To govern oil and water flow rate equations in the porous media, mass balance and Darcy equations are solved to consider the effect of each parameter. The equilibrium mass balance equations for each phase are derived as below;

$$-\frac{\partial}{\partial x}(\rho_m u_m) = \frac{\partial}{\partial t}(\rho_m \varphi S_m) \quad \text{(Eq.1)}$$

$$-\frac{\partial}{\partial x}(\rho_o u_o) = \frac{\partial}{\partial t}(\rho_o \varphi S_o) \quad \text{(Eq.2)}$$

The value of ρ can be replaced by the definition of the formation volume coefficient. Therefore, ρ could be derived as:

$$\rho_o = \frac{\rho_{osc}}{B_o} \quad \text{(Eq.3)}$$

$$\rho_m = \frac{\rho_{msc}}{B_m} \quad \text{(Eq.4)}$$

Where, ρ_{osc} and ρ_{msc} are the density of oil and drilling in the surface conditions. By replacing the equations 3 and 4 in Eq.1 and 2, they were derived as the following equations;

$$-\frac{\partial}{\partial x}\left(\frac{u_m}{B_m}\right) = \frac{\partial}{\partial t}\left(\frac{\varphi S_m}{B_m}\right) \quad \text{(Eq.5)}$$

$$-\frac{\partial}{\partial x}\left(\frac{u_o}{B_o}\right) = \frac{\partial}{\partial t}\left(\frac{\varphi S_o}{B_o}\right) \quad \text{(Eq.6)}$$

The velocity in the continuum correlation should be substituted in the Darcy equation in the porous medium, and it was written for each fluid as below;

$$u_o = -\frac{K k_{ro}}{\mu_o}\frac{\partial P_o}{\partial x} \quad \text{(Eq.7)}$$

$$u_m = -\frac{K k_{rm}}{\mu_m}\frac{\partial P_m}{\partial x} \quad \text{(Eq.8)}$$

By replacing equations 7 and 8 in equations 5 and 6, it is concluded that;

$$\frac{\partial}{\partial x}\left(\frac{Kk_{ro}}{B_o\mu_o}\frac{\partial P_o}{\partial x}\right)=\frac{\partial}{\partial t}\left(\frac{\varphi S_o}{B_o}\right) \quad \text{(Eq.9)}$$

-

$$\frac{\partial}{\partial x}\left(\frac{Kk_{rm}}{B_m\mu_m}\frac{\partial P_m}{\partial x}\right)=\frac{\partial}{\partial t}\left(\frac{\varphi S_m}{B_m}\right) \quad \text{(Eq.10)}$$

The above equations can be rearranged by definition of mobility as;

$$\frac{\partial}{\partial x}\left(K\lambda_o\frac{\partial P_o}{\partial x}\right)=\frac{\partial}{\partial t}\left(\frac{\varphi S_o}{B_o}\right) \quad \text{(Eq.11)}$$

$$\frac{\partial}{\partial x}\left(K\lambda_m\frac{\partial P_m}{\partial x}\right)=\frac{\partial}{\partial t}\left(\frac{\varphi S_m}{B_m}\right) \quad \text{(Eq.12)}$$

Where

Boundary conditions in the interior side of the core are defined as below;

$$\begin{cases} S_o = 1 - S_{wc} \\ P_o = P_i \\ P_m = P_i - P_c(S_{wc}) \end{cases} \quad \text{(Eq.13)}$$

When those side of the core which is contacted with the drilling mud, the pressure is equal to the pressure of the drilling mud, and therefore the water saturation reaches its maximum value and eq.14 derived as;

$$\begin{cases} S_m = 1 - S_{or} \\ P_o = P_{wellbore} \\ P_m = P_{wellbore} \end{cases} \quad \text{(Eq.14)}$$

Before the commencement of drilling operations and contacting the drilling mud with the formation, the formation pressure is equal to the reservoir pressure as below;

$$\begin{cases} P_{oi} = P_i \\ P_{mi} = P_{oi} - P_c(S_{oi}) \end{cases} \quad \text{(Eq.15)}$$

