

Date: 4/14/2023

## **2D Advection + Diffusion Finite Element analysis**

Professor Bruno Blais

Polytechnique Montréal

Writer: Ahmad Eliyon

### Contents

<b>1</b>	<b>Problem. Advection diffusion equation .....</b>	<b>2</b>
1.1	Introduction.....	2
1.2	Assumptions .....	2
1.3	2D advection-diffusion equation .....	2
<b>2</b>	<b>Solution.....</b>	<b>5</b>
<b>3</b>	<b>Result .....</b>	<b>6</b>
<b>4</b>	<b>Code of MATLAB.....</b>	<b>7</b>

# 1 Problem. Advection diffusion equation

## 1.1 Introduction

Numerical methods is used to calculate reactions, mass transfer, heat transfer and fluid dynamics in chemical engineering. Nowadays, with introduction of super computers, simulation methods saved much time and money as well less laboratory failure is recording. Finite Element methods is one of the best numerical methods that its reliability corroborated a lot of CFD programs such as ANSYS and so on. Furthermore, biological experiments are coupled with numerical methods. Hence, in this report a very simple version 2D advection diffusion equation is solved.

## 1.2 Assumptions

To calculate concentration distribution of a material in a square area some assumption considered.

- Steady state
- 2D dimension
- Properties are constant including mass transfer coefficient
- Constant velocity
- Boundary conditions

$$@ (x,y) \begin{cases} x = 0; & c = c_0 \\ y = 0; & c = 0.01 \\ y = 1; & c = 0.01 \\ x = 1; & c = \frac{c_0}{2} \end{cases}$$

## 1.3 2D advection-diffusion equation

$$u_x \frac{\partial c}{\partial x} + u_y \frac{\partial c}{\partial y} = D \left( \frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right)$$

$$I = \oint\oint w \left[ u_x \frac{\partial c}{\partial x} + u_y \frac{\partial c}{\partial y} \right] dx dy - \oint\oint w \left[ D \left( \frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) \right] dx dy = 0 \quad (1)$$

$$\left\{ \begin{aligned} \oint\oint u_x w \frac{\partial c}{\partial x} dx dy &= u_x \oint c w dy - u_x \iint \frac{\partial w}{\partial x} \frac{\partial c}{\partial x} dx dy = \dots = u_x \left[ - \iint \frac{\partial w}{\partial x} \frac{\partial c}{\partial x} dx dy + \oint \left[ w \frac{\partial c}{\partial x} \right] n_x d\Gamma \right] \\ \oint\oint u_y w \frac{\partial c}{\partial y} dx dy &= u_y \oint c w dx - u_y \iint \frac{\partial w}{\partial y} \frac{\partial c}{\partial y} dx dy = \dots = u_y \left[ - \iint \frac{\partial w}{\partial y} \frac{\partial c}{\partial y} dx dy + \oint \left[ w \frac{\partial c}{\partial y} \right] n_y d\Gamma \right] \\ \oint\oint w \left[ D \left( \frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) \right] dx dy &= - \iint D \left[ \frac{\partial w}{\partial x} \frac{\partial c}{\partial x} + \frac{\partial w}{\partial y} \frac{\partial c}{\partial y} \right] dx dy + \oint w \left[ \frac{\partial c}{\partial x} n_x + \frac{\partial c}{\partial y} n_y \right] d\Gamma = - \iint D \left[ \frac{\partial w}{\partial x} \frac{\partial c}{\partial x} + \frac{\partial w}{\partial y} \frac{\partial c}{\partial y} \right] dx dy + \oint w \left[ \frac{\partial c}{\partial n} \right] d\Gamma \end{aligned} \right.$$

Implementing in (1)

Green theorem 
$$I = - \oint\oint \left[ u_x \frac{\partial w}{\partial x} \frac{\partial c}{\partial x} + u_y \frac{\partial w}{\partial y} \frac{\partial c}{\partial y} \right] dx dy + \iint D \left[ \frac{\partial w}{\partial x} \frac{\partial c}{\partial x} + \frac{\partial w}{\partial y} \frac{\partial c}{\partial y} \right] dx dy = 0 \quad (2) \text{ weak form}$$

