```
!  ---------------------------------------------------------------
!     05/17/01
!
!    LES for helicity sabra model
!
!---------------------------------------------------------------

      program heli_sabra_III

    use stdtypes
    use mtprng
    implicit none

    integer :: idate(8)
    integer(INT32) :: seed
!   character(*), parameter :: fmt1 = "(2(3X,E15.8))"
    type(mtprng_state) :: state


      integer Ns,npower,nbar,nbar2,npower1,nbar3
      integer i,istart,iseed_m0,n,n_e,navg,nstep
      integer kk
!      parameter ( Ns=22,npower=9,nbar=100,nbar2=nbar/2,
      parameter ( Ns=9,npower=9,nbar=100,nbar2=nbar/2,&
                  npower1=9,nbar3=nbar*3)

      double precision halfdt,nu0,k0,dt,E,k(-1:Ns+2),pi,pi2,&
             t1,fac,aj,lambda,alpha
      double complex up(-1:Ns+2),um(-1:Ns+2),tt
      logical test(1)

      double precision ab,bb,cb,ah,bh,ch
      double precision expfac(-1:Ns+2),FFp(1:Ns),FFm(1:Ns)
      double complex gp(-1:Ns+2), gpold(-1:Ns+2)
      double complex gm(-1:Ns+2), gmold(-1:Ns+2)

      double precision sp(npower,Ns+2),tmp(Ns+2),tmp_e(Ns),tmp_h(Ns)
      double precision sn(npower1,Ns+2),flux_e(Ns),flux_h(Ns)
      double precision ssp(npower,Ns),ssh(npower,Ns)
      double precision sap(npower,Ns),sah(npower,Ns)
      double precision spectrum_E(1:Ns+2),spectrum_H(1:Ns+2)
      double precision e_input, h_input
      double precision dissp_E(1:Ns+2),dissp_H(1:Ns+2)
      double precision tmp_spec_E(Ns+2),tmp_spec_H(Ns+2)
      double precision tmp_dis_E(Ns+2),tmp_dis_H(Ns+2)

      integer step, i_start
```

```
      double precision drand48
      double precision thetap(1:Ns+2),upmag(1:Ns+2)
      double precision thetam(1:Ns+2),ummag(1:Ns+2)
      double precision multp(1:Ns+2),Deltap(1:Ns+2)
      double precision multm(1:Ns+2),Deltam(1:Ns+2)
      double precision UN,Delta_new,Delta_old
      double precision eng_bef,eng_aft,eng_dif
      double precision beta
      parameter (beta=2.00)
      double precision aaa,bbb,ccc,zzz,daa,dbb,errf,errx
      integer it, itmax
      parameter (itmax=50)
      double precision cont

      integer imet

      double precision timein, timeout


!!!! BEGIN RANDOM STUFF

!     call date_and_time(VALUES=idate)
!     print (*,*) idate
!     seed= idate(8)-500
!     write(*,*) seed

    open(89,file='/dev/random',access='stream',form='UNFORMATTED')
    read(89) seed
    close(89)
    write(*,*) 'seed', seed

    call mtprng_init(seed,state)

!!!! END RANDOM STUFF

      open(12,file='input.data',status='unknown')
      read(12,*)istart
      close(12)

      pi = datan(1.0d0)*4.0d0
      pi2 = 2.0d0*pi

!      dt = 1.0d-4
      dt = 1.0d-2
      halfdt   = 0.5d0*dt

!      nu0=1.0d-7
      nu0=1.0d-20
      lambda = 2.0d0
```

```
!  "ordinary" helicity correrspondes to alpha=1
      alpha = 1.0d0

      k0=lambda**(-4)

!      nstep=400000000
!      navg = 400000000
!      n_e = 1000000

!      nstep=400000000
!      navg = 100000000
!      n_e = 1000000

      nstep=5000000
      navg =2500000
      n_e =  5000

!      nstep=50000000
!      navg=25000000
!      n_e = 1000000

!       aaa=0.78d0; zzz=3d0*2**(-0.02d0)
        aaa=1.0d0; zzz=3.0d0
        it=1; errf=1d0; errx=1d0; bbb=1.0d0;
!        dbb =bbb**3+aaa*bbb**2+aaa**2*bbb+aaa**3-4.
