Encyclopædia Inflationaris

Christophe Ringeval
Centre for Cosmology, Particle Physics and Phenomenology
Institute of Mathematics and Physics
Louvain University, Louvain-la-Neuve, Belgium

Matsumoto, 08/2014

Outline

Introduction

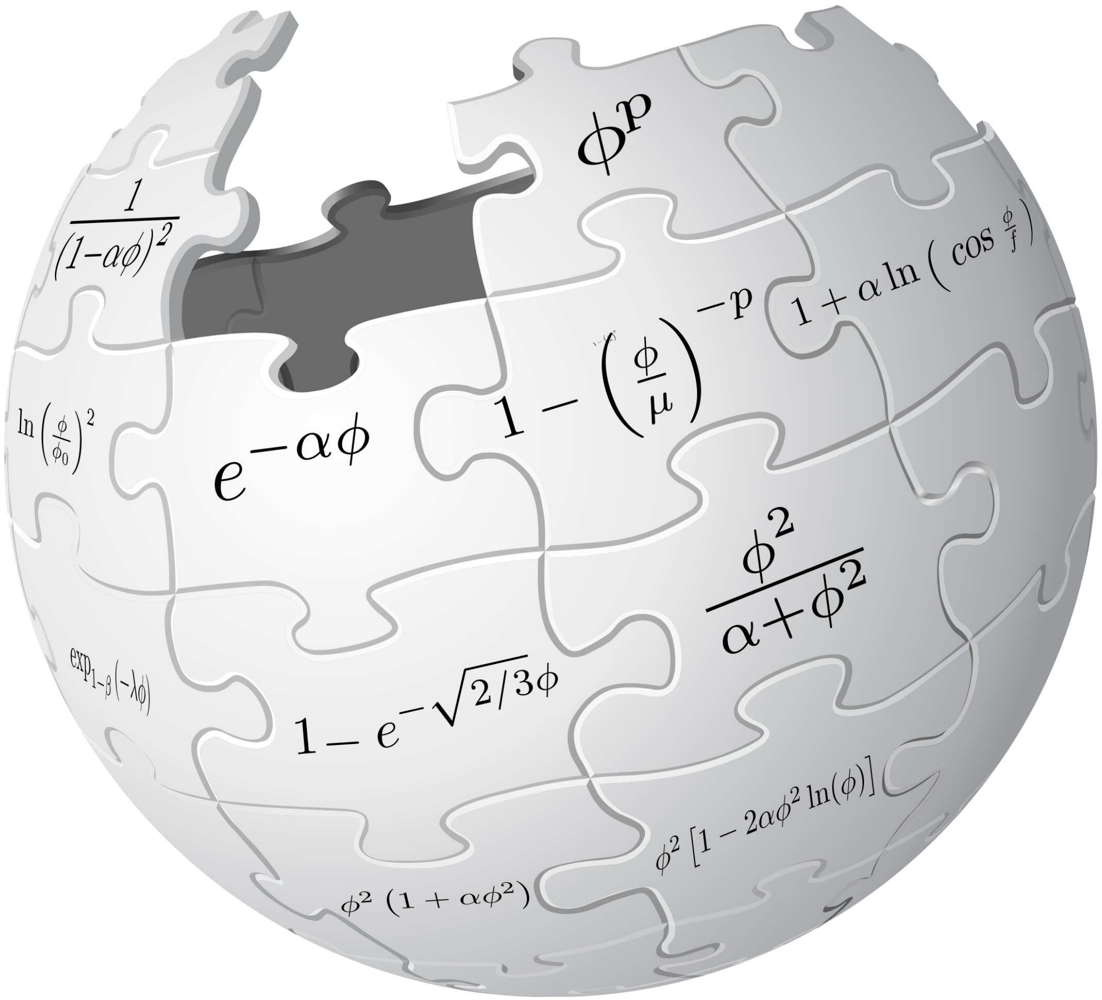
Outline

Introduction

The Encyclopædia
Purpose
The ΛCDM model of cosmology
The flatness problem
Unadressed questions within ΛCDM
The inflationary paradigm
Motivations

The Encyclopædia

- With J. Martin and V. Vennin



http://arxiv.org/abs/1303.3787
http://cp3.irmp.ucl.ac.be/~ringeval/aspic.html

Purpose

- Quasi-exhaustive analysis to derive reheating consistent observable predictions for all slow-roll single-field inflationary models
- Comes with a public code (`ASPIC`)
- Currently supports more than 50 motivated classes of potential

Name	Parameters	Sub-models	$V(\phi)$
HI	0	1	$M^4\left(1-e^{-\sqrt{2/3}\phi/M_{\rm Pl}}\right)$
RCHI	1	1	$M^4\left(1-2e^{-\sqrt{2/3}\phi/M_{\rm Pl}}+\frac{A_{\rm I}}{16\pi^2}\frac{\phi}{\sqrt{6}M_{\rm Pl}}\right)$
LFI	1	1	$M^4\left(\frac{\phi}{M_{\rm Pl}}\right)^p$
MLFI	1	1	$M^4\frac{\phi^2}{M_{\rm Pl}^2}\left[1+\alpha\frac{\phi^2}{M_{\rm Pl}^2}\right]$
RCMI	1	1	$M^4\left(\frac{\phi}{M_{\rm Pl}}\right)^2\left[1-2\alpha\frac{\phi^2}{M_{\rm Pl}^2}\ln\left(\frac{\phi}{M_{\rm Pl}}\right)\right]$
RCQI	1	1	$M^4\left(\frac{\phi}{M_{\rm Pl}}\right)^4\left[1-\alpha\ln\left(\frac{\phi}{M_{\rm Pl}}\right)\right]$
NI	1	1	$M^4\left[1+\cos\left(\frac{\phi}{f}\right)\right]$
ESI	1	1	$M^4\left(1-e^{-q\phi/M_{\rm Pl}}\right)$
PLI	1	1	$M^4e^{-\alpha\phi/M_{\rm Pl}}$
KMII	1	2	$M^4\left(1-\alpha\frac{\phi}{M_{\rm Pl}}e^{-\phi/M_{\rm Pl}}\right)$
HF1I	1	1	$M^4\left(1+A_1\frac{\phi}{M_{\rm Pl}}\right)^2\left[1-\frac{2}{3}\left(\frac{A_1}{1+A_1\phi/M_{\rm Pl}}\right)^2\right]$
CWI	1	1	$M^4\left[1+\alpha\left(\frac{\phi}{Q}\right)^4\ln\left(\frac{\phi}{Q}\right)\right]$
LI	1	2	$M^4\left[1+\alpha\ln\left(\frac{\phi}{M_{\rm Pl}}\right)\right]$
RpI	1	3	$M^4e^{-2\sqrt{2/3}\phi/M_{\rm Pl}}\left\|e^{\sqrt{2/3}\phi/M_{\rm Pl}}-1\right\|^{2p/(2p-1)}$
DWI	1	1	$M^4\left[\left(\frac{\phi}{\phi_0}\right)^2-1\right]^2$
MHI	1	1	$M^4\left[1-\mathrm{sech}\left(\frac{\phi}{\mu}\right)\right]$
RGI	1	1	$M^4\frac{(\phi/M_{\rm Pl})^2}{\alpha+(\phi/M_{\rm Pl})^2}$
MSSMI	1	1	$M^4\left[\left(\frac{\phi}{\phi_0}\right)^2-\frac{2}{3}\left(\frac{\phi}{\phi_0}\right)^6+\frac{1}{5}\left(\frac{\phi}{\phi_0}\right)^{10}\right]$
RIPI	1	1	$M^4\left[\left(\frac{\phi}{\phi_0}\right)^2-\frac{4}{3}\left(\frac{\phi}{\phi_0}\right)^3+\frac{1}{2}\left(\frac{\phi}{\phi_0}\right)^4\right]$
AI	1	1	$M^4\left[1-\frac{2}{\pi}\arctan\left(\frac{\phi}{\mu}\right)\right]$
CNAI	1	1	$M^4\left[3-\left(3+\alpha^2\right)\tanh^2\left(\frac{\alpha}{\sqrt{2}}\frac{\phi}{M_{\rm Pl}}\right)\right]$
CNBI	1	1	$M^4\left[\left(3-\alpha^2\right)\tan^2\left(\frac{\alpha}{\sqrt{2}}\frac{\phi}{M_{\rm Pl}}\right)-3\right]$
OSTI	1	1	$-M^4\left(\frac{\phi}{\phi_0}\right)^2\ln\left[\left(\frac{\phi}{\phi_0}\right)^2\right]$
WRI	1	1	$M^4\ln\left(\frac{\phi}{\phi_0}\right)^2$
SFI	2	1	$M^4\left[1-\left(\frac{\phi}{\mu}\right)^p\right]$

– 15 –

II	2	1	$M^4\left(\frac{\phi-\phi_0}{M_{\rm Pl}}\right)^{-\beta}-M^4\frac{\beta^2}{6}\left(\frac{\phi-\phi_0}{M_{\rm Pl}}\right)^{-\beta-2}$
KMIII	2	1	$M^4\left[1-\alpha\frac{\phi}{M_{\rm Pl}}\exp\left(-\beta\frac{\phi}{M_{\rm Pl}}\right)\right]$
LMI	2	2	$M^4\left(\frac{\phi}{M_{\rm Pl}}\right)^\alpha\exp\left[-\beta(\phi/M_{\rm Pl})^\gamma\right]$
TWI	2	1	$M^4\left[1-A\left(\frac{\phi}{\phi_0}\right)^2e^{-\phi/\phi_0}\right]$
GMSSMI	2	2	$M^4\left[\left(\frac{\phi}{\phi_0}\right)^2-\frac{2}{3}\alpha\left(\frac{\phi}{\phi_0}\right)^6+\frac{\alpha}{5}\left(\frac{\phi}{\phi_0}\right)^{10}\right]$
GRIPI	2	2	$M^4\left[\left(\frac{\phi}{\phi_0}\right)^2-\frac{4}{3}\alpha\left(\frac{\phi}{\phi_0}\right)^3+\frac{\alpha}{2}\left(\frac{\phi}{\phi_0}\right)^4\right]$
BSUSYBI	2	1	$M^4\left(e^{\sqrt{6}\frac{\phi}{M_{\rm Pl}}}+e^{\sqrt{6}\gamma\frac{\phi}{M_{\rm Pl}}}\right)$
TI	2	3	$M^4\left(1+\cos\frac{\phi}{\mu}+\alpha\sin^2\frac{\phi}{\mu}\right)$
BEI	2	1	$M^4\exp_{1-\beta}\left(-\lambda\frac{\phi}{M_{\rm Pl}}\right)$
PSNI	2	1	$M^4\left[1+\alpha\ln\left(\cos\frac{\phi}{f}\right)\right]$
NCKI	2	2	$M^4\left[1+\alpha\ln\left(\frac{\phi}{M_{\rm Pl}}\right)+\beta\left(\frac{\phi}{M_{\rm Pl}}\right)^2\right]$
CSI	2	1	$\frac{M^4}{\left(1-\alpha\frac{\phi}{M_{\rm Pl}}\right)^2}$
OI	2	1	$M^4\left(\frac{\phi}{\phi_0}\right)^4\left[\left(\ln\frac{\phi}{\phi_0}\right)^2-\alpha\right]$
CNCI	2	1	$M^4\left[\left(3+\alpha^2\right)\coth^2\left(\frac{\alpha}{\sqrt{2}}\frac{\phi}{M_{\rm Pl}}\right)-3\right]$
SBI	2	2	$M^4\left\{1+\left[-\alpha+\beta\ln\left(\frac{\phi}{M_{\rm Pl}}\right)\right]\left(\frac{\phi}{M_{\rm Pl}}\right)^4\right\}$
SSBI	2	6	$M^4\left[1+\alpha\left(\frac{\phi}{M_{\rm Pl}}\right)^2+\beta\left(\frac{\phi}{M_{\rm Pl}}\right)^4\right]$
IMI	2	1	$M^4\left(\frac{\phi}{M_{\rm Pl}}\right)^{-p}$
BI	2	2	$M^4\left[1-\left(\frac{\phi}{\mu}\right)^{-p}\right]$
RMI	3	4	$M^4\left[1-\frac{c}{2}\left(-\frac{1}{2}+\ln\frac{\phi}{\phi_0}\right)\frac{\phi^2}{M_{\rm Pl}^2}\right]$
VHI	3	1	$M^4\left[1+\left(\frac{\phi}{\mu}\right)^p\right]$
DSI	3	1	$M^4\left[1+\left(\frac{\phi}{\mu}\right)^{-p}\right]$
GMLFI	3	1	$M^4\left(\frac{\phi}{M_{\rm Pl}}\right)^p\left[1+\alpha\left(\frac{\phi}{M_{\rm Pl}}\right)^q\right]$
LPI	3	3	$M^4\left(\frac{\phi}{\phi_0}\right)^p\left(\ln\frac{\phi}{\phi_0}\right)^q$
CNDI	3	3	$\frac{M^4}{\left\{1+\beta\cos\left[\alpha\left(\frac{\phi-\phi_0}{M_{\rm Pl}}\right)\right]\right\}^2}$

