```
--- /cosmomc/source/driver.F90
+++ /MGCosmoMC_Jul_2015/source/driver.F90
@@ -9,6 +9,9 @@
     use GeneralSetup
     use DataLikelihoodList
     use RandUtils
+!-------MG
+use mgvariables
+!-------MG
     implicit none

     character(LEN=:), allocatable :: LogFileName,  numstr, fname, rootdir


@@ -76,6 +80,17 @@

     call Ini%Read('accuracy_level',AccuracyLevel)
     call Ini%Read('stop_on_error',stop_on_error)
+
+
+!--------MG
+call Ini%Read('model',model)
+call Ini%Read('GRtrans', GRtrans)
+
+
+print*, '--------------'
+print*, 'model', model, 'GRtrans', GRtrans
+print*, '--------------'
+!--------MG

     baseroot = Ini%ReadFileName('file_root', NotFoundFail = .true.)
     if(instance<=1) then
```

```
--- /cosmomc/source/Calculator_CAMB.f90
+++ /MGCosmoMC_Jul_2015/source/Calculator_CAMB.f90
@@ -1,6 +1,9 @@
     !Use CAMB
     module Calculator_CAMB
     use CosmologyTypes
+!------MG
+use mgvariables
+!------MG
     use CosmoTheory
     use CAMB, only : CAMB_GetResults, CAMB_GetAge, CAMBParams, CAMB_SetDefParams, &
         AccuracyBoost,  Cl_scalar, Cl_tensor, Cl_lensed, outNone, w_lam, wa_ppf,&


@@ -93,6 +101,34 @@
     P%Reion%delta_redshift = CMB%zre_delta
     w_lam = CMB%w
     wa_ppf = CMB%wa
+!----- MG
+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
+! Adding new models
+! SYMMETRON
+beta_star = CMB%beta_star
+xi_star = CMB%xi_star
```

```
+a_star = CMB%a_star
+! DILATON
+beta0 = CMB%beta0
+xi0 = CMB%xi0
+DilS = CMB%DilS
+DilR = CMB%DilR
+A_2 = CMB%A_2
+! HU-SAWICKI MODEL
+F_R0 = CMB%F_R0
+FRn = CMB%FRn
+!-------MG
+
     ALens = CMB%ALens
     ALens_Fiducial = CMB%ALensf
     P%InitialConditionVector(initial_iso_CDM) = &
```

```
--- /cosmomc/source/CosmologyTypes.f90
+++ /MGCosmoMC_Jul_2015/source/CosmologyTypes.f90
@@ -112,6 +112,29 @@
         real(mcp) YHe, nnu, iso_cdm_correlated, ALens, Alensf, 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)
+
+
+!------MG
+ 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
+
+! new models
+real :: beta_star, xi_star, a_star !symmetron params
+real :: beta0, xi0, DilS, DilR     ! dilaton params
+real :: F_R0, FRn                  ! hu-sawicki model
+real :: A_2                        ! for simple dilaton.
