```
module ConicalC2R3D
  use util_
  use ConicalRWE3D_wemig
  implicit none

  integer, private :: nsx,nsy,nsz,norm,cnx,cny,cnz,rnx,rny,rnz
  real   , private :: xmin,xmax,ymin,ymax,zmin,zmax,dxx,dyy,dzz
  integer, private :: ne1,ne2,ne3,nx1,nx2,nx3,ns1,ns2,nT
  integer, private :: nr1,nr2,method
  real, private :: de1,de2,de3,oe1,oe2,oe3,dr1,dr2,or1,or2,os1,os2
  real, private :: ox1,ox2,ox3,dx1,dx2,dx3,ds1,ds2,oT,dT,minang,maxang

contains

  subroutine C2R3D_axes(aE1,aE2,aE3,aX1,aX2,aX3,aS1,aS2,aR1,aR2,aT)
    type(myaxis)     :: aE1,aE2,aE3,aX1,aX2,aX3,aS1,aS2,aR1,aR2,aT

    !! . . Cartesian coordinates
    nx1=ax1%n; nx2=ax2%n; nx3=ax3%n
    ox1=ax1%o; ox2=ax2%o; ox3=ax3%o
    dx1=ax1%d; dx2=ax2%d; dx3=ax3%d

    !! . . RWE coordinates
    ne1=aE1%n; ne2=aE2%n; ne3=aE3%n;
    de1=aE1%d; de2=aE2%d; de3=aE3%d;
    oe1=aE1%o; oe2=aE2%o; oe3=aE3%o;

    !! . . Input data coordinates
    nr1=aR1%n; nr2=aR2%n;
    dr1=aR1%d; dr2=aR2%d;
    or1=aR1%o; or2=aR2%o

    !! . . Shot axis
    ns1=as1%n; ns2=as2%n;
    os1=as1%o; os2=as2%o;
    ds1=as1%d; ds2=as2%d

    !! . . Tilt axis
    nT=aT%n; dT=aT%d; oT=aT%o

  end subroutine C2R3D_axes

  !-----------------------------------------------------------------

  subroutine C2R3D_init(insx,insy,insz,inorm,idxx,idyy,idzz,&
  icnx,icny,icnz,irnx,irny,irnz,ixmin,iymin,izmin,imethod,iminang,imaxang)
    integer :: insx,insy,insz,inorm,icnx,icny,icnz,irnx,irny,irnz,imethod
    real    :: idxx,idyy,idzz,ixmin,iymin,izmin,iminang,imaxang
    nsx = insx;     nsy=insy;       nsz = insz;   norm= inorm
    xmin=ixmin;     ymin = iymin;   zmin= izmin
    cnx = icnx;     cny=icny;       cnz = icnz
    rnx = irnx;     rny = irny;     rnz = irnz
    dxx = idxx;     dyy=idyy;       dzz = idzz
    method=imethod; minang=iminang; maxang=imaxang

    !! Define max offsets
    xmax=dble(cnx-1.)*dxx+xmin
    ymax=dble(cny-1.)*dyy+ymin
    zmax=dble(cnz-1.)*dzz+zmin

    write(0,*)                                   '    * INTERP PARAMETERS'
    write(0,'(a,i6,i6,i6)')                  '     * nsx,nsy,nsz:',nsx,nsy,nsz
    write(0,'(a,i6,i6,i6)')                  '     * cnx,cny,cnz:',cnx,cny,cnz
    write(0,'(a,i6,i6,i6)')                  '     * rnx,rny,rnz:',rnx,rny,rnz
    write(0,'(a,2f11.2,2f11.2,2f11.2)')  '     * dxx,dyy,dzz:',dxx,dyy,dzz
    write(0,'(a,2f11.2,2f11.2,2f11.2)')  '     * xmn,ymn,zmn:',xmin,ymin,zmin
    write(0,'(a,2f11.2,2f11.2,2f11.2)')  '     * xmx,ymx,zmx:',xmax,ymax,zmax
  end subroutine C2R3D_init

  !-----------------------------------------------------------------
  subroutine C2R3D_CART(RC,CC,gnorm,adj,e1int1,e1int2,tilt_angle,is2,iT)
    integer          :: ix1,ix2,ix3,ie1,ie2,ie3,ix,jx,iz,jz,jy,iy,iT
    integer          :: e1int1,e1int2,is2,ith
    real             :: x,y,z,xx,yy,zz,ddx,ddy,ddz,r,s,dr,fx,fz,fy,pt
    real             :: w000,w001,w010,w011,w100,w101,w110,w111,sumw,y0
    real             :: d2,o2,r0,tilt_angle,tangle,x_init,z_init,z0,x0
    real             :: CC(:,:,:),RC(:,:,:),gnorm(:,:)
    real,allocatable :: rline(:),tline(:),CCtmp(:,:,:,:)
    logical          :: adj

    !! For OMP
    integer, external :: omp_get_num_threads,omp_get_thread_num
    integer  :: nth,id

    allocate( tline(ne3),rline(ne2) )
    call GET_LINES(tline,rline)

    !! Tilt angle to radians
    tangle = tilt_angle*acos(-1.)/180.