!        errf=dabs(dbb)
!        dbb = dbb/(3.*bbb**2+2.*aaa*bbb+aaa**2)
!        errx=dabs(dbb)
!        bbb=bbb-dbb
!       it=it+1
!        write(*,*) 'At iteration ', it
!        write(*,*) 'bbb=', bbb,' errx=', errx
        ccc=4.0d0-aaa**3
        daa = ccc**2+zzz*aaa*ccc+zzz**2d0*aaa**2-zzz**3
        errf=dabs(daa)
        daa=daa/(-6d0*ccc*aaa**2+zzz*ccc-3d0*zzz*aaa**3+2d0*zzz**2*aaa)
        errx=dabs(daa)
        write(*,*) 'At iteration ', it
        write(*,*) 'aaa=', aaa,' errx=', errx
     do while ((it.le.itmax).and.(errf.ge.1.D-15).and.(errx.ge.1.D-15))
       aaa=aaa-daa
       it=it+1
!        dbb =bbb**3+aaa*bbb**2+aaa**2*bbb+aaa**3-4.
!        errf=dabs(dbb)
!        dbb = dbb/(3.*bbb**2+2.*aaa*bbb+aaa**2)
!        errx=dabs(dbb)
!        bbb=bbb-dbb
!       it=it+1
!        write(*,*) 'At iteration ', it
```

```
!        write(*,*) 'bbb=', bbb,' errx=', errx
         ccc=4.0d0-aaa**3
         daa = ccc**2+zzz*aaa*ccc+zzz**2d0*aaa**2-zzz**3
         errf=dabs(daa)
         daa=daa/(-6d0*ccc*aaa**2+zzz*ccc-3d0*zzz*aaa**3+2d0*zzz**2*aaa)
         errx=dabs(daa)
          write(*,*) 'At iteration ', it
         write(*,*) 'aaa=', aaa,' errx=', errx
      end do

         bbb=(4.0d0-aaa**3)/zzz
         write(*,*) 'aaa=', aaa, 'bbb=', bbb
         cont=lambda**(1./3.)


       up = (0.0d0,0.0d0)
       gp = (0.0d0,0.0d0)
       um = (0.0d0,0.0d0)
       gm = (0.0d0,0.0d0)

       sp = 0.0d0
       sn = 0.0d0
       ssp = 0.0d0
       ssh = 0.0d0
       sap = 0.0d0
       sah = 0.0d0
       expfac = 0.0d0
       tmp_e = 0.0d0
       tmp_h = 0.0d0
       tmp = 0.0d0

       dissp_E=0.0d0
       dissp_H=0.0d0
       spectrum_E=0.0d0
       spectrum_H=0.0d0
       tmp_dis_E=0.0d0
       tmp_dis_H=0.0d0

       do n = -1, Ns+2
         k(n)=k0*lambda**n
       end do

       do n=1,Ns
        expfac(n)= dexp(-nu0*k(n)*k(n)*dt)
        write(*,*)'expfac(', n,')=', expfac(n)
       end do

       iseed_m0 = 123
       call srand(iseed_m0)
```

```
      fac = 0.1d0
      do n=1, Ns
         t1 = rand()*pi2
         up(n)=(k(n)**(-1./3.))*fac*cmplx(dcos(t1),dsin(t1))
      end do

      iseed_m0 = 456
      call srand(iseed_m0)

      fac = 1.0d-4
      do n=1, Ns
         t1 = rand()*pi2
         um(n)=(k(n)**(-1./3.))*fac*cmplx(dcos(t1),dsin(t1))
      end do


      if(istart.eq.1)then
         open(12,file='init.data',status='unknown')
           do i=1,Ns
             read(12,*)up(i),um(i)
           enddo
           close(12)
         print *, 'after reading'
      endif

              print *, 'nstep =', nstep
              print *,'navg =', navg
              print *,'n_e=',n_e
              print *, 'dt =', dt
              print *, 'k0 =', k0
              print *, 'nu0 =', nu0
              print *, 'lambda =', lambda
              print *, 'alpha =', alpha

              do n = 1, Ns
                      print *, 'up(', n ,')=', up(n)
              end do

              do n = 1, Ns
                      print *, 'um(', n ,')=', um(n)
              end do

!     calculate couplings

