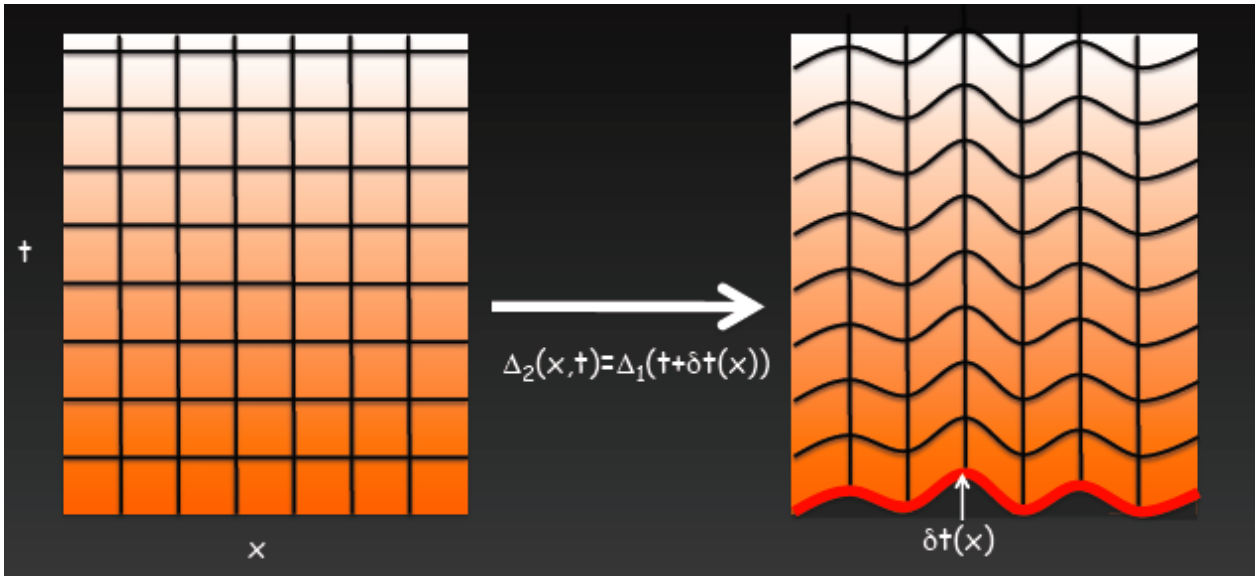
Lecture 12: the Primordial module

October 31, 2014

Adiabatic initial conditions and curvature perturbation

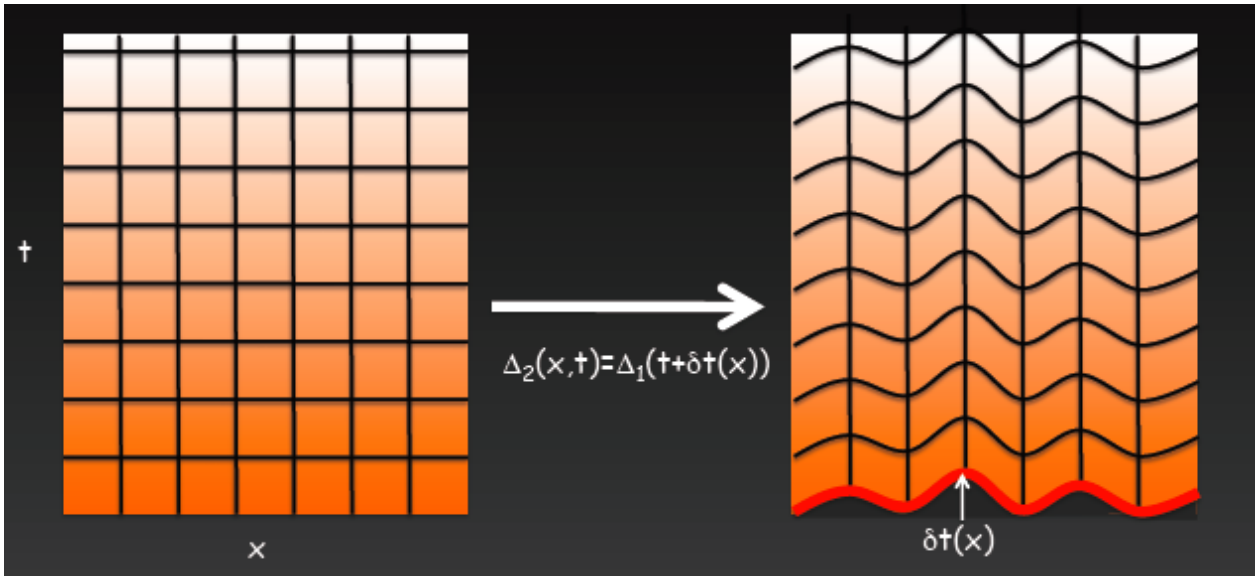
One way to generate perturbations = deform a homogeneous universe with a time-shifting function:



- physically motivated (single perturbed degree of freedom, thermal equilibrium)

Adiabatic initial conditions and curvature perturbation

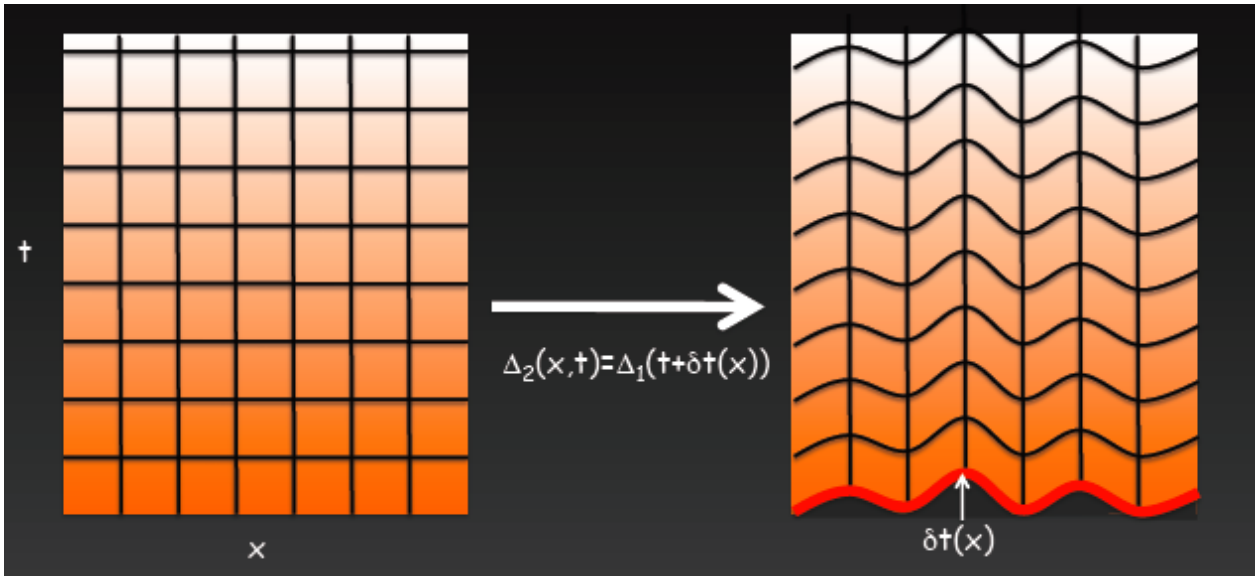
One way to generate perturbations = deform a homogeneous universe with a time-shifting function:



- physically motivated (single perturbed degree of freedom, thermal equilibrium)
- leads to $\forall(i,j)$, $\frac{\delta_i}{1+w_i} = \frac{\delta_j}{1+w_j}$, and to adiabatic sound speed