Assumptions

For c and w

$$\frac{\partial w_1}{\partial x} = -\frac{1}{4bc}(c-y), \frac{\partial w_1}{\partial y} = \frac{1}{4bc}(b-x), \frac{\partial w_2}{\partial x} = \frac{1}{4bc}(c-y), \frac{\partial w_2}{\partial y} = -\frac{1}{4bc}(b+x), \frac{\partial w_3}{\partial x} = \frac{1}{4bc}(c+y),$$

$$\frac{\partial w_3}{\partial y} = \frac{1}{4bc}(b+x), \frac{\partial w_4}{\partial x} = -\frac{1}{4bc}(c+y), \frac{\partial w_4}{\partial y} = \frac{1}{4bc}(b-x)$$

Then, eq. (2) can be shorted in this form of Matrixes:

$$I = -\oint \left[ u_x \begin{bmatrix} \frac{\partial w_1}{\partial x} \\ \frac{\partial w_2}{\partial x} \\ \frac{\partial w_3}{\partial x} \\ \frac{\partial w_4}{\partial x} \end{bmatrix} \left| \frac{\partial c_1}{\partial x} \frac{\partial c_2}{\partial x} \frac{\partial c_3}{\partial x} \frac{\partial c_4}{\partial x} \right| + u_y \begin{bmatrix} \frac{\partial w_1}{\partial y} \\ \frac{\partial w_2}{\partial y} \\ \frac{\partial w_3}{\partial y} \\ \frac{\partial w_4}{\partial y} \end{bmatrix} \left| \frac{\partial c_1}{\partial y} \frac{\partial c_2}{\partial y} \frac{\partial c_3}{\partial y} \frac{\partial c_4}{\partial y} \right| \right] dx dy$$

$$+ \iint D \left[ \begin{bmatrix} \frac{\partial w_1}{\partial x} \\ \frac{\partial w_2}{\partial x} \\ \frac{\partial w_3}{\partial x} \\ \frac{\partial w_4}{\partial x} \end{bmatrix} \left| \frac{\partial c_1}{\partial x} \frac{\partial c_2}{\partial x} \frac{\partial c_3}{\partial x} \frac{\partial c_4}{\partial x} \right| + \begin{bmatrix} \frac{\partial w_1}{\partial y} \\ \frac{\partial w_2}{\partial y} \\ \frac{\partial w_3}{\partial y} \\ \frac{\partial w_4}{\partial y} \end{bmatrix} \left| \frac{\partial c_1}{\partial y} \frac{\partial c_2}{\partial y} \frac{\partial c_3}{\partial y} \frac{\partial c_4}{\partial y} \right| \right] dx dy$$

$$= \sum_{i=0}^{nodes} \sum u_x \tilde{w}_x \check{c} + u_y \tilde{w}_y \check{c} + D \sum_{i=0}^{nodes} \sum \tilde{w}_x \check{c} + \tilde{w}_y \check{c} = 0$$

So, this equation shall be validated by each element. For calculation, MATLAB is used.

$$\left| -u_x \begin{pmatrix} \tilde{w}_{x_1}^{element 1^{st}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \tilde{w}_{x_n}^{element n^{th}} \end{pmatrix} - u_y \begin{pmatrix} \tilde{w}_{y_1}^{element 1^{st}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \tilde{w}_{y_n}^{element n^{th}} \end{pmatrix} + D \begin{pmatrix} \tilde{w}_{x_1}^{element 1^{st}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \tilde{w}_{x_n}^{element n^{th}} \end{pmatrix} \right|$$

$$+ D \begin{pmatrix} \tilde{w}_{y_1}^{element 1^{st}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \tilde{w}_{y_n}^{element n^{th}} \end{pmatrix} \left| \begin{matrix} c_1^{element 1^{st}} \\ \vdots \\ c_n^{element n^{th}} \end{matrix} \right| = \begin{vmatrix} 0 \\ \vdots \\ 0 \end{vmatrix}$$

