Name: Robert Heeter

EXAM 2 - BIOE 391
Take Home – 2022

This portion of the exam is !"#$%&!!'(!"#$%$!)#*. Any other resources used must be acknowledged. Please +,-.% -//% 0123+4530612 , manage your time effectively and answer the questions concisely but completely. Please upload a hard copy of the exam as a single PDF or Word file to Canvas. Handwritten formulas (equations) and diagrams are OK if the files are clearly scanned. Please make sure your file is clearly readable before uploading it. The recommended time investment in this take-home exam should be of no more than 7%8!9:*; although you are allowed to use more.

On my honor, I have neither given nor received any unauthorized aid on this exam.

Signature: Robert Heeter

Read Carefully!

Please comment your code as much as possible. This will help us to grade and give YOU partial credit.

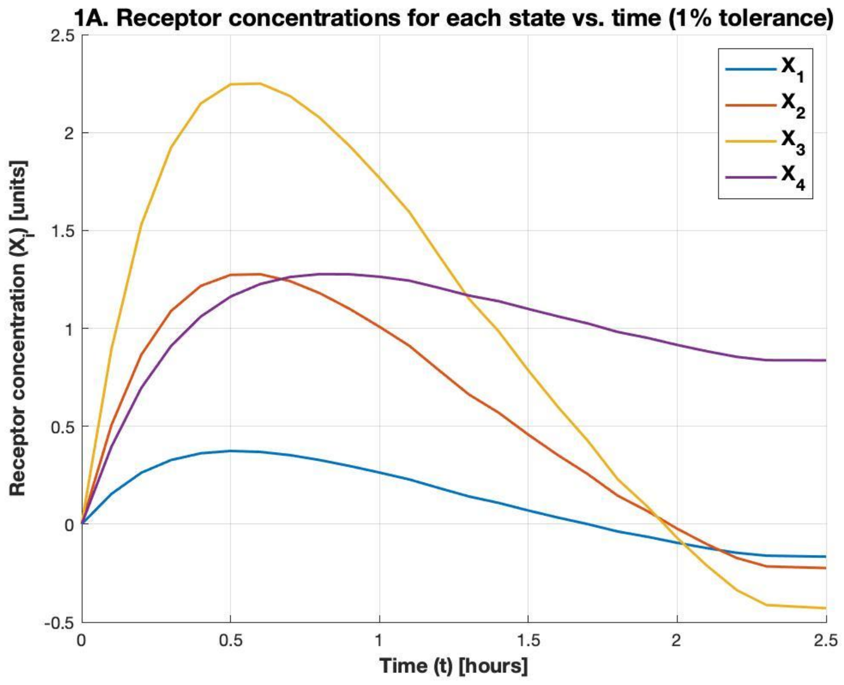
%
%

Robert Heeter
BIOE 391 Numerical Methods – Due 1 April 2022

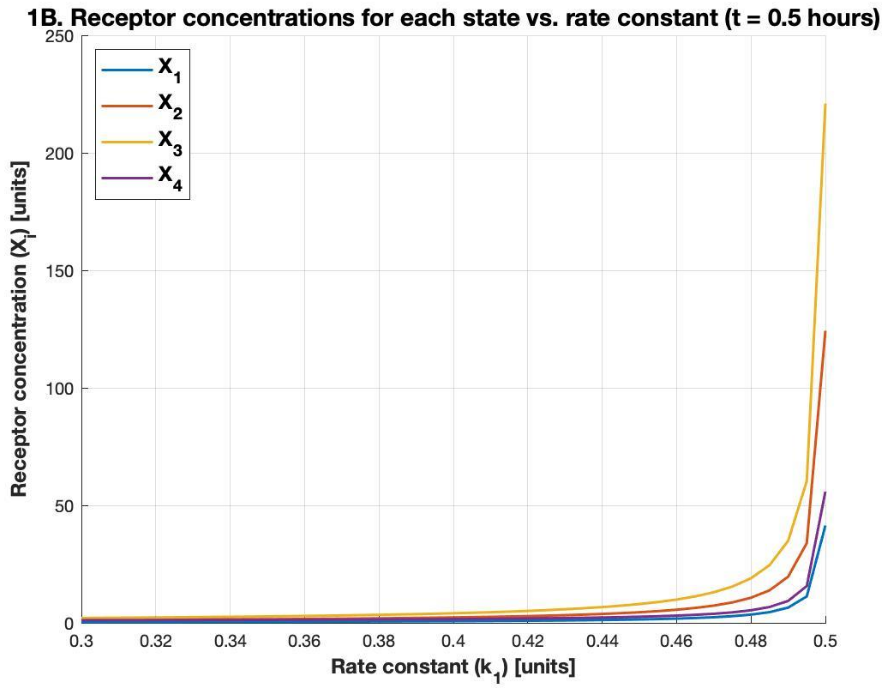
Exam 2 Solutions

1.