-

Governing equations for the discrete formulation

In this paper, the finite difference method is used to solve numerical equations of partial derivatives. The following equations are needed to be solved;

$$\begin{cases} S_o + S_m = 1 \\ P_c = P_o - P_m \end{cases} \quad \text{(Eq.16)}$$

$$\frac{\partial}{\partial x}\left(K\lambda_m \frac{\partial P_m}{\partial x}\right)_i \cong T_{m,i+\frac{1}{2}}\left(P_{m,i+1} - P_{m,i}\right) + T_{m,i-\frac{1}{2}}\left(P_{m,i-1} - P_{m,i}\right) \quad \text{(Eq.17)}$$

Where;

$$T_{m,i\pm\frac{1}{2}} = \frac{\lambda_{m,i\pm\frac{1}{2}} K_{i\pm\frac{1}{2}}}{\Delta x^2} \quad \text{(Eq.18)}$$

In Eq.19, $K_{i\pm\frac{1}{2}}$ is obtained from the harmonic average. The following equation can be used to calculate $K_{i\pm\frac{1}{2}}$;

$$K_{i\pm\frac{1}{2}} = \frac{2K_{i\pm1}K_i}{K_{i\pm1} + K_i} \quad \text{(Eq.19)}$$

The average mobility parameter can be calculated based on the upstream average correlation;

$$\lambda_{m,i\pm\frac{1}{2}} = \begin{cases} \lambda_{m,i\pm1} & if\ P_{m,i\pm1} \geq P_{m,i} \\ \lambda_{m,i} & if\ P_{m,i\pm1} < P_{m,i} \end{cases} \quad \text{(Eq.20)}$$

The right-hand side of the drilling mud flow equation (Eq.12) can be rearranged as follows;

$$\frac{\partial}{\partial t}\left(\frac{\varphi S_m}{B_m}\right) = \varphi S_m \frac{\partial\left(\frac{1}{B_m}\right)}{\partial P_m}\frac{\partial P_m}{\partial t} + \frac{S_m}{B_m}\frac{\partial \varphi}{\partial P_m}\frac{\partial P_m}{\partial t} + \frac{\varphi}{B_m}\frac{\partial S_m}{\partial t} \quad \text{(Eq.21)}$$

at this stage, we can use the definition of rock compressibility;

$$C_r = \frac{1}{\varphi}\frac{\partial \varphi}{\partial P} \quad \text{(Eq.24)}$$

By substituting equation 15 in Eq.24 with the definition of capillary pressure, the following equation is derived;

$$\frac{\partial}{\partial t}\left(\frac{\varphi S_m}{B_m}\right)_i \cong \frac{\varphi S_{m,i}}{\Delta t}\left[\frac{\partial\left(\frac{1}{B_m}\right)}{\partial P_m}+\frac{C_r}{B_m}\right]\left(P_{o,i}-{P_{o,i}}^t\right) + \left[\frac{\varphi}{B_m \Delta t}-\left[\frac{\partial\left(\frac{1}{B_m}\right)}{\partial P_m}+\frac{C_r}{B_m}\right]\frac{\partial P_c}{\partial S_m}\right]\left(S_{m,i}-{S_{m,i}}^t\right) \quad \text{(Eq.22)}$$

As a result, the discrete form of the equation of drilling mud is as follows;

$$T_{m,i+\frac{1}{2}}\left(P_{o,i+1}-P_{o,i}\right)+T_{m,i-\frac{1}{2}}\left(P_{o,i-1}-P_{o,i}\right)-T_{m,i+\frac{1}{2}}\left(P_{c,i+1}-P_{c,i}\right) - T_{m,i-\frac{1}{2}}\left(P_{c,i-1}-P_{c,i}\right)=C_{pom,i}\left(P_{o,i}-{P_{o,i}}^t\right)+C_{smm,i}\left(S_{m,i}-{S_{m,i}}^t\right) \quad \text{(Eq.23)}$$