Briefly, due to simplification of boundary, the overall equation is written in this form:

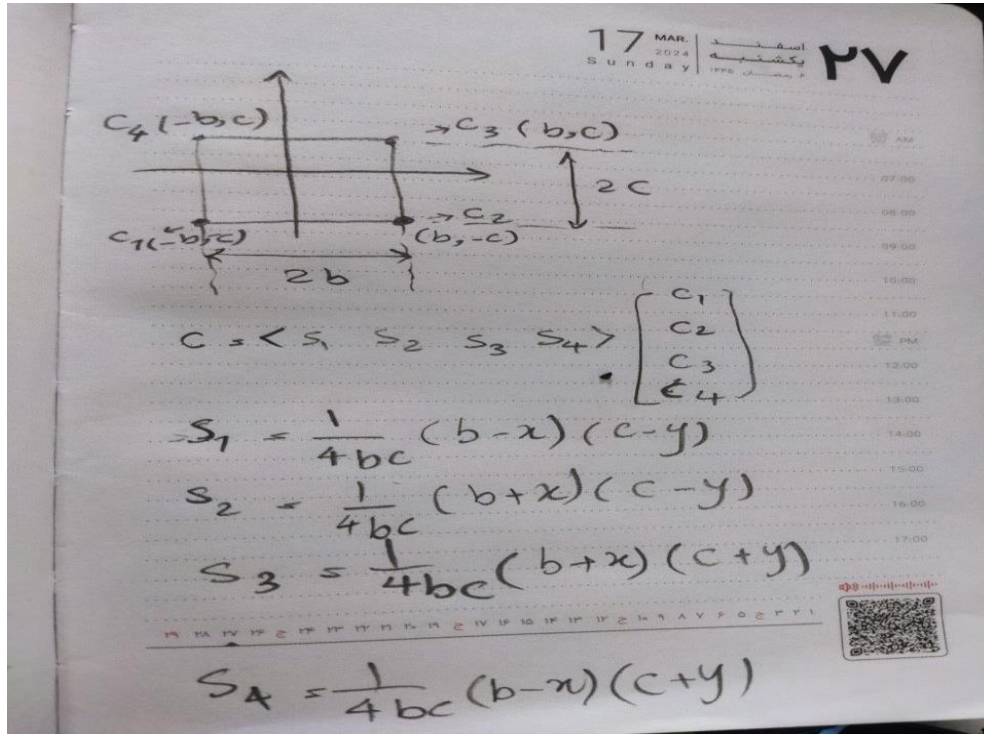
$$I = \tilde{w}_{global} \check{C} = 0$$

Where  $\tilde{w}_{global}$  is a Matrix from accumulation of local matrix of elements;

$\tilde{w}_{global}^{element\ n^{th}}_{4 \times 4}$

$$= \begin{bmatrix} -\frac{(u_x - D)b^2 + (u_y - D)c^2}{3bc} & -\frac{(u_x - D)b^2 - 2(u_y - D)c^2}{6bc} & \frac{(u_x - D)b^2 + (u_y - D)c^2}{6bc} & -\frac{(u_x - D)c^2 - 2(u_y - D)b^2}{6bc} \\ \frac{(u_x - D)b^2 - 2(u_y - D)c^2}{6bc} & -\frac{(u_x - D)b^2 + (u_y - D)c^2}{3bc} & -\frac{(u_x - D)c^2 - 2(u_y - D)b^2}{6bc} & \frac{(u_x - D)b^2 + (u_y - D)c^2}{6bc} \\ \frac{(u_x - D)b^2 + (u_y - D)c^2}{6bc} & \frac{(u_x - D)c^2 - 2(u_y - D)b^2}{6bc} & -\frac{(u_x - D)b^2 + (u_y - D)c^2}{3bc} & -\frac{(u_x - D)b^2 - 2(u_y - D)c^2}{6bc} \\ -\frac{(u_x - D)c^2 - 2(u_y - D)b^2}{6bc} & \frac{(u_x - D)b^2 + (u_y - D)c^2}{6bc} & \frac{(u_x - D)b^2 - 2(u_y - D)c^2}{6bc} & -\frac{(u_x - D)b^2 + (u_y - D)c^2}{3bc} \end{bmatrix}$$

