# Scalar field as dark matter

## ... and how to include it in CLASS

Luis A. Ureña-López
Departamento de Física, UG

Métodos Numéricos y Estadísticos en Cosmología Cinvestav 2017
en colaboración con Francisco Linares y Alma González
El código se puede descargar de:
https://github.com/lurena-lopez/class.QuintF.git

# Equations of motion: background (original)

$$H^2 = \frac{\kappa^2}{3}\left(\sum_I \rho_I + \rho_\phi\right), \tag{2.1a}$$

$$\dot{H} = -\frac{\kappa^2}{2}\left[\sum_I (\rho_I + p_I) + (\rho_\phi + p_\phi)\right], \tag{2.1b}$$

$$\dot{\rho}_I = -3H(\rho_I + p_I), \tag{2.1c}$$

$$\ddot{\phi} = -3H\dot{\phi} - m^2\phi, \tag{2.1d}$$

where $\kappa^2 = 8\pi G$, a dot denotes derivative with respect to cosmic time $t$, and $H$ is the Hubble parameter. The scalar field energy density $\rho_\phi$ and pressure $p_\phi$ are given by the known expressions:

$$\rho_\phi = \frac{1}{2}\dot{\phi}^2 + \frac{1}{2}m^2\phi^2, \quad p_\phi = \frac{1}{2}\dot{\phi}^2 - \frac{1}{2}m^2\phi^2. \tag{2.2}$$

# Equations of motion: background (transformed)

$$\frac{\kappa\dot{\phi}}{\sqrt{6}H} \equiv \Omega_\phi^{1/2}\sin(\theta/2), \qquad \frac{\kappa V^{1/2}}{\sqrt{3}H} \equiv \Omega_\phi^{1/2}\cos(\theta/2)\,, \tag{3a}$$

$$y_1 \equiv -2\,\sqrt{2}\,\frac{\partial_\phi V^{1/2}}{H}\,, \tag{3b}$$

simply given by

$$w_\phi \equiv \frac{p_\phi}{\rho_\phi} = \frac{x^2-y^2}{x^2+y^2} = -\cos\theta\,. \tag{2.6}$$

That is, the new angular variable $\theta$ is directly related to the scalar field EoS.

After some straightforward algebra, the KG equation (2.1d) becomes:

$$\theta' = -3\sin\theta + y_1\,, \tag{2.7a}$$

$$y_1' = \frac{3}{2}\,(1+w_{tot})\,y_1\,, \tag{2.7b}$$

$$\Omega_\phi' = 3(w_{tot}-w_\phi)\Omega_\phi\,. \tag{2.7c}$$

# Background: initial conditions

$$\frac{m}{H_0} = \frac{H_i}{H_0}\frac{y_{1i}}{2} = \frac{5}{2}\Omega_{r0}^{1/2} a_i^{-2} \theta_i$$

$$\Omega_{\phi i} \simeq a_i \frac{\Omega_{\phi 0}}{\Omega_{r0}} \left(\frac{a_i}{a_{osc}}\right)^3 = a_i \frac{\Omega_{\phi 0}}{\Omega_{r0}} \left[\frac{4\theta_i^2}{\pi^2}\left(\frac{9+\pi^2/4}{9+\theta_i^2}\right)\right]^{3/4} .$$

# Implementation in CLASS (initial conditions)

```
/** - Omega_0_lambda (cosmological constant), Omega0_fld (dark energy fluid), Omega0_scf (scalar field) */

class_call(parser_read_double(pfc,"Omega_Lambda",&param1,&flag1,errmsg),
           errmsg,
           errmsg);
class_call(parser_read_double(pfc,"Omega_fld",&param2,&flag2,errmsg),
           errmsg,
           errmsg);
class_call(parser_read_double(pfc,"Omega_scf",&param3,&flag3,errmsg),
           errmsg,
           errmsg);

class_test((flag1 == _TRUE_) && (flag2 == _TRUE_) && ((flag3 == _FALSE_) || (param3 >= 0.)),
           errmsg,
           "In input file, either Omega_Lambda or Omega_fld must be left unspecified, except if Omega_scf is set and <0.0, in which
              case the contribution from the scalar field will be the free parameter.");

/** - --> (flag3 == _FALSE_) || (param3 >= 0.) explained:
 *  it means that either we have not read Omega_scf so we are ignoring it
 *  (unlike lambda and fld!) OR we have read it, but it had a
 *  positive value and should not be used for filling.
 *  We now proceed in two steps:
 *  1) set each Omega0 and add to the total for each specified component.
 *  2) go through the components in order {lambda, fld, scf} and
 *     fill using first unspecified component.
*/
```