              ab = 1.0d0
              ah = 1.0d0
              bb = -(lambda**(-alpha)+lambda**alpha)*&
                    lambda/(lambda**(1.+alpha)+lambda)
```

```
              bh = -(lambda**(-alpha)+lambda**alpha)*&
                    lambda**(1.+alpha)/(lambda**(1.+alpha)+lambda)
              cb = (-lambda**(-1.)+lambda**(-1.-alpha))*&
                    lambda**2/(lambda**(1.+alpha)+lambda)
              ch = (-lambda**(-1.)+lambda**(-1.-alpha))*&
                    lambda**(2.+2.*alpha)/(lambda**(1.+alpha)+lambda)
              write(69,*) ab,bb,cb
              write(69,*) ah,bh,ch


            FFp = 0.0d0
            FFm = 0.0d0
            FFp(2)=1.5d-3
            FFp(3)=FFp(2)/lambda
            FFm(2)=0.0d0
            FFm(3)=0.0d0
            write(*,*) FFp
            write(*,*) FFm

            e_input=0.0d0
            h_input=0.0d0
            do i=1,Ns
              e_input = e_input + dble(FFp(i)+FFm(i))
              h_input = h_input + 2.*(k(i)**alpha)*dble(FFp(i)-FFm(i))
            enddo

            write(*,*) 'e_input=', e_input
            write(*,*) 'h_input=', h_input
            pause


            Delta_old = 0.0
            call cpu_time(timein)

!!!! begin main integration loop in time
  do step=1,nstep

!--------generate subgrid modes

          thetap(Ns-1) = datan2(dimag(up(Ns-1)),dble(up(Ns-1)))
          thetam(Ns-1) = datan2(dimag(um(Ns-1)),dble(um(Ns-1)))
          thetap(Ns) = datan2(dimag(up(Ns)),dble(up(Ns)))
          thetam(Ns) = datan2(dimag(um(Ns)),dble(um(Ns)))
          upmag(Ns)=dsqrt(dble(up(Ns))**2+dimag(up(Ns))**2)
          ummag(Ns)=dsqrt(dble(um(Ns))**2+dimag(um(Ns))**2)

          UN=1d0
!          UN = mtprng_rand_real3(state)
!          if (UN.lt.0.5) then
```

```
!           multp(Ns+1) = dsqrt(2.*UN)
!         else
!           multp(Ns+1) = 1./dsqrt(2.*(1.-UN))
!         endif
          multp(Ns+1) = aaa + (bbb-aaa)*UN

!         UN = mtprng_rand_real3(state)
!         if (UN.lt.0.5) then
!           multm(Ns+1) = dsqrt(2.*UN)
!         else
!           multm(Ns+1) = 1./dsqrt(2.*(1.-UN))
!         endif
          multm(Ns+1) = aaa + (bbb-aaa)*UN

!         UN = mtprng_rand_real3(state)
!         if (UN.lt.0.5) then
!           multp(Ns+2) = dsqrt(2.*UN)
!         else
!           multp(Ns+2) = 1./dsqrt(2.*(1.-UN))
!         endif
          multp(Ns+2) = aaa + (bbb-aaa)*UN

!         UN = mtprng_rand_real3(state)
!         if (UN.lt.0.5) then
!           multm(Ns+2) = dsqrt(2.*UN)
!         else
!           multm(Ns+2) = 1./dsqrt(2.*(1.-UN))
!         endif
          multm(Ns+2) = aaa + (bbb-aaa)*UN

!         do imet=1,100
!           Delta_new = pi2*(mtprng_rand_real3(state)-0.5)
!           eng_bef = beta*dsin(Delta_old)
!           eng_aft = beta*dsin(Delta_new)
!           eng_dif = eng_aft - eng_bef
!           if(mtprng_rand_real3(state).le.dexp(-eng_dif)) then
!                Delta_old = Delta_new
!           endif
!         enddo
!         Deltap(Ns+1) = Delta_old
          UN = mtprng_rand_real3(state)
          Deltap(Ns+1) = pi*(UN-1.)

