Visualising the output

Thomas Tram

Institute of Gravitation and Cosmology, Portsmouth

October 27, 2014

Outline

Some prerequisites

try.init

```
output = tCl
write background = yes
```

Adding verbose flags

Trick to add all `verbose_xxx = 1` flags:

```
tail explanatory.ini >> try.ini
```

Now run `./class try.ini` twice. (You can change try.ini slightly after the first run if you like!)

Add output path

Note that you must add `output/` in front of all filenames in this presentation!

Outline

Manual mode

Using Gnuplot

```
gnuplot> plot 'try00_cl.dat' using 1:2 with lines title 'Cl'
```

or equivalently

```
gnuplot> p 'try00_cl.dat' u 1:2 w l t 'Cl'
```

Some additional Gnuplot commands

Modifying scale and adding labels:

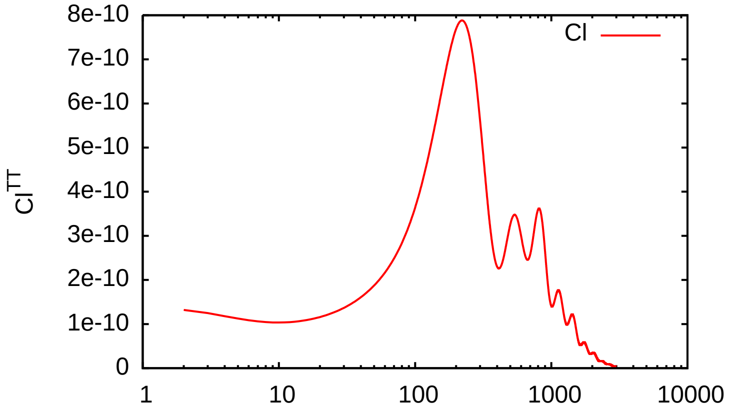
```
gnuplot> set logscale x
gnuplot> set ylabel 'Cl^{TT}'
```

Writing to a file:

```
gnuplot> set terminal pdf
gnuplot> set out 'my plot.pdf'
```

Gnuplot example output

```
gnuplot>p 'try00_cl.dat' u 1:2 w l t 'Cl'
```



Manual mode

Which column number?

Easy! Column number and title written automatically in file:

```
1:z   2:proper time [Gyr]   3:conf. time [Mpc] ...
  4:H [1/Mpc]   5:comov. dist.   6:ang.diam.dist. ...
     7:lum. dist.   8:comov.snd.hrz.   9:(.)rho_g ...
```

Other tools

Alternatives include MATLAB, Python, IDL, Mathematica, GNU Octave, ...

Outline

Automatic mode

CPU.py

- CPU: CLASS Plotting Utility
- Written in Python by BA

plot_CLASS_output.m

- Writen in MATLAB by TT
- Compatible* with GNU Octave

Automatic mode

CPU.py

Getting help:

```
python CPU.py --help
```

Plot certain quantities:

```
python CPU.py try00_background.dat -y rho_g rho_cdm
```

Plot quantities with a common string across several files:

```
python CPU.py try00_cl.dat try01_cl.dat -y E
```

Set scale: {`lin,loglog,loglin,george`}

```
python CPU.py ... --scale loglog
```

Set axis limits:

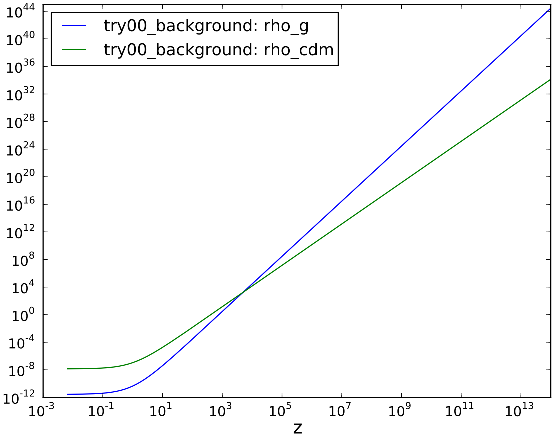
```
python CPU.py ... --xlim 0.1 10 --ylim 1e2 1e5
```

