```
--- /cosmomc-March-13/camb/inidriver.F90
+++ /MGcosmomc-Mar13/MGCAMB-Mar13/inidriver.F90
@@ -13,6 +13,7 @@
         use constants
         use Bispectrum
         use CAMBmain
+use mgvariables
 #ifdef NAGF95
         use F90_UNIX
 #endif
@@ -101,7 +102,54 @@
        call DarkEnergy_ReadParams(DefIni)

        P%h0     = Ini_Read_Double('hubble')
-
+
+model = Ini_Read_Int('model',0)
+write(*,*) "---------------------"
+write(*,*) "Model : ", model
+write(*,*) "---------------------"
+GRtrans= Ini_Read_Double('GRtrans',0.d0)
+
+if (model ==1) then
+B1= Ini_Read_Double('B1',0.d0)
+B2= Ini_Read_Double('B2',0.d0)
+lambda1_2= Ini_Read_Double('lambda1_2',0.d0)
+lambda2_2= Ini_Read_Double('lambda2_2',0.d0)
+ss= Ini_Read_Double('ss',0.d0)
+
+else if (model ==2) then
+MGQfix= Ini_Read_Double('MGQfix',1.d0)
+MGRfix= Ini_Read_Double('MGRfix',1.d0)
+
+else if (model ==3 ) then
+Qnot= Ini_Read_Double('Qnot',1.d0)
+Rnot= Ini_Read_Double('Rnot',1.d0)
+sss = Ini_Read_Double('sss',0.d0)
+
+else if (model ==4) then
+B1 = 4.d0/3.d0
+lambda1_2= Ini_Read_Double('B0',0.d0) ! it is considered as the B0 parameter here
+lambda1_2 = (lambda1_2*(299792458.d-3)**2)/(2.d0*p%H0**2)
+B2 = 0.5d0
+lambda2_2 = B1* lambda1_2
+ss = 4.d0
+
+else if (model ==5) then
+B1 = Ini_Read_Double('beta1',0.d0)
+lambda1_2= Ini_Read_Double('B0',0.d0)
+lambda1_2 = (lambda1_2*(299792458.d-3)**2)/(2.d0*p%H0**2)
+B2 = 2.d0/B1 -1.d0
+lambda2_2 = B1* lambda1_2
+ss= Ini_Read_Double('s',0.d0)
+
+else if (model ==6) then
+Linder_gamma = Ini_Read_Double('Linder_gamma',0.d0)
+
+else if (model /= 0) then
+print*, '***please choose a model***'
+stop
+end if
+
+
        if (Ini_Read_Logical('use_physical',.false.)) then

         P%omegab = Ini_Read_Double('ombh2')/(P%H0/100)**2


*********************************************************************************
```

```
--- /cosmomc-March-13/camb/equations.f90
+++ /MGcosmomc-Mar13/MGCAMB-Mar13/equations.f90
@@ -16,7 +16,18 @@
 !           optimized neutrino sampling, and reorganised neutrino integration functions
 ! Feb 2013: fixed various issues with accuracy at larger neutrino masses

-       module LambdaGeneral
+    module mgvariables
+    use precision
+    integer :: model
+    real(dl) :: GRtrans
+    real(dl) B1, B2, lambda1_2, lambda2_2, ss
+    real(dl) :: MGQfix, MGRfix, Qnot, Rnot, sss
+    real(dl) :: Linder_gamma
+    end module mgvariables
+
+
+
+      module LambdaGeneral
          use precision
          implicit none

@@ -26,6 +37,18 @@
           !(otherwise assumed constant, though this is almost certainly unrealistic)

          logical :: w_perturb = .true.
+
+! AH: Added but not used !
+         logical :: use_tabulated_w = .false.
+         real(dl) :: wa_ppf = 0._dl
+         real(dl) :: c_Gamma_ppf = 0.4_dl
+         integer, parameter :: nwmax = 5000, nde = 2000
+         integer :: nw_ppf
+         real(dl) w_ppf(nwmax), a_ppf(nwmax), ddw_ppf(nwmax)
+         real(dl) rde(nde),ade(nde),ddrde(nde)
+         real(dl), parameter :: amin = 1.d-9
+         logical :: is_cosmological_constant
+         private nde,ddw_ppf,rde,ade,ddrde,amin
        contains

         subroutine DarkEnergy_ReadParams(Ini)
@@ -1182,6 +1205,7 @@
         use ThermoData
         use lvalues
         use ModelData
+use mgvariables
         implicit none
         integer j
         type(EvolutionVars) EV
@@ -1201,6 +1225,18 @@
         real(dl) clxq, vq, diff_rhopi, octg, octgprime
         real(dl) sources(CTransScal%NumSources)
         real(dl) ISW
+
+real(dl) adotdota, term1, term2, term3, term4, term5, adotdotdota, Hdotdot, omm, ommdot, ommdotdot
+real(dl) cs2, opacity, dopacity
+real(dl) MG_gamma, MG_gammadot, MG_mu, MG_mudot, etadot
+real(dl) fmu,f1,f2
+real(dl) MG_rhoDelta, MG_alpha, MG_N, MG_D, MG_hdot, Hdot, dgqMG, dgrhoMG
