APPENDICES

A. C++ program to generate Stone (1991) pseudo functions:

➔ Program workflow:

The work flow of the C++ program, used to generate Stones (1991) pseudo functions, is illustrated in Figure A.1 and can be described as follows:

1. Run a fine-scale simulation in order to obtain the oil and water flow rates in the x direction (FLOOILI & FLOWATI), oil and water relative permeabilities (OILKR & WATKR), and water saturation in the fine cells (SWAT).

2. Calculate transmissibility between fine cells in the x direction (TRANX) and pore volume of the fine cells (PORV).

3. Read in all input data files from steps no.1 and 2.

4. Perform the following calculations at the down-stream edge of each coarse cell for all time steps:
    a. Sum up the oil flow rates and water flow rates and calculate the total flow rate.
    b. Calculate the average oil and water fractional flows using total flow rate weighting.
    c. Calculate the total mobility.
    d. Calculate the average total mobility using transmissibility weighting.

5. Generate the pseudo oil and water relative permeabilities for each coarse block.

6. Calculate the average water saturation using pore volume weighting.

7. Print out the pseudo functions to an output file.

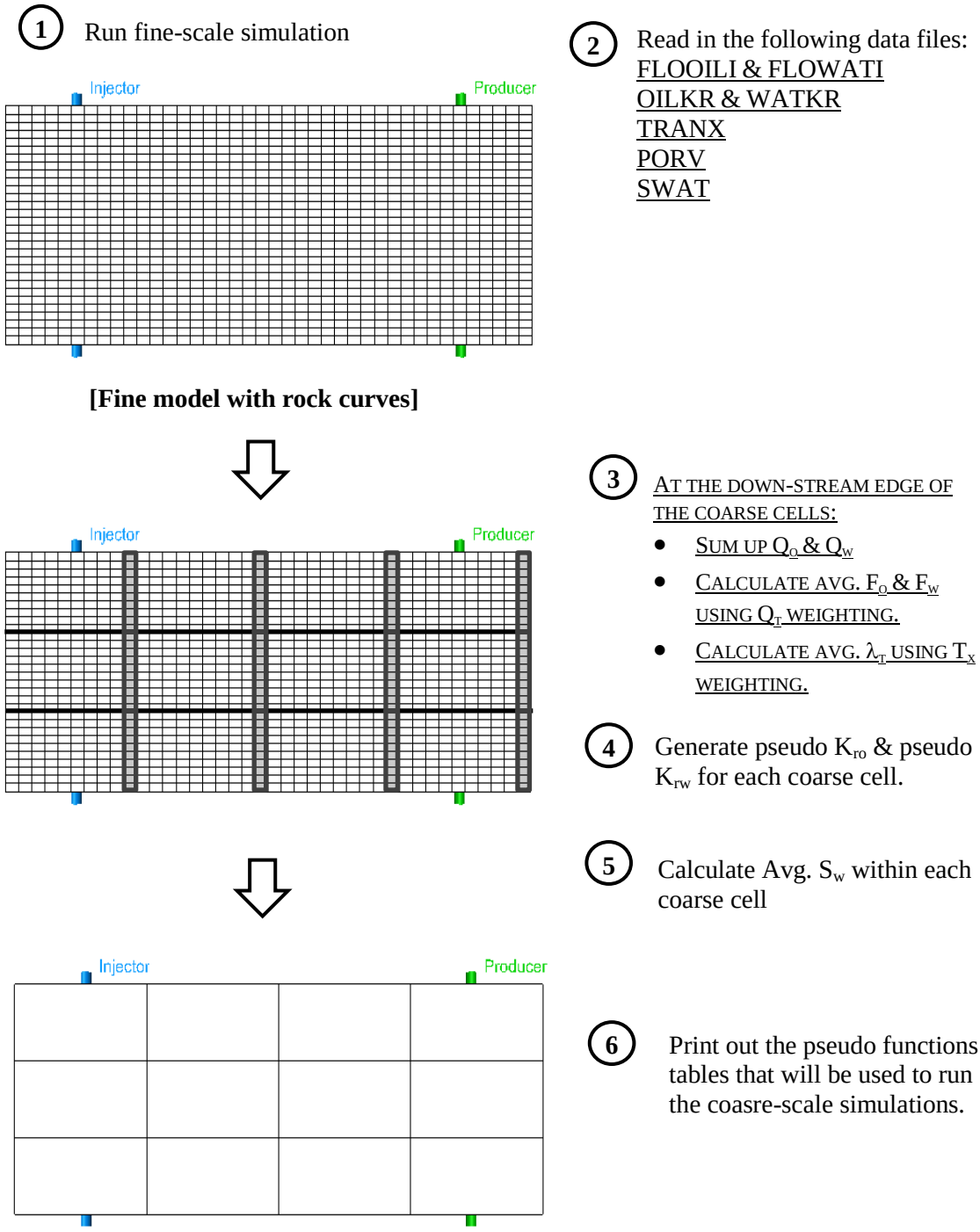

Figure A.1: Workflow of the C++ program (based on Stone's 1991 pseudo generation method).

➔ **Codes of the C++ program:**

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <iomanip>
using namespace std;
void Avg_FracFlow (int Ck,int Ci, int s, int ts, double avgfw [50],double avgfo [50], double qw [50][50], double qo [50][50]);
void Avg_TotMob (int Ck,int Ci,int s, int ts,double muo , double muw, double avgtotmob [50], double kro [50][50], double krw [50][50], double tranx [50][50]);
void Pseudos (int s, int ts, int nk, int ni, double muw, double muo,double pseudowat [50],double pseudooil [50], double avgfo [50], double avgfw [50], double avgtotmob [50]);
void Avg_Swat (int Ck, int Ci,int s,int ts,int ni, int nk, double AvgSwat [50] , double Sw [50][50], double Pv [50][50]);
int main ()
{
int nk, ni, ts, s;
string L;
double muo,muw,avgfw [50],avgfo [50],avgtotmob [50],pseudowat [50],pseudooil [50],swc,kroswc;
double qw [50][50],qo [50][50],kro [50][50],krw [50][50],tranx [50][50],Sw [50] [50],Pv [50][50];
double pseudokrw [50] [50], pseudokro [50][50], AvgSwat [50];
cout <<"Enter the following data:" <<endl;
cout <<"**************************" <<endl;
cout <<endl<< "No. of fine cells in the x direction:  ";
cin >> ni;
cout <<endl<< "No. of fine cells in the z direction:  ";
cin >> nk;
cout <<endl<< "No. of time steps:  ";
cin >> ts;
cout <<endl<<"Scale-up factor:  ";
```

```
cin >> s;
cout <<endl<< "Oil viscosity:   ";
cin>>muo;
cout <<endl<< "Water viscosity:   ";
cin>>muw;
cout <<endl<< "Connate water saturation:   ";
cin>>swc;
cout <<endl<< "Oil relative permeability @ Swc:   ";
cin>>kroswc;
ofstream pseudo4eclipse_outfile ("Pseudos.txt"); // print out file of pseudo functions.
for (int Ck = 1; Ck<nk; Ck=Ck+s) // Ck is a counter for coarse cells in the k direction.
{
   for (int Ci = s; Ci <= ni; Ci= Ci+s)  // Ci is a counter for coarse cells in the i direction.
   {
   //Read in the required input files.
   ifstream qw_infile ("FlowatI.txt");
   ifstream qo_infile ("FlooilI.txt");
   ifstream kro_infile ("Kro.txt");
   ifstream krw_infile ("Krw.txt");
   ifstream tranx_infile ("Tranx.txt");
   ifstream Pv_infile ("Porevol.txt");
   ifstream Sw_infile ("Swat.txt");
   //Skip header of 10 text lines in the input files.
   for (int t=0; t<10; t++)
   {
   getline (qw_infile,L);
   getline (qo_infile,L);
   getline (kro_infile,L);
   getline (krw_infile,L);
   getline (tranx_infile,L);
   getline (Pv_infile,L);
   getline (Sw_infile,L);
   }
   //Print out before each pseudo table: connate Sw value , Krw@Swc = 0, Kro@Swc
value and Pc@Swc = 0.
```

```
    pseudo4eclipse_outfile<<fixed<<swc<<"   "<< "0.00000"<<"
"<<fixed<<kroswc<<"   "<< "0.00000"<<endl;
    //Loop for time steps within each coarse cell.
    for (int tsteps = 1; tsteps <=ts; tsteps++ )
    {
    for (int k = 1; k<=nk; k++)         // loop for fine cells in the k direction within a coarse
cell.
        {
     for (int i = 1; i <=ni; i++)    // loop for fine cells in the i direction within a coarse cell.
            {
            // Read in data from input files to arrays.
            qw_infile>>qw [i] [k];
            qo_infile>>qo [i] [k];
            kro_infile>> kro [i][k];
            krw_infile>> krw [i][k];
            tranx_infile>>tranx [i] [k];
            Pv_infile>>Pv [i] [k];
            Sw_infile>>Sw [i] [k];
            }
        }
    Avg_FracFlow (Ck,Ci,s,ts,avgfw,avgfo,qw,qo); //function averages fractional flow.
    Avg_TotMob (Ck,Ci,s,ts,muo,muw,avgtotmob,kro,krw,tranx); // function averages
total mobility.
    Pseudos (s,ts,nk,ni,muw,muo,pseudowat,pseudooil,avgfo,avgfw,avgtotmob);//
function calculates the pseudos.
    Avg_Swat (Ck,Ci,s,ts,ni,nk,AvgSwat,Sw,Pv); // function calculates the average water
saturation.
    //Arrange the pseudos to be used in Eclipse.
    pseudo4eclipse_outfile.precision (5);
    pseudo4eclipse_outfile<<fixed<<AvgSwat [ts] <<"   " <<fixed<<pseudowat [ts] <<"
" <<fixed<<pseudooil [ts] <<"   "<<"0.00000"<<endl;
    cout<<fixed<<AvgSwat [ts] <<"   " <<fixed<<pseudowat [ts] <<"   "
<<fixed<<pseudooil [ts] <<"   "<<"0.00000"<<endl;
    //skip 3 text lines between time steps in the input files.
    for (int t=0; t<4; t++)
```

