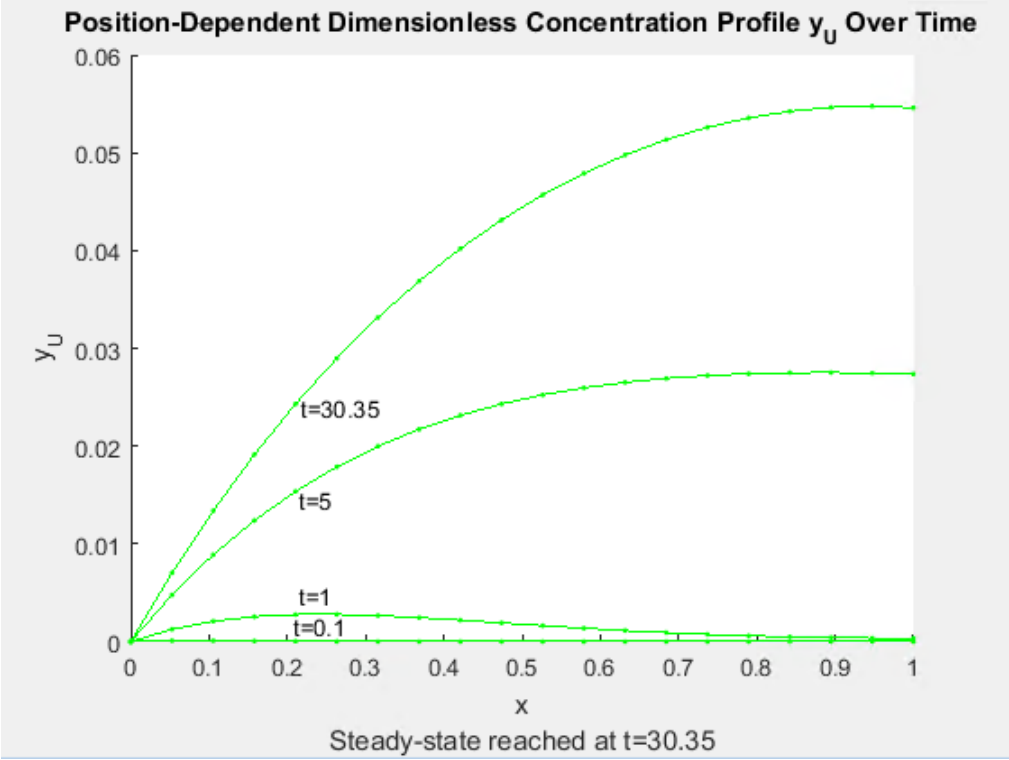


*Figure 6: profile $y_U$ over time*

# 4. Conclusions

We were able to successfully mathematically model a plug-flow diffusion reactor in a non-dimensional manner that allowed us to determine the final concentration profiles of reagents and products when the corresponding physical property coefficients are know. These coefficients were defined for three theoretical cases: A, B, and C. The concentration profiles were determined for those cases when the reactor reached steady-state, where there was no time dependence, using multivariate newtons. For case A, the final mole ratio of product U exiting to reagent A entering was 0. For case B, the final mole ratio of product U to reagent A was approximately 0.06. For case C, the final mole ratio of product U exiting to reagent A entering was 0.05. Finally, the reactor concentration profiles were determined iteratively, using Euler's method. As it varied over time, it was found to eventually have profiles matching the steady-state profiles, suggesting coherence between the two methods of determining the concentration profiles.

