

Coursework 2

Hassan Miah

4254974

Contents

[Part A 2](#_Toc8768919)

[Part B 5](#_Toc8768920)

[Discussion 13](#_Toc8768921)

[References 14](#_Toc8768922)

[References 14](#_Toc8768923)

[Appendix 15](#_Toc8768924)

[Figure 1: Part 1 of code 3](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679322)

[Figure 5: Loop 5](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679323)

[Figure 6: Part A results 5](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679324)

[Figure 11: Results of part B, a when cfl = 0.5 11](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679325)

[Figure 12: Results of part B, a when cfl = 0.9 12](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679326)

[Figure 13: Part B, a 13](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679327)

[Figure 22: Part B, b results 14](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679328)

[Figure 31: Part B, c results 15](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679329)

[Figure 32: Part 1 of code 16](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679330)

[Figure 40: Part B, d results 16](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679331)

[Figure 42: Part A, a code 17](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679332)

[Figure 43: Part A, b code 18](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679333)

[Figure 44: Part B, a code 19](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679334)

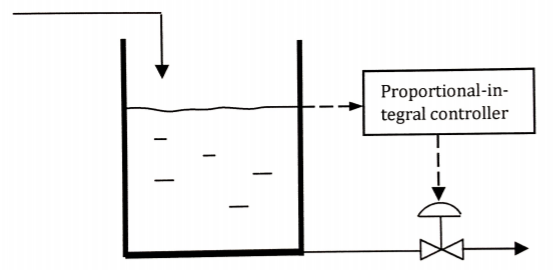
[Figure 45: Part B, b code 20](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679335)

[Figure 46: Part B, c code 21](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679336)

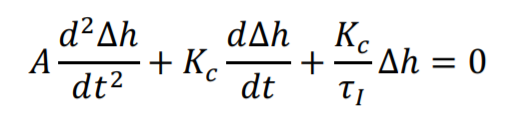
[Figure 47: Part B, d code 22](file:///C:\Users\Hassan\Documents\MATLAB\ACM%20cw2%20final.docx#_Toc8679337)

## Part A

The liquid level of a tank is controlled at a desired value when the inlet flow rate undergoes step change. As shown in figure below, a feedback control system is used. This control system measures the liquid level and compares it with the desired steady-state value. If the level is higher than the desired value, it increases the effluent flow rate by opening the control valve, while it closes the valve when the level is lower than the desired value.



If the controller is a proportional-integral one, the following second-order ODE can be set up.



where A is the cross-sectional area of the tank (=2 m2), Kc is the proportional gain. τ1 is the integral time constant, t is time, and ∆ℎ is the deviation height of the liquid level in the tank, i.e., the difference between the actual height and the desired value. For a certain step change of the inlet flow rate, the following initial conditions can be written: at t = 0, , and ∆ℎ = 0.

(20%) Build your own solver using your own choice of method and solve the ODE if Kc and τ1 are set to 1 m2/min and 0.1 min, respectively. Plot ∆ℎ as a function of time and observe the dynamic behaviour of the liquid level of the tank.

(20%) Using the worksheet in part a, vary the value of Kc (for example from 1 to 5) to learn about the effect of this parameter on the oscillatory behaviour of the dynamic response of the liquid level due to a step change of the inlet flow rate. If the steady state value should be reached quickly, discuss whether or not a large value of Kc is better than a small one.

Initial

Set parameters:

A = 2, taf1 = 0.1, h = 0.1, t\_final = 35, N = t\_fianl/h, Kc = 1.

Initial conditions:

t1(1) = 0, v1(1) = 2 and x1(1) = 0.

Equation: t1(i+1)

Example:

1st iteration t1(2) = 0.01

2nd iteration t1(3) = 0.02

Loop ends at i = 3500

Equation: v1(i+1)

Example:

1st iteration v1(2) = 1.99

2nd iteration v1(3) = 1.9791

Loop ends at i = 3500

Equation: x1(i+1)

Example:

1st iteration x1(2) = 0.02

2nd iteration x1(3) = 0.0399

Loop ends at i = 3500

Plot graph



Figure : Loop



Figure : Part A results

## Part B

In many engineering applications, advection (or convection) and diffusion are the dominant physical transport mechanisms over much of the domain of interest. Consider the case of a plume of contaminant being transported in flowing river. The well-known governing equation for such transport is the advection-diffusion equation below



where c is the concentration of the plume, D is the diffusivity (or diffusion coefficient), v is the velocity the concentration is moving.