# Implementation in CLASS (initial conditions)

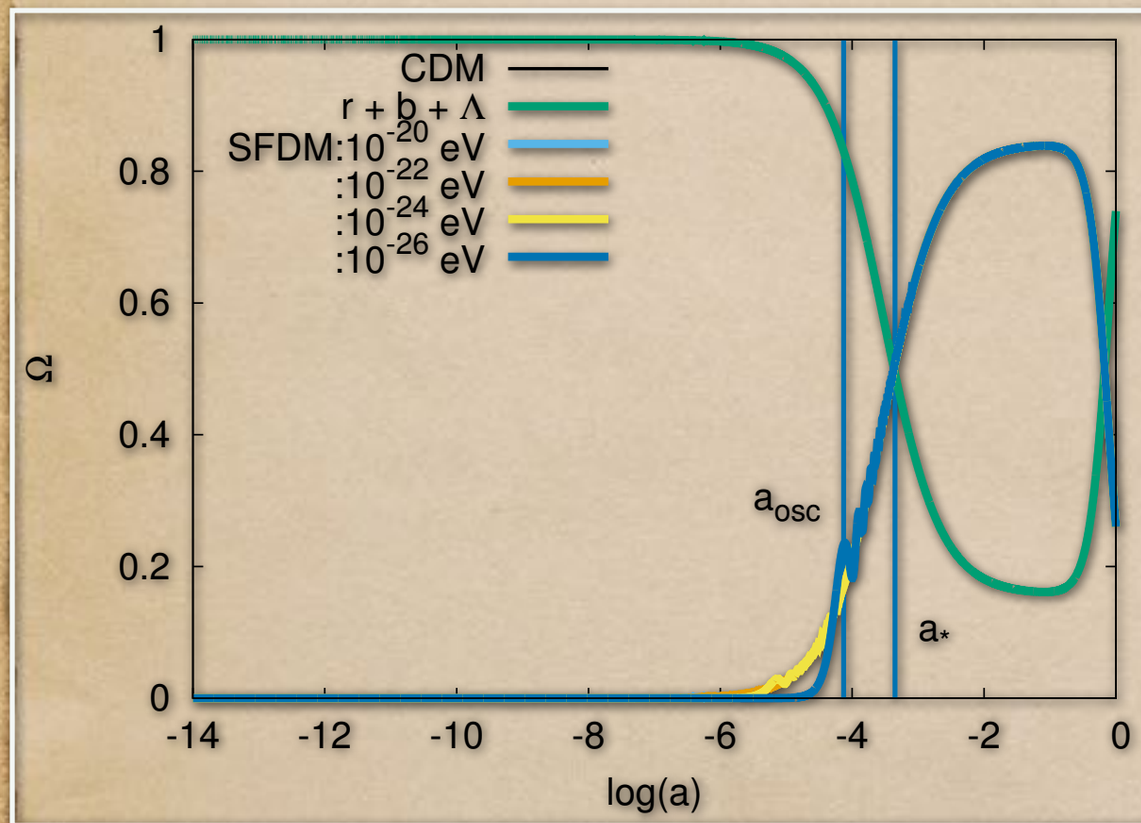
```
/* Additional SCF parameters: */
  if (pba->Omega0_scf != 0.){
  /** - Read parameters describing scalar field potential */
  class_call(parser_read_list_of_doubles(pfc,
                                         "scf_parameters",
                                         &(pba->scf_parameters_size),
                                         &(pba->scf_parameters),
                                         &flag1,
                                         errmsg),
             errmsg,errmsg);
  class_read_int("scf_tuning_index",pba->scf_tuning_index);
  class_test(pba->scf_tuning_index >= pba->scf_parameters_size,
             errmsg,
             "Tuning index scf_tuning_index = %d is larger than the number of entries %d in scf_parameters.
                 Check your .ini file.",pba->scf_tuning_index,pba->scf_parameters_size);
  /** - Assign shooting parameter */
  class_read_double("scf_shooting_parameter",pba->scf_parameters[pba->scf_tuning_index]);

  /** - Initial conditions for scalar field variables */
  /** - Conversion of the boson mass into initial conditions */
  theta_ini = 0.4*15.64*pba->scf_parameters[1]/(pow(pba->Omega0_g+pba->Omega0_ur,0.5)*pba->H0);
  /** - Find the scale factor at the start of field oscillations */
  aosc = pow((0.5*_PI_/theta_ini)/pow(1.+pow(_PI_,2)/36.,0.5),0.5);
  /** - Calculate pivot value of Omega_phi_init for the calculation of appropriate initial conditions */
  Omega_ini = pba->scf_parameters[pba->scf_tuning_index]+log(pba->Omega0_scf*1.e-14/(pow(aosc,3.)*(pba->
      Omega0_g+pba->Omega0_ur)));
  /** - Set up initial conditions */
  pba->Omega_phi_ini_scf = Omega_ini;
  pba->theta_phi_ini_scf = theta_ini;
```