```
    {
    getline (qw_infile,L);
    getline (qo_infile,L);
    getline (kro_infile,L);
    getline (krw_infile,L);
    getline (tranx_infile,L);
    getline (Pv_infile,L);
    getline (Sw_infile,L);
    }
    }//end of time steps loop.
    //print out after each pseudo table, Sw=1, Krw=1, Kro=0, and Pc=0. Afterwards print slash(/).
    pseudo4eclipse_outfile << "1.00000"<<"   "<< "1.00000"<<"   "<< "0.00000"<<" "<< "0.00000"<<endl;
    pseudo4eclipse_outfile << "/" <<endl;
}//end of Ci loop
}//end of Ck loop
cout<<endl;
cout <<"*************************************" <<endl;
cout <<"Pseudo functions in the x direction have been generated" <<endl;
cout <<"*************************************" <<endl<<endl;
}
// Function calculates average fractional flows at the downstream side of the coarse cells.
void Avg_FracFlow (int Ck,int Ci, int s, int ts, double avgfw [50],double avgfo [50], double qw [50][50], double qo [50][50])
{
double sumqt = 0;
double sumqw = 0;
double sumqo = 0;
for ( int k = Ck; k<Ck+s; k++) // loop for fine cells in the k direction at downstream side of a coarse cell.
{
sumqt = sumqt + qw [Ci] [k] + qo [Ci] [k];  //summation of flow rates.
sumqw = sumqw + qw [Ci] [k];   //summation of qw.
```

```
sumqo = sumqo + qo [Ci] [k];   //summation of qo.
}
if (sumqt!=0)                  //check sumqt value to avoid division by zero.
      {
      avgfw [ts] = sumqw / sumqt;   //calculate avg. fw.
      avgfo [ts] = sumqo / sumqt;   //calculate avg. fo.
      }
      else
      {
      avgfw [ts] = 0;   //if sumqt equals zero, set avgfw to zero instead of printing out
"nan".
      avgfo [ts] = 0;
      }
sumqt = 0;
sumqw = 0;
sumqo = 0;
}
// Function calculates average total mobility at the downstream side of the coarse cells.
void Avg_TotMob (int Ck,int Ci,int s, int ts, double muo , double muw, double
avgtotmob [50], double kro [50][50], double krw [50][50], double tranx [50][50])
{
double totmob [50][50];
double totmobtranx [50][50];
double sumtotmobtranx =0;
double sumtranx =0;
   for ( int k = Ck; k<Ck+s; k++)  // loop for fine cells in the k direction at the
downstream side of a coarse cell.
   {
   totmob [Ci][k] = (kro [Ci][k]/muo) + (krw [Ci][k]/muw);    //calculates total mobility.
   totmobtranx [Ci][k] = totmob [Ci] [k] * tranx [Ci] [k];   //calculates product of total
mobility * tranx.
   sumtotmobtranx = sumtotmobtranx + totmobtranx [Ci] [k];  //summation of total
mobility * tranx.
   sumtranx = sumtranx + tranx [Ci] [k];                  //sum of tranx.
   }
```

```
    if (sumtranx!=0)  //check sumtranx value to avoid division by zero.
        {
        avgtotmob [ts] = sumtotmobtranx / sumtranx;  //calculate avg. total mobility using transmissibility weighting.
        }
    else
        {
        avgtotmob [ts] = 0;    //if sumtranx equals zero, set avgtotmob to zero instead of printing out "nan".
        }
        sumtotmobtranx = 0;
        sumtranx = 0;
}
//Function calculates the pseudo functions.
void Pseudos (int s, int ts, int nk, int ni, double muw, double muo,double pseudowat [50],double pseudooil [50], double avgfo [50], double avgfw [50], double avgtotmob [50])
{
pseudowat [ts] = avgfw  [ts] * avgtotmob [ts] * muw; //calculate pseudo krw.
pseudooil [ts] = avgfo  [ts] * avgtotmob [ts] * muo; //calculate pseudo kro.
}
//Function calculates average water saturation in coarse cells.
void Avg_Swat (int Ck, int Ci,int s,int ts,int ni, int nk, double AvgSwat [50] , double Sw [50][50], double Pv [50][50])
{
double sumswpv = 0;
double sumpv = 0;
double swpv = 0;
for (int k = Ck; k<Ck+s; k++)         // loop for fine cells in the k direction within a coarse cell.
    {
    for (int i = Ci-s+1; i <Ci+1; i++)    // loop for fine cells in the i direction within a coarse cell.
        {
        sumpv = sumpv + Pv [i] [k]; // calculates sum of pore volume.
```

```
        swpv = Sw [i][k] * Pv [i][k]; //calculates the product: water saturation * pore volume.
        sumswpv = sumswpv + swpv; // calculates sum of Sw * Pv
        }
    }

AvgSwat [ts] = sumswpv/sumpv;    //calculation of avg. Sw using pore volume weighting.
}
```
//End of program.

B. C++ program to generate TWR pseudo functions for a 2D model:

➔ **Program workflow:**

The work flow of the C++ program, used to generate pseudo functions using the transmissibility weighted relative permeability (TWR) method, is described as follows:

1. Run fine-scale simulation of the selected sector in the fine model in order to obtain the oil and water relative permeabilities (OILKR & WATKR) and the water saturation in the fine cells (SWAT) at all time steps.
2. Calculate transmissibility between fine cells in the x and z directions (TRANX & TRANZ) and pore volume of the fine cells (PORV).
3. Read in all input data files from steps no.1 and 2.
4. Perform the following calculations at the down-stream edge of each coarse cell for all time steps:
    a. Multiply the oil and water relative permeabilities in each fine cell by the transmissibility in that cell, and then sum up the products along the down-stream edge between two adjacent coarse cells.
    b. Sum up the transmissibilities in the fine cells along the down-stream edge.
    c. Divide the results from steps a by b. This upscales oil and water relative permeabilities in the direction corresponding to the transmissibility used in the calculations. This means that pseudos in the x direction will be generated when TRANX is used as weighting, while pseudos in the z direction will be generated when TRANZ is used as weighting.
5. Generate the pseudo oil and water relative permeabilities for each coarse block.
6. Calculate the average water saturation using pore volume weighting, as in the Kyte and Berry (1975).
7. Print out the pseudo functions to an output file and proceed to the next coarse cell. Then repeat steps 4 to 6.

➔ **Codes of the C++ program:**

```
//Codes to calculate TWR pseudo functions for a 2D cross-sectional model.

#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <iomanip>

using namespace std;

void Pseudos_x (int Ci, int Ck, int s, int ts, double pseudokro [50], double pseudokrw [50], double kro [50][50], double krw [50][50], double tranx [50][50]);

void Pseudos_z (int Ci, int Ck, int s, int ts, double pseudokro [50], double pseudokrw [50], double kro [50][50], double krw [50][50], double tranz [50][50]);

void Avg_Swat (int Ck, int Ci, int s, int ts, int ni, int nk, double AvgSwat [50] , double Sw [50][50], double Pv [50][50]);

int main ()
{
int nk, ni, ts, s;
string L;
double pseudokro [50], pseudokrw [50], swc,kroswc;
double kro [50][50],krw [50][50], tranx [50][50],tranz [50][50];
double Sw [50] [50],Pv [50][50], AvgSwat [50];

cout <<"Enter the following data:" <<endl;
cout <<"**************************" <<endl;
cout <<endl<< "No. of cells in the x direction:  ";
cin >> ni;
cout <<endl<< "No. of cells in the z direction:  ";
cin >> nk;
cout <<endl<< "No. of time steps:  ";
cin >> ts;
cout <<endl<< "Scale-up factor:  ";
```

```
cin >> s;
cout <<endl<< "Connate water saturation:   ";
cin>>swc;
cout <<endl<< "Oil relative permeability @ Swc:   ";
cin>>kroswc;

ofstream pseudo4eclipse_outfile ("Pseudos.txt");    //print out the file of pseudo functions.

//Generating pseudos in the x direction
for (int Ck = 1; Ck<nk; Ck=Ck+s)        // Ck is a counter for coarse cells in the k direction.
        {
for (int Ci = s; Ci<=ni; Ci=Ci+s)  // Ci is a counter for coarse cells in the i direction.
            {
//Open the required input files.
            ifstream kro_infile ("Kro.txt");
            ifstream krw_infile ("Krw.txt");
            ifstream tranx_infile ("Tranx.txt");
            ifstream Pv_infile ("Porevol.txt");
            ifstream Sw_infile ("Swat.txt");

//Skip header of 10 text lines in the input files.
            for (int t=0; t<10; t++)
            {
            getline (kro_infile,L);
            getline (krw_infile,L);
            getline (tranx_infile,L);
            getline (Pv_infile,L);
            getline (Sw_infile,L);
            }

 //Print out before each pseudo table: connate Sw value , Krw@Swc = 0, Kro@Swc value and Pc@Swc = 0.
pseudo4eclipse_outfile<<"--Pseudo Function in the x direction for Coarse Cell no.: ["<<(Ck+s-1)/s<<"],["<<Ci/s<<"]"<<endl;
```

```
pseudo4eclipse_outfile<<fixed<<swc<<"    "<< "0.00000"<<"    "<<fixed<<kroswc<<"
"<< "0.00000"<<endl;

//Loop for time steps within each coarse cell.
for (int tsteps = 1; tsteps <=ts; tsteps++ )
            {
for (int k = 1; k<=nk; k++)     // loop for fine cells in the k direction within a coarse cell.
              {
for (int i = 1; i <=ni; i++)    // loop for fine cells in the i direction within a coarse cell.
                {
// Read in data relevant to the timestep from the input data files into arrays.
                kro_infile>> kro [i][k];
                krw_infile>> krw [i][k];
                tranx_infile>>tranx [i][k];
                Pv_infile>>Pv [i] [k];
                Sw_infile>>Sw [i] [k];
                }
              }

Pseudos_x (Ci,Ck,s,ts,pseudokro,pseudokrw,kro,krw,tranx);
Avg_Swat (Ck,Ci,s,ts,ni,nk,AvgSwat,Sw,Pv);

//Arrange the pseudos to be used in Eclipse.
 if (AvgSwat [ts] > swc)
{
 pseudo4eclipse_outfile.precision (5);
pseudo4eclipse_outfile<<fixed<<AvgSwat [ts] <<"   " <<fixed<<pseudokrw [ts] <<"   "
<<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
cout<<fixed<<AvgSwat    [ts]    <<"          "    <<fixed<<pseudokrw    [ts]    <<"          "
<<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
 }

//skip 3 text lines between time steps in the input files.
for (int t=0; t<4; t++)
            {
            getline (kro_infile,L);
```

```
            getline (krw_infile,L);
            getline (tranx_infile,L);
            getline (Pv_infile,L);
            getline (Sw_infile,L);
            }
            } //end of time steps loop.

