Milestone 1: The background evolution of the universe

Hans Winther

January 28, 2020

1 Problem statement and deliverables

The topic of the first milestone of the AST5220 project is the evolution of the uniform background in the universe. The final goal is to compute the Hubble parameter as x, the logarithm of the scale factor a, and the conformal time η as a function of x.

The deliverables are the following:

- A report defining the quantities to be computed, a short description of the algorithms used, and plots of $H(z)$, $H(x)$, $\eta(x)$, and one plot with $\Omega_b(x) + \Omega_{\rm CDM}(x)$, $\Omega_r(x)$ and $\Omega_\Lambda(x)$ together. If you are a Ph. d. student, you should also include Ω_ν in your computations.
- A transcript of the source code used for the evaluations.

Also try to time the code to see how fast it runs and add that to the report. There are no points subtracted for doing it slow. This is just useful in case it is slow so we can fix this problem early so you won't end up struggeling to get results in a reasonable time in future modules.

Note that there is a free choice of programming language. However, this is only recommended for experienced programmers; if you feel less certain, I strongly recommend C++ (or Fortran 90), and template codes will be provided these languages.

2 How to get started - C++

- Copy the skeleton source code to where you want it:
  - `cd where-you-want-your-AST5220-directory`
  - `mkdir AST5220`
  - `cd AST5220`
  - `cp -r /mn/stornext/u3/hansw/Teaching/AST5220/CXX_Template .`
- Look at following source files in CXX_Template/ – Main.cpp, BackgroundCosmology.h and BackgroundCosmology.cpp. If you use emacs, you do this by "emacs filename.cpp".
- The code is run from Main.cpp, but you don't need to change anything here. BackgroundCosmology.h contains the class definition and you don't need to change anything here either unless you make some new class functions or variables then you will need to add them here. The file BackgroundCosmology.cpp is where the work is done. Take a look at it and see the TODO lists for things you will have to implement.
- Look at the provided Makefile as well; see if you can understand what the different elements of this are. (Note: You don't have to understand this at all at this time, but it's useful to roughly know what's going on.)
- To compile the program, cd to the source directory, and write "make".
- On the ITA system you don't have to do anything more, however if you want to compile and run it on your laptop you will have to install the GSL library. See the README file in CXX_Template for instructions for how to do this or come ask me and we will get it done.

- To run the program after compiling it, write "./cmb". Just some info about the class will be shown at the present stage and then it will crash with an error "Spline eta not created". Implementing this is one of your tasks.
- If you are unfamiliar with C++, but do want to do the project in C++ I recommend the tutorial at https://www.w3schools.com/cpp/cpp_intro.asp
- See Example.cpp (and write make examples; to compils) for some examples on how to make splines and solve ODEs. See the slides from the lecture for more info about this.
- It is recommended to read the first two sections of Callin (2005); the goal of the present project is essentially to implement the equations in Section 2 of that paper.

3 How to get started - Fortran

- Copy the skeleton source code to where you want it:
  - `cd where-you-want-your-AST5220-directory`
  - `mkdir AST5220`
  - `cd AST5220`
  - `cp -r /mn/stornext/u3/hansw/Teaching/AST5220/Fortran_Template .`
- Look at following source files in Fortran_Template/ – params.f90, time_mod.f90 and cmbspec.f90. If you use emacs, you do this by "emacs filename.f90".
- Look at the provided Makefile as well; see if you can understand what the different elements of this are. (Note: You don't have to understand this at all at this time, but it's useful to roughly know what's going on. Ask HKE if you want a more detailed explanation.)
- To compile the program, cd to the source directory, and write "make".
- To run the program after compiling it, write "./cmbspec". No output will be given at the present stage, but it should run.
- To check that you can actually make changes to the program, open up cmbspec.f90 in an editor, and add the following line after the initialization statement: "write(*,*) 'Hello world!' "
- Recompile, and re-run the program. Now you should see "Hello world!" in the terminal, and you are ready to start working on your assignment :-)
- If you are unfamiliar with F90, but do want to do the project in F90, I strongly recommend Bo Einarsson's F90 manual (available both in Swedish and English). Links are given on the course web page.
- It is recommended to read the first two sections of Callin (2005); the goal of the present project is essentially to implement the equations in Section 2 of that paper.

4 Theoretical background

The task in this project is to compute the expansion history of the universe, and look at the uniform background densities of the various matter and energy components. Let us first define the Friedmann-Robertson-Walker metric for flat space ($k = 0$),

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

or in Cartesian coordinates simply

$$ds^2 = -c^2dt^2 + a^2(t)\left(dx^2 + dy^2 + dz^2\right) \tag{3}$$
$$= a^2(t)(-d\eta^2 + dx^2 + dy^2 + dz^2) \tag{4}$$

where $a(t)$ is the scale factor, which measures the size of the universe relative to today ($a_0 = a(\text{today}) = 1$), and η is called conformal time. As we will be looking at phenomena that varies strongly over a wide range of time scales, we will mostly be using the logarithm of the scale factor, $x \equiv \ln a$, as our time variable. A fifth time variable is the redshift, z, which is defined as $1 + z = a_0/a(t)$.