+real(dl) LKA1, LKA2
+real(dl) MG_phi, MG_psi, MG_phidot, MG_psidot
+integer tempmodel
+real(dl) ISW_MG
+real(dl) MGQ,MGR,MGQdot, MGRdot, fQ, k2alpha
+real(dl) polterdot, MG_alphadot

         yprime = 0
         call derivs(EV,EV%ScalEqsToPropagate,tau,y,yprime)
@@ -1261,11 +1297,27 @@
```

```
         adotoa=sqrt((grho+grhok)/3)

+adotdota=(adotoa*adotoa-gpres)/2.d0
+Hdot =adotdota-adotoa**2.d0
+
+if ( a.lt. GRtrans ) then
+  tempmodel = 0
+else
+  tempmodel = model
+end if
+
         if (EV%no_nu_multpoles) then
+if (tempmodel == 0) then
               z=(0.5_dl*dgrho/k + etak)/adotoa
               dz= -adotoa*z - 0.5_dl*dgrho/k
               clxr=-4*dz/k
               qr=-4._dl/3*z
+
+else ! tempmodel /= 0 , using the old expression
+    clxr = 2*(grhoc_t*clxc+grhob_t*clxb)/3/k**2
+    qr= clxr*k/sqrt((grhoc_t+grhob_t)/3)*(2/3._dl)
+end if ! tempmodel /= 0
+
               pir=0
               pirdot=0
         else
@@ -1276,6 +1328,7 @@
         end if

         if (EV%no_phot_multpoles) then
+if (tempmodel == 0) then
               z=(0.5_dl*dgrho/k + etak)/adotoa
               dz= -adotoa*z - 0.5_dl*dgrho/k
               clxg=-4*dz/k -4/k*opac(j)*(vb+z)
@@ -1285,6 +1338,12 @@
               octg=0
               octgprime=0
               qgdot = -4*dz/3
+else ! tempmodel /= 0 , using the old expression
+  clxg=2*(grhoc_t*clxc+grhob_t*clxb)/3/k**2
+  qg= clxg*k/sqrt((grhoc_t+grhob_t)/3)*(2/3._dl)
+  qgdot =yprime(EV%g_ix+1)
+end if ! tempmodel /= 0
+
         else
             if (EV%TightCoupling) then
               pig = EV%pig
@@ -1316,18 +1375,135 @@

 !  Get sigma (shear) and z from the constraints
 !  have to get z from eta for numerical stability
-        z=(0.5_dl*dgrho/k + etak)/adotoa
-        sigma=(z+1.5_dl*dgq/k2)/EV%Kf(1)
+    if (tempmodel /= 0) then
+
+    if (model==1 .or.model==4 .or.model==5.or.model==6) then
+
+    LKA1 = lambda1_2 * k2 * a**ss
+    LKA2 = lambda2_2 * k2 * a**ss
+
+    MG_mu = (1.d0 + B1 * LKA1)/(1.d0 + LKA1)
+
+    MG_mudot = ((B1 - 1.d0) * adotoa * ss * LKA1) / ((1.d0+LKA1)**2.d0)
+
+    MG_gamma = (1.d0 + B2 * LKA2)/(1.d0 +LKA2)
+
+    MG_gammadot = ((B2 - 1.d0) * adotoa * ss* LKA2) / ((1.d0+LKA2)**2.d0)
+
+    if ( model ==4) then
```

```
+    MG_mu = MG_mu/(1.d0 - 1.4d-8 * lambda1_2 * a**3)
+    MG_mudot = MG_mudot/(1.d0 - 1.4d-8 * lambda1_2 * a**3) + 3.d0 * MG_mu* adotoa *a**3 *(1.4d-8 *
lambda1_2 )/(1.d0 - 1.4d-8 * lambda1_2 * a**3)
+    end if
+
+    if ( model ==6) then
+    omm=(CP%omegab+CP%omegac)/((CP%omegab+CP%omegac)+(1-CP%omegab-CP%omegac)*a**3)
+    ommdot=-3.d0*omm**2*a**3*adotoa*(1-CP%omegab-CP%omegac)/(CP%omegab+CP%omegac)
+
+    MG_mu=2.d0/3.d0*omm**(Linder_gamma-1.d0)*&
+    (omm**Linder_gamma+2-3.d0*Linder_gamma+3.d0*(Linder_gamma-0.5d0)*omm)
+
+    MG_mudot = MG_mu/omm*(Linder_gamma-1.d0)*ommdot+&
+    2.d0/3.d0*omm**(Linder_gamma-1.d0)*ommdot*&
+    (Linder_gamma*omm**(Linder_gamma-1.d0)+3.d0*(Linder_gamma-0.5d0))
+
+    MG_gamma = 1.d0
+
+    MG_gammadot = 0.d0
+
+    end if
+
+    MG_rhoDelta = dgrho + 3._dl * adotoa * dgq/ k
+
+    MG_alpha = ( etak/k + MG_mu*(MG_gamma*MG_rhoDelta+(MG_gamma -1.d0)*2.d0* dgpi)/(2.d0*k2)) / adotoa
+
+    sigma = k * MG_alpha
+
+    fmu =k2+0.5d0*MG_gamma*MG_mu*(3.d0*(grhoc_t+grhob_t)+ 4.d0*(grhog_t+grhor_t))
+    f1 = k2+0.5d0*(3.d0*(grhoc_t+grhob_t)+ 4.d0*(grhog_t+grhor_t))
+
+    term1 = MG_gamma*MG_mu* f1 * dgq/k
+
+    term2 = k2*MG_alpha* (MG_mu* MG_gamma- 1.d0)*(grhoc_t+grhob_t+(4.d0/3.d0)*(grhog_t+grhor_t))
+
+    term3= (MG_mu * ( MG_gamma -1.d0)* adotoa - MG_gamma*MG_mudot - MG_gammadot*MG_mu )*MG_rhoDelta
+
+    term4 = (2.d0)*(MG_mu*(MG_gamma - 1.d0)*adotoa - &
+    (MG_gamma - 1.d0)*MG_mudot - MG_gammadot*MG_mu)* dgpi
+
+    term5= (2.d0) * MG_mu*( 1.d0 - MG_gamma)* (grhog_t * pigdot + grhor_t * pirdot)