//print out after each pseudo table: Sw=1, Krw=1, Kro=0, and Pc=0. Afterwards, print slash(/).
pseudo4eclipse_outfile << "1.00000"<<"   "<< "1.00000"<<"   "<< "0.00000"<<"   "<< "0.00000"<<endl;
pseudo4eclipse_outfile << "/" <<endl;

}//end of Ci loop
}//end of Ck loop


// Generating pseudos in the z direction

for (int Ck = s; Ck<=nk; Ck=Ck+s) // Ck is a counter for coarse cells in the k direction.
        {
for (int Ci = 1; Ci<ni; Ci=Ci+s)  // Ci is a counter for coarse cells in the i direction.
        {
//Open the required input files.
            ifstream kro_infile ("Kro.txt");
            ifstream krw_infile ("Krw.txt");
            ifstream tranz_infile ("Tranz.txt");
            ifstream Pv_infile ("Porevol.txt");
            ifstream Sw_infile ("Swat.txt");

//Skip header of 10 text lines in the input files.
            for (int t=0; t<10; t++)
            {
            getline (kro_infile,L);
            getline (krw_infile,L);
            getline (tranz_infile,L);
            getline (Pv_infile,L);
            getline (Sw_infile,L);
```

```
            }

//Print out before each pseudo table: connate Sw value , Krw@Swc = 0, Kro@Swc value and Pc@Swc = 0.
pseudo4eclipse_outfile<<"--Pseudo Function in the z direction for Coarse Cell no.: ["<<Ck/s<<"],["<<(Ci+s-1)/s<<"]"<<endl;
pseudo4eclipse_outfile<<fixed<<swc<<"    "<< "0.00000"<<"    "<<fixed<<kroswc<<"    "<< "0.00000"<<endl;

//Loop for time steps within each coarse cell.
for (int tsteps = 1; tsteps <=ts; tsteps++ )
            {
for (int k = 1; k<=nk; k++)   // loop for fine cells in the k direction within a coarse cell.
              {
for (int i = 1; i <=ni; i++)    // loop for fine cells in I the direction within a coarse cell.
                 {
// Read in data from input files to arrays.
                 kro_infile>> kro [i][k];
                 krw_infile>> krw [i][k];
                 tranz_infile>>tranz [i][k];
                 Pv_infile>>Pv [i][k];
                 Sw_infile>>Sw [i][k];
                 }
              }

Pseudos_z (Ci,Ck,s,ts,pseudokro,pseudokrw,kro,krw,tranz);
Avg_Swat (Ck,Ci,s,ts,ni,nk,AvgSwat,Sw,Pv);

 //Arrange the pseudos to be used in Eclipse.
            if (AvgSwat [ts] > swc)
            {
pseudo4eclipse_outfile.precision (5);
pseudo4eclipse_outfile<<fixed<<AvgSwat [ts] <<"   " <<fixed<<pseudokrw [ts] <<"   "<<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
cout<<fixed<<AvgSwat [ts] <<"   " <<fixed<<pseudokrw [ts] <<"   "<<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
```

```
            }

//skip 3 text lines between time steps in the input files.
            for (int t=0; t<4; t++)
            {
            getline (kro_infile,L);
            getline (krw_infile,L);
            getline (tranz_infile,L);
            getline (Pv_infile,L);
            getline (Sw_infile,L);
            }
            }  //end of time steps loop.

//print out after each pseudo table, Sw=1, Krw=1, Kro=0, and Pc=0. Afterwards print slash(/).
pseudo4eclipse_outfile << "1.00000"<<"   "<< "1.00000"<<"   "<< "0.00000"<<"   "<< "0.00000"<<endl;
pseudo4eclipse_outfile << "/" <<endl;
        } //end of Ci loop
        } //end of Ck loop

cout<<endl;
cout <<"*************************************" <<endl;
cout <<"Pseudo functions in the x and z directions have been generated" <<endl;
cout <<"*************************************" <<endl<<endl;
}

// Function calculates pseudos in the x direction.
void Pseudos_x (int Ci,int Ck, int s, int ts, double pseudokro [50], double pseudokrw [50], double kro [50][50], double krw [50][50], double tranx [50][50])
        {
           double sumkrotranx =0;
           double sumkrwtranx =0;
           double sumtranx =0;
           double krotranx [50][50];
           double krwtranx [50][50];

for ( int k = Ck; k<Ck+s; k++)          // loop for fine cells in the k direction.
```

```
                {
krotranx [Ci][k] = kro [Ci][k] * tranx [Ci][k];    //calculates product of kro * tranx at down-stream edge of coarse cell.
krwtranx [Ci][k] = krw [Ci][k] * tranx [Ci][k];    //calculates product of krw * tranx at down-stream edge of coarse cell.
sumkrotranx = sumkrotranx + krotranx [Ci][k];     // calculates the sum (kro*tranx)
sumkrwtranx = sumkrwtranx + krwtranx [Ci][k];     //calculates the sum (krw*tranx)
sumtranx = sumtranx + tranx [Ci][k]; //sums tranx at down-stream edge of a coarse cell.
                }
if (sumtranx!=0)  //check that sumtranx is not zero.
                {
pseudokro [ts] = sumkrotranx / sumtranx;  //calculate pseudo kro using transmissibility weighting.
pseudokrw [ts] = sumkrwtranx / sumtranx;  //calculate pseudo krw using transmissibility weighting.
                }
                else
                {
pseudokro [ts] = 0;
pseudokrw [ts] = 0;
                }
sumkrotranx =0;
sumkrwtranx =0;
sumtranx =0;
                }

// Function calculates pseudos in the z direction.
void Pseudos_z (int Ci,int Ck, int s, int ts, double pseudokro [50], double pseudokrw [50], double kro [50][50], double krw [50][50], double tranz [50][50])
        {
            double sumkrotranz =0;
            double sumkrwtranz =0;
            double sumtranz =0;
            double krotranz [50][50];
            double krwtranz [50][50];
```

```
for ( int i = Ci; i<Ci+s; i++)
                    {
krotranz [i][Ck] = kro [i][Ck] * tranz [i][Ck];    //calculates product of kro * tranx at down-stream edge of coarse cell.
krwtranz [i][Ck] = krw [i][Ck] * tranz [i][Ck];    //calculates product of krw * tranx at down-stream edge of coarse cell.
sumkrotranz = sumkrotranz + krotranz [i][Ck];     // sums kro*tranx
sumkrwtranz = sumkrwtranz + krwtranz [i][Ck];     //sums krw*tranx
sumtranz = sumtranz + tranz [i][Ck];                //sum of tranx at down-stream edge of coarse cell.
                    }
              if (sumtranz!=0)  //check that sumtranz is not zero.
                 {

pseudokro [ts] = sumkrotranz / sumtranz;  //calculate pseudo kro using transmissibility weighting.
pseudokrw [ts] = sumkrwtranz / sumtranz;  //calculate pseudo krw using transmissibility weighting.
                 }
                 else
                 {
                 pseudokro [ts] = 0;
                 pseudokrw [ts] = 0;
                 }
              sumkrotranz =0;
              sumkrwtranz =0;
              sumtranz =0;
                }

//Function calculates average water saturation in coarse cells.
void Avg_Swat (int Ck, int Ci,int s,int ts,int ni, int nk, double AvgSwat [50] , double Sw [50][50], double Pv [50][50])
        {
        double sumswpv = 0;
        double sumpv = 0;
        double swpv = 0;
```

