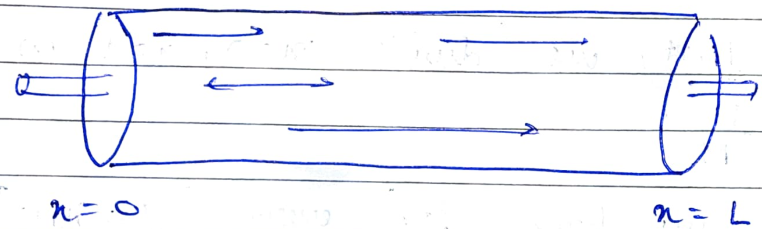
M T W T F S S
Page No.:
Date:
YOUVA

CL249: ASSIGNMENT 9

Himanshu Choudhary (200020059)

PROBLEM

We have to solve the following diff. eqn which describes the steady state conc. of substance that reacts with 1st order kinetics in an axially-dispersed plug flow reactor.



$$D\frac{d^2C}{dx^2} - u\frac{dc}{dx} = kc$$

D = diffusion const = 5000 m^2/hr
C = conc. of subs.
u = Axial velocity = 100 m/hr
K = Rate const = $2h^{-1}$
C_{inlet} = 100 mol/L

(1) $U C_{inlet} = u C_{x=0} - D\frac{dc}{dx}\Big|_{x=0}$

(2) $\frac{dc}{dx}\Big|_{x=L} = 0$

L = length = 100 m

We have to submit plot of y vs x for diff. h.

Description of method

Boundary value problem

we have a diff. eqn

$$D \frac{d^2c}{dx^2} - v\frac{dc}{dx} = kc$$

first, we divide $x=0$, $x=1$ in N parts

$h = \frac{L}{N}$;

then we have for every $[x_{i-1}, x_{i+1}]$

$$\frac{d^2c}{dx^2} = \frac{C_{i+1} - 2C_i + C_{i-1}}{h^2}$$

$$\frac{dc}{dx} = \frac{C_{i+1} - C_{i-1}}{2h}$$

$$D\left(\frac{C_{i+1}}{h^2} - \frac{2C_i}{h^2} + \frac{C_{i-1}}{h^2}\right) - v\left[\frac{C_{i+1} - C_{i-1}}{2h}\right] = kC_i$$

$$2D(C_{i+1} - 2C_i + C_{i-1}) - hv(C_{i+1} - C_{i-1}) = 2h^2kC_i$$

$$(2D + hv)C_{i-1} - (2h^2k + 4D)C_i + (2D - hv)C_{i+1} = 0 \quad \text{--- (1)}$$

and Bound. cond.

$$uC_{in} = uC_1 - D\left(\frac{C_2 - C_0}{2h}\right)$$

$$2hUC_{in} = 2hUC_1 - DC_2 + DC_0 \quad \text{--- (2)}$$

and

$$\frac{C_{n+2} - C_n}{2h} = 0 \quad \text{--- (3)}$$

we can define matrix in which

$$
\begin{bmatrix} D & -2hv-D & 0 & \cdots & 0 \\ 0 & 2D+hv & -2h^2k-4D & -2D+hv & \\ & & \cdots & & \\ & & \cdots & & \\ 0 & 0 & \cdots 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_{n+1} \end{bmatrix} = \begin{bmatrix} 2h\,v\,C_{in} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}
$$

solve this using gauss elim

Pseudo code

main.m

loop in N to increase values

$N = 2^i$

divide X in parts

get y from solun.