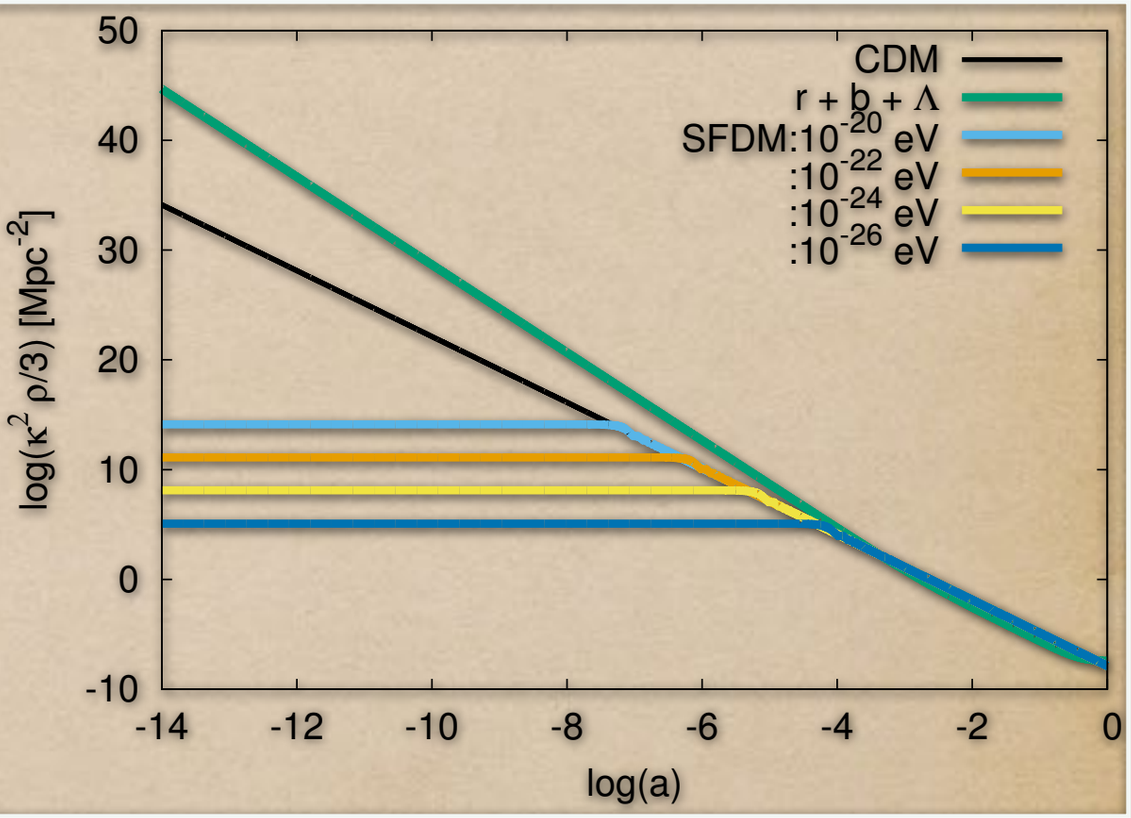
# Implementation in CLASS (equations of motion)

```
  /* Scalar field */
if (pba->has_scf == _TRUE_) {
  // First we update B quantities
  Omega_phi = pvecback_B[pba->index_bi_Omega_phi_scf];
  theta_phi = pvecback_B[pba->index_bi_theta_phi_scf];
  y1_phi = pvecback_B[pba->index_bi_y_phi_scf];
  pvecback[pba->index_bg_Omega_phi_scf] = Omega_phi; // value of the scalar field Omega_phi
  pvecback[pba->index_bg_theta_phi_scf] = theta_phi; // value of the scalar field theta_phi
  pvecback[pba->index_bg_y_phi_scf] = y1_phi; // value of the scalar field y1_phi
  pvecback[pba->index_bg_rho_scf] = exp(Omega_phi)*rho_tot/(1.-exp(Omega_phi)); // energy of the scalar field
  pvecback[pba->index_bg_p_scf] = -cos_scf(pba,theta_phi)*pvecback[pba->index_bg_rho_scf];// pressure of the scalar field
  rho_m += pvecback[pba->index_bg_rho_scf]; // add scalar field energy density into the total matter budget
  rho_tot += pvecback[pba->index_bg_rho_scf]; // add scalar field density to the total one
  p_tot += pvecback[pba->index_bg_p_scf]; // add scalar field pressure to the total one
}
```

```
if (pba->has_scf == _TRUE_){
  dy[pba->index_bi_Omega_phi_scf] = 3.*y[pba->index_bi_a]*pvecback[pba->index_bg_H]*
    (pvecback[pba->index_bg_w_tot]+cos_scf(pba,y[pba->index_bi_theta_phi_scf]));

  dy[pba->index_bi_theta_phi_scf] = y[pba->index_bi_a]*pvecback[pba->index_bg_H]*
    (-3.*sin_scf(pba,y[pba->index_bi_theta_phi_scf])+y[pba->index_bi_y_phi_scf]);

  dy[pba->index_bi_y_phi_scf] = y[pba->index_bi_a]*pvecback[pba->index_bg_H]*
    (1.5*(1.+pvecback[pba->index_bg_w_tot])*y[pba->index_bi_y_phi_scf]
     + y2_phi_scf(pba,y[pba->index_bi_Omega_phi_scf],y[pba->index_bi_theta_phi_scf],y[pba->index_bi_y_phi_scf])*
     exp(0.5*y[pba->index_bi_Omega_phi_scf])*sin_scf(pba,0.5*y[pba->index_bi_theta_phi_scf]));
}
```

# Numerical results



# Equations of motion: linear perturbations (original)

$$\ddot{\varphi} = -3H\dot{\varphi} - (k^2/a^2 + m^2)\varphi - \frac{1}{2}\dot{\phi}\dot{h}\,, \tag{3.1}$$

where $h$ is the trace of scalar metric perturbations (with $\dot{h}$ known as the metric continuity), and $k$ is a comoving wavenumber. The perturbations in density $\delta\rho_\phi$, pressure $\delta p_\phi$, and velocity divergence $\Theta_\phi$, are given, respectively, by[71, 72, 75]:

$$\delta\rho_\phi = \dot{\phi}\dot{\varphi} + \partial_\phi V\,\varphi\,, \tag{3.2a}$$

$$\delta p_\phi = \dot{\phi}\dot{\varphi} - \partial_\phi V\,\varphi\,, \tag{3.2b}$$

$$(\rho_\phi + p_\phi)\Theta_\phi = (k^2/a)\dot{\phi}\varphi\,. \tag{3.2c}$$

# Equations of motion: linear perturbations (transformed I)

$$u = \sqrt{\frac{2}{3}}\frac{\kappa\dot{\varphi}}{H} = -\Omega_\phi^{1/2} e^\alpha \cos(\vartheta/2)\,, \tag{3.3a}$$

$$v = \sqrt{\frac{2}{3}}\frac{\kappa m\varphi}{H} = -\Omega_\phi^{1/2} e^\alpha \sin(\vartheta/2)\,. \tag{3.3b}$$

on of the new variables $\vartheta$ and $\alpha$, it is useful to define the density $\delta_\phi \equiv$
$, \equiv \delta p_\phi/\rho_\phi$ contrasts, respectively, from the physical quantities (3.2).