```
for (int k = Ck; k<Ck+s; k++)          // loop for fine cells in the k direction within a coarse cell.
        {
for (int i = Ci-s+1; i <Ci+1; i++)    // loop for fine cells in the i direction within a coarse cell.
          {
sumpv = sumpv + Pv [i] [k]; // calculates sum of pore volume.
swpv = Sw [i][k] * Pv [i][k]; //calculates the product: water saturation * pore volume.
sumswpv = sumswpv + swpv; // calculates sum of Sw * Pv
          }
        }
      AvgSwat [ts] = sumswpv/sumpv;    //calculation of avg. Sw using pore volume weighting.
      }
```

//END of PROGRAM

C. C++ program to generate directional [positive] TWR pseudo functions for SPE 10 model 2 (4 layers):

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <iomanip>

using namespace std;

void Pseudos_x (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double tranx [60][220][4]);

void Pseudos_y (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double trany [60][220][4]);

void Pseudos_z (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double tranz [60][220][4]);

void Avg_Swatx (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwatx [100] , double Sw [60][220][4], double Pv [60][220][4]);

void Avg_Swaty (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwaty [100], double Sw [60][220][4], double Pv [60][220][4]);

void Avg_Swatz (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwatz [100], double Sw [60][220][4], double Pv [60][220][4]);

int main ()
        {
int nk, ni,nj, ts, s, injI, injJ;
string L;
double static pseudokro [100], pseudokrw [100], swc,kroswc;
double static kro [60][220][4],krw [60][220][4], tranx [60][220][4],trany [60][220][4],tranz [60][220][4];
```

```
double static Sw [60][220][4],Pv [60][220][4], AvgSwatx [100],AvgSwaty [100],AvgSwatz [100];


cout <<"Enter the following data:" <<endl;
cout <<"**************************" <<endl;
cout <<endl<< "No. of cells in the x direction:  ";
cin >> ni;
cout <<endl<< "No. of cells in the y direction:  ";
cin >> nj;
cout <<endl<< "No. of cells in the z direction:  ";
cin >> nk;
cout <<endl<< "No. of time steps:  ";
cin >> ts;
cout <<endl<<"Scale-up factor:  ";
cin >> s;
cout <<endl<< "Connate water saturation:   ";
cin>>swc;
cout <<endl<< "Oil relative permeability @ Swc:   ";
cin>>kroswc;

ofstream pseudo4eclipse_outfile ("Pseudos.txt"); // print out file of pseudo functions.

//Generation of pseudos in the x+ direction
for (int Ck = 1; Ck<nk; Ck=Ck+s) //Ck is a counter for coarse cells in the k direction.
        {
for (int Cj = 1; Cj<nj; Cj=Cj+s)  //Cj is a counter for coarse cells in the j direction.
            {
for (int Ci = s; Ci<ni; Ci=Ci+s)  //Ci is a counter for coarse cells in the i direction.
                {
//Open the required input files.
            ifstream kro_infile ("Kro.txt");
            ifstream krw_infile ("Krw.txt");
            ifstream tranx_infile ("Tranx.txt");
            ifstream Pv_infile ("Porevol.txt");
            ifstream Sw_infile ("Swat.txt");
```

```
//Skip header of 10 text lines in the input files.
        for (int t=0; t<10; t++)
        {
        getline (kro_infile,L);
        getline (krw_infile,L);
        getline (tranx_infile,L);
        getline (Pv_infile,L);
        getline (Sw_infile,L);
        }

//Print out before each pseudo table: connate Sw value , Krw@Swc = 0, Kro@Swc value and Pc@Swc = 0.
pseudo4eclipse_outfile<<"--Pseudo Function in the x+ direction for Coarse Cell no.:"<<"["<<Ci/s<<"],["<<(Cj+s-1)/s<<"],["<<(Ck+s-1)/s<<"]"<<endl;
pseudo4eclipse_outfile<<fixed<<swc<<"  "<< "0.00000"<<"  "<<fixed<<kroswc<<" "<< "0.00000"<<endl;

//Loop for time steps within each coarse cell.
for (int tsteps = 1; tsteps <=ts; tsteps++ )
        {
for (int k = 1; k<=nk; k++)     // loop for fine cells in the k direction within a coarse cell.
          {
for (int j = 1; j <=nj; j++)    // loop for fine cells in the j direction within a coarse cell.
            {
for (int i = 1; i <=ni; i++)    // loop for fine cells in the i direction within a coarse cell.
              {

// Read in data relevant to timestep from input files into arrays.
        kro_infile>> kro [i][j][k];
        krw_infile>> krw [i][j][k];
        tranx_infile>>tranx [i][j][k];
        Pv_infile>>Pv [i][j][k];
        Sw_infile>>Sw [i][j][k];
            }
            }
            }
```

```
Pseudos_x (Ci,Ck,Cj,s,ts,pseudokro,pseudokrw,kro,krw,tranx);
Avg_Swatx (Ck,Ci,Cj,s,ts,ni,nj,nk,AvgSwatx,Sw,Pv);

//Arrange the pseudos to be used in Eclipse.
if (AvgSwatx [ts] > swc)
        {
pseudo4eclipse_outfile.precision (5);
pseudo4eclipse_outfile<<fixed<<AvgSwatx [ts] <<"    " <<fixed<<pseudokrw [ts] <<"
" <<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
cout<<fixed<<AvgSwatx [ts] <<"        " <<fixed<<pseudokrw [ts] <<"        "
<<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
        }

//skip 3 text lines between time steps in the input files.
    for (int t=0; t<4; t++)
      {
      getline (kro_infile,L);
      getline (krw_infile,L);
      getline (tranx_infile,L);
      getline (Pv_infile,L);
      getline (Sw_infile,L);
      }
      } //end of time steps loop.

//print out after each pseudo table, Sw=1, Krw=1, Kro=0, and Pc=0. Afterwards print
slash(/).
pseudo4eclipse_outfile << "1.00000"<<"   "<< "1.00000"<<"   "<< "0.00000"<<"   "<<
"0.00000"<<endl;
pseudo4eclipse_outfile << "/" <<endl;


    }//end of Ci loop
    }
    }//end of Ck loop

// Generation of pseudos in the y+ direction
for (int Ck = 1; Ck<nk; Ck=Ck+s) // Ck is a counter for coarse cells in the k direction.
    {
```

```
for (int Cj = s; Cj<nj; Cj=Cj+s)  // Cj is a counter for coarse cells in the j direction.
        {
for (int Ci = 1; Ci<ni; Ci=Ci+s)  // Ci is a counter for coarse cells in the i direction.
          {

//Open the required input files.
        ifstream kro_infile ("Kro.txt");
        ifstream krw_infile ("Krw.txt");
        ifstream trany_infile ("Trany.txt");
        ifstream Pv_infile ("Porevol.txt");
        ifstream Sw_infile ("Swat.txt");

//Skip header of 10 text lines in the input files.
        for (int t=0; t<10; t++)
        {
        getline (kro_infile,L);
        getline (krw_infile,L);
        getline (trany_infile,L);
        getline (Pv_infile,L);
        getline (Sw_infile,L);
        }

//Print out before each pseudo table: connate Sw value , Krw@Swc = 0, Kro@Swc value and Pc@Swc = 0.
pseudo4eclipse_outfile<<"--Pseudo Function in the y+ direction for Coarse Cell no.:"<<"["<<(Ci+s-1)/s<<"],["<<Cj/s<<"],["<<(Ck+s-1)/s<<"]"<<endl;
pseudo4eclipse_outfile<<fixed<<swc<<"   "<< "0.00000"<<"   "<<fixed<<kroswc<<" "<< "0.00000"<<endl;

//Loop for time steps within each coarse cell.
for (int tsteps = 1; tsteps <=ts; tsteps++ )
        {
for (int k = 1; k<=nk; k++)    // loop for fine cells in the k direction within a coarse cell.
          {

for (int j = 1; j <=nj; j++)    // loop for fine cells in the j direction within a coarse cell.
            {
for (int i = 1; i <=ni; i++)    // loop for fine cells in the i direction within a coarse cell.
```

```
                {

// Read in data relevant to timestep from input files into arrays.
                kro_infile>> kro [i][j][k];
                krw_infile>> krw [i][j][k];
                trany_infile>>trany [i][j][k];
                Pv_infile>>Pv [i][j][k];
                Sw_infile>>Sw [i][j][k];
                        }
                        }
                        }

Pseudos_y (Ci, Ck, Cj, s, ts, pseudokro, pseudokrw, kro, krw, trany);
Avg_Swaty (Ck, Ci, Cj, s, ts, ni, nj, nk, AvgSwaty, Sw, Pv);

//Arrange the pseudos to be used in Eclipse.
                if (AvgSwaty [ts] > swc)
                {
pseudo4eclipse_outfile.precision (5);
pseudo4eclipse_outfile<<fixed<<AvgSwaty [ts] <<"    " <<fixed<<pseudokrw [ts] <<"
" <<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
cout<<fixed<<AvgSwaty    [ts]    <<"          "    <<fixed<<pseudokrw    [ts]    <<"          "
<<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
                }

//skip 3 text lines between time steps in the input files.
                for (int t=0; t<4; t++)
                {
                getline (kro_infile,L);
                getline (krw_infile,L);
                getline (trany_infile,L);
                getline (Pv_infile,L);
                getline (Sw_infile,L);
                }
                } //end of time steps loop.

//print out after each pseudo table, Sw=1, Krw=1, Kro=0, and Pc=0. Afterwards print
slash(/).
```

```
pseudo4eclipse_outfile << "1.00000"<<"    "<< "1.00000"<<"    "<< "0.00000"<<"    "<<
"0.00000"<<endl;
pseudo4eclipse_outfile << "/" <<endl;


        }//end of Ci loop
        }
        }//end of Ck loop

// Generation of pseudos in the z+ direction
   for (int Ck = s; Ck<nk; Ck=Ck+s) // Ck is a counter for coarse cells in the k direction.
        {

for (int Cj = 1; Cj<nj; Cj=Cj+s)  // Cj is a counter for coarse cells in the j direction.
            {

 for (int Ci = 1; Ci<ni; Ci=Ci+s)  // Ci is a counter for coarse cells in the i direction.
                {

//Open the required input files.
            ifstream kro_infile ("Kro.txt");
            ifstream krw_infile ("Krw.txt");
            ifstream tranz_infile ("Tranx.txt");
            ifstream Pv_infile ("Porevol.txt");
            ifstream Sw_infile ("Swat.txt");

//Skip header of 10 text lines in the input files.
            for (int t=0; t<10; t++)
            {
            getline (kro_infile,L);
            getline (krw_infile,L);
            getline (tranz_infile,L);
            getline (Pv_infile,L);
            getline (Sw_infile,L);
            }

//Print out before each pseudo table: connate Sw value , Krw@Swc = 0, Kro@Swc
value and Pc@Swc = 0.
```

```
pseudo4eclipse_outfile<<"--Pseudo Function in the z+ direction for Coarse Cell no.:"<<"["<<(Ci+s-1)/s<<"],["<<(Cj+s-1)/s<<"],["<<Ck/s<<"]"<<endl;
pseudo4eclipse_outfile<<fixed<<swc<<"    "<< "0.00000"<<"    "<<fixed<<kroswc<<" "<< "0.00000"<<endl;

