Driver Function

```
%---------------------------
% INPUTS
%---------------------------

%Specify Input XLSX Filepath in ./bin/inputfile.m and ./bin/readwells.m

%Input Kx,Ky,Porosity and Height Functions Folderpath
addpath('./functions/');

%----------------------------
% NO INPUTS BEYOND THIS POINT
%----------------------------

%clc;
%clear all;

%Add Auxiliary Functions to Path
addpath('./bin/');

%Parse Input
run('./bin/inputfile.m');

%Assemble Matrices
A = assembleA(nx,ny,dx,dy,deltim,mu,ct);

%Add Initial Pressure Data
B = assembleB(nx,ny,dx,dy,ct,Pn);

%Read Initial Well Data
run('./bin/readwells.m');

%Induct Initial Well Data into Matrices
[A,B] = addwells(A,B,welldata,numwells,deltim,dx,dy,nx,ny);

%Initialize Simulation Clock
curtim = 0;
stepnum = 0;

%Check Schedule at First Time Step and Rebuild Matrices
[welldata,curschedule,flag]=scheduler(scheduletime,schedulebook,curschedule,curtim,raw_data
,welldata);
if flag==1
   A = assembleA(nx,ny,dx,dy,deltim,mu,ct);
   B = assembleB(nx,ny,dx,dy,ct,Pn);
   [A,B] = addwells(A,B,welldata,numwells,deltim,dx,dy,nx,ny);
end
```

```
%Mark Inactive Cells
[ A,B,Pn ] = deactivate_cells( A,B,Pn,nx,ny,dx,dy,mu,deltim );
%size(A)

B

%Initialize Data Structures to Store Output
numsteps = endtim/deltim;
Pwf = zeros(numsteps,numwells);
Qwf = zeros(numsteps,numwells);
Psim = zeros(numsteps,nx*ny);
MatBal = zeros(numsteps,2);

%March Through Time
while curtim<=endtim
    stepnum = stepnum+1
    %A
    %B
    P = A\B;
    count = 0;
    %P = readd_cells(P,nx,ny);
    curtim = curtim + deltim;
    %disp('Current Time = ');
    %disp(curtim);
    %------------------
    %Store Well Outputs
    for i = 1:numwells
        [iwell,jwell] = locij(welldata(1,i),welldata(2,i),dx,dy);
        position = index(iwell,jwell,nx,ny);
        if welldata(6,i)==1
            Pwf(stepnum,i) = welldata(5,i)*welldata(7,i)/welldata(8,i) + P(position,1);
            Qwf(stepnum,i) = welldata(5,i)*welldata(7,i);
        end
        if welldata(6,i)==-1
            Pwf(stepnum,i) = welldata(7,i);
            Qwf(stepnum,i) = welldata(8,i)*(welldata(7,i)-P(position,1));
        end
    end
    %Store Pressure Profile
    Psim(stepnum,:)=P;
    %Calculate and Store Global Mass Balance Information
    lhs = 0;
    for i = 1:nx
        for j = 1:ny
            [xij,yij] = locxy(i,j,dx,dy);
            position = index(i,j,nx,ny);
            if(P(position,1)==-9999)
                count = count+1;
                continue;
```

```
            end
            lhs = lhs + ct*poro(xij,yij)*dx*dy*(P(position,1)-Pn(position,1))*hxy(xij,yij);
        end
    end
    rhs = 0;
    for i = 1:numwells
        rhs = rhs+deltim*Qwf(stepnum,i);
    end
    MatBal(stepnum,1) = stepnum;
    MatBal(stepnum,2) = rhs-lhs;
    %------------------
    %Check Well Schedule

[welldata,curschedule,flag]=scheduler(scheduletime,schedulebook,curschedule,curtim,raw_data
,welldata);
    %Rebuild Matrices with Modified Initial Conditions
    if flag == 1
        A = assembleA(nx,ny,dx,dy,deltim,mu,ct);
        [A,~] = addwells(A,B,welldata,numwells,deltim,dx,dy,nx,ny);
        [ A,~ ] = deactivate_cells( A,B,Pn,nx,ny,dx,dy,mu,deltim );
    end

    B = assembleB(nx,ny,dx,dy,ct,P);
    [~,B] = addwells(A,B,welldata,numwells,deltim,dx,dy,nx,ny);
    [ ~,B,~ ] = deactivate_cells( A,B,Pn,nx,ny,dx,dy,mu,deltim );
    Pn = P;
end

disp('Global Material Balance Information');
disp(MatBal)

%Visualize Output
```

Visualization Module

```
%------------------
% VISUALIZE
% The Data Visualization Module of ResSim v1
%------------------

% IMPORTANT
%------------------