# 5. Appendix A: Matlab Code

## DesignProjectP1.m

```
clear all;close all;clc;
constants = [[0 0 0.1 1.5 0.05 0 0 0];...
    [0.05 0 0.1 1.5 0.02 0.1 0.05 0.1];...
    [0.05 0.03 0.1 1.5 0.02 0.1 0.05 0.1]];
constant = constants(3,:);
%Defines the non-second-order terms of each dimensionless concentration ODE
f = {@(yA,yB,yU)-(yA.*(yB.^2))-constant(5)*yA, ...
    @(yA,yB,yU)-2*(yA.*(yB.^2))-(constant(1)*yB)+(constant(2)*yU), ...
    @(yA,yB,yU)constant(1)*yB-constant(2)*yU};
%Defines their derivatives with recpect to each variable(yA,yB,yU)
df = {@(yA,yB,yU)-(yB.^2)-constant(5) @(yA,yB,yU)-2*(yA.*yB) @(yA,yB,yU)0;...
    @(yA,yB,yU)-2*(yB.^2) @(yA,yB,yU)-4*(yA.*yB)-constant(1)
@(yA,yB,yU)constant(2);...
    @(yA,yB,yU)0 @(yA,yB,yU)constant(1) @(yA,yB,yU)-constant(2)};
x = [0 1];
N = 20;
initialGuess = 0;
dx = (x(2)-x(1))/(N-1);
odeCount = length(f);
unknownCount = N*odeCount;
%Function of the finite differentiated second-order ODE;
F = @(y1,y2,y3,func,yA,yB,yU)(dx^-2)*constant(3)*(y3-
(2*y2)+y1)+func(yA,yB,yU);

%Initialization of vectors needed for newtons method
Fv = zeros(unknownCount,1);
J = zeros(unknownCount,unknownCount);
y = initialGuess*ones(N,3);
y(1,3) = 0;

%Constant portion of the Jacobian
for i = 1:unknownCount
    if(i <= odeCount)
        if(i ~= 3)
            J(i,i+odeCount) = 2*constant(3)*(dx^-2);
            J(i,i) = -2*constant(3)*((1+dx)*(dx^-2));
        else
            J(i,i) = 1;
        end
    elseif(i > unknownCount-odeCount)
        J(i,i-odeCount) = 2*constant(3)*(dx^-2);
        if(i == unknownCount-odeCount+1)
            J(i,i) = -2*constant(3)*((1+constant(6)*dx)*(dx^-2));
        elseif(i == unknownCount-odeCount+3)
            J(i,i) = -2*constant(3)*((1+constant(8)*dx)*(dx^-2));
        end
    else
        J(i,i-odeCount) = (dx^-2)*constant(3);
        J(i,i+odeCount) = (dx^-2)*constant(3);
        J(i,i) = -2*(dx^-2)*constant(3);
    end
end
```

```
difference = Inf;
tolerance = 1.5e-7*ones(unknownCount,1);
while(any(difference > tolerance));
    Jacobian = J;
    %Formation of the current Jacobian
    for i = 1:odeCount:unknownCount-odeCount+1
        nodeIndex = ceil(i/odeCount);
        currentRow = num2cell(y(nodeIndex,:));
        if(i == 1)
            Fv(1) = constant(3)*(dx^-2)*(2*y(2,1)-2*(1+dx)*y(1,1)+2*dx)-
(y(1,1).*(y(1,2).^2))-constant(5)*y(1,1);
            Jacobian(1,1) = Jacobian(1,1) + df{1,1}(currentRow{:});
            Jacobian(1,2) = Jacobian(1,2) + df{1,2}(currentRow{:});
            Jacobian(1,3) = Jacobian(1,3) + df{1,3}(currentRow{:});
            Fv(2) = constant(3)*(dx^-2)*(2*y(2,2)-
2*(1+dx)*y(1,2)+2*dx*constant(4))-2*(y(1,1).*(y(1,2).^2))-
constant(1)*y(1,2)+constant(2)*y(1,3);
            Jacobian(2,2) = Jacobian(2,2) + df{2,2}(currentRow{:});
            Jacobian(2,1) = Jacobian(2,1) + df{2,1}(currentRow{:});
            Jacobian(2,3) = Jacobian(2,3) + df{2,3}(currentRow{:});
            Fv(3) = 0;
        else
            Jacobian(i,i) = Jacobian(i,i) + df{1,1}(currentRow{:});
            Jacobian(i,i+1) = Jacobian(i,i+1) + df{1,2}(currentRow{:});
            Jacobian(i,i+2) = Jacobian(i,i+2) + df{1,3}(currentRow{:});
            Jacobian(i+1,i+1) = Jacobian(i+1,i+1) + df{2,2}(currentRow{:});
            Jacobian(i+1,i+1-1) = Jacobian(i+1,i+1-1) + df{2,1}
(currentRow{:});
            Jacobian(i+1,i+1+1) = Jacobian(i+1,i+1+1) + df{2,3}
(currentRow{:});
            Jacobian(i+2,i+2) = Jacobian(i+2,i+2) + df{3,3}(currentRow{:});
            Jacobian(i+2,i+2-1) = Jacobian(i+2,i+2-1) + df{3,2}
(currentRow{:});
            Jacobian(i+2,i+2-2) = Jacobian(i+2,i+2-2) + df{3,1}
(currentRow{:});
            if(i == unknownCount-odeCount+1)
                Fv(i) = constant(3)*(dx^-2)*(2*y(nodeIndex-1,1)-
2*(1+constant(6)*dx)*y(nodeIndex,1))-(y(nodeIndex,1).*(y(nodeIndex,2).^2))-
constant(5)*y(nodeIndex,1);
                Fv(i+1) = constant(3)*(dx^-2)*(2*y(nodeIndex-1,2)-
2*(1+constant(7)*dx*y(nodeIndex,2))*y(nodeIndex,2))-
2*(y(nodeIndex,1).*(y(nodeIndex,2).^2))-
constant(1)*y(nodeIndex,2)+constant(2)*y(nodeIndex,3);
                Jacobian(i+1,i+1) = Jacobian(i+1,i+1) + constant(3)*(dx^-2)*(-
2*(1+2*constant(7)*dx*y(nodeIndex,2)));
                Fv(i+2) = constant(3)*(dx^-2)*(2*y(nodeIndex-1,3)-
2*(1+constant(8)*dx)*y(nodeIndex,3))+constant(1)*y(nodeIndex,2)-
constant(2)*y(nodeIndex,3);
            else
                Fv(i) = F(y(nodeIndex-
1,1),y(nodeIndex,1),y(nodeIndex+1,1),f{1},currentRow{:});
                Fv(i+1) = F(y(nodeIndex-
1,2),y(nodeIndex,2),y(nodeIndex+1,2),f{2},currentRow{:});
                Fv(i+2) = F(y(nodeIndex-
1,3),y(nodeIndex,3),y(nodeIndex+1,3),f{3},currentRow{:});
            end
```

```
        end
    end
    b = Jacobian*reshape(y.',[],1) - Fv;
    tempY = y;
    %Solution to algebraic system
    y = vec2mat(LUSolver(Jacobian,b),odeCount);
    difference = abs(y(:)-tempY(:));
end
save('steadyvals.mat','tempY');
y
xAxis = x(1,1):(x(1,2)-x(1,1))/(N-1):x(1,2);
plot(xAxis, y(:,1), 'r.-', xAxis, y(:,2), 'b.-', xAxis, y(:,3), 'g.-');
hold on;
title('Position-Dependent Dimensionless Concentration Profiles at Steady
State');
legend('yA', 'yB', 'yU'); ylabel('y'); xlabel('x');
```

