Project 2

PGE 392K- Simulation Problem 2

Introduction

The present work develops the code for the simulation of 2 phase flow in a reservoir with changing depth, and arbitrary size, given variables like porosity, permeability, and position and schedule of wells. The code is made to be as general as possible, and is validated against a one dimensional simulator for which we know the analytical solution from Buckley-Leverett theory.

Working equations

We start with the general form of the weak form of the mass conservation equation:

$$V\frac{d\left(\varphi(S_i / B_i)\right)}{dt}+\int_A\left(\vec{n}_i \bullet \frac{u_i}{B_i}\right)dA=\int_V q_i dV$$

The above equation can be written in terms of fluxes. We use finite differencing to discretize the reservoir into cell blocks, and calculate the fluxes across each face.

The flux can be assumed to be constant across the flux into the adjoining cells, from which we can eliminate the *P(i+1/2,j)* term to get the expression for flux in terms of cell block pressures:

$$P_{i+1/2,j}=\frac{\left(h\Delta y\frac{2k_x}{\mu\Delta x}\right)_{ij}P_{ij}-\left(h\Delta y\frac{2k_x}{\mu\Delta x}\right)_{i+1,j}P_{i+1,j}}{\left(h\Delta y\frac{2k_x}{\mu\Delta x}\right)_{ij}+\left(h\Delta y\frac{2k_x}{\mu\Delta x}\right)_{i+1,j}}$$

$$\left(hu_x\Delta y\right)_{i+1/2,j}=\left(T_x\right)_{i+1/2,j}\left(P_{ij}-P_{i+1,j}\right)$$

Where,

$$(T_x)_{i+1/2,j} = 2\left(\frac{1}{\left(\frac{k_x h \Delta y}{\mu \Delta x}\right)_{ij}} + \frac{1}{\left(\frac{k_x h \Delta y}{\mu \Delta x}\right)_{i+1,j}} \right)^{-1}$$

For the source term,

$$(\phi c_t)_{ij} \Delta x_{ij} \Delta y_{ij} h_{ij} = (c_t)_{ij} (V_p)_{ij}$$

The well rate in cell *(i,j)* is given by:

$$\Delta x_{ij} \Delta y_{ij} h_{ij} q_{ij}^m = Q_{ij}^m$$

Finally, the weak form of the mass conservation equations becomes:

$$c_{tij} V_{pij} \left(P_{ij}^{n+1} - P_{ij}^n\right) = \Delta t(\Delta_x T_x \lambda_{rt} \Delta_x P_1^{n+1} + \Delta_y T_y \lambda_{rt} \Delta_y P_1^{n+1}$$

$$+\Delta_x T_x \lambda_{r2} \Delta_x P_c^n + \Delta_y T_y \lambda_{r2} \Delta_y P_c^n$$

$$-\Delta_x T_x (\lambda_{r1}\gamma_1 + \lambda_{r2}\gamma_{2)} \Delta_x D - \Delta_y T_y (\lambda_{r1}\gamma_1 + \lambda_{r2}\gamma_{2)} \Delta_y D + Q_{1ij}^n + Q_{2ij}^n)$$

The notations are the same as in the previous project, with the addition of the capillary pressure and the gravity terms.

As before, when we solve the implicit form of the above equation, we get the system of equations in terms of Ax = b:

$$\vec{\vec{T}} \bullet \vec{P}^{n+1} = \vec{B}$$

Where, $\vec{\vec{T}}$ is the transmissibility matrix. The solution for the above equation is given by:

$$\vec{P}^{n+1} = \vec{\vec{T}}^{-1} \bullet \vec{B}$$

For construction of the B matrix, we must know the cell rates Q. If the problem is rate constrained, those rates are known beforehand. For pressure constrained problems, we must have a well model to find out the flow rate in the well:

$$Q_l^m = J_l \, (P_{w_f l}^m - P_{i_\ell, j_\ell}^m)$$

Where, J_l is the productivity index of the well. It can be predicted using a well model equation:

$$J_\ell = \frac{2\pi k_{i_\ell, j_\ell} h_{i_\ell, j_\ell}}{\mu\left[\frac{1}{2}\ln\left[\frac{4A_{i_\ell, j_\ell}}{\gamma C_A r_{w\ell}^2}\right] + \frac{1}{4} + s_\ell\right]}$$

$$k_{i_\ell, j_\ell} = \sqrt{k_{x, i_\ell, j_\ell} k_{y, i_\ell, j_\ell}}$$

$$A_{i_\ell, j_\ell} = \Delta x_{i_\ell, j_\ell} \Delta y_{i_\ell, j_\ell}$$

We use the IMPES formulation here, where we first solve the pressure equation described above at a given saturation using the implicit formulation, and then solve the saturation equation to get the saturations for the new pressure field. The saturation equation is explicit, and hence the name IMPES.

$$V_{pij}\,(S_1^{n+1} - S_1^n) = \Delta t(\Delta_x T_x \lambda_{r1} \Delta_x P_1^{n+1} + \Delta_y T_y \lambda_{r1} \Delta_y P_1^{n+1}$$

$$-\Delta_x T_x \lambda_{r1} \gamma_1 \Delta_x D - \Delta_y T_y \lambda_{r1} \gamma_1 \Delta_y D + Q_1^n)$$

The relative mobility, λ_{r1}, needs to be taken at the cell faces. An upwind scheme must be followed for this, as the weighting method used for calculation of transmissibility at the cell faces is unconditionally unstable.

$$\lambda_{r1,i+1/2,j} = \omega\lambda_{r1,ij} + (1-\omega)\lambda_{r1,i+1,j}$$

The mobility assumes the value of the higher potential cell.

Hence, the potential must also be calculated dynamically. The potential is defined as:

$$\varphi_j = \int_{P_{Datum}}^{P_j} \frac{dx}{\gamma_j} - D$$

$$\gamma_j = \rho_j g$$

$$\lambda_{r1} = \frac{kr1}{B_j \mu_j}$$

The relative permeabilities are defined using Brooks-Corey relations:

$$k_{rj} = k_{rj,0}\left(\frac{S_j - S_{jr}}{1 - S_{1r} - S_{2r}}\right)^{n_j}$$

Part 1: Validation

The simulator is validated for a one-dimensional reservoir, of 200m length, with grid size of 1 m, with unit length in the y-direction, and constant thickness. The results for the simulator are shown below for the analytical case versus the simulator, at breakthrough, which occurs at approximately 1.73e6 seconds, or ~20 days. As can be seen, we get a very good match. Near the shock front, the match is somewhat off, but that is to be expected, as the numerical technique will tend to smoothen the shock due to numerical dispersion. We inject 100 m3/day at one end, and maintain 100 kPa pressure at the other end.