Based on the finite difference method, the approximation of the left-hand side of the drilling mud flow equation is as follows;

$$\frac{\partial}{\partial x}\left(K\lambda_\alpha \frac{\partial P_\alpha}{\partial x}\right)_N \cong \frac{(K\lambda_\alpha)_{N-\frac{1}{2}}\left(\frac{P_{\alpha,N-1}-P_{\alpha,N}}{\Delta x}\right)-(K\lambda_\alpha)_{N+\frac{1}{2}}\left(\frac{8P_{\alpha,N+\frac{1}{2}}-9P_{\alpha,N}+P_{\alpha,N-1}}{3\Delta x}\right)}{\Delta x} \quad \text{(Eq.24)}$$

The Matlab codes that are used in this paper are as follows;

```
function Main(poro,krw0,K,muo,dp,b,swi,tinhr,isexpplot,str_plot)
% clc;
% clear;
% close all;
% Geometic Properties
Lx = 4e-2;  % ---> m
% Lx = 200;
D = sqrt(10.74*4/pi)*1e-2;  % ---> m


Ly = D;
Lz = pi*D^2/4/D;     % For equivalet cross section to circle;
nx = 50;
dxc = Lx/nx;
dx = dxc*ones(nx,1);
 n = nx;
 n_var = 0;
for ii = 1:nx
    n_var = n_var +1;
    Ax(n_var,1) = Ly*Lz; % ---> m^2
end
 poro = poro*ones(n,1);
% K = input('Enter the absolute perm. (md) : ');
K = K*9.87e-16*ones(n,1);    % ---> m^2
     Cr = 1e-9;              % ---> 1/pa
% Fluid Properties
% muo = input('Enter the oil viscosity : ');
muo = muo*1e-3;
```

```
muw = 0.92e-3;

Bo = 1;

Bw = 1;

swc = 0.12;

sor = 0.20;

nw = 2;

no = 2;

% krw0 = 0.12;

kro0 = 0.65;

f_krw = @(S) krw0*((S-swc)./(1-sor-swc)).^nw;

f_kro = @(S) kro0*(1 - (S-swc)./(1-sor-swc)).^no;

% b = input('Enter the Capillary Parameter : ');

b= b*6894.757; %--->pa

f_pc = @(S) pcm_fun(S,sor,swc,b);

Co = 1e-8;              % ---> 1/pa

Cw = 1e-8;              % ---> 1/pa

% Initialization

% dp = input('Enter the dp = Psand - Pmud in psi : ');

Pbhinjw = 0*6894.757;

Pbhinjo = dp*6894.757 + Pbhinjw;

p_i = Pbhinjo;              % ---> initial pressure(pa)

 % swi = input('Enter the initial water saturation : ');

Sw = swi*ones(n,1);

Po = p_i*ones(n,1);

Pw = p_i*ones(n,1) - f_pc(Sw);

vars = [Po;Sw];

Pc = f_pc(Sw);

% Simulation
```

```
% tinhr = input('Enter the Final Time in hour : ');
tFinal = tinhr*3600;          % ---> second
beta = 1.2;
dt = 1e-8;
dtmax = 1e1;
dSwmax = 1;
tSimulation = 0;
% Start Mail Loop of Time
counter_time_step = 1;


while tSimulation<tFinal


   newvars =
IMPES_Solver([Po(:,counter_time_step);Sw(:,counter_time_step)],nx,dx,Ax,Lx,Ly,Lz,dt,poro,K,f_kro,f_
krw,Bo,Bw,Co,Cw,Cr,muo,muw...
                  ,Pbhinjo,Pbhinjw,f_pc,sor,swc);
   max_Sw = max(abs(newvars(n+1:2*n,1) - Sw(:,counter_time_step)));
   if max_Sw>dSwmax
      dt = dt/beta;
      disp('Time step cut');
   else
      tSimulation = tSimulation+dt;
      if dt<=dtmax
