universe

MDPI

Review

Selected Topics in Numerical Methods for Cosmology

Sandro Dias Pinto Vitenti † and **Mariana Penna-Lima** *,†

Instituto de Física, Universidade de Brasília, Caixa Postal 04455, Brasília DF 70919-970, Brazil

* Correspondence: pennalima@unb.br

† These authors contributed equally to this work.

Received: 31 July 2019; Accepted: 26 August 2019; Published: 29 August 2019

Abstract: The large amount of cosmological data already available (and in the near future) makes the development of efficient numerical codes necessary. Many software products have been implemented to perform cosmological analyses considering one or few probes. The need of multi-task software is rapidly increasing, in order to combine numerous cosmological probes along with their specificity (e.g., astrophysical descriptions and systematic errors). In this work, we mention some of these libraries, bringing out some challenges they will face in the few-percent error era (on the cosmological parameters). We review some concepts of the standard cosmological model, and examine some specific topics on their implementation, bringing, for example, the discussion on how some quantities are numerically defined in different codes. We also consider implementation differences between public codes, mentioning their advantages/disadvantages.

Keywords: numerical methods; cosmology; cancellation errors

1. Introduction

Cosmology is dedicated to studying the evolution of the universe from its primordial epoch until today. The description and modeling of its different stages and observables are, in general, divided into two classes: deterministic and stochastic predictions. The standard model of cosmology is described by the General Relativity (GR) theory of gravitation with a homogeneous and isotropic background metric, which provides deterministic predictions for some cosmological observables. On top of this background, an inhomogeneous description is laid, usually through a perturbation theory, for which the predictions are of stochastic nature. For instance, the distances are among the cosmological/astrophysical observables related only with the background. In contrast, the perturbations describe all the deviations from a homogeneous and isotropic universe, for this reason, were it deterministic, it would have to describe all astrophysical object positions and properties in a given scale. Such deterministic description for perturbations is unfeasible and consequently a probabilistic approach is chosen. This means that, instead of describing the exact positions and properties of each astrophysical object in a given scale, we model the probability distribution of these quantities. For example, when describing the mass distribution in the universe, we use the matter power spectrum P_m, which is a simple way to express the (Gaussian) probability distribution of finding a given matter contrast function $\delta_m(x) \equiv \delta\rho_m(x)/\rho_m$ ($\delta\rho_m(x)$ denotes the perturbation of the matter energy density $\hat{\rho}_m(x)$ around its mean value ρ_m, that is, $\delta\rho_m(x) = \hat{\rho}_m(x) - \rho_m$).

Furthermore, the initial conditions for the very early universe are given in terms of the quantization of its initial perturbations (e.g., in an inflationary scenario—for a review, see [1]). Thus, apart from the problem of decoherence (the evolution from quantum to classical probability distributions, for example see [2]), the initial conditions are already given in terms of stochastic distributions. Nevertheless, observations provide positions and characteristics of actual astrophysical objects as well as the errors in these determinations. Consequently, we have two sources of indeterminacies: the probabilistic aspect of the observation errors and the stochastic nature of our modeling.

In practice, we have two classes of problems in numerical cosmology—the first being related to background observables, such as the distance modulus of type Ia supernovae (SNeIa), cosmic chronometers and H_0 determination. These depend on quantities determined solely by the background model. For this reason, the numerical methods involved are simple and usually require solutions of a small set of Ordinary Differential Equations (ODEs). Once the background is computed, the comparison with data are done by taking into consideration the statistical errors in the observables' determination. Any observable in this class is extremely useful since they depend only on background modeling and, consequently, do not rely on the more sophisticated (but more complex) description of the inhomogeneous universe. The second class is composed of the observables related to the inhomogeneities. For example, Cosmic Microwave Background (CMB) radiation, Large Scale Structure (LSS) observables, such as galaxy spatial correlation, galaxy cluster counts and correlation, and gravitational lensing. For some of these observables, e.g., CMB, the linear perturbation theory is enough (up to a given scale) to compute the observable statistical distribution,[1] whereas, for others, e.g., galaxy spatial correlation, the perturbation theory must be corrected for scales beyond its validity.

In the last two decades, Cosmology has became a data-driven science thanks to the large amount of high-quality observational data that has been released [3–6]. The complexity of the theoretical model of the observables along with their astrophysical features requires sophisticated numerical and statistical tools. In this chapter, we will discuss some of the numerical and statistical methods that have been largely used in Cosmology. Most of these methods can be found in the Numerical Cosmology (`NumCosmo`) library [7], which is a free software written in C on top of the `GObject`[2] and `GObject Introspection`[3] tools.[4] Popular codes such as CAMB (https://camb.info/) [8–10] and CLASS (http://class-code.net/) [11–14] are used to compute both background and perturbation (linear and nonlinear) quantities. Their statistical counterparts, respectively, CosmoMC [15,16] and MontePython [17], implement not only the observable likelihoods but the tools necessary to explore the parameter space of the underlying models mainly through the Markov Chain Monte Carlo (MCMC) method. In Section 2.1, we describe the numerical approaches to solve the background model and the observables related to it. In Section 3, we introduce the tools to solve the linear perturbation problem along with the related observable quantities. Finally, in Section 4, we discuss the most common cancellation errors found in numerical cosmology.

2. Numerical Methods

In the next sections, we will briefly outline the theoretical aspects of the background model and then move to the discussion on how to solve them numerically.

2.1. Background

In this section, we will describe the different ways to compute observables related to the background cosmology; for this reason, we introduce the necessary concepts as we go; for a detailed discussion on these, see, for example, [18,19]. The majority of the cosmological models relies on a

[1] The observed anisotropies in the CMB are very small ($\propto 10^{-5}$), and, for this reason, they are within the regime of validity of the first order perturbation theory.
[2] https://developer.gnome.org/gobject/stable/.
[3] https://developer.gnome.org/gi/stable/.
[4] Briefly speaking, `GObject` provides an object-oriented framework for C programs, while `GObject Introspection` generates bindings for other languages, such as `Python`, `Perl` and `JavaScript`.

metric theory. In particular, we consider the homogeneous and isotropic metric, also known as the Friedmann–Lemaître–Robertson–Walker metric (FLRW),

$$ds^2 = -c^2\,dt^2 + a^2(t)\left[dr^2 + S_K^2(r)(d\theta^2 + \sin^2\theta d\phi^2)\right], \tag{1}$$

where c is the speed of light and

$$S_K(r) = \begin{cases} r & K = 0, \\ \sin\left(r\sqrt{K}\right)/\sqrt{K}, & K > 0, \\ \sinh\left(r\sqrt{|K|}\right)/\sqrt{|K|}, & K < 0, \end{cases} \tag{2}$$

for respectively flat, spherical and hyperbolic spatial hyper-surfaces. This metric has just one function to determine the scale factor $a(t)$ and the Gaussian curvature K. In practice (which is the case for all cited software), we use instead of K, the curvature density parameter at the present day $\Omega_K^0 \equiv -Kc^2/(a_0H_0)^2$, where the super and subscript 0 indicate the quantity today, the Hubble parameter $H_0 = \dot{a}/a|_0$ and $\dot{}$ denotes the derivative with respect to the cosmic time t. Note that theoretical work usually defines $K \in \{-1,\ 0,\ 1\}$ (using a given unit, such as Mpc^{-2}) generating a one-to-one relation between a_0 and Ω_K^0, while, in numerical codes, such as CAMB, CLASS and NumCosmo, it is habitual to let a_0 as an arbitrary value defined by the user and use the definition above to determine K through Ω_K^0.

In practice, we do not measure $a(t)$ directly, but related quantities such as the distances to astronomical objects. Considering a null trajectory of photons emitted by a galaxy traveling along the radial direction to us, we have that

$$r = c\int_{t_e}^{t_0} \frac{\mathrm{d}t}{a(t)}, \tag{3}$$

where t_e and t_0 are the emitted and observed times (here, t_0 represents the present time). Note that r refers to the distance at the comoving hyper-surface, i.e., the hyper-surface where $a(t_c) = 1$ (see Equation (1)). This means that r is the distance between the point of departure and arrival both projected (through the Hubble flow) back to the comoving hyper-surface. In other words, using the coordinates in Equation (1), we define a foliation of the space-time where each sheet is defined by $t =$ constant, then the photon leaves the source at t_e at the point p_e within the t_e hyper-surface and arrives at point p_0 at the hyper-surface t_0. Since p_e and p_0 are defined at different hyper-surfaces, we need to transport them to the same hyper-surface in order to have a meaningful definition of a spatial distance between them. Then, we project both points p_e and p_0 at the comoving hyper-surface by following the Hubble flow that passes through these points back to the hyper-surface t_c $[a(t_c) = 1]$, where r gives the spatial distance between them.

In the literature, $a_0 = 1$ is often defined (for the flat case), that is, the comoving hyper-surface coincides with today's hyper-surface and r would provide the distance between points today. However, as discussed above in the numerical codes, one needs a general definition for the whole parameter space (in this case, for any value of Ω_K). For this reason, we keep a_0 arbitrary and define the comoving distance projected back to today's hyper-surface as

$$d_c(z) \equiv a_0 r = d_H \int_0^z \frac{\mathrm{d}\bar{z}}{E(\bar{z})}, \tag{4}$$

where the Hubble radius is $d_H = c/H_0$, $E(z) = H(z)/H_0$ is the normalized Hubble function, and the distance is now written in terms of the observable z that is the redshift,

$$1 + z \equiv \frac{a_0}{a(t)}. \tag{5}$$

Finally, similar to the comoving distance, we introduce the comoving time

$$\eta = c\int_0^t \mathrm{d}\bar{t}\frac{1}{a(\bar{t})}, \tag{6}$$

where we chose the conformal time such that $\eta(t=0)=0$. The only differences between the comoving and the conformal distances are a factor of a_0 and the integration limits.

