```
--- CAMB-Jan15/equations.f90
+++ MGCAMB-Jan15/equations.f90

@@ -16,6 +16,382 @@
     !            optimized neutrino sampling, and reorganised neutrino integration functions
     ! Feb 2013: fixed various issues with accuracy at larger neutrino masses
     ! Mar 2014: fixes for tensors with massive neutrinos
+
+    ! Feb 2016: MGCAMB upgrade and new models.
+    !           Fixed some issues with ISW effect
+    !           All the MG functions are now at the beginning of the file in the mgvariables module
+    !           By Alex Zucca azucca@sfu.ca
+
+!***********************************************************
+!* MGCAMB mod: new variables and MG functions
+
+ module mgvariables
+  use precision
+  use ModelParams
+  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
+  real(dl) :: beta_star, a_star, xi_star  ! for model 7 (symmetron)
+  real(dl) :: beta0, xi0, DilR, DilS, A_2 ! for model 8 and 10 (dilaton)
+  real(dl) :: F_R0, FRn                   ! for model 9 (large curvature f(R))
+
+
+contains
+!------------------------------------------------
+! mu(a,k) function
+function MGMu(a,adotoa,k2,model)
+    implicit none
+    integer :: model
+    real(dl) :: a,adotoa,k2,MGMu
+    real(dl) :: LKA1 ! \lambda_1^2 k^2 a^s
+    real(dl) :: LKA2 ! \lambda_1^2 k^2 a^s
+    real(dl) :: t1, t2, t1dot, t2dot
+    real(dl) :: omm, ommdot
+
+
+    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
+
+        MGMu = (1.d0 + B1 * LKA1)/(1.d0 + LKA1)
+
+        if (model ==4) then ! correction for f(R) mu function.
+            MGMu = MGMu/(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)
+
+            MGMu=2.d0/3.d0*omm**(Linder_gamma-1.d0)*&
+            (omm**Linder_gamma+2-3.d0*Linder_gamma+3.d0*(Linder_gamma-0.5d0)*omm)
+
+        end if
+
+    else if (model == 7 .or. model==8 .or. model==9 .or. model==10) then
+        t1 = (2.d0*MGBeta(a, adotoa, model)**2.d0)*k2
+        t2 = (MGM(a, adotoa, model)**2.d0)*a**2.d0
+
+        MGMu = (k2 + t1 + t2)/(k2 + t2)
+
+    end if
+end function MGMu
+
+!-----------------------------------------------
+! \dot{mu}(a,k) function
+function MGMuDot(a,adotoa,k2,Hdot,model)
+    implicit none
+    integer :: model
+    real(dl) :: a, adotoa, MGMuDot
+    real(dl) :: LKA1 ! \lambda_1^2 k^2 a^s
+    real(dl) :: LKA2 ! \lambda_1^2 k^2 a^s
+    real(dl) :: k2, t1,t2,t1dot,t2dot
+    real(dl) :: omm, ommdot
+    real(dl) :: Hdot
+
+    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
+
+        MGMuDot = ((B1 - 1.d0) * adotoa * ss * LKA1) / ((1.d0+LKA1)**2.d0)
+
+        if ( model ==4) then ! correction for f(R) mu function.