         if abs(tFinal-tSimulation)<=beta*dt
            dt = (tFinal-tSimulation);
         else
            if (dtmax-dt*beta)<=0
               dt = dtmax;
            else
```

```
            dt = dt*beta;
        end
    end
end
counter_time_step = counter_time_step+1;
vars(:,counter_time_step) = newvars;
Po(:,counter_time_step) = newvars(1:n,1);

Sw(:,counter_time_step) = newvars(n+1:2*n,1);

%       Pw(:,counter_time_step) = newvars(2*n+1:3*n,1);

for ii = 1:n
    if Sw(ii,counter_time_step) < swc
        Sw(ii,counter_time_step) = swc;
    elseif Sw(ii,counter_time_step) > 1-sor
        Sw(ii,counter_time_step) = 1- sor;
    end
end
if ~isreal(Sw(:,counter_time_step))
    ii;
end
Pc(:,counter_time_step) = f_pc(Sw(:,counter_time_step));
Simulation_Time(counter_time_step,1) = tSimulation;
WC(counter_time_step) = (f_krw(Sw(1,counter_time_step))/muw/Bw) /
(f_krw(Sw(1,counter_time_step))/muw/Bw + f_kro(Sw(1,counter_time_step))/muo/Bo);
end
fprintf(' Simulation Time is %8.6e\n',tSimulation);
1;
```

```
end
% Post Processing
Swp = Sw(:,end);


hold on;
plot(1-(dxc/2:dxc:Lx)/Lx,Swp,str_plot,'LineWidth',2);
xlabel('\bf Dimensionless Length (x/L)');
ylabel('\bf Filterated Mud Saturation');
if isexpplot
   data = xlsread('expf.xls');
   plot(data(:,1),data(:,2),'rs-','LineWidth',2,'MarkerFaceColor','b');
end
grid on;


function out =
IMPES_Solver3(xn,nx,dx,Ax,Lx,Ly,Lz,dt,poro,K,f_kro,f_krw,Bo,Bw,Co,Cw,Cr,muo,muw...
   ,Pbhinjo,Pbhinjw,f_pc,sor,swc)
n = length(xn)/2;



Po = xn(1:n);
Sw = xn(n+1:2*n);


Pc = f_pc(Sw);
Pw = Po - Pc;


% Pc = Po - Pw;


Tor = zeros(n,1);
```

```
Tol = zeros(n,1);

Twr = zeros(n,1);

Twl = zeros(n,1);



Cop = zeros(n,1);

Cow = zeros(n,1);

Cwp = zeros(n,1);

Cww = zeros(n,1);


n_var = 0;

for ii = 1:nx

   n_var = n_var + 1;

   Vb = dx(ii)*Ly*Lz;

   if ii == 1

      % right oil

      K_r = K(n_var+1);

      Axr = Ax(n_var+1);

      if Po(n_var+1)>=Po(n_var)

         mobilityo_r_up = f_kro(Sw(n_var+1))/muo/Bo;

      else

         mobilityo_r_up = f_kro(Sw(n_var))/muo/Bo;

      end

      Tor(n_var,1) =
(2*Ax(n_var)*K(n_var)*Axr*K_r)/(Ax(n_var)*K(n_var)*dx(ii)+Axr*K_r*dx(ii+1))*mobilityo_r_up;

      % right water

      K_r = K(n_var+1);

      Axr = Ax(n_var+1);

      if Pw(n_var+1)>=Pw(n_var)
```

```
        mobilityw_r_up = f_krw(Sw(n_var+1))/muw/Bw;
    else
        mobilityw_r_up = f_krw(Sw(n_var))/muw/Bw;
    end
    Twr(n_var,1) =
(2*Ax(n_var)*K(n_var)*Axr*K_r)/(Ax(n_var)*K(n_var)*dx(ii)+Axr*K_r*dx(ii+1))*mobilityw_r_up;

    % left oil
    K_l = K(n_var);
    Axl = Ax(n_var);