– 16 –

The ΛCDM model of cosmology

- Homogeneous + isotropic Friedmann–Lemaître scenario

$$\mathrm{d}s^2 = -\mathrm{d}t^2 + a^2(t)\gamma_{ij}\mathrm{d}x^i\mathrm{d}x^j, \quad H(t) = \frac{\mathrm{d}\ln a}{\mathrm{d}t}$$

  - Gravitation: $G_{\mu\nu} + \Lambda g_{\mu\nu} = \kappa^2 T_{\mu\nu}$
  - Contains: cold dark matter, baryons, photons

$$\rho_{\mathrm{mat}} = (\Omega_{\mathrm{dm}} + \Omega_{\mathrm{b}})\frac{\rho_{\mathrm{cri}}}{a^3}, \quad \rho_{\mathrm{rad}} = \Omega_{\mathrm{rad}}\frac{\rho_{\mathrm{cri}}}{a^4}, \quad \rho_{\mathrm{cri}} = 3\kappa^{-2}{H_0}^2$$

- Plus linear perturbations: origin of CMB and galaxies
  - Need some initial conditions

$$\langle X^*(\boldsymbol{k}, t_{\mathrm{ini}})X(\boldsymbol{k}', t_{\mathrm{ini}})\rangle = (2\pi)^3 P_X(k)\delta(\boldsymbol{k} - \boldsymbol{k}')$$

  - A priori as many $P_X(k)$ as species are required!

The flatness problem

- Evolution of the curvature
  - ✦ Friedmann-Lemaître equations for a perfect fluid

$$\left.\begin{aligned} T_{\mu\nu} &= (\rho+P)u_\mu u_\nu - P g_{\mu\nu} \\ \gamma_{ij}\mathrm{d}x^i\mathrm{d}x^j &= \frac{\mathrm{d}r^2}{1-\mathcal{K}^2r^2} + r^2\mathrm{d}\Omega^2 \end{aligned}\right\} \Rightarrow \left\{\begin{aligned} H^2 &= \kappa^2\frac{\rho}{3} - \frac{\mathcal{K}}{a^2} \\ \frac{\ddot{a}}{a} &= -\frac{\kappa^2}{6}(\rho+3P) \end{aligned}\right.$$

  - ✦ Curvature density parameter: $\Omega_{\mathrm{K}} \equiv -\dfrac{\mathcal{K}}{a^2H^2}$

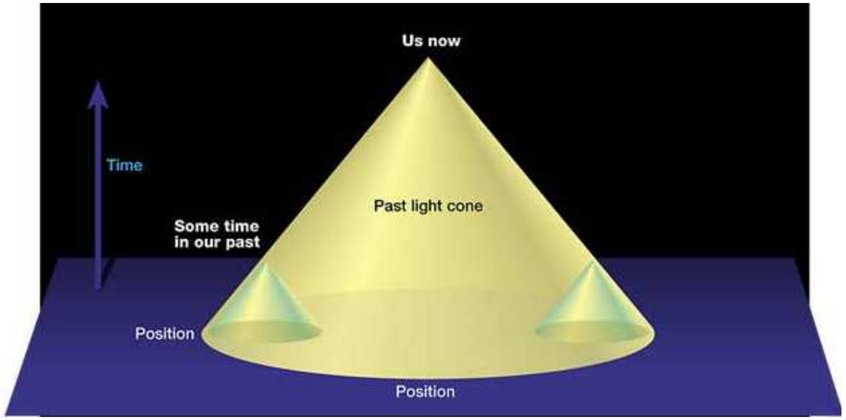
$$\omega \equiv \frac{\Omega_{\mathrm{K}}}{1-\Omega_{\mathrm{K}}} \quad \Rightarrow \quad \frac{\mathrm{d}\ln\omega}{\mathrm{d}\ln a} = 1 + 3\frac{P}{\rho}$$

  - ✦ For a constant equation of state $P = w\rho \Rightarrow \omega \propto a^{1+3w}$

- Flatness is instable during matter $(w=0)$ and radiation $(w=1/3)$ eras

Unadressed questions within ΛCDM

- We see causally disconnected regions from the past at any time
  - ✦ Distance to the particle horizon: $d_\mathrm{h} = a(t)\int_0^t \frac{\mathrm{d}t'}{a(t')} = a(\eta)\eta \propto t$
  - ✦ $(\eta_0/\eta_\mathrm{CMB})^3 \simeq 10^5$ causally disconnected patches: CMB?



- Acausal initial conditions for structure formation

$$\lambda \propto a(t) \propto t^{2/(3+3w)} \Rightarrow \lambda_\mathrm{ini} > d_\mathrm{h}(t_\mathrm{ini})$$

- Monopole problem: $\pi_2(G/H) \neq 1$ for $U(1) \subset H$

The inflationary paradigm

- Proposed in the 80's to solve these issues

[Grishchuk ,Starobinsky, Sato, Guth, Linde, Albrecht, Steinhardt, Sasaki, Mukhanov]

- Flatness, horizon and monopole problems solved for $w < -1/3$

inflation = accelerated expansion of the scale factor

  - Quasi de Sitter: $w \simeq -1 \Rightarrow H$ is constant $\Rightarrow a(t) \propto \mathrm{e}^{Ht}$

$$\frac{d_\mathrm{h}(t_\mathrm{end})}{d_\mathrm{h}(t_\mathrm{ini})} \simeq \mathrm{e}^{H\Delta t} > \frac{\eta_0}{\eta_\mathrm{Pl}} \simeq 10^{28} \Leftarrow N = H\Delta t \gtrsim 60$$

- Isotropy: Bianchi smoothed out during inflation ($\rightarrow$ FLRW)

$$H^2 = \kappa^2 \frac{\rho}{3} - f(a_x, a_y, a_z), \quad f \lesssim \frac{1}{(a_x a_y a_z)^{2/3}}$$

- Structure formation from quantum fluctuations

Motivations from current observations

- Planck 2013 measurements in favour of inflation
  - ✦ Flatness ($\Omega_\mathrm{K} = 0$) is instable during decelerated expansion

$$\Omega_\mathrm{K} = 1 - \Omega_\mathrm{dm} - \Omega_\mathrm{b} - \Omega_\Lambda - \Omega_\mathrm{rad} = 0.000^{+0.0066}_{-0.0067} \quad \texttt{(PLANCK+WP+BAO)}$$

  - ✦ Adiabatic initial conditions: isocurvature modes are constrained

$$\forall X \quad P_X(k) = P(k)$$

  - ✦ Quasi scale invariance

$$k^3 P(k) = A \left(\frac{k}{k_*}\right)^{n_\mathrm{S}-1} \Rightarrow n_\mathrm{S} = 0.9619 \pm 0.0073$$

  - ✦ Dark energy? [CR, Suyama, Takahashi, Yamaguchi, Yokoyama]
  - ✦ Gaussianity of the CMB anisotropies

$$f_\mathrm{NL}^\mathrm{loc} = 2.7 \pm 5.8, \quad f_\mathrm{NL}^\mathrm{eq} = -42 \pm 75, \quad f_\mathrm{NL}^\mathrm{ortho} = -25 \pm 39$$

- The simplest framework: single-field inflation
  - ✦ Makes extra-predictions: $f_\mathrm{NL}^\mathrm{loc} = \mathcal{O}(n_\mathrm{S} - 1)$ and $\exists r > 0$

Slow-roll inflation

Outline

Basic theoretical assumptions

- Dynamics given by ($\kappa^2 = 1/M_{\mathrm{P}}^2$)

$$S = \int \mathrm{d}x^4 \sqrt{-g} \left[\frac{1}{2\kappa^2} R + \mathcal{L}(\phi) \right] \quad \text{with} \quad \mathcal{L}(\phi) = -\frac{1}{2} g^{\mu\nu} \partial_\mu \phi \partial_\nu \phi - V(\phi)$$

- Can be used to describe:
  - ✦ Minimally coupled scalar field to General Relativity
  - ✦ Scalar-tensor theory of gravitation in the Einstein frame

    the graviton' scalar partner is also the inflaton (HI, RPI1,...)