+
+
+    etadot = (term1 + term2 + term3 + term4 + term5)/( 2.d0 *fmu)
+
+    z = sigma - 3.d0 * etadot/k
+
+    MG_psi = - MG_mu * ( MG_rhoDelta + 4.d0* dgpi)/(2.d0*k2)
+
+    MG_phi = MG_gamma * MG_psi + MG_mu* 2.d0*dgpi/k2
+
+    MG_phidot = etadot - adotoa * (MG_psi - adotoa * MG_alpha)- Hdot * MG_alpha
+
+    else if ( model ==2.or.model ==3) then
+    if (model ==2) then
+    MGQ = MGQfix
+    MGR=MGRfix
+    MGQdot = 0.d0
+    MGRdot = 0.d0
+
+    else if (model ==3) then
+    MGQ = 1.d0 + (Qnot - 1.d0)* a**sss
+    MGR = 1.d0 + (Rnot - 1.d0)* a**sss
+    MGQdot = (Qnot - 1.d0)*adotoa* sss* a**(sss)
+    MGRdot = (Rnot - 1.d0)*adotoa* sss* a**(sss)
+
+    end if
+
+    MG_rhoDelta = dgrho + 3._dl * adotoa * dgq/ k
+
```

```
+    MG_phi = - MG_rhoDelta * MGQ/(2.d0*k2)
+    sigma = (etak - k * MG_phi)/adotoa
+    MG_alpha = sigma/k
+
+
+    fQ=k2+(3.d0/2.d0)*MGQ*(grhob_t+grhoc_t+(4.d0/3.d0)*(grhor_t+grhog_t))
+    f1=k2+(3.d0/2.d0)*(grhob_t+grhoc_t+(4.d0/3.d0)*(grhor_t+grhog_t))
+    k2alpha= k * sigma
+
+    term1 = MGQ * f1 * dgq/k
+    term2 = (MGQ - 1.d0) * k2alpha * (grhob_t+grhoc_t+(4.d0/3.d0)*(grhor_t+grhog_t))
+    term3 = -( MGQdot + (MGR-1.d0) * MGQ * adotoa) * MG_rhoDelta
+
+
+    etadot = (term1 + term2 + term3)/( 2.d0 *fQ)
+
+    z = sigma - 3.d0 * etadot/k
+
+    MG_psi = MGR * MG_phi - MGQ * 2.d0 * dgpi/k2
+
+    MG_phidot = etadot - adotoa * (MG_psi - adotoa * MG_alpha)- Hdot * MG_alpha
+
+    end if
+
+    else !GR limit ( model = 0 )
+
+    z=(0.5_dl*dgrho/k + etak)/adotoa
+    sigma=z+1.5_dl*dgq/k2
+    end if
+
+
+


         polter = 0.1_dl*pig+9._dl/15._dl*ypol(2)

-        if (CP%flat) then
-        x=k*(CP%tau0-tau)
-        divfac=x*x
-        else
-        x=(CP%tau0-tau)/CP%r
-        divfac=(CP%r*rofChi(x))**2*k2
-        end if
+if (CP%flat) then
+x=k*(CP%tau0-tau)
+divfac=x*x
+else if (model ==0) then
+x=(CP%tau0-tau)/CP%r
+divfac=(CP%r*rofChi(x))**2*k2
+else
+Stop " MGCAMB is working for flat universe at the moment. Please check www.sfu.ca/~aha25/MGCAMB.html for
updates."
+end if


         if (EV%TightCoupling) then
@@ -1343,6 +1519,8 @@

         pidot_sum =  pidot_sum + grhog_t*pigdot + grhor_t*pirdot
         diff_rhopi = pidot_sum - (4*dgpi+ dgpi_diff )*adotoa
+
+if(tempmodel == 0 ) then

 !Maple's fortran output - see scal_eqs.map
 !2phi' term (\phi' + \psi' in Newtonian gauge)
@@ -1362,6 +1540,26 @@
     9.D0/80.D0*EV%Kf(2)*octgprime)/k+(-9.D0/160.D0*dopac(j)*pig-21.D0/10.D0*dgpi-27.D0/ &
     80.D0*dopac(j)*ypol(2))/k**2)*vis(j)+(3.D0/16.D0*ddvis(j)*pig+9.D0/ &
     8.D0*ddvis(j)*ypol(2))/k**2+21.D0/10.D0/k/EV%Kf(1)*vis(j)*etak
+
+else
```

```
+
+if (model==1 .or. model==4 .or. model==5.or. model==6) MG_psidot = (MG_phidot - MG_gammadot * MG_psi -
MG_mu*MG_gamma* pidot_sum/k2 &
+-(MG_mudot*MG_gamma+MG_mu*MG_gammadot)*2.d0*dgpi/k2 )/MG_gamma
+if (tempmodel==2.or.tempmodel==3)&
+MG_psidot = MGR * MG_phidot + MGRdot * MG_phi - ( MGQdot * 2.d0 * dgpi + MGQ * pidot_sum)/k2
+MG_alphadot= MG_psi - adotoa * MG_alpha
+polterdot=9._dl/15._dl*ypolprime(2) + 0.1_dl*pigdot
+
+ISW_MG= expmmu(j) * (MG_phidot + MG_psidot)
+ISW=ISW_MG
+
+sources(1) = ISW+ vis(j)* (clxg/4.D0+polter/1.6d0 + vbdot/k -9.D0*(polterdot)/k2*opac
(j)/16.D0-9.D0/16.D0*dopac(j)* polter/k2&
++ 2.1d0*MG_alphadot + 3.D0/40.D0 *qgdot/k +21.D0/10.D0*dgpi/k2&
++(-3.D0/8.D0*EV%Kf(2)*ypolprime(3) - 9.D0/80.D0*EV%Kf(2)*octgprime)/k)&
++ (MG_alpha+vb/k+30.0d0/8.0d0 *polterdot/k2)*dvis(j)+ ddvis(j)*30.0d0/16.0d0*polter/k2
+
+end if
+


 ! Doppler term
@@ -1402,7 +1600,14 @@
          phi = -(dgrho +3*dgq*adotoa/k)/(k2*EV%Kf(1)*2)
             ! - (grhor_t*pir + grhog_t*pig+ pinu*gpnu_t)/k2

-         sources(3) = -2*phi*f_K(tau-tau_maxvis)/(f_K(CP%tau0-tau_maxvis)*f_K(CP%tau0-tau))