a. Figure



b. Figure



Discussion
Looking at the figure produced in Problem 1B, the numerical solutions for the receptor concentrations for the 4 different states are very sensitive to changes in the rate constant (k_1) as the rate constant approaches 0.5. Near $k_1 = 0.5$, small uncertainties in the rate constant can have enormous effects on the calculated receptor concentrations. At lower values of the rate constant, small deviations do not have as large of an effect on the concentration solutions. There appears to be a horizontal asymptote at 0 concentration towards smaller rate constants (nearer to $k_1 = 0.3$) and a vertical asymptote at 0.5 rate constant.

2.

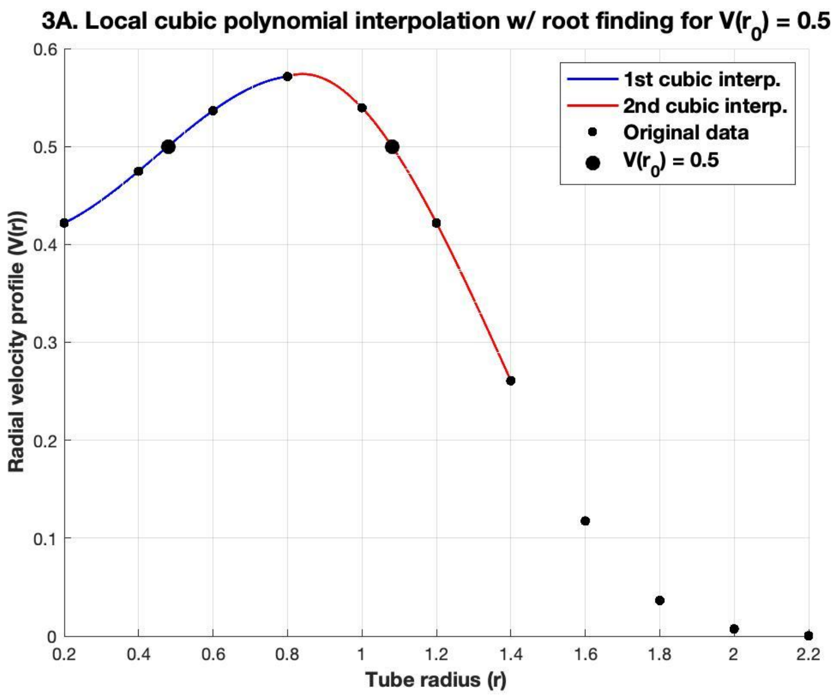
a. **Output**
Avg. concentration of nutrient = 2.250000 [mM]

b. **Justification**
A Simpson's ⅓ rule integration method was used to compute the average concentration of the nutrient in the chamber. Simpson's ⅓ rule is exact for 3rd-order functions (error proportional to 4th derivative), so with a simple division of the given cubic function into 3 equally-spaced points, the method should give an exact answer for the integral. This is interesting as the integration is only based on 3 points (which would normally define a parabola) but gives an exact answer.

3.