!         do imet=1,100
!           Delta_new = pi2*(mtprng_rand_real3(state)-0.5)
!           eng_bef = beta*dsin(Delta_old)
!           eng_aft = beta*dsin(Delta_new)
!           eng_dif = eng_aft - eng_bef
!           if(mtprng_rand_real3(state).le.dexp(-eng_dif)) then
```

```
!               Delta_old = Delta_new
!             endif
!            enddo
!            Deltam(Ns+1) = Delta_old
             UN = mtprng_rand_real3(state)
             Deltam(Ns+1) = pi*(UN-1.)

!            do imet=1,100
!              Delta_new = pi2*(mtprng_rand_real3(state)-0.5)
!              eng_bef = beta*dsin(Delta_old)
!              eng_aft = beta*dsin(Delta_new)
!              eng_dif = eng_aft - eng_bef
!              if(mtprng_rand_real3(state).le.dexp(-eng_dif)) then
!                   Delta_old = Delta_new
!              endif
!            enddo
!            Deltap(Ns+2) = Delta_old
             UN = mtprng_rand_real3(state)
             Deltap(Ns+2) = pi*(UN-1.)

!            do imet=1,100
!              Delta_new = pi2*(mtprng_rand_real3(state)-0.5)
!              eng_bef = beta*dsin(Delta_old)
!              eng_aft = beta*dsin(Delta_new)
!              eng_dif = eng_aft - eng_bef
!              if(mtprng_rand_real3(state).le.dexp(-eng_dif)) then
!                   Delta_old = Delta_new
!              endif
!            enddo
!            Deltam(Ns+2) = Delta_old
             UN = mtprng_rand_real3(state)
             Deltam(Ns+2) = pi*(UN-1.)

             thetap(Ns+1) = thetam(Ns-1)+thetam(Ns)-Deltap(Ns+1)
             thetam(Ns+1) = thetap(Ns-1)+thetap(Ns)-Deltam(Ns+1)
             thetap(Ns+2) = thetam(Ns)+thetam(Ns+1)-Deltap(Ns+2)
             thetam(Ns+2) = thetap(Ns)+thetap(Ns+1)-Deltam(Ns+2)
             upmag(Ns+1) = multp(Ns+1)*upmag(Ns)/cont
             ummag(Ns+1) = multm(Ns+1)*ummag(Ns)/cont
             upmag(Ns+2) = multp(Ns+2)*upmag(Ns+1)/cont
             ummag(Ns+2) = multm(Ns+2)*ummag(Ns+1)/cont

             up(Ns+1)= upmag(Ns+1)*&
                       dcmplx(dcos(thetap(Ns+1)),dsin(thetap(Ns+1)))
             um(Ns+1)= ummag(Ns+1)*&
                       dcmplx(dcos(thetam(Ns+1)),dsin(thetam(Ns+1)))
             up(Ns+2)= upmag(Ns+2)*&
                       dcmplx(dcos(thetap(Ns+2)),dsin(thetap(Ns+2)))
             um(Ns+2)= ummag(Ns+2)*&
```

```
                dcmplx(dcos(thetam(Ns+2)),dsin(thetam(Ns+2)))


!--------RECORD SOME QUANTITIES

! Calculating structure function
!
!         tmp = sqrt(real(up(1:Ns+2)*conjg(up(1:Ns+2)))
!     1             +real(um(1:Ns+2)*conjg(um(1:Ns+2))))
!
!         do i=1,npower
!          sp(i,:) = sp(i,:) + tmp**i
!         end do
!
!         do i=1,npower1
!          aj=-0.1d0*i
!          sn(i,:) = sn(i,:) + tmp**aj
!         end do
!
!