2.2. Time Variables

When solving the background, it is usually useful to choose which variable to use as time. This is particularly important if the evolution of the perturbations will also be computed, since it requires the evaluation of several background observables for each time step in the perturbations' integration.

We first note that there is a one-to-one relation between z, a and d_c (both a_0 and t_0 are fixed quantities).[5] The relation between these variables and the cosmic time requires the choice of a reference hypersurface to anchor the time definition. For instance, it is common to choose $t=0$ to mark the singularity of the classical FLRW metric, i.e., $a(0)=0$. This choice has some drawbacks. In order to compute $t(z)$ or $\eta(z)$, for example, we need to evaluate

$$t = \int_z^\infty \mathrm{d}\bar{z}\frac{1}{(1+\bar{z})H(\bar{z})} \quad \text{and} \quad \eta = \frac{1}{a_0}\int_z^\infty \mathrm{d}\bar{z}\frac{1}{H(\bar{z})}. \tag{7}$$

However, the above integrals include the whole history of the background metric, from $t=0$ ($z=\infty$) to z. This means that, in practice, we need to know the values of $H(z)$ for this whole range, including the matter-dominated phase, radiation phase, inflationary or bounce phase, etc. The available numerical codes include the computation of $t(z)$ and/or $\eta(z)$. However, one should remark that these quantities are obtained from integrals as above and extrapolating the radiation phase till the singularity, i.e., they ignore any inflationary or bounce phases and assume a radiation dominated singularity in the past.

Both NumCosmo and CAMB compute the times t and η integrating the Hubble function $H(z)$ and extrapolating the radiation era until the singularity. CLASS uses the approximation of $\eta(z)\approx(1+z)/(a_0H)=1/(aH)$ which result from assuming $H(z)\propto(1+z)^2$ for z in the radiation era. This means that there will probably be small shifts between the η variable when comparing different codes; furthermore, the choice of precision (and technique) used in the integration may also result in small differences.

Another option for time variable, used by different objects in NumCosmo, is the scale factor a or its logarithm $\alpha \equiv \pm\ln(a)$.[6] Of course, such variable cannot be used when the Hubble function H change signs (for example, in a bounce). However, this variable is useful when describing the expansion phase since it is possible to obtain an analytic expression for $H(a)$ in different scenarios. In these cases, we do not need to solve the integrals (7) to relate H and the time variable.

---

5 The relation between z and d_c is monotonic since $E>0$.

6 The $\pm$ sign in the definition of α is included in order to have it monotonically increasing with t or η in the expansion/contraction, respectively.

2.3. Distances

The base of several cosmological observers are the cosmological distances. Besides the comoving distance (Equation (4)), other useful distances are: the transverse comoving $d_M(z)$, the angular diameter $d_A(z)$ and luminosity distances $d_L(z)$, respectively defined as [20]:

$$d_M = \begin{cases} d_H \frac{1}{\sqrt{\Omega_K}} \sinh\left(\sqrt{\Omega_K} d_c/d_H\right) & \text{for} \quad \Omega_K > 0 \\ d_c & \text{for} \quad \Omega_K = 0 \\ d_H \frac{1}{\sqrt{|\Omega_K|}} \sin\left(\sqrt{|\Omega_K|} d_c/d_H\right) & \text{for} \quad \Omega_K < 0, \end{cases} \tag{8}$$

$$d_A(z) = \frac{d_M(z)}{(1+z)} \quad \text{and} \quad d_L(z) = (1+z) d_M(z). \tag{9}$$

In turn, the Hubble function $H(z)$ can be recovered using observational data, which allows for the model-independent approaches (see [21] and references therein) besides the usual modeling assuming a theory of gravitation, i.e., independent from the equations of motion discussed in the last section.

Figure 1 shows the comoving distance for different values of the Hubble parameter H_0, the matter density parameter Ω_m and the dark energy (DE) equation of state w_{DE} = constant. Note that these quantities are sensitive to the cosmological parameters and, therefore, observables related to the distance measurements are important to discriminate among the possible values of these parameters. Three of these observables are the distance modulus of type Ia supernovae (SNe Ia) [22,23], the baryon acoustic oscillations (BAO) [24] and the so-called cosmic chronometers.

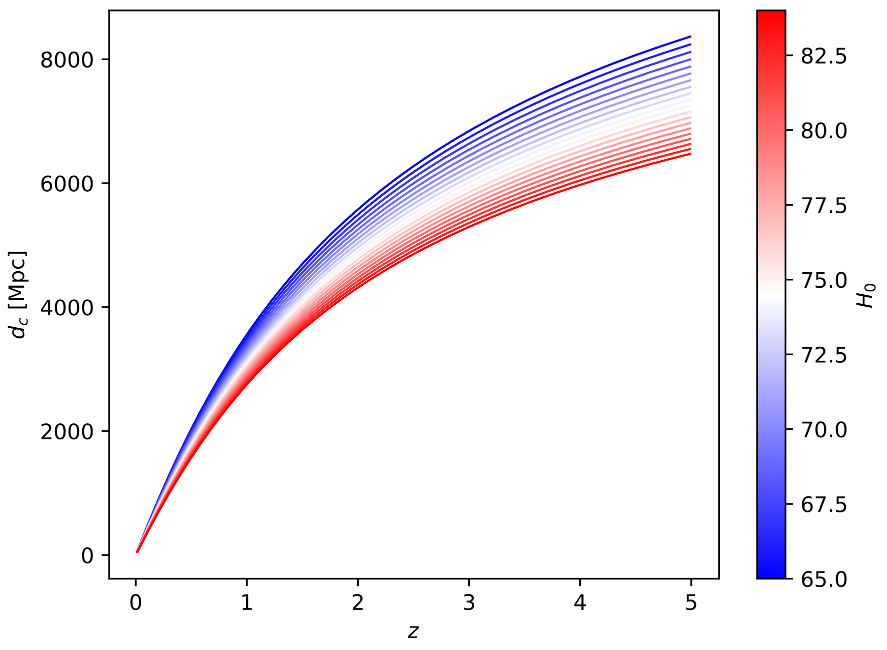
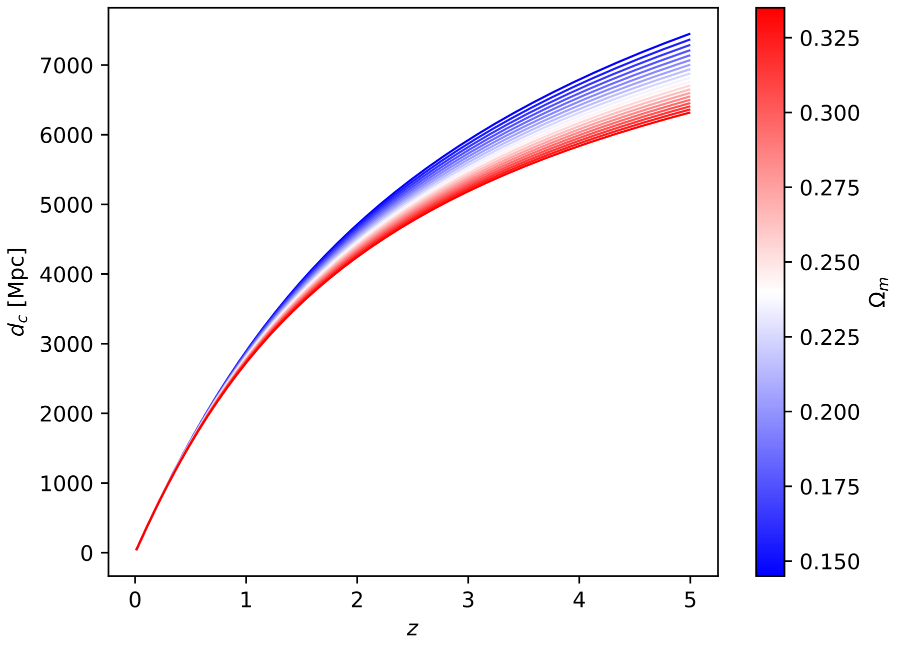


Figure 1. *Cont.*



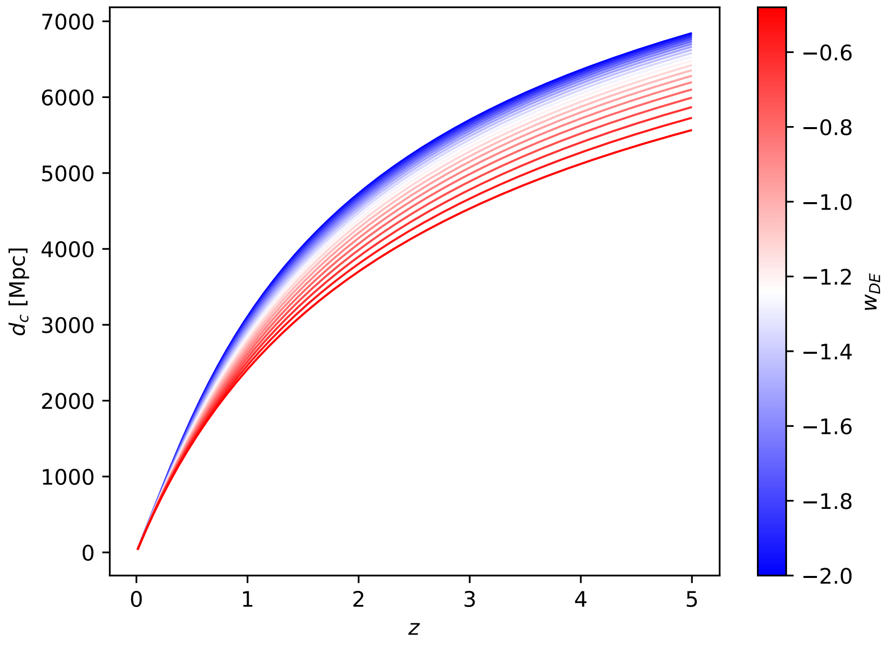


Figure 1. Comoving distance for different values of the Hubble constant H_0 (**upper** panel), matter density parameter Ω_m (**medium**) and the DE equation of state w_{DE} (**lower**).