$$\delta_\phi = -e^\alpha \sin(\theta/2 - \vartheta/2)\,, \tag{3.4a}$$

$$\delta_{p_\phi} = -e^\alpha \sin(\theta/2 + \vartheta/2)\,, \tag{3.4b}$$

$$(\rho_\phi + p_\phi)\Theta_\phi = -\frac{k^2}{am}\rho_\phi e^\alpha \sin(\theta/2)\sin(\vartheta/2)\,. \tag{3.4c}$$

# Equations of motion: linear perturbations (transformed I)

$$\sqrt{\frac{2}{3}}\frac{\kappa\dot{\varphi}}{H} = -\Omega_\phi^{1/2} e^{\alpha}\cos(\vartheta/2)\,,$$

$$\sqrt{\frac{2}{3}}\frac{\kappa m\varphi}{H} = -\Omega_\phi^{1/2} e^{\alpha}\sin(\vartheta/2)\,.$$

$$\vartheta' = 3\sin\vartheta + y_1 + 2\omega\sin^2(\vartheta/2) - 2e^{-\alpha}h'\sin(\theta/2)\sin(\vartheta/2)\,,$$

$$\alpha' = -\frac{3}{2}\left(\cos\vartheta + \cos\theta\right) - \frac{\omega}{2}\sin\vartheta + e^{-\alpha}h'\sin(\theta/2)\cos(\vartheta/2)\,.$$

# Equations of motion: linear perturbations (transformed II)

$$\delta_0' = \left[-3\sin\theta - \frac{k^2}{k_J^2}(1-\cos\theta)\right]\delta_1 + \frac{k^2}{k_J^2}\sin\theta\,\delta_0$$
$$-\frac{\bar{h}'}{2}(1-\cos\theta)\,,$$

$$\delta_1' = \left[-3\cos\theta - \frac{k^2}{k_J^2}\sin\theta + \frac{\lambda\Omega_\phi}{2y_1}\sin\theta\right]\delta_1$$
$$+\left(\frac{k^2}{k_J^2} - \frac{\lambda\Omega_\phi}{2y_1}\right)(1+\cos\theta)\;\delta_0 - \frac{\bar{h}'}{2}\sin\theta\,,$$

$$k_J^2 \equiv a^2 H^2 y_1$$

# Linear perturbations: initial conditions

$$\vartheta = (5/6)\theta\,, \quad e^{\alpha} = (1/7)h\theta\,.$$

$$\delta_{0i} = -\bar{h}_i\theta_i^2/84 \text{ and } \delta_{1i} = -\bar{h}_i\theta_i/7,$$