• Solve the problems below by generating your own finite difference method and without resorting to any built-in MATLAB solvers.

Initial value problems:

You could imagine this problem as when a plume of contaminant is released at an initial time on a moving bed of fluid.

1. Consider a one-dimensional problem where diffusion effects are omitted. The equation above reduces to

The initial distribution (t = 0) of concentration c0 is a Gaussian i.e.

. On the domain 0 < x < 2, with constant velocity v = 1, the initial profile of c, as well as c at t = 0.5, and 1.

Figure : Finite difference methods (MULJADI, 2019)



Figure 4: Part B, a result when cfl = 0.9

Lax-Wendroff

i = 2:n-1

Example:

1st iteration fnlw (1) = 5.09e-11

2nd iteration fnlw (2) = 3.47e-11

Loop ends at i = 499

Lax-Wendroff

Lax-Friederich

i = 2:n-1

Example:

1st iteration fnlf (1) = 0

2nd iteration fnlf (2) = 1.59e-114

Loop ends at i = 499

First order upwind

i = 2:n

Example:

1st iteration fnlw (1) = 5.09e-11

2nd iteration fnlw (2) = 3.49e-11

Loop ends at i = 499

Time = [0, 0.5, 1]

Subplot



K = 1:nt

Number of plots for methods

i = 2:n

Example:

1st iteration x(i) = 0

2nd iteration x(i) = 0.004

Loop ends at i = 499

Input

Set parameters for xlength =2, n = 500, h = xlength/n-1, x = zeros (1,n), cfl = 0.9, a = 1d0, dt = h\*cfl/a.

i = 1:n

Example:

1st iteration f(i) = 5.09e-11

1st iteration flw(i) = 5.09e-11

1st iteration flf(i) = 0

Loop ends at i = 499

Figure 5: Part B, a loop

1. Now consider the case where the migration of the plume is subject only to diffusion process with D = 1. Still with the same initial condition, plot the concentration profiles at t = 0.5, and 1.



Figure 6: Part B, b results

1. Finally, consider the case where the migration of the plume is subject to both advection, and diffusion. With all the parameters above maintained the same, plot the concentration profiles at t = 0.5, and 1.

Figure 7: Part B, c results

1. Boundary value problems:

(20%) Consider the case where the migration of contaminant is subject to both advection, and diffusion. This time, instead of a given initial plume profile, the concentration (of contaminant) at the inlet boundary is maintained at c(x=0) = 1. Solve the contaminant transport problem and plot the concentration profile along x at t = 0, 0.25, 0.5, 1, 2, 3.



Figure 8: Part B, d results

## Discussion

Part A

Figure 1 is a diagram of the loops in the MATLAB code for part A of the course work. The code starts with the initial input parameters; these input parameters and its corresponding values are used by the MATLAB code equations. The MATLAB code equations generate the next corresponding value (i+1) in the sequence. The equation found in the code is based off the euler method which converts the equation 1 to 2 separate first order and then the shooting method is used as the initial value problem is given. An example is shown where the first and second iteration values are calculated this continues on until iteration (i) reaches 3500. Finally, the values of t1 (time in min) and x1 (delta h) are plotted in the graph where the output is shown in figure 3.

Figure 2 displays the deviation from liquid level of tank from its set point; Kc (controller gain) is varied from 1 to 5 and a behaviour is observed as increasing Kc increases the stability. The time for stability decreases from 25 seconds to 5 seconds. The maximum delta h (amplitude of the wave) also decreases from 0.75 to 0.3 as Kc increase from 1 to 5. The proportional gain K is usually a fixed property of the controller, but in some proportional controllers K is manually adjustable. If K is increased, the sensitivity of the controller to error is increased but the stability is impaired. The system approaches the behaviour of on-off controlled systems and its response becomes oscillatory. As a rule, it is customary to adjust cfl so as to stabilize the system at a state slightly different from the set point. The difference between the measurement delta h and the set point at steady (stable) state is called the offset. It is possible to eliminate the offset by adjusting the bias. This action is called reset. However, adjusting the controller to zero offset under a given set of process conditions would require readjusting every time the load or any other process condition changes. In automatic control, this would be highly problematic. (Berk, 2013)