2.4. Equations of Motion

Up to this point, we discussed the kinetic aspects of our choice of metric. Nevertheless, we need a way to compute the time evolution of $a(t)$ in order to calculate the quantities above. In this section, we will discuss the equations of motion in the context of GR.

The standard cosmological model assumes GR and the FLRW on large-scales and thus the equations of motion are given by the so-called Friedmann equations,

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{8\pi G\rho}{3c^2} - \frac{Kc^2}{a^2}, \tag{10}$$

$$\frac{\ddot{a}}{a} = -\frac{4\pi G}{3c^2}\left(\rho + 3p\right), \tag{11}$$

where G is the gravitational constant, ρ and p are the energy density and pressure of the matter-energy content of the universe, which is composed of radiation (photons and relativistic neutrinos), matter (Dark Matter (DM), baryons and neutrinos) and DE. Their equation of state, $p = w\rho$, are

$$\text{Radiation} : w_r = \frac{1}{3}, \tag{12}$$

$$\text{Matter} : w_m = 0, \tag{13}$$

$$\text{Dark Energy} : w_{DE} \leqslant -\frac{1}{3}. \tag{14}$$

The DE component of the standard model is, in general, described by the cosmological constant Λ, in which $w_{DE} = -1$.

It is clear from the Friedmann equations that we need additional equations of motion to close the system, i.e., we need equations of motion that determine ρ_i and p_i (or w_i), where i represents the possible components. If the matter component is minimally coupled with the theory of gravitation and its energy-momentum tensor is "conserved" (satisfy $\nabla_\mu T_i^{\mu\nu} = 0$), the imposition that this tensor is compatible with a Friedmann metric results in the following equation of motion for ρ_i:

$$\dot{\rho}_i + 3H(\rho_i + p_i) = 0. \tag{15}$$

There are two important lessons to get from the above equation. Firstly, Equation (15) is not enough to close the system of equations and, therefore, we still need a way to determine p_i. This occurs due to the fact that Equation (10) is a constraint equation (see Section 2.5 for more details) and, consequently, Equation (11) can be derived from Equations (10) and (15). One common example to solve the system and obtain $\rho_i(t)$ is to consider simple components, as discussed above, where w_i is a constant. We have different options for more complicated fluids. For instance, for a scalar field, ρ_i and p_i are functions of the field variable and the equation of motion of the scalar field can be used to close the system (in this case, Equation (15) is redundant). For a statistical description, one can use the background compatible Boltzmann equation to describe the phase-space distribution $f_i(x^\mu, p^\mu)$ and obtain ρ_i and p_i from it (which is the case of the equations of state Equations (12)–(14)). Finally, one can model p_i (or w_i) directly using a phenomenological approach.

The second lesson refers to the choice of the time variable. That is, if we rewrite Equation (15) using a as a time variable, we obtain

$$\frac{\partial\rho_i}{\partial a} + \frac{3}{a}(\rho_i + p_i) = 0. \tag{16}$$

This simple manipulation has an important consequence. The factor H that couples the original equation with all other components was eliminated. This means that, for the case where $p(\rho)$ is a known function, this equation can be solved without knowing any of the other components present in the system. This is one of the reasons why this kind of modeling ($p_i(\rho_i)$, $w_i(a)$, $\dots$) is so popular. It allows for obtaining a solution for ρ_i independently of the other matter components and/or the gravitational theory.

Note that the density is a function of the time, $\rho = \rho(t) = \rho_r(t) + \rho_m(t) + \rho_{DE}(t)$, and it is not feasible to use observational data in order to recover $\rho(t)$ as given in the present form. Therefore, similarly to the curvature parameter K (see Section 2.1), we rewrite Equation (10) in terms of the redshift z and define the density parameters as

$$\Omega_i = \frac{\rho_i(z\ = 0)}{\rho_{\rm crit,0}}, \quad \rho_{\rm crit,0} = \frac{3c^2H_0^2}{8\pi G}, \tag{17}$$

where i stands for the radiation (r), matter (m) and DE components.[7] Thus, Equation (10) now reads

$$\begin{aligned} E^2(z) \equiv & \left(\frac{H(z)}{H_0}\right)^2 \\ & = \Omega_r(1+z)^4 + \Omega_m(1+z)^3 + \Omega_K(1+z)^2 \\ & + \Omega_{DE} \exp\left\{3\int_0^z \frac{[1+w_{DE}(z')]}{1+z'}\mathrm{d}z'\right\}. \end{aligned} \tag{18}$$

Computing this equation at $z = 0$, we obtain the constraint

$$\Omega_r + \Omega_m + \Omega_{DE} + \Omega_K = 1, \tag{19}$$

which reduces by one the size of the parameter space to study the recent expansion evolution of the universe. In short, in the context of GR and homogeneous and isotropic metric, one needs to estimate the following set of cosmological parameters: $\{H_0, \Omega_r, \Omega_m, \Omega_{DE}, w(z)\}$, where Ω_K is then determined by Equation (19). Assuming a flat universe, $\Omega_K = 0$, which is largely considered in the literature given the CMB constraints [3,5], we end up with the restriction $\Omega_{DE} = 1 - \Omega_r - \Omega_m$.

2.5. Integrating and Evaluating the Background

In the last sections, we described the observables related to the background model. In order to compute them, we need to integrate this model and save the results in a way that they can be used in statistical calculations. Both tasks are crucial in numerical cosmology since the evolution of the perturbations relies on several evaluations of the background model at different times and, in a statistical analysis, these quantities need to be recomputed at different points of the parameter space. For these reasons, the computation and evaluation times are bottlenecks for this kind of analysis.

2.5.1. Integration Strategies

There are different strategies to integrate the background equations of motion. First of all, we need to choose how to integrate the gravitational part. The Friedmann equations comprehend a dynamical Equation (11) and a constraint Equation (10). This means that, in principle, we need to solve the dynamical equation and impose the constraint equation in our initial conditions. In other words, to solve Equation (11), we need the initial conditions for both a and $\dot{a}$, but this choice must satisfy Equation (10). It is worth noting that, since the choice of an initial value for a is arbitrary, Equation (11) ends up determining the initial conditions completely. Therefore, in this context, there are no dynamical degrees of freedom left for the gravitational sector, i.e., the imposition of homogeneous and isotropic hypersurfaces removes the dynamical degrees of freedom.

In practice, we want to solve the Friedmann equations in the expansion branch ($H > 0$). For this reason, we know a priori the sign of H. Therefore, we can take the square root of Equation (10),

$$\frac{\dot{a}}{a} = \sqrt{\frac{8\pi G\rho}{3c^2} - \frac{kc^2}{a^2}}, \tag{20}$$

[7] Note that here we are defining Ω_i as a constant evaluated today, in the literature, the definition as a function of z is also frequently found, i.e., $\Omega_i(z) \equiv \rho_i(z)/\rho_{\rm crit}(z)$, where $\rho_{\rm crit}(z) = 3c^2H^2(z)/(8\pi G)$.

and solve it instead of solving Equation (11). We can even go further, if all matter components decouples from gravity, as we discussed in Section 2.4, then we have an analytical solution for H given by Equation (18) if we choose z (or a) as our time variable.

It is useful to review the steps that allowed an analytical solution for H—first, all matter components decoupled from the system as we argued in Equation (16). For this reason, we were able to find solutions for ρ_i as functions of a (or any other time algebraically related to a). Second, we used the first order Friedmann Equation (10) to determine H^2. Finally, we assumed that the model will always be in the expansion branch $H > 0$ allowing then the determination of H as a function of a. If any of these conditions were missing, i.e., one component not decoupling or the indetermination of H sign, the solution would involve the whole system.

In the general purpose code, like CLASS or CAMB, one cannot assume that all matter components decouples from H. This happens because they include the possibility of modeling the dark energy as a scalar field and the scalar field equation of motion does not decouple from H. For instance, a scalar field φ with a canonical kinetic term and arbitrary potential $V(\varphi)$ satisfies

$$\ddot{\varphi} + 3H\dot{\varphi} + \frac{\partial V}{\partial \varphi} = 0. \tag{21}$$

In particular, CLASS chooses the conformal time in units of mega-parsec (Mpc) to integrate all quantities. We summarize the strategy implemented in CLASS in the following. They first analytically integrate every component that decouples from the system, e.g., the cold dark matter and baryon fluids, and implement the energy density and pressure related to these quantities as functions of a. Moreover, they implement the energy density and pressure of more complicated components in terms of their internal degrees of freedom. For instance, a scalar field from the above example has

$$\rho_\varphi = \frac{\dot{\varphi}^2}{2} + V, \quad p_\varphi = \frac{\dot{\varphi}^2}{2} - V,$$

where the internal degrees of freedom are φ and $\dot{\varphi}$. They use label 'A' to refer to the first set, the functions of a and the internal degrees of freedom ($\rho_\varphi(\varphi, \dot{\varphi})$ and $p_\varphi(\varphi, \dot{\varphi})$ in the last example), and label 'B' for all internal degrees of freedom ($a(t)$, $\varphi(t)$, $\dot{\varphi}(t)$ in the last example). All other functionals of these quantities, such as the distances defined in Section 2.3, are labeled as 'C'. Finally, they implement the equations of motion for the 'B+C' set and integrate all variables with respect to the chosen time variable η.

We have two options to perform the 'B + C' integration: (i) integrate all at once or (ii) to first integrate 'B' and then use the respective results to integrate 'C'. Some possible advantages of integrating all variables 'B + C' together are:

1. simplicity, one needs to integrate a system of variables with respect to time just once;
2. less integration overhead, the integration software itself has a computational cost. Each time it is used, there are the costs of initialization, step computation and destruction.