+!-------MG
+
+
     end Type CMBParams

     Type, extends(TParameterization) :: TCosmologyParameterization
```

```
--- /cosmomc/source/CosmologyParameterizations.f90
+++ /MGCosmoMC_Jul_2015/source/CosmologyParameterizations.f90
@@ -68,7 +68,14 @@
     if (CosmoSettings%compute_tensors) call Names%Add('paramnames/derived_tensors.paramnames')
     this%num_derived = Names%num_derived
     !set number of hard parameters, number of initial power spectrum parameters
-    call this%SetTheoryParameterNumbers(16,last_power_index)
+    !call this%SetTheoryParameterNumbers(16,last_power_index)
+
+!-------MG
+call this%SetTheoryParameterNumbers(37, last_power_index)
```

```
+!-------MG
+
+
+

     end subroutine TP_Init

@@ -279,6 +286,9 @@

     subroutine SetForH(Params,CMB,H0, firsttime,error)
     use bbn
+!---------MG
+use mgvariables
+!---------MG
     real(mcp) Params(num_Params)
     logical, intent(in) :: firsttime
     Type(CMBParams) CMB
@@ -325,7 +335,44 @@
         CMB%ALens = Params(14)
         CMB%ALensf = Params(15)
         CMB%fdm = Params(16)
-        call SetFast(Params,CMB)
+!------MG
+! PREVIOUS VERSION MODELS PARAMS
+ CMB%B1 = Params(17)
+ CMB%lambda1_2 = 10.d0**Params(18)
+ CMB%B2 = Params(19)
+ CMB%lambda2_2 = Params(20)
+ CMB%ss = Params(21)
+ CMB%MGQfix = Params(22)
+ CMB%MGRfix = Params(23)
+ CMB%Qnot = Params(24)
+ CMB%Rnot = Params(25)
+ CMB%sss = Params(26)
+ CMB%Linder_gamma = Params(27)
+
+!SYMMETRON
+ CMB%beta_star = Params(28)
+! changing to log scale for some parameters
+! CMB%xi_star = Params(30)
+ CMB%xi_star = 10.d0**Params(30)
+ CMB%a_star = Params(29)
+
+! DILATON
+ CMB%beta0 = Params(31)
+! CMB%xi0 = Params(32)
+ CMB%xi0 = 10.d0**Params(32)
+ CMB%DilS = Params(33)
+ CMB%DilR = Params(34)
+
+! HU-SAWICKI MODEL
+! CMB%F_R0 = Params(35)
+ CMB%F_R0 = 10.d0**Params(35)
+ CMB%FRn = Params(36)
+
+! SIMPLE DILATON
+!CMB%A_2 = Params(37)
+ CMB%A_2 = 10.d0**(- Params(37))
+!-------MG
+
     end if

     CMB%h = CMB%H0/100
@@ -335,6 +382,22 @@
     CMB%omnu = CMB%omnuh2/h2
     CMB%omdm = CMB%omdmh2/h2
     CMB%omv = 1- CMB%omk - CMB%omb - CMB%omdm
+
+!-------MG
+ if (model == 4 ) then
+ CMB%B1 = 4.d0/3.d0
+ CMB%lambda1_2 = 10.d0**Params(19)* ((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 = 10.d0**Params(19)* ((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
+!-------MG
+call SetFast(Params, CMB)

     end subroutine SetForH

@@ -352,6 +415,9 @@
     end subroutine BK_Init

     subroutine BK_ParamArrayToTheoryParams(this, Params, CMB)
+!--------MG
+use mgvariables
+!--------MG
     class(BackgroundParameterization) :: this
     real(mcp) Params(:)
     class(TTheoryParams), target :: CMB
@@ -385,6 +451,54 @@
         CMB%fdm=0
         CMB%iso_cdm_correlated=0
         CMB%Alens=1
+
+!--------MG
+
+ CMB%B1 = Params(17)
+ CMB%lambda1_2 = Params(18)
+ CMB%B2 = Params(19)
+ CMB%lambda2_2 = Params(20)
+ CMB%ss = Params(21)
+ CMB%MGQfix = Params(22)
+ CMB%MGRfix = Params(23)
+ CMB%Qnot = Params(24)
+ CMB%Rnot = Params(25)
+ CMB%sss = Params(26)
+ CMB%Linder_gamma = Params(27)
+
+! SYMMETRON
+ CMB%beta_star = Params(28)
+ CMB%xi_star = 10.d0**Params(30)
+ CMB%a_star = Params(29)
+
+! DILATON
+ CMB%beta0 = Params(31)
+ CMB%xi0 = 10.d0**Params(32)
+ CMB%DilS = Params(33)
+ CMB%DilR = Params(34)
+
+! HU-SAWICKI
+ CMB%F_R0 = 10.d0**Params(35)
+ !print*, "log10 fR0 = ", Params(35), "fR0 = ", CMB%F_R0
+ CMB%FRn = Params(36)
+
+! SIMPLE DILATON
+ CMB%A_2 =10.d0**(- Params(37))
+
+ if (model == 4 ) then
+ CMB%B1 = 4.d0/3.d0
+ CMB%lambda1_2 = Params(19)* ((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(19)* ((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
+!--------MG
+
     end select
     end subroutine BK_ParamArrayToTheoryParams





--- /cosmomc/camb/equations_ppf.f90
+++ /MGCosmoMC_Jul_2015/camb/equations_ppf.f90
@@ -18,7 +18,31 @@
