Supplementary Material

1. This program is a 2D Generalized Lattice Boltzmann model using a double distribution function to simulate Gas Transport in Coal Reservoir.

2. The whole project is composed of 1.par and GLBM.f90.

3. '1.par' is a header file, where some common blocks are defined, all the subroutines are integrated into the 'GLBM.f90', where 'subroutine FlowField' is employed to reconstruct the images of 2D porous media after binarization by the MATLAB, ‘subroutine evolution' and 'subroutine diffusion' is implemented to govern the fluid bulk flow and diffusion, respectively, ‘subroutine initial’ and ‘subroutine output’ is used to initialize the flow field and record the simulation results, respectively.

4. Before running this project, please create a new sub-folder 'out' under the working directory.

5. In the simulation, all the parameters are dimensionless, and the output data can be dimensionalized using the principle of similitude.

1.par

```
implicit none
integer kk,i,j,k,a,b
integer,parameter::m=199,n=199

!D2Q9 Weights
      real(8),parameter:: w(0:8) = (/4.0d0/9.0d0&
     &,1.0d0/9.0d0,1.0d0/9.0d0,1.0d0/9.0d0,1.0d0/9.0d0&
      &,1.0d0/36.0d0,1.0d0/36.0d0,1.0d0/36.0d0,1.0d0/36.0d0/)
!D2Q9 Directions
      integer,parameter:: cx(0:8)=(/0,1,0,-1,0,1,-1,-1,1/)
      integer,parameter:: cy(0:8)=(/0,0,1,0,-1,1,1,-1,-1/)
!D2Q9 Anti-Directions
      integer,parameter:: opposite(0:8) = (/0,3,4,1,2,7,8,5,6/)

!D2Q5 Weights
      real(8),parameter:: wg(0:4) = (/1.0d0/3.0d0&
     &,1.0d0/6.0d0,1.0d0/6.0d0,1.0d0/6.0d0,1.0d0/6.0d0/)
!D2Q5 Directions
      integer,parameter:: cgx(0:4)=(/0,1,0,-1,0/)
```

```
      integer,parameter:: cgy(0:4)=(/0,0,1,0,-1/)

character*64 time,filename

real*8 dx,dy,dt,rho0,detap,Gx,Gy,tauf,niu,niue,u0(0:n),po_s,po_f
real*8 umax,c_kc,Fpo(0:m,0:n),dp,perm(0:m,0:n),time_begin,time_end,rhow,rhoe
real*8 u(0:m,0:n),v(0:m,0:n),rho(0:m,0:n),f(0:8,0:m,0:n),feq(0:8,0:m,0:n)
real*8 Da,J0,Re,aa,bb,uc,poro(0:m,0:n),Field(0:m,0:n),Field0(0:m,0:n)
real*8 g(0:4,0:m,0:n),geq(0:4,0:m,0:n),Cs(0:m,0:n),Cs_in(0:m,0:n),Nt(0:m,0:n)
real*8 ka,kd,Nm,N0,taug,taus,Ds,Dsp,c_in,Cs_0,Rs(0:m,0:n)

integer,parameter:: fluid=0,insolid=1,wall=2,corner=3,swall=4,inlet=10,outlet=11

common/C1/dx,dy,dt,rho0,detap,Gx,Gy,tauf,niu,niue,u0,umax,c_kc,Fpo,dp,perm
common/C2/time,filename,Field,Field0,Rs,time_begin,time_end
common/C3/u,v,rho,f,feq,Da,J0,Re,aa,bb,uc,poro,rhow,rhoe,po_s,po_f
common/C4/g,geq,Cs,Cs_in,Nt,ka,kd,Nm,N0,taug,taus,Ds,Dsp,c_in,Cs_0

program GLBM
  include '1.par'
real*8 rr,temp

! //check the picture
      call FlowField

     open(111,file='image1.dat')
      do j=0,n
      write(111,*)(Field(i,j),i=0,m)
      end do
      close(111)

! //lattice parameters
     dx=1.d0
     dy=dx
     dt=1.d0
! //macro parameters
      rho0=1.00860448645776d0
      tauf=0.8d0
     niu=(tauf-0.5d0)*dt/3.0d0

! //The characteristic constant
!po is porosity of the porous media
      po_s=0.05d0
     po_f=1.0d0
!dp is the diameter of the solid particle.