a. **Output**
First V(r0) = 0.5: r0 = 0.478885
Second V(r0) = 0.5: r0 = 1.080675

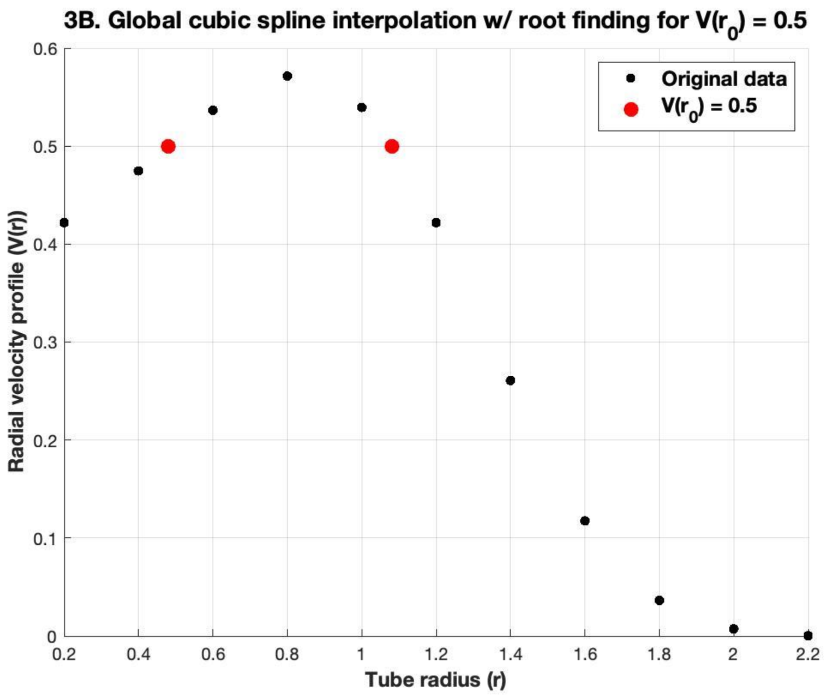
Figure



b. Output

First V(r0) = 0.5: r0 = 0.479152
Second V(r0) = 0.5: r0 = 1.082458

Figure



c. **Output**

Largest positive r0 for V(r0) = 0.5: r0 = 1.206303

Logarithmic transformed equation: $ln[V(r)] \;=\; ln\left(V_0\right) - \beta r^3 + \gamma r^2$

Figure (normal axes)

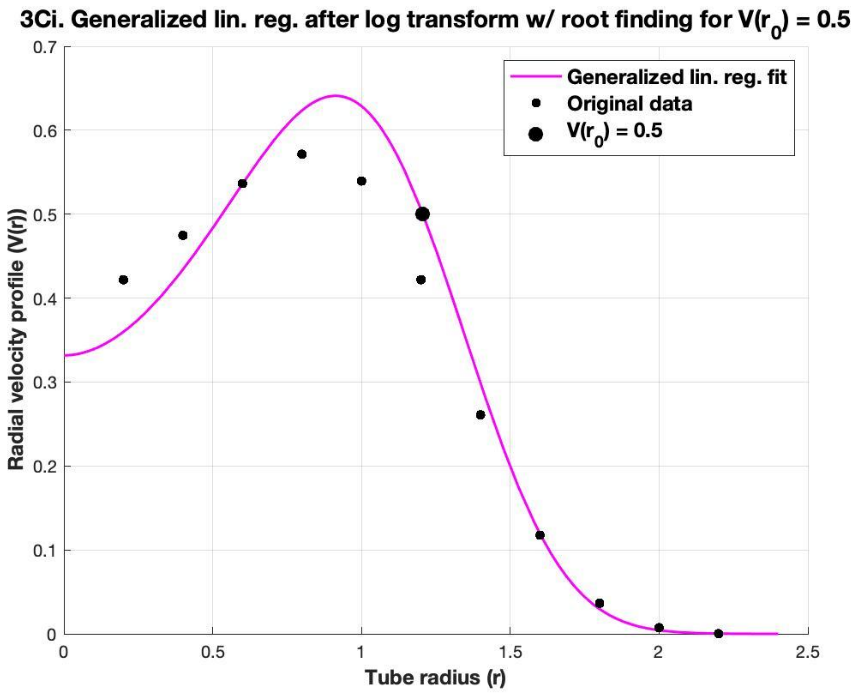
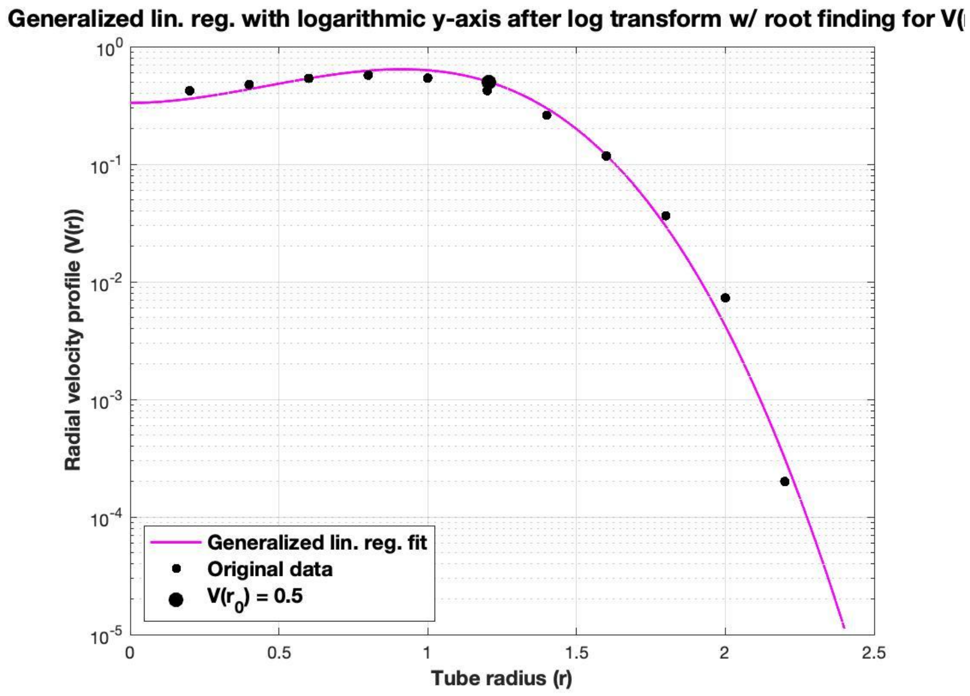


