# CLASS: Cosmological Boltzman Code

Resuelve el sistema de perturbaciones cosmológicas en un universo de FRW tipo LDCM (aunque incluye otras varias opciones), calcula el fondo cósmico de microondas y observables de formación de estructura a grandes escalas. Para ello:

- Resuelve las ecuaciones del background (factor de escala, distancias, densidades promedio)
- Ecuaciones de Boltzman en recombinación y reionización

http://www.class-code.net

Dos formas de obtener el código:

1) Descargar el archivo comprimido .tar.gz del sitio web y descomprimirlo en consola:

$ tar -zxvf class_public-vx.y.z.tar.gz

2) Usar git en la terminal:

$ git clone https://github.com/lesgourg/class_public.git class

# INSTALAR Y CORRER CLASS (FORMA TRADICIONAL)

https://github.com/lesgourg/class_public/wiki/Installation
http://lesgourg.github.io/class_public/class_public-2.7.0/doc/manual/html/index.html

II- Compilar:

REQUERIMIENTOS LINUX: gcc y python (de preferencia anaconda o bien tener instalado pip3)

```
$ cd class/
class]$ make
```

Genera: ejecutable, libclass.a y un wrapper de python (classy).

O

```
class]$ make class
```

Genera: ejecutable no se crea libclass.a

Genera: ejecutable, libclass.a y un wrapper de python (classy).

III- Correr class:

```
Class]$ ./class explanatory.ini
```

Genera: outputs en archivos de texto

```
Writing output files in ...