Adiabatic initial conditions and curvature perturbation

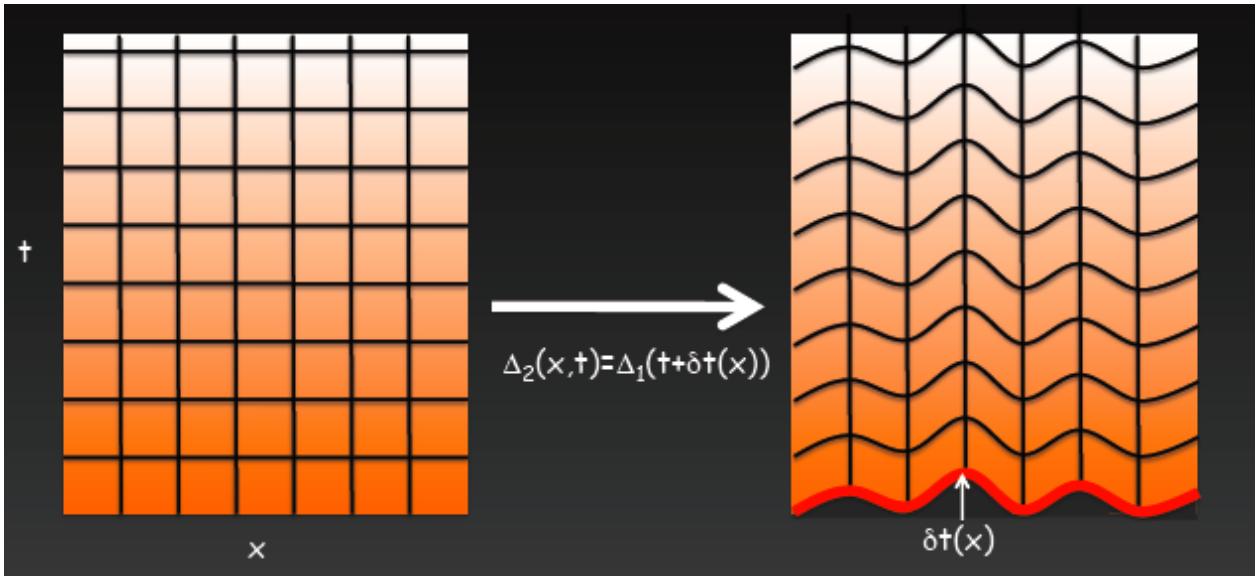
One way to generate perturbations = deform a homogeneous universe with a time-shifting function:



- physically motivated (single perturbed degree of freedom, thermal equilibrium)
- leads to $\forall(i,j)$, $\frac{\delta_i}{1+w_i} = \frac{\delta_j}{1+w_j}$, and to adiabatic sound speed
- can be characterised by initial curvature perturbation $\mathcal{R}(\tau_{\rm ini}, \vec{k})$ (constant on super-Hubble scales)

Adiabatic initial conditions and curvature perturbation

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



- physically motivated (single perturbed degree of freedom, thermal equilibrium)
- leads to $\forall(i,j)$, $\frac{\delta_i}{1+w_i} = \frac{\delta_j}{1+w_j}$, and to adiabatic sound speed
- can be characterised by initial curvature perturbation $\mathcal{R}(\tau_{\rm ini}, \vec{k})$ (constant on super-Hubble scales)
- 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.

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 $k/(aH) \longrightarrow 0$.

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`).

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 $\mathcal{S}(\tau_{\rm ini}, \vec{k})$.

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

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

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:

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

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

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

Different modes for primordial spectra

P_k_ini type =	modes =	ic =
analytic_Pk	one or several of s,t	one or several of ad,bi,cdi,nid,niv
two_scales	one or several of s,t	at most two of ad,bi,cdi,nid,niv
inflation_V	s,t	ad
inflation_H	s,t	ad
inflation_V_end	s,t	ad
external_Pk	one or several of s,t	ad