% These scripts only work if main.m has already completed execution
% Please mark your modifications in this script clearly for easy use later
% Use Right-Click --> Comment to enable disable segments of the code
% Please add to the code capabilties section to keep track of changes

% CODE CAPABILITIES
%------------------
% 1. Plot Bottomhole Pressures vs Time
% 2. Plot Well Flow Rates vs Time
% 3. Plot Contours of Pressure at a particular Time



timestrand = [0:deltim:endtim];

%-----------------------------------------------------------------------
% Bottom Hole Pressure Profiles With Time
%-----------------------------------------------------------------------
wellid = 8; % Well ID of well for which plot is desired
plotdata = Pwf(:,wellid);
plot(timestrand,plotdata,'LineWidth',2);
grid on;
xlabel('Time [days]');
ylabel('Bottomhole Pressure [MPa]');
leg1 = strcat('Well# -',int2str(wellid));
l1=legend(leg1);
title('Bottomhole Pressure vs Time');
hold all;

wellid = 4; % Well ID of well for which plot is desired
plotdata = Pwf(:,wellid);
plot(timestrand,plotdata,'LineWidth',2);
grid on;
xlabel('Time [days]');
ylabel('Bottomhole Pressure [MPa]');
leg2 = strcat('Well# -',int2str(wellid));
legend(leg1,leg2);
hold all;
```

```
wellid = 7; % Well ID of well for which plot is desired
plotdata = Pwf(:,wellid);
plot(timestrand,plotdata,'LineWidth',2);
grid on;
xlabel('Time [days]');
ylabel('Bottomhole Pressure [MPa]');
leg3 = strcat('Well# -',int2str(wellid));
legend(leg1,leg2,leg3);
hold off;

%-----------------------------------------------------------------------
% Well Flow Rates With Time
%-----------------------------------------------------------------------
% wellid = 5; % Well ID of well for which plot is desired
% plotdata = Qwf(:,wellid);
% plot(timestrand,plotdata,'LineWidth',2);
% grid on;
% xlabel('Time [days]');
% ylabel('Well Flow Rate [m^3/day]');
% leg1 = strcat('Well# -',int2str(wellid));
% l1=legend(leg1);
% hold all;
%
% wellid = 10; % Well ID of well for which plot is desired
% plotdata = Qwf(:,wellid);
% plot(timestrand,plotdata,'LineWidth',2);
% grid on;
% xlabel('Time [days]');
% ylabel('Well Flow Rate [m^3/day]');
% leg2 = strcat('Well# -',int2str(wellid));
% legend(leg1,leg2);
% hold all;
%
% wellid = 2; % Well ID of well for which plot is desired
% plotdata = Qwf(:,wellid);
% plot(timestrand,plotdata,'LineWidth',2);
% grid on;
% xlabel('Time [days]');
% ylabel('Well Flow Rate [m^3/day]');
% title('Well Flow Rate vs Time');
% leg3 = strcat('Well# -',int2str(wellid));
% legend(leg1,leg2,leg3);
% hold off;

%-----------------------------------------------------------------------
% Pressure Contours and Heat Map at a Particular Time
%-----------------------------------------------------------------------
% plottime = 1000; %Select time step at which to plot pressure contours
% tstep = floor(plottime/deltim);
```

```
% X = [dx/2:dx:xdim-dx/2];
% Y = [dy/2:dy:ydim-dy/2];
% plotdata = reshape(Psim(tstep,:),nx,ny);
% plotdata(plotdata==-9999) = NaN;
% contourf(Y,X,plotdata,12,'LineWidth',1);
% xlabel('Domain X-dimension');
% ylabel('Domain Y-dimension');
% tempstring = 'Contour Map of Pressure [MPa] at time=';
% tempstring2 = 'days';
% tempstring = strcat(tempstring,num2str(plottime),tempstring2);
% title(tempstring);
% colormap jet;
% colorbar('location','southoutside');
```

Dependencies

```
function [ A ] = assembleA( nx,ny,dx,dy,deltim,mu,ct )
%ASSEMBLEAB Sets up the matrix for the first iteration
%Accounts for no-flow boundary conditions
%Does not implement wells or inactive cells
%Initializes with boundary conditions Pn