```

# Inifile: explanatory.ini

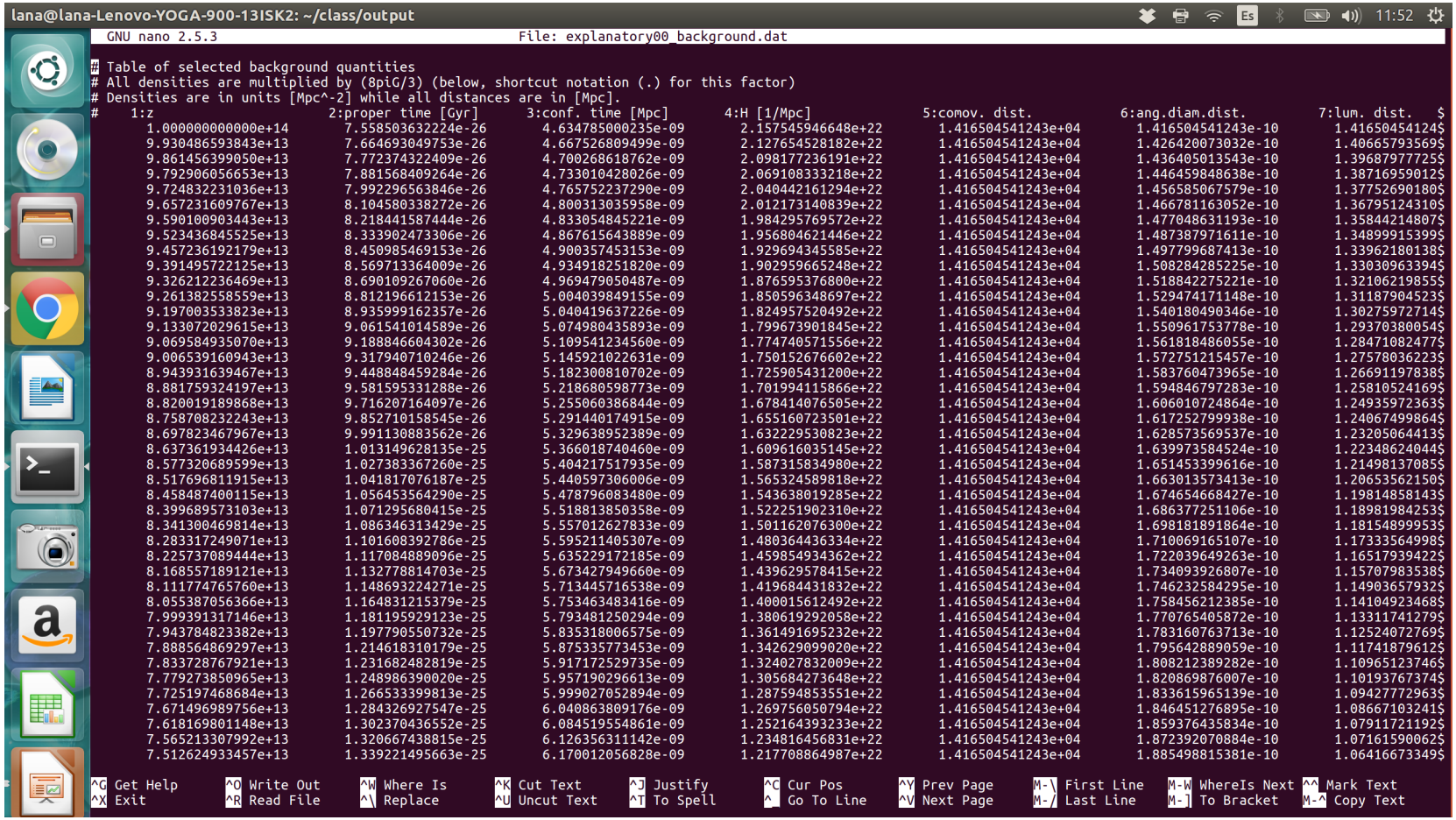
```
1) Hubble parameter : either 'H0' in km/s/Mpc or 'h' or '100*theta_s' where the
   latter is the peak scale parameter 100(ds_dec/da_dec) close to 1.042143
   (default: 'h' set to 0.67556)

#H0 = 67.556
h =0.67556
#100*theta_s = 1.042143

2) photon density: either 'T_cmb' in K or 'Omega_g' or 'omega_g' (default:
   'T_cmb' set to 2.7255)

T_cmb = 2.7255
#Omega_g =
#omega_g =

3) baryon density: either 'Omega_b' or 'omega_b' (default: 'omega_b' set to
   0.022032)

#Omega_b =
omega_b = 0.022032
```

```
5) density of cdm (cold dark matter): 'Omega_cdm' or 'omega_cdm' (default:
   'omega_cdm' set to 0.12038)

#Omega_cdm =
omega_cdm = 0.12038


7) curvature: 'Omega_k' (default: 'Omega_k' set to 0)

Omega_k = 0.


8a) Dark energy contributions. At least one out of three conditions must be satisfied:
    i)   'Omega_Lambda' unspecified.
    ii)  'Omega_fld' unspecified.
    iii) 'Omega_scf' set to a negative value. [Will be refered to as
         unspecified in the following text.]
    The code will then use the first unspecified component to satisfy the
    closure equation (sum_i Omega_i) equals (1 + Omega_k)
    (default: 'Omega_fld' and 'Omega_scf' set to 0 and 'Omega_Lambda' inferred
    by code)

Omega_Lambda = 0.7
#Omega_fld = 0
#Omega_scf = 0
```