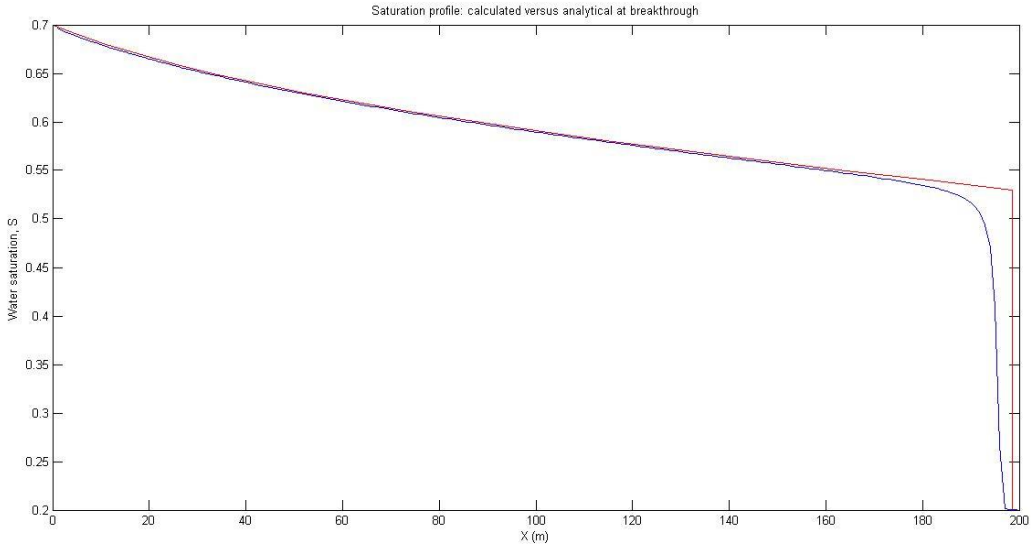

Figure 1:Validation case- Analytical vs Simulation

Part2: Time Step Effects

In this section, the effect of the time steps is analyzed. Dimensionless times and distances are used here, hence the graph is plotted accordingly. The volumetric injection rate is kept fixed at 100 m3/day, with the same grid dimensions. The viscosities are however made equal, and the residual phase saturations are made zero, while the relative permeabilities are made equal to the respective phase saturations.

$$t_D = \frac{qt}{Vp} \qquad t = Vp * \frac{t_D}{q}$$

$$dt = A \frac{dx}{L} Vp/q$$

The figure below shows the saturation profile for $\Delta t_D = 0.5\Delta x_D$, at $t_D = 0.5$. This case is stable.

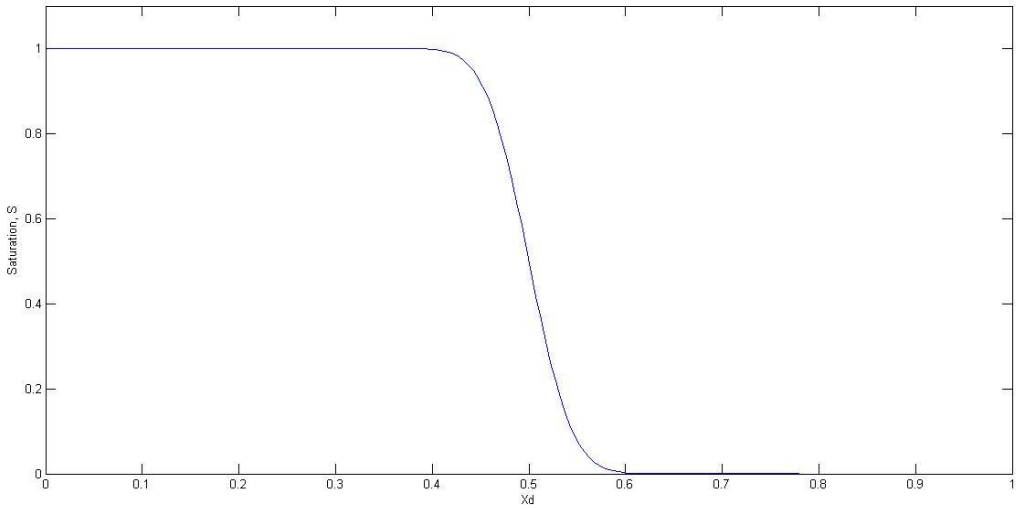

Figure 2: Δt_D=0.5Δx_D, at t_D=0.5

The figure below shows the saturation profile for $\Delta t_D = 0.95\Delta x_D$, at $t_D = 0.5$. There is significant sharpening of the front, but this is also stable.

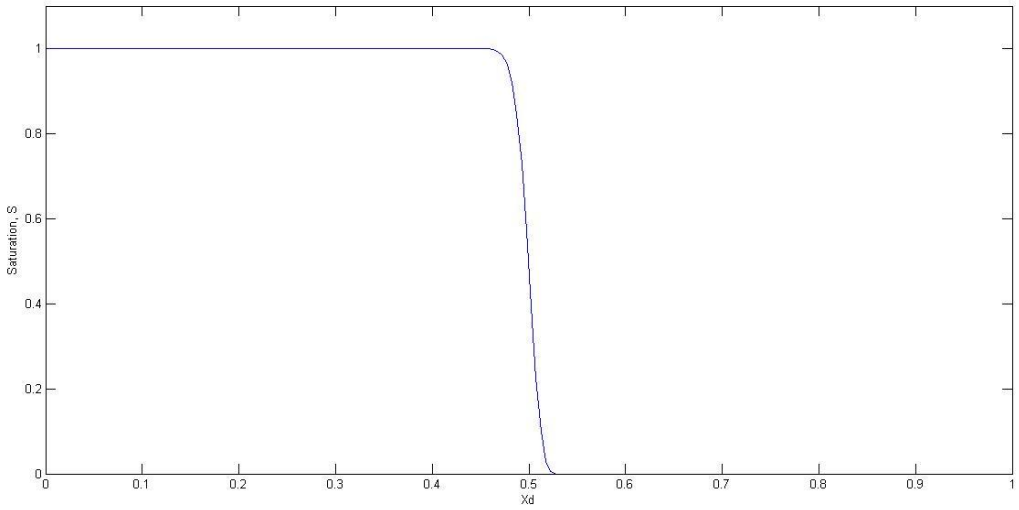

Figure 3: Δt_D=0.95Δx_D, at t_D=0.5

The last case was for $\Delta t_D = 1.5\Delta x_D$, at $t_D = 0.5$. We got unphysical values for the saturation, and this case is not plotted here. Hence this case is unstable. This confirms the stability analysis of the system, that for stability, $\Delta t_D < \Delta x_D$.

Part 3: Application Case:

Finally, the simulator was run on an actual dipping reservoir to see the migration of injected water which pushes out the oil through 3 wells. When water cut reaches 0.95 in a given well, it is turned into a water injector. The last well is shut off when water cut reaches 0.95.

The simulator predicted that the last well would shut off in 6169 days, using a time step of 1 day. The results from the simulator are shown below.

The initial oil in place was estimated using the saturation field in the beginning. It came out to be 15.89 million barrels.

The relative permeability and capillary pressure data are read from a table. The data is fitted inside the simulator using a sixth order polynomial for the relative permeability, and an exponential model for the capillary pressure. The results are shown in Figure 4 and 5.