    if Pbhinjo>=Po(n_var)
        mobilityo_l_up = f_kro(swc)/muo/Bo;
    else
        mobilityo_l_up = f_kro(Sw(n_var))/muo/Bo;
    end
    Tol(n_var,1) = Ax(n_var)*K(n_var)*mobilityo_l_up; % --> Caution: for consant K and A
    % left water
    K_l = K(n_var);
    Axl = Ax(n_var);
    if (Pbhinjo-f_pc(swc))>=Pw(n_var)
        mobilityw_l_up = f_krw(swc)/muw/Bw;
    else
        mobilityw_l_up = f_krw(Sw(n_var))/muw/Bw;
    end
    Twl(n_var,1) = Ax(n_var)*K(n_var)*mobilityw_l_up; % --> Caution: for consant K and A
elseif ii == nx
    % right oil
    K_r = K(n_var);
    Axr = Ax(n_var);
```

```
if Pbhinjw>=Po(n_var)
   mobilityo_r_up = f_kro(1-sor)/muo/Bo;
else
   mobilityo_r_up = f_kro(Sw(n_var))/muo/Bo;
end
Tor(n_var,1) = Ax(n_var)*K(n_var)*mobilityo_r_up; % --> Caution: for consant K and A
% right water
K_r = K(n_var);
Axr = Ax(n_var);
if Pbhinjw>=Pw(n_var)
   mobilityw_r_up = f_krw(1-sor)/muw/Bw;
else
   mobilityw_r_up = f_krw(Sw(n_var))/muw/Bw;
end
Twr(n_var,1) = Ax(n_var)*K(n_var)*mobilityw_r_up; % --> Caution: for consant K and A
% left oil
K_l = K(n_var-1);
Axl = Ax(n_var-1);
if Po(n_var-1)>=Po(n_var)
   mobilityo_l_up = f_kro(Sw(n_var-1))/muo/Bo;
else
   mobilityo_l_up = f_kro(Sw(n_var))/muo/Bo;
end
   Tol(n_var,1) = (2*Ax(n_var)*K(n_var)*Axl*K_l)/(Ax(n_var)*K(n_var)*dx(ii)+Axl*K_l*dx(ii-
1))*mobilityo_l_up;
   % left water
   K_l = K(n_var-1);
   Axl = Ax(n_var-1);
   if Pw(n_var-1)>=Pw(n_var)
```

```
        mobilityw_l_up = f_krw(Sw(n_var-1))/muw/Bw;
    else
        mobilityw_l_up = f_krw(Sw(n_var))/muw/Bw;
    end
    Twl(n_var,1) = (2*Ax(n_var)*K(n_var)*Axl*K_l)/(Ax(n_var)*K(n_var)*dx(ii)+Axl*K_l*dx(ii-
1))*mobilityw_l_up;
  else
    % right oil
    K_r = K(n_var+1);
    Axr = Ax(n_var+1);
    if Po(n_var+1)>=Po(n_var)
        mobilityo_r_up = f_kro(Sw(n_var+1))/muo/Bo;
    else
        mobilityo_r_up = f_kro(Sw(n_var))/muo/Bo;
    end
    Tor(n_var,1) =
(2*Ax(n_var)*K(n_var)*Axr*K_r)/(Ax(n_var)*K(n_var)*dx(ii)+Axr*K_r*dx(ii+1))*mobilityo_r_up;
    % right water
    K_r = K(n_var+1);
    Axr = Ax(n_var+1);
    if Pw(n_var+1)>=Pw(n_var)
        mobilityw_r_up = f_krw(Sw(n_var+1))/muw/Bw;
    else
        mobilityw_r_up = f_krw(Sw(n_var))/muw/Bw;
    end
    Twr(n_var,1) =
(2*Ax(n_var)*K(n_var)*Axr*K_r)/(Ax(n_var)*K(n_var)*dx(ii)+Axr*K_r*dx(ii+1))*mobilityw_r_up;
    % left oil
    K_l = K(n_var-1);
```

```
    Axl = Ax(n_var-1);
    if Po(n_var-1)>=Po(n_var)
       mobilityo_l_up = f_kro(Sw(n_var-1))/muo/Bo;
    else
       mobilityo_l_up = f_kro(Sw(n_var))/muo/Bo;
    end
    Tol(n_var,1) = (2*Ax(n_var)*K(n_var)*Axl*K_l)/(Ax(n_var)*K(n_var)*dx(ii)+Axl*K_l*dx(ii-
