Lecture V: Primordial fluctuations

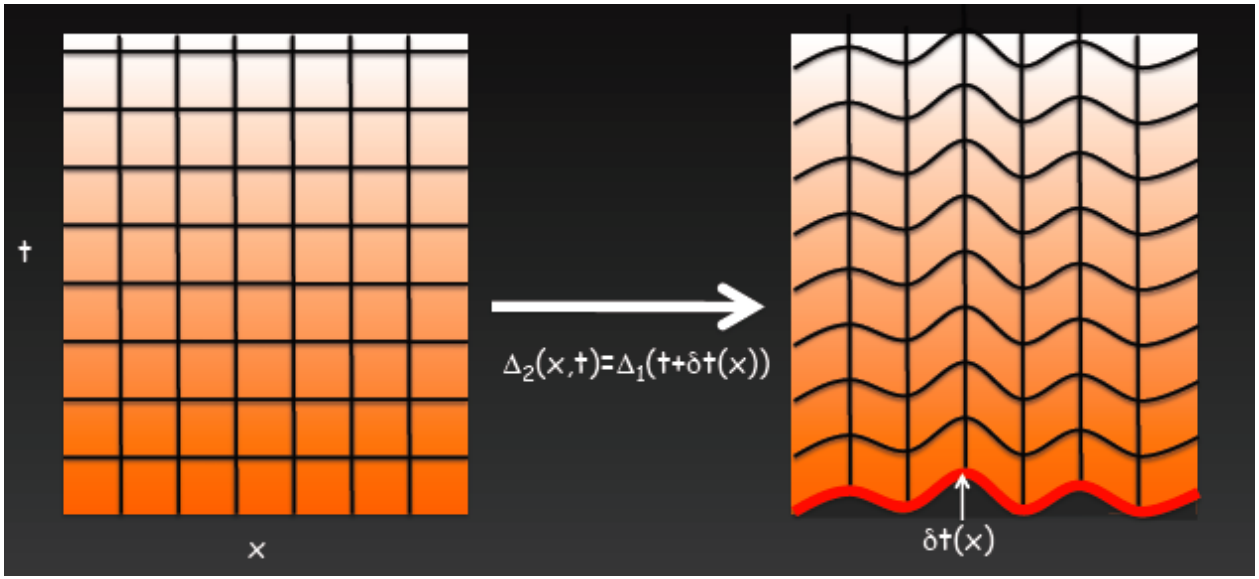
Jesús Torrado — Julien Lesgourgues

Instituut-Lorentz — EPFL & CERN

London, 15.05.2014

Adiabatic initial conditions

One way to generate perturbations = deform a homogeneous universe with a time-shifting function:



- then $\rho_i(\tau, \vec{x}) = \bar{\rho}_i(\tau + \delta\tau(\vec{x})) = \bar{\rho}_i(\tau) + \bar{\rho}_i(\tau)'\delta\tau(\vec{x})$
- implies $\delta\rho_i(\tau, \vec{x}) = -3\frac{a'}{a}(1 + w_i)\bar{\rho}_i(\tau)\delta\tau(\vec{x})$
- so $\Rightarrow \forall(i, j)$, $\frac{\delta_i}{1+w_i} = \frac{\delta_j}{1+w_j}$
- moreover $\Rightarrow \delta p_{\rm tot} = c_s^2 \delta\rho_{\rm tot}$
- hence called adiabatic initial conditions

Adiabatic initial conditions

- adiabatic perturbations are physically motivated because they correspond to perturbations generated by single degree of freedom (e.g. single inflaton)
- even when more than one perturbed d.o.f. in early universe, thermal equilibrium generally forces the system to adiabatic initial conditions.
- ways out: inflaton decaying into ever-decoupled species; inflation + topological defects; leptonic asymmetry with perturbed neutrino chemical potential...
- to get relation with metric perturbations and time evolution, need Einstein equations. Assuming

$$\forall(i,j)\ , \qquad \frac{\delta_i}{1+w_i} = \frac{\delta_j}{1+w_j}$$

one gets that for the non-decaying mode the comoving curvature perturbation $\mathcal{R} = \Psi - \frac{1}{3}\frac{\delta\rho_{\rm tot}}{\bar{\rho}_{\rm tot}+\bar{p}_{\rm tot}}$ is conserved on super-Hubble scale. Used to define the primordial spectrum,

$$\mathcal{P}_{\mathcal{R}}(k) = \frac{k^3}{2\pi^2}\left\langle \left|\mathcal{R}(\tau_{\rm ini},\vec{k})\right|^2 \right\rangle$$