- Everything is consistently solved in the slow-roll approximation
  - ✦ Background evolution $\phi(t)$ (attractor)
  - ✦ Linear perturbations for the field-metric system $\zeta(t, \boldsymbol{x})$, $\delta\phi(t, \boldsymbol{x})$

- Inclusion of the reheating era at the background level
  - ✦ A new parameter R_{rad}

Self-gravitating scalar field

- Stress tensor for a homogeneous scalar field in a flat FLRW metric

$$T_{\mu\nu} \equiv \frac{2}{\sqrt{|g|}} \frac{\delta S_\phi}{\delta g^{\mu\nu}} = \partial_\mu \phi \partial_\nu \phi - g_{\mu\nu} \left[\frac{1}{2} (\nabla\phi)^2 - V(\phi) \right]$$

$$\phi(x^\mu) = \phi(t) \Rightarrow \begin{cases} \rho = \frac{1}{2}\dot{\phi}^2 + V(\phi) \\ P = \frac{1}{2}\dot{\phi}^2 - V(\phi) \end{cases}$$

✦ Potential dominated regime: $P \simeq -\rho \Rightarrow w \simeq -1 \Rightarrow \ddot{a} > 0$

- Friedmann-Lemaître equations: $\delta S / \delta g^{\mu\nu} = 0$

$$3H^2 = \kappa^2 \left(\frac{1}{2}\dot{\phi}^2 + V \right), \qquad \dot{H} = -\frac{\kappa^2}{2}\dot{\phi}^2$$

- Klein-Gordon equation: $\delta S / \delta \phi = 0$

$$\ddot{\phi} + 3H\dot{\phi} + V_{,\phi} = 0$$

Decoupling field and space-time evolution

- Time measured in e-fold: $N \equiv \ln a$
- Deviations from de-Sitter measured by Hubble flow hierarchy [Schwarz 01]

$$\epsilon_0 = \frac{H_{\mathrm{ini}}}{H}, \qquad \epsilon_{i+1} = \frac{\ln|\epsilon_i|}{\mathrm{d}N}$$

- Friedmann-Lemaître equations in e-fold time (with $M_{\mathrm{P}}^2 = 1$)

$$\begin{cases} H^2 = \dfrac{1}{3}\left(\dfrac{1}{2}\dot{\phi}^2 + V\right) \\ \dfrac{\ddot{a}}{a} = -\dfrac{1}{3}\left(\dot{\phi}^2 - V\right) \end{cases} \Rightarrow \begin{cases} H^2 = \dfrac{V}{3 - \dfrac{1}{2}\left(\dfrac{\mathrm{d}\phi}{\mathrm{d}N}\right)^2} \\ -\dfrac{\mathrm{d}\ln H}{\mathrm{d}N} = \dfrac{1}{2}\left(\dfrac{\mathrm{d}\phi}{\mathrm{d}N}\right)^2 \end{cases} \Leftrightarrow \begin{cases} H^2 = \dfrac{V}{3 - \epsilon_1} \\ \epsilon_1 = \dfrac{1}{2}\left(\dfrac{\mathrm{d}\phi}{\mathrm{d}N}\right)^2 \end{cases}$$

- Klein-Gordon equation in e-folds: relativistic kinematics with friction

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}N^2} + \left(3 + \frac{\mathrm{d}\ln H}{\mathrm{d}N}\right)\frac{\mathrm{d}\phi}{\mathrm{d}N} + \frac{V_{,\phi}}{H^2} = 0 \quad \Rightarrow \quad \frac{1}{3-\epsilon_1}\frac{\mathrm{d}^2\phi}{\mathrm{d}N^2} + \frac{\mathrm{d}\phi}{\mathrm{d}N} = -\frac{\mathrm{d}\ln V}{\mathrm{d}\phi}$$

Background evolution

- The friction term ensures the existence of a "terminal velocity"

$$\frac{\mathrm{d}\phi}{\mathrm{d}N} = -\frac{\mathrm{d}\ln V}{\mathrm{d}\phi} \quad \Rightarrow \quad \epsilon_1 \simeq \frac{1}{2}\left(\frac{\mathrm{d}\ln V}{\mathrm{d}\phi}\right)^2, \quad \epsilon_2 \simeq 2\left[\left(\frac{V_{,\phi}}{V}\right)^2 - \frac{V_{,\phi\phi}}{V}\right]\dots$$

  - ✦ As for a "sky diver" it does not depend on the initial conditions
  - ✦ Inflation occurs for $\epsilon_1 < 1 \Leftarrow \ln[V(\phi)]$ should be flat enough

$$\epsilon_1 = -\frac{\ln H}{\mathrm{d}N} = -\frac{\dot{H}}{H^2} = 1 - \frac{1}{H^2}\frac{\ddot{a}}{a}$$

- Deviations from terminal velocity behaviour are encoded in ϵ_2

$$\epsilon_2 = \frac{\mathrm{d}\ln\epsilon_1}{\mathrm{d}N} \Rightarrow \frac{\mathrm{d}^2\phi}{\mathrm{d}N^2} = \frac{\epsilon_2}{2}\frac{\mathrm{d}\phi}{\mathrm{d}N}$$

  - ✦ Klein-Gordon equation also reads

$$\frac{\mathrm{d}\phi}{\mathrm{d}N} = -\frac{3-\epsilon_1}{3-\epsilon_1+\frac{\epsilon_2}{2}}\frac{\mathrm{d}\ln V}{\mathrm{d}\phi}$$

Slow-roll approximation

- Assume that all $\epsilon_i = \mathcal{O}(\epsilon)$ and $\epsilon_i < 1$
  - ✦ The trajectory can be solved for N (taking $N_{\rm ini} = 0$)

$$N = \mathcal{I}(\phi_{\rm ini}) - \mathcal{I}(\phi) \quad \text{with} \quad \mathcal{I}(\phi) \equiv \int^{\phi} \frac{V(\psi)}{V_{,\psi}(\psi)} \mathrm{d}\psi$$

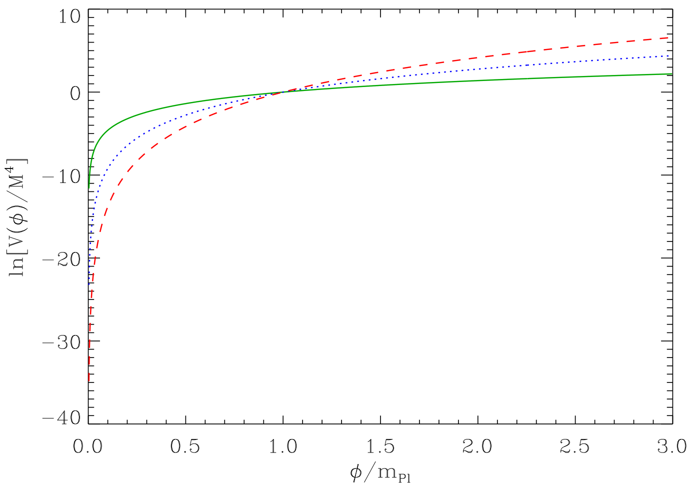
  - ✦ In terms of the field values at the end of inflation

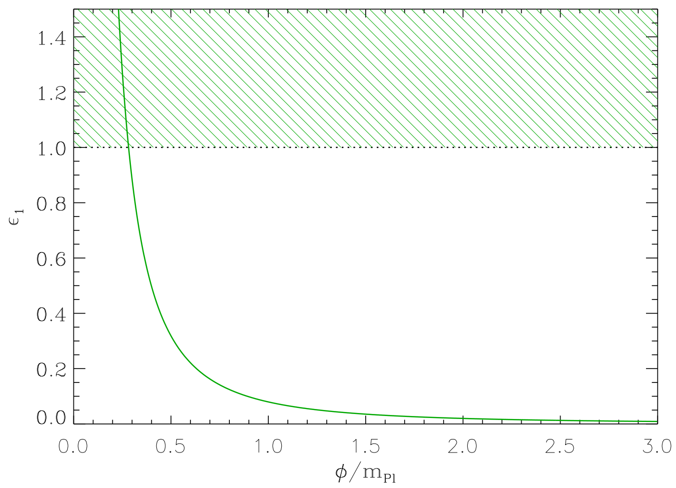
$$N - N_{\rm end} = \mathcal{I}(\phi_{\rm end}) - \mathcal{I}(\phi)$$

- The end of inflation
  - ✦ Inflation naturally ends when $\epsilon_1 > 1$: $\phi_{\rm end}$ is solution of the algebraic equation $\epsilon_1(\phi_{\rm end}) = 1$
  - ✦ Or, there is another mechanism ending inflation (tachyonic instability) and $\phi_{\rm end}$ is a model parameter that must be specified
- The reheating stage: everything after $N_{\rm end}$ till radiation domination

Example: large field inflation

- LFI potential has two-parameters: $V(\phi) = M^4\phi^p$
  - ✦ M^4 do not affect the background evolution (see KG)





- Field trajectory

$$\mathcal{I}(\phi) = \int^{\phi} \frac{M^4\psi^p}{M^4 p \psi^{p-1}}\,\mathrm{d}\psi = \frac{\phi^2}{2p} \quad \Rightarrow \quad N - N_{\mathrm{end}} = \frac{1}{2p}\left(\phi^2_{\mathrm{end}} - \phi^2\right)$$

- End of inflation at $\epsilon_1(\phi_{\mathrm{end}}) \simeq \dfrac{p^2}{2\phi^2_{\mathrm{end}}} = 1$

$$\phi_{\mathrm{end}} \simeq \frac{p}{\sqrt{2}} \quad \Rightarrow \quad \phi(N) = \sqrt{2p(N_{\mathrm{end}} - N) + \frac{p^2}{2}} \qquad (\phi > 1)$$

Background quantities in ASPIC

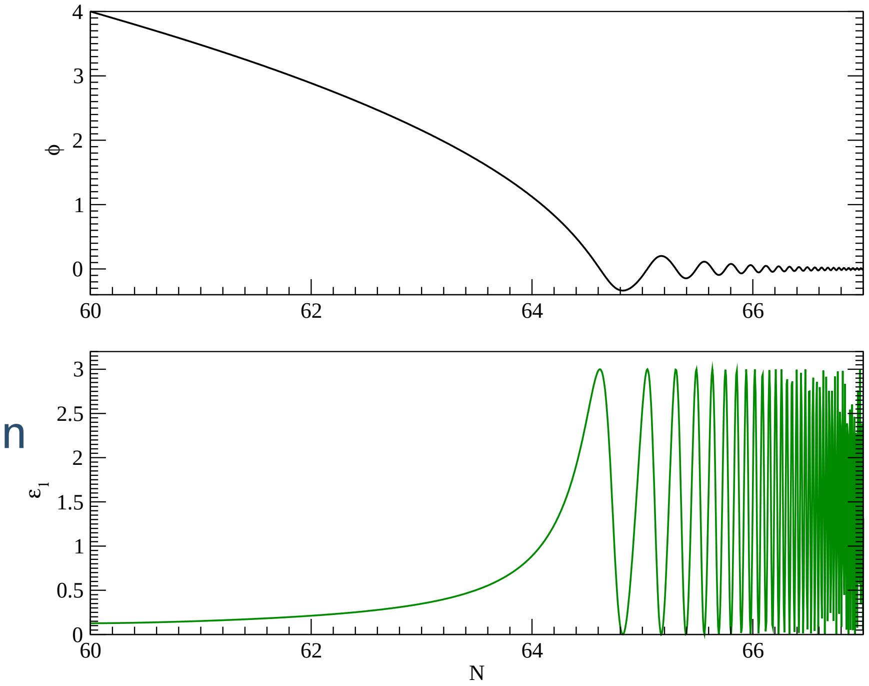
- For all *Encyclopædia Inflationaris* models, all background quantities are coded in ASPIC
- Example LFI: from the input of field value and potential parameters

```
!slow-roll functions for the large field potential
!
!V(phi) = M^4 x^p
!
!x = phi/Mp

module lfisr
  use infprec, only : kp
  implicit none

  private

  public  lfi_norm_potential, lfi_epsilon_one, lfi_epsilon_two
  public  lfi_epsilon_three
  public  lfi_x_endinf, lfi_efold_primitive, lfi_x_trajectory
  public  lfi_norm_deriv_potential, lfi_norm_deriv_second_potential

contains
!returns V/M^4
  function lfi_norm_potential(x,p)
    implicit none
    real(kp) :: lfi_norm_potential
    real(kp), intent(in) :: x,p

    lfi_norm_potential = x**p

  end function lfi_norm_potential


!returns the first derivative of the potential with respect to x, divided by M^4
  function lfi_norm_deriv_potential(x,p)
    implicit none
    real(kp) :: lfi_norm_deriv_potential
    real(kp), intent(in) :: x,p

   lfi_norm_deriv_potential = p*x**(p-1._kp)

  end function lfi_norm_deriv_potential



!returns the second derivative of the potential with respect to x, divided by M^
4
  function lfi_norm_deriv_second_potential(x,p)
    implicit none
    real(kp) :: lfi_norm_deriv_second_potential
    real(kp), intent(in) :: x,p

    lfi_norm_deriv_second_potential =  p*(p-1._kp)*x**(p-2._kp)

  end function lfi_norm_deriv_second_potential


!epsilon_one(x)
  function lfi_epsilon_one(x,p)
    implicit none
    real(kp) :: lfi_epsilon_one
    real(kp), intent(in) :: x,p

    lfi_epsilon_one = 0.5_kp*(p/x)**2

  end function lfi_epsilon_one


!epsilon_two(x)
  function lfi_epsilon_two(x,p)
    implicit none
    real(kp) :: lfi_epsilon_two
    real(kp), intent(in) :: x,p

    lfi_epsilon_two = 2._kp*p/x**2

  end function lfi_epsilon_two


!epsilon_three(x)
  function lfi_epsilon_three(x,p)
    implicit none
    real(kp) :: lfi_epsilon_three
    real(kp), intent(in) :: x,p

    lfi_epsilon_three = lfi_epsilon_two(x,p)

  end function lfi_epsilon_three


!returns x at the end of inflation defined as epsilon1=1
  function lfi_x_endinf(p)
    implicit none
    real(kp), intent(in) :: p
    real(kp) :: lfi_x_endinf

    lfi_x_endinf = p/sqrt(2._kp)

  end function lfi_x_endinf


!this is integral[V(phi)/V'(phi) dphi]
  function lfi_efold_primitive(x,p)
    implicit none
    real(kp), intent(in) :: x,p
    real(kp) :: lfi_efold_primitive

    if (p.eq.0._kp) stop 'lfi_efold_primitive: p=0!'

    lfi_efold_primitive = 0.5_kp*x**2/p

  end function lfi_efold_primitive


!returns x at bfold=-efolds before the end of inflation, ie N-Nend
  function lfi_x_trajectory(bfold,xend,p)
    implicit none
    real(kp), intent(in) :: bfold, p, xend
    real(kp) :: lfi_x_trajectory

    lfi_x_trajectory = sqrt(-2._kp*p*bfold + xend**2)


  end function lfi_x_trajectory


end module lfisr
```