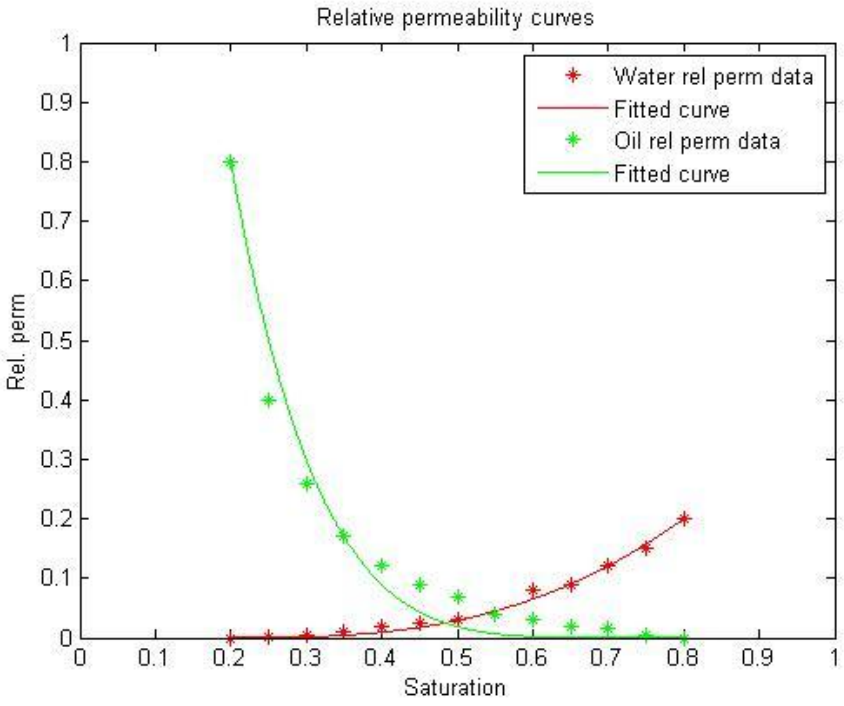

Figure 4: Relative permeability curves

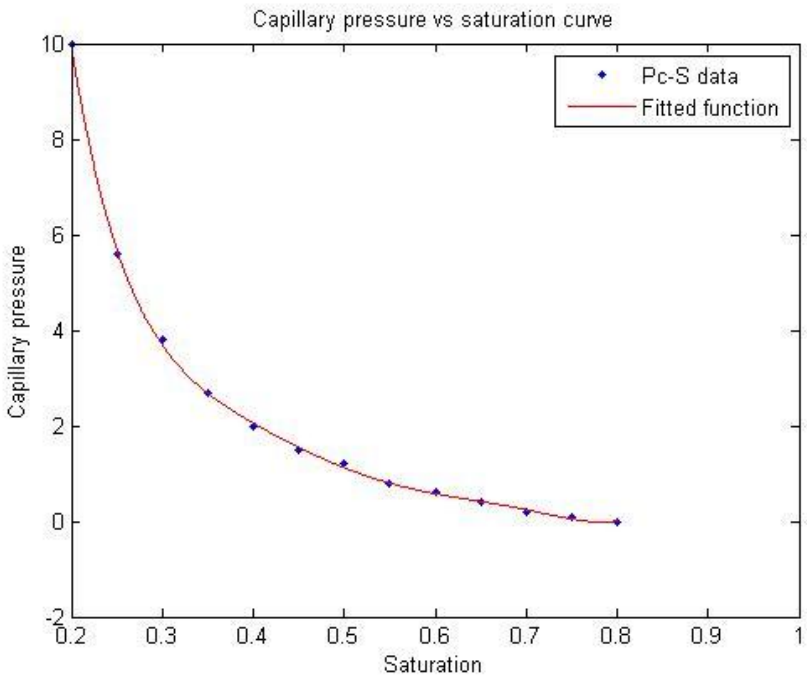

Figure 5: Capillary pressure curves

Figure 6 gives the oil pressure field at the end of time stepping.

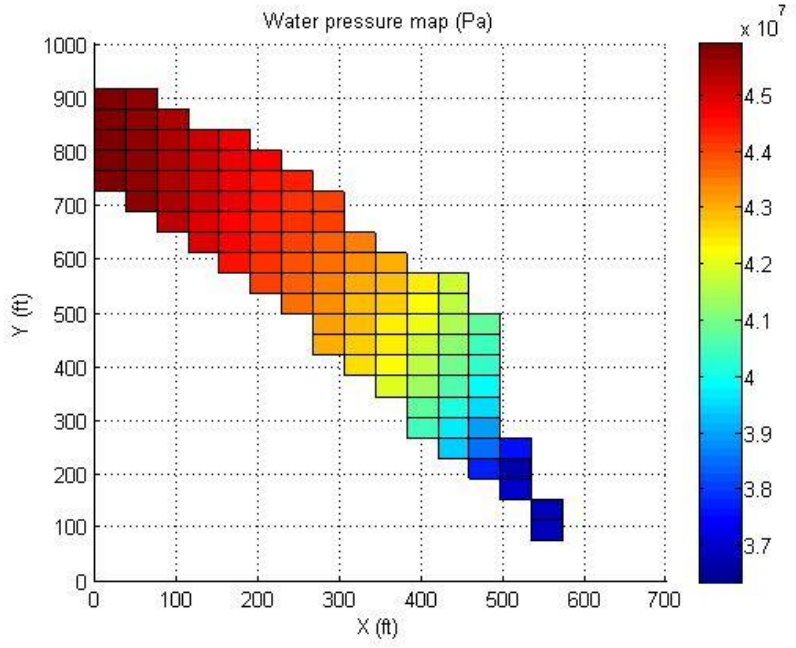

Figure 6: Water Pressure colormap

Figure 7 below shows the water saturation color map. As can be observed, there are peaks in saturation where the wells have turned into injectors. This shows the simulator is giving realistic values.