## DesignProjectP2.m

```
clear all;close all;clc;
constant = [0.05 0.03 0.1 1.5 0.02 0.1 0.05 0.1];
N = 20;
x = [0 1];
dx = (x(2)-x(1))/(N-1);
xAxis = x(1,1):dx:x(1,2);
initialValues = 0;
load('steadyvals.mat');

%Discretized ODE's in time
df = {@(y1,yA,y3,yB,yU)constant(3)*(dx^-2)*(y1-2*yA+y3)-(yA.*(yB.^2))-
constant(5)*yA ...
    @(y1,yB,y3,yA,yU)constant(3)*(dx^-2)*(y1-2*yB+y3)-2*(yA.*(yB.^2))-
constant(1)*yB+constant(2)*yU ...
    @(y1,yU,y3,yA,yB)constant(3)*(dx^-2)*(y1-2*yU+y3)+constant(1)*yB-
constant(2)*yU};

odeCount = length(df);
unknownCount = N*odeCount;
DeriVals = zeros(unknownCount,1);
y = zeros(N,3);
t = 0;
h = 1e-2;
while (any(~any(y>=0.99*tempY)'))
    for i = 1:odeCount:unknownCount-odeCount+1
        nodeIndex = ceil(i/odeCount);
        %Calculates the derivative at the next time step
        if(i == 1)
            DeriVals(i) = constant(3)*(dx^-2)*(2*y(nodeIndex+1,1)-
2*(1+dx)*y(nodeIndex,1)+2*dx)-(y(nodeIndex,1).*(y(nodeIndex,2).^2))-
constant(5)*y(nodeIndex,1);
            DeriVals(i+1) = constant(3)*(dx^-2)*(2*y(nodeIndex+1,2)-
2*(1+dx)*y(nodeIndex,2)+2*dx*constant(4))-
2*(y(nodeIndex,1).*(y(nodeIndex,2).^2))-
constant(1)*y(nodeIndex,2)+constant(2)*y(nodeIndex,3);
            DeriVals(i+2) = 0;
```

```
        elseif(i == unknownCount-odeCount+1)
            DeriVals(i) = constant(3)*(dx^-2)*(2*y(nodeIndex-1,1)-
2*(1+constant(6)*dx)*y(nodeIndex,1))-(y(nodeIndex,1).*(y(nodeIndex,2).^2))-
constant(5)*y(nodeIndex,1);
            DeriVals(i+1) = constant(3)*(dx^-2)*(2*y(nodeIndex-1,2)-
2*(1+constant(7)*dx*y(nodeIndex,2))*y(nodeIndex,2))-
2*(y(nodeIndex,1).*(y(nodeIndex,2).^2))-
constant(1)*y(nodeIndex,2)+constant(2)*y(nodeIndex,2);
            DeriVals(i+2) = constant(3)*(dx^-2)*(2*y(nodeIndex-1,3)-
2*(1+constant(8)*dx)*y(nodeIndex,3))+constant(1)*y(nodeIndex,2)-
constant(2)*y(nodeIndex,3);
        else
            DeriVals(i) = df{1}(y(nodeIndex-
1,1),y(nodeIndex,1),y(nodeIndex+1,1),y(nodeIndex,2),y(nodeIndex,3));
            DeriVals(i+1) = df{2}(y(nodeIndex-
1,2),y(nodeIndex,2),y(nodeIndex+1,2),y(nodeIndex,1),y(nodeIndex,3));
            DeriVals(i+2) = df{3}(y(nodeIndex-
1,3),y(nodeIndex,3),y(nodeIndex+1,3),y(nodeIndex,1),y(nodeIndex,2));
        end
    end
    %Adds the derivative values multipled by h to the initial values
    y = vec2mat(reshape(y.',[],1) + h*DeriVals,odeCount);
    %Plots the profiles at give times
    if(abs(t-0.1) <= h/2 || abs(t-1) <= h/2 || abs(t-5) <= h/2)
        figure(1); hold on;
        plot(xAxis, y(:,1), 'r.-');
        figure(2); hold on;
        plot(xAxis, y(:,2), 'b.-');
        figure(3); hold on;
        plot(xAxis, y(:,3), 'g.-');
    end
    t = t+h
end
%Plots the steady-state
figure(1);
plot(xAxis, y(:,1), 'r.-'); ylabel('y_A');
title('Position-Dependent Dimensionless Concentration Profile y_A Over Time');
xlabel(sprintf('x\nSteady-state reached at t=%3.2f',t));
figure(2);
plot(xAxis, y(:,2), 'b.-'); ylabel('y_B');
title('Position-Dependent Dimensionless Concentration Profile y_B Over Time');
xlabel(sprintf('x\nSteady-state reached at t=%3.2f',t));
figure(3);
plot(xAxis, y(:,3), 'g.-'); ylabel('y_U');
title('Position-Dependent Dimensionless Concentration Profile y_U Over Time');
xlabel(sprintf('x\nSteady-state reached at t=%3.2f',t));
```