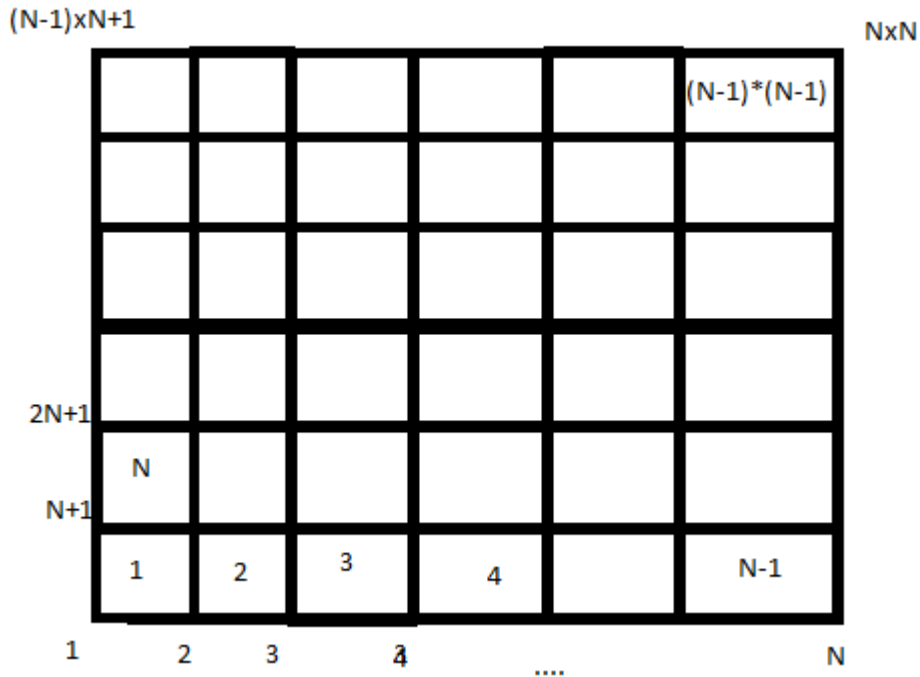
For calculating each element concentration this formula is considered, 2<sup>nd</sup> order:



## 2 Solution

Following steps is considered in this work:

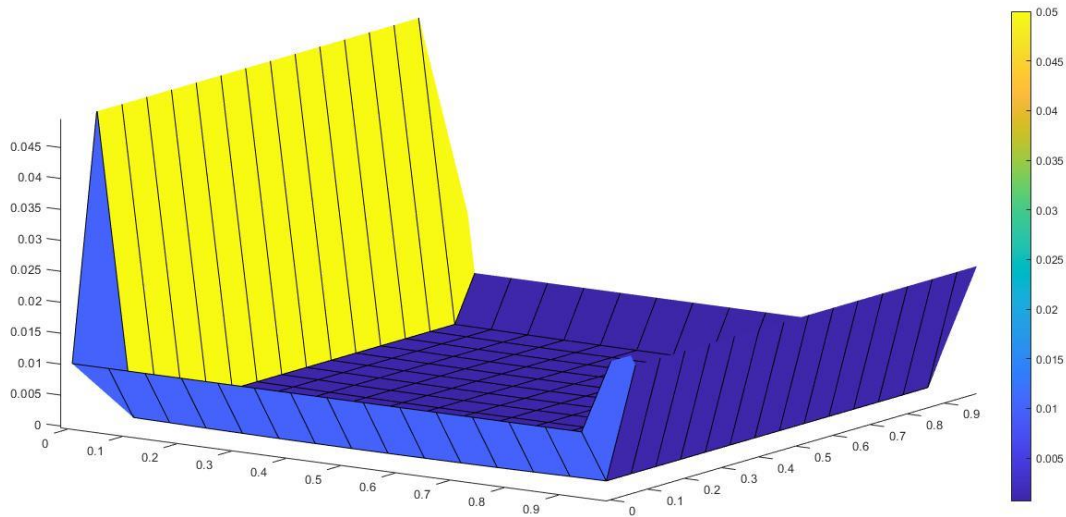
- Collect Mesh Information  
In this work, Quadrilateral method is applied.