On the other hand, the possible shortcomings are:

1. step sizes smaller than necessary. When performing the integration, the Ordinary Differential Equations (ODE) solver evaluates the steps of all components being integrated and adapts the time step such that the error bounds are respected by all components. For this reason, including the 'C' set in the integration can result in a larger set of steps for all quantities 'B + C';
2. difficult to efficiently modularize, in some analysis, just a few (or just one) observables from 'C' are necessary. When performing all integration at once, you have two options, integrate everything every time, even when they are not all necessary, or to create a set of flags that control which quantities should be integrated by branching (e.g., if-statements). Naturally, the if-clauses create an unnecessary overhead if they are placed inside the integration loop. Alternatively, if one

decides to create a loop for every combination to avoid this overhead, then he/she will end-up with 2^n different loops (where n is the number of variables in 'C') which creates new problems such as code repetition and harder code maintenance.

In CLASS, they integrate first 'B' and then use the results to compute 'C'. However, they do not integrate the variables in 'C'. They just get the resulting knots from 'B' integration and compute at the same knots the values of 'C' , and then add to a large background table enclosing every variable in 'A + B + C'. This table is then used to compute the background variables at any time using a spline interpolation. This means that no error control was used to compute the 'A' and 'C' variables, even though it was used to compute 'B'.

In NumCosmo, they also integrate 'B' first, but the 'C' set is handled differently. First, all variables that are algebraically related to each other are identified. For example, the distances discussed in Section 2.3 can be computed from the comoving distance without any additional integration. Then, a minimal set of variables in 'C' is identified and, for each one, a different object is built. For instance, the ones related to distances are included in the NcDistance object. This object, when used, integrates the comoving distance using the results from 'B' present in the basic background model described by a NcHICosmo object. The integration is done requiring the same error bounds as in 'B' and a different spline is created for the comoving distance, with different time intervals.

At this point, the main differences between CLASS and NumCosmo are that the first does not integrate 'C', it simply interpolates them based on a fixed grid choice, and does not have a modular structure for the computations of 'C'. Nevertheless, the non-modular design choice is understandable. When CLASS was first conceptualized, it intended to be a Boltzmann solver; thus, it is natural to always integrate all quantities in 'C' that are needed to compute the time evolution of the perturbations. However, now, CLASS is slowly migrating to a general purpose code as the cosmological basis for different numerical experiments usually performed by MontePython [17]. At the same time, NumCosmo was designed from the ground-up to be a modular general purpose library to handle different cosmological computations.

2.5.2. Evaluating the Background

As we previously described, the integration output is usually saved in memory such that it can be used later through interpolation. In principle, it would be also possible to just integrate everything necessary at once. This can work for a simple code, e.g., if we just need the conformal distance at some predefined set of redshifts. However, in many other cases, this would lead to a very complicated code. For example, when integrating perturbations, we need to integrate it for different values of the mode k. This means that we would have to integrate the background and all modes k at the same time. Not only would that be complex (a multi-purpose code written like this would require a huge number of branches to accommodate the different code usage), but sometimes one does not know a priori which modes k need to be integrated.

For this reason, a good interpolation method is a central part of any numerical cosmology code. The most common approach is the use of splines, which avoids the Runge phenomenon for interpolation with several knots. A spline is defined as a piecewise polynomial interpolation where each interval is described by a polynomial of order k and the whole function is required to be C^{k-1}, i.e., a function with $k-1$ continuous derivative, see [25] for a detailed description of spline and its characteristics. It is clear that such piecewise function has $(n\ -1)(k\ +1)$ degrees of freedom where n is the number of knots, imposing continuity up to the $k-1$ derivative gives us $(n-2)k$ constraints (remember that the continuity is imposed only on the internal knots); therefore, there are $(n\ -1)(k\ +1)-(n-2)k=n+k-1$ degrees left to determine. In practice, we interpolate functions that we know its values at the n knots, still leaving us with $k-1$ degrees of freedom to determine.

The simplest choice is the linear spline ($k=1$); in this case, there are no extra degrees of freedom to determine; nonetheless, the resulting function is not very smooth; it is actually only continuous (C^0), and the interpolation error is proportional to $h^1 \max|f'(x)|$, where h is the largest distance between two adjacent knots and $f(x)$ is the function being interpolated and $'$ the derivative with respect to x. Hence, a linear spline is appropriated only for really small h (large number of knots) or for functions with small first derivatives. When we move to larger k, we end up with the problem of choosing k and then determining the extra $k-1$ degrees of freedom. The first problem is solved in a geometrical manner, the cubic splines $k=3$ are the ones that minimize $\int_a^b \mathrm{d}x\,[f''(x)]^2$, where a and b are the endpoints of the complete interval being interpolated. This means that the cubic interpolation produces functions with small curvature that still matches $f(x)$ at the knots. This is a reasonable requirement; we are usually interested in interpolating functions that does not wiggle strongly between knots.

Choosing the cubic spline, we then need to fix the remaining two degrees of freedom. This is usually done by imposing boundary conditions on the interpolation function $p(x)$. For example, one can impose a value for $p'(a) = f'(a)$ and $p'(b) = f'(b)$, which requires the knowledge of the derivative of $f(x)$ at the interval extremities. Since this information is usually not available, other approaches are necessary. The so-called natural splines impose that the second derivatives of the interpolating function be zero at a and b, i.e., $p''(a) = p''(b) = 0$, this algorithm can be found at the GNU Scientific Library (GSL) [26]. Nevertheless, the imposition of an arbitrary value for the second derivative results in a global interpolation error proportional to h^2, instead of the original h^4. Another approach is to estimate the derivative at the boundaries and use it to fix $p'(a)$ and $p'(b)$; this is the approach followed by CLASS (at least up to the version 2.6); however, here we need to find a procedure that will result in a $\mathcal{O}(h^4)$ error bound. Currently, CLASS uses the forward/backward three-point difference method which has an error bound of only $\mathcal{O}(h^3)$, which spoils the global $\mathcal{O}(h^4)$ error bound of the spline interpolation. To keep this error bound, a higher order approximation for the derivatives is necessary [25,27]. Finally, this can also be solved by imposing an additional condition on the interpolating function; in the not-a-knot procedure, we impose that the third derivative is also continuous at the second and one before last knots. This condition maintains the global $\mathcal{O}(h^4)$ error bound and can be easily integrated in the tridiagonal system used to determine the spline coefficients, see [25] and ([7], ncm_spline_cubic_notaknot.c) for an implementation.

It is also worth mentioning that there are other similar choices of interpolation. For example, in the Akima interpolation ([25], pg. 42), one estimates the first derivative at each knot using a simple two-points forward/backward finite difference method and then use them and the function values to determine a cubic polynomial at each interval. The resulting interpolation function is not a spline since it is only C^1 and the interpolation error $\mathcal{O}(h^2)$ is worse than a not-a-knot spline; nevertheless, it is a local algorithm since it depends only on the knots and their nearest neighbors and also simpler and faster. On the other hand, the difference in speed for a spline algorithm is usually irrelevant. Notably, the LSST-DESC Core Cosmology Library (CCL) [28] is using the Akima interpolation as its default method (up to version v1.0.0).

When using interpolation through piecewise functions, we have an additional computational cost when evaluating the function. Given an evaluation point x, we need to determine to which interval this point belongs to. This is usually accomplished performing a binary search (see [29], for example), which is in the worst case $\mathcal{O}(\log n)$, where n is the number of knots. Some libraries, GSL for example, also provide an accelerated spline, i.e., in a nutshell, it saves the interval of the last evaluation and tries it first in the next one. The rational here is that one usually evaluates the spline in an ascending/descending in small steps (for instance, when integrating). However, this has some disadvantages. First, if the evaluation is not done in an ascending/descending order, it becomes useless. Since it saves the last step in memory and modifies it in every step, it cannot be used in a multi-threaded environment in a simple way.[8] Finally, the determination of the interval usually contributes very little in the computation time, thus, in general, it is safer and simpler to not use this kind of optimization.

[8] It would be necessary to save/read the last evaluation interval in a per-thread basis.

The last point about the evaluation of piecewise functions is the determination of the knots. In most codes, this is done through some heuristic algorithm; in most cases, the programmer uses the fixed end-points a and b and simply chooses the knots with linear/logarithm spacing, namely,

$$x_i = a + \frac{(b-a)}{n-1} i, \qquad x_i = \exp\left[\ln a + \frac{(\ln b - \ln a)}{n-1} i\right],$$

where i ranges from 0 to $n-1$ and n is the number of knots (sometimes, a combination of these two is used). This approach has several pitfalls; first, the relation between n and the final interpolation error is not clear. Hence, the final user has to vary the value of n until he/she gets the desired precision. In a complex code, there are sometimes a very large number of splines and, consequently, the user has to play with a large set of control variables (for example, n_j where j labels the different splines in the code, the point where we can change from a linear to a logarithm scale, etc.) to attain a certain precision. Actually, the problem is even worse, and the user can choose all control parameters, but, in practice, one does that for a given model with one fixed set of parameters (in the best case scenario, a large number of parameter sets are used). In a statistical analysis, the model is evaluated at different points of the parameter space and nothing guarantees that the precision at these points will remain the same as in the points where it was calibrated. This problem is magnified in large parameter spaces considering that, in this case, it is harder to check for precision in the whole parameter space.

We will close this section discussing two different methods to determine the knots in a way that is adaptive to the desired precision. However, this is usually algorithm-specific and needs to be dealt with case by case. First, consider the case where the function to be interpolated is the result of an integral or a solution of an ODE system. When it comes from an integral, we have

$$F(x) = \int_a^b \mathrm{d}x f(x). \tag{22}$$

It is easy to transform this problem in a one-dimensional ODE, i.e.,

$$F'(x) = f(x), \qquad F(a) = 0. \tag{23}$$

Now, the integral or the ODE system can be solved with the many available ODE integrators (see, for example, the excellent library SUNDIALS [30]). The ODE solvers adapt the step automatically in order to obtain the desired precision; moreover the step procedure is based on a polynomial interpolation. For these reasons, we can use the same steps to define the knots to be used in the interpolation. In other words, the ODE solver computes the steps necessary to integrate a function such that, inside these steps, the function is well approximated by a polynomial (given the required tolerance); therefore, it is natural to use the same steps to interpolate the function afterwards.