Reheating: from inflation to radiation

- Example $V = M^4\phi^2$ (LFI_2)
  - ✦ Harmonic oscillator ($\omega \gg H$)

$$\left\langle \frac{1}{2}\phi_{,t}^2 \right\rangle = \langle V \rangle \Rightarrow \begin{cases} \langle \rho \rangle = \langle \phi_{,t}^2 \rangle \\ \langle P \rangle = 0 \end{cases}$$



  - ✦ Coherent oscillations: $\phi \to$ radiation (non pert. decay)
  - ✦ Last $\Delta N_{\mathrm{reh}} \equiv N_{\mathrm{reh}} - N_{\mathrm{end}}$ e-folds

- Total energy density at the end of reheating ρ_{reh}

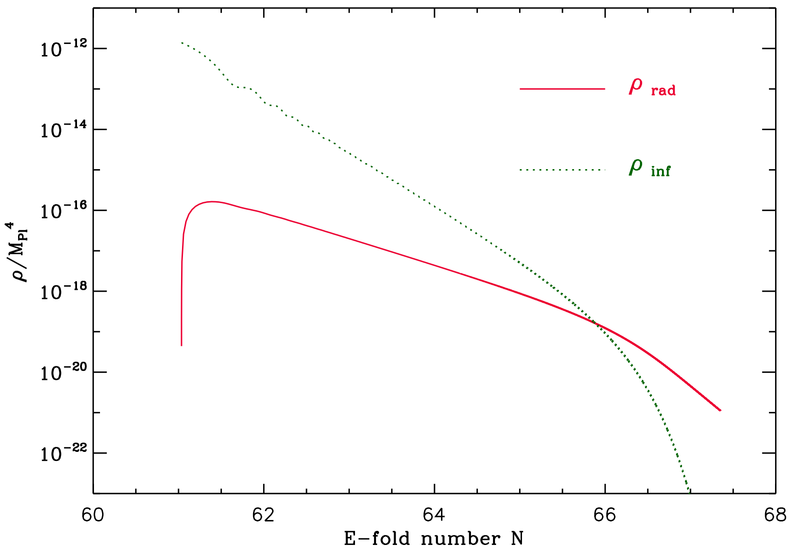
$$\begin{cases} \dot{\rho} = -3H(P+\rho) \\ \bar{w}_{\mathrm{reh}} \equiv \dfrac{1}{\Delta N_{\mathrm{reh}}} \displaystyle\int_{N_{\mathrm{end}}}^{N_{\mathrm{reh}}} \frac{P(N)}{\rho(N)} \mathrm{d}N \end{cases} \Rightarrow \quad \ln\left(\frac{\rho_{\mathrm{reh}}}{\rho_{\mathrm{end}}}\right) = -3(1+\bar{w}_{\mathrm{reh}})\Delta N_{\mathrm{reh}}$$

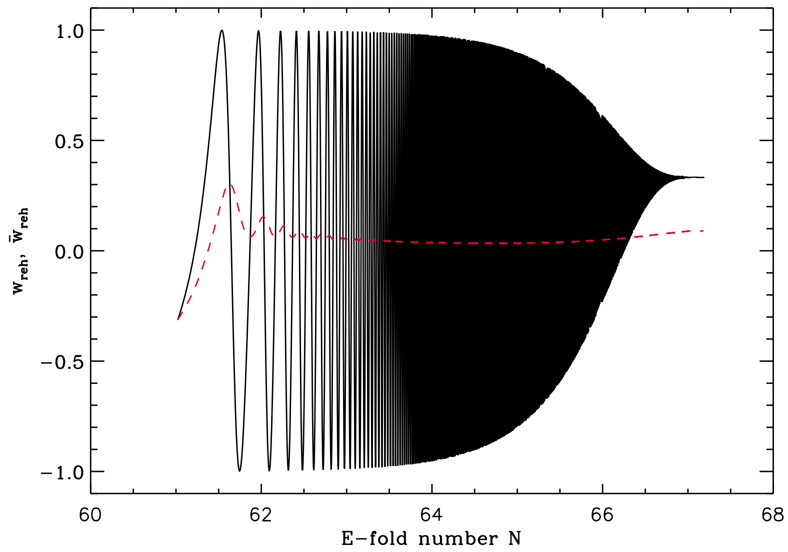
A phenomenological example

- Inflaton decay rate Γ [Turner 83]

$$\ddot{\phi} + (3H + \Gamma)\dot{\phi} + V_{,\phi} = 0, \qquad \frac{\mathrm{d}\rho_{\mathrm{rad}}}{\mathrm{d}N} + 4\rho_{\mathrm{rad}} = \frac{\Gamma}{H}\rho_{\phi}$$

- Inflaton energy is converted into radiation fluid





- At the end of reheating $\rho_{\mathrm{reh}} = \rho_{\phi}(N_{\mathrm{reh}}) + \rho_{\mathrm{rad}}(N_{\mathrm{reh}}) \simeq \rho_{\mathrm{rad}}(N_{\mathrm{reh}})$

Redshift at which reheating ends

- At $N = N_{\rm reh}$ the Universe is radiation dominated
  - ✦ If thermalized, and no extra entropy production: $a_{\rm reh}^3 s_{\rm reh} = a_0^3 s_0$

$$\begin{cases} s_{\rm reh} = q_{\rm reh} \dfrac{2\pi^2}{45} T_{\rm reh}^3 \\ \rho_{\rm reh} = g_{\rm reh} \dfrac{\pi^2}{30} T_{\rm reh}^4 \end{cases} \Rightarrow \quad \frac{a_0}{a_{\rm reh}} = \left(\frac{q_{\rm reh}^{1/3} g_0^{1/4}}{q_0^{1/3} g_{\rm reh}^{1/4}} \right) \frac{\rho_{\rm reh}^{1/4}}{\rho_\gamma^{1/4}}$$

$$\text{or} \quad 1 + z_{\rm reh} = \left(\frac{\rho_{\rm reh}}{\tilde{\rho}_\gamma} \right)^{1/4}$$

- Depends on $\rho_{\rm reh}$ and $\tilde{\rho}_\gamma \equiv \mathcal{Q}_{\rm reh} \rho_\gamma$
  - ✦ Energy density of radiation today: $\rho_\gamma = 3 \dfrac{H_0^2}{M_{\rm P}^2} \Omega_{\rm rad}$ (CMB photons)
  - ✦ Change in the number of entropy and energy relativistic degrees of freedom (small effect compared to $\rho_{\rm reh}/\rho_\gamma$)

$$\mathcal{Q}_{\rm reh} \equiv \frac{g_{\rm reh}}{g_0} \left(\frac{q_0}{q_{\rm reh}} \right)^{1/4}$$

Redshift at which inflation ends

- Depends on how the reheating proceeds

$$1 + z_{\mathrm{end}} = \frac{a_0}{a_{\mathrm{end}}} = \frac{a_{\mathrm{reh}}}{a_{\mathrm{end}}}(1 + z_{\mathrm{reh}}) = \frac{a_{\mathrm{reh}}}{a_{\mathrm{end}}}\left(\frac{\rho_{\mathrm{reh}}}{\tilde{\rho}_\gamma}\right)^{1/4} = \frac{1}{R_{\mathrm{rad}}}\left(\frac{\rho_{\mathrm{end}}}{\tilde{\rho}_\gamma}\right)^{1/4}$$

  - ✦ The reheating parameter $R_{\mathrm{rad}} \equiv \dfrac{a_{\mathrm{end}}}{a_{\mathrm{reh}}}\left(\dfrac{\rho_{\mathrm{end}}}{\rho_{\mathrm{reh}}}\right)^{1/4}$
  - ✦ Encodes any deviations from a radiation-like or instantaneous reheating $R_{\mathrm{rad}} = 1$

- R_{rad} can be expressed in terms of $(\rho_{\mathrm{reh}}, \bar{w}_{\mathrm{reh}})$ or $(\Delta N_{\mathrm{reh}}, \bar{w}_{\mathrm{reh}})$

$$\ln R_{\mathrm{rad}} = \frac{\Delta N_{\mathrm{reh}}}{4}(3\bar{w}_{\mathrm{reh}} - 1) = \frac{1 - 3\bar{w}_{\mathrm{reh}}}{12(1 + \bar{w}_{\mathrm{reh}})} \ln\left(\frac{\rho_{\mathrm{reh}}}{\rho_{\mathrm{end}}}\right)$$