+            MGMuDot = MGMuDot/(1.d0 - 1.4d-8 * lambda1_2 * a**3) + 3.d0 * &
+            MGMu(a,adotoa,k2,4)* 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)
+
+            MGMuDot = MGMu(a,adotoa, k2,6)/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))
+
+        end if
+
+    else if (model == 7 .or. model==8 .or. model==9 .or. model==10) then
+        t1 = (2.d0*MGBeta(a, adotoa, model)**2.d0)*k2
+        t2 = (MGM(a, adotoa, model)**2.d0)*a**2.d0
+        t1dot = 4.d0*MGBeta(a,adotoa, model)*MGBetaDot(a,adotoa, model)*k2
+        t2dot = (2.d0*a**2.d0)*(MGM(a,adotoa, model)*MGMDot(a,adotoa, Hdot, model) + (MGM(a,adotoa,
model)**2.d0) *adotoa)
+
+        MGMuDot = (t1dot*(k2 + t2) - t1*t2dot)/((k2 + t2)**2.d0)
+
+    end if
+
+end function MGMuDot
+
+
+!-----------------------------------------------
+! gamma(a,k) function
+function MGGamma(a,adotoa,k2, model)
+    implicit none
+    integer :: model
+    real(dl) :: a, adotoa, k2, MGGamma
+    real(dl) :: LKA1 ! \lambda_1^2 k^2 a^s
+    real(dl) :: LKA2 ! \lambda_1^2 k^2 a^s
+    real(dl) :: t1,t2, t1dot, t2dot
+
+    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
+
+        MGGamma = (1.d0 + B2 * LKA2)/(1.d0 +LKA2)
+
+        if ( model ==6) then
+            MGGamma = 1.d0
+
+        end if
+
+    else if (model == 7 .or. model==8 .or. model==9 .or. model==10) then
+        t1 = (2.d0*MGBeta(a, adotoa, model)**2.d0)*k2
+        t2 = (MGM(a, adotoa, model)**2.d0)*a**2.d0
+
+        MGGamma = (k2 - t1 + t2)/(k2 + t1 + t2)
+
+    end if
+
+end function MGGamma
+
+
+!-----------------------------------------------
+! \dot{gamma}(a,k) function
+function MGGammaDot(a,adotoa,k2,model)
+    implicit none
+    integer :: model
+    real(dl) :: a, adotoa, MGGammaDot
+    real(dl) :: LKA1 ! \lambda_1^2 k^2 a^s
+    real(dl) :: LKA2 ! \lambda_1^2 k^2 a^s
+    real(dl) :: k2
+    real(dl) :: t1,t2,t1dot,t2dot
+    real(dl) :: Hdot
+
+    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
+
+        MGGammaDot = ((B2 -1.d0)*adotoa * ss* LKA2)/((1.d0+LKA2)**2.d0)
+
+        if ( model ==6) then
+            MGGammaDot = 0.d0
+
+        end if
+
+    else if (model == 7 .or. model==8 .or. model==9 .or. model==10) then
+
+        t1 = (2.d0*MGBeta(a, adotoa, model)**2.d0)*k2
+        t2 = (MGM(a, adotoa, model)**2.d0)*a**2.d0
+        t1dot = 4.d0*MGBeta(a,adotoa, model)*MGBetaDot(a,adotoa, model)*k2
+        t2dot = (2.d0*a**2.d0)*(MGM(a,adotoa, model)*MGMDot(a,adotoa, Hdot, model) + (MGM(a,adotoa,
model)**2.d0) *adotoa)
+
+        MGGammaDot = 2.d0*(t1*t2dot-t1dot*(k2 + t2))/((k2 + t1 + t2)**2.d0)
+
+    end if
+
+
+end function MGGammaDot
+
+!*************************************************
+!* MGCAMB new models:
+!* m(a), beta(a) parametrization
+!*************************************************
+
+!-----------------------------------------------
+! m(a) function
+function MGM(a,adotoa,model)
+    implicit none
+    integer :: model
+    real(dl) :: a, adotoa
+    real(dl) :: MGM
+    real(dl) :: FRm0
+
+    ! SYMMETRON
+    if(model == 7) then
+        MGM = (CP%H0/3.0D05) / (xi_star) * sqrt(1.d0-(a_star/a)**3.d0)
+
+    ! DILATON: based on 1206.3568
+    else if (model==8) then
+        MGM = (CP%H0/3.0D05) /(xi0) * a**(- DilR)
+
+    ! Hu-Sawicki f(R) model: m, beta parametrization as in 1305.5647
+    else if (model == 9)then
+        FRm0 = (CP%h0/3.0D05)*sqrt((4.d0*CP%omegav + CP%omegab + CP%omegac)/((FRn+1.d0)*F_R0))!note 
factor of c here
+        MGM = FRm0 * ((4.d0 * CP%omegav + (CP%omegab + CP%omegac)*a**(-3.d0))/(4.d0 * CP%omegav + CP%
omegab &
+        + CP%omegac))**(FRn/2.d0+1.d0)
+
+    ! Simpler DILATON model
+    else if (model ==10)then
+        MGM = sqrt(3.d0*A_2)*(adotoa/a)  ! H(a) = da/dtau/a**2 = adotoa/a
+
+    end if
+
+end function MGM
+
+
+!-----------------------------------------------
+! \dot{m}(a) function
+function MGMDot(a, adotoa, Hdot,  model)
+    implicit none
+    integer  :: model
+    real(dl) :: a, adotoa, Hdot, MGMDot
+    real(dl) :: FRm0
+
+    ! SYMMETRON
+    if(model == 7) then
+        MGMDot = 1.5d0* (CP%H0/3.0D05)/(xi_star) *((a_star/a)**3.d0 * adotoa)/( sqrt(1.d0-(a_star/
a)**3.d0))
+
+    ! DILATON
+    else if (model==8) then
+        MGMDot = - DilR * MGM(a,adotoa,model) * adotoa ! complete this
+
+    ! Hu-Sawicki f(R) model
+    else if (model == 9)then
+        FRm0 = (CP%h0/3.0D05)*sqrt((4.d0*CP%omegav + CP%omegab + CP%omegac)/((FRn+1.d0)*F_R0))
+        MGMDot = MGM(a,adotoa,9) / (4.d0 * CP%omegav + (CP%omegab + CP%omegac)*a**(-3.d0)) * (-3.d0*
FRn / 2.d0 - 3.d0) *&
+        ((CP%omegab + CP%omegac)* a**(-3.d0) * adotoa )!/(4.d0 * CP%omegav + CP%omegab + CP%omegac)) !