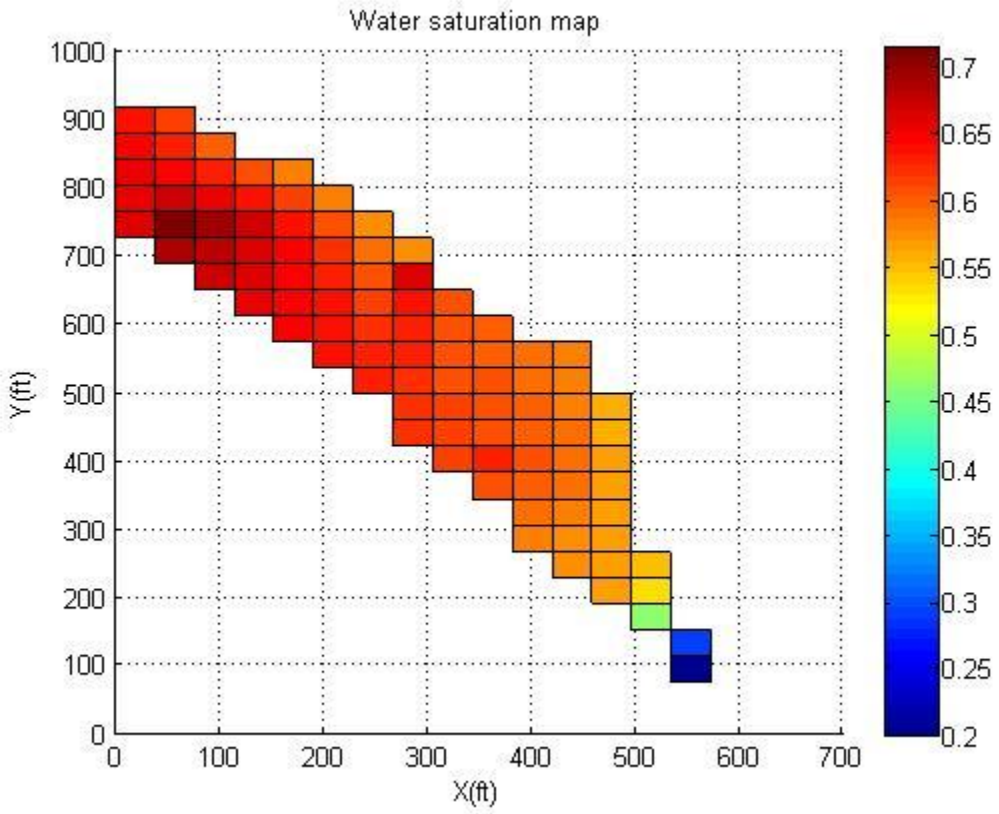

Figure 7: Water Saturation map

The average water saturation in the reservoir is shown in Figure 8 as a function of time. The jumps in the curve show the times when wells were turned into water injectors.

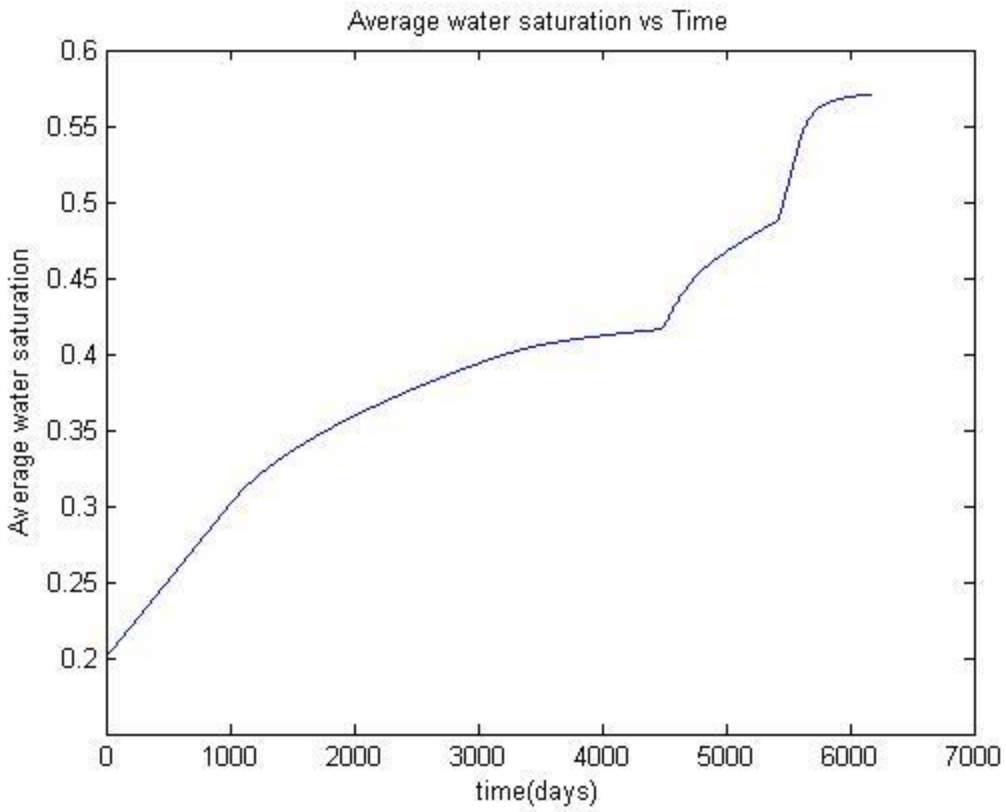

Figure 8: Water Saturation vs Time

The water production rate for wells 2-4 with time is shown in Figure 9. The water production goes to zero once a well is made into an injector. Figure 10 shows the variation of water-cut with time.

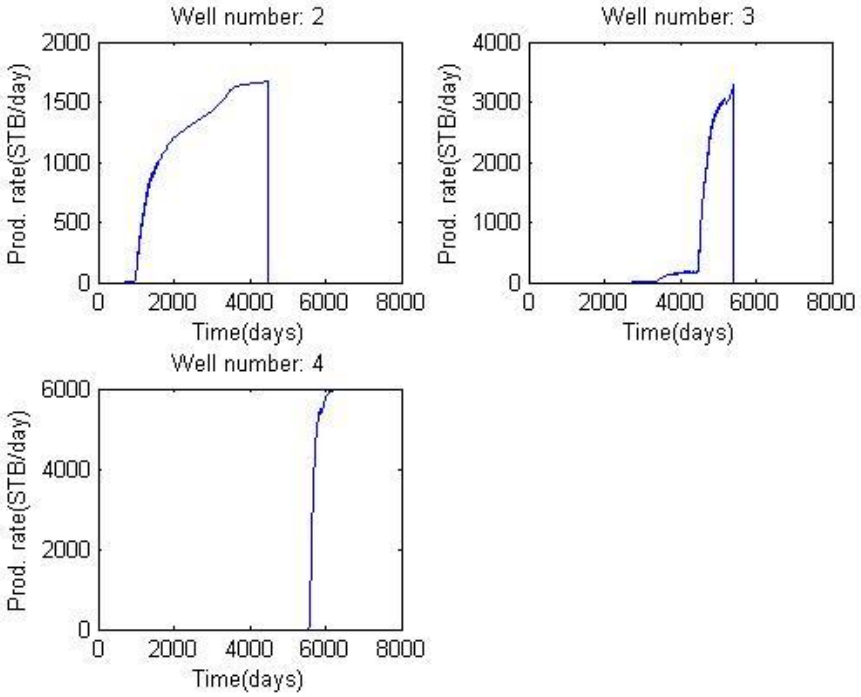

Figure 9: Water production rates for the three wells

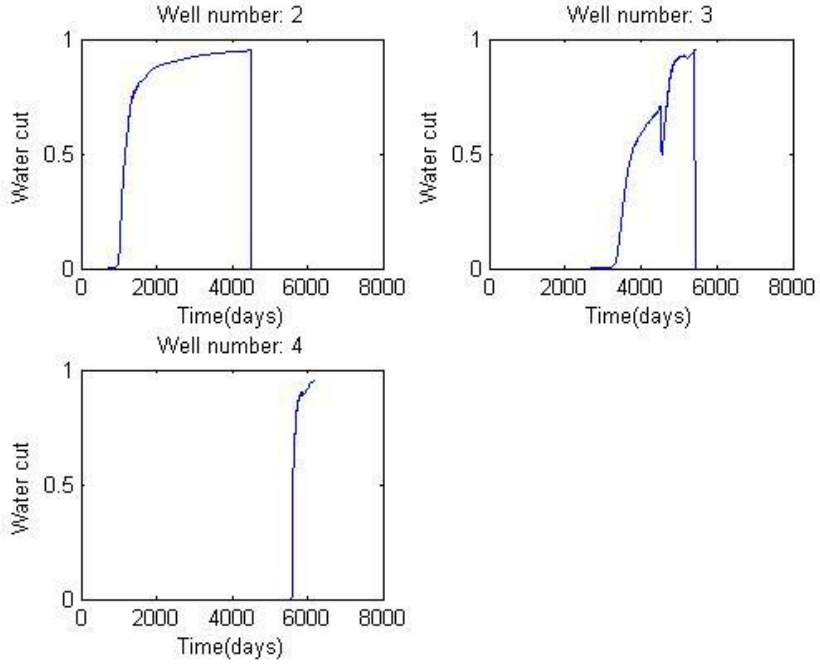

Figure 10: Water-cut for the three wells

Finally, the oil production from the reservoir is shown as a function of time in figure 11.