The second method is applicable when we have a function $f(x)$ with a high computational cost. When this function needs to be evaluated several times, it is more efficient to build a spline approximation first and then evaluate the spline. As a rule of thumb, this is useful if the function $f(x)$ will be evaluated more than n times, where n is the number of knots you need to create the spline interpolation. The method consists of comparing the value of the function $f(x)$ and its spline approximation $p(x)$ inside of each interval of $p(x)$, i.e., if $p(x)$ is defined with n knots, we compute the difference

$$e_i = |p(x_{i+1/2}) - f(x_{i+1/2})|,$$

for each $n-1$ intervals, where $x_{i+1/2} = (x_i + x_{i+1})/2$. For each interval where e_i does not satisfy the tolerance requirements ($e_i < f(x_{i+1/2})r + a$, where r is the relative tolerance and a the absolute tolerance), we update the spline adding this new point (creating two new intervals) and mark them as not OK (**NOK**). The intervals where the tolerance is satisfied are marked as **OK**. After the first iteration throughout all knots repeat the same procedure for all intervals marked **NOK**, then repeat until all intervals are marked **OK**. An implementation of this algorithm can be found at ([7], ncm_spline_func.c). Some variants of this algorithm are also useful, for instance, instead of using the midpoint $(x_i + x_{i+1})/2$ one can also use the log-midpoint $\exp\left[(\ln x_i + \ln x_{i+1})/2\right]$ for positive definite x_i.[9]

We presented above two methods to control errors when computing interpolation functions. We argue that this kind of error control based on a single tolerance parameter is essential for the numerical tools aimed for cosmology and astrophysics.[10] The sheer amount of different computations needed to evaluate the cosmological and astrophysical observables require complex codes that handle many different numerical techniques. For this reason, well designed local error controls are necessary to have a final error control based on the tolerance required by the observable. This is also crucial when evaluating non-standard models; in these cases, the codes tend to be less checked and tested, and, after all, they are usually only used by the group that developed it. This same problem is also present in standard models but in regions of the parametric space far from the current expectations. These regions tend to be much less tested and are sometimes excluded from the allowed parameters' range.

3. Linear Perturbations

On top of the background standard model discussed in Section 2.1, we have the perturbation theory with which we describe the evolution of Universe from the initial primordial fluctuations to the structure formation and their imprints on the observables we measure. In this section, we will have a lightning review of perturbation theory in cosmology (a full description can be found in [19]) and then we explore a single component scalar perturbation equation in order to exemplify the common numerical challenges involved in this context.

First, we need to extend our assumptions about the space-time geometry. Let us call the background metric components in Equation (1) by $g^{(0)}_{\mu\nu}$. Now, we assume that the metric describing our space-time is given by

$$g_{\mu\nu} = g^{(0)}_{\mu\nu} + h_{\mu\nu}, \tag{24}$$

where

$$\begin{aligned} h_{00} &= 2\phi, \\ h_{0i} &= -aD_iB, \\ h_{ij} &= 2a^2(\psi\gamma_{ij} - D_iD_jE), \end{aligned} \tag{25}$$

where D_i is the covariant derivative with respect to the spatial projection of the background metric. Here, we are assuming that the metric is described by a Friedmann metric plus small deviations, all degrees of freedom but the scalars are ignored, leaving us with the fields $(\phi,\ \psi,\ B,\ E)$. The physical idea is that a Friedmann metric is a good approximation and all deviations from it can be described by a small perturbation. There are several important theoretical aspects of this description that we are not going to discuss here, ranging from gauge dependency [31–33], size validity of the approximation [34,35] and its theory of initial conditions [19]. Instead, we focus only on the numerical methods to solve the perturbation equations of motion and evaluating the result.

---

9 When x_i are not positive definite one can use a similar algorithm with $\ln x_i \to \operatorname{arctanh}(x_i/s)$ and $\exp \to \tanh$, where the is a scale s controls the transition between linear and log scale.

10 Usually, the relative tolerance controls the final error of the code while the absolute tolerance is application specific and, in the few cases where it is used, it serves to avoid excess in the tolerance. For example, if a given function f is zero at a point x and we want to compute its approximation p, the error control is

$$|f(x) - p(x)| < rf(x) + a \to |p(x)| < a.$$

Thus, without the absolute tolerance, the error control would never accept the approximation, unless it was a perfect approximation $p(x) = 0$.

In order to be compatible with our choice of metric, we require that the energy momentum tensor must also be split into a background plus perturbations, following a similar decomposition that we made for the metric. This produces the following scalar degrees of freedom ($\delta\rho$, δp, V, Π), where $\delta\rho$ and δp are the perturbations of the total energy density and pressure, respectively, V the velocity potential and Π the anisotropic stress.

Now, we have a much more complicated problem than the background. There are eight degrees of freedom to determine among the metric and energy momentum tensor perturbations. This problem can be simplified by writing the equations of motion in terms of geometric quantities, i.e.,

$$D_i a^i = -D^2\phi, \qquad D_iD_j\sigma^{ij} = \frac{2}{3}D^2D_K^2\sigma, \qquad \sigma \equiv B - \dot{E} + 2HE,$$
$$\delta H = \dot{\psi} + H\phi + \frac{D^2\sigma}{3}, \qquad \delta\mathcal{R} = -4D_K^2\psi.$$

Here, all variables are defined with respect to the background foliation, D^2 is the spatial Laplacian and $D_K^2 \equiv D^2 + 3K/a^2$ (where K is the spatial curvature of the background metric (1)), a^i represent the acceleration of the Hubble flow, σ the shear potential for the Hubble flow lines, δH the perturbation on the Hubble function and finally $\delta\mathcal{R}$ the perturbation on the spatial Ricci tensor. In terms of these variables, the first order Einsteins equations are

$$3H\delta H + \frac{\delta\mathcal{R}}{4} = \frac{\kappa}{2}\delta\rho, \tag{26}$$
$$3\delta H - D_K^2\sigma = \frac{3\kappa(\rho+p)}{2}V, \tag{27}$$
$$3\delta\dot{H} + 9H\delta H + 3\dot{H}\phi + D^2\phi + \frac{\delta\mathcal{R}}{4} = -\frac{3\kappa}{2}\delta p, \tag{28}$$
$$\psi - \phi - \dot{\sigma} - H\sigma = \kappa\Pi, \tag{29}$$

where $\kappa \equiv 8\pi G$ and taking $c = 1$. The energy-momentum conservation provides two additional equations:

$$\dot{\delta\rho} + \dot{\rho}\phi + 3H(\delta\rho + \delta p) + (3\delta H + D^2V)(\rho + p) = 0, \tag{30}$$
$$\dot{V} - \phi + \frac{\dot{p}}{\rho+p}V + \frac{\delta p}{\rho+p} - \frac{2D_K^2\Pi}{3(\rho+p)} = 0. \tag{31}$$

However, it is easy to check that these two equations are not independent from Einstein's equations. It is a straightforward exercise to show that they can be obtained, respectively, by differentiating Equations (26) and (27) and combining with the Equations (26)–(29). Thus, we have eight variables and only four equations of motion. Note, however that E and B do not appear explicitly in the equations of motion, actually they appear only in the variable σ. This is an artifact from the gauge dependency of the perturbations, but, since these equations are invariant through spatial gauge transformation, they are automatically written in terms of variables invariant under this gauge transformation (for mode details, see [33], for example).

The gauge transformations with respect to the scalar degrees of freedom involve two scalar variables, one representing the spatial transformations that we just discussed and another generating time transformations. This means that, instead of fixing the gauge choosing a gauge condition, we wrote the equations of motion in a gauge invariant way. Note that this is operationally equal to fix the gauge; for example, had we fixed the spatial gauge freedom by choosing $B = 0$, instead of σ appearing in Equations (26)–(29), we would have the E variable. Following this approach, we fix the temporal gauge by rewriting the system of equations using the gauge invariant variables

$$\overline{\delta\rho} \equiv \delta\rho - \dot{\rho}\sigma, \qquad \overline{\delta p} \equiv \delta p - \dot{p}\sigma, \qquad \overline{V} \equiv V + \sigma, \tag{32}$$

$$\Psi \equiv \psi - H\sigma, \qquad \Phi \equiv \phi + \dot{\sigma}, \qquad \zeta \equiv (1-l)\Psi + H\overline{V}, \tag{33}$$

where $l \equiv 2K/[a^2\kappa(\rho+p)]$ and we introduced an additional variable, the Mukhanov–Sasaki variable (ζ) that will be useful later. In terms of these variables, we have Einstein's equations recast to

$$3H(\dot{\Psi} + H\Phi) - D_K^2\Psi = \frac{\kappa}{2}\overline{\delta\rho}, \tag{34}$$

$$\dot{\Psi} + H\Phi = \frac{\kappa(\rho+p)}{2}\overline{V}, \tag{35}$$

$$\left(\frac{\mathrm{d}}{\mathrm{d}t} + 3H\right)(\dot{\Psi} + H\Phi) - \frac{\kappa(\rho+p)\Phi}{2} - D_K^2\left(\frac{\Psi-\Phi}{3}\right) = -\frac{\kappa}{2}\overline{\delta p}, \tag{36}$$

$$\Psi - \Phi = \kappa\Pi, \tag{37}$$

and the energy-momentum tensor conservation to

$$\dot{\overline{\delta\rho}} + \dot{\rho}\Phi + 3H\left(\overline{\delta\rho} + \overline{\delta p}\right) + \left(3\dot{\Psi} + 3H\Phi + D^2\overline{V}\right)(\rho+p) = 0, \tag{38}$$

$$\dot{\overline{V}} - \Phi + \frac{\dot{p}}{\rho+p}\overline{V} + \frac{\overline{\delta p}}{\rho+p} - \frac{2D_K^2\Pi}{3(\rho+p)} = 0. \tag{39}$$

Hence, we reduced the problem from eight to six variables by using the gauge dependency. Nonetheless, we still have only four independent equations of motion.