complete this
+
+    ! Simple DILATON model
+    else if (model ==10)then
+        MGMDot = sqrt(3.d0*A_2)*(Hdot- adotoa**2.d0)/a !/3.0D05
+
+    end if
+
+
+end function MGMDOt
+
+!-----------------------------------------------
+! beta(a) function
+function MGBeta(a, adotoa, model)
+    implicit none
+    integer :: model
+    real(dl) :: a, adotoa, MGBeta
+
+    ! SYMMETRON
+    if(model == 7) then
+        MGBeta =  beta_star * sqrt(1.d0-(a_star/a)**3.d0)
+
+    ! DILATON
+    else if (model==8) then
+        MGBeta = beta0 * exp((DilS)/(2.d0* DilR - 3.d0)*(a**(2.d0* DilR - 3.d0)-1.d0))
+
+    ! Hu-Sawicki f(R) model
+    else if (model == 9)then
+        MGBeta = beta0
+
+    ! Simple DILATON model
+    else if (model ==10)then
+        MGBeta = beta0*(a**3.d0)
+    end if
+
+end function MGBeta
+
+!-----------------------------------------------
+! \dot{beta}(a) function
+function MGBetaDot(a, adotoa, model)
+    implicit none
+    integer :: model
+    real(dl) :: a, adotoa, MGBetaDot
+
+    ! SYMMETRON
+    if(model == 7) then
+        MGBetaDot = 1.5d0 * (beta_star * (a_star/a)**3.d0 * adotoa) /( sqrt(1.d0-(a_star/a)**3.d0))
+
+    ! DILATON
+    else if (model==8) then
+        MGBetaDot = MGBeta(a, adotoa, 8) * (DilS * a**(2.d0* DilR - 3.d0) * adotoa)
+
+    ! Hu-Sawicki f(R) model
+    else if (model == 9)then
+        MGBetaDot = 0.d0
+
+    ! Simple DILATON model
+    else if (model ==10)then
+        MGBetaDot = 3.d0 *MGBeta(a,adotoa,10)*adotoa
+
+    end if
+
+end function MGBetaDot
+
+!*************************************************
+!* Q,R parametrization
+!*************************************************
+
+!-----------------------------------------------
+! Q(a,k) function
+function MG_Q(a,adotoa, model)
+    implicit none
+    integer :: model
+    real(dl) :: MG_Q, a, adotoa
+
+    if (model ==2) then
+        MG_Q = MGQfix
+
+    else if (model ==3) then
+        MG_Q = 1.d0 + (Qnot - 1.d0)* a**sss
+
+    end if
+
+end function MG_Q
+
+!-----------------------------------------------
+! \dot{Q}(a,k) function
+function MG_QDot(a,adotoa, model)
+    implicit none
+    integer :: model
+    real(dl) :: MG_QDot, a, adotoa
+
+    if (model ==2) then
+        MG_QDot = 0.d0
+
+    else if (model ==3) then
+        MG_QDot = (Qnot - 1.d0)*adotoa* sss* a**(sss)
+    end if
+
+end function MG_QDot
```

```
+
+!-----------------------------------------------
+! R(a,k) function
+function MG_R(a,adotoa, model)
+    implicit none
+    integer :: model
+    real(dl) :: a,adotoa, MG_R
+
+    if (model ==2) then
+        MG_R=MGRfix
+
+    else if (model ==3) then
+        MG_R = 1.d0 + (Rnot - 1.d0)* a**sss
+
+    end if
+
+end function MG_R
+
+!-----------------------------------------------
+! \dot{R}(a,k) function
+function MG_RDot(a, adotoa, model)
+    implicit none
+    integer :: model
+    real(dl) :: a,adotoa, MG_RDot
+
+    if (model ==2) then
+        MG_RDot = 0.d0
+
+    else if (model ==3) then
+        MG_RDot = (Rnot - 1.d0)*adotoa* sss* a**(sss)
+
+    end if
+
+end function MG_RDot
+
+
+end module mgvariables
+!* MGCAMB mode: end
+!*************************************************************************************
+

     module LambdaGeneral
     use precision
@@ -32,6 +408,30 @@
     !If you are tempted to set this = .false. read
     ! http://cosmocoffee.info/viewtopic.php?t=811
     ! http://cosmocoffee.info/viewtopic.php?t=512
+
+!******************************
+! MGCAMB mod:
+! adding some other variables
+!******************************
+
+! AH: Added but not used !