- Connectivity information  
The area is continuous and solid. Consist of  $R^2$
- Coordinate information  
xy-coordination , without any transformation, is applied.
- Boundary information  
All of boundary were considered in Dirichlet boundary.
- Look over all elements  
All of calculation was performed by MATLAB.
- Solve system of algebraic equations  
While loop with tolerance  $<0.0001$  considered until numerical calculation converged.
- Visualize the result

### 3 Result and discussion

The result of calculation is depicted in a 3D-plot figure. Overall, about validations of result cannot be confirmed as the concentration distribution is not logical. The solution shows 7 steps



from governing equation to manifesting the result. The real challenge of this is that resources about applying FEM for mass transfer problem is quite rare or is not accessible in Iran. Complexity of solution can be elaborated by adding Neumann or Robin boundary condition- in equation shall be interfaced, unsteady state- finite difference methods for calculating changes in time, triangulation, more realistic frame, distributed velocity- in this case fluid velocity equations shall be solved simultaneously, Gauss, physical properties and etc. Regarding concentration, inner nodes are calculated roughly zero, even though initial guess was 1.0 for each node.

Notes:

- 1- The figure is not logical, however it could be better if mesh size was finer.

## 4 Code of MATLAB

```
clc
close all
clear all
% An Advection and diffusion equation is solved by this code
%  $u_x + c/x + u_y + c/y = D((\partial^2 c)/(\partial x^2) + (\partial^2 c)/(\partial y^2))$ 
% Quadrilateral method

%% enter the number of element in each side ( square shape is considered)
N1=input('enter number of elements in each row');

N=power(N1,2);
%% the number of nodes is calculated
node_num=power(N1+1,2);

%% elem matrix is the definition of each element according to it's nodes
elem=ones(N,4);
dx=1/N1;
for i=1:N
    for j=1:4
        if j==1
            if mod(i,N1)==0
                elem(i,j)=floor(i/(N1))+i-1;
            else
                elem(i,j)=floor(i/(N1))+i;
            end
        elseif j==2
            elem(i,j)=elem(i,j-1)+1;
        elseif j==3
            elem(i,j)=elem(i,j-1)+N1;
        else
            elem(i,j)=elem(i,j-1)+1;
        end
    end
end
%% node matrix is the location of each element according to the xy-
coordination
node=zeros(node_num,2);

for i=1:node_num
    for j=1:2
        if j==1
            if mod(i,N1+1)==0
                node(i,j)=1;
            else
                node(i,j)=(mod(i,(N1+1))-1)*1/N1;
            end
        elseif j==2
            if mod(i,N1+1)==0
                node(i,j)=(floor(i/(N1+1))-1)*1/N1;
            else
                node(i,j)=floor(i/(N1+1))*1/N1;
            end
        end
    end
end
```

```

        end

    end

end

%% bdFlag matrix is the state of boundary condition each element;
% _ 0 for non-boundary sides;
% _ 1 for the first type, i.e., Dirichlet boundary;
% _ 2 for the second type, i.e., Neumann boundary;
% _ 3 for the third type, i.e., Robin boundary
bdFlag=zeros(N1+1,4);
for i=1:N
    for j=1:4
        if j==1
            if i/(N1)<=1
                bdFlag(i,j)=1;
            end
        elseif j==2
            if mod(i,N1)==0
                bdFlag(i,j)=1;
            end
        elseif j==3
            if i/(N1)>N1-1
                bdFlag(i,j)=1;
            end
        else
            if mod(i,N1)==1
                bdFlag(i,j)=1;
            end
        end
    end
end

end

%% % % Calculation
% constants
D= 8*10^-5;    %mass transfer coefficient
u_x=10;        %x-vector of velocity
u_y=10;        %y-vector of velocity
c0= .05;       % saturated concentration in Dirichlet boundary

% % matrix of global weight

% % % % % Matrix of local weight

b=dx/2;
c=dx/2;
w44=zeros(4,4);

for i=1:4
    w44(i,i)=-((u_x-D)*b^2+(u_y-D)*c^2)/(3*b*c);
end
w44(1,2)=-((u_x-D)*b^2-2*(u_y-D)*c^2)/(6*b*c);
w44(1,3)=((u_x-D)*b^2+(u_y-D)*c^2)/(6*b*c);
w44(1,4)=-((u_x-D)*c^2-2*(u_y-D)*b^2)/(6*b*c);
w44(3,4)=w44(1,2);
w44(2,3)=w44(1,4);