+if(tempmodel == 0 ) then
+sources(3) = -2*phi*f_K(tau-tau_maxvis)/(f_K(CP%tau0-tau_maxvis)*f_K(CP%tau0-tau))
+else
+if (model==1 .or. model==4 .or. model==5.or. model==6)&
+sources(3) = -MG_mu*(1+MG_gamma)*phi*f_K(tau-tau_maxvis)/(f_K(CP%tau0-tau_maxvis)*f_K(CP%tau0-tau))
+if(model==2.or.model==3)&
+sources(3) = -MGQ*(1+MGR)*phi*f_K(tau-tau_maxvis)/(f_K(CP%tau0-tau_maxvis)*f_K(CP%tau0-tau))
+end if
 !         sources(3) = -2*phi*(tau-tau_maxvis)/((CP%tau0-tau_maxvis)*(CP%tau0-tau))
          !We include the lensing factor of two here
        else
@@ -1981,6 +2186,7 @@
 !  ayprime is not necessarily GaugeInterface.yprime, so keep them distinct
         use ThermoData
         use MassiveNu
+use mgvariables
         implicit none
         type(EvolutionVars) EV

@@ -2006,6 +2212,14 @@
         !non-flat vars
         real(dl) cothxor !1/tau in flat case

+real(dl) dgpi, term1, term2, term3,term4, term5, adotdotdota, Hdotdot, omm, ommdot, ommdotdot
+real(dl) MG_gamma, MG_gammadot, MG_mu, MG_mudot, etadot
+real(dl) fmu,f1,f2
+real(dl) MG_rhoDelta, MG_alpha, MG_N, MG_D, MG_hdot, Hdot, dgqMG, dgrhoMG
+real(dl) LKA1, LKA2
+integer tempmodel
+real(dl) MGQ,MGR,MGQdot, MGRdot, fQ, k2alpha, MG_phi, MG_psi, MG_phidot
+

         k=EV%k_buf
         k2=EV%k2_buf
@@ -2049,20 +2263,40 @@
         dgrho=grhob_t*clxb+grhoc_t*clxc
 !  8*pi*a*a*SUM[(rho_i+p_i)*v_i]
         dgq=grhob_t*vb
+
+dgpi = grhor_t*pir + grhog_t*pig
+
```

```
         if (CP%Num_Nu_Massive > 0) then
            call MassiveNuVars(EV,ay,a,grho,gpres,dgrho,dgq, wnu_arr)
         end if

-        if (CP%flat) then
-         adotoa=sqrt(grho/3)
-         cothxor=1._dl/tau
-        else
-         adotoa=sqrt((grho+grhok)/3._dl)
-         cothxor=1._dl/tanfunc(tau/CP%r)/CP%r
-        end if
+if (CP%flat) then
+  adotoa=sqrt(grho/3)
+
+  gpres=gpres + (grhog_t+grhor_t)/3.d0 +grhov_t*w_lam
+  adotdota=(adotoa*adotoa-gpres)/2.d0
+  Hdot =adotdota-adotoa**2.d0
+
+  cothxor=1._dl/tau
+
+else if (model ==0) then
+   adotoa=sqrt((grho+grhok)/3._dl)
+   cothxor=1._dl/tanfunc(tau/CP%r)/CP%r
+else
+   Stop " MGCAMB is working for flat universe at the moment. Please check www.sfu.ca/~aha25/MGCAMB.html
for updates."
+end if
+
+if ( a.lt. GRtrans ) then
+   tempmodel = 0
+else
+   tempmodel = model
+end if
+
+

-        if (w_lam /= -1 .and. w_Perturb) then
+      !  if (w_lam /= -1 .and. w_Perturb) then
+if (w_lam /= -1 .and. w_Perturb.and. ay(1).lt.GRtrans) then
            clxq=ay(EV%w_ix)
            vq=ay(EV%w_ix+1)
            dgrho=dgrho + clxq*grhov_t
@@ -2117,17 +2351,184 @@

 !  Get sigma (shear) and z from the constraints
 ! have to get z from eta for numerical stability
-        z=(0.5_dl*dgrho/k + etak)/adotoa
-        if (CP%flat) then
- !eta*k equation
-         sigma=(z+1.5_dl*dgq/k2)
-         ayprime(2)=0.5_dl*dgq
-        else
-         sigma=(z+1.5_dl*dgq/k2)/EV%Kf(1)
-         ayprime(2)=0.5_dl*dgq + CP%curv*z
-        end if
-
-        if (w_lam /= -1 .and. w_Perturb) then
+
+if (tempmodel /= 0) then
+
+if (model==1 .or.model==4 .or.model==5.or.model==6) then
+
+LKA1 = lambda1_2 * k2 * a**ss
+LKA2 = lambda2_2 * k2 * a**ss
+
+MG_mu = (1.d0 + B1 * LKA1)/(1.d0 + LKA1)
+
+MG_mudot = ((B1 - 1.d0) * adotoa * ss * LKA1) / ((1.d0+LKA1)**2.d0)
+
+MG_gamma = (1.d0 + B2 * LKA2)/(1.d0 +LKA2)
```

```
+
+MG_gammadot = ((B2 - 1.d0) * adotoa * ss* LKA2) / ((1.d0+LKA2)**2.d0)
+
+if ( model ==4) then
+MG_mu = MG_mu/(1.d0 - 1.4d-8 * lambda1_2 * a**3)
+MG_mudot = MG_mudot/(1.d0 - 1.4d-8 * lambda1_2 * a**3) + 3.d0 * MG_mu* adotoa *a**3 *(1.4d-8 * lambda1_2
)/(1.d0 - 1.4d-8 * lambda1_2 * a**3)
+end if
+
+if ( model ==6) then
+omm=(CP%omegab+CP%omegac)/((CP%omegab+CP%omegac)+(1-CP%omegab-CP%omegac)*a**3)
+ommdot=-3.d0*omm**2*a**3*adotoa*(1-CP%omegab-CP%omegac)/(CP%omegab+CP%omegac)