Isocurvature initial conditions

- whenever perturbations are non-adiabatic, they can be expressed in some basis.
- particular basis is that of isocurvature modes, where entropy perturbations $\left(\frac{\delta_i}{1+w_i} - \frac{\delta_j}{1+w_j}\right)$ are arranged in such a way that $\mathcal{R} \longrightarrow 0$ when $k/(aH) \longrightarrow 0$.
- CDM isocurvature (`cdi`), baryon isocurvature (`bi`), neutrino density isocurvature (`nid`), neutrino velocity isocurvature (`niv`).
- in each of these, it is still true that all quantities relate to single variable $\mathcal{S}(\tau_{\rm ini}, \vec{k})$.
- transfer functions normalised to $\mathcal{S} = 1$ and primordial spectrum is

$$\mathcal{P}_{\mathcal{S}}(k) = \frac{k^3}{2\pi^2} \left\langle \left| \mathcal{S}(\tau_{\rm ini}, \vec{k}) \right|^2 \right\rangle$$

- might have correlations:

$$\mathcal{P}_{\rm cross}(k) = \frac{k^3}{2\pi^2} \left\langle \mathcal{R}(\tau_{\rm ini}, \vec{k})^* \mathcal{S}(\tau_{\rm ini}, \vec{k}) \right\rangle$$

- then

$$C_l^{TT} = 4\pi \int \frac{dk}{k} \left\{ \left(\Theta_l^{\mathcal{R}}(\tau_0, k)\right)^2 \mathcal{P}_{\mathcal{R}}(k) + \left(\Theta_l^{\mathcal{S}}(\tau_0, k)\right)^2 \mathcal{P}_{\mathcal{S}}(k) + \Theta_l^{\mathcal{R}}(\tau_0, k) \Theta_l^{\mathcal{S}}(\tau_0, k) \mathcal{P}_{\rm cross}(k) \right\}$$

The module primordial.c

External functions:

- `primordial_spectrum_at_k()` returns the primordial spectrum at a given wavenumber, interpolated in the table `ppm->lnpk`.
- `primordial_init()` computes the spectrum and fills `ppm->lnpk`.
- `primordial_free()`

Depending on the input, the module may need to return some of the following spectra:

- scalar adiabatic,
- one or several scalar isocurvature, one or several scalar cross-spectra,
- tensor.

The module primordial.c

Depends on input parameters:

```
modes = s, t
```

and if scalar modes are selected,

```
ic = ad (bi, cdi, nid, niv)
```

What the module really does depends on the input parameter:

```
P_k_ini type = analytic_Pk , two scales , inflation_V or external_Pk
```

The case analytic_Pk

The case `analytic_Pk` assumes that each of them is of the form

$$\mathcal{P}(k) = A \exp\left((n-1)\log(k/k_{\rm pivot}) + \alpha \log(k/k_{\rm pivot})^2\right)$$

The format for input parameters is (see the details, units, default values, etc. in `explanatory.ini`):

```
k_pivot = 0.05
/* exemple of scalar adiabatic parameters */
A_s = 2.3e-9
n_s = 1.
alpha_s = 0.
/* exemple of tensor parameters */
r = 1.
n_t = scc
alpha_t = scc
/* exemple of isocurvature mode parameters */
f_nid=1.
n_nid=2.
alpha_nid= 0.01

c_ad_nid = -0.5
n_ad_nid = -0.2
alpha_ad_nid = 0.002
```

The case two_scales

The case `two_scales` assumes that each spectrum is a power-law defined by an amplitude at two different scales k_1, k_2.

The format for input parameters is (see the details, units, default values, etc. in `explanatory.ini`):

```
k1=0.002
k2=0.1
/* scalar amplitudes */
P_{RR}^1 = 2.3e-9
P_{RR}^2 = 2.3e-9
/* isocurvature amplitudes
    */
P_{II}^1 = 1.e-11
P_{II}^2 = 1.e-11
P_{RI}^1 = -1.e-13
|P_{RI}^2| = 1.e-13
```

That was used in the Planck papers for getting isocurvature constraints.