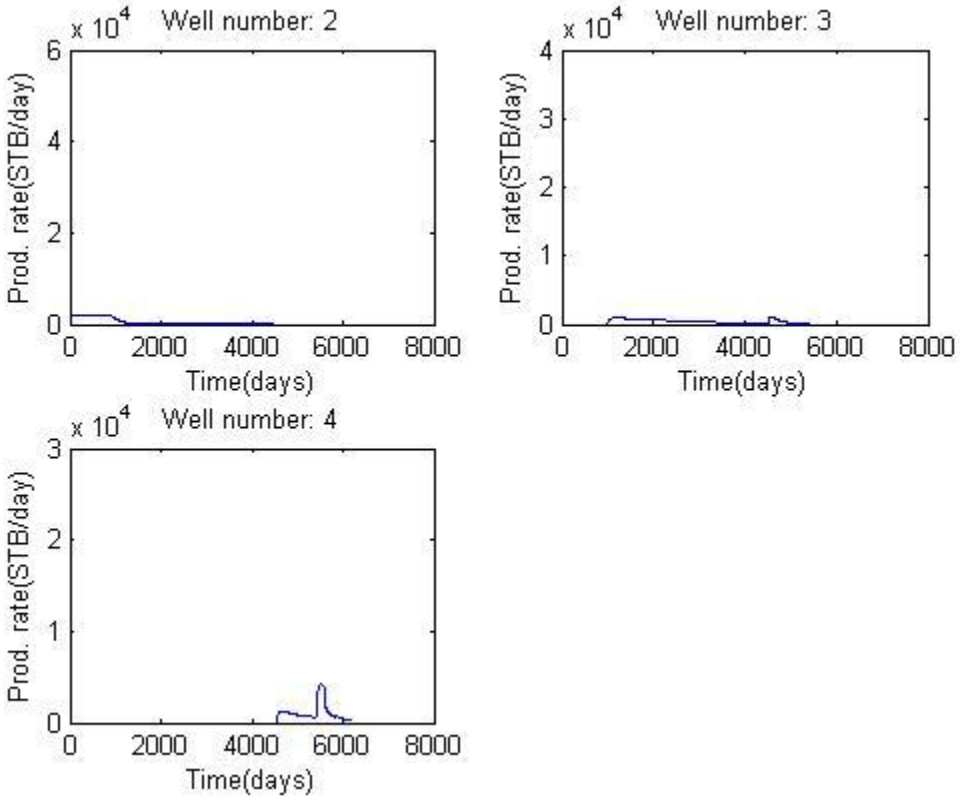

Figure 11: Oil production wrt time

The total oil recovered from the field over its producing life is 7.35 million barrels. Figure 12 shows the cumulative oil recovery in standard barrels, as a function of time.

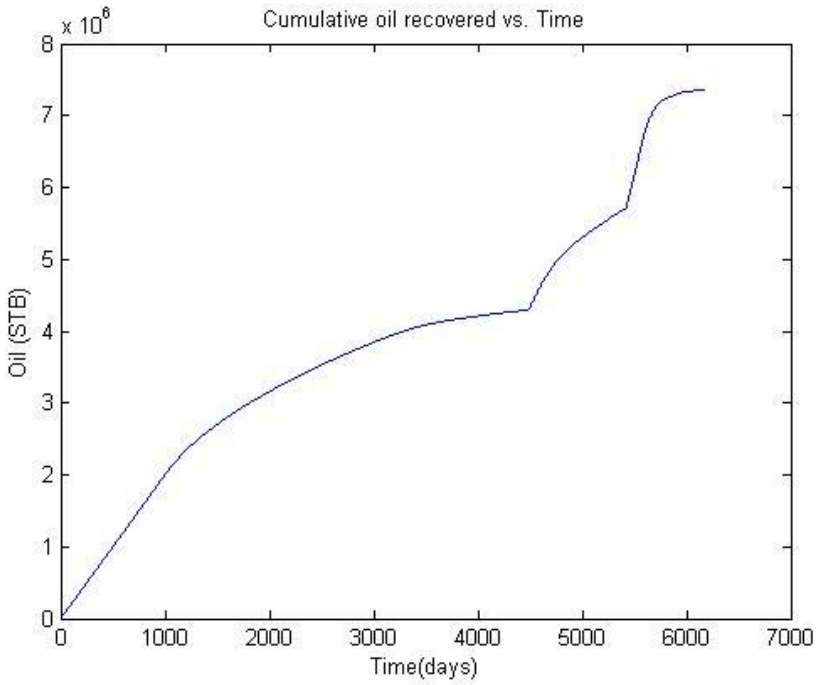

Figure 2: Cumulative oil recovery

Using the cumulative oil recovered, and the original oil in place, we get a recovery efficiency of 46.27%.

The producing life of the reservoir is 6169 days.

Conclusion

The simulator was validated against a 1-D case for 2 phase flow, and agreed with the predictions from Buckley-Leverett theory. The subsequent application case which incorporated effects of capillary pressure and gravity was also working well for the case of a real reservoir.