Save plot to PDF file:

```
python CPU.py ... -p
```

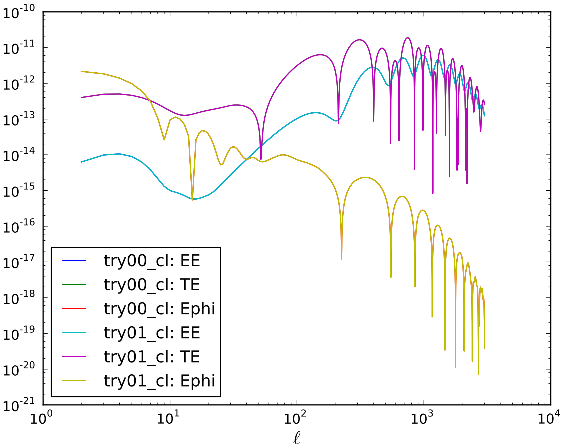
CPU example output

```
python CPU.py try00_background.dat
  -y rho_g rho_cdm --scale loglog -p
```



CPU example output

```
python CPU.py try00_cl.dat try01_cl.dat
  -y E --scale loglog -p
```



Automatic mode

plot_CLASS_output.m

Getting help:

```
help plot_CLASS_output
```

Plot certain quantities:

```
plot_C('try00_background.dat',{'rho_g','rho_cdm'})
```

Plot quantities with a common string across several files:

```
plot_C({'try00_cl.dat', 'try01_cl.dat'},'E')
```

Set xscale and yscale:

```
plot_C(...,...,'xscale','log',yscale','log')
```

Set axis limits:

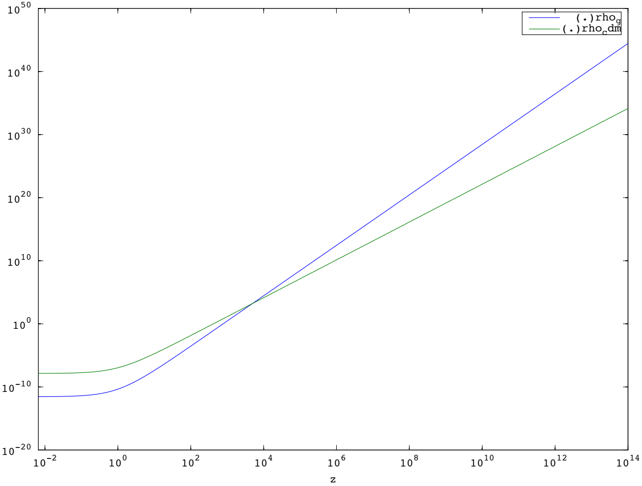
```
plot_C(...,...,'xlim',[0.1, 10])
```

Specify name of EPS file:

```
plot_C(...,...,'EpsFilename','myplot.eps')
```

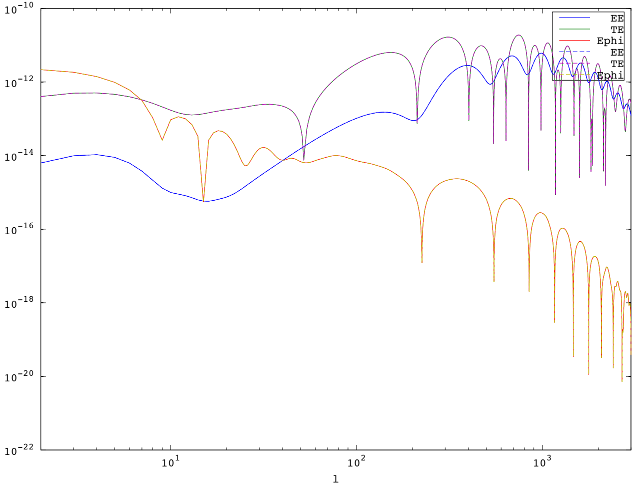
plot_CLASS_output example output

```
octave> plot_CLASS_output('try00_background.dat',
  {'rho_g','rho_cdm'}, 'xvariable', 'z')
```



plot_CLASS_output example output

```
octave> plot_CLASS_output(
  {'try00_cl.dat', 'try01_cl.dat'},'E','xscale','log')
```



Outline

Interactive mode

IPython Notebook

IPython Notebook is a Mathematica style (cell) interface to IPython.