      dp=43.2852415859745d0
```

```
!J0=niue/niu
      J0=1.d0
       niue=niu*J0

!Gx,Gy is the component of body force
       Gx=0.0
       Gy=0.0
!rhow,rhoe is the density at inlet and outlet,respectively.
      rhow=1.00860464783447d0
      rhoe=rho0
      detap=(rhow-rhoe)/(m+1)

! //parameters for diffusion
!constant concentration at inlet
       c_in=0.1d0
!initial concentration in the field
       Cs_0=0.0
!adsorption and desorption constant
      ka=0.0519535824329524d0
      kd=0.015194244719982d0
      Nm=3.0
      N0=0.0

!diffusivity coefficient
      Ds=2.d0
      Dsp=po_s*1.0/30
      taug=3.0d0*Ds+0.5
      taus=3.0d0*Dsp+0.5
!————————————————————————————————————————————
call initial

      write(*,*) '--------------New Simulation--------------'
      write(*,*) 'D2Q9 LBM for gas flow at REV scale@pengzhgg'
       write(*,*) 'lx=',m+1,'ly=',n+1
      write(*,*) 'dx=',dx,'dy=',dy
      write(*,*) 'dt=',dt,'tauf=',tauf,'niu=',niu

      call CPU_TIME(time_begin)
! //main loop
do kk=1,1000000

       call evolution
       call diffusion

if(mod(kk,10000)==0) then
       ! call error_judge
```

```
        print *, '====================='
        print *, '====',kk,'========'
        print *, '====================='
        print *, 'rho=',',',rho(0,n/2),',',rho(m/2,n/2),',',rho(m,n/2)
        print *, 'u='    ,',',u(0,n/2),',',u(m/2,n/2),',',u(m,n/2)
        print *, 'v='    ,',',v(0,n/2),',',v(m/2,n/2),',',v(m,n/2)
        print *, 'Cs='    ,',',Cs(0,n/2),',',Cs(m/2,n/2),',',Cs(m,n/2)
end if

if(mod(kk,10000)==0.or.kk==1)then
        write(time,*)kk
        call output
end if
        end do

        call CPU_TIME(time_end)
        print *, '-------------- End --------------'
        print *, 'total time=',time_end-time_begin,'seconds'&
       &,(time_end-time_begin)/3600,'hour',kk

end program GLBM

subroutine FlowField
    include'1.par'

integer ip,jp

       open(12,file='image.dat')
        do j=n,0,-1
       read(12,*)(Field(i,j),i=0,m)
        end do
       close(12)

        do j=1,n-1
if(Field(0,j)==fluid)then
        Field(0,j)=inlet
end if
if(Field(m,j)==fluid)then
        Field(m,j)=outlet
end if
       end do
       do i=0,m
if(Field(i,0)==fluid)then
       Field(i,0)=swall
end if
if(Field(i,n)==fluid)then
```

```
      Field(i,n)=swall
end if
      end do

       do i=0,m
       do j=0,n
       Field0(i,j)=Field(i,j)
       end do
       end do

      do i=0,m
      do j=0,n

!identify the solid wall
if(Field0(i,j)==insolid)then
      do a=1,4
      ip=i+cx(a)
      jp=j+cy(a)
if(ip>=0.and.ip<=m.and.jp>=0.and.jp<=n)then
if(Field0(ip,jp)==fluid)then
       Field(i,j)=wall
end if
if(Field(ip,jp)==inlet.or.Field(ip,jp)==outlet)then
       Field(i,j)=wall
end if
if(Field(ip,jp)==swall)then
      Field(i,j)=corner
end if
end if
end do
end if
      end do
      end do

      do i=0,m
      do j=0,n
if(Field(i,j)==insolid)then

!identify the concave corner
if(Field(i+1,j)==wall.and.Field(i,j+1)==wall)then
if(Field(i+1,j+1)==fluid)then
      Field(i,j)=corner
end if
end if
if(Field(i-1,j)==wall.and.Field(i,j+1)==wall)then
if(Field(i-1,j+1)==fluid)then
```

```
      Field(i,j)=corner
end if
end if
if(Field(i-1,j)==wall.and.Field(i,j-1)==wall)then