Appendix: Source code of the simulator (in Matlab)

```
clc;
clear;
close all
```

```
%Define input files for grid, depth, porosity, permeability and fluid
%properties
geomfile = 'Grid.xlsx';
por_file = 'Porosity.xlsx';
perm_file = 'Permeability.xlsx';
props_file = 'Fluid properties.xlsx';
thickness_file = 'Thickness.xlsx';
data_file = 'Data.xlsx';
well_props_file = 'Wells.xlsx';

D = xlsread(geomfile);
D=-1*0.3048*D;
[Ny Nx] = size(D);
phi = xlsread(por_file);

time_step = input('Enter time step in days: ');
del_t = 24*60*60*time_step;
t_max=10000*del_t;
steps = t_max/del_t;

disp('The fluid properties, and initial pressure at water-oil contact are
taken from the input file, "Fluid properties.xlsx"')
disp('The file can be modified for different fluid properties as needed');
%Read in fluid properties from input file
props = xlsread(props_file);

mu1 = props(1); mu2 = props(2); B1 = props(3)*ones(Ny,Nx);
B2 = props(4)*ones(Ny,Nx); c1 = props(5); c2 = props(6);
cf = props(7); P1_init = props(8)*ones(Ny,Nx)*6894; rho1 = props(9);
rho2 = props(10); g = 9.8;

gam1=rho1*g./B1; gam2=rho2*g./B2;
k_x=xlsread(perm_file)*1e-15;
k_y=xlsread(perm_file)*1e-15;

del_x=125*.3048*ones(Ny,Nx);
del_y=125*.3048*ones(Ny,Nx);

h=0.3048*xlsread(thickness_file);
wat_oil_contact = 14000*0.3048;

Vp = phi.*del_x.*del_y.*h;


%Getting well properties
well_props = xlsread(well_props_file);
well_xcoord = well_props(:,1);
well_ycoord = well_props(:,2);
water_rates = well_props(:,3);
oil_rates = well_props(:,4);
well_pwf = well_props(:,5);
well_skin = well_props(:,6);
well_radii = well_props(:,7);

q1 = zeros(Ny,Nx);q2 = zeros(Ny,Nx);
```

```
q1(sub2ind(size(q1),well_ycoord,well_xcoord)) =
water_rates*0.3048^3*5.615/(24*60*60);
q2(sub2ind(size(q1),well_ycoord,well_xcoord)) =
oil_rates*0.3048^3*5.615/(24*60*60);
Pwf=zeros(Ny,Nx);
Pwf(sub2ind(size(q1),well_ycoord,well_xcoord)) = well_pwf;
s = zeros(Ny,Nx);
s(sub2ind(size(q1),well_ycoord,well_xcoord)) = well_skin;
rw = zeros(Ny,Nx);
rw(sub2ind(size(q1),well_ycoord,well_xcoord)) = well_radii;
Gama = 1.73; CA = 31;

num_wells = size(well_xcoord,1);

%some additional variables needed for generating final results
S_avg = zeros(steps,1);
water_prod = zeros(num_wells,steps); oil_prod=zeros(num_wells,steps);
Water_cut=zeros(num_wells,steps);

ind = 0;

%read permeability and capillary pressure data, fit functions and plot the
results
data = xlsread(data_file);

S_data = data(:,1);
kr_o = data(:,2);
kr_w = data(:,3);
Pc_data = data(:,4);

S1r = S_data(1);
S2r = 1-S_data(end);

kr10 = kr_w(end);
kr20 = kr_o(1);

g1 = @(n1, x) kr10*((x-S1r)./(1-S2r-S1r)).^n1;
g2 = @(n2, x) kr20*((1-x-S2r)./(1-S2r-S1r)).^n2;

kr1_f = fit(S_data,kr_w, g1,'StartPoint',1.5,'Robust','LAR');
kr2_f = fit(S_data,kr_o, g2,'StartPoint',1.5,'Robust','LAR');
Pc_f = fit(S_data, Pc_data,'poly6');
S_Pc_f = fit(Pc_data,S_data,'exp1');

n1 = coeffvalues(kr1_f);
n2 = coeffvalues(kr2_f);

P1_init= P1_init-gam1.*(wat_oil_contact-D);
Pc_init = (gam1-gam2).*(wat_oil_contact-D); %calculate initial distribution
of Pc
Pc_init = Pc_init/6894;
S1_init = S_Pc_f(Pc_init); %get saturations for given Pc
%{
for i = 1:Nx
    for j = 1:Ny
```

```
        objective = @(satn)Pc_f(satn)*6894-Pc_init(j,i);
        S1_init(j,i) = fzero(objective, 0.2);               %calculate
saturation for given capillary pressure from fitted Pc_f function
    end
end
%}

S1_init(S1_init<0.2) = 0.2; %if calculated saturation is less than S1r, set
it to S1r
S1_init = reshape(S1_init,[Nx Ny])';
S2_init = 1-S1_init;
S1_init(h==0) = 0;
S2_init(h==0) = 0;

P = P1_init;
S = S1_init;
Init_oil = sum(sum(S2_init.*Vp./B2))/5.615/0.3046^3;
fprintf('\nAmount of oil initially in place is %4.2f million
barrels\n',Init_oil/1e6);


% Transmissibility matrix terms
%T_x(i+1/2,j)
term1 = (del_x)./(k_x.*h.*del_y);
term2 = circshift(term1, [0 -1]);

T_x_right = 2.*(term1+term2).^(-1);
T_x_right(:,Nx) = 0;

%T_x(i-1/2,j)
term2 = circshift(term1, [0 1]);
T_x_left = 2.*(term1+term2).^(-1);
T_x_left(:,1) = 0;

%T_y(i,j+1/2)
term1 = (del_y)./(k_y.*h.*del_x);
term2 = circshift(term1, [-1 0]);
T_y_top = 2.*(term1+term2).^(-1);
T_y_top(Ny,:) = 0;

%T_y(i,j-1/2)
term2 = circshift(term1, [1 0]);
T_y_bot = 2.*(term1+term2).^(-1);
T_y_bot(1,:) = 0;

for t = del_t:del_t:t_max        %Main time loop
    ind = ind+1;
    ct = c1*S + c2*(1-S) + cf;

    kr1 = g1(n1,S);
    kr1(S<=0.2) = 0;
    kr1(S>0.8) = 0.2;

    kr2 = g2(n2,S);
    kr2(S<=0.2) = 0.8;
```

```
    kr2(S>=0.8) = 0;

    Pc = reshape(feval(Pc_f,S),size(P));
    Pc(S<=S1r) = 10;
    Pc(S>=(1-S1r)) = 0;
    Pc = Pc*6894;

    q1(sub2ind(size(q1),well_ycoord,well_xcoord)) =
water_rates*0.3048^3*5.615/(24*60*60);
    q2(sub2ind(size(q1),well_ycoord,well_xcoord)) =
oil_rates*0.3048^3*5.615/(24*60*60);
    Pwf(sub2ind(size(q1),well_ycoord,well_xcoord)) = well_pwf;

Jl=(2*pi*sqrt(k_x.*k_y).*h)./(0.5*log((4*del_x.*del_y)./(Gama*CA.*(rw.^2)))+1
/4+s);
    Jl(Pwf == 0) = 0;