Einstein's General Relativity describes how the metric evolves with time, given some matter and density components. The relevant equation for our purposes here is the Friedmann equation, which may be written on the following form,

$$H = H_0\sqrt{(\Omega_b + \Omega_{\rm CDM})a^{-3} + (\Omega_r + \Omega_\nu)a^{-4} + \Omega_\Lambda}, \tag{5}$$

where $H \equiv \dot{a}/a$ is the Hubble parameter (and dot denotes derivatives with respect to physical time, $\dot{} = d/dt$), and Ω_b, $\Omega_{\rm CDM}$, Ω_r, Ω_ν, and Ω_Λ are the relative densities of baryonic matter, dark matter, radiation, neutrinos and dark energy, respectively. Also note that Callin and the Fortran template calls $\Omega_{\rm CDM} = \Omega_m$. (Recall that $\Omega_x = \rho_x/\rho_c$, where $\rho_c = 3H^2/8\pi G$.) If you don't need neutrinos, simply set $\Omega_\nu = 0$ here. For convenience, we also introduce a scaled Hubble parameter, $\mathcal{H} \equiv aH$; in the template code, this is referred to as "H_p" (F90) and "Hp" (C++) – "H prime".

The Friedmann equations also describe how each component evolve with time, and here we simply write down the results:

$$\rho_{\rm CDM} = \rho_{\rm CDM,0}a^{-3} \tag{6}$$
$$\rho_b = \rho_{b,0}a^{-3} \tag{7}$$
$$\rho_r = \rho_{b,0}a^{-4} \tag{8}$$
$$\rho_\nu = \rho_{\nu,0}a^{-4} \tag{9}$$
$$\rho_\Lambda = \rho_{\Lambda,0} \tag{10}$$

Here, quantities with subscripts 0 indicate today's values.

Another crucial concept for CMB computations is that of the "horizon". This is simply the distance that light may have travelled since the Big Bang, $t = 0$. If the universe was static, this would simply have been ct, but since the universe also expands, it will be somewhat larger. Note that the horizon is a strictly increasing quantity with time, and we can therefore use it as a time variable. This is often called "conformal time", and is denoted η.

To find a computable expression for η, we note that

$$\frac{d\eta}{dt} = \frac{c}{a}. \tag{11}$$

The left-hand side of this equation may be rewritten into

$$\frac{d\eta}{dt} = \frac{d\eta}{da}\frac{da}{dt} = \frac{d\eta}{da}aH, \tag{12}$$

such that

$$\frac{d\eta}{da} = \frac{c}{a^2H} = \frac{c}{a\mathcal{H}} \tag{13}$$

This is a differential equation for η, that can either be solved numerically by direct integration, or by plugging the expression into a ordinary differential equation solver.

5 Implementation - C++

The implementation in this milestone consists of completing the provided class "BackgroundCosmology". Explicitly, the tasks to be completed are the following:

- Set up a grid of x values between some initial and end time, as specified in the code for which to compute η on. The range of x-values should correspond to a values in the range $a_{\rm min} = 10^{-7}$ to $a = 1$ and you should use about $n \sim 100 - 1000$ points.
- Compute the conformal time at each of these grid points, and compute a spline through the resulting data points.
- Complete all the functions marked TODO like "H_of_x", "H_of_a", "Hp_of_x", "get_OmegaCDM" etc. These should return the value of the Hubble function, the scaled Hubble function $\mathcal{H} \equiv aH$ and the density parameters.

Once this is completed, you should output and plot the requested functions. A good plotting program for this purpose is "python". For example if we have a file with x in the first column and eta in the second column then we can plot it as follows (assumming you have the matplotlib library):

- import matplotlib.pyplot as plt
- import numpy as np
- data = np.loadtxt('cosmology.txt')
- x = data[:,0]
- eta = data[:,1]
- plt.plot(x,eta)
- plt.show()

If you want the y axis to be log-scaled then you can add plt.yscale('log') before plotting. If you want to change the axes you can use plt.xlim(xmin,xmax) and plt.ylim(ymin,ymin). For more info search for "matplotlib examples".

6 Implementation - Fortran

The implementation in this milestone consists of completing the provided "time_mod.f90" Fortran 90 module (or equivalent in another language). Explicitly, the tasks to be completed are the following:

- Set up a grid of x values between some initial and end time, as specified in the code. (See Callin 2005 for explanation of the chosen values; for later integration of the Boltzmann and Einstein equations, we need a fine grid during recombination (200 points between $z = 1630$ and $z = 614$), and a coarser grid between the end of recombination and today (300 points between $z = 614$ and $z = 0$).)
- Set up a corresponding grid of scale factors, a.
- Set up a second and independent grid of x-values for the conformal time. This grid should start at $a = 10^{-10}$ and end at $a = 1$, and contain 1000 points.
- Compute the conformal time at each of these grid points, and compute a spline through the resulting data points. (Use the function "spline" in "spline_1D_mod.f90"; this evaluates the second derivatives at each grid point, which later will be used for interpolating *between* grid points. Set yp1=yp2=10^{30}, corresponding to a so-called "natural spline", ie., with zero second derivatives at the end points. For more details on splines, see section 3.3 in Numerical Recipes.)
- Complete the functions called "get_H", "get_H_p" and "get_dH_p"; these should return the value of the Hubble parameter, the scaled Hubble parameter $\mathcal{H} \equiv aH$ and the derivative of the latter, $d\mathcal{H}/dx$, as functions of x. Note that you have access to all the global cosmological parameters listed in "params.f90".
- Complete the function "get_eta", that returns the conformal time at arbitrary times. Use spline interpolation for this, through the function called "splint" in "spline_1D_mod.f90", and by taking advantage of the previously precomputed η tables.

Once this is completed, you should output and plot the requested functions. A good plotting program for this purpose is "xmgrace", which should be available on all computers. To use this, simply output the desired function into a text file with two columns (x and x), and write "xmgrace filename.dat". Then you will be able to manipulate the plot in a graphical user interface.

6.1 Concluding remark

If you are unfamiliar with Linux, programming, C++, Fortran 90, cosmology or whatever, and find that you're having problems with completing the project: Don't despair! And don't spend a lot of time trying to figure out what's wrong on your own: Ask me, and we'll try to solve the problem together!