+ logical :: use_tabulated_w = .false.
+ ! this parameter is already used in CAMB 2015... I comment the following line
+ !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
+
+!* MGCAMB mod: end
+!************************************
+
+
+

     contains

@@ -1184,6 +1584,7 @@
     use ThermoData
     use lvalues
     use ModelData
+    use mgvariables
     implicit none
     integer j
     type(EvolutionVars) EV
@@ -1203,6 +1604,26 @@
     real(dl) clxq, vq, diff_rhopi, octg, octgprime
     real(dl) sources(CTransScal%NumSources)
     real(dl) ISW
+
+!*********************************************************
+!* MGCAMB:
+!* adding some local variables
+!*********************************************************
+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
+real(dl) :: MG_rhoDeltadot, term0, dgpidot
+!* MGCAMB mod: end
+!***********************************************************
+

     yprime = 0
     call derivs(EV,EV%ScalEqsToPropagate,tau,y,yprime)
@@ -1260,12 +1681,40 @@
     end if

     adotoa=sqrt((grho+grhok)/3)
+
+!****************************************************************
+!* MGCAMB:
+!* computing a'' and H' deciding whether or not to switch to MG
+!****************************************************************
+adotdota=(adotoa*adotoa-gpres)/2.d0
+Hdot =adotdota-adotoa**2.d0
+
+! In symmetron GRtrans is replaced by a_star, so distinguish the cases.
+if (model == 7) then
+    if (a< a_star) then
+       tempmodel = 0
+    else
+      tempmodel = model
+    end if
+else
+   if ( a.lt. GRtrans ) then
+      tempmodel = 0
+   else
+      tempmodel = model
+   end if
+end if
+

     if (EV%no_nu_multpoles) 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
+        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
@@ -1276,15 +1725,25 @@
     end if

     if (EV%no_phot_multpoles) 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)
-        qg=-4._dl/3*z
-        pig=0
-        pigdot=0
-        octg=0
-        octgprime=0
-        qgdot = -4*dz/3
+        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)
+            qg=-4._dl/3*z
+            pig=0
+            pigdot=0
+            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)
+            pig=0
+            pigdot=0
+            octg=0
+            octgprime=0
+        end if ! tempmodel /= 0
     else
         if (EV%TightCoupling) then
             pig = EV%pig
@@ -1309,6 +1768,9 @@
         qgdot =yprime(EV%g_ix+1)
     end if

+!* MGCAMB: end
+!*****************************************************************
+
     dgrho = dgrho + grhog_t*clxg+grhor_t*clxr
     dgq   = dgq   + grhog_t*qg+grhor_t*qr
     dgpi  = dgpi  + grhor_t*pir + grhog_t*pig
@@ -1316,18 +1778,114 @@

     !  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)
+
+!********************************************************************
+!* MGCAMB:
+!* if MG then use modified Einstein equations.