$\text{error} = \dfrac{y - y_{pre}}{y}$

get maximum error

if max err > tol

normal plot

else bold plot

solve.m

define constants

take inputs

A, B = zeros

loop to define general matrix

$A(i, i+1) = $ —

define boundary cond. in the matrix

solve using gauss elim

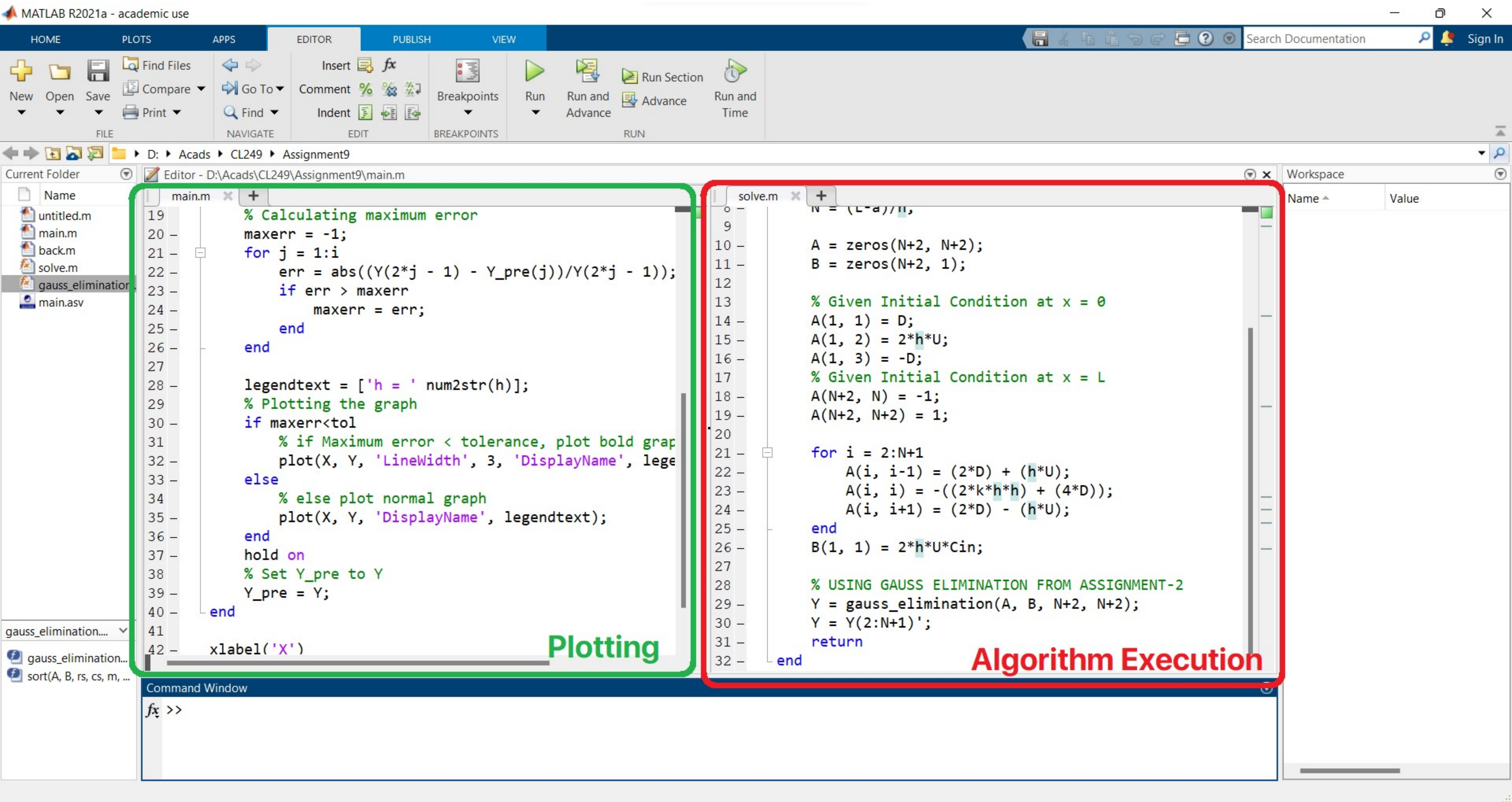
return value

```
% Boundary
a = 0;
b = 100;

% Tolerance
tol = 10^-4;

% Defining Y_pre to store previous Y matrix for error calculation
% Size = 4 due to first iteration of the loop
Y_pre = zeros(1, 4);
for i = 2:10
    % Increasing N exponentially
    N = 2^i;
    h = (b-a)/N;
    % X interval
    X = linspace(a, b, N);
    % Getting Y through solver
    Y = solve(a, b, h);
    % Calculating maximum error
    maxerr = -1;
    for j = 1:i
        err = abs((Y(2*j - 1) - Y_pre(j))/Y(2*j - 1));
        if err > maxerr
            maxerr = err;
        end
    end

    legendtext = ['h = ' num2str(h)];
    % Plotting the graph
    if maxerr<tol
        % if Maximum error < tolerance, plot bold graph to highlight
        plot(X, Y, 'LineWidth', 3, 'DisplayName', legendtext);
    else
        % else plot normal graph
        plot(X, Y, 'DisplayName', legendtext);
    end
    hold on
    % Set Y_pre to Y
    Y_pre = Y;
end

xlabel('X')
ylabel('Concentration')
title('Plot of Concentration (x) vs Distance from origin (x)')
legend
```

```
function Y = solve(a, b, h)
    Cin = 100;
    L = b;
    k = 2;
    U = 100;
    D = 5000;

    N = (L-a)/h;

    A = zeros(N+2, N+2);
    B = zeros(N+2, 1);

    % Given Initial Condition at x = 0
    A(1, 1) = D;
    A(1, 2) = 2*h*U;
    A(1, 3) = -D;
    % Given Initial Condition at x = L
    A(N+2, N) = -1;
    A(N+2, N+2) = 1;

    for i = 2:N+1
        A(i, i-1) = (2*D) + (h*U);
        A(i, i) = -((2*k*h*h) + (4*D));
        A(i, i+1) = (2*D) - (h*U);
    end
    B(1, 1) = 2*h*U*Cin;

    % USING GAUSS ELIMINATION FROM ASSIGNMENT-2
    Y = gauss_elimination(A, B, N+2, N+2);
    Y = Y(2:N+1)';
    return
end
```

```
function X = gauss_elimination(A, B, m, n)
    operations = 0;
    X = zeros(n,1); % Initialize X

    % Sorting initially
    [A, B] = sort(A, B, 1, 1, m, n);

    for c = 1:n
        % Sorting A and B (max. diagonal element)
        [A, B] = sort(A, B, c, c, m, n);
        for r = m:-1:c+1
            if (A(r,c) ~= 0)
                factor = A(r, c)/A(c, c); % 1 operation
                A(r,:) = A(r, :) - (factor*A(c, :)); % 2*n operations
                B(r) = B(r) - (factor*B(c)); % 2 operations
                operations = operations + (2*n) + 3;
            end
        end
    end

    % Back-Substitution
    X(n) = (B(n)/A(n,n));
    operations = operations + 1;
    for i = m-1:-1:1
        sum = 0;
        for j = n:-1:i+1
            sum = sum + (X(j)*A(i, j));
            operations = operations + 2;
        end
        X(i) = (B(i) - sum)/A(i, i);
        operations = operations + 2;
    end

end

% Sorting Function
function [mat1, mat2] = sort(A, B, rs, cs, m, n)
    for s = rs:m-1
        for r = rs:m-1
            if abs(A(r, cs)) < abs(A(r+1, cs))
                temp1 = A(r, :);
                A(r, :) = A(r+1, :);
                A(r+1, :) = temp1;
                temp2 = B(r);
                B(r) = B(r+1);
                B(r+1) = temp2;
            end
        end
    end
    mat1 = A;
```

```
    mat2 = B;
end
```