     ! Feb 2013: fixed various issues with accuracy at larger neutrino masses
     ! Oct 2013: fix PPF, consistent with updated equations_cross
     ! Mar 2014: fixes for tensors with massive neutrinos
-
+    ! Jun 2015: MGCAMB patch added (by Alex Zucca azucca@sfu.ca)
+
+!*****************************
+!* MGCAMB mod:
+!* adding the MG variables
+!*****************************
+ 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
+  !***********************************
+  !* New models' parameters
+  !***********************************
+  real(dl) :: beta_star, a_star, xi_star    ! for model 7: symmetron
+  real(dl) :: beta0, xi0, DilR, DilS, A_2   ! for dilaton: model 8 and 10
+  real(dl) :: F_R0, FRn                     ! for model 9: hu-sawicki model
+  real(dl) :: A_2
+ end module mgvariables
+!* MGCAMB mod: end
+!************************************
+
+
     module LambdaGeneral
     use precision
     use  ModelParams

@@ -1333,6 +1371,13 @@
     use ThermoData
     use lvalues
     use ModelData
+!*****************************
+!* MGCAMB mod:
+!* adding MG variables module
+!*****************************
+    use mgvariables
+!* MGCAMB mod: end
+!*****************************
     implicit none
     integer j
     type(EvolutionVars) EV
@@ -1355,6 +1400,32 @@
     real(dl) ISW
     real(dl) w_eff
     real(dl) hdotoh,ppiedot
+
+!****************************************
+!* MGCAMB mod:
+!* adding local variable
+!****************************************
+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
+!* local variables needed for new models
+real(dl) m_a    ! this is m(a)
+real(dl) beta_a ! this is beta(a)
+real(dl) beta_adot, m_adot ! beta'(a), m'(a)
+real(dl) FRm0, FRr
+real(dl) g7, g7dot, h7, h7dot
+!* MGCAMB mod: end
+!*********************************************************************************
+
+

     yprime = 0
     call derivs(EV,EV%ScalEqsToPropagate,tau,y,yprime)
@@ -1416,6 +1487,35 @@
     end if

     adotoa=sqrt((grho+grhok)/3)
+
+
+!****************************************
+!* MGCAMB mod:
+!* decide whether or not to turn MG on
+!****************************************
+adotdota=(adotoa*adotoa-gpres)/2.d0
+Hdot =adotdota-adotoa**2.d0
+!************************************************
+!* In symmetron GRtrans is replaced by a_star,
+!* so here I 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
+!* MGCAMB mod: end
+!************************************************
+
+

     if (EV%no_nu_multpoles) then
         z=(0.5_dl*dgrho/k + etak)/adotoa
@@ -1469,31 +1569,240 @@
     dgq   = dgq   + grhog_t*qg+grhor_t*qr
     dgpi  = dgpi  + grhor_t*pir + grhog_t*pig

-
+
     !  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)
-
+    !  have to get z from eta for numerical stabili
+!*****************************************************
+!* MGCAMB mod:
+!* if MG is on the modify the equations
+!*****************************************************
+if (tempmodel /= 0) then
+!*****************************************************
+!* models which use the 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
+    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 ! correction for f(R) mu function.
+           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
+    !***********************************************************
+    !* Adding Brax et al. parametrization {m(a), beta(a)}
+    !***********************************************************
+    else if(model == 7 .or. model ==8 .or. model == 9 .or. model == 10) then
+       !**********************************************************************
+       !* In case of symmetron and dilaton the mu and gamma parametrizations
+       !* is given in terms of the m,beta params
+       !* computation of the m, beta parametrization of the
+       !* symmetron and dilaton models in the following.