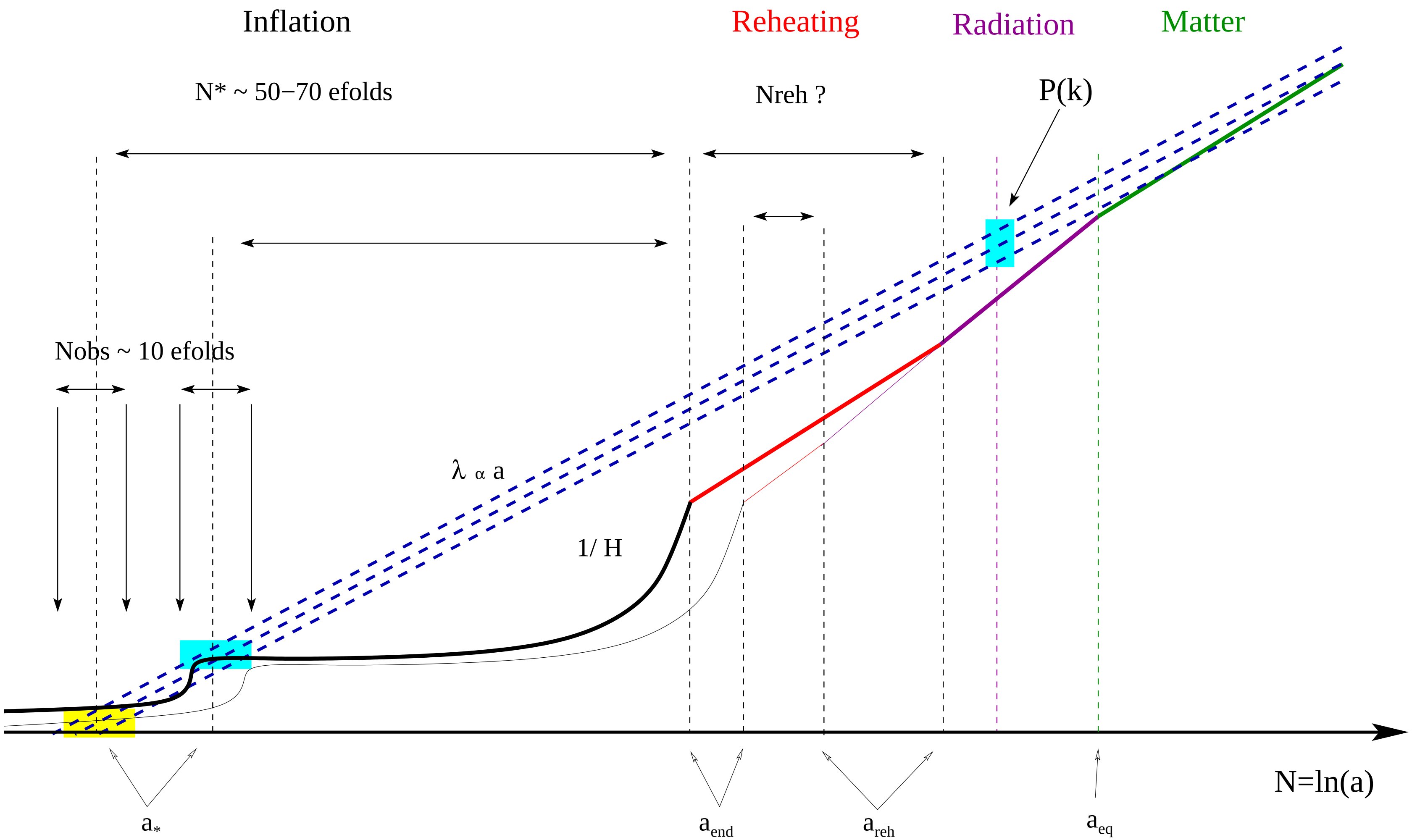
- A fixed inflationary parameters, z_{end} can still be affected by R_{rad}

Primordial power spectra

Outline

Primordial power spectra

Cosmological perturbations of inflationary origin



- Primordial power spectra for tensor and scalar perturbations are generated during inflation from quantum fluctuations $T = H/(2\pi)$

A toy example: test scalar fields in de Sitter

- Test = field fluctuations only ($m \ll H_{\mathrm{inf}}$): $\varphi(t,\boldsymbol{x}) = \phi(t) + \delta\phi(t,\boldsymbol{x})$
- Homogeneous part ($N \equiv \ln a$ "e-folds number")

$$\phi_{,tt} + 3H_{\mathrm{inf}}\phi_{,t} + m^2\phi = 0 \Rightarrow \phi(N) = \phi_0 e^{-Nm^2/(3H_{\mathrm{inf}}^2)} \to 0$$

- Fluctuations in Fourier space: $\mu \equiv a\delta\phi_{\boldsymbol{k}}$ ($aH_{\mathrm{inf}} = -1/\eta$)

$$\delta\phi_{\boldsymbol{k},tt} + 3H_{\mathrm{inf}}\delta\phi_{\boldsymbol{k},t} + \left(k^2 + m^2\right)\delta\phi_{\boldsymbol{k}} = 0 \Rightarrow \mu'' + \left(m^2 + k^2 - \frac{2}{\eta^2}\right)\mu = 0$$

- Free field quantization: positive energy waves for $k\eta \gg 1$

$$\mu = e^{i(\nu+1/2)\pi/2}\sqrt{\frac{\pi}{4k}}\sqrt{k\eta}H_\nu^1(k\eta), \quad \nu = \sqrt{\frac{9}{4} - \frac{m^2}{H_{\mathrm{inf}}^2}}$$

- Power spectra after Hubble exit: $\mathcal{P}_{\delta\phi} = \lim_{k\eta\ll 1} \dfrac{k^3}{2\pi^2}\left|\dfrac{\mu}{a}\right|^2$

Scale invariant primordial power spectrum

- For light fields $m \ll H_{\mathrm{inf}}$

$$\mathcal{P}_{\delta\phi} \simeq \frac{H_{\mathrm{inf}}^2}{4\pi^2}\left(\frac{k}{aH_{\mathrm{inf}}}\right)^{2m^2/(3H_{\mathrm{inf}}^2)} = \frac{H_{\mathrm{inf}}^2}{4\pi^2} + \dots$$

- Does not depend on k (scale invariant) and Gaussian
- Could explain the amplitude of CMB anisotropies $\delta T/T \simeq 10^{-5}$ for $H_{\mathrm{inf}} \simeq 10^{-5} M_{\mathrm{P}}$ (GUT scale)
- But test scalar fields cannot induce gravity perturbations, by definition
- Gravity perturbations must be included!
  - However, this is the right result for primordial gravity waves (up to a polarization factor)
- And ultra-light test scalar fields can explain dark energy!

Relic vacuum energy density from inflation

- Field variance in physical space after N e-folds

$$\langle\delta\phi^2\rangle = \int_{a_i H_{\rm inf}}^{a H_{\rm inf}} \frac{\mathrm{d}^3\boldsymbol{k}}{(2\pi)^3}\left|\delta\phi_{\boldsymbol{k}}\right|^2 = \frac{3H_{\rm inf}^4}{8\pi^2 m^2}\left[1-e^{-N(2m^2)/(3H_{\rm inf}^2)}\right] \to \frac{3H_{\rm inf}^4}{8\pi^2 m^2}$$

- Energy density expectation value (does not depend on m)

$$\langle V(\phi)\rangle = \frac{1}{2}m^2\langle\delta\phi^2\rangle = \frac{3H_{\rm inf}^4}{16\pi^2}$$

- Universal, does not even depend on V (for test fields)

$$P(\delta\phi|H_{\rm inf}) \propto \exp\left[-\frac{8\pi^2}{3H_{\rm inf}^4}V(\delta\phi)\right] \Rightarrow \langle V\rangle \simeq \frac{3H_{\rm inf}^4}{8\pi^2}$$

- This is dark energy provided: $H_{\rm inf} = (\Omega_\Lambda)^{1/4}\sqrt{4\pi H_0 M_{\rm P}}$

$$H_{\rm inf} \simeq 6\times 10^{-3}\,\mathrm{eV}, \quad \rho_{\rm inf}^{1/4} = (3M_{\rm P}^2 H_{\rm inf}^2)^{1/4} \simeq 5\,\mathrm{TeV}$$

Linear perturbations during inflation

- Perturbed FLRW metric (longitudinal gauge): $\delta\phi, \Phi, \Psi, h_{ij}$

$$\mathrm{d}s^2 = a^2(1+2\Phi)\mathrm{d}\eta^2 - a^2\left[(1-2\Psi)\delta_{ij} + h_{ij}\right]\mathrm{d}x^i\mathrm{d}x^j$$

- Perturbed Einstein + Klein-Gordon equations: $\delta G_{\mu\nu} = \kappa^2 \delta T_{\mu\nu}$
  - ✦ Equations of motion ($' = \partial_\eta$)

$$h_{ij}'' + 2\mathcal{H}h_{ij}' - \Delta h_{ij} = 0 \quad \Phi = \Psi, \quad \zeta' = \frac{2aH}{\phi'^2}\Delta\Psi$$

$$\text{where} \quad \zeta \equiv \Psi - \frac{\mathcal{H}}{\mathcal{H}' - \mathcal{H}^2}\left(\Psi' + \mathcal{H}\Phi\right) = \Psi + H\frac{\delta\phi}{\dot{\phi}}$$

  - ✦ Comoving curvature perturbation ζ and h are conserved on large scales $\Delta \sim k^2$ (single-field only!)

- Primordial power spectra can be evaluated anytime after Hubble exit

$$\mathcal{P}_\zeta(k) = \frac{k^3}{2\pi^2}\left|\zeta\right|^2, \qquad \mathcal{P}_h(k) = \frac{2k^3}{\pi^2}\left|h\right|^2 \quad \leftarrow 2 \text{ polarizations}$$

Scalar and tensor modes evolution

- Parametric oscillators

$$\left.\begin{aligned} \mu_{\mathrm{T}} &\equiv a h \\ \mu_{\mathrm{S}} &\equiv a\sqrt{2}\phi_{,N}\zeta \end{aligned}\right\} \Rightarrow \mu_{\mathrm{TS}}'' + \left[k^2 - \frac{(a\sqrt{\epsilon_1})''}{a\sqrt{\epsilon_1}}\right]\mu_{\mathrm{TS}} = 0$$

- Can be recast in terms of Hubble flow functions $\epsilon_i(\eta)$
  - ✦ Using $f' = aH\, f_{,N}\ldots$

$$\frac{\nu^2(\eta) - 1/4}{\eta^2} \equiv \frac{\left(a\sqrt{\epsilon_1}\right)''}{\left(a\sqrt{\epsilon_1}\right)} = \mathcal{H}^2\left(2 - \epsilon_1 + \frac{3}{2}\epsilon_2 + \frac{1}{4}\epsilon_2^2 - \frac{1}{2}\epsilon_1\epsilon_2 + \frac{1}{2}\epsilon_2\epsilon_3\right)$$