Figure (logarithmic *y*-axis)



d. Output

Smallest positive r0 for V(r0) = 0.5: r0 = 0.479363

Figure (normal axes)

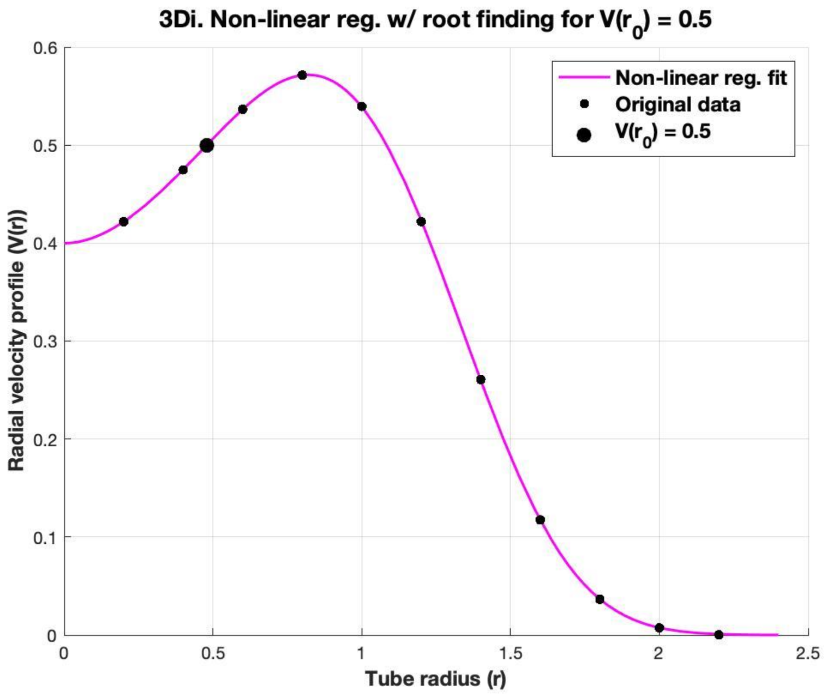
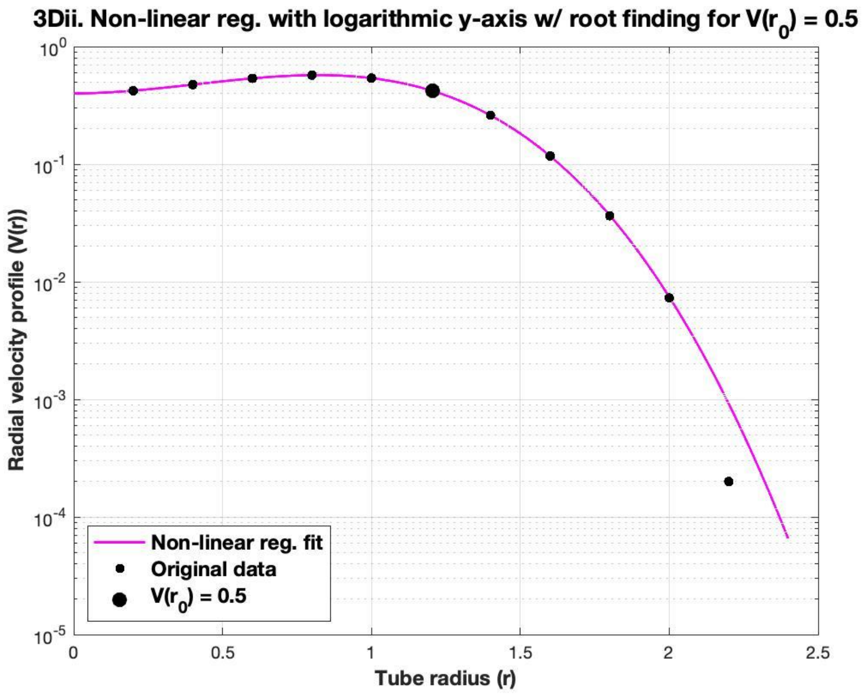