    Jl((P+Pc<=Pwf) & Pwf ~= 0) = 0;

    w=abs(Jl.*(kr1./mu1).*(Pwf-P)./B1);
    o=abs(Jl.*(kr2./mu2).*(Pwf-P-Pc)./B2);
    wor = w./(w+o);

    q1(wor>=0.95) = 2000*5.615*.3048^3/(24*60*60);
    Jl(wor>=0.95) = 0;
    Pwf(wor>=0.95) = 0;

    if(wor(well_ycoord(end),well_xcoord(end))> 0.95)
        break;
    end

    water_prod_cur = abs(Jl.*(kr1./mu1).*(Pwf-P)./B1);
    oil_prod_cur = abs(Jl.*(kr2./mu2).*(Pwf-P-Pc)./B2);
    water_cut_cur = water_prod_cur./(water_prod_cur+oil_prod_cur);
    water_cut_cur(Jl == 0) = 0;

    water_prod(:,ind)=
water_prod_cur(sub2ind(size(q1),well_ycoord,well_xcoord));
    oil_prod(:,ind)= oil_prod_cur(sub2ind(size(q1),well_ycoord,well_xcoord));
    Water_cut(:,ind) =
water_cut_cur(sub2ind(size(q1),well_ycoord,well_xcoord));



    lamb_r1 = kr1/mu1;
    lamb_r2 = kr2/mu2;

    %Upstreaming of mobility
    %Potential calculation at each cell - Phase 1
    phi1 = (P./gam1)-D;


    phi1_x_r = diff(phi1,1,2);      %Ny by Nx-1 matrix, get phi(i+1)-phi(i)
    phi1_x_r(:,Nx) = phi1_x_r(:,Nx-1);
    phi1_y_t = diff(phi1,1,1);      %Ny - 1 by Nx matrix get phi(j+1)-phi(j)
```

```
    if(Ny~=1)
        phi1_y_t(Ny,:) = phi1_y_t(Ny-1,:);
    end


    w_x_r = zeros(Ny,Nx);
    w_x_l = zeros(Ny,Nx);

    w_x_r(phi1_x_r>0) = 0;
    w_x_r(phi1_x_r<=0) = 1;

    w_x_r(:,Nx) = 1;      %No right cell, so use value of current cell

    lamb1_right = w_x_r.*lamb_r1 + (1-w_x_r).*circshift(lamb_r1, [0 -1]);

    phi1_x_l = circshift(phi1_x_r, [0 1]);
    w_x_l(phi1_x_l>=0) = 1;       %Since it is phi_(i+1)-phi_i, the values
reverse from the lamb_right
    w_x_l(phi1_x_l<0) = 0;

    w_x_l(:,1) = 1;       %No left cell, so use value of current cell
    lamb1_left = w_x_l.*lamb_r1 + (1-w_x_l).*circshift(lamb_r1, [0 1]);

    w_y_t = zeros(Ny,Nx);
    w_y_b = zeros(Ny,Nx);

    w_y_t(phi1_y_t>0) = 0;
    w_y_t(phi1_y_t<=0) = 1;
    w_y_t(Ny,:) = 1;      %No cell above this, so use current cell

    lamb1_top = w_y_t.*lamb_r1 + (1-w_y_t).*circshift(lamb_r1, [-1 0]);

    phi1_y_b = circshift(phi1_y_t, [1 0]);
    w_y_b(phi1_y_b>=0) = 1;
    w_y_b(phi1_y_b<0) = 0;

    w_y_b(1,:) = 1;       %No cell below this, so use current cell
    lamb1_bot = w_y_b.*lamb_r1 + (1-w_y_b).*circshift(lamb_r1, [1 0]);

    %Potential calculation at each cell - Phase 2
    phi2 = (P./gam2)-D;

    phi2_x_r = diff(phi2,1,2);      %Ny by Nx-1 matrix, get phi(i+1)-phi(i)
    phi2_x_r(:,Nx) = phi2_x_r(:,Nx-1);
    phi2_y_t = diff(phi2,1,1);      %Ny - 1 by Nx matrix get phi(j+1)-phi(j)
    if(Ny~=1)
        phi2_y_t(Ny,:) = phi2_y_t(Ny-1,:);
    end
    w_x_r = zeros(Ny,Nx);
    w_x_r(phi2_x_r>0) = 0;
    w_x_r(phi2_x_r<=0) = 1;

    w_x_r(:,Nx) = 1;      %No right cell, so use value of current cell
```

```
    lamb2_right = w_x_r.*lamb_r2 + (1-w_x_r).*circshift(lamb_r2, [0 -1]);

    phi2_x_l = circshift(phi2_x_r, [0 1]);
    w_x_l(phi2_x_l>=0) = 1;       %Since it is phi_(i+1)-phi_i, the values
reverse from the lamb_right
    w_x_l(phi2_x_l<0) = 0;

    w_x_l(:,1) = 1;       %No left cell, so use value of current cell
    lamb2_left = w_x_l.*lamb_r2 + (1-w_x_l).*circshift(lamb_r2, [0 1]);

    w_y_t = zeros(Ny,Nx);
    w_y_t(phi2_y_t>0) = 0;
    w_y_t(phi2_y_t<=0) = 1;
    w_y_t(Ny,:) = 1;      %No cell above this, so use current cell

    lamb2_top = w_y_t.*lamb_r2 + (1-w_y_t).*circshift(lamb_r2, [-1 0]);

    phi2_y_b = circshift(phi2_y_t, [1 0]);
    w_y_b(phi2_y_b>=0) = 1;
    w_y_b(phi2_y_b<0) = 0;

    w_y_b(1,:) = 1;       %No cell below this, so use current cell
    lamb2_bot = w_y_b.*lamb_r2 + (1-w_y_b).*circshift(lamb_r2, [1 0]);

    gradD_x_r = diff(D,1,2);
    gradD_x_r(:,Nx) = gradD_x_r(:,Nx-1);
    gradPc_x_r = diff(Pc,1,2);
    gradPc_x_r(:,Nx) = gradPc_x_r(:,Nx-1);

    gradD_x_l = circshift(gradD_x_r, [0 1]);
    gradD_x_l(:,1) = gradD_x_l(:,2);
    gradPc_x_l = circshift(gradPc_x_r, [0 1]);
    gradPc_x_l(:,1) = gradPc_x_l(:,2);

    if(Ny~=1)
        gradD_y_t = diff(D,1,1);
        gradD_y_t(Ny,:) = gradD_y_t(Ny-1,:);
        gradPc_y_t = diff(Pc,1,1);
        gradPc_y_t(Ny,:) = gradPc_y_t(Ny-1,:);

        gradD_y_b = circshift(gradD_y_t,[1 0]);
        gradD_y_b(1,:) = gradD_y_b(2,:);
        gradPc_y_b = circshift(gradPc_y_t,[1 0]);
        gradPc_y_b(1,:) = gradPc_y_b(2,:);
    else
        gradD_y_t = zeros(Ny,Nx);
        gradPc_y_t = zeros(Ny,Nx);

        gradD_y_b = zeros(Ny,Nx);
        gradPc_y_b = zeros(Ny,Nx);
    end

    T = zeros(Nx*Ny,Nx*Ny);
```

```
    for i2 = 1:Nx
        for j2 = 1:Ny
            k2 = (j2-1)*Nx +i2;
            if(h(j2,i2) == 0)
                T(k2,k2) = 1;
            else

                T(k2,k2) =
ct(j2,i2)*Vp(j2,i2)+del_t*T_x_right(j2,i2)*(lamb1_right(j2,i2)+lamb2_right(j2
,i2))+del_t*T_x_left(j2,i2)*(lamb1_left(j2,i2)+lamb2_left(j2,i2))+del_t*T_y_t
op(j2,i2)*(lamb1_top(j2,i2)+lamb2_top(j2,i2))+del_t*T_y_bot(j2,i2)*(lamb1_bot
(j2,i2)+lamb2_bot(j2,i2)) ...