- Has Tab-completion of variables and function names.
- Nicely presents the documentation of each function.
- Easy way to get started on Python.

Python wrapper

CLASS is called through the Python wrapper `classy`.

Interactive mode

Launching IPython Notebook

Write the following command to launch the notebook:

```
ipython notebook --pylab=inline
    --InlineBackend.figure_format=svg
```

You probably want to alias this command to e.g. `inote`.

You can open an existing notebook by

```
inote MyFirstCLASSNotebook.ipynb
```

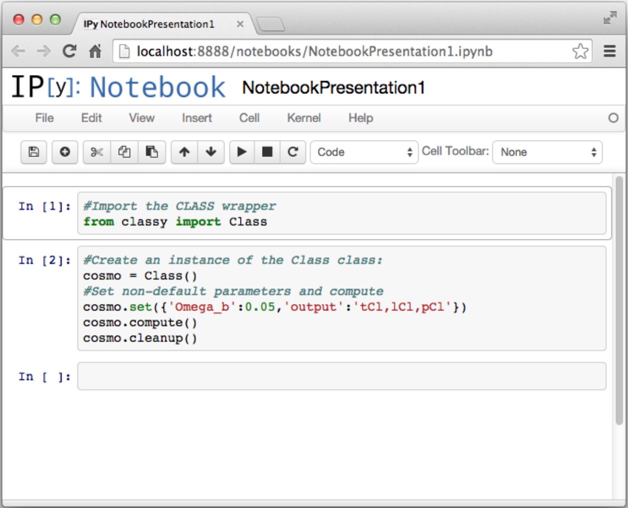
Aliasing

To make an alias, open your shell {`bash`, `zsh`, ...} startup script: {`~\.bashrc`, `~\.zshrc`, ...}

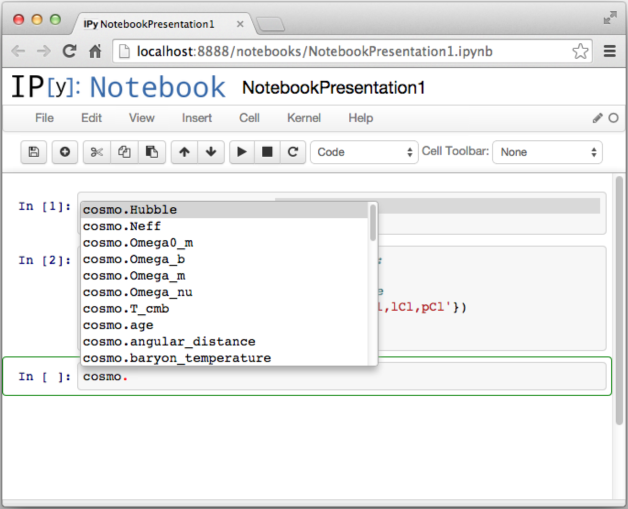
At the bottom of the file, add the line

```
alias inote="some command"
```