Figure (logarithmic *y*-axis)



e. **Comparison**
The fit produced using non-linear least-squares regression is significantly better than the fit produced with a logarithmic-transformed linear regression model, and is more appropriate for this application. It is clear from the graphs between parts C and D that the generalized linear regression from the transformed function deviates more from the original data than the non-linear regression, which fits nearly perfectly. Most of the points do not lie on the fit line in the transformed linear regression case.

Looking at the normally-scaled axes, the transformed linear regression (part C) gives an r_0 that does not appear to fit the natural curve of the data (the fit "bulges" out), whereas the non-linear regression (part D) gives an r_0 that "lines up" with the neighboring data.

Looking at the logarithmic-scaled *y*-axis graphs, the transformed linear regression (part C) gives an r_0 that is noticeably elevated from the rest of the data, while the non-linear regression (part D) gives an r_0 point that lies on the well-fitted curve.

Robert Heeter
BIOE 391 Numerical Methods – Due 1 April 2022

Complete MATLAB Code

```
% Robert Heeter
% BIOE 391 Numerical Methods
% EXAM 2 MATLAB SCRIPT

clc, clf, clear, close all

%% PROBLEM 1, PART A
disp('PROBLEM 1');

A = 2; % constant
k1 = 0.4; % rate constant
t = 0:0.1:2.5; % time interval (hours)

X1 = zeros(length(t),1); % preallocate vectors for each state receptor concentration
X2 = zeros(length(t),1);
X3 = zeros(length(t),1);
X4 = zeros(length(t),1);

for i = 1:length(t)
    b = [A*(1-exp(-1*t(i))); 0; 0; -1.9*(1-exp(-2*t(i)))]; % time-dependent influx/efflux of receptor
forms
    Amat = [-10 2 k1 1; 8.99 -7 2 1; 1 5 -3 0; 0 0 0.5 -2]; % matrix
    X = gaussseidel(Amat,b,1,1000); % use gaussseidel function below with tolerance es = 1%
    X1(i) = X(1);
    X2(i) = X(2);
    X3(i) = X(3);
    X4(i) = X(4);
end

% Plot results
figure
hold on
plot(t,X1,'LineWidth',1.5);
plot(t,X2,'LineWidth',1.5);
plot(t,X3,'LineWidth',1.5);
plot(t,X4,'LineWidth',1.5);
xlabel('Time (t) [hours]','FontSize',12,'FontWeight','bold');
ylabel('Receptor concentration (X_i) [units]','FontSize',12,'FontWeight','bold');
title('1A. Receptor concentrations for each state vs. time (1%
tolerance)','FontSize',14,'FontWeight','bold');
legend('X_1','X_2','X_3','X_4','FontSize',12,'FontWeight','bold','Location','NorthEast');
grid on
hold off


%% PROBLEM 1, PART B

A = 2; % constant
k1 = 0.3:0.005:0.5; % rate constant values
t = 0.5; % time (hours)

X1 = zeros(length(t),1); % preallocate vectors for each state receptor concentration
X2 = zeros(length(t),1);
X3 = zeros(length(t),1);
X4 = zeros(length(t),1);

for i = 1:length(k1)
    X = concentration(k1(i),t,A); % use concentration function below to find X vector
    X1(i) = X(1);
    X2(i) = X(2);
    X3(i) = X(3);
    X4(i) = X(4);
end
```