//Loop for time steps within each coarse cell.
for (int tsteps = 1; tsteps <=ts; tsteps++ )
            {
for (int k = 1; k<=nk; k++)   // loop for fine cells in the k direction within a coarse cell.
              {
for (int j = 1; j <=nj; j++)    // loop for fine cells in the j direction within a coarse cell.
                {
for (int i = 1; i <=ni; i++)    // loop for fine cells in the i direction within a coarse cell.
                {

// Read in data relevant to timestep from input files into arrays.
          kro_infile>> kro [i][j][k];
          krw_infile>> krw [i][j][k];
          tranz_infile>>tranx [i][j][k];
          Pv_infile>>Pv [i][j][k];
          Sw_infile>>Sw [i][j][k];
                }
                }
                }

Pseudos_z (Ci,Ck,Cj,s,ts,pseudokro,pseudokrw,kro,krw,tranx);
Avg_Swatz (Ck,Ci,Cj,s,ts,ni,nj,nk,AvgSwatz,Sw,Pv);

//Arrange the pseudos to be used in Eclipse.
            if (AvgSwatz [ts] > swc)
            {

pseudo4eclipse_outfile.precision (5);

pseudo4eclipse_outfile<<fixed<<AvgSwatz [ts] <<"    " <<fixed<<pseudokrw [ts] <<" " <<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
cout<<fixed<<AvgSwatz [ts] <<" "<<fixed<<pseudokrw [ts] <<" " <<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
```

```
        }

//skip 3 text lines between time steps in the input files.
        for (int t=0; t<4; t++)
        {
        getline (kro_infile,L);
        getline (krw_infile,L);
        getline (tranz_infile,L);
        getline (Pv_infile,L);
        getline (Sw_infile,L);
        }
        }//end of time steps loop.

//print out after each pseudo table, Sw=1, Krw=1, Kro=0, and Pc=0. Afterwards print slash(/).
pseudo4eclipse_outfile << "1.00000"<<"   "<< "1.00000"<<"   "<< "0.00000"<<"   "<< "0.00000"<<endl;
pseudo4eclipse_outfile << "/" <<endl;


        }//end of Ci loop
        }
        }//end of Ck loop

cout<<endl;
cout <<"*************************************" <<endl;
cout <<"Pseudo functions in the x+, y+ and z+ directions have been generated" <<endl;
cout <<"*************************************" <<endl<<endl;
        }

// Function calculates pseudos in coarse cells in the i direction.
void Pseudos_x (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double tranx [60][220][4])
        {
        double sumkrotranx =0;

double sumkrwtranx =0;
double sumtranx =0;
```

```
double static krotranx [60][220][4];
double static krwtranx [60][220][4];

for ( int k = Ck; k<Ck+s; k++)          // loop for fine cells in the k direction.
                {
              for ( int j = Cj; j<Cj+s; j++)          // loop for fine cells in the j direction.
                {

krotranx [Ci][j][k] = kro [Ci][j][k] * tranx [Ci][j][k];   //calculates product of kro * tranx at the down-stream edge of a coarse cell.
krwtranx [Ci][j][k] = krw [Ci][j][k] * tranx [Ci][j][k];    //calculates product of krw * tranx at the down-stream edge of a coarse cell.
          sumkrotranx = sumkrotranx + krotranx [Ci][j][k];     // sums kro*tranx
          sumkrwtranx = sumkrwtranx + krwtranx [Ci][j][k];     //sums krw*tranx

sumtranx = sumtranx + tranx [Ci][j][k];  //sum of tranx at the down-stream edge of a coarse cell.
                }
                }

if (sumtranx!=0)  //check that sumtranx is not zero.
                {

pseudokro [ts] = sumkrotranx / sumtranx;  //calculate pseudo kro using transmissibility weighting.
pseudokrw [ts] = sumkrwtranx / sumtranx;  //calculate pseudo krw using transmissibility weighting.
                }
                else
                {
                pseudokro [ts] = 0;
                pseudokrw [ts] = 0;
                }
sumkrotranx =0;
sumkrwtranx =0;
            sumtranx =0;
          }
```

```
// Function calculates pseudos in coarse cell in the j direction
void Pseudos_y (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double trany [60][220][4])
        {
           double sumkrotrany =0;
           double sumkrwtrany =0;
           double sumtrany =0;
           double static krotrany [60][220][4];
           double static krwtrany [60][220][4];

for ( int k = Ck; k<Ck+s; k++)          // loop for fine cells in the k direction.
                    {
for ( int i = Ci; i<Ci+s; i++)          // loop for fine cells in the i direction.
                    {

krotrany [i][Cj][k] = kro [i][Cj][k] * trany [i][Cj][k];   //calculates product of kro * tranx at down-stream edge of coarse cell.
krwtrany [i][Cj][k] = krw [i][Cj][k] * trany [i][Cj][k];    //calculates product of krw * tranx at down-stream edge of coarse cell.
                    sumkrotrany = sumkrotrany + krotrany [i][Cj][k];     // sums kro*tranx
                    sumkrwtrany = sumkrwtrany + krwtrany [i][Cj][k];     //sums krw*tranx

sumtrany = sumtrany + trany [i][Cj][k];  //sum of tranx at the down-stream edge of a coarse cell.
                    }
                    }
                  if (sumtrany!=0)  //check that sumtranx is not zero.
                  {

pseudokro [ts] = sumkrotrany / sumtrany;  //calculate pseudo kro using transmissibility weighting.
pseudokrw [ts] = sumkrwtrany / sumtrany;  //calculate pseudo krw using transmissibility weighting.
                  }
                  else
                  {
```

```
            pseudokro [ts] = 0;
            pseudokrw [ts] = 0;
            }
        sumkrotrany =0;
        sumkrwtrany =0;
        sumtrany =0;
    }

// Function calculates pseudos in coarse cell in the z+ direction
void Pseudos_z (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double tranz [60][220][4])
    {
        double sumkrotranz =0;
        double sumkrwtranz =0;
        double sumtranz =0;
        double static krotranz [60][220][4];
        double static krwtranz [60][220][4];

            for ( int j = Cj; j<Cj+s; j++)          // loop for fine cells in the k direction.
            {
                for ( int i = Ci; i<Ci+s; i++)       // loop for fine cells in the k direction.
            {

krotranz [i][j][Ck] = kro [i][j][Ck] * tranz [i][j][Ck];   //calculates product of kro * tranx at down-stream edge of coarse cell.
krwtranz [i][j][Ck] = krw [i][j][Ck] * tranz [i][j][Ck];    //calculates product of krw * tranx at down-stream edge of coarse cell.
                sumkrotranz = sumkrotranz + krotranz [i][j][Ck];     // sums kro*tranx
                sumkrwtranz = sumkrwtranz + krwtranz [i][j][Ck];     //sums krw*tranx

sumtranz = sumtranz + tranz [i][j][Ck];  //sum of tranx at the down-stream edge of a coarse cell.
                }
                }
            if (sumtranz!=0)  //check that sumtranx is not zero.
            {
```

```
        pseudokro [ts] = sumkrotranz / sumtranz;  //calculate pseudo kro using transmissibility weighting.
        pseudokrw [ts] = sumkrwtranz / sumtranz;  //calculate pseudo krw using transmissibility weighting.
        }
        else
        {
        pseudokro [ts] = 0;
        pseudokrw [ts] = 0;
        }
    sumkrotranz =0;
    sumkrwtranz =0;
    sumtranz =0;
}

//Function calculates average water saturation in coarse cells in the i direction.
void Avg_Swatx (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwatx [100], double Sw [60][220][4], double Pv [60][220][4])
{
double sumswpv = 0;
double sumpv = 0;
double swpv = 0;

for (int k = Ck; k<Ck+s; k++)        // loop for fine cells in the k direction within a coarse cell.
    {
      for (int j = Cj; j<Cj+s; j++)          // loop for fine cells in the  j direction within a coarse cell.
    {
    for (int i = Ci-s+1; i <Ci+1; i++)     // loop for fine cells in the i direction within a coarse cell.
        {
sumpv = sumpv + Pv [i][j][k]; // calculates sum of pore volume.
swpv = Sw [i][j][k] * Pv [i][j][k]; //calculates the product: water saturation * pore volume.
```

```
sumswpv = sumswpv + swpv; // calculates sum of Sw * Pv
            }
            }
            }

AvgSwatx [ts] = sumswpv/sumpv;     //calculation of avg. Sw using pore volume weighting.
        }

//Function calculates average water saturation in coarse cells in the j direction
void Avg_Swaty (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwaty [100], double Sw [60][220][4], double Pv [60][220][4])
        {
        double sumswpv = 0;
        double sumpv = 0;
        double swpv = 0;


for (int k = Ck; k<Ck+s; k++) // loop for fine cells in the k direction within a coarse cell.
            {

for (int j = Cj-s+1; j<Cj+1; j++) // loop for fine cells in the j direction within a coarse cell.
            {

for (int i = Ci; i <Ci+s; i++)    // loop for fine cells in the i direction within a coarse cell.
                {
        sumpv = sumpv + Pv [i][j][k]; // calculates sum of pore volume.
        swpv = Sw [i][j][k] * Pv [i][j][k]; //calculates the product: water saturation * pore volume.
                sumswpv = sumswpv + swpv; // calculates sum of Sw * Pv
                }
            }
            }
        AvgSwaty [ts] = sumswpv/sumpv;    //calculation of avg. Sw using pore volume weighting.
        }
```

```
//Function calculates average water saturation in coarse cells in the k direction
void Avg_Swatz (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwatz [100], double Sw [60][220][4], double Pv [60][220][4])
        {
        double sumswpv = 0;
        double sumpv = 0;
        double swpv = 0;

for (int k = Ck-s+1; k<Ck+1; k++)          // loop for fine cells in the k direction within a coarse cell.
            {

for (int j = Cj; j<Cj+s; j++)          // loop for fine cells in the j direction within a coarse cell.
            {

for (int i = Ci; i <Ci+s; i++)    // loop for fine cells in the i direction within a coarse cell.
              {

sumpv = sumpv + Pv [i][j][k]; // calculates sum of pore volume.
swpv = Sw [i][j][k] * Pv [i][j][k]; //calculates the product: water saturation * pore volume.
sumswpv = sumswpv + swpv; // calculates sum of Sw * Pv
            }
            }
            }
        AvgSwatz [ts] = sumswpv/sumpv;    //calculation of avg. Sw using pore volume weighting.
        }
```