## LUSolver.m

```
function x = LUSolver(A, b)
%Solves for the x vector of a system of equations with the form Ax=b using
%the LU method.
%A must be an N by N matrix.
%b must be an N length column vector.
%The returned value of the x is an N length column vector.

N = size(A);
%Conditional to catch over/under-determined systems
if(N(1) ~= N(2))
    error('Matrix A must be an N by N matrix.');
end
N = N(1);
%Conditional to catch invalid b vectors
if(size(b) ~= [N 1])
    error('Vector b must be a column vector and have the same number of rows
as matrix A');
end

x = zeros(N,1);
y = zeros(N,1);
L = A;
U = eye(N,N);
%Pivots rows
Labs = abs(L);
[val m]=max(Labs(1:N,1)); %Find max value from 1 through n to pivot to
L([1 m],:)=L([m 1],:); %Swap the 1st row with the max row
b([1 m])=b([m 1]);
for i=1:N-1 %For diagonal elements, k, eliminate in the column below
    %Forward elimination
    for j=i+1:N
        U(i,j)=L(i,j)/L(i,i); %Find the elimination factor
        L(:,j)=L(:,j)-L(:,i)*U(i,j); %row operation
    end
end
%Forward substitution for y
for i = 1:N
    y(i) = (b(i) - L(i,:)*y)/L(i,i);
end
%Backward substitution for x
for i = N:-1:1
    x(i) = (y(i) - U(i,:)*x)/U(i,i);
end
end
```