Robert Heeter
BIOE 391 Numerical Methods – Due 1 April 2022

```
% Plot results
figure
hold on
plot(k1,X1,'LineWidth',1.5);
plot(k1,X2,'LineWidth',1.5);
plot(k1,X3,'LineWidth',1.5);
plot(k1,X4,'LineWidth',1.5);
xlabel('Rate constant (k_1) [units]','FontSize',12,'FontWeight','bold');
ylabel('Receptor concentration (X_i) [units]','FontSize',12,'FontWeight','bold');
title('1B. Receptor concentrations for each state vs. rate constant (t = 0.5
hours)','FontSize',14,'FontWeight','bold');
legend('X_1','X_2','X_3','X_4','FontSize',12,'FontWeight','bold','Location','NorthWest');
grid on
hold off


%% PROBLEM 2, PART A
disp('PROBLEM 2');

Cx = @(x) (x-1).^3 + 1; % concentration equation
x = linspace(0,3,3); % set of 3 equally-spaced points for Simpson's 1/3 rule

I = simpson13(Cx,x); % use simpson13 function below for integration

Cavg = I/3; % average value of concentration over length interval(3mm)

% Display results
fprintf('PART A:\nAvg. concentration of nutrient = %f [mM]\n\n',Cavg);


%% PROBLEM 2, PART B

% No code necessary; see submission document for justification.


%% PROBLEM 3, PART A
disp('PROBLEM 3');

% Original data
r = (0.2:0.2:2.2)';
Vr = [0.4218 0.4747 0.5365 0.5714 0.5395 0.4219 0.2608 0.1175 0.0364 0.0073 0.0002]';

% First r_0
P3a1 = polyfit(r(1:4),Vr(1:4),3); % use in-built polyfit function for cubic interpolation of data near
first expected r0
P3a1_adjust = P3a1;
P3a1_adjust(end) = P3a1(end) - 0.5; % adjust final polynomial coefficient for root-finding
Ra1 = roots(P3a1_adjust); % use in-built roots function to find root
r0a1 = Ra1(2); % second root of cubic is target root

% Second r_0
P3a2 = polyfit(r(4:7),Vr(4:7),3); % use in-built polyfit function for cubic interpolation of data near
second expected r0
P3a2_adjust = P3a2;
P3a2_adjust(end) = P3a2(end) - 0.5; % adjust final polynomial coefficient for root-finding
Ra2 = roots(P3a2_adjust); % use in-built roots function to find root
r0a2 = Ra2(2); % second root of cubic is target root

% Display results
fprintf('PART A:\nFirst V(r0) = 0.5: r0 = %f\nSecond V(r0) = 0.5: r0 = %f\n\n',r0a1,r0a2);

% Plot results
xa1 = 0.2:0.0001:0.8; % vectors of cubic interpolation fit data for graphing
P3a1x = polyval(P3a1,xa1);
xa2 = 0.8:0.0001:1.4;
```

```
P3a2x = polyval(P3a2,xa2);

figure
hold on
plot(xa1,P3a1x,'-b','LineWidth',1.5);
plot(xa2,P3a2x,'-r','LineWidth',1.5);
plot(r,Vr,'.k','MarkerSize',15);
plot(r0a1, polyval(P3a1,r0a1),'.k','MarkerSize',30);
plot(r0a2, polyval(P3a2,r0a2),'.k','MarkerSize',30);
xlabel('Tube radius (r)','FontSize',12,'FontWeight','bold');
ylabel('Radial velocity profile (V(r))','FontSize',12,'FontWeight','bold');
title('3A. Local cubic polynomial interpolation w/ root finding for V(r_0) =
0.5','FontSize',14,'FontWeight','bold');
legend('1st cubic interp.','2nd cubic interp.','Original data','V(r_0) =
0.5','FontSize',12,'FontWeight','bold','Location','NorthEast');
grid on
hold off


%% PROBLEM 3, PART B

% First r_0
r0b1 = fzero(@(r0) spline(r,Vr,r0)-0.5,0.4); % use in-built fzero function to find first root of
in-built not-a-knot spline function

% Second r_0
r0b2 = fzero(@(r0) spline(r,Vr,r0)-0.5,1.2); % use in-built fzero function to find second root of
in-built not-a-knot spline function

% Display results
fprintf('PART B:\nFirst V(r0) = 0.5: r0 = %f\nSecond V(r0) = 0.5: r0 = %f\n\n',r0b1,r0b2);

% Plot results
figure
hold on