+       !**********************************************************************
+        if(model == 7) then      ! SYMMETRON
+
+            beta_a =  beta_star * sqrt(1.d0-(a_star/a)**3.d0)
+            m_a = (CP%H0/3.0D05) / (xi_star) * sqrt(1.d0-(a_star/a)**3.d0)
+            beta_adot = 3.d0/2.d0 * (beta_star * (a_star/a)**3.d0 * adotoa) /( sqrt(1.d0-(a_star/
a)**3.d0))
+            m_adot = 3.d0/2.d0* (CP%H0/3.0D05)/(xi_star) *( (a_star/a)**3.d0 * adotoa) /( sqrt(1.d0-
(a_star/a)**3.d0))
+
+         else if (model==8) then    ! DILATON
+            !*****************************************************
+            !* Dilaton changed on June 16 with the parametrization
+            !* of paper 1206.3568
+            !*****************************************************
+            m_a = (CP%H0/3.0D05) /(xi0) * a**(- DilR)
+            beta_a = beta0 * exp((DilS)/(2.d0* DilR - 3.d0)*(a**(2.d0* DilR - 3.d0)-1.d0))
+            m_adot = - DilR * m_a * adotoa
+            beta_adot = beta_a * (DilS * a**(2.d0* DilR - 3.d0) * adotoa)
+
+         else if (model == 9)then  ! large curvature f(R)
+
+             beta_a = beta0
+            beta_adot = 0.d0
+            FRm0 = (CP%h0/3.0D05)*sqrt((4.d0*CP%omegav + CP%omegab + CP%omegac)/((FRn+1.d0)*F_R0))!note
factor of c here
+            !************************************************
+            !* parametrization in paper 1205.6583              *
+            !************************************************
+
+             !FRr = 3.d0 * FRn/ 2.d0 +3.d0
+            !m_a = FRm0*a**(-FRr)
+            !m_adot = -FRr * m_A *adotoa
+
+
+             !*************************************
+            !* parametrization of paper 1305.5647 *
+            !*************************************
+            m_a = 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)
+            m_adot = m_a / (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))
+
+         else if (model ==10)then ! simple dilaton model
+
+            beta_a = beta0*(a**3.d0)
+            beta_adot = 3.d0 *beta_a*adotoa
+
+            m_a = sqrt(3.d0*A_2)*(adotoa/a)!/3.0D05 ! H(a) = da/dtau/a**2 = adotoa/a
+            !* Hdot is different from (H/a)dot...
+            m_adot = sqrt(3.d0*A_2)*(Hdot- adotoa**2.d0)/a !/3.0D05
+
+        end if
+        !*********************************************************
+        !* Computing mu and gamma (with derivatives) starting from the
+        !*  m(a) beta(a) parametrization given above
+        !*********************************************************
+        g7 = (2.d0*beta_a**2.d0)*k2
+        h7 = (m_a**2.d0)*a**2.d0
+        g7dot = 4.d0*beta_a*beta_adot*k2
+        h7dot = (2.d0*a**2.d0)*(m_a*m_adot + (m_a**2.d0) *adotoa)
+
+        MG_mu = (k2 + g7 + h7)/(k2 + h7)
+        MG_mudot = (g7dot*(k2 + h7) - g7*h7dot)/((k2 + h7)**2.d0)
+        MG_gamma = (k2 - g7 + h7)/(k2 + g7 + h7)
+        MG_gammadot = 2.d0*(g7*h7dot-g7dot*(k2 + h7))/((k2 + g7 + h7)**2.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/EV%Kf(1)
+ end if
+!* MGCAMB mod:end
```

```
+!*********************************************************************************************
+
+
+
+!*********************************************************
+!* This part is not present in equations.f90. Should I add
+!* if (model == 0) then (do this part)? else nothing..