+!********************************************************************
```

```
+if (tempmodel /= 0) then
+ ! MU, GAMMA parametrization
+ if (model==1 .or.model==4 .or.model==5.or.model==6 .or. model==7 .or. model==8 .or. model == 9 .or.
model == 10) then
+
+       MG_mu = MGMu(a,adotoa,k2,model)
+       MG_mudot = MGMuDot(a,adotoa,k2,Hdot,model)
+       MG_gamma = MGGamma(a,adotoa,k2,model)
+       MG_gammadot = MGGammaDot(a,adotoa,k2,model)
+
+    ! MG_rhoDelta = \kappa a^2 \sum_i \rho_i (\delta_i + 3 adotoa (1+w_i))
+    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
+    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)
+
+    ! \dot{\eta}
+    etadot = (term1 + term2 + term3 + term4 + term5)/( 2.d0 *fmu)
+
+    !Z
+    z = sigma - 3.d0 * etadot/k
+
+    MG_psi = - MG_mu * ( MG_rhoDelta + 2.d0* dgpi)/(2.d0*k2)
+
+    MG_phi = MG_gamma * MG_psi + MG_mu*1.d0*dgpi/k2
+
+    MG_phidot = etadot - adotoa * (MG_psi - adotoa * MG_alpha)- Hdot * MG_alpha
+
+ ! Q,R parametrization
+ else if ( model ==2.or.model ==3) then
+
+         MGQ = MG_Q(a,adotoa, model)
+         MGR = MG_R(a,adotoa, model)
+         MGQdot = MG_QDot(a,adotoa, model)
+         MGRdot = MG_RDot(a,adotoa, model)
+
+
+      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 * 1.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
+
+!* MGCAMB mod: end
+!********************************************************
 
     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
+!********************************************************
+!* MGCAMB mod:
+!* MGCAMB works only with flat  models
+!********************************************************
+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
+!* MGCAMB mod: end
+!********************************************************
 
 
     if (EV%TightCoupling) then
@@ -1344,6 +1902,19 @@
     pidot_sum =  pidot_sum + grhog_t*pigdot + grhor_t*pirdot
     diff_rhopi = pidot_sum - (4*dgpi+ dgpi_diff )*adotoa
 
+!**********************************************************
+!* MGCAMB: modified ISW effect
+!**********************************************************
+!adding term 0 for MG_rhoDeltadot
+term0 = k2 + 3.d0* (adotoa**2.d0 - Hdot)
+
+!adding MG_rhoDeltadot
+MG_rhoDeltadot = -term0 * dgq/k - (grho + gpres)* k*z - adotoa * MG_rhoDelta - 2.d0 * adotoa * dgpi
+!adding dgpidot
+dgpidot = pidot_sum - (2.d0*dgpi+ dgpi_diff )*adotoa
+
+! GR ISW effect
+if(tempmodel == 0 ) then
     !Maple's fortran output - see scal_eqs.map
     !2phi' term (\phi' + \psi' in Newtonian gauge)
     ISW = (4.D0/3.D0*k*EV%Kf(1)*sigma+(-2.D0/3.D0*sigma-2.D0/3.D0*etak/adotoa)*k &
@@ -1362,7 +1933,42 @@
     vbdot+3.D0/40.D0*qgdot- 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
-
```

```
+
+
+! MG ISW effect
+else
+    ! ISW for mu,gamma parametrization
+    if(model==1 .or. model==4 .or. model==5.or. model==6 .or. model == 7 .or. model ==8 .or. model ==
9 .or. model ==10) then
+        ISW_MG = - (MG_gammadot* MG_mu + MG_gamma* MG_mudot)*0.5d0/k2 * (dgrho + 2.d0*dgpi) - MG_mu*
MG_gamma*0.5d0/k2*&
+                 (MG_rhoDeltadot + 2.d0* dgpidot) - MG_mudot*0.5d0/k2*dgrho - MG_mu*0.5d0/
k2*MG_rhoDeltadot
+
+
+    ! ISW for Q,R parametrization: I have to fix this
+       else if (tempmodel==2.or.tempmodel==3) then
+           MG_psidot = MGR * MG_phidot + MGRdot * MG_phi - ( MGQdot * 2.d0 * dgpi + MGQ * pidot_sum)/k2
+
+
+        ISW_MG = 0.5d0/k2 * ((MGRdot * MGQ + (1.d0 + MGR)* MGQdot)*MG_rhoDelta + (1.d0 +
MGR)*MGQ*MG_rhoDeltadot - 2.d0 &
+                 * MGQdot* dgpi - 2.d0 * MGQ*dgpidot)
+
+    end if
+
+        ISW_MG= expmmu(j) * ISW_MG
+        ISW=ISW_MG
+
+        MG_alphadot= MG_psi - adotoa * MG_alpha
+        polterdot=9._dl/15._dl*ypolprime(2) + 0.1_dl*pigdot
+
+           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
+!* MGCAMB mode end
+!*******************************************************
+
     ! Doppler term
     !   sources(1)=  (sigma+vb)/k*dvis(j)+((-2.D0*adotoa*sigma+vbdot)/k-1.D0/k**2*dgpi)*vis(j) &
     !         +1.D0/k/EV%Kf(1)*vis(j)*etak
@@ -1397,9 +2003,21 @@
         if (tau>tau_maxvis .and. CP%tau0-tau > 0.1_dl) then
             !phi_lens = Phi - 1/2 kappa (a/k)^2 sum_i rho_i pi_i
             phi = -(dgrho +3*dgq*adotoa/k)/(k2*EV%Kf(1)*2) - dgpi/k2/2
-
-            sources(3) = -2*phi*f_K(tau-tau_maxvis)/(f_K(CP%tau0-tau_maxvis)*f_K(CP%tau0-tau))
-            !We include the lensing factor of two here
+!*******************************************************
+!* MGCAMB: MG lensing source
+!*******************************************************
+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 .or. model == 7 .or. model ==8 .or. model==9 .or.