%Add path containing input functions
%addpath('../inputfunctions/');

%Add path containing auxiliary functions
%addpath('../bin/');

%Initialize Matrices
A = zeros(nx*ny);

for i=1:nx
    for j=1:ny
        [xij,yij] = locxy(i,j,dx,dy); %(x,y) coordinates of (i,j)
        hij = hxy(xij,yij); %height h(x,y) of (i,j)
        row = index(i,j,nx,ny); %matrix location of (i,j)
        %IMPLEMENT KX KY IN TRANSX TRANSY
        phi = poro(xij,yij);
        [trxf,trxb]=transx(dx,dy,nx,ny,mu,i,j);
        [tryu,tryl]=transy(dx,dy,nx,ny,mu,i,j);
        A(row,row)=A(row,row)+deltim*(tryu+tryl+trxb+trxf)+phi*ct*dx*dy*hij;
        if (i~=1)
            A(row,index(i-1,j,nx,ny))=A(row,index(i-1,j,nx,ny))-deltim*trxb;
        end
        if (i~=nx)
            A(row,index(i+1,j,nx,ny))=A(row,index(i+1,j,nx,ny))-deltim*trxf;
        end
        if (j~=1)
            A(row,index(i,j-1,nx,ny))=A(row,index(i,j-1,nx,ny))-deltim*tryl;
        end
        if (j~=ny)
            A(row,index(i,j+1,nx,ny))=A(row,index(i,j+1,nx,ny))-deltim*tryu;
        end
    end
end
return
end
```

```
function [ B ] = assembleB( nx,ny,dx,dy,ct,Pn )
%ASSEMBLEB assembles and returns matrix B with initial conditions

B = zeros(nx*ny,1);
for i=1:nx
    for j=1:ny
        [xij,yij] = locxy(i,j,dx,dy); %(x,y) coordinates of (i,j)
        hij = hxy(xij,yij); %height h(x,y) of (i,j)
        row = index(i,j,nx,ny); %matrix location of (i,j)
        phi = poro(xij,yij);
        B(row,1)=phi*ct*dx*dy*hij*Pn(row);
    end
end

return
end
```

```
%----INPUT SPECIFICATION

%Script to read Basic Inputs

%------------------------------------------------
%Porosity, Kx, Ky, Height are input as functions
%------------------------------------------------

inp_data = xlsread('../datainput.xlsx','BasicInput');

% Mesh Parameters
xdim = inp_data(1,1);
ydim = inp_data(2,1);
nx = inp_data(3,1);
ny = inp_data(4,1);
dx = xdim/nx;
dy = ydim/ny;

% Physical Parameters
mu = inp_data(6,1)*10^(-9)/(3600*24);
ct = inp_data(7,1)*10^(-3);

% Time Stepping Parameters
endtim = inp_data(9,1);
deltim = inp_data(10,1);

% Initial Conditions
Pinit = inp_data(12,1);
Pn = Pinit*ones(nx*ny,1);
```

```
%----------------
% INPUT
%----------------

%Gamma
gam = 1.781;

%Dietz Shape Factor
dief = 31.62;

addpath('../functions/');

%Script to read the well data
raw_data = xlsread('../datainput.xlsx','WellInput');

%----------------------------
% NO INPUTS BEYOND THIS POINT
%----------------------------

[~,numwells] = size(raw_data);

welldata = raw_data(1:8,:); %Well Data at time t=0 days

wellindex = [1:numwells]; %Keep track of wells

%Calculating Productivity Indices
for i = 1:numwells
   hiwell = hxy(welldata(1,i),welldata(2,i));
   kxwell = kx(welldata(1,i),welldata(2,i));
   kywell = ky(welldata(1,i),welldata(2,i));
   Aijwell = dx*dy;
   bradwell = welldata(3,i);
   sfactorwell = welldata(4,i);
   temp1 = 2*pi*sqrt(kxwell*kywell)*hiwell/mu;
   temp2 = 0.5*log(4*Aijwell/(gam*dief*bradwell^2))+0.25+sfactorwell;
   welldata(8,i) = temp1/temp2;
end

%Bookkeeping for Scheduling Tasks
schedulebook = []; %Well associated with each schedule change
scheduletime = []; %Time at which schedule changes
productivity_index = [];

for i=1:numwells

   for j=1:raw_data(8,i)
      scheduletime = [scheduletime,raw_data(9+(j-1)*4,i)];
      schedulebook = [schedulebook,i];
   end
```

```
end

curschedule = zeros(1,numwells); %Number of schedule updates per well
```

```
function [ A,B ] = addwells( A,B,welldata,numwells,deltim,dx,dy,nx,ny )
%ADDWELLS adds well data to matrices A and B

for i=1:numwells
   [iwell,jwell] = locij(welldata(1,i),welldata(2,i),dx,dy);