//END of Program

D. C++ program to generate directional [negative] TWR pseudo functions for SPE 10 model 2 (4 layers):

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <iomanip>

using namespace std;

void Pseudos_x (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double tranx [60][220][4]);

void Pseudos_y (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double trany [60][220][4]);

void Pseudos_z (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double tranz [60][220][4]);

void Avg_Swatx (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwatx [100] , double Sw [60][220][4], double Pv [60][220][4]);

void Avg_Swaty (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwaty [100], double Sw [60][220][4], double Pv [60][220][4]);

void Avg_Swatz (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwatz [100], double Sw [60][220][4], double Pv [60][220][4]);

int main ()
        {
int nk, ni,nj, ts, s, injI, injJ;
string L;
double static pseudokro [100], pseudokrw [100], swc,kroswc;
double static kro [60][220][4],krw [60][220][4], tranx [60][220][4],trany [60][220][4],tranz [60][220][4];
```

```
double static Sw [60][220][4],Pv [60][220][4], AvgSwatx [100],AvgSwaty [100],AvgSwatz [100];

cout <<"Enter the following data:" <<endl;
cout <<"**************************" <<endl;
cout <<endl<< "No. of cells in the x direction:  ";
cin >> ni;
cout <<endl<< "No. of cells in the y direction:  ";
cin >> nj;
cout <<endl<< "No. of cells in the z direction:  ";
cin >> nk;
cout <<endl<< "No. of time steps:  ";
cin >> ts;
cout <<endl<<"Scale-up factor:  ";
cin >> s;
cout <<endl<< "Connate water saturation:   ";
cin>>swc;
cout <<endl<< "Oil relative permeability @ Swc:   ";
cin>>kroswc;

ofstream pseudo4eclipse_outfile ("Pseudos.txt"); // print out file of pseudo functions.

//Generating pseudos in the x- direction

for (int Ck = 1; Ck<nk; Ck=Ck+s) // Ck is a counter for coarse cells in the k direction.
        {
for (int Cj = 1; Cj<nj; Cj=Cj+s)  // Cj is a counter for coarse cells in the j direction.
        {
for (int Ci = 1; Ci<ni; Ci=Ci+s)  // Ci is a counter for coarse cells in the i direction.
        {
//Open the required input files.
        ifstream kro_infile ("Kro.txt");
        ifstream krw_infile ("Krw.txt");
        ifstream tranx_infile ("Tranx.txt");
        ifstream Pv_infile ("Porevol.txt");
        ifstream Sw_infile ("Swat.txt");
```

```
//Skip header of 10 text lines in the input files.
        for (int t=0; t<10; t++)
           {
           getline (kro_infile,L);
           getline (krw_infile,L);
           getline (tranx_infile,L);
           getline (Pv_infile,L);
           getline (Sw_infile,L);
           }
//Print out before each pseudo table: connate Sw value , Krw@Swc = 0, Kro@Swc value and Pc@Swc = 0.
pseudo4eclipse_outfile<<"--Pseudo Function in the x- direction for Coarse Cell no.:"<<"["<<(Ci+s-1)/s<<"],["<<(Cj+s-1)/s<<"],["<<(Ck+s-1)/s<<"]"<<endl;
pseudo4eclipse_outfile<<fixed<<swc<<"   "<< "0.00000"<<"   "<<fixed<<kroswc<<" "<< "0.00000"<<endl;

//Loop for time steps within each coarse cell.
for (int tsteps = 1; tsteps <=ts; tsteps++ )
           {
for (int k = 1; k<=nk; k++)    // loop for fine cells in the k direction within a coarse cell.
             {
for (int j = 1; j <=nj; j++)    // loop for fine cells in the j direction within a coarse cell.
               {
for (int i = 1; i <=ni; i++)    // loop for fine cells in the i direction within a coarse cell.
                 {
// Read in data relevant to timestep from input files into arrays.
                   kro_infile>> kro [i][j][k];
                   krw_infile>> krw [i][j][k];
                   tranx_infile>>tranx [i][j][k];
                   Pv_infile>>Pv [i][j][k];
                   Sw_infile>>Sw [i][j][k];
                   }
               }
           }
```

```
Pseudos_x (Ci,Ck,Cj,s,ts,pseudokro,pseudokrw,kro,krw,tranx);
Avg_Swatx (Ck,Ci,Cj,s,ts,ni,nj,nk,AvgSwatx,Sw,Pv);

//Arrange the pseudos to be used in Eclipse.
        if (AvgSwatx [ts] > swc)
        {
        pseudo4eclipse_outfile.precision (5);

pseudo4eclipse_outfile<<fixed<<AvgSwatx [ts] <<"    " <<fixed<<pseudokrw [ts] <<"
" <<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
cout<<fixed<<AvgSwatx [ts] <<"        " <<fixed<<pseudokrw [ts] <<"        "
<<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
        }

//skip 3 text lines between time steps in the input files.
        for (int t=0; t<4; t++)
        {
        getline (kro_infile,L);
        getline (krw_infile,L);
        getline (tranx_infile,L);
        getline (Pv_infile,L);
        getline (Sw_infile,L);
        }
        }//end of time steps loop.

//print out after each pseudo table, Sw=1, Krw=1, Kro=0, and Pc=0. Afterwards print
slash(/).
pseudo4eclipse_outfile << "1.00000"<<"   "<< "1.00000"<<"   "<< "0.00000"<<"   "<<
"0.00000"<<endl;
pseudo4eclipse_outfile << "/" <<endl;


    }//end of Ci loop
    }
    }//end of Ck loop
```

```
//Generate pseudos in the y- direction
for (int Ck = 1; Ck<nk; Ck=Ck+s) // Ck is a counter for coarse cells in the k direction.
        {
for (int Cj = 1; Cj<nj; Cj=Cj+s)  // Cj is a counter for coarse cells in the j direction.
          {
for (int Ci = 1; Ci<ni; Ci=Ci+s)  // Ci is a counter for coarse cells in the i direction.
            {

//Open the required input files.
            ifstream kro_infile ("Kro.txt");
            ifstream krw_infile ("Krw.txt");
            ifstream trany_infile ("Trany.txt");
            ifstream Pv_infile ("Porevol.txt");
            ifstream Sw_infile ("Swat.txt");

//Skip header of 10 text lines in the input files.
            for (int t=0; t<10; t++)
            {
            getline (kro_infile,L);
            getline (krw_infile,L);
            getline (trany_infile,L);
            getline (Pv_infile,L);
            getline (Sw_infile,L);
            }

//Print out before each pseudo table: connate Sw value , Krw@Swc = 0, Kro@Swc value and Pc@Swc = 0.
pseudo4eclipse_outfile<<"--Pseudo Function in the y- direction for Coarse Cell no.:"<<"["<<(Ci+s-1)/s<<"],["<<(Cj+s-1)/s<<"],["<<(Ck+s-1)/s<<"]"<<endl;
pseudo4eclipse_outfile<<fixed<<swc<<"   "<< "0.00000"<<"   "<<fixed<<kroswc<<"   "<< "0.00000"<<endl;

//Loop for time steps within each coarse cell.
for (int tsteps = 1; tsteps <=ts; tsteps++ )
            {
for (int k = 1; k<=nk; k++)    // loop for fine cells in the k direction within a coarse cell.
              {
```

```
for (int j = 1; j <=nj; j++)    // loop for fine cells in the j direction within a coarse cell.
                {
for (int i = 1; i <=ni; i++)    // loop for fine cells in the i direction within a coarse cell.
                  {
// Read in data relevant to timestep from input files into arrays.
                  kro_infile>> kro [i][j][k];
                  krw_infile>> krw [i][j][k];
                  trany_infile>>trany [i][j][k];
                  Pv_infile>>Pv [i][j][k];
                  Sw_infile>>Sw [i][j][k];
                  }
                 }
                }

Pseudos_y (Ci,Ck,Cj,s,ts,pseudokro,pseudokrw,kro,krw,trany);
Avg_Swaty (Ck,Ci,Cj,s,ts,ni,nj,nk,AvgSwaty,Sw,Pv);


//Arrange the pseudos to be used in Eclipse.
            if (AvgSwaty [ts] > swc)
            {
pseudo4eclipse_outfile.precision (5);
pseudo4eclipse_outfile<<fixed<<AvgSwaty [ts] <<"    " <<fixed<<pseudokrw [ts] <<"
" <<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
cout<<fixed<<AvgSwaty   [ts]   <<"          "   <<fixed<<pseudokrw   [ts]   <<"          "
<<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
            }

//skip 3 text lines between time steps in the input files.
            for (int t=0; t<4; t++)
            {
            getline (kro_infile,L);
            getline (krw_infile,L);
            getline (trany_infile,L);
            getline (Pv_infile,L);
            getline (Sw_infile,L);
            }
```

```
        } //end of time steps loop.