model ==10)&
+    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
+! MGCAMB mod end
+!*******************************************************
+
+    !We include the lensing factor of two here
         else
            sources(3) = 0
         end if
@@ -1926,6 +2544,7 @@
```

```
     !  ayprime is not necessarily GaugeInterface.yprime, so keep them distinct
     use ThermoData
     use MassiveNu
+    use mgvariables
     implicit none
     type(EvolutionVars) EV

@@ -1951,6 +2570,19 @@
     real(dl) dgpi,dgrho_matter,grho_matter, clxnu_all
     !non-flat vars
     real(dl) cothxor !1/tau in flat case
+
+!*******************************************************
+!* MGCAMB:
+!* adding local variables
+!*******************************************************
+real(dl) 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
@@ -1994,25 +2626,50 @@
     dgrho_matter=grhob_t*clxb+grhoc_t*clxc
     !  8*pi*a*a*SUM[(rho_i+p_i)*v_i]
     dgq=grhob_t*vb
+

     if (CP%Num_Nu_Massive > 0) then
         call MassiveNuVars(EV,ay,a,grho_matter,gpres,dgrho_matter,dgq, wnu_arr)
     end if

     grho = grho_matter+grhor_t+grhog_t+grhov_t
-
-    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
-
     dgrho = dgrho_matter

-    if (w_lam /= -1 .and. w_Perturb) then
-        clxq=ay(EV%w_ix)
+!* MGCAMB works only with flat models
+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.htmlfor
updates."
+end if
+
+
+! switch MG on according to the model (in model 7 GRtrans is replaced by a_star)
+if (model == 7) then
+    if (a< a_star) then
+       tempmodel = 0
+    else
+      tempmodel = model
+    end if
+else
+   if ( a.lt. GRtrans ) then
+      tempmodel = 0
+   else
+      tempmodel = model
+   end if
+end if
+
+     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
         dgq = dgq + vq*grhov_t*(1+w_lam)
@@ -2021,11 +2678,17 @@
     if (EV%no_nu_multpoles) then
         !RSA approximation of arXiv:1104.2933, dropping opactity terms in the velocity
         !Approximate total density variables with just matter terms
-        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
-        pir=0
+        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
+            pir=0
+        else ! tempmodel /=0
+            clxr=2*(grhoc_t*clxc+grhob_t*clxb)/3/k**2
+            qr= clxr*k/sqrt((grhoc_t+grhob_t)/3)*(2/3._dl)
+            pir=0
+        end if ! tempmodel
     else
         !  Massless neutrinos
         clxr=ay(EV%r_ix)
@@ -2034,16 +2697,22 @@
     endif

     if (EV%no_phot_multpoles) then
-        if (.not. EV%no_nu_multpoles) then
-            z=(0.5_dl*dgrho/k + etak)/adotoa
-            dz= -adotoa*z - 0.5_dl*dgrho/k
-            clxg=-4*dz/k-4/k*opacity*(vb+z)
-            qg=-4._dl/3*z
-        else
-            clxg=clxr-4/k*opacity*(vb+z)
-            qg=qr
-        end if
-        pig=0
+        if (tempmodel == 0 ) then
+            if (.not. EV%no_nu_multpoles) then
+                z=(0.5_dl*dgrho/k + etak)/adotoa
+                dz= -adotoa*z - 0.5_dl*dgrho/k
+                clxg=-4*dz/k-4/k*opacity*(vb+z)
+                qg=-4._dl/3*z
+            else
+                clxg=clxr-4/k*opacity*(vb+z)
+                qg=qr
+            end if
+            pig=0
+        else ! tempmodel /= 0
+            clxg=2*(grhoc_t*clxc+grhob_t*clxb)/3/k**2
+            qg= clxg*k/sqrt((grhoc_t+grhob_t)/3)*(2/3._dl)
+            pig=0
+        end if ! tempmodel
     else
         !  Photons
         clxg=ay(EV%g_ix)
@@ -2063,8 +2732,134 @@
 
     ayprime(1)=adotoa*a
 
-
-    !  Get sigma (shear) and z from the constraints
+! MGCAMB: anisotropic contribution from massive neutrinos
+dgpi = 0
+if (CP%Num_Nu_Massive > 0) then
+call MassiveNuVarsOut(EV,ay,ayprime,a,dgpi=dgpi)
+end if
+