+!*********************************************************
     if (is_cosmological_constant) then
         ppiedot=0
     else
         hdotoh=(-3._dl*grho-3._dl*gpres -2._dl*grhok)/6._dl/adotoa
         ppiedot=3._dl*EV%dgrho_e_ppf+EV%dgq_e_ppf*(12._dl/k*adotoa+k/adotoa-3._dl/k*(adotoa+hdotoh))+ &
-             grhov_t*(1+w_eff)*k*z/adotoa -2._dl*k2*EV%Kf(1)*(yprime(EV%w_ix)/adotoa-2._dl*y(EV%w_ix))
+         grhov_t*(1+w_eff)*k*z/adotoa -2._dl*k2*EV%Kf(1)*(yprime(EV%w_ix)/adotoa-2._dl*y(EV%w_ix))
         ppiedot=ppiedot*adotoa/EV%Kf(1)
     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
-
+
+
+!*************************************
+!* MGCAMB mod:
+!* restrict to FLAT model
+!*************************************
+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
         if (second_order_tightcoupling) then
@@ -1501,33 +1810,68 @@
             ypolprime(2)= (pigdot/4._dl)*(1+(5._dl/2._dl)*(dopac(j)/opac(j)**2))
         else
             pigdot = -dopac(j)/opac(j)*pig + 32._dl/45*k/opac(j)*(-2*adotoa*sigma  &
-                 +etak/EV%Kf(1)-  dgpi/k +vbdot )
+             +etak/EV%Kf(1)-  dgpi/k +vbdot )
             ypolprime(2)= pigdot/4
         end if
     end if

     pidot_sum =  pidot_sum + grhog_t*pigdot + grhor_t*pirdot
     diff_rhopi = pidot_sum - (4*dgpi+ dgpi_diff )*adotoa + ppiedot
+
+!******************************************
+!* MGCAMB mod:
+!* sources(1) in GR
+!******************************************
+if(tempmodel == 0 ) then
+!* MGCAMB mod: end
+!******************************************
```

```
     !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 &
-         -diff_rhopi/k**2-1.D0/adotoa*dgrho/3.D0+(3.D0*gpres+5.D0*grho)*sigma/k/3.D0 &
-         -2.D0/k*adotoa/EV%Kf(1)*etak)*expmmu(j)
+    -diff_rhopi/k**2-1.D0/adotoa*dgrho/3.D0+(3.D0*gpres+5.D0*grho)*sigma/k/3.D0 &
+    -2.D0/k*adotoa/EV%Kf(1)*etak)*expmmu(j)

     !e.g. to get only late-time ISW
     !  if (1/a-1 < 30) ISW=0

     !The rest, note y(9)->octg, yprime(9)->octgprime (octopoles)
     sources(1)= ISW +  ((-9.D0/160.D0*pig-27.D0/80.D0*ypol(2))/k**2*opac(j)+ &
-         (11.D0/10.D0*sigma- 3.D0/8.D0*EV%Kf(2)*ypol(3)+vb-9.D0/80.D0*EV%Kf(2)*octg+3.D0/40.D0*qg)/k- &
-         (-180.D0*ypolprime(2)-30.D0*pigdot)/k**2/160.D0)*dvis(j) + &
-         (-(9.D0*pigdot+ 54.D0*ypolprime(2))/k**2*opac(j)/160.D0+pig/16.D0+clxg/4.D0+3.D0/8.D0*ypol(2) + &
-         (-21.D0/5.D0*adotoa*sigma-3.D0/8.D0*EV%Kf(2)*ypolprime(3) + &
-         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
+    (11.D0/10.D0*sigma- 3.D0/8.D0*EV%Kf(2)*ypol(3)+vb-9.D0/80.D0*EV%Kf(2)*octg+3.D0/40.D0*qg)/k- &
+    (-180.D0*ypolprime(2)-30.D0*pigdot)/k**2/160.D0)*dvis(j) + &
+    (-(9.D0*pigdot+ 54.D0*ypolprime(2))/k**2*opac(j)/160.D0+pig/16.D0+clxg/4.D0+3.D0/8.D0*ypol(2) + &
+    (-21.D0/5.D0*adotoa*sigma-3.D0/8.D0*EV%Kf(2)*ypolprime(3) + &
+    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
+
+!*************************************************
+!* MGCAMB mod:
+!* sources(1) in MG
+!*************************************************
+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) 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
     !   sources(1)=  (sigma+vb)/k*dvis(j)+((-2.D0*adotoa*sigma+vbdot)/k-1.D0/k**2*dgpi)*vis(j) &
@@ -1563,8 +1907,24 @@
         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))
+!*************************************************
+!* MGCAMB mod:
+!* sources(3): GR or MG.