  - ✦ Expanding the conformal time in terms of ϵ_i

$$\eta = \int \frac{\mathrm{d}t}{a} = \int \frac{1}{a^2}\frac{\mathrm{d}a}{H} = -\frac{1}{aH} + \int \frac{1}{a}\frac{\mathrm{d}H^{-1}}{\mathrm{d}a}\mathrm{d}a = -\frac{1}{\mathcal{H}} + \int \frac{\epsilon_1}{a^2 H}\mathrm{d}a$$

$$= -\frac{1}{\mathcal{H}} - \frac{1}{a}\frac{\epsilon_1}{H} + \int \frac{1}{a}\frac{\mathrm{d}\left(\epsilon_1 H^{-1}\right)}{\mathrm{d}a}\mathrm{d}a = -\frac{1+\epsilon_1}{\mathcal{H}} + \int \frac{1}{a^2 H}\epsilon_1\left(\epsilon_1 + \epsilon_2\right)\mathrm{d}a$$

Slow-roll expansion for the perturbations

- Within the slow-roll approximation $\epsilon_i < 1$ and $\epsilon_i = \mathcal{O}(\epsilon)$
  - Consistent expansion at first order in slow-roll

$$\mathcal{H} = -\frac{1+\epsilon_1}{\eta} + \mathcal{O}(\epsilon^2) \Rightarrow \nu^2(\eta) = \frac{9}{4} + 3\epsilon_1(\eta) + \frac{3}{2}\epsilon_2(\eta) + \mathcal{O}(\epsilon^2)$$

  - Expanding Hubble flow functions around a particular time $\eta_\diamond$ $(N_\diamond)$

$$\left.\begin{aligned} \epsilon_i(N) &= \epsilon_i(N_\diamond) + (N - N_\diamond) \left.\frac{\mathrm{d}\epsilon_i}{\mathrm{d}N}\right|_{N_\diamond} + \ldots \\ N - N_\diamond &= -(1+\epsilon_{1\diamond}) \ln\left(\frac{\eta}{\eta_\diamond}\right) + \mathcal{O}(\epsilon^2) \end{aligned}\right\} \Rightarrow \epsilon_i(N) = \epsilon_i(N_\diamond) + \mathcal{O}(\epsilon^2)$$

- At first order (only) in slow-roll $\nu(\eta) = \nu_\diamond + \mathcal{O}(\epsilon^2)$ is constant

$$\nu_\diamond = \frac{9}{4} + 3\epsilon_{1\diamond} + \frac{3}{2}\epsilon_{2\diamond} \quad \Rightarrow \left\{ \begin{array}{l} \mu_\mathrm{S}'' + \left(k^2 - \dfrac{\nu_\diamond^2 - 1/4}{\eta^2}\right)\mu_\mathrm{S} = 0 \\ \text{this is a Bessel equation} \end{array}\right.$$

Quantum initial conditions

- Canonical quantization of $\mu_{\rm S}$ (and $\mu_{\rm T}$) + Bunch-Davies vacuum

$$\hat{\mu}(\eta,\mathbf{x}) = \int \frac{\mathrm{d}^3\mathbf{k}}{\sqrt{2k}}\left[c_{\mathbf{k}}(\eta_{\rm ini})\xi_k(\eta)\mathrm{e}^{i\mathbf{k}\mathbf{x}} + c_{\mathbf{k}}^{\dagger}(\eta_{\rm ini})\xi_k^*(\eta)\mathrm{e}^{-i\mathbf{k}\mathbf{x}}\right]$$

$$\xi_k(\eta) \underset{k\eta_{\rm ini}\to\infty}{\to} 1 \times \mathrm{e}^{-ik(\eta-\eta_{\rm ini})} + 0 \times \mathrm{e}^{+ik(\eta-\eta_{\rm ini})}$$

- For each mode, we set the equivalent classical initial conditions

$$\mu_{\rm TS}(\eta_{\rm ini}) = \kappa\sqrt{2}\frac{1}{\sqrt{2k}}, \quad \mu'_{\rm TS}(\eta_{\rm ini}) = -i\kappa\sqrt{2}\sqrt{\frac{k}{2}}$$

- The solution is uniquely determined and depends on $\eta_\diamond$
  - ✦ $\eta_\diamond$ should be chosen for each mode k around Hubble exit: $k\eta_\diamond = -1$. Other choices are possible, for instance $k = a(\eta_\diamond)H(\eta_\diamond)$

- The power spectra are obtained in the super-Hubble limit: $k\eta \to 0$

Scalar primordial power spectrum

- In the super-Hubble limit $k\eta \to 0$ one gets a time-independent expression ($C \equiv \gamma + \ln 2 - 2$)

$$\mathcal{P}_\zeta(\eta_\diamond) = \frac{H_\diamond^2}{8\pi^2 M_{\mathrm{P}}^2 \epsilon_{1\diamond}} \left[1 - 2(C+1)\epsilon_{1\diamond} - C\epsilon_{2\diamond} + \mathcal{O}(\epsilon^2)\right]$$

- Dependency in k is hidden in the definition of $\eta_\diamond \equiv -1/k$

- Can be made explicit with a pivot expansion around k_*
  - For instance $k_* = 0.05\,\mathrm{Mpc}^{-1} \Rightarrow \eta_* = -1/k_*$
  - All $f_\diamond$ quantities can be slow-roll expanded around η_*

$$H_\diamond = H_* + (N_\diamond - N_*) \left.\frac{\mathrm{d}H}{\mathrm{d}N}\right|_{N_*} + \cdots = H_* \left(1 - \epsilon_{1*} \ln\frac{\eta_*}{\eta_\diamond}\right) + \mathcal{O}(\epsilon^2)$$

$$\epsilon_{1\diamond} = \epsilon_{1*} + \epsilon_{1*}\epsilon_{2*} \ln\frac{\eta_*}{\eta_\diamond} + \mathcal{O}(\epsilon^3)$$

- Pivot expanded scalar power spectrum

$$\mathcal{P}_\zeta(k) = \frac{H_*^2}{8\pi^2 M_{\mathrm{P}}^2 \epsilon_{1*}} \left[1 - 2(C+1)\epsilon_{1*} - C\epsilon_{2*} - (2\epsilon_{1*} + \epsilon_{2*}) \ln\left(\frac{k}{k_*}\right)\right]$$

Primordial power spectrum

- At second order, after pivot expansion, one gets

$$\mathcal{P}_\zeta = \frac{H_*^2}{8\pi^2 M_{\rm P}^2 \epsilon_{1*}} \Bigg\{ 1 - 2(1+C)\epsilon_{1*} - C\epsilon_{2*} + \left(\frac{\pi^2}{2} - 3 + 2C + 2C^2\right)\epsilon_{1*}^2 + \left(\frac{7\pi^2}{12} - 6 - C + C^2\right)\epsilon_{1*}\epsilon_{2*} + \left(\frac{\pi^2}{8} - 1 + \frac{C^2}{2}\right)\epsilon_{2*}^2 + \left(\frac{\pi^2}{24} - \frac{C^2}{2}\right)\epsilon_{2*}\epsilon_{3*} + \Big[-2\epsilon_{1*} - \epsilon_{2*} + (2+4C)\epsilon_{1*}^2 + (-1+2C)\epsilon_{1*}\epsilon_{2*} + C\epsilon_{2*}^2 - C\epsilon_{2*}\epsilon_{3*}\Big] \ln\left(\frac{k}{k_*}\right) + \left[2\epsilon_{1*}^2 + \epsilon_{1*}\epsilon_{2*} + \frac{1}{2}\epsilon_{2*}^2 - \frac{1}{2}\epsilon_{2*}\epsilon_{3*}\right] \ln^2\left(\frac{k}{k_*}\right) \Bigg\},$$

$$\mathcal{P}_h = \frac{2H_*^2}{\pi^2 M_{\rm P}^2} \Bigg\{ 1 - 2(1+C)\epsilon_{1*} + \left[-3 + \frac{\pi^2}{2} + 2C + 2C^2\right]\epsilon_{1*}^2 + \left[-2 + \frac{\pi^2}{12} - 2C - C^2\right]\epsilon_{1*}\epsilon_{2*} + \Big[-2\epsilon_{1*} + (2+4C)\epsilon_{1*}^2 + (-2-2C)\epsilon_{1*}\epsilon_{2*}\Big] \ln\left(\frac{k}{k_*}\right) + \left(2\epsilon_{1*}^2 - \epsilon_{1*}\epsilon_{1*}\right) \ln^2\left(\frac{k}{k_*}\right) \Bigg\}$$

Power law parameters

- Amplitude and spectral indices: $n_{\mathrm{T}} \equiv \left.\frac{\mathrm{d}\ln\mathcal{P}_h}{\mathrm{d}\ln k}\right|_{k_*}$, $n_{\mathrm{S}} - 1 \equiv \left.\frac{\mathrm{d}\ln\mathcal{P}_\zeta}{\mathrm{d}\ln k}\right|_{k_*}$

$$P_* = \mathcal{P}_\zeta(k_*), \quad n_{\mathrm{T}} = -2\epsilon_{1*} - 2\epsilon_{1*}^2 - 2(1+C)\epsilon_{1*}\epsilon_{2*} + \mathcal{O}\left(\epsilon^3\right)$$

$$n_{\mathrm{S}} = 1 - (2\epsilon_{1*} + \epsilon_{2*}) - 2\epsilon_{1*}^2 - (3+2C)\epsilon_{1*}\epsilon_{2*} - C\epsilon_{2*}\epsilon_{3*} + \mathcal{O}\left(\epsilon^3\right)$$

- Running of the spectral index: $\alpha \equiv \left.\frac{\mathrm{d}^2\ln\mathcal{P}}{\mathrm{d}(\ln k)^2}\right|_{k_*}$

$$\alpha_{\mathrm{S}} = -2\epsilon_{1*}\epsilon_{2*} - \epsilon_{2*}\epsilon_{3*} + \mathcal{O}\left(\epsilon^3\right), \qquad \alpha_{\mathrm{T}} = -2\epsilon_{1*}\epsilon_{2*} + \mathcal{O}\left(\epsilon^3\right)$$

- Tensor-to-scalar ratio $r \equiv \frac{\mathcal{P}_\zeta(k_*)}{\mathcal{P}_h(k_*)} = 16\epsilon_{1*}(1 + C\epsilon_{2*}) + \mathcal{O}\left(\epsilon^3\right)$

- Running of the running: $\beta \equiv \left.\frac{\mathrm{d}^3\ln\mathcal{P}}{\mathrm{d}(\ln k)^3}\right|_{k_*} = \mathcal{O}\left(\epsilon^3\right)$

$$\beta_{\mathrm{T}} = -2\epsilon_{1*}\epsilon_{2*}\left(\epsilon_{2*} + \epsilon_{3*}\right) + \mathcal{O}\left(\epsilon^4\right)$$