+
+MG_mu=2.d0/3.d0*omm**(Linder_gamma-1.d0)*&
+(omm**Linder_gamma+2-3.d0*Linder_gamma+3.d0*(Linder_gamma-0.5d0)*omm)
+
+MG_mudot = MG_mu/omm*(Linder_gamma-1.d0)*ommdot+&
+2.d0/3.d0*omm**(Linder_gamma-1.d0)*ommdot*&
+(Linder_gamma*omm**(Linder_gamma-1.d0)+3.d0*(Linder_gamma-0.5d0))
+
+MG_gamma = 1.d0
+
+MG_gammadot = 0.d0
+
+end if
+
+MG_rhoDelta = dgrho + 3._dl * adotoa * dgq/ k
+
+MG_alpha = ( etak/k + MG_mu*(MG_gamma*MG_rhoDelta+(MG_gamma -1.d0)*2.d0* dgpi)/(2.d0*k2)) / adotoa
+
+sigma = k * MG_alpha
+! old comment:Small k: potential problem with stability, using full equations earlier is NOT more
accurate in general
+! Easy to see instability in k \sim 1e-3 by tracking evolution of vb
+
+! Use explicit equation for vb if appropriate
+
+
+if (EV%no_nu_multpoles) then
+pirdot = 0.d0
+else
+! Old expression
+! pirdot=k*(0.4_dl*qr-0.6_dl*ay(EV%lmaxg+10)+8._dl/15._dl*sigma)
+
+! New expression,
+
+if (EV%lmaxnr>2) then
+pirdot=EV%denlk(2)*qr- EV%denlk2(2)*ay(ix+1)+8._dl/15._dl*k*sigma
+else
+
+pirdot=EV%denlk(2)*qr +8._dl/15._dl*k*sigma
+end if
+end if
+
+
+if (EV%no_phot_multpoles) then
+pigdot = 0.d0
+else
+
+if (EV%tightcoupling) then
+pigdot = 0.d0 ! It could improve to second order
+
+else
+
+polter = pig/10+9._dl/15*E2 !2/15*(3/4 pig + 9/2 E2)
+
+! Old expression
+!pigdot=0.4_dl*k*qg-0.6_dl*k*ay(9)-opacity*(pig - polter) +8._dl/15._dl*k*sigma
+
+! New expression
```

```
+if (EV%lmaxg>2) then
+pigdot=EV%denlk(2)*qg-EV%denlk2(2)*ay(ix+1)-opacity*(pig - polter) &
++8._dl/15._dl*k*sigma
+else !closed case
+pigdot=EV%denlk(2)*qg-opacity*(pig - polter) +8._dl/15._dl*k*sigma
+endif
+end if
+
+end if !no_phot_multpoles
+
+fmu =k2+0.5d0*MG_gamma*MG_mu*(3.d0*(grhoc_t+grhob_t)+ 4.d0*(grhog_t+grhor_t))
+f1 = k2+0.5d0*(3.d0*(grhoc_t+grhob_t)+ 4.d0*(grhog_t+grhor_t))
+
+term1 = MG_gamma*MG_mu* f1 * dgq/k
+
+term2 = k2*MG_alpha* (MG_mu* MG_gamma- 1.d0)*(grhoc_t+grhob_t+(4.d0/3.d0)*(grhog_t+grhor_t))
+
+term3= (MG_mu * ( MG_gamma -1.d0)* adotoa - MG_gamma*MG_mudot - MG_gammadot*MG_mu )*MG_rhoDelta
+
+term4 = (2.d0)*(MG_mu*(MG_gamma - 1.d0)*adotoa - &
+(MG_gamma - 1.d0)*MG_mudot - MG_gammadot*MG_mu)* dgpi
+
+term5= (2.d0) * MG_mu*( 1.d0 - MG_gamma)* (grhog_t * pigdot + grhor_t * pirdot)
+
+
+etadot = (term1 + term2 + term3 + term4 + term5)/( 2.d0 *fmu)
+
+z = sigma - 3.d0 * etadot/k
+
+MG_psi = - MG_mu * ( MG_rhoDelta + 4.d0* dgpi)/(2.d0*k2)
+
+MG_phi = MG_gamma * MG_psi + MG_mu* 2.d0*dgpi/k2
+
+MG_phidot = etadot - adotoa * (MG_psi - adotoa * MG_alpha)- Hdot * MG_alpha
+
+else if ( model ==2.or.model ==3) then
+if (model ==2) then
+MGQ = MGQfix
+MGR=MGRfix
+MGQdot = 0.d0
+MGRdot = 0.d0
+
+else if (model ==3) then
+MGQ = 1.d0 + (Qnot - 1.d0)* a**sss
+MGR = 1.d0 + (Rnot - 1.d0)* a**sss
+MGQdot = (Qnot - 1.d0)*adotoa* sss* a**(sss)
+MGRdot = (Rnot - 1.d0)*adotoa* sss* a**(sss)
+
+end if
+
+MG_rhoDelta = dgrho + 3._dl * adotoa * dgq/ k
+
+MG_phi = - MG_rhoDelta * MGQ/(2.d0*k2)
+sigma = (etak - k * MG_phi)/adotoa
+MG_alpha = sigma/k
+
+
+fQ=k2+(3.d0/2.d0)*MGQ*(grhob_t+grhoc_t+(4.d0/3.d0)*(grhor_t+grhog_t))
+f1=k2+(3.d0/2.d0)*(grhob_t+grhoc_t+(4.d0/3.d0)*(grhor_t+grhog_t))
+k2alpha= k * sigma
+
+term1 = MGQ * f1 * dgq/k
+term2 = (MGQ - 1.d0) * k2alpha * (grhob_t+grhoc_t+(4.d0/3.d0)*(grhor_t+grhog_t))
+term3 = -( MGQdot + (MGR-1.d0) * MGQ * adotoa) * MG_rhoDelta
+
+
+etadot = (term1 + term2 + term3)/( 2.d0 *fQ)
+
+z = sigma - 3.d0 * etadot/k
+
```

```
+MG_psi = MGR * MG_phi - MGQ * 2.d0 * dgpi/k2
+
+MG_phidot = etadot - adotoa * (MG_psi - adotoa * MG_alpha)- Hdot * MG_alpha
+
+end if
+ayprime(2)= k*etadot
+
+else !GR limit ( model = 0 )
+! Get sigma (shear) and z from the constraints
+! have to get z from eta for numerical stability
+z=(0.5_dl*dgrho/k + etak)/adotoa
+if (CP%flat) then
+!eta*k equation