Part B

Figure 4 shows the concentration profile of a one-dimensional problem where a plume of contaminant is released at t = 0, 0.5 and 1 in a flowing river, only advection occurs and no diffusion. The initial distribution is expressed by the equation at t = 0, this is represented by the blue graph in both Lax-Wendroff and First Order Upwind. The red graph represents t = 0.5 it can be seen on the first order upwind graph is shifted down to 0.7355 and on the Lax-Wendroff it remains at 0.7499 concentration. The yellow graph is the profile at t = 1 and the maximum concentration further drops to 0.7217 on the First Order Upwind and remains constant in the Lax-Wendroff graph. The Lax-Wendroff method does not dampen the solution like the Lax-Friederichs or Upwind methods (that is, there is less dissipation), but it does shift the solution slightly with decreasing values of α (there is dispersion) (Lynch, 2004). The cfl number = 0.9 in this code, cfl should be between 0 and 1. The lower the cfl number the more accurate the graph is as this increases the number of time intervals nt; in this code nt = 277 if the cfl was lowered to 0.5 the nt = 499. A higher nt increases the number of points on the graph hence it is more accurate. However, lowering the cfl number increases the dampening affect on the First Order Upwind and Lax-Friederichs graph for t = 0.5 and 1 as shown in figure 15. Lax-Wendroff method is a second-order difference method in both time and space this considers second order difference unlike First Order Upwind and Lax-Friederichs methods. Using a first order scheme requires a very high resolution and a cfl number close to 1 to get a satisfying solution. A characteristic of the first order schemes is that they are highly diffusive (amplitude decrease). The first order solution will lose amplitude as time passes. The second order schemes give a much higher accuracy on smooth solutions, than the first order schemes (Engineers, 2019). Figure 5 loop has an explanation to the loop in figure 1 which follows a similar method which can be followed in the diagram.

Figure 6 shows the migration of the plume in the river subject only to diffusion. The concentration varies across a distance which starts at 0 m displacement up to 2 m. The blue line graph represents the concentration profile at t = 0.5 s and its maximum concentration is 0.02727 at distance 0.965 m. The red line graph represents the concentration profile at t = 1 s and its maximum concentration is 0.007923 at distance 1 m. Figure 7 shows the migration of the plume in the river subject to diffusion and advection. The concentration varies across a distance which starts at 0 m displacement up to 2 m. The blue line graph represents the concentration profile at t = 0.5 s and its maximum concentration is 0.03169 at distance 1.155 m. The red line graph represents the concentration profile at t = 1 and its maximum concentration is 0.008295 at distance 1.2 m. By comparing figure 6 diffusion with figure 7 diffusion and advection the graph is translated to the right of the graph. In contrast, diffusion refers to the transport of the contaminant plume through the action of random motions. Diffusion works to eliminate sharp discontinuities in concentration and results in smoother, flatter concentration profiles. In the example of contaminant plume in a river, while advection moves the centre of mass of the concentration downstream, diffusion spreads out the concentrated spot of dye to a larger, less concentrated region (HONRATH, 2019).

Figure 8 shows the migration of the plume in the river subject to diffusion and advection with the boundary source (distance = 0 m) is at a constant concentration of 1. There are 6 graphs with different times for t = 0, 0.25, 0.5, 1, 2 and 3. The concentration varies across a distance which starts at 0 m displacement up to 2 m. The graph t = 0 concentration profile is c(x=0) = 1 as stated in the question and then goes to 0 after 0.02 m. When time t increases from 0 to 3 the graphs concentration also increases along the distance from 0 to 2 m where beyond 2 m there is no concentration of plume contaminant.

# References

Berk, Z. (2013). Food Process Engineering and Technology. *Food Science and Technology*, 141-164.

Engineers, N. M. (2019). *Flux limiters*. Retrieved from folk.ntnu: http://folk.ntnu.no/leifh/teaching/tkt4140/.\_main074.html

HONRATH, R. E. (2019). *cee.mtu.edu*. Retrieved from Mass Transport Processes: http://www.cee.mtu.edu/~reh/courses/ce251/251\_notes\_dir/node4.html

Lynch, L. (2004). *Numerical Integration of Linear and Nonlinear Wave Equations.* Florida: Florida Atlantic University.