1))*mobilityo_l_up;
    % left water
    K_l = K(n_var-1);
    Axl = Ax(n_var-1);
    if Pw(n_var-1)>=Pw(n_var)
       mobilityw_l_up = f_krw(Sw(n_var-1))/muw/Bw;
    else
       mobilityw_l_up = f_krw(Sw(n_var))/muw/Bw;
    end
    Twl(n_var,1) = (2*Ax(n_var)*K(n_var)*Axl*K_l)/(Ax(n_var)*K(n_var)*dx(ii)+Axl*K_l*dx(ii-
1))*mobilityw_l_up;


  end
  % Accumulation Terms
  poro_p = Cr*poro(n_var);
  Bop = Co/Bo;
  Bwp = Cw/Bw;
  Cop(n_var,1) =   Vb/dt*(poro_p/Bo + poro(n_var)*Bop)*(1-Sw(n_var,1));
  Cow(n_var,1) = - Vb/dt*(poro(n_var)/Bo);
  Cwp(n_var,1) =   Vb/dt*(poro_p/Bw + poro(n_var)*Bwp)*(Sw(n_var,1));
  der = (f_pc(1.01*Sw(n_var,1)) - f_pc(Sw(n_var,1)))/(0.01*Sw(n_var,1));
  if isnan(der)
```

```
    der = 0;
  end
  Cww(n_var,1) =   Vb/dt*(poro(n_var)/Bw) - der*Cwp(n_var,1);
  alpha(n_var,1) = - Cow(n_var,1)/Cww(n_var,1);
end



A = zeros(n);
F = zeros(n,1);
n_var = 0;
for ii = 1:nx
  n_var = n_var + 1;
  if ii == 1
    %          A(n_var , n_var - 1) = Tol(n_var) + alpha(n_var)*Twl(n_var);
    A(n_var , n_var + 1) = (Tor(n_var) + alpha(n_var)*Twr(n_var)) + ...
                  (Tol(n_var) + alpha(n_var)*Twl(n_var))/(((dx(n_var)+dx(n_var+1)/2)^2-
dx(n_var)/2*(dx(n_var)+dx(n_var+1)/2))/(dx(n_var)/2));
    A(n_var , n_var) = - (Tor(n_var) + alpha(n_var)*Twr(n_var))...
                - (Cop(n_var) + alpha(n_var)*Cwp(n_var))...
                - (Tol(n_var) + alpha(n_var)*Twl(n_var))
*(2+dx(n_var+1)/dx(n_var))^2/(((dx(n_var)+dx(n_var+1)/2)^2-
dx(n_var)/2*(dx(n_var)+dx(n_var+1)/2))/(dx(n_var)/2));
    F(n_var , 1) = - (Cop(n_var) + alpha(n_var)*Cwp(n_var))*Po(n_var) ...
      + alpha(n_var)*Twr(n_var)*(Pc(n_var + 1) - Pc(n_var)) ...
      + alpha(n_var)*Twl(n_var)*(Pc(n_var+1) -
(2+dx(n_var+1)/dx(n_var))^2*Pc(n_var))/(((dx(n_var)+dx(n_var+1)/2)^2-
dx(n_var)/2*(dx(n_var)+dx(n_var+1)/2))/(dx(n_var)/2))...
      - Tol(n_var)*Pbhinjo*((2+dx(n_var+1)/dx(n_var))^2 - 1)/(((dx(n_var)+dx(n_var+1)/2)^2-
dx(n_var)/2*(dx(n_var)+dx(n_var+1)/2))/(dx(n_var)/2))...
      - alpha(n_var)*Twl(n_var)*(Pbhinjo - f_pc(swc))*((2+dx(n_var+1)/dx(n_var))^2 -
1)/(((dx(n_var)+dx(n_var+1)/2)^2-dx(n_var)/2*(dx(n_var)+dx(n_var+1)/2))/(dx(n_var)/2));
```

```
   elseif ii == nx

      A(n_var , n_var - 1) = (Tol(n_var) + alpha(n_var)*Twl(n_var)) + ...

                        (Tor(n_var) + alpha(n_var)*Twr(n_var))/(((dx(n_var)+dx(n_var-1)/2)^2-
dx(n_var)/2*(dx(n_var)+dx(n_var-1)/2))/(dx(n_var)/2));

      %           A(n_var , n_var + 1) = Tor(n_var) + alpha(n_var)*Twr(n_var);

      A(n_var , n_var) =  - (Tol(n_var) + alpha(n_var)*Twl(n_var))...