//print out after each pseudo table, Sw=1, Krw=1, Kro=0, and Pc=0. Afterwards print slash(/).
pseudo4eclipse_outfile << "1.00000"<<"   "<< "1.00000"<<"   "<< "0.00000"<<"   "<< "0.00000"<<endl;
pseudo4eclipse_outfile << "/" <<endl;
        }//end of Ci loop
        }
        }//end of Ck loop

//Generating pseudos in the z- direction.
for (int Ck = 1; Ck<nk; Ck=Ck+s) // Ck is a counter for coarse cells in the k direction.
        {
for (int Cj = 1; Cj<nj; Cj=Cj+s)  // Cj is a counter for coarse cells in the j direction.
          {
for (int Ci = 1; Ci<ni; Ci=Ci+s)  // Ci is a counter for coarse cells in the i direction.
            {

//Open the required input files.
          ifstream kro_infile ("Kro.txt");
          ifstream krw_infile ("Krw.txt");
          ifstream tranz_infile ("Tranx.txt");
          ifstream Pv_infile ("Porevol.txt");
          ifstream Sw_infile ("Swat.txt");

//Skip header of 10 text lines in the input files.
          for (int t=0; t<10; t++)
          {
          getline (kro_infile,L);
          getline (krw_infile,L);
          getline (tranz_infile,L);
          getline (Pv_infile,L);
          getline (Sw_infile,L);
          }

//Print out before each pseudo table: connate Sw value , Krw@Swc = 0, Kro@Swc value and Pc@Swc = 0.
```

```
pseudo4eclipse_outfile<<"--Pseudo Function in the z- direction for Coarse Cell no.:"<<"["<<(Ci+s-1)/s<<"],["<<(Cj+s-1)/s<<"],["<<(Ck+s-1)/s<<"]"<<endl;
pseudo4eclipse_outfile<<fixed<<swc<<"    "<< "0.00000"<<"    "<<fixed<<kroswc<<"    "<< "0.00000"<<endl;

//Loop for time steps within each coarse cell.
for (int tsteps = 1; tsteps <=ts; tsteps++ )
            {
for (int k = 1; k<=nk; k++)   // loop for fine cells in the k direction within a coarse cell.
              {
for (int j = 1; j <=nj; j++)    // loop for fine cells in the j direction within a coarse cell.
                {
for (int i = 1; i <=ni; i++)    // loop for fine cells in the i direction within a coarse cell.
                {

// Read in data relevant to timestep from input files into arrays.
                kro_infile>> kro [i][j][k];
                krw_infile>> krw [i][j][k];
                tranz_infile>>tranx [i][j][k];
                Pv_infile>>Pv [i][j][k];
                Sw_infile>>Sw [i][j][k];
                }
              }
            }

Pseudos_z (Ci,Ck,Cj,s,ts,pseudokro,pseudokrw,kro,krw,tranx);
Avg_Swatz (Ck,Ci,Cj,s,ts,ni,nj,nk,AvgSwatz,Sw,Pv);

//Arrange the pseudos to be used in Eclipse.
            if (AvgSwatz [ts] > swc)
            {
            pseudo4eclipse_outfile.precision (5);

pseudo4eclipse_outfile<<fixed<<AvgSwatz [ts] <<"    " <<fixed<<pseudokrw [ts] <<" " <<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
cout<<fixed<<AvgSwatz [ts] <<"    " <<fixed<<pseudokrw [ts] <<"    " <<fixed<<pseudokro [ts] <<"   "<<"0.00000"<<endl;
```

```
            }
//skip 3 text lines between time steps in the input files.
            for (int t=0; t<4; t++)
            {
            getline (kro_infile,L);
            getline (krw_infile,L);
            getline (tranz_infile,L);
            getline (Pv_infile,L);
            getline (Sw_infile,L);
            }
            }//end of time steps loop.

//print out after each pseudo table, Sw=1, Krw=1, Kro=0, and Pc=0. Afterwards print slash(/).
pseudo4eclipse_outfile << "1.00000"<<"   "<< "1.00000"<<"   "<< "0.00000"<<"   "<< "0.00000"<<endl;
pseudo4eclipse_outfile << "/" <<endl;
        }//end of Ci loop
        }
        }//end of Ck loop

cout<<endl;
cout <<"**********************************************************" <<endl;
cout <<"TWR pseudos in the x- , y- and z- directions have been generated" <<endl;
cout<<"**********************************************************"
<<endl<<endl;
        }

// Function calculates pseudos in the x- direction.
void Pseudos_x (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double tranx [60][220][4])
        {

double sumkrotranx =0;
            double sumkrwtranx =0;
            double sumtranx =0;
```

```
            double static krotranx [60][220][4];
            double static krwtranx [60][220][4];

for ( int k = Ck; k<Ck+s; k++)          // loop for fine cells in the k direction.
                    {

for ( int j = Cj; j<Cj+s; j++)          // loop for fine cells in the k direction.
                    {

krotranx [Ci][j][k] = kro [Ci][j][k] * tranx [Ci][j][k];   //calculates product of kro * tranx at the down-stream edge of a coarse cell.
krwtranx [Ci][j][k] = krw [Ci][j][k] * tranx [Ci][j][k];    //calculates product of krw * tranx at the down-stream edge of a coarse cell.
sumkrotranx = sumkrotranx + krotranx [Ci][j][k];     // sums kro*tranx
sumkrwtranx = sumkrwtranx + krwtranx [Ci][j][k];     //sums krw*tranx
sumtranx = sumtranx + tranx [Ci][j][k]; //sum of tranx at the down-stream edge of a coarse cell.
                    }
                    }

if (sumtranx!=0)  //check that sumtranx is not zero.
                {
pseudokro [ts] = sumkrotranx / sumtranx;  //calculate pseudo kro using transmissibility weighting.
pseudokrw [ts] = sumkrwtranx / sumtranx;  //calculate pseudo krw using transmissibility weighting.
                }
                else
                {
          pseudokro [ts] = 0;
          pseudokrw [ts] = 0;
                }
          sumkrotranx =0;
          sumkrwtranx =0;

sumtranx =0;
          }
```

```
// Function calculates pseudos in the y- direction.
void Pseudos_y (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double trany [60][220][4])
        {
double sumkrotrany =0;
double sumkrwtrany =0;
double sumtrany =0;
double static krotrany [60][220][4];
double static krwtrany [60][220][4];

for ( int k = Ck; k<Ck+s; k++)          // loop for fine cells in the k direction.
                {

for ( int i = Ci; i<Ci+s; i++)          // loop for fine cells in the i direction.
                {

krotrany [i][Cj][k] = kro [i][Cj][k] * trany [i][Cj][k];   //calculates product of kro * tranx at the down-stream edge of a coarse cell.
krwtrany [i][Cj][k] = krw [i][Cj][k] * trany [i][Cj][k];    //calculates product of krw * tranx at the down-stream edge of a coarse cell.
sumkrotrany = sumkrotrany + krotrany [i][Cj][k];     // sums kro*tranx
sumkrwtrany = sumkrwtrany + krwtrany [i][Cj][k];     //sums krw*tranx
sumtrany = sumtrany + trany [i][Cj][k];      //sum of tranx at the down-stream edge of a coarse cell.
                }
                }
if (sumtrany!=0)  //check that sumtranx is not zero.
                {

pseudokro [ts] = sumkrotrany / sumtrany;  //calculate pseudo kro using transmissibility weighting.
pseudokrw [ts] = sumkrwtrany / sumtrany;  //calculate pseudo krw using transmissibility weighting.
                }
                else
                {
```

```
pseudokro [ts] = 0;
pseudokrw [ts] = 0;
                }
        sumkrotrany =0;
        sumkrwtrany =0;
        sumtrany =0;
        }

// Function calculates pseudos in the z- direction.
void Pseudos_z (int Ci,int Ck, int Cj,int s, int ts, double pseudokro [100], double pseudokrw [100], double kro [60][220][4], double krw [60][220][4], double tranz [60][220][4])
        {
           double sumkrotranz =0;
           double sumkrwtranz =0;
           double sumtranz =0;
           double static krotranz [60][220][4];
           double static krwtranz [60][220][4];

for ( int j = Cj; j<Cj+s; j++)          // loop for fine cells in the k direction.
                {

for ( int i = Ci; i<Ci+s; i++)          // loop for fine cells in the k direction.
                {

krotranz [i][j][Ck] = kro [i][j][Ck] * tranz [i][j][Ck];   //calculates product of kro * tranx at down-stream edge of coarse cell.
krwtranz [i][j][Ck] = krw [i][j][Ck] * tranz [i][j][Ck];    //calculates product of krw * tranx at down-stream edge of coarse cell.
sumkrotranz = sumkrotranz + krotranz [i][j][Ck];     // sums kro*tranx
sumkrwtranz = sumkrwtranz + krwtranz [i][j][Ck];     //sums krw*tranx
sumtranz = sumtranz + tranz [i][j][Ck];              //sum of tranx at down-stream edge of coarse cell.
                }
                }

if (sumtranz!=0)  //check that sumtranx is not zero.
                {
```

```
pseudokro [ts] = sumkrotranz / sumtranz;  //calculate pseudo kro using transmissibility weighting.
pseudokrw [ts] = sumkrwtranz / sumtranz;  //calculate pseudo krw using transmissibility weighting.
                }
                else
                {
                pseudokro [ts] = 0;
                pseudokrw [ts] = 0;
                }
            sumkrotranz =0;
            sumkrwtranz =0;
            sumtranz =0;
        }

//Function calculates average water saturation in coarse cells the i direction.
void Avg_Swatx (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwatx [100], double Sw [60][220][4], double Pv [60][220][4])
        {
        double sumswpv = 0;
        double sumpv = 0;
        double swpv = 0;


for (int k = Ck; k<Ck+s; k++) // loop for fine cells in the k direction within a coarse cell.
            {
for (int j = Cj; j<Cj+s; j++)  // loop for fine cells in the j direction within a coarse cell.
            {

for (int i = Ci; i <Ci+s; i++)    // loop for fine cells in the i direction within a coarse cell.
                {
sumpv = sumpv + Pv [i][j][k]; // calculates sum of pore volume.
swpv = Sw [i][j][k] * Pv [i][j][k]; //calculates the product: water saturation * pore volume.
sumswpv = sumswpv + swpv; // calculates sum of Sw * Pv
                }
            }
```