MULJADI, D. B. (2019). *Advanced Computational Methods (CHEE4004 UNUK) (SPR1 18-19) (H84ACM).* Retrieved from University of Nottingham Moodle : https://moodle.nottingham.ac.uk/course/view.php?id=69105

## Appendix

clear all;

clc;

A = 2; % cross-sectional area (m2)

taf1 = 0.1; % ?1 (min)

h = 0.01; %gap between grid points

t\_final = 35; % final time

N = t\_final/h; % Number of grid points

Kc1 = 1; % (m2/min), proportional gain

% initial condtions:

t1(1) = 0; % time

v1(1) = 2; % velocity

x1(1) = 0; % delta h

for i = 1:N % number of iterations

t1(i+1) = t1(i)+h; % time at an interval

x1(i+1) = x1(i)+h\*(v1(i)); % delta h at an interval

v1(i+1) = v1(i)+h\*((-Kc1\*v1(i)/A)-(Kc1/(taf1\*A))\*x1(i)); %

end

subplot(3,2,1) % Plots graphs

plot(t1,x1,'r')

title('Kc = 1')

xlabel('Time(min)')

ylabel('delta h')

grid on

Figure 9: Part A, a code

clear all;

clc;

A = 2; % cross-sectional area (m2)

taf1 = 0.1; % ?1 (min)

h = 0.01; %gap between grid points

t\_final = 35; % final time

N = t\_final/h; % Number of grid points

Kc1 = 1; % (m2/min), proportional gain

% initial condtions:

t1(1) = 0; % time

v1(1) = 2; % velocity

x1(1) = 0; % delta h

for i = 1:N % number of iterations

t1(i+1) = t1(i)+h; % time at an interval

x1(i+1) = x1(i)+h\*(v1(i)); % delta h at an interval

v1(i+1) = v1(i)+h\*((-Kc1\*v1(i)/A)-(Kc1/(taf1\*A))\*x1(i)); %

end

Kc2 = 2;

t2(1) = 0;

v2(1) = 2;

x2(1) = 0;

for i = 1:N

t2(i+1) = t2(i)+h;

x2(i+1) = x2(i)+h\*(v2(i));

v2(i+1) = v2(i)+h\*((-Kc2\*v2(i)/A)-(Kc2/(taf1\*A))\*x2(i));

end

Kc3 = 3;

t3(1) = 0;

v3(1) = 2;

x3(1) = 0;

for i = 1:N

t3(i+1) = t3(i)+h;

x3(i+1) = x3(i)+h\*(v3(i));

v3(i+1) = v3(i)+h\*((-Kc3\*v3(i)/A)-(Kc3/(taf1\*A))\*x3(i));

end

Kc4 = 4;

t4(1) = 0;

v4(1) = 2;

x4(1) = 0;

for i = 1:N

t4(i+1) = t4(i)+h;

x4(i+1) = x4(i)+h\*(v4(i));

v4(i+1) = v4(i)+h\*((-Kc4\*v4(i)/A)-(Kc4/(taf1\*A))\*x4(i));

end

Kc5 = 5;

t5(1) = 0;

v5(1) = 2;

x5(1) = 0;

for i = 1:N

t5(i+1) = t5(i)+h;

x5(i+1) = x5(i)+h\*(v5(i));

v5(i+1) = v5(i)+h\*((-Kc5\*v5(i)/A)-(Kc5/(taf1\*A))\*x5(i));

end

subplot(3,2,1) % Plots graphs

plot(t1,x1,'r')

title('Kc = 1')

xlabel('Time(min)')

ylabel('delta h')

grid on

subplot(3,2,2)

plot( t2 ,x2, 'g')

title('Kc = 2')

xlabel('Time(min)')

ylabel('delta h')

grid on

subplot(3,2,3)

plot( t3, x3,'b')

title('Kc = 3')

xlabel('Time(min)')

ylabel('delta h')

grid on

subplot(3,2,4)

plot(t4, x4,'y')

title('Kc = 4')

xlabel('Time(min)')

ylabel('delta h')

grid on

subplot(3,2,5)

plot(t5, x5,'k')

title('Kc = 5')

xlabel('Time(min)')

ylabel('delta h')

grid on

Figure 10: Part A, b code

clear all;

clc;

close all;