+dgpi = dgpi + grhor_t*pir + grhog_t*pig
+! Computing Z and sigma with modified Einstein equation
+if (tempmodel /= 0) then
+    ! mu, gamma parametrization
+       if (model == 1 .or. model ==4 .or. model == 5 .or. model == 6 .or. model== 7 .or. model == 8 .or. model == 9 .or. model ==10) then
+
+               MG_mu = MGMu(a,adotoa,k2,model)
+               MG_mudot = MGMuDot(a,adotoa,k2,Hdot, model)
+               MG_gamma = MGGamma(a,adotoa,k2,model)
+               MG_gammadot = MGGammaDot(a,adotoa,k2,model)
+
+        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 moreaccurate 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 + 2.d0* dgpi)/(2.d0*k2)
+
+        MG_phi = MG_gamma * MG_psi + MG_mu* 1.d0*dgpi/k2
+
+        MG_phidot = etadot - adotoa * (MG_psi - adotoa * MG_alpha)- Hdot * MG_alpha
+
+    ! Q,R parametrization
+    else if ( model ==2.or.model ==3) then
+
+        MGQ = MG_Q(a,adotoa, model)
+        MGR = MG_R(a,adotoa, model)
+        MGQdot = MG_QDot(a,adotoa,model)
+        MGRdot = MG_RDot(a,adotoa, model)
+
+        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_psi = MGR * MG_phi - MGQ * 1.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
@@ -2072,12 +2867,17 @@
         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
+    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 .and. ay(1).lt.GRtrans) then

-    if (w_lam /= -1 .and. w_Perturb) then
-        ayprime(EV%w_ix)= -3*adotoa*(cs2_lam-w_lam)*(clxq+3*adotoa*(1+w_lam)*vq/k) &
+!*****************************************************************
+    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

         ayprime(EV%w_ix+1) = -adotoa*(1-3*cs2_lam)*vq + k*cs2_lam*clxq/(1+w_lam)
@@ -2138,9 +2938,21 @@
             !  8*pi*G*a*a*SUM[rho_i*sigma_i]
             dgs = grhog_t*pig+grhor_t*pir

+
+!*****************************************************************
+!* MGCAMB:
+!* shear derivative
+!*****************************************************************
             ! Define shear derivative to first order
-            sigmadot = -2*adotoa*sigma-dgs/k+etak
-
+            !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
+!* MGCAMB mod end
+!*****************************************************************
+
             !Once know slip, recompute qgdot, pig, pigdot
             qgdot = k*(clxg/4._dl-pig/2._dl) +opacity*slip
```

```
--- CAMB-Jan15/inidriver.F90
+++ MGCAMB-Jan15/inidriver.F90

@@ -14,6 +14,10 @@
     use Bispectrum
     use CAMBmain
     use NonLinear
+!******************************
+!* MGCAMB:
+    use mgvariables
+!******************************
 #ifdef NAGF95
     use F90_UNIX
 #endif
@@ -103,6 +107,93 @@
     call DarkEnergy_ReadParams(DefIni)

     P%h0     = Ini_Read_Double('hubble')
+
+
+
+
+!**************************************
+!* MGCAMB mod:
+!* reading models and params
+!**************************************
+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)
+
+! New models added in the last version
+else if (model == 7) then
+! SYMMETRON
```

```
+beta_star = Ini_Read_Double('beta_star', 0.d0)
+xi_star = Ini_Read_Double ('xi_star', 0.d0)
+a_star = Ini_Read_Double('a_star', 0.d0)
+GRtrans = a_star
+
+else if (model == 8) then
+! GENERALIZED DILATON
+beta0 = Ini_Read_Double('beta0', 0.d0)
+xi0 = Ini_Read_Double('xi0', 0.d0)
+DilR = Ini_Read_Double('DilR', 0.d0)
+DilS = Ini_Read_Double('DilS', 0.d0)
+
+else if (model == 9) then
+! HU SAWICKI MODEL
+F_R0 = Ini_Read_Double('F_R0', 0.d0)
+FRn = Ini_Read_Double('FRn', 0.d0)
+beta0 = 1.d0/sqrt(6.d0)
+
+else if (model ==10) then
+! SIMPLE DILATON
+beta0 = Ini_Read_Double('beta0', 0.d0)
+A_2 = Ini_Read_Double('A2',0.d0)
+
+else if (model /= 0) then
+print*, '***please choose a model***'
+stop
+end if
+!* MGCAMB mod end.