# OUTPUTS

```
lana@lana-Lenovo-YOGA-900-13ISK2:~/class/output$ ls
explanatory00_background.dat  explanatory00_cl_lensed.dat   explanatory00_unused_parameters
explanatory00_cl.dat          explanatory00_parameters.ini
lana@lana-Lenovo-YOGA-900-13ISK2:~/class/output$
```

Output$ nano/explanatory00_background.dat



# INSTALAR Y CORRER CLASS EN PYTHON (FORMA “HUMAN-FRIENDLY”)

Dentro de python/ se encuentra el WRAPPER de PYTHON classy escrito en cython
El cual se puede instalar como librería de python en el sistema:

```
class]$ cd python/
python]$ python setup.py install –user
```

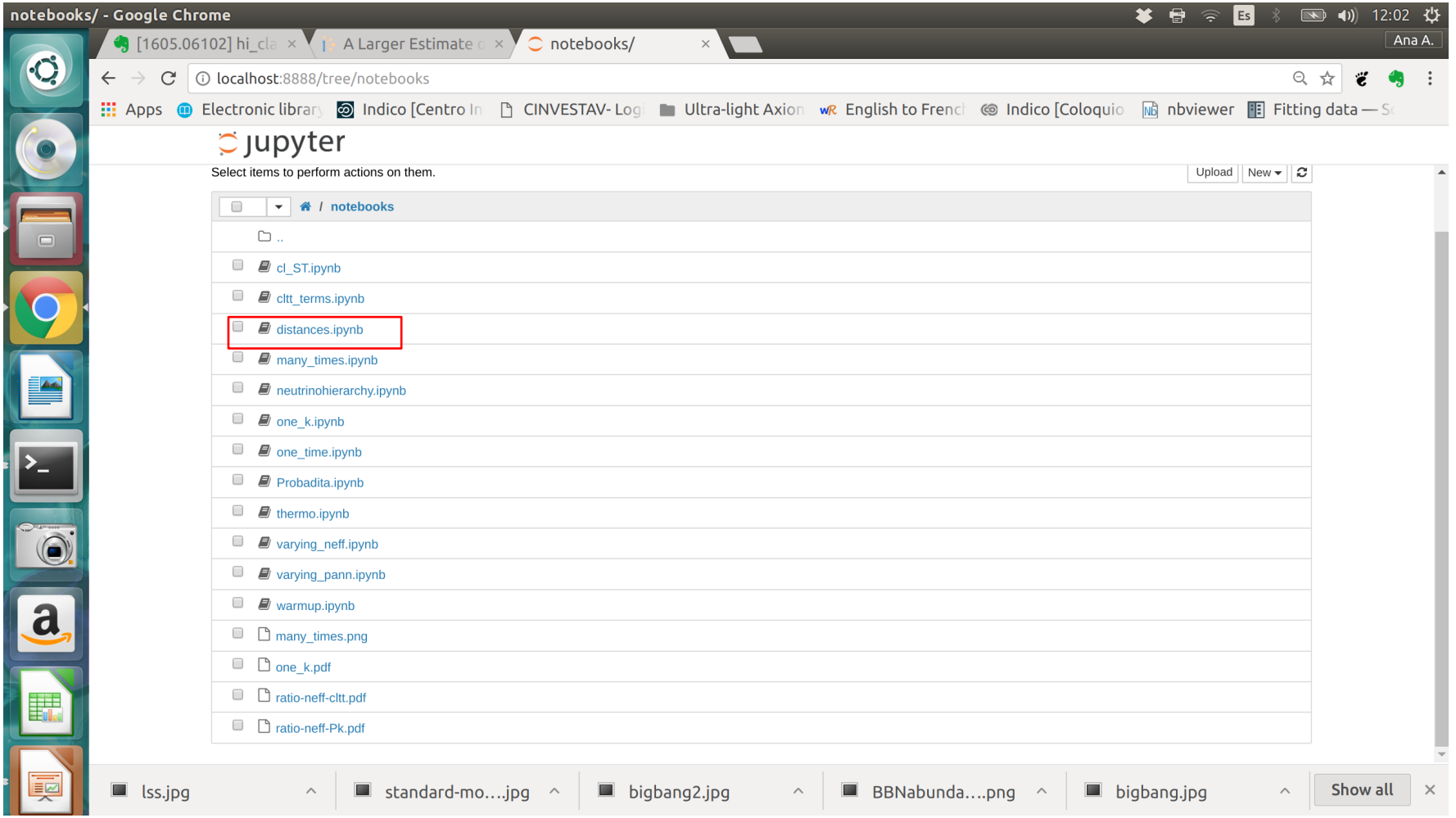
Para checar que la instalación fue exitosa:

```
Python]$ python
>>> from classy import Class
```

Si no aparece error classy esta bien instalada

TRES OPCIONES:

1. Usar python en terminal
2. Hacer un script.py y correr $ python script.py
3. Usar un notebook de jupyter:
   con anaconda instalado y classy instalado:
   correr dentro de la carpeta notebooks en indico $ jupyter notebook


notebooks/ - Google Chrome
12:02
Ana A.
[1605.06102] hi_cla
A Larger Estimate
notebooks/
localhost:8888/tree/notebooks
Apps
Electronic library
Indico [Centro In
CINVESTAV- Logi
Ultra-light Axion
English to Frenc
Indico [Coloquio
nbviewer
Fitting data — Sc
jupyter
Select items to perform actions on them.
Upload
New
/ notebooks
..
cl_ST.ipynb
cltt_terms.ipynb
distances.ipynb
many_times.ipynb
neutrinohierarchy.ipynb
one_k.ipynb
one_time.ipynb
Probadita.ipynb
thermo.ipynb
varying_neff.ipynb
varying_pann.ipynb
warmup.ipynb
many_times.png
one_k.pdf
ratio-neff-cltt.pdf
ratio-neff-Pk.pdf
lss.jpg
standard-mo….jpg
bigbang2.jpg
BBNabunda….png
bigbang.jpg
Show all

jupyter distances Last Checkpoint: 09/06/2018 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Python [conda root]

Code CellToolbar

In [1]:
```
%matplotlib notebook
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from classy import Class
```

In [2]:
```
font = {'size'   : 20, 'family':'STIXGeneral'}
axislabelfontsize='large'
matplotlib.rc('font', **font)
matplotlib.mathtext.rcParams['legend.fontsize']='medium'
```

In [3]:
```
#Lambda CDM
LCDM = Class()
LCDM.set({'Omega_cdm':0.25,'Omega_b':0.05})
LCDM.compute()
```