plot(r,Vr,'.k','MarkerSize',15);
plot(r0b1,spline(r,Vr,r0b1),'.r','MarkerSize',30);
plot(r0b2,spline(r,Vr,r0b2),'.r','MarkerSize',30);
xlabel('Tube radius (r)','FontSize',12,'FontWeight','bold');
ylabel('Radial velocity profile (V(r))','FontSize',12,'FontWeight','bold');
title('3B. Global cubic spline interpolation w/ root finding for V(r_0) =
0.5','FontSize',14,'FontWeight','bold');
legend('Original data','V(r_0) = 0.5','FontSize',12,'FontWeight','bold','Location','NorthEast');
grid on
hold off


%% PROBLEM 3, PART C

% Linear least-squares to find coefficients
y = log(Vr); % log-transformed Vr
Z = [ones(size(r)) r.^3 r.^2]; % Z-matrix of terms
a = (Z'*Z)\(Z'*y); % vector of coefficients

% Convert log-transformed terms back to original form
V0 = exp(a(1));
beta = -1*a(2);
gamma = a(3);

% Find root
Vr_tlreg = @(r) V0.*exp((-1.*beta.*r.^3) + (gamma.*r.^2));
r0c = fzero(@(r) Vr_tlreg(r) - 0.5,2); % use in-built fzero function to find root of regression fit

% Display results
fprintf('PART C:\nLargest positive r0 for V(r0) = 0.5: r0 = %f\n\n',r0c);
```

```
% Plot results
figure
hold on
fplot(Vr_tlreg,[0,2.4],'-m','LineWidth',1.5);
plot(r,Vr,'.k','MarkerSize',15);
plot(r0c,Vr_tlreg(r0c),'.k','MarkerSize',30);
xlabel('Tube radius (r)','FontSize',12,'FontWeight','bold');
ylabel('Radial velocity profile (V(r))','FontSize',12,'FontWeight','bold');
title('3Ci. Generalized lin. reg. after log transform w/ root finding for V(r_0) =
0.5','FontSize',14,'FontWeight','bold');
legend('Generalized lin. reg. fit','Original data','V(r_0) =
0.5','FontSize',12,'FontWeight','bold','Location','NorthEast');
grid on
hold off

xc = 0:0.0001:2.4;
Vrc = Vr_tlreg(xc);

figure
semilogy(xc,Vrc,'-m','LineWidth',1.5);
hold on
semilogy(r,Vr,'.k','MarkerSize',15);
semilogy(r0c,Vr_tlreg(r0c),'.k','MarkerSize',30);
xlabel('Tube radius (r)','FontSize',12,'FontWeight','bold');
ylabel('Radial velocity profile (V(r))','FontSize',12,'FontWeight','bold');
title('3Cii. Generalized lin. reg. with logarithmic y-axis after log transform w/ root finding for
V(r_0) = 0.5','FontSize',14,'FontWeight','bold');
legend('Generalized lin. reg. fit','Original data','V(r_0) =
0.5','FontSize',12,'FontWeight','bold','Location','SouthWest');
grid on
hold off


%% PROBLEM 3, PART D

% Use fminsearch to find constants that minimize sum of squares of estimate residuals
a = fminsearch(@(a) VrSSR(a,r,Vr),[V0 beta gamma]'); % use function VrSSR below for sum of squares of
residuals

% Find root
Vr_nlreg = @(r) a(1).*exp((-1.*a(2).*r.^3) + (a(3).*r.^2));
r0d = fzero(@(r) Vr_nlreg(r) - 0.5,0.5); % use in-built fzero function to find root of non-linear
regression fit

% Display results
fprintf('PART D:\nSmallest positive r0 for V(r0) = 0.5: r0 = %f\n\n',r0d);

% Plot results
figure
hold on
fplot(Vr_nlreg,[0,2.4],'-m','LineWidth',1.5);
plot(r,Vr,'.k','MarkerSize',15);
plot(r0d,Vr_nlreg(r0d),'.k','MarkerSize',30);
xlabel('Tube radius (r)','FontSize',12,'FontWeight','bold');
ylabel('Radial velocity profile (V(r))','FontSize',12,'FontWeight','bold');
title('3Di. Non-linear reg. w/ root finding for V(r_0) = 0.5','FontSize',14,'FontWeight','bold');
legend('Non-linear reg. fit','Original data','V(r_0) =
0.5','FontSize',12,'FontWeight','bold','Location','NorthEast');
grid on
hold off

xd = 0:0.0001:2.4;
Vrd = Vr_nlreg(xc);

figure
semilogy(xd,Vrd,'-m','LineWidth',1.5);
```