!-------------------------------------------------
! calculate the spectra of energy and helicity
         do n=1,Ns+2
           spectrum_E(n)=real(up(n)*conjg(up(n))+um(n)*conjg(um(n)))/&
                         (k(n)-k(n-1))/2.0d0
           spectrum_H(n)=k(n)**alpha&
                         *real(up(n)*conjg(up(n))-um(n)*conjg(um(n)))/&
                         (k(n)-k(n-1))
           tmp_spec_E(n)=tmp_spec_E(n)+spectrum_E(n)
           tmp_spec_H(n)=tmp_spec_H(n)+spectrum_H(n)
         end do


!-------------------------------------------------
! calculate the dissipation of energy and helicity
         do n=1,Ns+2
           dissp_E(n)=nu0*(k(n)**2)*(tmp(n)**2)
           dissp_H(n)=2.0*nu0*(k(n)**(2.+alpha))&
                      *real(up(n)*conjg(up(n))-um(n)*conjg(um(n)))
           tmp_dis_E(n)=tmp_dis_E(n)+dissp_E(n)
           tmp_dis_H(n)=tmp_dis_H(n)+dissp_H(n)
         end do

!-------------------------------------------------
! Calculating the energy and helicity flux

         do n=1,Ns
          flux_e(n) =&
          -ab*k(n)*aimag(um(n)*um(n+1)*conjg(up(n+2)))&
```

```
          +cb*k(n-1)*aimag(um(n-1)*um(n)*conjg(up(n+1)))&
          -ab*k(n)*aimag(up(n)*up(n+1)*conjg(um(n+2)))&
          +cb*k(n-1)*aimag(up(n-1)*up(n)*conjg(um(n+1)))
          tmp_e(n)=tmp_e(n)+flux_e(n)
         end do

         do n=1,Ns
          flux_h(n) =&
          ah*k(n)**(1.+alpha)*aimag(um(n)*um(n+1)*conjg(up(n+2)))&
          +ch*k(n-1)**(1.+alpha)*aimag(um(n-1)*um(n)*conjg(up(n+1)))&
          -ah*k(n)**(1.+alpha)*aimag(up(n)*up(n+1)*conjg(um(n+2)))&
          -ch*k(n-1)**(1.+alpha)*aimag(up(n-1)*up(n)*conjg(um(n+1)))
          flux_h(n) = 2.0*flux_h(n)
          tmp_h(n)=tmp_h(n)+flux_h(n)
         end do


!          do i=1,npower
!           ssp(i,:)=ssp(i,:)+dsign(1.0d0,flux_e)*abs(flux_e)**(i/3.)
!           sap(i,:)=sap(i,:)+abs(flux_e)**(i/3.)
!          end do
!
!          do i=1,npower
!           ssh(i,:)=ssh(i,:)+dsign(1.0d0,flux_h)*abs(flux_h)**(i/3.)
!           sah(i,:)=sah(i,:)+abs(flux_h)**(i/3.)
!          end do


         write(13,*) dfloat(step-1)*dt, flux_e(Ns)
         write(14,*) dfloat(step-1)*dt, flux_h(Ns)

         if(mod(step-1,n_e).eq.0)&
         write(15,*)dfloat(step-1)*dt,&
                    0.5*real(sum(up*conjg(up))+sum(um*conjg(um)))
         if(mod(step-1,n_e).eq.0)&
         write(16,*)dfloat(step-1)*dt,&
                     real(sum((k**alpha)*up*conjg(up))&
                     -sum((k**alpha)*um*conjg(um)))


!--------time-step equations


         do n = 1,Ns
         tt=    ab*k(n)*um(n+2)*conjg(up(n+1))&
                +bb*k(n-1)*um(n+1)*conjg(up(n-1))&
                -cb*k(n-2)*um(n-1)*um(n-2)
         gp(n) = cmplx(0.0d0,1.0d0) * tt
         tt=    ab*k(n)*up(n+2)*conjg(um(n+1))&
```

```
              +bb*k(n-1)*up(n+1)*conjg(um(n-1))&
              -cb*k(n-2)*up(n-1)*up(n-2)
           gm(n) = cmplx(0.0d0,1.0d0) * tt
           end do


           do n=2,3
           gp(n)=gp(n)+ (FFp(n)*cmplx(1.0d0,1.0d0))/conjg(up(n))
           gm(n)=gm(n)+ (FFm(n)*cmplx(1.0d0,1.0d0))/conjg(um(n))
           end do

          if(step.eq.1)then

            up = up + dt*gp
            gpold=gp
            um = um + dt*gm
            gmold=gm

            write(15,*)step,real(sum(up*conjg(up))+sum(um*conjg(um)))/2.