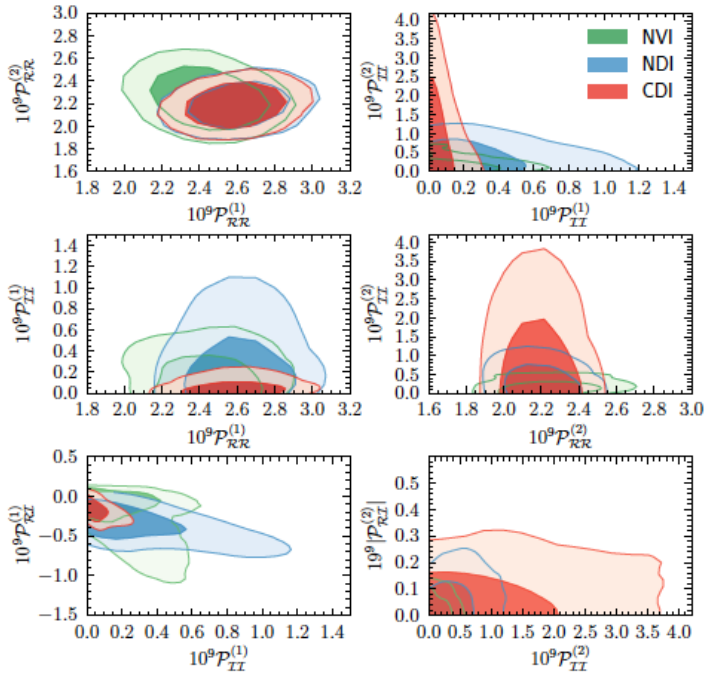


Fig. 22. Two dimensional distributions for power in isocurvature modes, using *Planck*+WP data.

The case inflation_V

The code simulates inflation and integrates the evolution of scalar/tensor perturbations around horizon crossing.

The format for input parameters is (see the details, units, default values, etc. in `explanatory.ini`):

```
potential = polynomial

V_0=1.e-13
V_1=-1.e-14
V_2=7.e-14
V_3=
V_4=
```

Used in Planck paper to reconstruct observable inflaton potential $V(\phi - \phi_*)$.

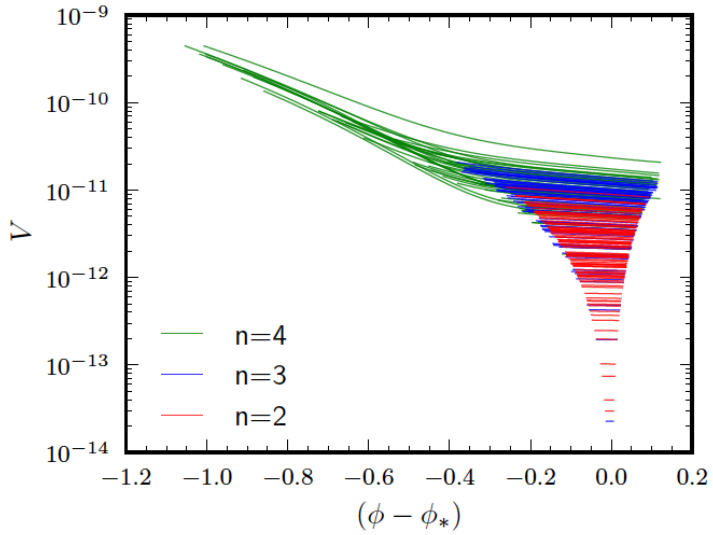


Fig. 14. Observable range of the best-fitting inflaton potentials, when $V(\phi)$ is Taylor expanded at the nth order around the pivot value ϕ_*, in natural units (where $\sqrt{8\pi}M_{\rm pl} = 1$), assuming a flat prior on ϵ_V, η_V, ξ_V^2, and ϖ_V^3, and using *Planck*+WP data.

User can easily define new classes of potentials within the function `int primordial_inflation_potential()`.

The case external_Pk

You can ask CLASS to use your own primordial spectrum.
All you need is a shell command that throws a table with columns

```
k [Mpc^-1]   |   P_s(k)   |   P_t(k) (opt)
```

This means you can either:

- Precompute the spectrum somewhere else, save it to a file `example.txt` and tell CLASS to run `command = cat example.txt`
- Write your spectrum calculator in your favourite language and ask CLASS to run it, e.g. `command = python calculate_Pk.py`

In the second case, you can specify up to 10 command line arguments for your script:

```
command       = python calculate_Pk.py
custom1       = 0.05     # k_pivot
custom2       = 2.2e-9   # A_s
custom3       = 0.96     # n_s
custom4       = 0.15     # some model-specific paramenter
```

will launch the command

```
python calculate_Pk.py 0.05 2.2e-9 0.96 0.15  [and 0's up to 10 args]
```

The `custom#` parameters can be used for sampling with Monte Python!
[Full documentation in `class/external_Pk/README.md`]

Suggested exercises

inflation_V

external_Pk