# Implementation in CLASS (initial conditions)

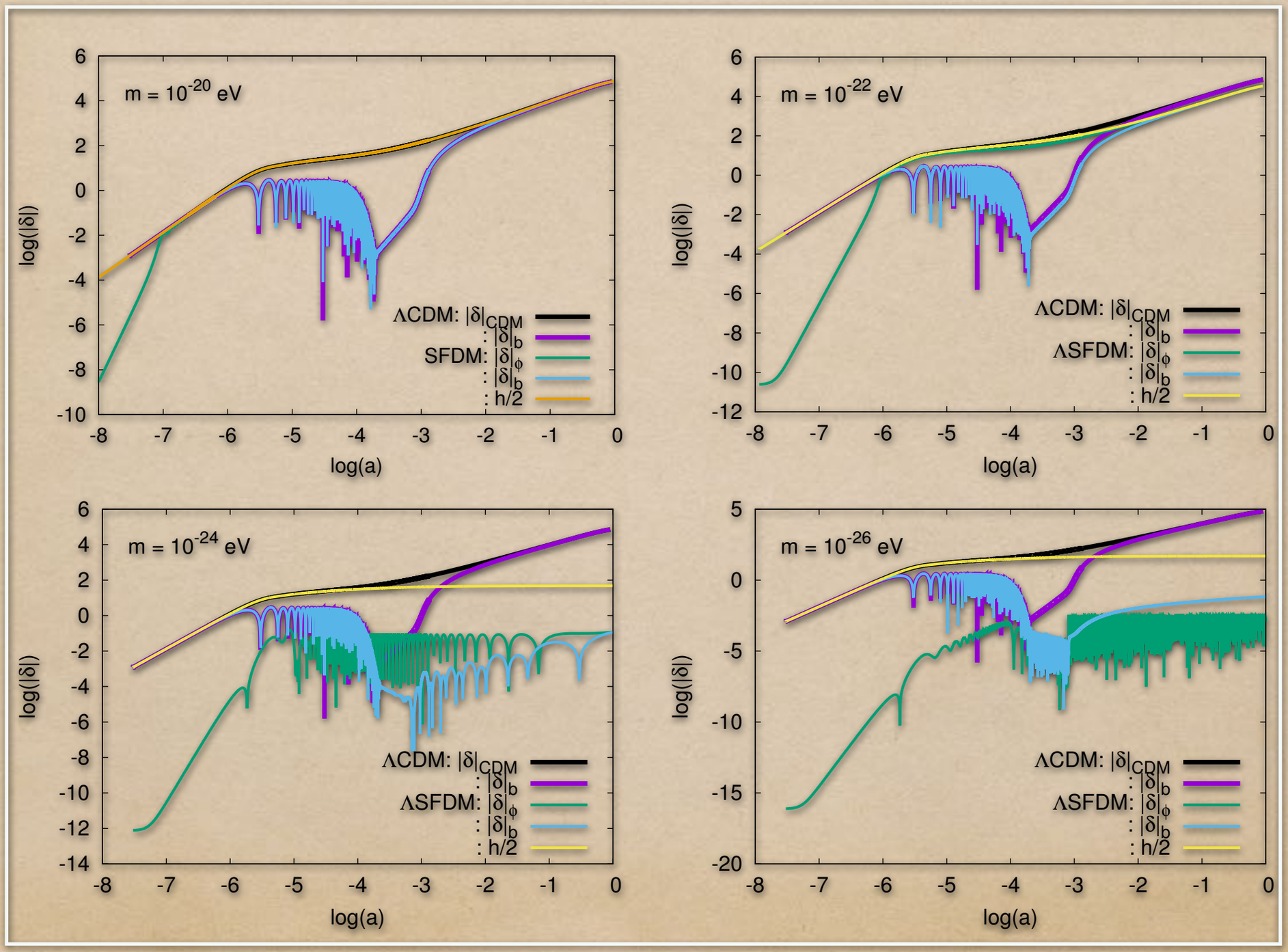
```
if (pba->has_scf == _TRUE_) {
/* also check that the scf is slowly-rolling */
Omega_phi = ppw->pvecback[pba->index_bg_Omega_phi_scf];
theta_phi = ppw->pvecback[pba->index_bg_theta_phi_scf];
y1_phi = ppw->pvecback[pba->index_bg_y_phi_scf];
/** Following expression comes from a trigonometric-hyperbolic identity for mass_scf */
m_scf_over_H = 0.5*pow(pow(y1_phi,2.)+2.*y2_phi_scf(pba,Omega_phi,theta_phi,y1_phi)*exp(0.5*Omega_phi)*
    cos_scf(pba,0.5*theta_phi),0.5);

 if (m_scf_over_H > 1.e-2)
 is_early_enough = _FALSE_;
 }
```

```
if (pba->has_scf == _TRUE_) {

    /** Scalar field variables from the background */
    theta_phi = ppw->pvecback[pba->index_bg_theta_phi_scf];
    y1_phi = ppw->pvecback[pba->index_bg_y_phi_scf];

    /** Initial conditions in new formalism. Notice that by definition: omega=k^2/(H^2 y1) */
    ppw->pv->y[ppw->pv->index_pt_omega_scf] = k*k/(pow(a*ppw->pvecback[pba->index_bg_H],2.)*y1_phi);

    /** Canonical field (solving for the perturbations):
     Attractor solution around the critical point of the perturbation equations */
    ppw->pv->y[ppw->pv->index_pt_delta0_scf] = (3./7.)*ppw->pv->y[ppw->pv->index_pt_delta_g]*sin(0.5*
        theta_phi)*sin(theta_phi/12.);
    ppw->pv->y[ppw->pv->index_pt_delta1_scf] = (3./7.)*ppw->pv->y[ppw->pv->index_pt_delta_g]*sin(0.5*
        theta_phi)*cos(theta_phi/12.);
}
```