+del_t*Jl(j2,i2)*lamb_r1(j2,i2)+del_t*Jl(j2,i2)*lamb_r2(j2,i2);
                if (k2 > 1)
                    T(k2, k2-1) = -
del_t*T_x_left(j2,i2)*(lamb1_left(j2,i2)+lamb2_left(j2,i2));
                end

                if (k2 < Nx*Ny)
                    T(k2, k2+1) = -
del_t*T_x_right(j2,i2)*(lamb1_right(j2,i2)+lamb2_right(j2,i2));
                end

                if (k2-Nx > 0)
                    T(k2, k2 - Nx) = -
del_t*T_y_bot(j2,i2)*(lamb1_bot(j2,i2)+lamb2_bot(j2,i2));
                end

                if (k2+Nx <= Nx*Ny)
                    T(k2, k2 + Nx) = -
del_t*T_y_top(j2,i2)*(lamb1_top(j2,i2)+lamb2_top(j2,i2));
                end
            end
        end
    end

    B = ct.*Vp.*P -
del_t.*T_x_right.*gradD_x_r.*(lamb1_right.*gam1+lamb2_right.*gam2) +
del_t.*T_x_left.*gradD_x_l.*(lamb1_left.*gam1+lamb2_left.*gam2)+del_t.*T_y_bo
t.*gradD_y_b.*(lamb1_bot.*gam1+lamb2_bot.*gam2) ...
        -
del_t.*T_y_top.*gradD_y_t.*(lamb1_top.*gam1+lamb2_top.*gam2)+del_t.*(T_x_righ
t.*lamb2_right.*gradPc_x_r -
T_x_left.*lamb2_left.*gradPc_x_l+T_y_top.*lamb2_top.*gradPc_y_t ...
        -T_y_bot.*lamb2_bot.*gradPc_y_b)+ del_t*B1.*q1 + del_t*B2.*q2
+del_t*Jl.*lamb_r1.*Pwf+del_t*Jl.*lamb_r2.*(Pwf-Pc);

    B(h==0) = 1e17;
    B = reshape(B', [Nx*Ny 1]);
    X = T\B;
    P = (reshape(X, [Nx Ny]))';
```

```
    gradP_x_r = diff(P,1,2);
    gradP_x_r(:,Nx) = gradP_x_r(:,Nx-1);
    gradP_x_l = circshift(gradP_x_r, [0 1]);
    gradP_x_l(:,1) = gradP_x_l(:,2);

    if(Ny~=1)
        gradP_y_t = diff(P,1,1);
        gradP_y_t(Ny,:) = gradP_y_t(Ny-1,:);

        gradP_y_b = circshift(gradP_y_t,[1 0]);
        gradP_y_b(1,:) = gradP_y_b(2,:);
    else
        gradP_y_t = zeros(Ny,Nx);

        gradP_y_b = zeros(Ny,Nx);
    end

    S_avg(ind) = sum(sum(S.*Vp))/sum(sum(Vp));

    kr_d = (n1-1)*(kr10/(1-S2r-S1r))*((S-S1r)./(1-S2r-S1r)).^(n1-1);
    kr_d(S <0.2) =0 ;

    S = S +((del_t./Vp).*(T_x_right.*lamb1_right.*gradP_x_r-
T_x_left.*lamb1_left.*gradP_x_l)
+(del_t./Vp).*(T_y_top.*lamb1_top.*gradP_y_t-T_y_bot.*lamb1_bot.*gradP_y_b)
...
    -(del_t./Vp).*(T_x_right.*lamb1_right.*gam1.*gradD_x_r-
T_x_left.*lamb1_left.*gam1.*gradD_x_l) -
(del_t./Vp).*(T_y_top.*lamb1_top.*gam1.*gradD_y_t-
T_y_bot.*lamb1_bot.*gam1.*gradD_y_b))./(1-del_t.*Jl.*(Pwf-
P).*kr_d./mu1./Vp)+kr1.*del_t.*Jl.*(Pwf-P)./(Vp.*mu1-del_t.*Jl.*kr_d.*(Pwf-
P))+(del_t./Vp).*q1;

    S(isnan(S)) = 0.2;

end

P_end = P;
S1_final = S;

cumul_oil_prod=sum(sum(Vp./B2))*(.8-(1-S_avg))/5.615/.3046^3;
fprintf('\nUltimate recovery efficiency = %4.2f
%%\n',max(max(cumul_oil_prod))/(Init_oil)*100);
fprintf('\nProducing life of the field, in days = %5.0f\n',ind*time_step);
fprintf('\nCumulative oil produced over entire life of reservoir is %4.2f
million barrels\n',max(max(cumul_oil_prod))/1e6);


P_end(P_end ==-10^17)=NaN;
S1_final(h == 0)=NaN;
P_end(h == 0)=NaN;

x=1:del_x(1,1):Nx*del_x(1,1);
y=1:del_y(1,1):Ny*del_y(1,1);
```

```
[X Y]=meshgrid(x,y);

fig1 = figure;
plot(kr1_f,'r',S_data,kr_w,'r*');
hold on;
plot(kr2_f,'g',S_data,kr_o,'g*');
legend('Water rel perm data','Fitted curve','Oil rel perm data','Fitted
curve');
title('Relative permeability curves');
xlim([0 1]);
ylim([0 1]);
xlabel('Saturation');
ylabel('Rel. perm');

fig2 = figure;
plot(Pc_f,S_data,Pc_data);
title('Capillary pressure vs saturation curve');
legend('Pc-S data','Fitted function');
xlabel('Saturation');
ylabel('Capillary pressure');

fig3 = figure;
surf(X,Y,P_end);view(2);
colorbar
title('Water pressure map (Pa)')
xlabel('X (ft)')
ylabel('Y (ft)')
zlabel('Pressure (Pa)')

fig4 = figure;
surf(X,Y,S1_final);view(2);
colorbar
title('Water saturation map')
xlabel('X(ft)')
ylabel('Y(ft)')
zlabel('Saturation')

time=1:time_step:(ind-1)*time_step;

fig5 = figure('Name','Water production rates for each
well','NumberTitle','off');
for well_num=2:num_wells
    subplot(2,2,well_num-1)
    name = ['Well number: ' num2str(well_num)];
    plot(time,water_prod(well_num,time)/.3048^3/5.615*24*3600);
    title(name)
    xlabel('Time(days)')
    ylabel('Prod. rate(STB/day)')
end

fig6 = figure('Name','Oil Production rates for each
well','NumberTitle','off');
for well_num=2:num_wells
    subplot(2,2,well_num-1)
    name = ['Well number: ' num2str(well_num)];
```

```
    plot(time,oil_prod(well_num,time)/.3048^3/5.615*24*3600);
    title(name)
    xlabel('Time(days)')
    ylabel('Prod. rate(STB/day)')
end

fig7 = figure('Name','Water cuts for each well','NumberTitle','off');
for well_num=2:num_wells
    subplot(2,2,well_num-1)
    name = ['Well number: ' num2str(well_num)];
    plot(time,Water_cut(well_num,time));
    title(name)
    xlabel('Time(days)')
    ylabel('Water cut')
end

fig8 = figure;
plot(time,S_avg(time));
title('Average water saturation vs Time')
xlabel('time(days)')
ylabel('Average water saturation')

fig9 = figure;
plot(time,cumul_oil_prod(time));
title('Cumulative oil recovered vs. Time')
xlabel('Time(days)')
ylabel('Oil (STB)')
```