MATLAB R2021a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
FILE
Go To
Find
NAVIGATE
Insert
Comment
Indent
EDIT
Breakpoints
BREAKPOINTS
Run
Run and Advance
Run Section
Advance
Run and Time
RUN
D: ▸ Acads ▸ CL249 ▸ Assignment9
Current Folder
Name
untitled.m
main.m
back.m
solve.m
gauss_elimination
main.asv
Editor - D:\Acads\CL249\Assignment9\main.m
main.m
% Calculating maximum error
maxerr = -1;
for j = 1:i
err = abs((Y(2*j - 1) - Y_pre(j))/Y(2*j - 1));
if err > maxerr
maxerr = err;
end
end
legendtext = ['h = ' num2str(h)];
% Plotting the graph
if maxerr<tol
% if Maximum error < tolerance, plot bold grap
plot(X, Y, 'LineWidth', 3, 'DisplayName', lege
else
% else plot normal graph
plot(X, Y, 'DisplayName', legendtext);
end
hold on
% Set Y_pre to Y
Y_pre = Y;
end
xlabel('X')
Plotting
solve.m
A = zeros(N+2, N+2);
B = zeros(N+2, 1);
% Given Initial Condition at x = 0
A(1, 1) = D;
A(1, 2) = 2*h*U;
A(1, 3) = -D;
% Given Initial Condition at x = L
A(N+2, N) = -1;
A(N+2, N+2) = 1;
for i = 2:N+1
A(i, i-1) = (2*D) + (h*U);
A(i, i) = -((2*k*h*h) + (4*D));
A(i, i+1) = (2*D) - (h*U);
end
B(1, 1) = 2*h*U*Cin;
% USING GAUSS ELIMINATION FROM ASSIGNMENT-2
Y = gauss_elimination(A, B, N+2, N+2);
Y = Y(2:N+1)';
return
end
Algorithm Execution
Workspace
Name
Value
gauss_elimination...
gauss_elimination...
sort(A, B, rs, cs, m, ...
Command Window
fx >>