+!*****************************************************
+
+
+
+
+
+

     if (Ini_Read_Logical('use_physical',.false.)) then
         P%omegab = Ini_Read_Double('ombh2')/(P%H0/100)**2
```

```
--- CAMB-Jan15/params.ini
+++ MGCAMB-Jan15/params.ini
@@ -1,3 +1,70 @@
+#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= 7 : Symmetron model (introduced in June 2015)
+#model= 8 : Dilaton model (intorduced in June 2015)
+#model= 9 : Large curvature f(R) (introduced in June 2015)
+#model= 10: Aaron dilaton model (introduced in July 2015)
+
+
+model = 0
+
+#Scale factor at which MG is turned on (in model 7 it is replaced by a_star)
+GRtrans= 0.001
+
+#BZ parameters:
+#B1 = 1.3333333
+B1 = 0
+#lambda1_2 = 7500
+lambda1_2 = 0
+#B2 = 0.5
+B2 = 0
+#lambda2_2 = 10000
+lambda2_2 = 0
+#ss = 4
+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 = 0.0001
+beta1 = 1.3333333
+s = 4
+
+# Linder's gamma :
+Linder_gamma = 0.545
+
+#Symmetron models
+beta_star = 1.0d0
+a_star = 0.5d0
+xi_star = 0.001d0
+
+# Dilaton parameters (Simple model uses beta0 and A2, generalized model uses beat0, xi0, S and R)
+beta0 = 1.d0
+xi0 = 0.0001
+DilS = 0.24d0
+DilR = 1.d0
+A2 = 1e3
+
+# Hu-Sawicki model params
+F_R0 = 0.0001d0
+FRn = 1.d0
+
+
+
 #Parameters for CAMB

 #output_root is prefixed to output file names
@@ -7,7 +74,7 @@
 get_scalar_cls = T
 get_vector_cls = F
 get_tensor_cls = F
-get_transfer   = F
+get_transfer   = T

 #if do_lensing then scalar_output_file contains additional columns of l^4 C_l^{pp} and l^3 C_l^{pT}
 #where p is the projected potential. Output lensed CMB Culs (without tensors) are in lensed_output_file
below.
@@ -64,6 +131,7 @@
 #so Neff = massless_neutrinos + sum(massive_neutrinos)
 #For full neutrino parameter details see http://cosmologist.info/notes/CAMB.pdf
 massless_neutrinos = 2.046
+#massless_neutrinos = 3.046

 #number of distinct mass eigenstates
 nu_mass_eigenstates = 1
@@ -159,9 +227,10 @@
 #transfer_k_per_logint=5 samples fixed spacing in log-k
 #transfer_interp_matterpower =T produces matter power in regular interpolated grid in log k;
 # use transfer_interp_matterpower =F to output calculated values (e.g. for later interpolation)
-transfer_high_precision = F
+transfer_high_precision = T
 transfer_kmax           = 2
-transfer_k_per_logint   = 0
+#transfer_k_per_logint   = 0
+transfer_k_per_logint   = 5
 transfer_num_redshifts  = 1
 transfer_interp_matterpower = T
 transfer_redshift(1)    = 0
```