            write(16,*)step,real(sum((k**alpha)*up*conjg(up))&
                           -sum((k**alpha)*um*conjg(um)))

          else

              up = up + halfdt*(3.0*gp-expfac*gpold)
              up = up*expfac
              gpold = gp
              um = um + halfdt*(3.0*gm-expfac*gmold)
              um = um*expfac
              gmold = gm

          endif


!  output the results
!-----------------------------------------------------------------
          if(mod(step,navg).eq.0)then

              tmp_e = tmp_e/float(navg)
!              write(18,*)'#k  ',step*dt
              do i=1,Ns
               write(18,*)i,tmp_e(i)
              end do

              tmp_h = tmp_h/float(navg)
!              write(19,*)'#k  ',step*dt
              do i=1,Ns
               write(19,*)i,tmp_h(i)
              end do
```

```
!-----------------------------------------------------

            tmp_dis_E=tmp_dis_E/float(navg)
 !           write(20,*)'#k  ',step*dt
            do i=1,Ns
             write(20,*)i,tmp_dis_E(i)
            end do

            tmp_dis_H=tmp_dis_H/float(navg)
 !           write(21,*)'#k  ',step*dt
            do i=1,Ns
             write(21,*)i,tmp_dis_H(i)
            end do

            tmp_spec_E=tmp_spec_E/float(navg)
!            write(22,*)'#k  ',step*dt
            do i=1,Ns
             write(22,*)i,tmp_spec_E(i)
            end do

            tmp_spec_H=tmp_spec_H/float(navg)
!            write(23,*)'#k  ',step*dt
            do i=1,Ns
             write(23,*)i,tmp_spec_H(i)
            end do


!-----------------------------------------------------
! structure functions
!
!           sp=sp/float(navg)
!           sn=sn/float(navg)
!
!           ssp=ssp/float(navg)
!           ssh=ssh/float(navg)
!
!           sap=sap/float(navg)
!           sah=sah/float(navg)
!
!           do i=1,npower
!            write(30+i,*)'#k  ',step*dt
!             do j=1,Ns+2
!               write(30+i,*)j,sp(i,j)
!             end do
!           end do
!
!           do i=1,npower1
!            write(50+i,*)'#k  ',step*dt
```

```
!              do j=1,Ns+2
!                write(50+i,*)j,sn(i,j)
!              end do
!            end do
!
!            do i=1,npower
!             write(60+i,*)'#k  ',step*dt
!              do j=1,Ns+2
!                write(60+i,*)j,ssp(i,j)
!              end do
!            end do
!
!            do i=1,npower
!             write(70+i,*)'#k  ',step*dt
!              do j=1,Ns+2
!                write(70+i,*)j,ssh(i,j)
!              end do
!            end do
!
!            do i=1,npower
!             write(80+i,*)'#k  ',step*dt
!              do j=1,Ns+2
!                write(80+i,*)j,sap(i,j)
!              end do
!            end do
!
!            do i=1,npower
!             write(90+i,*)'#k  ',step*dt
!              do j=1,Ns+2
!                write(90+i,*)j,sah(i,j)
!              end do
!            end do

            sp=0.0d0
            sn=0.0d0
            ssp=0.0d0
            ssh=0.0d0
            sap=0.0d0
            sah=0.0d0
            tmp_e=0.0d0
            tmp_h=0.0d0
            tmp_dis_E=0.0d0
            tmp_dis_H=0.0d0
            tmp_spec_E=0.0d0
            tmp_spec_H=0.0d0

         endif

!!!! end main integration loop in time
```

```
end do

         open(12,file='final.data',status='unknown')
         do i=1,Ns
           write(12,*) up(i),um(i)
         enddo
         close(12)

         call cpu_time(timeout)
         write(*,*) 'cpu_time=', timeout-timein


!1002    format(2(1x,E17.8))
        stop

        end
```