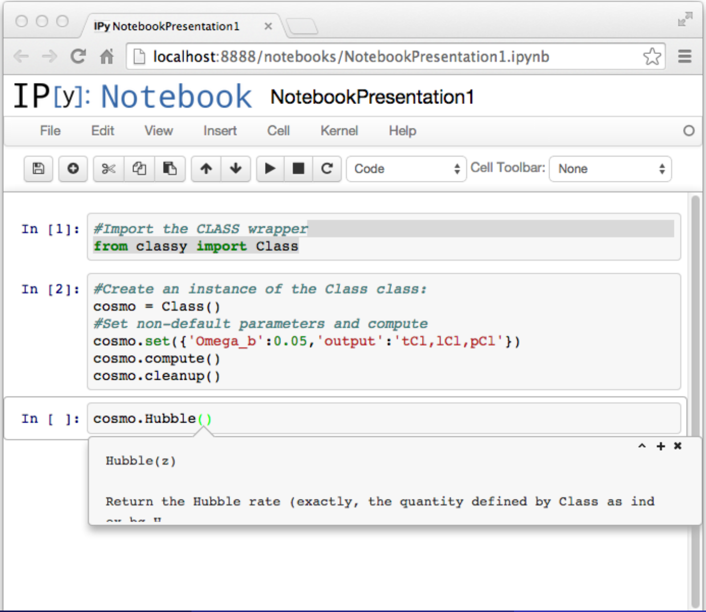
The notebook



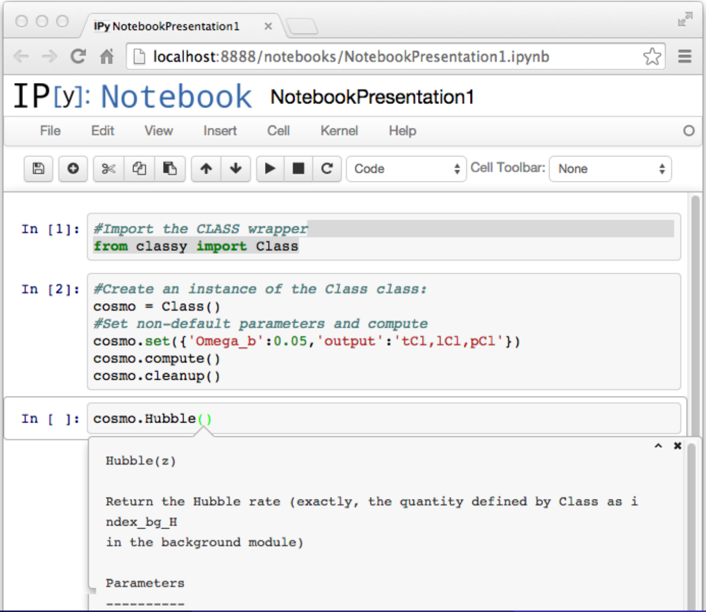
Tab: Available class methods



Shift+Tab: Help on method



Shift+Tab: More help



Summary of plotting

Three ways

- **Manual mode**: good if you already have a favourite plotting pipeline.
- **Automatic mode**: very fast and easy to get a quick look at output.
- **Interactive mode**: excellent when you need to run several models, post-proces the output or want to keep track of what you are doing.

Python wrapper

CLASS is called through the Python wrapper `classy`.

Outline

Exercises 1

All these exercises consist in running CLASS with the correct set of input parameters (cosmological parameters are unimportant), and plotting its different outputs with `CPU.py`, or `plot_CLASS_output.m`, or with your own favorite software.

1a

Check the difference between the lensed and unlensed C_l^{TT} of scalars, to see effect of smoothing of the peak contrast, and extra damping.

1b

Check the difference between the lensed and unlensed C_l^{BB} of tensors, to see that B modes are dominated by lensing at least on small scales. Use $r = 0.2$ and like in BICEP results!

Exercises I

1c

Check the difference between the unlensed C_l^{TT} of scalar modes for adiabatic and CDM isocurvature (CDI) initial conditions (with index $n_{\rm cdi} = 1$) , to check that peaks are suppressed in amplitude and shifted in scale. Do the same with NID isocurvature modes (with index $n_{\rm nid} = 1$) to check that the suppression in amplitude is less pronounced and the phase of NID and CDI are different.

1d

Check the difference between the linear and non-linear matter power spectrum at $z = 0$ and $z = 2$, to see that at low redshift non-linear corrections are present on larger scales.