    call COORD_SHIFTS(method,tangle,x_init,z_init)

    !$OMP PARALLEL
```

```
    nth = omp_get_num_threads()
    !$OMP END PARALLEL

    if (nr2 .eq. 1) then
       r0 = dble(ne3-1.)*dr1/2.
    else
       r0 = abs(or2) !dble(ne3-1.)*dr2/2.
    end if

    dr = sqrt( dxx**2 + dyy**2 + dzz**2 )
    if (adj) then !! . . Interpolating RWE Images -> Cart

       if (rny .eq. 1 ) then !! If 2D

          dr = sqrt( dxx**2 + dzz**2 )
          allocate( CCtmp(cnz,cnx,1,nth) )
          !$OMP PARALLEL DO PRIVATE(ie3,ie1,x,z,ix,iz,pt,id,xx,ddx,zz,ddz,r,s,x0,z0)
          EE3Loop: do ie3=1,rnz
             x0 = abs(r0)*sinh(tline(ie3))*sin(tangle)
             z0 = abs(r0)*sinh(tline(ie3))*cos(tangle)
             EE1Loop: do ie1=1,rnx

                if (method .eq. 1) then
                    x = or1 + dble(ie1+e1int1-2.)*dr1 + real(iT-1.)*ds1
                   z = dble(ie3-1.)*de3
                else
                    x=or1 + dble(ie1+e1int1-2.)*dr1*cos(tangle)+x0-x_init + real(iT-1.)*ds1
                   if (tangle .gt. 0.) then
                      z = z0 - dble(    iE1-1.)*dr1*sin(tangle)
                   else
                      z = z0 + dble(ne1-iE1+1.)*dr1*sin(tangle)
                   end if
                end if
                ix = floor((x-xmin)/dxx)+1
                iz = floor((z-zmin)/dzz)+1
                id = omp_get_thread_num()+1
                pt = RC(ie1,1,ie3)*gnorm(1,ie3)

                XX1Loop: do jx = ix-nsx,ix+nsx
                   ddx = (jx-ix-1.)*dxx !xx-x

                   XX3Loop: do jz = iz-nsz,iz+nsz
                      zz = zmin+(jz-1.)*dzz
                      ddz = zz-z
                      if(jx .le. nsx .or. jx .gt. cnx-nsx) cycle
                      if(jz .le. nsz .or. jz .gt. cnz-nsz) cycle
                      r = sqrt( ddz**2 + ddx**2 ) / dr
                      s = min(sin(r)/r,1.0)
                      CCtmp(jz,jx,1,id)=CCtmp(jz,jx,1,id) + pt * s

                   end do XX3Loop
                end do XX1Loop
             end do EE1Loop
          end do EE3Loop
          !$OMP END PARALLEL DO
          !! . . Sum cumulative image
          do ith=1,nth
             !$OMP PARALLEL DO PRIVATE(ix,iz)
                do ix=1,cnx
                   do iz=1,cnz
                      CC(iz,ix,1) = CC(iz,ix,1) + CCtmp(iz,ix,1,ith)
                   end do
                end do
             !$OMP END PARALLEL DO
          end do
       else !! If 3D
          allocate ( CCtmp(cnz,cnx,cny,nth) )
          y0 = (is2-1.)*ds2+os2
          !$OMP PARALLEL DO PRIVATE(x0,z0,ie3,ie2,ie1,x,y,z,ix,iy,iz,&
                   pt,id,xx,ddx,yy,ddy,zz,ddz,r,s)
          E3Loop: do ie3=1,rnz
             if ( method .ne. 1) then
                x0 = abs(r0)*sinh(tline(ie3))*sin(tangle)
                z0 = abs(r0)*sinh(tline(ie3))*cos(tangle)
             end if
             E2Loop: do ie2=1,rny
                E1Loop: do ie1=1,rnx

                   !! Check X
                   if (method .eq. 1) then
                      x = dble(ie1+e1int1-2.)*dr1 + or1 - x_init + real(iT-1.)*ds1
                      y = dble(ie2-1.)*dr2 + or2 + y0
                      z = dble(ie3-1.)*de3 - z_init
                   else
                      x=or1+dble(ie1+e1int1-2.)*dr1*cos(tangle)+x0*sin(rline(ie2))-x_init+ real(iT-1
.)*ds1
                      y=-abs(r0)*cosh(tline(ie3))*cos(rline(ie2))+y0
                      if (tangle .gt. 0.) then