Robert Heeter
BIOE 391 Numerical Methods – Due 1 April 2022

```
hold on
semilogy(r,Vr,'.k','MarkerSize',15);
semilogy(r0c,Vr_nlreg(r0c),'.k','MarkerSize',30);
xlabel('Tube radius (r)','FontSize',12,'FontWeight','bold');
ylabel('Radial velocity profile (V(r))','FontSize',12,'FontWeight','bold');
title('3Dii. Non-linear reg. with logarithmic y-axis w/ root finding for V(r_0) =
0.5','FontSize',14,'FontWeight','bold');
legend('Non-linear reg. fit','Original data','V(r_0) =
0.5','FontSize',12,'FontWeight','bold','Location','SouthWest');
grid on
hold off


%% PROBLEM 3, PART E

% No code necessary; see submission document for explanation.


%% Additional Functions

function X = concentration(k1,t,A)
% ABOUT: For Problem 1B.

    b = [A*(1-exp(-1*t)); 0; 0; -1.9*(1-exp(-2*t))]; % time-dependent influx/efflux of receptor forms
    Amat = [-10 2 k1 1; 8.99 -7 2 1; 1 5 -3 0; 0 0 0.5 -2]; % matrix
    X = Amat\b; % use backslash to solve linear system

end


function I = simpson13(func,x)
% ABOUT: Simpson's 1/3 rule equation for Problem 2A.

I = (x(3)-x(1))*(func(x(1))+(4*func(x(2)))+func(x(3)))/6;

end


function ssr = VrSSR(a,r,Vr)
% ABOUT: Non-linear regression squared residual summation function for
% Problem 3D.

Vrp = a(1).*exp((-1.*a(2).*r.^3) + (a(3).*r.^2));
ssr = sum((Vr-Vrp).^2);

end


function x = gaussseidel(A,b,es,maxit)
% ABOUT: Gauss-Seidel method from textbook .m file.
% INPUTS: A = coefficient matrix; b = right side vector; es = relative
% error threshold (default = 0.00001%); maxit = max iterations (default =
% 50)
% OUTPUTS: x = solution vector

if nargin < 2
    error('At least 2 input arguments required.')
end
if nargin<4 || isempty(maxit)
    maxit=50;
end
if nargin<3 || isempty(es)
    es=0.00001;
end

[m,n] = size(A);
```

Robert Heeter
BIOE 391 Numerical Methods – Due 1 April 2022

```
if m~=n
    error('Matrix A must be square.');
end

C = A;
x = zeros(1,n);
d = zeros(1,n);

for i = 1:n
  C(i,i) = 0;
end
x = x';
for i = 1:n
  C(i,1:n) = C(i,1:n)/A(i,i);
end
for i = 1:n
  d(i) = b(i)/A(i,i);
end

iter = 0;
ea = 100;
while (1)
  xold = x;
  for i = 1:n
    x(i) = d(i)-C(i,:)*x;
    if x(i) ~= 0
      ea = abs((x(i) - xold(i))/x(i)) * 100;
    end
  end
  iter = iter+1;
  if max(ea)<=es || iter >= maxit
      break
  end
end

end
```