Similarly to what happens to the background, Einstein's equations and energy-momentum conservation do not provide all the dynamics necessary to describe the perturbations. Naturally, the matter components have their particular degrees of freedom, which, in turn, determine $\delta\rho$, V, δp and Π. There are some codes specialized in solving these set of equations when the matter content is described by a distribution and its evolution by Boltzmann equations, including CAMB, CLASS. Here, we are going to focus on a much simpler problem, which, nevertheless, displays the numerical difficulties found when solving the equations of state for the perturbations.

We can simplify the problem by including two assumptions. First, the matter content does not produce anisotropic stress, i.e., $\Pi = 0$. For the second assumption, we first define the entropy perturbation as

$$S \equiv \delta p - c_s^2\delta\rho, \qquad c_s^2 \equiv \frac{\dot{p}}{\dot{\rho}}. \tag{40}$$

This definition is convenient since it defines a gauge invariant variable. For instance, using Equation (32), we have $S = \overline{\delta p} - c_s^2\overline{\delta\rho}$. Thus, our second assumption is simply that there is no entropy perturbation, i.e., $S = 0$. Note that this will be true for any barotropic fluid (a fluid satisfying $p(\rho)$). We can better understand why these assumptions simplify the system by computing the equations of motion for ζ, differentiating it with respect to time and combining with the other equations. Doing so, we get

$$\dot{\zeta} = H\left(\frac{2c_s^2D^2\Psi}{\kappa(\rho+p)} + \frac{2D^2\Pi}{3(\rho+p)} - \frac{S}{\rho+p}\right), \tag{41}$$

and, to close the system, we combine Equations (35) and (37) to obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{a^3D^2\Psi}{H}\right) = \frac{\kappa(\rho+p)a^3D^2\zeta}{2H^2} + \kappa a^3D^2\Pi. \tag{42}$$

We now recast into a more familiar form. Defining

$$P_\zeta \equiv \frac{a^3 D^2 \Psi}{H}, \qquad z^2 \equiv \frac{\kappa(\rho+p)a^3}{2H^2 c_s^2}, \tag{43}$$

the equations are

$$\dot{\zeta} = \frac{P_\zeta}{z^2} + \frac{2HD^2\Pi}{3(\rho+p)} - \frac{HS}{\rho+p}, \tag{44}$$

$$\dot{P}_\zeta = z^2 c_s^2 D^2 \zeta + \kappa a^2 D^2 \Pi. \tag{45}$$

The pair of equations above reduce to a simple second order differential equation when $\Pi = 0 = S$; this is the simplification that we are looking for.

In retrospect, we started with eight variables and six equations of motion, of the six only four equations of motion were independent. Then, we used the gauge dependency to reduce the system to six variables. Furthermore, we combined the system of equations of motion to arrive at two Equations (44)–(45) and four variables (ζ, P_ζ, S, Π). It is also worth noting that this combination of variables ζ is not arbitrary; it comes naturally from the Hamiltonian of the system when the constraints are reduced [36–38]. Finally, when we apply the assumptions of $S = \Pi = 0$, we get

$$\dot{\zeta}_k = \frac{P_{\zeta k}}{z^2}, \qquad \dot{P}_{\zeta k} = -z^2 w^2 \zeta_k, \quad \text{or} \quad \ddot{\zeta}_k + 2\frac{\dot{z}}{z}\dot{\zeta}_k + w^2\zeta_k = 0, \tag{46}$$

an harmonic oscillator with time dependent mass z^2 and frequency $w \equiv c_s k/a$, where k/a is the square root of the Laplacian eigenvalue used in the mode of the harmonic decomposition.[11]

Numerical Solution

Equations (46) are a reduced and simplified version of the cosmological perturbation equations of motion. Nevertheless, they exhibit many of the numerical challenges also present in a more complicated scenario. For this reason in this section, we discuss the numerical approaches used to solve them in different contexts, then we make a short discussion about the difficulties that arrive in a more complicated scenario.

Equation (46) has three important regimes. In order to understand them, it is easier to first rewrite the equations using the variable $v \equiv z\zeta$, producing the equation of motion

$$\ddot{v}_k + \left(w^2 - \frac{\ddot{z}}{z}\right) v_k = 0. \tag{47}$$

A single fluid with constant equation of state (constant p/ρ) on flat hypersurfaces $K = 0$ has $z \propto a^{3/2}$. In this case, the potential appearing in the equation above is (see Equation (43)),

$$V_z \equiv \frac{\ddot{z}}{z} = \frac{3\ddot{a}}{2a} + \frac{3\dot{a}^2}{4a^2}. \tag{48}$$

In other words, the potential is proportional to a combination of $\dot{H}$ and H^2 and consequently to the Ricci scalar. This amounts to showing that the potential in this case produces a distance scale close

[11] We can write $\zeta = \zeta_k \mathcal{Y}_k$ and $P_\zeta = P_{\zeta k}\mathcal{Y}_k$, for an eigenfunction $\mathcal{Y}_k$ of $-D^2$ with eigenvalue k^2/a^2, i.e., $D^2\mathcal{Y}_k = -k^2/a^2\mathcal{Y}_k$.

to the Hubble radius squared at time t, i.e., $[c/H(t)]^2$.[12] Inspired by this, we define the potential scale $d_V \equiv 1/\sqrt{|V_Z|}$. The regime I takes place when

$$\mathrm{I}: \qquad w^{-1} = \frac{a\lambda}{c_s} \ll d_V, \qquad \lambda \equiv \frac{1}{k},$$

that is, when the mode physical wave-length $a\lambda$ (where λ is the conformal wave-length) over the sound speed is much smaller than the potential scale. The regime II happens when the physical wave-length is much larger than the potential scale d_V, i.e.,

$$\mathrm{II}: \qquad w^{-1} = \frac{a\lambda}{c_s} \gg d_V,$$

and finally the regime III is characterized by

$$\mathrm{III}: \qquad w^{-1} = \frac{a\lambda}{c_s} \approx d_V.$$

Each one of these different regimes have a particular numerical approach to handle them. Moreover, there are situations where it is necessary to put initial conditions on I and evolve up to III and vice versa. For example, in inflationary or bouncing primordial cosmological models, one begins with quantum fields in a vacuum state. In these cases, the modes of interest are in regime I and must be evolved into regime III. Now, Boltzmann codes start the evolution in the radiation dominated expansion past; in this regime, all modes of interest are in regime III, some components evolve to regime I and some stop at regime II. For these reasons, it is necessary to have tools to handle both cases.

For instance, regimes II and III can be solved using common codes of numerical integration since they do not present an oscillatory behavior as regime I. To deal with this last regime, it is worth using the Wentzel–Kramers–Brillouin (WKB) approximation, as we describe in the following.[13] Equation (47) has an approximate solution written in terms of the time-dependent coefficients and their derivatives. To understand the nature of this solution, let us first assume that all coefficients are constants. In this case, the solutions would be

$$v_k^{\mathrm{s}} = A_{\mathrm{s}} \sin\left(\int \mathrm{d}t\, w\right), \qquad v_k^{\mathrm{c}} = A_{\mathrm{c}} \cos\left(\int \mathrm{d}t\, w\right), \tag{49}$$

and a general solution a linear combination of these two. Note that we wrote $\int \mathrm{d}t$ instead of $t - t_0$, such that we can use it as a tentative solution for the time-dependent coefficients. Computing $\ddot{v}_k$ using these tentative solutions results in

$$\ddot{v}_k^{\mathrm{s}} = -\left(w^2 - \frac{\ddot{A}_{\mathrm{s}}}{A_{\mathrm{s}}}\right) v_k^{\mathrm{s}} + \left[\frac{\mathrm{d}}{\mathrm{d}t} \ln\left(A_{\mathrm{s}}\sqrt{w}\right)\right] 2wA_{\mathrm{s}} \cos\left(\int \mathrm{d}t\, w\right), \tag{50}$$

$$\ddot{v}_k^{\mathrm{c}} = -\left(w^2 - \frac{\ddot{A}_{\mathrm{c}}}{A_{\mathrm{c}}}\right) v_k^{\mathrm{c}} - \left[\frac{\mathrm{d}}{\mathrm{d}t} \ln\left(A_{\mathrm{c}}\sqrt{w}\right)\right] 2wA_{\mathrm{c}} \sin\left(\int \mathrm{d}t\, w\right). \tag{51}$$

Naturally, our solution is not exact when the coefficients are time dependent. The expressions above match the leading term for $\ddot{v}_k$ in this phase ($w^2 = c_s^2k^2/a^2$), and the extra terms represent the error present in this approximation. Moreover, if we consider the limit of our approximation ($k \to +\infty$), the error grows to infinity. For this reason, we can use the freedom in choosing the function $A_{\mathrm{s,c}}$ to

[12] It is for this reason that some authors use the expressions super-/sub-Hubble scales, scales much larger/smaller than $c/H(t)$. We can also find the expressions super-/sub-horizon to refer to the same scales. Nevertheless, this nomenclature is based on the fact that, for some simple models, the horizon is also proportional to $c/H(t)$, which can be wrong and counter-intuitive in many cases and as such should be avoided.