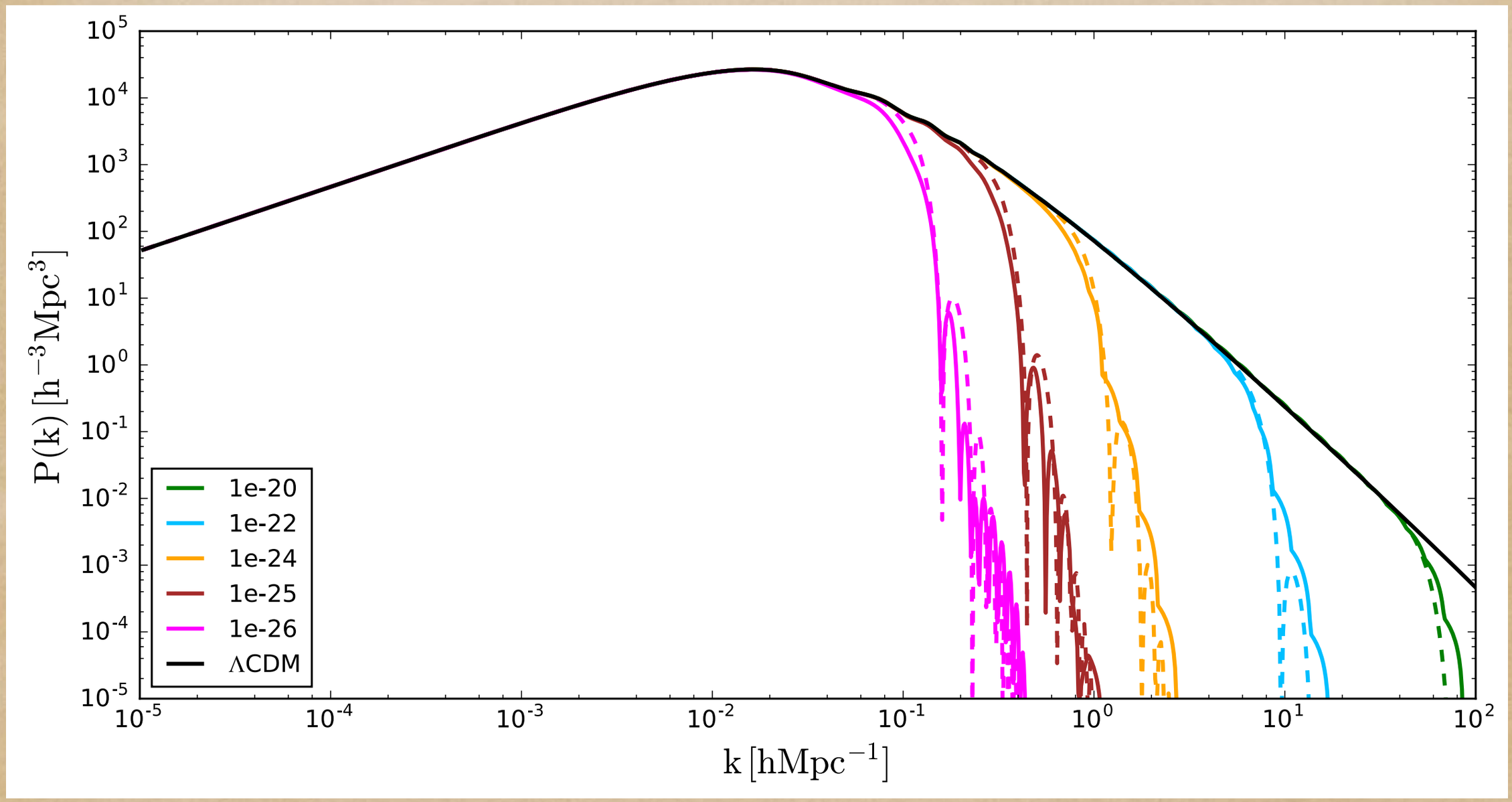
# Implementation in CLASS (equations of motion)

```
if (pba->has_scf == _TRUE_) {

  /** ---> Klein Gordon equation under new formalism */
  /** Auxiliary variables */
  Omega_phi_scf = pvecback[pba->index_bg_Omega_phi_scf];
  theta_phi_scf = pvecback[pba->index_bg_theta_phi_scf];
  y1_phi_scf = pvecback[pba->index_bg_y_phi_scf];
  omega_scf = y[pv->index_pt_omega_scf];
  delta0_scf = y[pv->index_pt_delta0_scf];
  delta1_scf = y[pv->index_pt_delta1_scf];

  /** ---> Equations of motion for omega */
    dy[pv->index_pt_omega_scf] = a_prime_over_a*(1.5*pvecback[pba->index_bg_w_tot]-0.5-y2_phi_scf(pba,
        Omega_phi_scf,theta_phi_scf,y1_phi_scf)*
    exp(0.5*Omega_phi_scf)*sin_scf(pba,0.5*theta_phi_scf)/y1_phi_scf)*y[pv->index_pt_omega_scf];

/** ---> Equations of motion for the density contrasts */
    dy[pv->index_pt_delta0_scf] = -a_prime_over_a*((3.*sin_scf(pba,theta_phi_scf)+omega_scf*(1.-cos_scf(pba
        ,theta_phi_scf)))*delta1_scf
                                                  -omega_scf*sin_scf(pba,theta_phi_scf)*delta0_scf)
                                                  -metric_continuity*(1.-cos_scf(pba,theta_phi_scf)); //
                                                      metric_continuity = h'/2

    dy[pv->index_pt_delta1_scf] = -a_prime_over_a*((3.*cos_scf(pba,theta_phi_scf)+omega_scf*sin_scf(pba,
        theta_phi_scf)
    -exp(0.5*Omega_phi_scf)*sin_scf(pba,0.5*theta_phi_scf)*y2_phi_scf(pba,Omega_phi_scf,theta_phi_scf,
        y1_phi_scf)/y1_phi_scf)*delta1_scf
                                                  -omega_scf*(1.+cos_scf(pba,theta_phi_scf)
                                                  +exp(0.5*Omega_phi_scf)*cos_scf(pba,0.5*theta_phi_scf)*
                                                      y2_phi_scf(pba,Omega_phi_scf,theta_phi_scf,
                                                      y1_phi_scf)/y1_phi_scf)*delta0_scf)
                                                  -metric_continuity*sin_scf(pba,theta_phi_scf); //
                                                      metric_continuity = h'/2
}
```

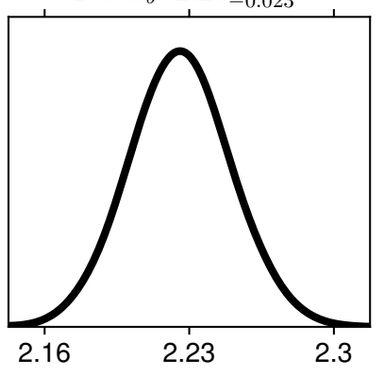
# Numerical results



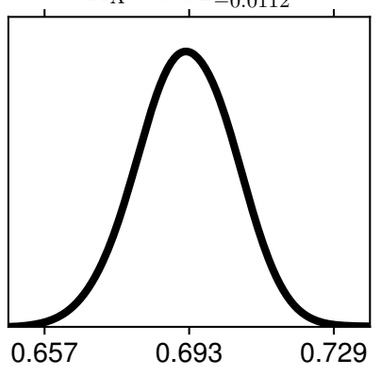
# Numerical results



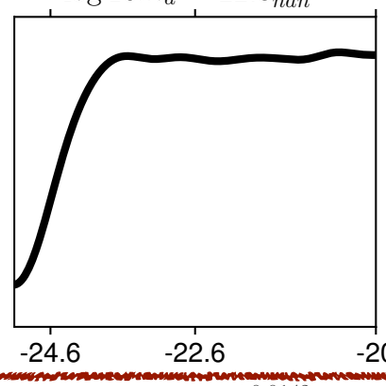
$100\ \omega_b$=$2.23^{+0.0214}_{-0.023}$