# 6. Appendix B: Value Tables

## Figure values

| Steady-state profile values (Case A) | | | | Steady-state profile values (Case B) | | | | Steady-state profile values (Case C) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| x | $y_A$ | $y_B$ | $y_U$ | x | $y_A$ | $y_B$ | $y_U$ | x | $y_A$ | $y_B$ | $y_U$ |
| **0.000** | 0.4132 | 0.5942 | 0.0000 | **0.000** | 0.4917 | 0.5336 | 0.0000 | **0.000** | 0.4891 | 0.5357 | 0.0000 |
| **0.053** | 0.3846 | 0.5506 | 0.0000 | **0.053** | 0.4671 | 0.4870 | 0.0074 | **0.053** | 0.4642 | 0.4892 | 0.0069 |
| **0.105** | 0.3598 | 0.5134 | 0.0000 | **0.105** | 0.4457 | 0.4472 | 0.0142 | **0.105** | 0.4428 | 0.4496 | 0.0132 |
| **0.158** | 0.3381 | 0.4815 | 0.0000 | **0.158** | 0.4271 | 0.4130 | 0.0203 | **0.158** | 0.4240 | 0.4155 | 0.0189 |
| **0.211** | 0.3190 | 0.4539 | 0.0000 | **0.211** | 0.4107 | 0.3833 | 0.0259 | **0.211** | 0.4075 | 0.3860 | 0.0239 |
| **0.263** | 0.3022 | 0.4299 | 0.0000 | **0.263** | 0.3962 | 0.3576 | 0.0309 | **0.263** | 0.3929 | 0.3604 | 0.0285 |
| **0.316** | 0.2874 | 0.4091 | 0.0000 | **0.316** | 0.3834 | 0.3351 | 0.0354 | **0.316** | 0.3800 | 0.3381 | 0.0326 |
| **0.368** | 0.2743 | 0.3909 | 0.0000 | **0.368** | 0.3719 | 0.3155 | 0.0395 | **0.368** | 0.3684 | 0.3186 | 0.0363 |
| **0.421** | 0.2628 | 0.3750 | 0.0000 | **0.421** | 0.3617 | 0.2984 | 0.0431 | **0.421** | 0.3581 | 0.3017 | 0.0395 |
| **0.474** | 0.2526 | 0.3612 | 0.0000 | **0.474** | 0.3526 | 0.2834 | 0.0463 | **0.474** | 0.3490 | 0.2869 | 0.0424 |
| **0.526** | 0.2437 | 0.3493 | 0.0000 | **0.526** | 0.3444 | 0.2705 | 0.0492 | **0.526** | 0.3407 | 0.2741 | 0.0449 |
| **0.579** | 0.2359 | 0.3389 | 0.0000 | **0.579** | 0.3372 | 0.2593 | 0.0516 | **0.579** | 0.3334 | 0.2630 | 0.0471 |
| **0.632** | 0.2293 | 0.3301 | 0.0000 | **0.632** | 0.3307 | 0.2497 | 0.0537 | **0.632** | 0.3269 | 0.2535 | 0.0489 |
| **0.684** | 0.2236 | 0.3226 | 0.0000 | **0.684** | 0.3251 | 0.2416 | 0.0554 | **0.684** | 0.3212 | 0.2455 | 0.0504 |
| **0.737** | 0.2189 | 0.3165 | 0.0000 | **0.737** | 0.3201 | 0.2349 | 0.0568 | **0.737** | 0.3162 | 0.2389 | 0.0517 |
| **0.789** | 0.2151 | 0.3115 | 0.0000 | **0.789** | 0.3158 | 0.2295 | 0.0579 | **0.789** | 0.3119 | 0.2336 | 0.0526 |
| **0.842** | 0.2122 | 0.3077 | 0.0000 | **0.842** | 0.3121 | 0.2253 | 0.0587 | **0.842** | 0.3082 | 0.2295 | 0.0533 |
| **0.895** | 0.2101 | 0.3050 | 0.0000 | **0.895** | 0.3090 | 0.2224 | 0.0591 | **0.895** | 0.3051 | 0.2266 | 0.0537 |
| **0.947** | 0.2089 | 0.3034 | 0.0000 | **0.947** | 0.3065 | 0.2205 | 0.0593 | **0.947** | 0.3027 | 0.2248 | 0.0538 |
| **1.000** | 0.2085 | 0.3029 | 0.0000 | **1.000** | 0.3046 | 0.2199 | 0.0591 | **1.000** | 0.3008 | 0.2241 | 0.0536 |