+!*************************************************
+            !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 .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
+!************************************************
+
+            !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
         else
             sources(3) = 0

@@ -2091,6 +2451,13 @@
     !  ayprime is not necessarily GaugeInterface.yprime, so keep them distinct
     use ThermoData
     use MassiveNu
+!******************************
+!* MGCAMB mod:
+!* adding mgvariables
+!******************************
+    use mgvariables
+!* MGCAMB mod: end
+!******************************
     implicit none
     type(EvolutionVars) EV



@@ -2119,6 +2486,28 @@
     !ppf
     real(dl) Gamma,S_Gamma,ckH,Gammadot,Fa,dgqe,dgrhoe, vT
     real(dl) w_eff, grhoT
+
+!******************************************
+!* MGCAMB mod:
+!* adding local variables
+!******************************************
+! the variable dgpi is already used in CAMB 2015, I will comment the following line
+!real(dl) dgpi
+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
+
+real(dl) beta_a, beta_adot
+real(dl) m_a, m_adot
+real(dl) FRm0, FRr
+real(dl) g7, g7dot, h7, h7dot !other useful numerical variables
+!* MGCAMB mod: end
+!*********************************************
+

     k=EV%k_buf
     k2=EV%k2_buf
@@ -2165,29 +2554,68 @@
     dgrho_matter=grhob_t*clxb+grhoc_t*clxc
     !  8*pi*a*a*SUM[(rho_i+p_i)*v_i]
     dgq=grhob_t*vb
+
+!**************************************
+!* Adding this part (as in MGCAMB)
```

```
+!*************************************
+dgpi = grhor_t*pir + grhog_t*pig
+!*************************************

     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)
-    !    vq=ay(EV%w_ix+1)
-    !    dgrho=dgrho + clxq*grhov_t
-    !    dgq = dgq + vq*grhov_t*(1+w_lam)
+ 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
+
+!***********************************************
+!* MGCAMB mod:
+!* deicide whether or not to turn MG on
+!***********************************************
+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
+
+
+dgrho = dgrho_matter
+!* mGCAMB mod: end
+!**************************************
+
+
+    !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
+     !   dgq = dgq + vq*grhov_t*(1+w_lam)
     !end if
+

     if (EV%no_nu_multpoles) then
         !RSA approximation of arXiv:1104.2933, dropping opactity terms in the velocity
@@ -2234,6 +2662,10 @@

     ayprime(1)=adotoa*a

+
+!*****************************************************
+!* The following part is not present in equations.f90
+!*****************************************************
     if (.not. is_cosmological_constant) then
         !ppf
         grhoT = grho - grhov_t
@@ -2269,28 +2701,280 @@

     !  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
-    !
-    !   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)
-    !
+
+!******************************
+!* MGCAMB mod:
+!* if MG then changed equations
+!******************************
+
+    !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 .or. model== 7 .or. model == 8 .or.
model == 9 .or. model == 10) 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
+            else if(model == 7 .or. model ==8 .or. model == 9 .or. model == 10) then
+     !***************************************************************************
+     !* In case of symmetron and dilaton the mu and gamma parametrizations
+     !* is given in terms of the m,beta params
+     !* computation of the m, beta parametrization of the
+     !* symmetron and dilaton models in the following.