[13] For a complete exposition about this subject see [39].

remove the diverging term, i.e., $A_{\mathrm{s,c}} = A_w \equiv 1/\sqrt{w}$. This choice has other advantages, it removed the term that mixed solutions of different phases (it mixed the cosine and sine solutions). Using this choice for $A_{\mathrm{s,c}} = A_w$, the second time derivatives are expressed by

$$\ddot{v}_k^{\mathrm{s,c}} = -\left[\left(w^2 - \frac{\ddot{z}}{z}\right) - \frac{\ddot{A}_w}{A_w} + \frac{\ddot{z}}{z}\right] v_k^{\mathrm{s,c}},$$

where we wrote explicitly the error $-\ddot{A}_w/A_w + \ddot{z}/z$, which is an improvement, since now it does not diverge in the limit $k \to +\infty$, but it stays constant in this limit.

We can further improve our approximation. Starting with the same functional forms in Equation (49) but with an arbitrary time-dependent frequency ν, we arrive at the same second derivatives in Equations (50) and (51) but with $w \to \nu$. Then, making the equivalent choice for $A_{\mathrm{s,c}}$ ($A_{\mathrm{s,c}} = A_\nu = 1/\sqrt{\nu}$), we finally get

$$\ddot{v}_k^{\mathrm{s,c}} = -\left[\left(w^2 - \frac{\ddot{z}}{z}\right) + \nu^2 - w^2 - \frac{\ddot{A}_\nu}{A_\nu} + \frac{\ddot{z}}{z}\right] v_k^{\mathrm{s,c}}.$$

We already know that the choice $\nu = w$ results in a reasonable approximation with an error $\mathcal{O}(k^0)$. Now, if we can try to correct ν to improve the approximation, for example using $\nu = w + f_1/(2w)$, we get the error

$$E_1 = f_1 + \frac{f_1^2}{4w^2} - \frac{\ddot{A}_\nu}{A_\nu} + \frac{\ddot{z}}{z} = f_1 - \frac{\ddot{A}_w}{A_w} + \frac{\ddot{z}}{z} + \mathcal{O}\left(k^{-2}\right). \tag{52}$$

Then, if we choose $f_1 = \ddot{A}_w/A_w - \ddot{z}/z$, our error improves to $\mathcal{O}\left(k^{-2}\right)$.

In the approximations above, we note that, for an error $\mathcal{O}\left(k^0\right)$, we need only the background variables and, for a $\mathcal{O}\left(k^{-2}\right)$, second derivatives of the background variables. It is easy to see that the same pattern continues, i.e., for a $\mathcal{O}\left(k^{-2n}\right)$ error, we need the $2n$ derivatives of the background variables. This points out the first numerical problem with WKB approximation, the need for higher derivatives for better approximations. In practice, the background variables are determined numerically most of the time as we discussed in Section 2.5. This poses a natural problem for the WKB approximation since we would need to compute the derivative numerically or to obtain it during the background integration using analytic expressions. The equations of motion themselves can be used to compute derivatives from the variable states. Nevertheless, both approaches are limited, and numerical differentiation usually results in increasing errors for higher order derivatives, which limits the usage of higher order WKB approximation. The analytical approach through equations of motions provide an easy way to compute low order derivatives, but, for high order derivatives, it becomes more and more complex. The complicated expressions resulting from this analytical approach must be treated carefully, large and complicated analytical expressions frequently produce large cancellation errors when computed naively, see Section 4 for a discussion on cancellation errors.

4. Cancellation Errors

In this section, we discuss the most common cancellation errors that happens in numerical cosmology, for a more in-depth discussion, see, for example, [40]. One common pitfall that plagues numerical computations is the cancellation error. The problem is a natural consequence of the finite precision of the computer representation of real numbers, the Floating-Point (FP) numbers. In this representation, a real number is decomposed in base and exponent. For example, in a decimal base, it is 1.2345×10^5, where the base is 1.2345 and the exponent is 5. In this example, the base occupies 5 decimal places, which defines the precision of our number.[14] Now, the common arithmetic operations, multiplication, sum and division can produce round-off errors, i.e., operating on truncated and rounded numbers produce a different result than operating on the actual numbers and then truncating. For example, adding n positive numbers leads to a $n\epsilon$ roundoff error in the final result,[15] where ϵ is the machine epsilon, the roundoff unit, the smallest number such that $1+\epsilon \neq 1$ in the FP representation, see [40] for more details. In modern implementations of 64-bits double precision FP number $\epsilon \approx 2\times 10^{-16}$. This means that, when adding positive numbers, the roundoff contributes to $n \times 2 \times 10^{-16}$ error, usually much smaller than other sources of errors.[16] For this reason, for these operations, the roundoff errors can be, in most cases, safely ignored. However, the cancellation error happens when we subtract two close numbers. Namely, using the example above with five decimal digits in the base, if we have two real numbers represented exactly by $x = 1.23400000$ and $y = 1.23456789$, their subtraction is given by $\delta_{xy} \equiv y - x = 0.00056789$. Now, the FP representation of these same numbers are $\bar{x} = 1.2340$ and $\bar{y} = 1.2346$ and their subtraction $\bar{\delta}_{xy} = \bar{y} - \bar{x} = 0.0006$. Note that $\bar{\delta}_{xy} = 6\times 10^{-4}$ now only has one significant digit, while the correct result is $\bar{\delta}_{xy} = 5.6789\times 10^{-4}$. This is a very serious problem; any operation done using $\bar{\delta}_{xy}$ will provide a result with at most one significant digit. Going further, in many cases, the two numbers should be equal in the FP representation such that the result is either zero or $\propto \bar{y}\epsilon$ if they differ by a rounding operation.

Fortunately, there are many cases where the problem can be avoided. For instance, given the function

$$f(x) = \sqrt{1+x^2} - 1,$$

when the computer calculates its value for a given FP value of x, it divides the task in several steps:[17]

$$\bar{x} := x; \quad \bar{y} := \bar{x}\times\bar{x}; \quad \bar{y} := \bar{y} + 1.0; \quad \bar{y} := \sqrt{\bar{y}}; \quad \bar{w} = 1.0; \quad \bar{f} = \bar{y} - \bar{w},$$

where $:=$ represents the attribution of a variable, the bar variables the FP representations and $\bar{f}$ the final result. If $x < \sqrt{\epsilon}$, then we have that the final value of $\bar{y}$ is exactly 1.0; consequently, $\bar{f}$ will be either zero or $\propto \epsilon$ depending on rounding errors. This is a catastrophic result, for $x < \sqrt{\epsilon}$, this function produces no significant digits! To make matters worse, such expressions frequently appear within more complicated calculations, e.g., integrating $\int_0^1 \mathrm{d}x f(x) g(x)$ for a numerically well behaved function $g(x)$ produces a result with all or even no significant digits depending on where $g(x)$ peaks (to the left or to the right of $x \approx \sqrt{\epsilon}$), the integration routines estimate the error assuming that all digits are significant, so the final error estimate produced by the integrator can be much lower than the actual error. Nevertheless, there is a simple solution for this problem. Multiplying the numerator and the denominator of $f(x)$ by $\sqrt{1+x^2}+1$ gives

$$f(x) = \frac{x^2}{\sqrt{1+x^2}+1},$$

which is numerically well behaved for any x inside the FP representation. This shows that, in some cases, a simple manipulation of the expressions is enough to cure the cancellation error.

[14] The precision is usually defined in binary basis, since it is that which are used in the computer representation. This precision does not translate to a fixed number of decimal places. For instance, in the IEEE 754 standard, the common used double precision FP number has a 53 bits base and 11 bits exponent; this basis roughly translates to 15.95 decimal places.

[15] However, in practice, we have the addition of positive and negative numbers. In this case, the error sign of the individual operations are mixed and, consequently, some cancel out. As a rule of thumb, some authors suggest the use of the square root of the number of operations to estimate the roundoff error, see [41] for more details about the statistical analysis of the accumulated roundoff error.

[16] With the exception of computations that involves a very large number of operations, n must be of the order of trillions ($\propto 10^{12}$) to produce a 0.01% error.

[17] In many cases, the order of the operations follow a restrict precedence rule dependent on the compiler/specification.

A more subtle example is the spherical Bessel function of order one

$$j_1(x) = \frac{\sin x - x\cos x}{x^2}.$$

In this representation in terms of trigonometric functions, the function is not well behaved for $x \ll 1$. For $x \lesssim \sqrt{\epsilon}$, $\overline{\sin}(\bar{x}) = x$ and $\overline{\cos}(\bar{x}) = 1.0$, again the result is either zero or $\propto \epsilon/x^2$, for really small x, the x^2 can underflow to zero and produce $0/0$ or $\epsilon/0$, represented as the FP `nan` (not a number) or `inf` (infinity). In this case, there is no simple trick to make this function well behaved. The solution is to split the computation in two cases $x < c$ and $x \geq c$ for a large enough cut value of c. For $x \geq c$, we can use the expression above to compute $j_1(x)$; for the other branch, we use the Maclaurin Series, i.e.,

$$j_1(x) = \frac{x}{3} - \frac{x^3}{30} + \frac{x^5}{840} + \dots,$$

where the cut c and the amount of terms in the series used depend on the FP precision.

This last example is actually a real world case, in order to compute the top-hat filtering (see for example [42]) of the matter power spectrum, one needs to integrate the power spectrum in k times $[j_1(kR)]^2$.

5. Conclusions

In this review, we presented a summary of subjects in numerical cosmology, aiming at describing primary steps necessary to develop a numerical cosmology computation engine. For this, we followed the structure of the standard cosmological model. That is, we began by describing a homogeneous and isotropic background in the context of GR; then, we discussed the small perturbations around the background. Related to the first, we approached the organization of the codes and how some respective functions (and/or ODE systems) are integrated. Regarding the perturbations, we focused on the single field case to describe the three different regimes that require different numerical methods. We concluded by discussing the cancellation errors that can appear in the context of both background and perturbative scenarios. These specific topics were chosen to match those commonly found in the currently available codes (e.g., CCL, CLASS, NumCosmo). The aim is to provide to the reader initial concepts necessary to understand and contribute to these codes.