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      # either numbers or the single-field
alpha_t = scc # slow-roll consistency condition
/* exemple of isocurvature mode parameters */
f_nid=1.          # fraction of nid
n_nid=2.          # tilt of nid
alpha_nid= 0.01   # running of nid

c_ad_nid = -0.5        # correlation of ad-nid (-1 to 1)
n_ad_nid = -0.2        # tilt of ad-nid (-1 to 1)
alpha_ad_nid = 0.002   # running of ad-nid (-1 to 1)
```

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.

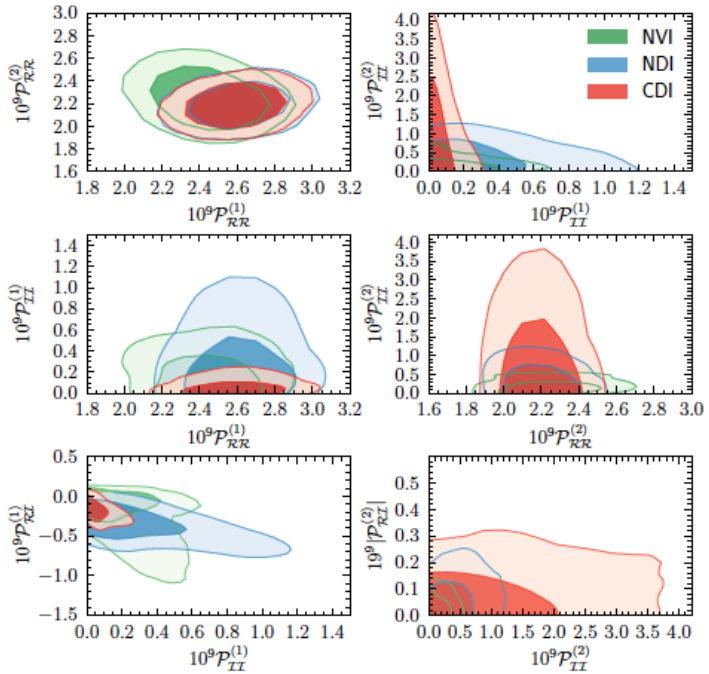


Fig. 22. Two dimensional distributions for power in isocurvature modes, using *Planck*+WP data.

The case inflation_V

For given $V(\phi - \phi_*)$, the code simulates inflation in observable range, 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
# Natural units!
V_0=1.e-13   | PSR_0= 2.18e-9
V_1=-1.e-14  | PSR_1= 0.001989
V_2=7.e-14   | PSR_2= 0.0003979
V_3=         | PSR_3=
V_4=         | PSR_4=
```

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

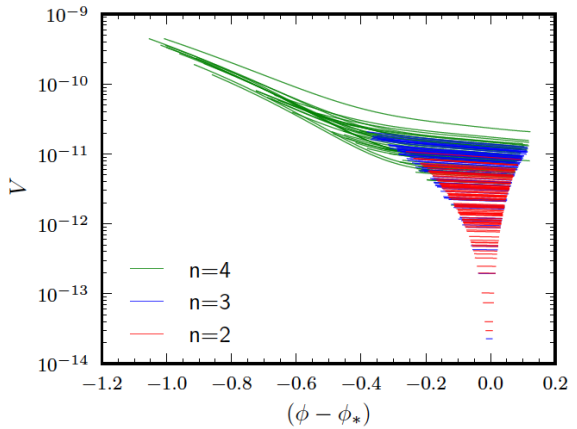


Fig. 14. Observable range of the best-fitting inflaton potentials, when $V(\phi)$ is Taylor expanded at the nth order around the pivot value ϕ_*, in natural units (where $\sqrt{8\pi}M_{\rm pl} = 1$), assuming a flat prior on ϵ_V, η_V, ξ_V^2, and ϖ_V^3, and using *Planck*+WP data.