Getting the power law parameters in ASPIC

- From the input of ϵ_{i*}

```fortran
module srflow
!this is in k* defined by k* eta* = -1
  use infprec, only : kp, pi, CConst

  implicit none

  private

  real(kp), parameter :: pi2 = pi*pi

  logical, parameter :: display = .false.

  public slowroll_violated
  public tensor_to_scalar_ratio
  public scalar_spectral_index, tensor_spectral_index
  public scalar_running, tensor_running
  public scalar_running_running, tensor_running_running
  public slowroll_corrections, ln_slowroll_corrections

! because 1/slowroll_corrections >< inverse_slowroll_correction
s
! it is better to numerically stick to one choice only
!  public inverse_slowroll_corrections, ln_inverse_slowroll_cor
rections
contains


  function slowroll_violated(eps)
    implicit none
    logical :: slowroll_violated
    real(kp), dimension(:), intent(in) :: eps

    slowroll_violated =  any(abs(eps).gt.1._kp)

  end function slowroll_violated



  function scalar_spectral_index(eps)
    implicit none
    real(kp) :: scalar_spectral_index
    real(kp), intent(in), dimension(:) :: eps
    real(kp) :: nsm1
    integer :: neps

    neps = size(eps,1)

    select case (neps)

    case (2)
       nsm1 = -2._kp*eps(1) - eps(2)

    case (3)
       nsm1 = -2._kp*eps(1) - eps(2)
       nsm1 = nsm1 - 2._kp*eps(1)**2 - (3._kp+2._kp*CConst)*eps
(1)*eps(2) &
            - CConst*eps(2)*eps(3)

    case default
       stop 'scalar_spectral_index: neps not implemented!'
    end select


    scalar_spectral_index = nsm1 + 1._kp

  end function scalar_spectral_index



  function tensor_spectral_index(eps)
    implicit none
    real(kp) :: tensor_spectral_index
    real(kp), intent(in), dimension(:) :: eps
    real(kp) :: nt
    integer :: neps

    neps = size(eps,1)

    select case (neps)

    case (1)
       nt = -2._kp*eps(1)

    case (2)
       nt = -2._kp*eps(1) &
            - 2._kp*eps(1)**2 - 2._kp*(1._kp+CConst)*eps(1)*eps
(2)

    case(3)
       nt = -2._kp*eps(1) &
            - 2._kp*eps(1)**2 - 2._kp*(1._kp+CConst)*eps(1)*eps
(2) &
            - 2._kp*eps(1)**3 - (14._kp+6._kp*CConst-pi2)*eps(1
)**2*eps(2) &
            - (2._kp + 2._kp*CConst*(1._kp+CConst) -pi2/12._kp)
*eps(1)*eps(2)*(eps(2)+eps(3))

    case default
       stop 'tensor_spectral_index: neps not implemented!'
    end select


    tensor_spectral_index = nt

  end function tensor_spectral_index
```

```fortran
  function tensor_to_scalar_ratio(eps)
    implicit none
    real(kp) :: tensor_to_scalar_ratio
    real(kp), intent(in), dimension(:) :: eps
    real(kp) :: r
    integer :: neps

    neps = size(eps,1)

    select case (neps)

    case (1)
       r = 1._kp

    case (2)
       r = 1._kp + CConst*eps(2)

    case (3)
       r = 1._kp + CConst*eps(2) &
            + (CConst - pi2/2._kp + 4._kp)*eps(1)*eps(2) &
            + (0.5_kp*CConst**2 - pi2/8._kp + 1._kp)*eps(2)**2
&
            + (0.5_kp*CConst**2 - pi2/24._kp)*eps(2)*eps(3)

    case default
       stop 'tensor_to_scalar_ratio: neps not implemented!'
    end select

!that may happen if eps2 > 1, i.e. when slow-roll is violated.
If
!this is the case, we use the zero order formula. If this is no
t the
!case, this is nasty and we abort.
    if (r.lt.0._kp) then
       if (slowroll_violated(eps)) then
          if (display) write(*,*),'tensor_to_scalar_ratio: eps(:)= ',eps(:
)
          tensor_to_scalar_ratio = 16._kp*eps(1)
       else
          stop 'tensor_to_scalar_ratio: r < 0!'
       endif
    else
       tensor_to_scalar_ratio = 16._kp*eps(1)*r
    end if



  end function tensor_to_scalar_ratio


  function scalar_running(eps)
    implicit none
    real(kp) :: scalar_running
    real(kp), intent(in), dimension(:) :: eps
    real(kp) :: alpha
    integer :: neps

    neps = size(eps,1)

    select case (neps)

    case (1,2)
       alpha = 0._kp

    case (3)
       alpha = -2._kp*eps(1)*eps(2) - eps(2)*eps(3)

    case default
       stop 'scalar_running: neps not implemented!'
    end select

    scalar_running = alpha

  end function scalar_running



  function tensor_running(eps)
    implicit none
    real(kp) :: tensor_running
    real(kp), intent(in), dimension(:) :: eps
    real(kp) :: alpha
    integer :: neps

    neps = size(eps,1)

    select case (neps)

    case (1)
       alpha = 0._kp

    case (2)
       alpha = -2._kp*eps(1)*eps(2)

    case (3)
       alpha = -2._kp*eps(1)*eps(2) &
            - 6._kp*eps(1)**2*eps(2) - 2._kp*(1._kp+CConst)*eps
(1)*eps(2)**2 &
            - 2._kp*(1._kp+CConst)*eps(1)*eps(2)*eps(3)

    case default
       stop 'scalar_running: neps not implemented!'
    end select

    tensor_running = alpha

  end function tensor_running
```

```fortran
  function scalar_running_running(eps)
    implicit none
    real(kp) :: scalar_running_running
    real(kp), intent(in), dimension(:) :: eps
    real(kp) :: beta
    integer :: neps

    neps = size(eps,1)

    select case (neps)

    case (1,2,3)
       beta = 0._kp

    case (4)
       beta = -2._kp*eps(1)*eps(2)**2 - 2._kp*eps(1)*eps(2)*eps
(3) &
            - eps(2)*eps(3)**2 - eps(2)*eps(3)*eps(4)

    case default
       stop 'tensor_running_running: neps not implemented!'
    end select

    scalar_running_running = beta

  end function scalar_running_running



  function tensor_running_running(eps)
    implicit none
    real(kp) :: tensor_running_running
    real(kp), intent(in), dimension(:) :: eps
    real(kp) :: beta
    integer :: neps

    neps = size(eps,1)

    select case (neps)

    case (1,2)
       beta = 0._kp

    case (3)
       beta = -2._kp*eps(1)*eps(2)*(eps(2)+eps(3))

    case default
       stop 'tensor_running_running: neps not implemented!'
    end select

    tensor_running_running = beta

  end function tensor_running_running



!this gives the normal slow-roll expansion of P(k*)/[H*^2/(8pi^
2 eps1*)]
  function slowroll_corrections(eps)
    implicit none
    real(kp) :: slowroll_corrections
    real(kp), dimension(:), intent(in) :: eps
    integer :: neps

    neps = size(eps,1)

    if (slowroll_violated(eps)) then
       write(*,*)'eps= ',eps
       stop 'slowroll_corrections not reliable!'
    endif

    select case (neps)

    case (1)
       slowroll_corrections = 1._kp

    case (2)
       slowroll_corrections = 1._kp - 2._kp*(1._kp+Cconst)*eps(
1) &
            - CConst*eps(2)

    case (3)
       slowroll_corrections = 1._kp - 2._kp*(1._kp+Cconst)*eps(
1) &
            - CConst*eps(2) &
            + (-3._kp + 2._kp*CConst + 2._kp*CConst**2 + pi2/2.
_kp) &
            *eps(1)**2 &
            + (-6._kp - Cconst + CConst**2 + 7._kp*pi2/12._kp)
&
            *eps(1)*eps(2) &
            + (-1._kp + CConst**2/2._kp + pi2/8._kp)*eps(2)**2
&
            + (-CConst**2/2._kp + pi2/24._kp)*eps(2)*eps(3)

    case default
       stop 'inverse_slow_corrections: order not implemented!'

    end select
```

```fortran
  end function slowroll_corrections



!this gives [H*^2/(8pi^2 eps1*)]/P(k*) consitently expanded in
slow-roll
!hence the name "inverse" as opposed to normal slow-roll expans
ion
  function inverse_slowroll_corrections(eps)
    implicit none
    real(kp) :: inverse_slowroll_corrections
    real(kp), dimension(:), intent(in) :: eps
    integer :: neps

    neps = size(eps,1)

    if (slowroll_violated(eps)) then
       write(*,*)'eps= ',eps
       stop 'inverse_slowroll_corrections not reliable!'
    endif

    select case (neps)

    case (1)
       inverse_slowroll_corrections = 1._kp

    case (2)
       inverse_slowroll_corrections = 1._kp + 2._kp*(1._kp+Ccon
st)*eps(1) &
            + CConst*eps(2)

    case (3)
       inverse_slowroll_corrections = 1._kp + 2._kp*(1._kp+Ccon
st)*eps(1) &
            + CConst*eps(2) &
            + (7._kp + 6._kp*CConst + 2._kp*CConst**2 - pi2/2._
kp) &
            *eps(1)**2 &
            + (6._kp + 5._kp*Cconst + 3._kp*CConst**2 - 7._kp*p
i2/12._kp) &
            *eps(1)*eps(2) &
            + (1._kp + CConst**2/2._kp - pi2/8._kp)*eps(2)**2 &
            + (CConst**2/2._kp - pi2/24._kp)*eps(2)*eps(3)

    case default
       stop 'inverse_slow_corrections: order not implemented!'

    end select

  end function inverse_slowroll_corrections



!this gives ln[P(k*)] - ln[H*^2/(8pi^2 eps1*)] consistently exp
anded
  function ln_slowroll_corrections(eps)
    implicit none
    real(kp) :: ln_slowroll_corrections
    real(kp), dimension(:), intent(in) :: eps
    integer :: neps

    neps = size(eps,1)

    if (slowroll_violated(eps)) then
       write(*,*)'eps= ',eps
       stop 'inverse_slowroll_corrections not reliable!'
    endif

    select case (neps)

    case (1)
       ln_slowroll_corrections = 0._kp

    case (2)
       ln_slowroll_corrections = -2._kp*(1._kp+Cconst)*eps(1) &
            - CConst*eps(2)

    case (3)
       ln_slowroll_corrections = -2._kp*(1._kp+Cconst)*eps(1) &
            - CConst*eps(2) &
            - (5._kp + 2._kp*CConst - pi2/2._kp)*eps(1)**2 &
            - (6._kp + 3._kp*Cconst + CConst**2 - 7._kp*pi2/12.
_kp) &
            *eps(1)*eps(2) &
            - (1._kp - pi2/8._kp)*eps(2)**2 &
            - (CConst**2/2._kp - pi2/24._kp)*eps(2)*eps(3)

    case default
       stop 'ln_slow_corrections: order not implemented!'

    end select


  end function ln_slowroll_corrections


!this gives ln[H*^2/(8pi^2 eps1*)] - ln[P(k*)] consistently exp
anded
  function ln_inverse_slowroll_corrections(eps)
    implicit none
    real(kp) :: ln_inverse_slowroll_corrections
    real(kp), dimension(:), intent(in) :: eps
    integer :: neps

    ln_inverse_slowroll_corrections = -ln_slowroll_corrections(
eps)

  end function ln_inverse_slowroll_corrections
```

Observable quantities during inflation

- All quantities entering $\mathcal{P}(k)$ are evaluated at η_* such that $k_*\eta_* = -1$
  - ✦ Hubble flow functions: $\epsilon_{i*} = \epsilon_i(\phi_*)$ where $\eta(\phi_*) = -1/k_*$