% numerical grid

xlength = 2; % time length on graph, (x-axis)

n = 500; % number of grid points

h = xlength/(n-1); % gap between grid points

x = zeros(1,n);

% set numerical & physical parameters

cfl = 0.9; % Courant–Friedrichs–Lewy

a = 1d0;

dt = h \* cfl / a; % change in time

for time = [0 0.5 1]; % plume released at these time intervals

x = zeros (1,n);

fn = zeros (1,n);

fnlw = zeros (1,n);

f = zeros (1,n);

flw = zeros (1,n);

fnlf = zeros (1,n);

flf = zeros (1,n);

x(1) = 0D0;

% assigning values to array x(i)

for i=2:n; % 499 iterations

x(i)=x(i-1)+h;

end

for i = 1:n;

f(i) = 0.75\*exp(-((x(i)-0.5)/0.1).^2); % distribution of conc. with time

flw(i) = 0.75\*exp(-((x(i)-0.5)/0.1).^2);

flf(i) = 0.75\*exp(-((x(i)-0.5)/0.1).^2);

end

nt = time/dt; % number of time intervals

for k = 1:nt

for i = 2:n %first order upwind

flux = a \* (f(i)-f(i-1));

fn(i) = f(i)-(dt/h)\*flux;

end

fn(1) = fn(n);

f = fn;

for i = 2:n-1 %Lax-Wendroff

l0 = (dt/(2\*h))\* a \* (flw(i+1)-flw(i-1));

h0 = (dt^2/(2\*h^2))\* a^2 \* (flw(i+1)-(2\*flw(i))+flw(i-1));

fnlw(i) = flw(i) - l0 + h0;

end

fnlw(1) = fn(n);

fnlw(n) = fn(1);

flw = fnlw;

for i = 2:n-1 %Lax-Friederich

fnlf(i) = 0.5\*(flf(i-1)+flf(i+1))-(dt/(2\*h))\*a\*(flf(i+1)-flf(i-1));

end

% Boundary conditions

fnlf(i) = fn(i);

fnlf(n) = fnlf(1)

flf = fnlf;

end

hold on

% graph plot

subplot(3,1,1)

plot(x,f)

title('First Order Upwind')

xlabel('Time (t)')

ylabel('Concentration')

legend('t = 0','t = 0.5','t = 1')

shg

pause(dt)

hold on

subplot(3,1,2)

plot(x,flw)

title('Lax-Wendroff')

xlabel('Time (t)')

ylabel('Concentration')

legend('t = 0','t = 0.5','t = 1')

shg

pause(dt)

hold on

subplot(3,1,3)

plot(x,flf)

title('Lax-Friedrich')

xlabel('Time (t)')

ylabel('Concentration')

legend('t = 0','t = 0.5','t = 1')

shg

pause(dt)

end

Figure 1: Part B, a code

Figure 12: Part B, b code

clear all;

clc;

close all;

%this code solves, and simulates a scalar conservation law equation of

% df(x,t)/dt + a df(x,t)/dx = 0

hold on

%numerical grid

xlength=2; %grid length, upper limit of domain set to 2

n=1000; %number of grid points

h=xlength/(n-1); %gap between grid points

% set numerical and physical parameters

D=1;

U=0;

dt = 0.000001;

x = zeros(1,n);

f = zeros(1,n);

fn = zeros(1,n);

freal= zeros(1,n);

x(1) = 0D0;

% assigning values to array x(i)

for i=2:n

x(i)=x(i-1)+h;

end

% initialising function

for i = 1:n

f(i) = 0.75\*exp(-((x(i)-0.5)/0.1).^2);

end

for time= [0.5:0.5:1] %creates time matrix values with increments of 0.5

nt = time/dt; %find out %number of iterations (nt)

for k = 1:nt

for i = 2:n-1

%Determining difference equation for diffusion-only situation

fn(i) = f(i)-((U\*(dt/(2\*h)))\*(f(i+1)-f(i-1)))+((D\*(dt/(h^2)))\*(f(i+1)-(2\*f(i))+f(i-1)));

end

%boundary condition

fn(1) = fn(n);

fn(n) = fn(1);

f = fn;

end

warning off

plot(x,f) %Plots Distance vs Concentration for diffusion-only situation

grid on

title('Diffusion only')

xlabel('Distance (m)');

ylabel('Concentration (c)')

legend('t = 0.5','t = 1')

end

hold off;