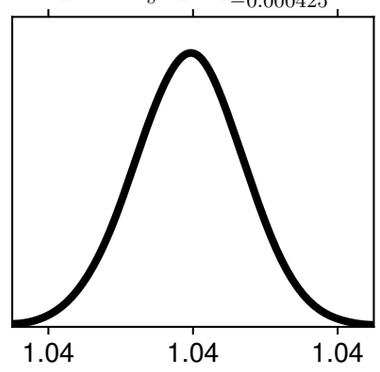

$\Omega_\Lambda$=$0.692^{+0.012}_{-0.0112}$



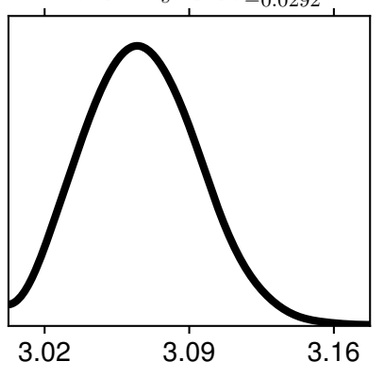

$\log 10m_a$=$-22.3^{+nan}_{nan}$



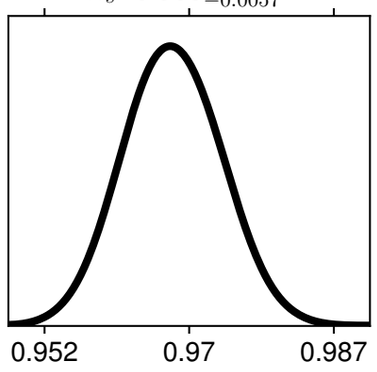

$100 * \theta_s$=$1.04^{+0.000426}_{-0.000425}$



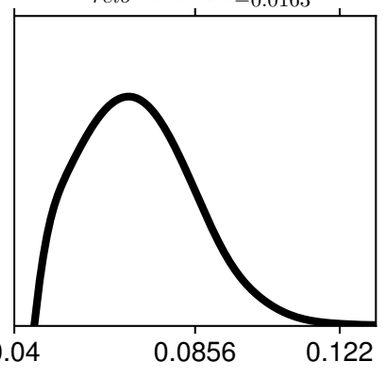

$ln10^{10}A_s$=$3.07^{+0.0254}_{-0.0292}$



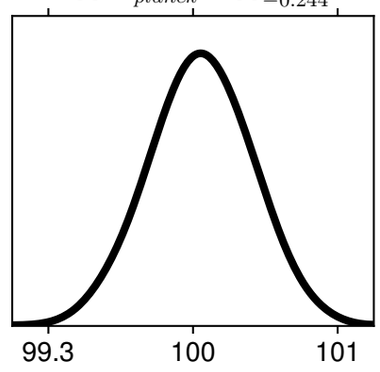

$n_s$=$0.968^{+0.00557}_{-0.0057}$



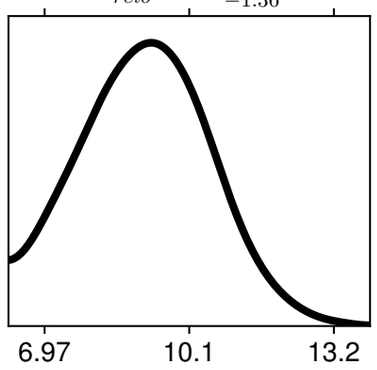

$\tau_{reio}$=$0.0696^{+0.0142}_{-0.0163}$



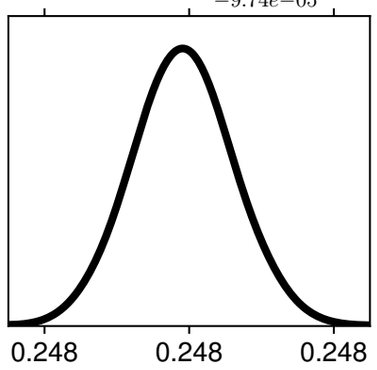

$100\ A_{planck}$=$100^{+0.251}_{-0.244}$



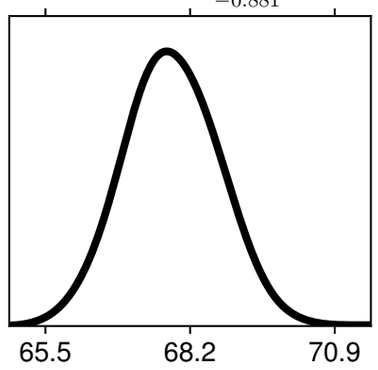

$z_{reio}$=$9.14^{+1.4}_{-1.36}$



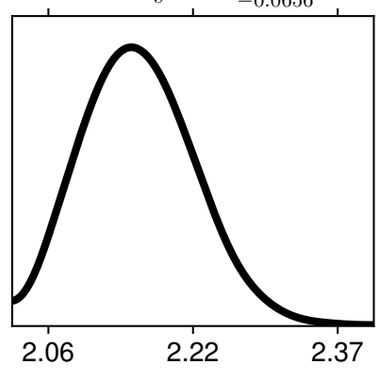

$YHe$=$0.248^{+9.21e-05}_{-9.74e-05}$



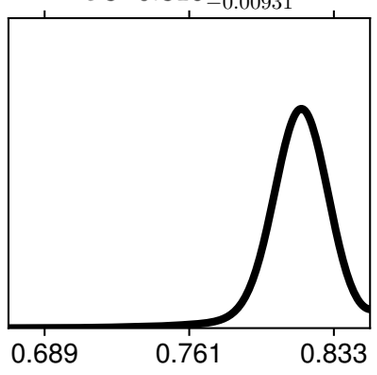

$H0$=$67.9^{+0.857}_{-0.881}$




$10^{+9}A_s$=$2.16^{+0.052}_{-0.0656}$




$\sigma 8$=$0.815^{+0.0129}_{-0.00931}$



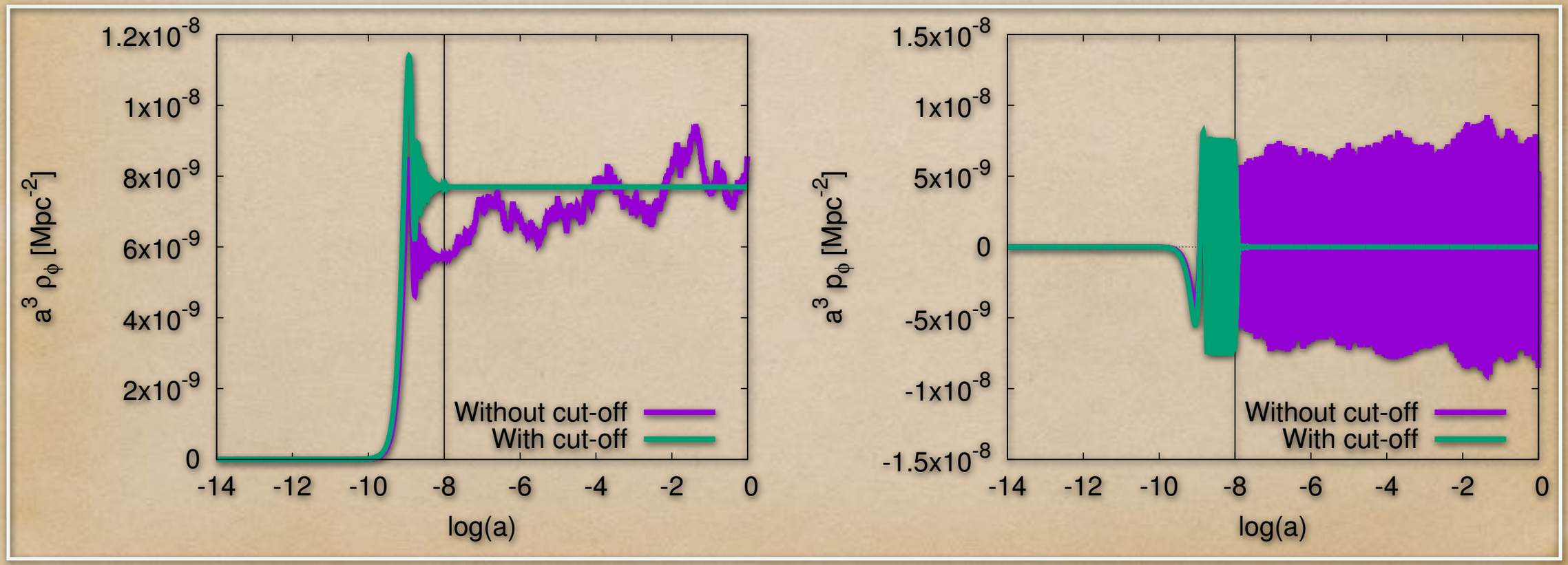
# Rapid oscillations

$$\{\cos_\star \gamma, \sin_\star \gamma\} \equiv (1/2)\left[1 - \tanh(\gamma^2 - \gamma_\star^2)\right]\{\cos\gamma, \sin\gamma\}\,,$$

```c
/** Cosine and sine modified functions to kill oscillations with a very high frequency */