if(Field(i-1,j-1)==fluid)then
      Field(i,j)=corner
end if
end if
if(Field(i+1,j)==wall.and.Field(i,j-1)==wall)then
if(Field(i+1,j-1)==fluid)then
      Field(i,j)=corner
end if
end if
end if

!identify the convex corner
if(Field(i,j)==wall)then
if(Field(i+1,j)==wall.and.Field(i,j+1)==wall)then
if(Field(i-1,j-1)==fluid)then
      Field(i,j)=corner
end if
end if
if(Field(i-1,j)==wall.and.Field(i,j+1)==wall)then
if(Field(i+1,j-1)==fluid)then
      Field(i,j)=corner
end if
end if
if(Field(i-1,j)==wall.and.Field(i,j-1)==wall)then
if(Field(i+1,j+1)==fluid)then
      Field(i,j)=corner
end if
end if
if(Field(i+1,j)==wall.and.Field(i,j-1)==wall)then
if(Field(i-1,j+1)==fluid)then
      Field(i,j)=corner
end if
end if
end if
      end do
      end do
end

subroutine initial
  include '1.par'

real*8 uu,eu
```

```
        do i=0,m
        do j=0,n
        u(i,j)=0.0
        v(i,j)=0.0
        rho(i,j)=rho0
        end do
        end do

!identify the free flow region and porous media by porosity
        do i=0,m
       do j=0,n
if(Field(i,j)==insolid.or.Field(i,j)==corner)then
        poro(i,j)=po_s
else
        poro(i,j)=po_f
end if
!Fpo is the geometric function.
        perm(i,j)=poro(i,j)*(poro(i,j)*dp)**2/(150*(1.0-poro(i,j))**2)
       Fpo(i,j)=1.75d0/(dsqrt(150.d0*poro(i,j))*poro(i,j))
        end do
       end do

! //initialize the velocity field
        do i=0,m
        do j=0,n
        uu=u(i,j)*u(i,j)+v(i,j)*v(i,j)
        do a=0,8
        eu=cx(a)*u(i,j)+cy(a)*v(i,j)
        feq(a,i,j)=rho(i,j)*w(a)*(1.0+3.0*eu+4.50*eu*eu/poro(i,j)-1.50*uu/poro(i,j))
        f(a,i,j)=feq(a,i,j)
        end do
        end do
        end do

! //initialize the concentration field
        do i=0,m
        do j=0,n
if(Field(i,j)==insolid.or.Field(i,j)==corner.or.Field(i,j)==wall)Nt(i,j)=N0
        do b=0,4
if(Field(i,j)==insolid.or.Field(i,j)==corner.or.Field(i,j)==wall)then
        g(b,i,j)=wg(b)*Cs_0
else
        g(b,i,j)=0.0d0
end if
if(i.eq.0) g(b,i,j)=wg(b)*c_in
```

```
        end do
        Cs_in(i,j)=0.d0
        end do
        end do
end

subroutine evolution
    include'1.par'

integer is,js
real*8 uu,eu,absu,Fx,Fy,force1,force2,force(0:8),uw,ue
real*8 c0,c1,sumu,sumv,sumf,absutmep,uvtemp,ftem(0:8),feq1(0:8,0:m,0:n)
real*8 utemp(0:m,0:n),vtemp(0:m,0:n),ftemp(0:8,0:m,0:n)

        do j=0,n
        do i=0,m
        sumf=0.0
        do a=0,8
        sumf=sumf+f(a,i,j)
        end do
        rho(i,j)=sumf
        end do
        end do

c0=0.5*(1.0+0.5*dt*poro(i,j)*niu/perm(i,j))
c1=0.5*dt*poro(i,j)*Fpo(i,j)/dsqrt(perm(i,j))

        do i=0,m
        do j=0,n
        sumu=0.0
        sumv=0.0
        do a=0,8
        sumu=sumu+f(a,i,j)*cx(a)+0.5*dt*poro(i,j)*rho(i,j)*Gx
        sumv=sumv+f(a,i,j)*cy(a)+0.5*dt*poro(i,j)*rho(i,j)*Gy
        end do
        utemp(i,j)=sumu/rho(i,j)
        vtemp(i,j)=sumv/rho(i,j)

       absutmep=dsqrt(utemp(i,j)*utemp(i,j)+vtemp(i,j)*vtemp(i,j))
       uvtemp=c0+dsqrt(c0**2+c1*absutmep)
       u(i,j)=utemp(i,j)/uvtemp
        v(i,j)=vtemp(i,j)/uvtemp
        end do
        end do

! //collision
```

```
      do i=0,m
       do j=0,n
          uu=u(i,j)*u(i,j)+v(i,j)*v(i,j)
         absu=dsqrt(uu)
!F's x and y component, respectively.