$\phi = \phi_*$ automatically matched to $k_{\rm pivot} = aH$, no need to pass input on end of inflation/reheating

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

The case inflation_H

For given $H(\phi - \phi_*)$, the code simulates inflation in observable range, 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
# Natural units!
H_0=1.e-13   | HSR_0= 2.18e-9
H_1=-1.e-14  | HSR_1= 0.001989
H_2=7.e-14   | HSR_2= 0.0003979
H_3=         | HSR_3=
H_4=         | HSR_4=
```

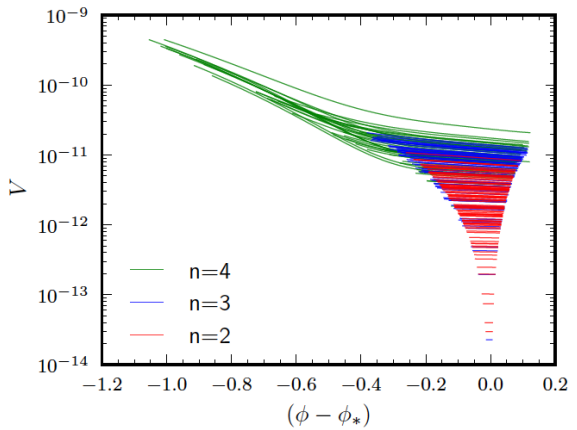


Fig. 14. Observable range of the best-fitting inflaton potentials, when $V(\phi)$ is Taylor expanded at the nth order around the pivot value ϕ_*, in natural units (where $\sqrt{8\pi}M_{\rm pl} = 1$), assuming a flat prior on ϵ_V, η_V, ξ_V^2, and ϖ_V^3, and using *Planck*+WP data.

$\phi = \phi_*$ automatically matched to $k_{\rm pivot} = aH$, no need to pass input on end of inflation/reheating

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

The case inflation_V_end

For given $V(\phi)$, the code simulates inflation in observable range AND till the end of 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`):

```
full_potential = polynomial, natural, higgs_inflation
# Natural units!
phi_end =       # only if inflation ends abruptly

Vparam0 =
Vparam1 =
Vparam2 =
Vparam3 =
Vparam4 =

ln_aH_ratio = [55 / auto]  # log (aHend/aHpivot)
N_star = 60 # log (aend/apivot)
```

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

The case inflation_V_end

Try to run with the "chaotic inflation" potential

```
$ tail explanatory.ini > test_V.ini
```

Add to this file:

```
output=tCl
modes=s,t
P_k_ini type = inflation_V_end
Vparam0 = 0
Vparam1 = 0
Vparam2 = 1.6e-12   # m2
Vparam3 = 0
Vparam4 = 0
ln_aH_ratio = 55
```

Run CLASS and check that you get reasonable values of A_s, n_s, α_s, r, ...
Increase `primordial_verbose` to 2 to get useful information: φ_{stop}, φ_{pivot}, $[\varphi_{min}, \varphi_{max}]$ of observable window...

The case external_Pk

You can ask CLASS to use your own primordial spectrum.
All you need is a shell command that spits out 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       = 0.5e-9     # some model-specific paramenter
```

will launch the command

```
python calculate_Pk.py 0.05 2.2e-9 0.96 0.5e-9  [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`]

external_Pk

Check how external_Pk works

Keep previous input file but just change one line + add one line:

```
P_k_ini type = external_Pk
command = cat external_Pk/Pk_example_w_tensors.dat
```

Run and check that A_s, n_s, α_s, r, ... sound reasonable. Have a look in the file `external_Pk/Pk_example_w_tensors.dat`.

external_Pk

Check how external_Pk works

Check on-line, on the terminal, that the following code works:

```
$ python external_Pk/generate_Pk_example_w_tensors.py 0.05
    2.2e-9 0.96 0.5e-9 0
```

Then paste it in the input file:

```
P_k_ini type = external_Pk
command=python external_Pk/generate_Pk_example_w_tensors.py
    0.05 2.2e-9 0.96 0.5e-9 0
```

and run. Finally, get that you check the same results (same A_s, n_s, α_s, r, ...) with the alternative

```
P_k_ini type = external_Pk
command=python external_Pk/generate_Pk_example_w_tensors.py
custom1 = 0.05
custom2 = 2.2e-9
custom3 = 0.96
custom4 = 0.5e-9
custom5 = 0
```