                        - (Cop(n_var) + alpha(n_var)*Cwp(n_var))...

                        - (Tor(n_var) + alpha(n_var)*Twr(n_var)) *(2+dx(n_var-
1)/dx(n_var))^2/(((dx(n_var)+dx(n_var-1)/2)^2-dx(n_var)/2*(dx(n_var)+dx(n_var-1)/2))/(dx(n_var)/2));

      F(n_var , 1) = - (Cop(n_var) + alpha(n_var)*Cwp(n_var))*Po(n_var) ...

         + alpha(n_var)*Twl(n_var)*(Pc(n_var - 1) - Pc(n_var)) ...

         + alpha(n_var)*Twr(n_var)*(Pc(n_var-1) - (2+dx(n_var-
1)/dx(n_var))^2*Pc(n_var))/(((dx(n_var)+dx(n_var-1)/2)^2-dx(n_var)/2*(dx(n_var)+dx(n_var-
1)/2))/(dx(n_var)/2))...

         - (Tor(n_var) + alpha(n_var)*Twr(n_var))*Pbhinjw*((2+dx(n_var-1)/dx(n_var))^2 -
1)/(((dx(n_var)+dx(n_var-1)/2)^2-dx(n_var)/2*(dx(n_var)+dx(n_var-1)/2))/(dx(n_var)/2));

   else

      A(n_var , n_var - 1) = Tol(n_var) + alpha(n_var)*Twl(n_var);

      A(n_var , n_var + 1) = Tor(n_var) + alpha(n_var)*Twr(n_var);

      A(n_var , n_var) = sum(- A(n_var,:)) - (Cop(n_var) + alpha(n_var)*Cwp(n_var));

      F(n_var , 1) = - (Cop(n_var) + alpha(n_var)*Cwp(n_var))*Po(n_var) ...

         + alpha(n_var)*Twl(n_var)*(Pc(n_var - 1) - Pc(n_var)) ...

         + alpha(n_var)*Twr(n_var)*(Pc(n_var + 1) - Pc(n_var)) ;

   end

end

P2 = A \ F;


Sw2 = zeros(n,1);

n_var = 0;

for ii = 1:nx
```

```
    n_var = n_var + 1;
    if ii == 1
        Sw2(n_var,1) = Sw(n_var,1) + 1/Cow(n_var,1)*(Tor(n_var)*(P2(n_var+1) - P2(n_var))...
            + Tol(n_var)*(Pbhinjo*((2+dx(n_var+1)/dx(n_var))^2-1) - (2+dx(n_var+1)/dx(n_var))^2*P2(n_var)
+ P2(n_var+1))/(((dx(n_var)+dx(n_var+1)/2)^2-dx(n_var)/2*(dx(n_var)+dx(n_var+1)/2))/(dx(n_var)/2))...
            - Cop(n_var)*(P2(n_var) - Po(n_var)) ...
            );
    elseif ii == nx
        Sw2(n_var,1) = Sw(n_var,1) + 1/Cow(n_var,1)*(Tol(n_var)*(P2(n_var-1) - P2(n_var))...
            + Tor(n_var)*(Pbhinjw*((2+dx(n_var-1)/dx(n_var))^2-1) - (2+dx(n_var-1)/dx(n_var))^2*P2(n_var)
+ P2(n_var-1))/(((dx(n_var)+dx(n_var-1)/2)^2-dx(n_var)/2*(dx(n_var)+dx(n_var-1)/2))/(dx(n_var)/2))...
            - Cop(n_var)*(P2(n_var) - Po(n_var)) ...
            );
    else
        Sw2(n_var,1) = Sw(n_var,1) + 1/Cow(n_var,1)*(Tol(n_var)*(P2(n_var-1) - P2(n_var))...
            + Tor(n_var)*(P2(n_var+1) - P2(n_var))...
            - Cop(n_var)*(P2(n_var) - Po(n_var)) ...
            );
    end
end

out(1:n,1) = P2;
out(n+1:2*n,1) = Sw2;

function pc = pcm_fun(S,Sor,Swc,b)
Se = (S-Swc)./(1-Sor-Swc);
pc = zeros(size(S));
for ii = 1:length(S)
    if Se(ii) < 0.001
```

```
        pc(ii) = - b*log(0.001);
    else
        pc(ii) = - b*log(Se(ii));
    end
end
```