          Fx=-poro(i,j)*niu/perm(i,j)*u(i,j)-poro(i,j)*Fpo(i,j)*absu*u(i,j)/dsqrt(perm(i,j))+poro(i,j)*Gx
         Fy=-poro(i,j)*niu/perm(i,j)*v(i,j)-poro(i,j)*Fpo(i,j)*absu*v(i,j)/dsqrt(perm(i,j))+poro(i,j)*Gy
       do a=0,8
          eu=cx(a)*u(i,j)+cy(a)*v(i,j)
          feq(a,i,j)=rho(i,j)*w(a)*(1.0+3.0*eu+4.50*eu*eu/poro(i,j)-1.50*uu/poro(i,j))
!Guo's force scheme
         force1=3.*(cx(a)-u(i,j)/poro(i,j))*Fx+3.*(cy(a)-v(i,j)/poro(i,j))*Fy
         force2=9.*eu*(cx(a)*Fx+cy(a)*Fy)/poro(i,j)
!Fi
         force(a)=(1.-1./(2.*tauf))*rho(i,j)*w(a)*(force1+force2)
          f(a,i,j)=f(a,i,j)+(feq(a,i,j)-f(a,i,j))/tauf+dt*force(a)
       end do
       end do
       end do

! //steaming
       ftemp=f

      do a=1,8
      do i=0,m
      do j=0,n
         is=i-cx(a)
         js=j-cy(a)
         if(is>=0.and.is<=m.and.js>=0.and.js<=n)then
         f(a,i,j)=ftemp(a,is,js)
      end if
      end do
      end do
      end do

! //inlet and outlet
!periodic boundary
       ! do j=0,n
          ! f(1,0,j)=f(1,m-1,j)
          ! f(5,0,j)=f(5,m-1,j)
          ! f(8,0,j)=f(8,m-1,j)
          ! f(3,m,j)=f(3,1,j)
          ! f(6,m,j)=f(6,1,j)
          ! f(7,m,j)=f(7,1,j)
       ! end do
```

```
!pressure differece boundary
      do i=0,m
       do j=0,n
      if(Field(i,j)==inlet)then
       uw=1-(f(0,0,j)+f(2,0,j)+f(4,0,j)+2*(f(3,0,j)+f(6,0,j)+f(7,0,j)))/rhow
       f(1,0,j)=f(3,0,j)+2.0/3.0*rhow*uw
       f(5,0,j)=f(7,0,j)-0.5*(f(2,0,j)-f(4,0,j))+1.0/6.0*rhow*uw
       f(8,0,j)=f(6,0,j)+0.5*(f(2,0,j)-f(4,0,j))+1.0/6.0*rhow*uw
       end if
      if(Field(i,j)==outlet)then
       ue=-1+(f(0,m,j)+f(2,m,j)+f(4,m,j)+2*(f(1,m,j)+f(5,m,j)+f(8,m,j)))/rhoe
       f(3,m,j)=f(1,m,j)-2.0/3.0*rhoe*ue
       f(6,m,j)=f(8,m,j)-0.5*(f(2,m,j)-f(4,m,j))-1.0/6.0*rhoe*ue
       f(7,m,j)=f(5,m,j)+0.5*(f(2,m,j)-f(4,m,j))-1.0/6.0*rhoe*ue
       end if
      end do
       end do

! //solid wall
do i=0,m
       f(4,i,n)=f(2,i,n)
       f(7,i,n)=f(5,i,n)
       f(8,i,n)=f(6,i,n)
       f(2,i,0)=f(4,i,0)
       f(5,i,0)=f(7,i,0)
       f(6,i,0)=f(8,i,0)
end do
end

subroutine diffusion
   include'1.par'
   real*8 eu,sumg

      do i=0,m
      do j=0,n
      sumg=0.0
      do b=0,4
      sumg=sumg+g(b,i,j)
      end do
      Cs(i,j)=sumg
      end do
      end do

      do i=0,m
      do j=1,n-1