+sigma=(z+1.5_dl*dgq/k2)
+ayprime(2)=0.5_dl*dgq
+else
+sigma=(z+1.5_dl*dgq/k2)/EV%Kf(1)
+ayprime(2)=0.5_dl*dgq + CP%curv*z
+end if
+end if
+
+
+
+
+!        if (w_lam /= -1 .and. w_Perturb) then
+
+if (w_lam /= -1 .and. w_Perturb .and. ay(1).lt.GRtrans) then

            ayprime(EV%w_ix)= -3*adotoa*(cs2_lam-w_lam)*(clxq+3*adotoa*(1+w_lam)*vq/k) &
                -(1+w_lam)*k*vq -(1+w_lam)*k*z
@@ -2171,7 +2572,11 @@
              dgs = grhog_t*pig+grhor_t*pir

              ! Define shear derivative to first order
-             sigmadot = -2*adotoa*sigma-dgs/k+etak
+if ( tempmodel ==0) then
+  sigmadot = -2*adotoa*sigma-dgs/k+etak
+else
+  sigmadot = k * (MG_psi - adotoa * MG_alpha)
+end if

              !Once know slip, recompute qgdot, pig, pigdot
              qgdot = k*(clxg/4._dl-pig/2._dl) +opacity*slip

********************************************************************************

--- /cosmomc-March-13/camb/params.ini
+++ /MGcosmomc-Mar13/MGCAMB-Mar13/params.ini
@@ -1,3 +1,43 @@
+#MG variables
+#model= 0 : default GR
+#model= 1 : BZ(mu,gamma) ( introduced in arXiv:0809.3791 )
+#model= 2 : (Q,R) ( introduced in arXiv:1002.4197 )
+#model= 3 : (Q0,R0,s)( introduced in arXiv:1002.4197 )
+#model= 4 : f(R) ( introduced in arXiv:0909.2045 )
+#model= 5 : Chameleon ( introduced in arXiv:0909.2045 )
+#model= 6 : Linder's gamma (introduced in arXiv:0507263 )
+
+model = 0
+
+#Scale factor at which MG is turned on
+GRtrans= 0.0
+
+#BZ parameters:
+B1 = 1.3333333
+lambda1_2 = 7500
+B2 = 0.5
+lambda2_2 = 10000
+ss = 4
+
```

```
+#Bean parameters :
+#(Q,R)
+MGQfix=1.
+MGRfix=1.
+
+#(Q0,R0,s)
+Qnot=1.
+Rnot=1.
+sss=0.
+
+#f(R) and Chameleon models :
+B0 = 0.5
+beta1 = 0.1
+s = 3
+
+# Linder's gamma :
+Linder_gamma = 0.545
+
+
 #Parameters for CAMB

 #output_root is prefixed to output file names


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


--- /cosmomc-March-13/batch1/params_CMB_defaults.ini
+++ /MGcosmomc-Mar13/batch1/params_CMB_defaults.ini
@@ -33,6 +33,60 @@
 param[Alens] = 1 1 1 0 0
 param[fdm] = 0 0 0 0 0

+#MG variables
+#model= 0 : default GR
+#model= 1 : B-Z(mu,gamma) ( introduced in arXiv:0809.3791 )
+#model= 2 : (Q,R) ( introduced in arXiv:1002.4197 )
+#model= 3 : (Q0,R0,s)( introduced in arXiv:1002.4197 )
+#model= 4 : f(R), only lambda1_2 is used and the value is considered for B0 ( introduced in
arXiv:0909.2045 )
+#model= 5 : Chameleon ( Yukawa-type dark matter interaction ), only B1, lambda1_2, SS are used. Agian,
lambda1_2 is considered as B0 ( introduced in arXiv:0909.2045 )
+#model= 6 : Linder's gamma (introduced in arXiv:0507263 )
+
+model = 2
+
+
+GRtrans= 0.0
+
+param[B1] = 0 0 0 0 0
+#1.125 1.1 1.14 0.1 0.1
+
+#For BZ models :
+param[lambda1_2] = 0 0 0 0 0
+#0.67e4 0.6e4 0.7e4 10 10
+
+# For f(R) and chameleon models :
+#param[lambda1_2] = 0.001 0 1. 0.05 0.05
+
+param[B2] = 0 0 0 0 0
+#0.78 0.6 0.9 0.1 0.1
+
+param[lambda2_2] = 0 0 0 0 0
+#1.0e4 0.1e4 10.0e4 1 1
+
+param[ss] = 0 0 0 0 0
+#2 1 4 0.1 0.1
+
+param[MGQfix]= 1 0.5 1.5 0.03 0.03
+
```

```
+param[MGRfix] = 1 0.5 1.5 0.03 0.03
+
+param[Qnot] = 0 0 0 0 0
+#1 0.5 1.5 0.03 0.03
+
+
+param[Rnot]= 0 0 0 0 0
+#1 0.5 1.5 0.03 0.03
+
+#0 0 0 0 0
+
+param[sss] = 0 0 0 0 0
+#1 0.5 1.5 0.03 0.03
+
+
+param[Linder_gamma] =0.545 0.545 0.545 0 0
+
+
+
 param[ns] = 0.96 0.9 1.1 0.004 0.004
 #log[10^10 A_s]
 param[logA] = 3.1 2.7 4 0.001 0.001

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

--- /cosmomc-March-13/params_CMB.paramnames
+++ /MGcosmomc-Mar13/params_CMB.paramnames
@@ -13,6 +13,17 @@
 deltazrei     {\Delta}z_{re} #width of reionization
 Alens         A_{L}     #lensing potential scaled by sqrt(A_lens)
 fdm           \epsilon_0 f_d    #CosmoRec dark matter annihilation parameter, 0910.3663
+B1           \beta_1
+lambda1_2     B_0