                         z = z0*sin(rline(ie2))-dble(iE1-1.    )*dr1*sin(tangle)
                      else
                         z = z0*sin(rline(ie2))+dble(ne1-iE1+1.)*dr1*sin(tangle)
                      end if
                   end if
```

```
                ix = (x-xmin)/dxx+1.5
                iy = (y-ymin)/dyy+1.5
                iz = (z-zmin)/dzz+1.5
                id = omp_get_thread_num()+1
                pt = RC(ie1,ie2,ie3)*gnorm(ie2,ie3)**2

                X1Loop: do jx=ix-nsx,ix+nsx
                   if(jx .lt. 1 .or. jx .gt. cnx) cycle
                   ddx=(jx-ix-1.)*dxx

                   X2Loop: do jy=iy-nsy,iy+nsy
                      if(jy .lt. 1 .or. jy .gt. cny) cycle
                      yy=ymin+(jy-1.)*dyy
                      ddy=yy-y

                      X3Loop: do jz=iz-nsz,iz+nsz
                         if(jz .lt. 1 .or. jz .gt. cnz) cycle
                         zz=zmin+(jz-1.)*dzz
                         ddz=zz-z

                         r = sqrt( ddz**2 + ddy**2 + ddx**2 ) / dr
                         s = min(sin(r)/r,1.0)
                         CCtmp(jz,jx,jy,id) = CCtmp(jz,jx,jy,id)+pt*s

                      end do X3Loop
                   end do X2Loop
                end do X1Loop

             end do E1Loop
          end do E2Loop
       end do E3Loop
       !$OMP END PARALLEL DO
       !! . . Sum cumulative image
       do ith=1,nth
          !$OMP PARALLEL DO PRIVATE(ix,iy,iz)
          do iy=1,cny
             do ix=1,cnx
                do iz=1,cnz
                   CC(iz,ix,iy) = CC(iz,ix,iy) + CCtmp(iz,ix,iy,ith)
                end do
             end do
          end do
          !$OMP END PARALLEL DO
       end do
   end if

    deallocate( CCtmp )

 else  !! . . Interpolating Cartesian velocity models -> RWE

    if (ne2 .eq. 1) then !! 2D
       !$OMP PARALLEL DO PRIVATE(x0,z0,ie3,ie1,x,z,ix,iz,jx,jz,fx,fz,&
       w000,w010,w100,w110,sumw)
       do ie3=1,rnz
          x0 = abs(r0)*sinh(tline(ie3))*sin(tangle)
          z0 = abs(r0)*sinh(tline(ie3))*cos(tangle)
          do ie1=1,rnx
             if (method .eq. 1) then
                x = (ie1-1.)*dr1 + or1 - x_init + real(iT-1.)*ds1
                z = (ie3-1.)*de3 - z_init
             else
                x=or1+(ie1+e1int1-2.)*dr1*cos(tangle)+x0-x_init + real(iT-1.)*ds1
                if (tangle .gt. 0.) then
                   z = z0 - dble(    ie1-1.)*dr1*sin(tangle)
                else
                   z = z0 + dble(ne1-ie1+1.)*dr1*sin(tangle)
                end if
             end if
             x = min(x,xmax)
             x = max(x,xmin)
             z = min(z,zmax)
             z = max(z,zmin)
             ix=floor((x-xmin)/dxx)+1
             ix = min(ix  ,cnx)
             ix = max(ix  , 1 )
             jx = min(ix+1,cnx)
             iz = min(iz  ,cnz)
             iz = max(iz  , 1 )
             jz = min(iz+1,cnz)
             iz=floor((z-zmin)/dzz)+1
             fz= z - zmin-(iz-1.)*dzz
             fx= x - xmin-(ix-1.)*dxx
             fz = max(0., fz)
             fz = min(1., fz)
             fx = max(0., fx)
             fx = min(1., fx)

             !! . . Calculate weights
             w000 = (1.-fz)*(1.-fx)
             w010 = (1.-fz)*(   fx)
             w100 = (   fz)*(1.-fx)
             w110 = (   fz)*(   fx)
             sumw = w000+w010+w100+w110
```

```
                RC(ie1,1,ie3) = RC(ie1,1,ie3) &
                +( w000 * CC(iz,ix,1) + w100 * CC(jz,ix,1) &