if(Field(i,j)==insolid.or.Field(i,j)==corner.or.Field(i,j)==wall)then
```

```
      Rs(i,j)=ka*Cs(i,j)*(Nm-Nt(i,j))-kd*Nt(i,j)
      Cs_in(i,j)=Cs(i,j)
end if
      end do
      end do

! //collision
      do i=0,m
       do j=0,n
       do b=0,4
       eu=cgx(b)*u(i,j)+cgy(b)*v(i,j)
       geq(b,i,j)=Cs(i,j)*wg(b)*(1.0+3.0*eu)
if(Field(i,j)==insolid.or.Field(i,j)==corner.or.Field(i,j)==wall)then
      g(b,i,j)=g(b,i,j)+(geq(b,i,j)-g(b,i,j))/taus-Rs(i,j)*wg(b)
else
       g(b,i,j)=g(b,i,j)+(geq(b,i,j)-g(b,i,j))/taug
end if
       end do
       end do
       end do

! //steaming
      do j=0,n
      do i=1,m
      g(1,m-i,j)=g(1,m-i-1,j)
      g(2,i-1,j)=g(2,i,j)
      end do
      end do
      do i=0,m
      do j=1,n
      g(3,i,n-j)=g(3,i,n-j-1)
      g(4,i,j-1)=g(4,i,j)
      end do
      end do

! //boundary condition

      do j=1,n-1
       g(1,0,j)=c_in/3.0-g(2,0,j)
      g(3,0,j)=c_in/3.0-g(4,0,j)
      end do

      do j=1,n-1
      g(0,m,j)=g(0,m-1,j)
      g(1,m,j)=g(1,m-1,j)
      g(2,m,j)=g(2,m-1,j)
```

```
      g(3,m,j)=g(3,m-1,j)
      g(4,m,j)=g(4,m-1,j)
      end do

      do i=0,m
      g(0,i,n)=g(0,i,n-1)
      g(1,i,n)=g(1,i,n-1)
      g(2,i,n)=g(2,i,n-1)
      g(3,i,n)=g(3,i,n-1)
      g(4,i,n)=g(4,i,n-1)
      end do

      do i=0,m
      g(0,i,0)=g(0,i,1)
      g(1,i,0)=g(1,i,1)
      g(2,i,0)=g(2,i,1)
      g(3,i,0)=g(3,i,1)
      g(4,i,0)=g(4,i,1)
      end do

! //Adsorption/Desorption
      do i=0,m
      do j=0,n
if(Field(i,j)==insolid.or.Field(i,j)==corner.or.Field(i,j)==wall)then
      Nt(i,j)=Nt(i,j)+ka*Cs(i,j)*(Nm-Nt(i,j))-kd*Nt(i,j)
end if
if(Nt(i,j).lt.0)then
      Nt(i,j)=0.0
end if
      end do
      end do
end

subroutine output
   include '1.par'

filename="out/2D"//trim(adjustl(time))//".dat"

open(2,file=filename)

! ————————————————————————————————————————————————————————————————
write(2,*)"VARIABLES =X, Y, U, V, rho, Cs, Cs_in, Nt"
write(2,*)"ZONE ","I=",m+1,"J=",n+1,","F=BLOCK"
do j=0,n
write(2,*)(i,i=0,m)
end do
```

```
do j=0,n
write(2,*)(j,i=0,m)
end do
do j=0,n
write(2,*)(u(i,j),i=0,m)
end do
do j=0,n
write(2,*)(v(i,j),i=0,m)
end do
do j=0,n
write(2,*)(rho(i,j),i=0,m)
end do
do j=0,n
write(2,*)(Cs(i,j),i=0,m)
end do
do j=0,n
write(2,*)(Cs_in(i,j),i=0,m)
end do
do j=0,n
write(2,*)(Nt(i,j),i=0,m)
end do
end
```