+B2           \beta_2
+lambda2_2     {\lambda2_2}^2
+ss           s
+MGQfix       MGQfix
+MGRfix        MGRfix
+Qnot          Qnot
+Rnot          Rnot
+sss           s
+Linder_gamma  \gamma_L
 ns            n_s          #beware that pivot scale can change in .ini file
 nt            n_t
 nrun          n_{run}

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

--- /cosmomc-March-13/source/driver.F90
+++ /MGcosmomc-Mar13/source/driver.F90
@@ -13,6 +13,7 @@
     use MatrixUtils
     use IO
     use ParamNames
+ use mgvariables
     use camb
     use GaugeInterface, only : Eqns_name
     use DefineParameterization
@@ -39,6 +40,8 @@
 #ifdef MPI
     double precision intime
     integer ierror
+
+

     call mpi_init(ierror)
     if (ierror/=MPI_SUCCESS) stop 'MPI fail: rank'
@@ -99,6 +102,8 @@
     end if
```

```
     action = Ini_Read_Int('action',action_MCMC)
+
+

     propose_scale = Ini_Read_Real('propose_scale',2.4)

@@ -138,6 +143,13 @@
     end if

 #endif
+
+
+    model = Ini_Read_Int('model',0)
+write(*,*) "---------------------"
+write(*,*) "Model : ", model
+write(*,*) "---------------------"
+    GRtrans = Ini_Read_Double('GRtrans',0.d0)

     stop_on_error = Ini_Read_logical('stop_on_error',stop_on_error)

*******************************************************************************

--- /cosmomc-March-13/source/settings.f90
+++ /MGcosmomc-Mar13/source/settings.f90
@@ -31,7 +31,7 @@
     logical :: new_chains = .true.

     integer, parameter :: max_data_params = 100
-    integer, parameter :: max_theory_params = 30
+    integer, parameter :: max_theory_params = 41
     integer, parameter :: max_num_params = max_theory_params + max_data_params

     logical :: use_fast_slow = .false.

*******************************************************************************

--- /cosmomc-March-13/source/CMB_Cls_simple.f90
+++ /MGcosmomc-Mar13/source/CMB_Cls_simple.f90
@@ -1,12 +1,14 @@
     !Use CAMB
     module CMB_Cls
     use cmbtypes
+use mgvariables
     use CAMB, only : CAMB_GetResults, CAMB_GetAge, CAMBParams,
CAMB_SetDefParams,Transfer_GetMatterPower, &
-    AccuracyBoost,  Cl_scalar, Cl_tensor, Cl_lensed, outNone, w_lam, wa_ppf,&
+    AccuracyBoost,  Cl_scalar, Cl_tensor, Cl_lensed, outNone, w_lam,  wa_ppf,&
     CAMBParams_Set, MT, CAMBdata, NonLinear_Pk, Nonlinear_lens, Reionization_GetOptDepth,
CAMB_GetZreFromTau, &
     CAMB_GetTransfers,CAMB_FreeCAMBdata,CAMB_InitCAMBdata, CAMB_TransfersToPowers,
Transfer_SetForNonlinearLensing, &
     initial_adiabatic,initial_vector,initial_iso_baryon,initial_iso_CDM, initial_iso_neutrino,
initial_iso_neutrino_vel, &
-    HighAccuracyDefault, highL_unlensed_cl_template, ThermoDerivedParams, nthermo_derived,
BackgroundOutputs
+    HighAccuracyDefault, highL_unlensed_cl_template, ThermoDerivedParams, nthermo_derived,
BackgroundOutputs&
+,model, B1, lambda1_2, B2, lambda2_2, ss, GRtrans, MGQfix, MGRfix, Qnot, Rnot, sss,Linder_gamma
     use Errors !CAMB
     use settings
     use IO
@@ -53,6 +55,19 @@
     P%Reion%delta_redshift = CMB%zre_delta
     w_lam = CMB%w
     wa_ppf = CMB%wa
+
+B1 = CMB%B1
+lambda1_2 = CMB%lambda1_2
+B2 = CMB%B2
+lambda2_2 = CMB%lambda2_2
```

```
+ss = CMB%ss
+MGQfix = CMB%MGQfix
+MGRfix = CMB%MGRfix
+Qnot = CMB%Qnot
+Rnot = CMB%Rnot
+sss = CMB%sss
+Linder_gamma = CMB%Linder_gamma
+
     ALens = CMB%ALens
     P%InitialConditionVector(initial_iso_CDM) = &
     sign(sqrt(abs(CMB%iso_cdm_correlated) /(1-abs(CMB%iso_cdm_correlated))),CMB%iso_cdm_correlated)

*******************************************************************************
--- /cosmomc-March-13/source/cmbtypes.f90
+++ /MGcosmomc-Mar13/source/cmbtypes.f90
@@ -75,6 +75,17 @@
         real(mcp) YHe, nnu, iso_cdm_correlated, ALens, fdm !fdm is dark matter annihilation, eg,.