                +  w010 * CC(iz,jx,1) + w110 * CC(jz,jx,1)  ) /sumw
             end do
          end do
          !$OMP END PARALLEL DO
       else
          y0 = (is2-1.)*ds2+os2
          !$OMP PARALLEL DO PRIVATE(x0,z0,ie3,ie2,ie1,x,y,z,ix,iy,iz,jx,jy,jz,fx,fy,fz,&
          w000,w001,w010,w011,w100,w101,w110,w111,sumw)
          do ie3=1,rnz
             do ie2=1,rny
                x0 = abs(r0)*sinh(tline(ie3))*sin(tangle)
                z0 = abs(r0)*sinh(tline(ie3))*cos(tangle)
                do ie1=1,rnx
                   if (method .eq. 1) then
                      x = (ie1-1.)*dr1 + or1 - x_init + real(iT-1.)*ds1
                      y = (ie2-1.)*dr2 + or2 + y0
                      z = (ie3-1.)*de3 - z_init
                   else
                      x=or1+(ie1-1.)*dr1*cos(tangle)+x0*sin(rline(ie2))-x_init + real(iT-1.)*ds1
                      y=-abs(r0)*cosh(tline(ie3))*cos(rline(ie2)) + y0
                      if (tangle .gt. 0.) then
                         z = z0*sin(rline(ie2))-(    iE1-1.)*dr1*sin(tangle)
                      else
                         z = z0*sin(rline(ie2))+(ne1-iE1+1.)*dr1*sin(tangle)
                      end if
                   end if

                   !! X ind location
                   x = min(x,xmax)
                   x = max(x,xmin)
                   ix=floor((x-xmin)/dxx)+1
                   ix = min(ix  ,cnx)
                   ix = max(ix  , 1 )
                   jx = min(ix+1,cnx)

                   !! Y ind location
                   y = min(y,ymax)
                   y = max(y,ymin)
                   iy=floor((y-ymin)/dyy)+1
                   iy = min(iy  ,cny)
                   iy = max(iy  , 1 )
                   jy = min(iy+1,cny)

                   !! Z ind location
                   z = min(z,zmax)
                   z = max(z,zmin)
                   iz=floor((z-zmin)/dzz)+1
                   iz = min(iz  ,cnz)
                   iz = max(iz  , 1 )
                   jz = min(iz+1,cnz)

                   !! Weighting functions
                   fx= x - xmin-(ix-1.)*dxx
                   fx = max(0., fx)
                   fx = min(1., fx)

                   fy= y - ymin-(iy-1.)*dyy
                   fy = max(0., fy)
                   fy = min(1., fy)

                   fz= z - zmin-(iz-1.)*dzz
                   fz = max(0., fz)
                   fz = min(1., fz)


                   !! . . Calculate weights
                   w000 = (1.-fz)*(1.-fx)*(1.-fy)
                   w001 = (1.-fz)*(1.-fx)*(   fy)
                   w010 = (1.-fz)*(   fx)*(1.-fy)
                   w100 = (   fz)*(1.-fx)*(1.-fy)
                   w101 = (   fz)*(1.-fx)*(   fy)
                   w110 = (   fz)*(   fx)*(1.-fy)
                   w011 = (1.-fz)*(   fx)*(   fy)
                   w111 = (   fz)*(   fx)*(   fy)
                   sumw = w000+w001+w010+w011+w100+w101+w110+w111

                   RC(ie1,ie2,ie3) = RC(ie1,ie2,ie3) &
                   +( w000 * CC(iz,ix,iy) + w100 * CC(jz,ix,iy) &
                   +  w010 * CC(iz,jx,iy) + w110 * CC(jz,jx,iy) &
                   +  w001 * CC(iz,ix,jy) + w101 * CC(jz,ix,jy) &
                   +  w011 * CC(iz,jx,jy) + w111 * CC(jz,jx,jy) ) /sumw
                end do
             end do
          end do
          !$OMP END PARALLEL DO
       end if
    end if

    deallocate( rline,tline )
  end subroutine C2R3D_CART
```

```
end module ConicalC2R3D
```