+     !***************************************************************************
+         if(model == 7) then      ! SYMMETRON
+
+             beta_a =  beta_star * sqrt(1.d0-(a_star/a)**3.d0)
+             m_a = (CP%H0/3.0D05) / (xi_star) * sqrt(1.d0-(a_star/a)**3.d0)
+             beta_adot = 3.d0/2.d0 * (beta_star * (a_star/a)**3.d0 * adotoa) /( sqrt(1.d0-(a_star/
a)**3.d0))
+             m_adot = 3.d0/2.d0* (CP%H0/3.0D05)/(xi_star) *( (a_star/a)**3.d0 * adotoa) /( sqrt(1.d0-
(a_star/a)**3.d0))
+
+          else if (model==8) then    ! DILATON
+
+             m_a = (CP%H0/3.0D05) /(xi0) * a**(- DilR)
+             beta_a = beta0 * exp((DilS)/(2.d0* DilR - 3.d0)*(a**(2.d0* DilR - 3.d0)-1.d0))
+             m_adot = - DilR * m_a * adotoa
+             beta_adot = beta_a * (DilS * a**(2.d0* DilR - 3.d0) * adotoa)
+
+           else if (model == 9)then  ! large curvature f(R)
+
+              beta_a = beta0
+             beta_adot = 0.d0
+             FRm0 = (CP%h0/3.0D05)*sqrt((4.d0*CP%omegav + CP%omegab + CP%omegac)/((FRn+1.d0)*F_R0))!note
factor of c here
+             !**************************************************
+             !* parametrization in paper 1205.6583              *
+             !**************************************************
+
+              !FRr = 3.d0 * FRn/ 2.d0 + 3.d0
+             !m_a = FRm0*a**(-FRr)
+             !m_adot = -FRr * m_A *adotoa
+
+              !*************************************
+             !* parametrization in paper 1305.5647 *
+             !*************************************
+             m_a = 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)
+             m_adot = m_a / (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))
+          else if (model ==10)then ! Aaron's dilaton model
+
+             beta_a = beta0*(a**3.d0)
+             beta_adot = 3.d0 *beta_a*adotoa
+
+             m_a = sqrt(3.d0*A_2)*(adotoa/a)!/3.0D05 ! H(a) = da/dtau/a**2 = adotoa/a
+
+             ! Hdot is different from (H/a)dot....
+             m_adot = sqrt(3.d0*A_2)*(Hdot- adotoa**2.d0)/a !/3.0D05
+
+        end if
+
+
+                g7 = (2.d0*beta_a**2.d0)*k2
+                h7 = (m_a**2.d0)*a2
+                g7dot = 4.d0*beta_a*beta_adot*k2
+                h7dot = (2.d0*a**2.d0)*(m_a*m_adot + (m_a**2.d0) *adotoa)
+
+                MG_mu = (k2 + g7 + h7)/(k2 + h7)
+                MG_mudot = (g7dot*(k2 + h7) - g7*h7dot)/((k2 + h7)**2.d0)
+                MG_gamma = (k2 - g7 + h7)/(k2 + g7 + h7)
+                MG_gammadot = 2.d0*(g7*h7dot-g7dot*(k2 + h7))/((k2 + g7 + h7)**2.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
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 + 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
+!* MGCAMB mod: end
+!*******************************************
+
+
+
+! 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
+
+      !  ayprime(EV%w_ix+1) = -adotoa*(1-3*cs2_lam)*vq + k*cs2_lam*clxq/(1+w_lam)
+    !end if

     if (associated(EV%OutputTransfer)) then
-        EV%OutputTransfer(Transfer_kh) = k/(CP%h0/100._dl)
+        EV%OutputTransfer(Transfer_kh) = k/(CP%h0/100._dl)
         EV%OutputTransfer(Transfer_cdm) = clxc
         EV%OutputTransfer(Transfer_b) = clxb
         EV%OutputTransfer(Transfer_g) = clxg


@@ -2344,19 +3028,30 @@
             !  8*pi*G*a*a*SUM[rho_i*sigma_i]
             dgs = grhog_t*pig+grhor_t*pir

+!*********************************************
+!* MGCAMB mod:
+!* sigmadot in GR or MG.