   position = index(iwell,jwell,nx,ny);
   if (welldata(6,i)==1)
      B(position,1)=B(position,1)+welldata(5,i)*deltim*welldata(7,i);
   end
   if (welldata(6,i)==-1)
      A(position,position)=A(position,position)+deltim*welldata(8,i);
      B(position,1)=B(position,1)+deltim*welldata(8,i)*welldata(7,i);
   end
end

end
```

```
function [ welldata,curschedule,flag ] =
scheduler( scheduletime,schedulebook,curschedule,curtime,raw_data,welldata )
%SCHEDULER schedule handler
welldata = welldata;
flag = 0;
[~,nc]=size(scheduletime);
for i=1:nc
    elt = scheduletime(i);
    if curtime==elt
        flag = 1;
        wellid = schedulebook(i);
        curschedule(i) = curschedule(i)+1;
        welliter = curschedule(i);
        temp_ind = 10 + (welliter-1)*4;
        welldata(5:7,wellid)=raw_data(temp_ind:temp_ind+2,wellid);
    end
end
end
```

```
function [ A,B,Pn ] = deactivate_cells( A,B,Pn,nx,ny,dx,dy,mu,deltim )
%DEACTIVATE_CELLS marks inactive cells with a negative pressure
raw_data = xlsread('./datainput.xlsx','InactiveCells');
[nrow,~]=size(raw_data);
for k = 1:nrow
    i = raw_data(k,1);
    j = raw_data(k,2);
    position = index(i,j,nx,ny);
    A(position,:)=0;
    A(:,position)=0;
    A(position,position)=1;
    B(position,1) = -9999;
    %Pn(position,1)
    [trxf,trxb]=transx(dx,dy,nx,ny,mu,i,j);
    [tryu,tryl]=transy(dx,dy,nx,ny,mu,i,j);
    if i~=1
        row = index(i-1,j,nx,ny);
        if A(row,row) ~= 1
            A(row,row) = A(row,row) - deltim*trxb;
        end
    end
    if i~=nx
        row = index(i+1,j,nx,ny);
        if A(row,row)~=1
            A(row,row) = A(row,row) - deltim*trxf;
        end
    end
    if j~=1
        row = index(i,j-1,nx,ny);
        if A(row,row) ~= 1
            A(row,row) = A(row,row) - deltim*tryl;
        end
    end
    if j~=ny
        row = index(i,j+1,nx,ny);
        if A(row,row) ~= 1
            A(row,row) = A(row,row) - deltim*tryu;
        end
    end
end



return



end
```

```
function [ trf,trb ] = transx( dx,dy,nx,ny,mu,i,j )
%TRANSX calculates the x-transmissibility of face i+1/2,j and i-1/2,j

%true x y coordinates can be determined from i,j
%in next version, use functions of h and k to get values at desired
%location
[x,y] = locxy(i,j,dx,dy);

a = (hxy(x,y)*dy*kx(x,y)/(mu*dx)); %at i,j
b = (hxy(x+dx,y)*dy*kx(x+dx,y)/(mu*dx)); %at i+1,j
c = (hxy(x-dx,y)*dy*kx(x-dx,y)/(mu*dx)); %at i-1,j

trf = (2*a*b)/(a+b);
trb = (2*a*c)/(a+c);

if (i<=1)
    %disp('Boundary!');
    trb = 0;
end

if (i>=nx)
    %disp('Boundary!');
    trf = 0;
end

return


end
```

```
function [ tru,trl ] = transy( dx,dy,nx,ny,mu,i,j )
%TRANSY calculates the y-transmissibility of face i,j+1/2 and i,j-1/2

%true x y coordinates can be determined from i,j
%in next version, use functions of h and k to get values at desired
%location
[x,y] = locxy(i,j,dx,dy);

a = (hxy(x,y)*dx*ky(x,y)/(mu*dy)); %at i,j
b = (hxy(x,y+dy)*dx*ky(x,y+dy)/(mu*dy)); %at i,j+1
c = (hxy(x,y-dy)*dx*ky(x,y-dy)/(mu*dy)); %at i,j-1

tru = (2*a*b)/(a+b);
trl = (2*a*c)/(a+c);

if (j<=1)
   %disp('Boundary!');
   trl = 0;
end

if (j>=ny)
   %disp('Boundary!');
   tru = 0;
end

return
end
```

```
function [ ind ] = index( i,j,nx,ny )
%INDEX returns the cardinal matrix position for element (i,j)
ind = (j-1)*nx+i;
return
end
```

```
function [ i,j ] = locij( x,y,dx,dy )
%LOCIJ returns (i,j) of given (x,y)
i = floor(x/dx)+1;
j = floor(y/dy)+1;
return
end
```

```
function [ x,y ] = locxy( i,j,dx,dy )
%LOCXY finds the (x,y) coordinates of cell center of (i,j)
x = (i-1)*dx + dx/2;
y = (j-1)*dy + dy/2;
return
end
```