In [9]:
```
#Einstein-de Sitter
CDM = Class()
CDM.set({'Omega_cdm':0.95,'Omega_b':0.05})
CDM.compute()

# Just to cross-check that Omega_Lambda is negligible
# (but not exactly zero because we neglected radiation)
Omega_lambda = LCDM.get_current_derived_parameters(['Omega0_lambda'])
Omega_lambda
```

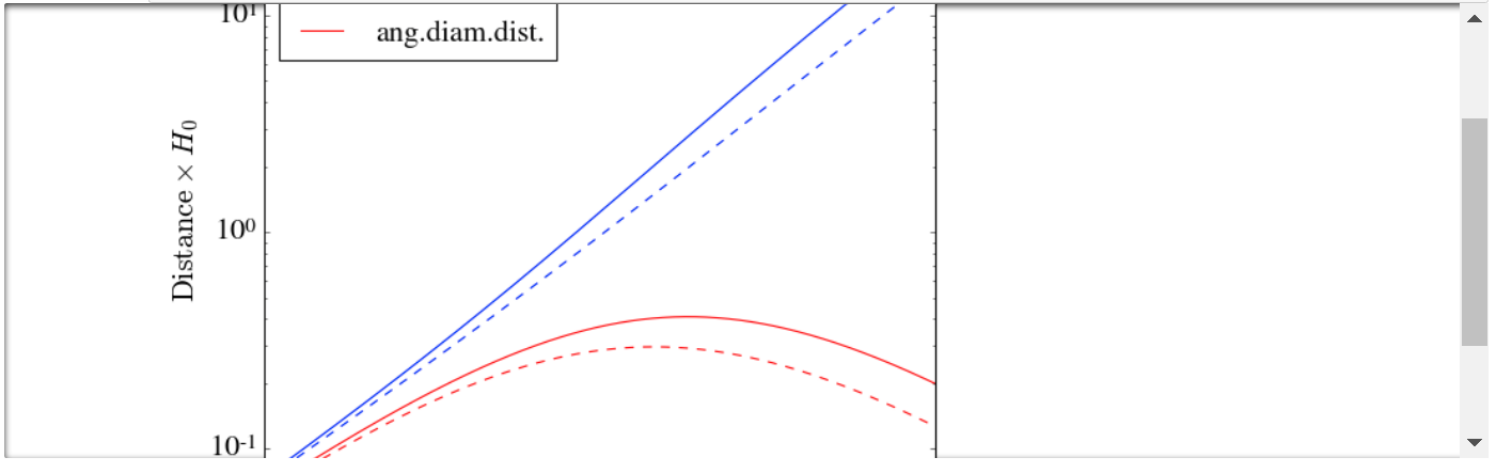
Out[9]:
```
{'Omega0_lambda': 0.6999083286434546}
```

In [5]:
```
#Get background quantities and recover their names:
baLCDM = LCDM.get_background()
baCDM = CDM.get_background()
```

In [6]:
```
#Get background quantities and recover their names:
baLCDM = LCDM.get_background()
baCDM = CDM.get_background()
```

In [7]:
```
#Get H_0 in order to plot the distances in this unit
fLCDM = LCDM.Hubble(0)
fCDM = CDM.Hubble(0)
```

In [12]:
```
#namelist = ['lum. dist.','comov. dist.','ang.diam.dist.']
namelist = ['lum. dist.','ang.diam.dist.']
colours = ['b','r']
for name in namelist:
    idx = namelist.index(name)
    plt.loglog(baLCDM['z'],fLCDM*baLCDM[name],colours[idx]+'-')
plt.legend(namelist,loc='upper left')
for name in namelist:
    idx = namelist.index(name)
    plt.loglog(baCDM['z'],fCDM*baCDM[name],colours[idx]+'--')
plt.xlim([0.07, 10])
plt.ylim([0.08, 20])

plt.xlabel(r"$z$")
plt.ylabel(r"$\mathrm{Distance}\times H_0$")
```




In [ ]:
```
plt.savefig('distances.pdf')
```

# USO BÁSICO

1. Importar el módulo Class

2. Crear un “caso” (instance)

3. Crear un diccionario con los parametros que uno quiere variar

4. Inicializar las estructuras con “compute”

5. Jugar con las funciones de Class!
   Resolver el background, resolver perturbaciones, CMB, etc.

6. Limpiar las estructuras  con “struct_cleanup”

7. Borrar parametros con “empty”

8. Empezar con un nuevo caso en 2.

DOS OPCIONES:

1. Crear un script de .py
2. Crear un notebook de jupyter