```
        }

AvgSwatx [ts] = sumswpv/sumpv;     //calculation of avg. Sw using pore volume weighting.
        }

//Function calculates average water saturation in coarse cells in the  j direction
void Avg_Swaty (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwaty [100], double Sw [60][220][4], double Pv [60][220][4])
        {

double sumswpv = 0;
double sumpv = 0;
double swpv = 0;


for (int k = Ck; k<Ck+s; k++) // loop for fine cells in the k direction within a coarse cell.
            {

for (int j = Cj; j<Cj+s; j++)  // loop for fine cells in the j direction within a coarse cell.
            {

for (int i = Ci; i <Ci+s; i++)    // loop for fine cells in the i direction within a coarse cell.
                {

sumpv = sumpv + Pv [i][j][k]; // calculates sum of pore volume.
swpv = Sw [i][j][k] * Pv [i][j][k]; //calculates the product: water saturation * pore volume.
sumswpv = sumswpv + swpv; // calculates sum of Sw * Pv
            }
            }
            }
        AvgSwaty [ts] = sumswpv/sumpv;    //calculation of avg. Sw using pore volume weighting.
        }

//Function calculates average water saturation in coarse cells in the k direction
void Avg_Swatz (int Ck, int Ci, int Cj,int s,int ts,int ni,int nj, int nk, double AvgSwatz [100], double Sw [60][220][4], double Pv [60][220][4])
```

```
        {
        double sumswpv = 0;
        double sumpv = 0;
        double swpv = 0;

for (int k = Ck; k<Ck+s; k++) // loop for fine cells in the k direction within a coarse cell.
            {
for (int j = Cj; j<Cj+s; j++)   // loop for fine cells in the j direction within a coarse cell.
            {
for (int i = Ci; i <Ci+s; i++)   // loop for fine cells in the i direction within a coarse cell.
              {
sumpv = sumpv + Pv [i][j][k]; // calculates sum of pore volume.
swpv = Sw [i][j][k] * Pv [i][j][k]; //calculates the product: water saturation * pore volume.
sumswpv = sumswpv + swpv; // calculates sum of Sw * Pv
            }
            }
            }

AvgSwatz [ts] = sumswpv/sumpv;     //calculation of avg. Sw using pore volume weighting.
        }
```

//End of the program.

E. C++ program to generate well pseudos (2D model):

```
// Codes to generate well pseudos for a 2D cross-sectional model.

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

#include <iomanip>

using namespace std;

void kro_avg (int ts,int s,int Ck, int Iw,double krooilavg [50],double krowatavg
[50],double Twf [50], double Kro [50][50], double Krw [50][50]);

void Pseudo_Kr (int ts, double moboilavg [50], double mobwatavg [50], double PsKro
[50], double PsKrw [50],double muo, double muw);

void Avg_Swat (int I1, int In, int ts,int s, int Ck, int Iw, double AvgSwat [50] , double
Sw [50][50], double Pv [50][50]);


int main ()

{

string L;

int I1, In, ni, nk, ts,s, Iw, A, B;

double muo,muw;

double kroavg [50],krwavg [50],PsKro [50],PsKrw [50];

double Twf [50], Pv [50][50],Kro [50][50],Krw [50][50];

double Sw [50] [50], AvgSwat [50];


// Data to be entered by user.

cout <<"Enter the following data:" <<endl;

cout <<"**************************" <<endl;

cout <<endl<< "No. of fine cells in the I direction:  ";

cin >> ni;

cout <<endl<< "No. of fine cells in the K direction:  ";

cin >> nk;

cout <<endl<< "No. of time steps:  ";
```

```
cin >> ts;
cout <<endl<< "Scale up factor:  ";
cin >> s;
cout <<endl<<"Fine cell [I-direction] where the well is placed:  ";
cin >> Iw;
cout <<endl<<" First fine cell [I-direction] in the well's coarse cell:  ";
cin >> I1;
cout <<endl<<" Last fine cell [I-direction] in the well's coarse cell:  ";
cin >> In;
cout <<endl<<"Perforation Top [K-index]:  ";
cin >> A;
cout <<endl<<"Perforation Bottom [K-index]:  ";
cin >> B;
cout <<endl<<"oil viscosity:  ";
cin >> muo;
cout <<endl<<"water viscosity:  ";
cin >> muw;

// Print out well pseudos for Eclipse.
ofstream pseudos_outfile ("Well Pseudos.txt");

for (int Ck = 1; Ck <nk; Ck = Ck+s) // Ck is a counter for coarse cells in the k direction where the well is placed.
{
 // Read in all input files
ifstream Kro_infile ("Kro.txt"),Krw_infile ("Krw.txt"),Twf_infile ("Twf.txt"), Pv_infile ("Porevol.txt"),Sw_infile ("Swat.txt");

  // Skip 10 text lines header in all input files.
    for (int t=0; t<10; t++)
    {
```

```
getline (Kro_infile,L); getline(Krw_infile,L); getline(Pv_infile,L);
getline(Sw_infile,L);
   }
//print out the connate Sw value, Krw@Swc, Kro@Swc and Pc@Swc on top of each well pseudo table.
pseudos_outfile<<fixed<<"0.20000"<<"  "<< "0.00000"<<"  "<<fixed<<"0.90000"<<" "<< "0.00000"<<endl;
for (int k = A; k<=B; k++)
          {
 Twf_infile>> Twf [k]; // read in fine model well connections from data file to array.
          }
for (int tsteps = 1; tsteps <=ts; tsteps++ ) // loop for time steps.
         {
          for (int k = 1; k<=nk; k++)
          {
             for (int i = 1; i<=ni; i++)
             {
 // read in data from input files to arrays.
             Kro_infile>> Kro [i][k];
             Krw_infile>> Krw [i][k];
             Sw_infile>> Sw [i][k];
             Pv_infile>> Pv [i][k];
             }
          }
kro_avg (ts,s,Ck,Iw,kroavg,krwavg,Twf,Kro,Krw); // calculate average mobility using Twf weighting.
Pseudo_Kr (ts,kroavg,krwavg,PsKro,PsKrw,muo,muw);// calculate well pseudo.
Avg_Swat (I1, In, ts,s,Ck,Iw,AvgSwat,Sw,Pv); // calculate avg. Sw.

if (AvgSwat [ts]>0.2)
{
```

```
   pseudos_outfile.precision (5); // limits the decimal to fixed 5 digits.
 //arrange the well pseudo output to be used in Eclipse.
pseudos_outfile<<fixed<<AvgSwat [ts] <<"   " <<fixed<<PsKrw [ts] <<"   " <<fixed<<
PsKro [ts] <<"   "<<"0.00000"<<endl;

}
              for (int t=0; t<4; t++)     //skip 3 text lines in the input files.
              {
              getline (Kro_infile,L);
              getline (Krw_infile,L);
              getline (Sw_infile,L);
              }
 }
//Before proceeding to next well pseudo, print out Sw=1, Krw=1, Kro=1, and Pc=0 then
(/)at the bottom of each well pseudo table.
 pseudos_outfile << "1.00000"<<"   "<< "1.00000"<<"   "<< "0.00000"<<"   "<<
"0.00000"<<endl;
 pseudos_outfile << "/" <<endl;
}
}
// Function averages mobility in the well connections within each coarse cell penetrated
by the well
void kro_avg (int ts,int s,int Ck, int Iw,double kroavg [50],double krwavg [50],double
Twf [50], double Kro [50][50], double Krw [50][50])
{
double sumtwf = 0;
double krotwf, krwtwf;
double sumkrotwf = 0;
double sumkrwtwf = 0;
for ( int k = Ck; k<Ck+s ; k++) // loop for fine cells in the k direction where the well is
placed.
              {
```

```
for ( int i = Iw; i==Iw ; i++)      //Iw is the first fine cell in the column of cells where the well is placed.
                {
sumtwf = sumtwf + Twf [k];  //summing well connection factors (within a coarse cell).
                krotwf = Kro [i][k] * Twf [k];
                krwtwf = Krw [i][k] * Twf [k];
sumkrotwf = sumkrotwf + krotwf;    //summing product of mobility * connection factor (within a coarse cell)
                sumkrwtwf = sumkrwtwf + krwtwf;
                }
            }
kroavg [ts] = sumkrotwf/sumtwf;   // calculate pressure diff. between the coarse cell's avg. pressure and well BHP.
   krwavg [ts] = sumkrwtwf/sumtwf;
}

// Function calculates well pseudo.
void Pseudo_Kr (int ts,double kroavg [50], double krwavg [50], double PsKro [50], double PsKrw [50],double muo, double muw)
{
 PsKro [ts] = kroavg [ts] ;
 PsKrw [ts] = krwavg [ts] ;
}

//Function calculates Avg. water saturation in the coarse cells where the well is placed.
void Avg_Swat (int I1, int In, int ts,int s, int Ck, int Iw, double AvgSwat [50] , double Sw [50][50], double Pv [50][50])
{
double sumswpv = 0;
double sumpv = 0;
double swpv = 0;
```

```
for ( int k = Ck; k<Ck+s ; k++)
{
   for ( int i = I1; i<=In ; i++)
      {
      sumpv = sumpv + Pv [i] [k];
      swpv = Sw [i][k] * Pv [i][k];
      sumswpv = sumswpv + swpv;
      }
}
AvgSwat [ts] = sumswpv/sumpv;    //calculation of avg. water saturation using pore volume weighting.
}
```

//End of the program