```



```

w44(2,4)=w44(1,3);
w44=w44+triu(w44,1)';

%% % % % % Matrix of global wieght
wglobe=zeros(node_num);
for i=1:node_num-3
    for j=1:4
        for k=1:4
            wglobe(i+j-1,i+k-1)=wglobe(i+j-1,i+k-1)+w44(j,k);
        end
    end
end

%% Matrix of Concentration
C=ones(node_num,1);
[a,b]=find(node(:,1)==0);
[aa,bb]=find(node(:,2)==0);
[aaa,bbb]=find(node(:,2)==1);
[aaaa,bbbb]=find(node(:,1)==1);
for i=a
    C(i,1)=c0;
end

for i=aa
    C(i,1)=.01;
end
for i=aaa
    C(i,1)=.01500;
end
for i=aaaa
    C(i,1)=c0/2;
end
node1=node;
for i=1: length(a)
wglobe(:,a(i))=0;
    wglobe(a(i),:)=0;
    wglobe(a(i),a(i))=1;
    wglobe(:,aa(i))=0;
    wglobe(aa(i),:)=0;
    wglobe(aa(i),aa(i))=1;
    wglobe(:,aaa(i))=0;
    wglobe(aaa(i),:)=0;
    wglobe(aaa(i),aaa(i))=1;
    wglobe(:,aaaa(i))=0;
    wglobe(aaaa(i),:)=0;
    wglobe(aaaa(i),aaaa(i))=1;
end

%% slover
CC=linsolve(wglobe,C);

%% Calculation for concentration of elements
AAA = [a; aa;aaa;aaaa];
AAA=unique(AAA);
A(:,1)=1:node_num;
AA=setdiff(A,AAA);

```

```

C_elem=zeros(N,1);
error=2;
Tol=0.0001;
counter=0;
C1=C;
while sum(abs(error))>Tol
CC=linsolve(wglobe,C1);

for i=1:N
    S(i,1)=1/(4*b*c)*(b-node(elem(i,1),1))*(c-node(elem(i,1),2));
    S(i,2)=1/(4*b*c)*(b+node(elem(i,2),1))*(c-node(elem(i,2),2));
    S(i,3)=1/(4*b*c)*(b+node(elem(i,3),1))*(c+node(elem(i,3),2));
    S(i,4)=1/(4*b*c)*(b-node(elem(i,4),1))*(c+node(elem(i,4),2));
    C_elem(i,1)=S(i,:)[CC(elem(i,1)); CC(elem(i,2)); CC(elem(i,3));
CC(elem(i,4))];
end

for i=1:length(AA)

    [tt,b]=find(elem(:,')==AA(i));
for j=1:length(tt)
    C1(AA(i),1)=CC(AA(i),1)+1/length(tt)*C_elem(j,1);

end
end
error=C1-CC;

counter=counter+1;
end

%% Printing figures
%%3D figure
for i=1:N1+1
    C1(1:N1+1,i)=C1((i-1)*(N1+1)+1:(i-1)*(N1+1)+N1+1,1);

end
C1(N1+2:node_num,:)=[];
figure(1)
hold on
[X,Y] = meshgrid(0:1/(N1):1);
surf(Y,X,C1)

hold off

```