+!*********************************************
             ! 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



--- /cosmomc/batch2/params_CMB_defaults.ini
+++ /MGCosmoMC_Jul_2015/batch2/params_CMB_defaults.ini

@@ -61,3 +78,86 @@
 #defining l_max for actual calculation, and higher L template file
 highL_theory_cl_template = %DATASETDIR%HighL_lensedCls.dat

+
+
+
+#====================================
+#
+# Adding MGCAMB patch
+#
+#====================================
+
+#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= 7 : Symmetron    (introduced in June 2015)
+#model= 8 : Dilaton      (introduced in June 2015)
+#model= 9 : Large curvature f(R)    (introduced in June 2015)
+#model=10 : Aaron dilaton model (introduced for comparison in July 2015)
+
+model = 0
+
+#Use GRtrans = 0.001 to avoid problems at early time cosmology.
+GRtrans= 0.001
+
+param[B1] = 1.0 1.0 1.0 0 0
+#1.125 1.1 1.14 0.1 0.1
+
+#For BZ models :
+#param[log10lambda1_2] = 750 750 750 0 0
+#0.67e4 0.6e4 0.7e4 10 10
+
+# For f(R) and chameleon models (this is log B0) :
+param[log10lambda1_2] = 0.5 0.5 0.5 0 0
+
+param[B2] = 0.5 0.5 0.5 0 0
+#0.78 0.6 0.9 0.1 0.1
+
+param[lambda2_2] = 1000 1000 1000 0 0
+#1.0e4 0.1e4 10.0e4 1 1
+
+param[ss] = 4.0 4.0 4.0 0 0
+#2 1 4 0.1 0.1
+
+param[MGQfix]= 0.6 0.6 0.6 0 0
+
+param[MGRfix] = 0.7 0.7 0.7 0 0
+
+param[Qnot] = 0.5  0.5 0.5 0 0
+#1 0.5 1.5 0.03 0.03
+
+
+param[Rnot]= 0.6 0.6 0.6 0 0
+#1 0.5 1.5 0.03 0.03
```

```
+
+#0 0 0 0 0
+
+param[sss] = 1 1 1 0 0
+#1 0.5 1.5 0.03 0.03
+
+
+param[Linder_gamma] =0.545 0.545 0.545 0 0
+
+#Symmetron parameters
+param[log10xi_star] = -5.0 -10.0 -0.3 0 0
+param[beta_star] = 1.0 1.0 1.0 0 0
+param[a_star] = 0.5 0.5 0.5 0 0
+
+#Dilaton parameters (simple model uses beta0 and A_2 only, generalized model uses beta0, xi0, s and r)
+param[beta0] = 0.5 0.5 0.5 0 0
+param[log10xi0] = -5.0 -10.0 -0.3 0 0
+param[DilS] = 0.24 0.24 0.24 0 0
+param[DilR] = 1.0 1.0 1.0 0 0
+param[-log10A_2] = -6.0 -10.0 -3.0 0 0
+
+#Large Curvature f(R)
+param[log10F_R0] = -5.0 -9.0 1.0 0.3 0.3
+param[FRn] = 1.0
+# 0.01 50.0  1.0 1.0
+#=====================================================0
+
```

```
--- /cosmomc/paramnames/params_CMB.paramnames
+++ /MGCosmoMC_Jul_2015/paramnames/params_CMB.paramnames
@@ -14,6 +14,27 @@
 Alens         A_{L}     #lensing potential scaled by sqrt(A_lens)
 Alensf        A^f_L     #amplitude of smoothing power from fiducial models
 fdm           \epsilon_0 f_d    #CosmoRec dark matter annihilation parameter, 0910.3663
+B1            \beta_1
+log10lambda1_2     \log_{10}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
+beta_star     \beta_{\star}
+a_star        a_{\star}
+log10xi_star  \log_{10} \xi_{\star}
+beta0         \beta_0
+log10xi0      \log_{10} \xi_0
+DilS          s_{dil}
+DilR          r_{dil}
+log10F_R0     \log_{10}  f_{R, 0}
+FRn           n_{f(R)}
+-log10A_2     - \log_{10} A_2
 logA          {\rm{ln}}(10^{10} A_s)
 ns            n_s          #beware that pivot scale can change in .ini file
 nrun          n_{\rm run}
```