Author Contributions: Conceptualization: S.D.P.V.; Writing: both authors; Writing–review and editing: M.P.-L.

Funding: This research received no external funding.

Acknowledgments: S.D.P.V. would like to thank the financial support from the PNPD/CAPES (Programa Nacional de Pós-Doutorado/Capes, reference 88887.311171/2018-00).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GR	General Relativity
SNeIa	Type Ia Supernovae
ODE	Ordinary Differential Equation
CMB	Cosmic Microwave Background
LSS	Large Scale Structure
NumCosmo	Numerical Cosmology
CAMB	Code for Anisotropies in the Microwave Background
CLASS	The Cosmic Linear Anisotropy Solving System
MCMC	Markov Chain Monte Carlo

FLRW	Friedmann–Lemaître–Robertson–Walker
DM	Dark Matter
DE	Dark Energy
WKB	Wentzel–Kramers–Brillouin
FP	Floating-Point

References

1. Linde, A. On the problem of initial conditions for inflation. *arXiv* **2017**, arXiv:1710.04278v1.
2. Martin, J.; Vennin, V. Observational constraints on quantum decoherence during inflation. *arXiv* **2018**, arXiv:1801.09949v1.
3. Hinshaw, G.; Larson, D.; Komatsu, E.; Spergel, D.N.; Bennett, C.L.; Dunkley, J.; Nolta, M.R.; Halpern, M.; Hill, R.S.; Odegard, N.; et al. Nine-year wilkinson microwave anisotropy probe (wmap) observations: Cosmological parameter results. *Astrophys. J. Suppl. Ser.* **2013**, *208*, 19. [CrossRef]
4. SDSS-III Collaboration. SDSS-III: Massive Spectroscopic Surveys of the Distant Universe, the Milky Way Galaxy, and Extra-Solar Planetary Systems. Available online: www.sdss3.org/collaboration/description.pdf (accessed on 28 September 2019).
5. Ade, P.A.; Aghanim, N.; Arnaud, M.; Ashdown, M.; Aumont, J.; Baccigalupi, C.; Banday, A.J.; Barreiro, R.B.; Bartlett, J.G.; Bartolo, N.; et al. [Planck Collaboration] Planck 2015 results. XIII. Cosmological parameters. *Astron. Astrophys.* **2015**, *594*, A13. [CrossRef]
6. Abbott, T.M.C.; Abdalla, F.B.; Allam, S.; Amara, A.; Annis, J.; Asorey, J.; Avila, S.; Ballester, O.; Banerji, M.; Barkhouse, W.; et al. The Dark Energy Survey Data Release 1. *arXiv* **2018**, arXiv:1801.03181v1.
7. Vitenti, S.D.P.; Penna-Lima, M. Numerical Cosmology—NumCosmo. *arXiv* **2014**, arXiv:1408.013.
8. Lewis, A.; Challinor, A.; Lasenby, A. Efficient Computation of CMB anisotropies in closed FRW models. *Astrophys. J.* **2000**, *538*, 473–476. [CrossRef]
9. Howlett, C.; Lewis, A.; Hall, A.; Challinor, A. CMB power spectrum parameter degeneracies in the era of precision cosmology. *J. Cosmol. Astropart. Phys.* **2012**, *2012*, 027. [CrossRef]
10. Lewis, A. CAMB Notes. Available online: http://cosmologist.info/notes/CAMB.pdf (accessed on 28 September 2019).
11. Lesgourgues, J. The Cosmic Linear Anisotropy Solving System (CLASS) I: Overview. *arXiv* **2011**, arXiv:1104.2932.
12. Blas, D.; Lesgourgues, J.; Tram, T. The Cosmic Linear Anisotropy Solving System (CLASS). Part II: Approximation schemes. *J. Cosmol. Astropart. Phys.* **2011**, *2011*, 034. [CrossRef]
13. Lesgourgues, J. The Cosmic Linear Anisotropy Solving System (CLASS) III: Comparision with CAMB for LambdaCDM. *arXiv* **2011**, arXiv:1104.2934.
14. Lesgourgues, J.; Tram, T. The Cosmic Linear Anisotropy Solving System (CLASS) IV: Efficient implementation of non-cold relics. *J. Cosmol. Astropart. Phys.* **2011**, *9*, 32. [CrossRef]
15. Lewis, A.; Bridle, S. Cosmological parameters from CMB and other data: A Monte Carlo approach. *Phys. Rev.* **2002**, *D66*, 103511. [CrossRef]
16. Lewis, A. Efficient sampling of fast and slow cosmological parameters. *Phys. Rev. D* **2013**, *87*, 103529. [CrossRef]
17. Audren, B.; Lesgourgues, J.; Benabed, K.; Prunet, S. Conservative constraints on early cosmology with MONTE PYTHON. *J. Cosmol. Astropart. Phys.* **2013**, *2013*, 001. [CrossRef]
18. Weinberg, S. *Cosmology*; Oxford University Press: Oxford, UK, 2008.
19. Peter, P.; Uzan, J.P. *Primordial Cosmology*; Oxford University Press: Oxford, UK, 2009.
20. Hogg, D.W. Distance measures in cosmology. *arXiv* **2007**, arXiv:9905116.
21. Vitenti, S.D.P. Unitary evolution, canonical variables and vacuum choice for general quadratic Hamiltonians in spatially homogeneous and isotropic space-times. *arXiv* **2015**, arXiv:1505.01541.
22. Branch, D.; Miller, D.L. Type IA supernovae as standard candles. *Astrophys. J. Lett.* **1993**, *405*, L5–L8. [CrossRef]
23. Betoule, M.; Kessler, R.; Guy, J.; Mosher, J.; Hardin, D.; Biswas, R.; Astier, P.; El-Hage, P.; Konig, M.; Kuhlmann, S.; et al. Improved cosmological constraints from a joint analysis of the SDSS-II and SNLS supernova samples. *Astron. Astrophys.* **2014**, *568*, A22. [CrossRef]

24. Bassett, B.; Hlozek, R., Baryon acoustic oscillations. In *Dark Energy: Observational and Theoretical Approaches*; Cambridge University Press: Cambridge, UK; New York, NY, USA, 2010; p. 246.
25. De Boor, C. *A Practical Guide to Splines*; Springer: Berlin, Germany, 2001.
26. GSL Project Contributors. *GSL—GNU Scientific Library—GNU Project—Free Software Foundation*; Free Software Foundation, Inc.: Boston, MS, USA, 2010.
27. Behforooz, G.H.; Papamichael, N. End conditions for cubic spline interpolation. *IMA J. Appl. Math.* **1979**, *23*, 355–366. [CrossRef]
28. Chisari, N.E.; Alonso, D.; Krause, E.; Leonard, C.D.; Bull, P.; Neveu, J.; Villarreal, A.; Singh, S.; McClintock, T.; Ellison, J.; et al. [LSST Dark Energy Science Collaboration] Core Cosmology Library: Precision Cosmological Predictions for LSST. *arXiv* **2018**, arXiv:1812.05995.
29. Heineman, G.; Pollice, G.; Selkow, S. *Algorithms in a Nutshell (In a Nutshell (O'Reilly))*; O'Reilly Media: Springfield, MO, USA, 2008.
30. Hindmarsh, A.C.; Brown, P.N.; Grant, K.E.; Lee, S.L.; Serban, R.; Shumaker, D.E.; Woodward, C.S. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.* **2005**, *31*, 363–396. [CrossRef]
31. Bardeen, J.M. Gauge-invariant cosmological perturbations. *Phys. Rev. D* **1980**, *22*, 1882–1905. [CrossRef]
32. Mukhanov, V.F.; Feldman, H.A.; Brandenberger, R.H. Theory of cosmological perturbations. *Phys. Rep.* **1992**, *215*, 203–333. [CrossRef]
33. Vitenti, S.D.P.; Falciano, F.T.; Pinto-Neto, N. Covariant Bardeen perturbation formalism. *Phys. Rev. D* **2014**, *89*, 103538. [CrossRef]
34. Vitenti, S.D.P.; Pinto-Neto, N. Large Adiabatic Scalar Perturbations in a Regular Bouncing Universe. *Phys. Rev. D* **2012**, *85*, 023524. [CrossRef]
35. Pinto-Neto, N.; Vitenti, S.D.P. Comment on "Growth of covariant perturbations in the contracting phase of a bouncing universe". *Phys. Rev. D* **2014**, *89*, 028301. [CrossRef]
36. Vitenti, S.D.P.; Falciano, F.T.; Pinto-Neto, N. Quantum Cosmological Perturbations of Generic Fluids in Quantum Universes. *Phys. Rev. D* **2013**, *87*, 103503. [CrossRef]
37. Falciano, F.T.; Pinto-Neto, N.; Vitenti, S.D.P. Scalar field perturbations with arbitrary potentials in quantum backgrounds. *Phys. Rev. D* **2013**, *87*, 103514. [CrossRef]
38. Peter, P.; Pinto-Neto, N.; Vitenti, S.D.P. Quantum cosmological perturbations of multiple fluids. *Phys. Rev. D* **2016**, *93*, 023520. [CrossRef]
39. Bender, C.M.; Orszag, S.A. *Advanced Mathematical Methods for Scientists and Engineers: Asymptotic Methods and Perturbation Theory*; Springer: Berlin, Germany, 1978.
40. Kincaid, D.; Cheney, W. *Numerical Analysis: Mathematics of Scientific Computing*; American Mathematical Society: Providence, RI, USA, 2009.
41. Higham, N. *Accuracy and Stability of Numerical Algorithms*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2002.
42. Penna-Lima, M.; Makler, M.; Wuensche, C.A. Biases on cosmological parameter estimators from galaxy cluster number counts. *J. Cosmol. Astropart. Phys.* **2014**, *2014*, 39. [CrossRef]

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).