Figure 13: Part B, c code

clear all;

clc;

close all;

%this code solves, and simulates a scalar conservation law equation of

% df(x,t)/dt + a df(x,t)/dx = 0

hold on

%numerical grid

xlength=2; %grid length, upper limit of domain set to 2

n=1000; %number of grid points

h=xlength/(n-1); %gap between grid points

% set numerical and physical parameters

D=1;

U=1;

dt = 0.000001;

x = zeros(1,n);

f = zeros(1,n);

fn = zeros(1,n);

freal= zeros(1,n);

x(1) = 0D0;

% assigning values to array x(i)

for i=2:n

x(i)=x(i-1)+h;

end

% initialising function

for i = 1:n

f(i) = 0.75\*exp(-((x(i)-0.5)/0.1).^2);

end

for time= [0.5:0.5:1] % creates time matrix values with increments of 0.5

nt = time/dt; % find out %number of iterations (nt)

for k = 1:nt

for i = 2:n-1

% Determining difference equation for diffusion-only situation

% flux= D \* (f(i+1)-(2\*f(i))+f(i-1));

fn(i) = f(i)-((U\*(dt/(2\*h)))\*(f(i+1)-f(i-1)))+((D\*(dt/(h^2)))\*(f(i+1)-(2\*f(i))+f(i-1)));

end

% boundary condition

fn(1) = fn(n);

fn(n) = fn(1);

f = fn;

end

warning off

% Plots Distance vs Concentration for diffusion-only situation

plot(x,f)

grid on

title('Advection and Diffusion')

xlabel('Distance (m)');

ylabel('Concentration (c)')

legend('t = 0.5','t = 1')

end

hold off;

clear all; %Clear all previously generated variables

clc; % Close the command window

close all; % Close all previously opened figure windows

xlength = 2; % Grid length

n = 100; % Number of grid points

h = xlength/(n-1); % gap between the grid points

dt = 0.0001;

time = 0;

D = 1;

U = 1;

% Creating matrix of zeros to intiate the code

x=zeros(1,n);

f=zeros(1,n);

fn=zeros(1,n);

x(1)=0D0;

Time=[0;0.25;0.5;1;2;3];%Time matric values

hold on

for J = 1:6

nt = Time(J)/dt;

for i = 2:n

x(i) = x(i-1)+h;

end

% Initiating function

f(1) = 1; % Initial condition ofr the upwind method

for i = 2:n % Initiating iteration loop

f(i) = 0; % Intial condition for the upwind method

end

for k = 1:nt % from one up to "nt", indicating maximum number of iteration

for i = 2:n-1 % creates array of range for n and initiates iteration loop.

% Determing the function of each of mass transport contribution such as:

% Advection and Molecular Diffusion (Ficks Law)

flux1 = D \* (f(i+1)-(2\*f(i))+f(i-1)); % Diffusion Term

flux2 = -U \* (f(i)-f(i-1)); % Advection Term

fn(i) = f(i)+(dt/(h^2))\*flux1+(dt/h)\*flux2; % Assigning difference equation

% Difference equation includes the advection and diffusion contribution

% Studying the net effect of both contribution under boundary condition

end

fn(1) = 1; % Initial boundary condition when the plume is released

fn(n) = 0; % Final boundary condition when the plume is dispersed

f = fn;

% In here we are not making the starting condition the samea as the final

% hence for this reason we dont get symmetrical curves like in part

% (a,b,c)

end

time = nt\*dt;

plot(x(1:n),f(1:n)) % plot graph

end

warning off

grid on

xlabel('Distance (m)');

ylabel('Concentration (c)')

legend('t = 0','t = 0.25','t = 0.5','t = 1','t = 2','t = 3')

title('Advection and Diffusion with Boundary Condition')

hold off

Figure 14: Part B, d code



Figure 15: Part B, a result when cfl = 0.5