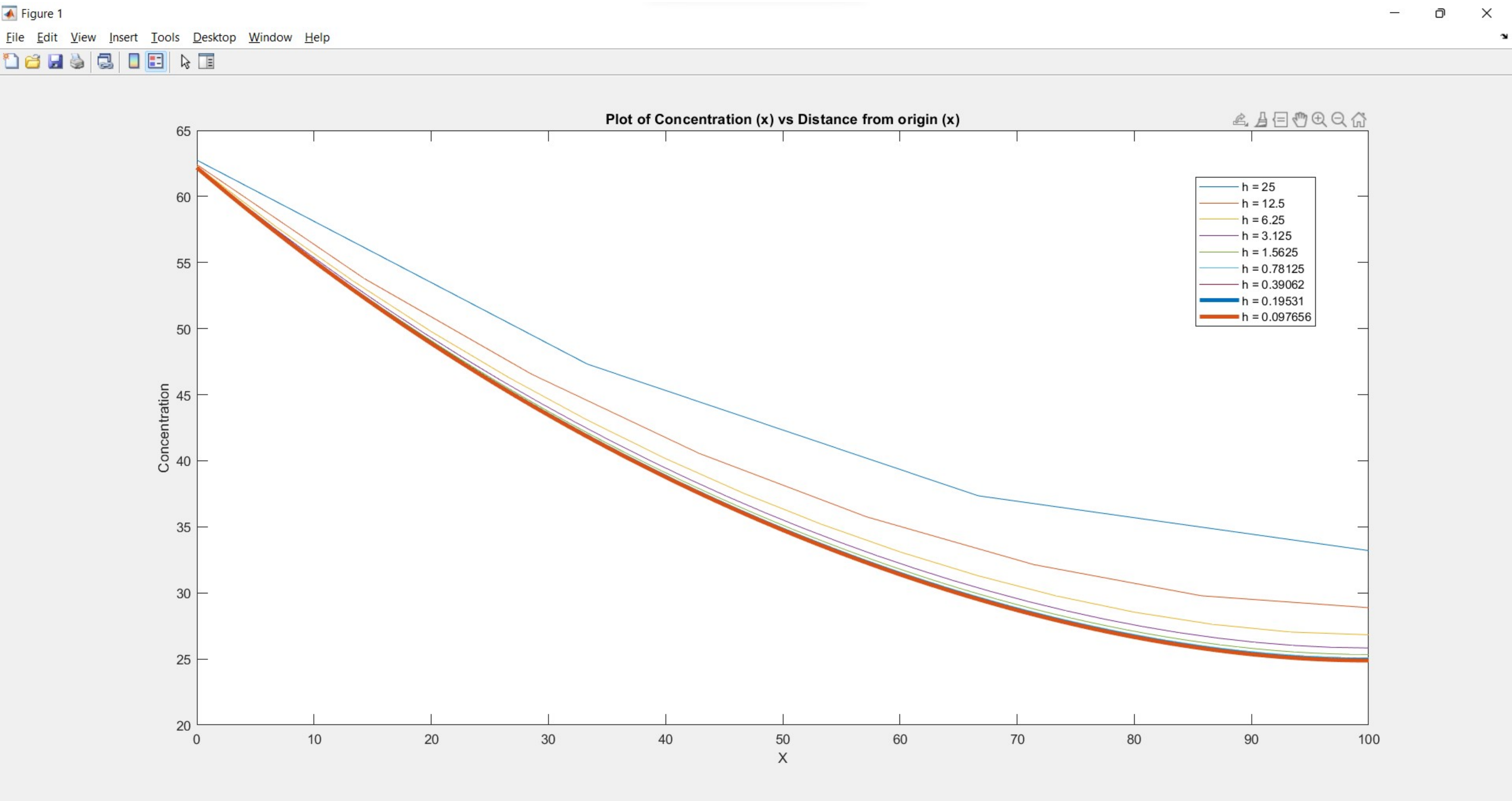


Figure 1
File Edit View Insert Tools Desktop Window Help
Plot of Concentration (x) vs Distance from origin (x)
Concentration
X
h = 25
h = 12.5
h = 6.25
h = 3.125
h = 1.5625
h = 0.78125
h = 0.39062
h = 0.19531
h = 0.097656
65
60
55
50
45
40
35
30
25
20
0
10
20
30
40
50
60
70
80
90
100