0910.3663
         real(mcp) :: omnuh2_sterile = 0._mcp  !note omnhu2 is the sum of this + standard neutrinos
         real(mcp) reserved(5)
+real B1
+real lambda1_2
+real B2
+real lambda2_2
+real ss
+real MGQfix
+real MGRfix
+real Qnot
+real Rnot
+real sss
+real Linder_gamma
     end Type CMBParams

     Type, extends(TTheoryPredictions) :: TheoryPredictions

*******************************************************************************
--- /cosmomc-March-13/source/params_CMB.f90
+++ /MGcosmomc-Mar13/source/params_CMB.f90
@@ -62,7 +62,8 @@
     Type(TIniFile) :: Ini
     Type(TParamNames) :: Names
     character(LEN=Ini_max_string_len) prior
-    call SetTheoryParameterNumbers(15,6)
+    call SetTheoryParameterNumbers(26,6)
+!    call SetTheoryParameterNumbers(15,6)

     this%H0_min = Ini_Read_Double_File(Ini, 'H0_min',this%H0_min)
     this%H0_max = Ini_Read_Double_File(Ini, 'H0_max',this%H0_max)
@@ -202,6 +203,8 @@
     subroutine SetForH(Params,CMB,H0, firsttime)
     use CMB_Cls
     use bbn
+   use mgvariables
+
     real(mcp) Params(num_Params)
     logical, intent(in) :: firsttime
     Type(CMBParams) CMB
@@ -239,6 +242,19 @@
         CMB%zre_delta = Params(13)
         CMB%ALens = Params(14)
         CMB%fdm = Params(15)
+
+ CMB%B1 = Params(16)
+ CMB%lambda1_2 = Params(17)
+ CMB%B2 = Params(18)
+ CMB%lambda2_2 = Params(19)
+ CMB%ss = Params(20)
+
+ CMB%MGQfix = Params(21)
+ CMB%MGRfix = Params(22)
```

```
+ CMB%Qnot = Params(23)
+ CMB%Rnot = Params(24)
+ CMB%sss = Params(25)
+ CMB%Linder_gamma = Params(26)
         call SetFast(Params,CMB)
     end if

@@ -250,6 +266,19 @@
     CMB%omdm = CMB%omdmh2/h2
     CMB%omv = 1- CMB%omk - CMB%omb - CMB%omdm

+ if (model == 4 ) then
+ CMB%B1 = 4.d0/3.d0
+ CMB%lambda1_2 = Params(17)* ((299792458.d-3)**2)/(2.d0 * CMB%H0**2)
+ CMB%B2 = 0.5d0
+ CMB%lambda2_2 = CMB%B1* CMB%lambda1_2
+ CMB%ss = 4.d0
+ end if
+ if (model == 5 ) then
+ CMB%lambda1_2 = Params(17)* ((299792458.d-3)**2)/(2.d0 * CMB%H0**2)
+ CMB%B2 = 2.d0/CMB%B1 -1.d0
+ CMB%lambda2_2 = CMB%B1* CMB%lambda1_2
+ end if
+
     end subroutine SetForH

     !!! Simple parameterization for background data, e.g. Supernovae only (no thermal history)
@@ -258,13 +287,15 @@
     Type(TIniFile) :: Ini
     Type(TParamNames) :: Names

-    call SetTheoryParameterNumbers(7,0)
+    call SetTheoryParameterNumbers(26,0)
+!    call SetTheoryParameterNumbers(7,0)
     this%late_time_only = .true.
     call this%Init(Ini,Names, 'params_background.paramnames')

     end subroutine BK_Init

     subroutine BK_ParamArrayToTheoryParams(this, Params, CMB)
+ use mgvariables
     class(BackgroundParameterization) :: this
     real(mcp) Params(:)
     class(TTheoryParams), target :: CMB
@@ -298,6 +329,33 @@
         CMB%fdm=0
         CMB%iso_cdm_correlated=0
         CMB%Alens=1
+
+ CMB%B1 = Params(16)
+ CMB%lambda1_2 = Params(17)
+ CMB%B2 = Params(18)
+ CMB%lambda2_2 = Params(19)
+ CMB%ss = Params(20)
+
+ CMB%MGQfix = Params(21)
+ CMB%MGRfix = Params(22)
+ CMB%Qnot = Params(23)
+ CMB%Rnot = Params(24)
+ CMB%sss = Params(25)
+ CMB%Linder_gamma = Params(26)
+
+ if (model == 4 ) then
+ CMB%B1 = 4.d0/3.d0
+ CMB%lambda1_2 = Params(17)* ((299792458.d-3)**2)/(2.d0 * CMB%H0**2)
+ CMB%B2 = 0.5d0
+ CMB%lambda2_2 = CMB%B1* CMB%lambda1_2
+ CMB%ss = 4.d0
+ end if
+ if (model == 5 ) then
```

```
+ CMB%lambda1_2 = Params(17)* ((299792458.d-3)**2)/(2.d0 * CMB%H0**2)
+ CMB%B2 = 2.d0/CMB%B1 -1.d0
+ CMB%lambda2_2 = CMB%B1* CMB%lambda1_2
+ end if
+
     end select
     end subroutine BK_ParamArrayToTheoryParams


*******************************************************************************

*******************************************************************************
```