double cos_scf(struct background *pba,
         double theta_phi
         ) {
  double theta_thresh = 1.e2;
    double theta_tol = 1.;//1.e-2;
  return 0.5*(1.-tanh(theta_tol*(theta_phi*theta_phi-theta_thresh*theta_thresh)))*cos(theta_phi);
}

double sin_scf(struct background *pba,
         double theta_phi
         ) {
  double theta_thresh = 1.e2;
    double theta_tol = 1.;//1.e-2;
  return 0.5*(1.-tanh(theta_tol*(theta_phi*theta_phi-theta_thresh*theta_thresh)))*sin(theta_phi);
}
```

# Rapid oscillations

$$\{\cos_\star \gamma, \sin_\star \gamma\} \equiv (1/2)\left[1 - \tanh(\gamma^2 - \gamma_\star^2)\right]\{\cos\gamma, \sin\gamma\}\,,$$



# Rapid oscillations

$$\{\cos_\star \gamma, \sin_\star \gamma\} \equiv (1/2)\left[1 - \tanh(\gamma^2 - \gamma_\star^2)\right]\{\cos\gamma, \sin\gamma\}\,,$$

## Justificación teórica

$$\begin{aligned}\Omega_\phi(t) &= \Omega_{\phi\star}\exp\left[3\int_{N_\star}^{N} w_{tot}\,dx\right]\exp\left[3p\int_{\theta_\star}^{\theta}\frac{\cos x}{x}dx\right]\,,\\ &= \Omega_{\phi\star}\exp\left[3\int_{N_\star}^{N} w_{tot}\,dx\right]\exp\left[3p(\mathrm{Ci}(\theta_\star) - \mathrm{Ci}(\theta))\right]\,,\end{aligned}$$