- At leading order in slow-roll: $k_* = a_* H_* = e^{N_* - N_{\mathrm{end}}} \dfrac{a_{\mathrm{end}}}{a_0} a_0 H_*$

$$\frac{k_*}{a_0} = \frac{e^{\Delta N_*}}{1+z_{\mathrm{end}}} H_* = e^{\Delta N_*} R_{\mathrm{rad}} \left(\frac{\rho_{\mathrm{end}}}{\tilde{\rho}_\gamma}\right)^{-1/4} \left(\frac{H_*}{\sqrt{\epsilon_{1*}}}\right)\sqrt{\epsilon_{1*}}$$

- This is a non-trivial integral equation: $\rho_{\mathrm{end}}(\phi_*)$ through M^4
  - ✦ FL equation: $\rho_{\mathrm{end}} = 3H_{\mathrm{end}}^2 = \dfrac{3V_{\mathrm{end}}}{3-\epsilon_{1\mathrm{end}}} = 3\epsilon_{1*}\dfrac{H_*^2}{\epsilon_{1*}}\dfrac{V_{\mathrm{end}}}{V_*}\dfrac{3-\epsilon_{1*}}{3-\epsilon_{1\mathrm{end}}}$
  - ✦ Defining $N_0 \equiv \ln\left(\dfrac{k_*}{a_0}\dfrac{1}{\tilde{\rho}_\gamma^{1/4}}\right)$ (number of e-folds of deceleration)

$$\Delta N_* = -\ln R_{\mathrm{rad}} + N_0 + \frac{1}{4}\ln\left(\frac{3}{\epsilon_{1*}}\frac{V_{\mathrm{end}}}{V_*}\frac{3-\epsilon_{1*}}{3-\epsilon_{1\mathrm{end}}}\right) - \frac{1}{4}\ln\left(\frac{H_*^2}{\epsilon_{1*}}\right)$$

Solving for the time of pivot crossing

- Depends on: model + how inflation ends + reheating + data

$$-\left[\int_{\phi_{\mathrm{end}}}^{\phi_*}\frac{V(\psi)}{V'(\psi)}\mathrm{d}\psi\right] = \ln R_{\mathrm{rad}} - N_0 + \frac{1}{4}\ln(8\pi^2 P_*) - \frac{1}{4}\ln\left\{\frac{9}{\epsilon_1(\phi_*)[3-\epsilon_1(\phi_{\mathrm{end}})]}\frac{V(\phi_{\mathrm{end}})}{V(\phi_*)}\right\}$$

- The rescaled reheating parameter: $\ln R_{\mathrm{reh}} \equiv \ln R_{\mathrm{rad}} + \frac{1}{4}\ln \rho_{\mathrm{end}}$

$$-\left[\int_{\phi_{\mathrm{end}}}^{\phi_*}\frac{V(\psi)}{V'(\psi)}\mathrm{d}\psi\right] = \ln R_{\mathrm{reh}} - N_0 - \frac{1}{2}\ln\left\{\frac{9}{3-\epsilon_1(\phi_{\mathrm{end}})}\frac{V(\phi_{\mathrm{end}})}{V(\phi_*)}\right\}$$

- Assuming $-1/3 < \bar{w}_{\mathrm{reh}} < 1$ and $\rho_{\mathrm{nuc}} \equiv (10\,\mathrm{MeV})^4 < \rho_{\mathrm{reh}} < \rho_{\mathrm{end}} < 1$

$$-46 < \ln R_{\mathrm{reh}} < 15 + \frac{1}{3}\ln\rho_{\mathrm{end}}$$

Reheating consistent slow-roll in ASPIC

- For all *Encyclopædia Inflationaris* models, the reheating equations and their integration is done in ASPIC
- Example LFI: input potential parameters, $\ln R_{\mathrm{rad}}$, $\ln R_{\mathrm{reh}}$, $(\bar{w}_{\mathrm{reh}}, \rho_{\mathrm{reh}})$

```
!large field model reheating functions in the slow-roll approximations

module lfireheat
  use infprec, only : kp, tolkp, transfert
  use inftools, only : zbrent
  use srreheat, only : get_calfconst, find_reheat, slowroll_validity
  use srreheat, only : display, pi, Nzero, ln_rho_endinf
  use srreheat, only : ln_rho_reheat
  use srreheat, only : get_calfconst_rrad, find_reheat_rrad
  use srreheat, only : get_calfconst_rreh, find_reheat_rreh
  use lfisr, only : lfi_epsilon_one, lfi_epsilon_two, lfi_epsilon_three
  use lfisr, only : lfi_norm_potential
  use lfisr, only : lfi_x_endinf, lfi_efold_primitive
  implicit none

  private

  public lfi_x_star, lfi_lnrhoreh_max
  public lfi_xp_fromepsilon, lfi_lnrhoreh_fromepsilon

  public lfi_x_rrad, lfi_x_rreh

contains

!returns x given potential parameters, scalar power, wreh and
!lnrhoreh. If present, returns the corresponding bfoldstar
  function lfi_x_star(p,w,lnRhoReh,Pstar,bfoldstar)
    implicit none
    real(kp) :: lfi_x_star
    real(kp), intent(in) :: p,lnRhoReh,w,Pstar
    real(kp), intent(out), optional :: bfoldstar

    real(kp), parameter :: tolzbrent=tolkp
    real(kp) :: mini,maxi,calF,x
    real(kp) :: primEnd,epsOneEnd,xend,potEnd

    type(transfert) :: lfiData

    if (w.eq.1._kp/3._kp) then
       if (display) write(*,*)'w = 1/3 : solving for rhoReh = rhoEnd'
    endif

    xEnd = lfi_x_endinf(p)
    epsOneEnd = lfi_epsilon_one(xEnd,p)
    potEnd = lfi_norm_potential(xEnd,p)
    primEnd = lfi_efold_primitive(xEnd,p)

    calF = get_calfconst(lnRhoReh,Pstar,w,epsOneEnd,potEnd)

    lfiData%real1 = p
    lfiData%real2 = w
    lfiData%real3 = calF + primEnd

    mini = xEnd
    maxi = 1._kp/epsilon(1._kp)

    x = zbrent(find_lfi_x_star,mini,maxi,tolzbrent,lfiData)
    lfi_x_star = x

    if (present(bfoldstar)) then
       bfoldstar = - (lfi_efold_primitive(x,p) - primEnd)
    endif

  end function lfi_x_star

  function find_lfi_x_star(x,lfiData)
    implicit none
    real(kp) :: find_lfi_x_star
    real(kp), intent(in) :: x
    type(transfert), optional, intent(inout) :: lfiData

    real(kp) :: primStar,p,w,CalFplusprimEnd,potStar
    real(kp) :: epsOneStar

    p=lfiData%real1
    w = lfiData%real2
    CalFplusprimEnd = lfiData%real3

    primStar = lfi_efold_primitive(x,p)
    epsOneStar = lfi_epsilon_one(x,p)
    potStar = lfi_norm_potential(x,p)

    find_lfi_x_star = find_reheat(primStar,calFplusprimEnd,w&
         ,epsOneStar,potStar)

  end function find_lfi_x_star

!returns x given potential parameters, scalar power, and lnRrad.
!If present, returns the corresponding bfoldstar
  function lfi_x_rrad(p,lnRrad,Pstar,bfoldstar)
    implicit none
    real(kp) :: lfi_x_rrad
    real(kp), intent(in) :: p,lnRrad,Pstar
    real(kp), intent(out), optional :: bfoldstar

    real(kp), parameter :: tolzbrent=tolkp
    real(kp) :: mini,maxi,calF,x
    real(kp) :: primEnd,epsOneEnd,xend,potEnd

    type(transfert) :: lfiData


    if (lnRrad.eq.0._kp) then
       if (display) write(*,*)'Rrad=1 : solving for instantaneous reheating!'
    endif

    xEnd = lfi_x_endinf(p)
```

```
    epsOneEnd = lfi_epsilon_one(xEnd,p)
    potEnd = lfi_norm_potential(xEnd,p)
    primEnd = lfi_efold_primitive(xEnd,p)

    calF = get_calfconst_rrad(lnRrad,Pstar,epsOneEnd,potEnd)

    lfiData%real1 = p
    lfiData%real2 = calF + primEnd

    mini = xEnd
    maxi = 1._kp/epsilon(1._kp)

    x = zbrent(find_lfi_x_rrad,mini,maxi,tolzbrent,lfiData)
    lfi_x_rrad = x

    if (present(bfoldstar)) then
       bfoldstar = - (lfi_efold_primitive(x,p) - primEnd)
    endif

  end function lfi_x_rrad

  function find_lfi_x_rrad(x,lfiData)
    implicit none
    real(kp) :: find_lfi_x_rrad
    real(kp), intent(in) :: x
    type(transfert), optional, intent(inout) :: lfiData

    real(kp) :: primStar,p,w,CalFplusprimEnd
    real(kp) :: potStar
    real(kp) :: epsOneStar

    p=lfiData%real1
    CalFplusprimEnd = lfiData%real2

    primStar = lfi_efold_primitive(x,p)
    epsOneStar = lfi_epsilon_one(x,p)
    potStar = lfi_norm_potential(x,p)

    find_lfi_x_rrad = find_reheat_rrad(primStar,calFplusprimEnd &
         ,epsOneStar,potStar)

  end function find_lfi_x_rrad


!returns x given potential parameters and lnR (no need of Pstar, lnR
!is optimal for CMB). If present, returns the corresponding bfoldstar
  function lfi_x_rreh(p,lnRreh,bfoldstar)
    implicit none
    real(kp) :: lfi_x_rreh
    real(kp), intent(in) :: p,lnRreh
    real(kp), intent(out), optional :: bfoldstar

    real(kp), parameter :: tolzbrent=tolkp
    real(kp) :: mini,maxi,calF,x
    real(kp) :: primEnd,epsOneEnd,xend,potEnd

    type(transfert) :: lfiData


    if (lnRreh.eq.0._kp) then
       if (display) write(*,*)'Rreh=1 : solving for instantaneous reheating!'
    endif

    xEnd = lfi_x_endinf(p)
    epsOneEnd = lfi_epsilon_one(xEnd,p)
    potEnd = lfi_norm_potential(xEnd,p)
    primEnd = lfi_efold_primitive(xEnd,p)

    calF = get_calfconst_rreh(lnRreh,epsOneEnd,potEnd)

    lfiData%real1 = p
    lfiData%real2 = calF + primEnd

    mini = xEnd
    maxi = 1._kp/epsilon(1._kp)

    x = zbrent(find_lfi_x_rreh,mini,maxi,tolzbrent,lfiData)
    lfi_x_rreh = x

    if (present(bfoldstar)) then
       bfoldstar = - (lfi_efold_primitive(x,p) - primEnd)
    endif

  end function lfi_x_rreh

  function find_lfi_x_rreh(x,lfiData)
    implicit none
    real(kp) :: find_lfi_x_rreh
    real(kp), intent(in) :: x
    type(transfert), optional, intent(inout) :: lfiData

    real(kp) :: primStar,p,w,CalFplusprimEnd
    real(kp) :: potStar

    p=lfiData%real1
    CalFplusprimEnd = lfiData%real2

    primStar = lfi_efold_primitive(x,p)
    potStar = lfi_norm_potential(x,p)

    find_lfi_x_rreh = find_reheat_rreh(primStar,calFplusprimEnd &
         ,potStar)

  end function find_lfi_x_rreh
```

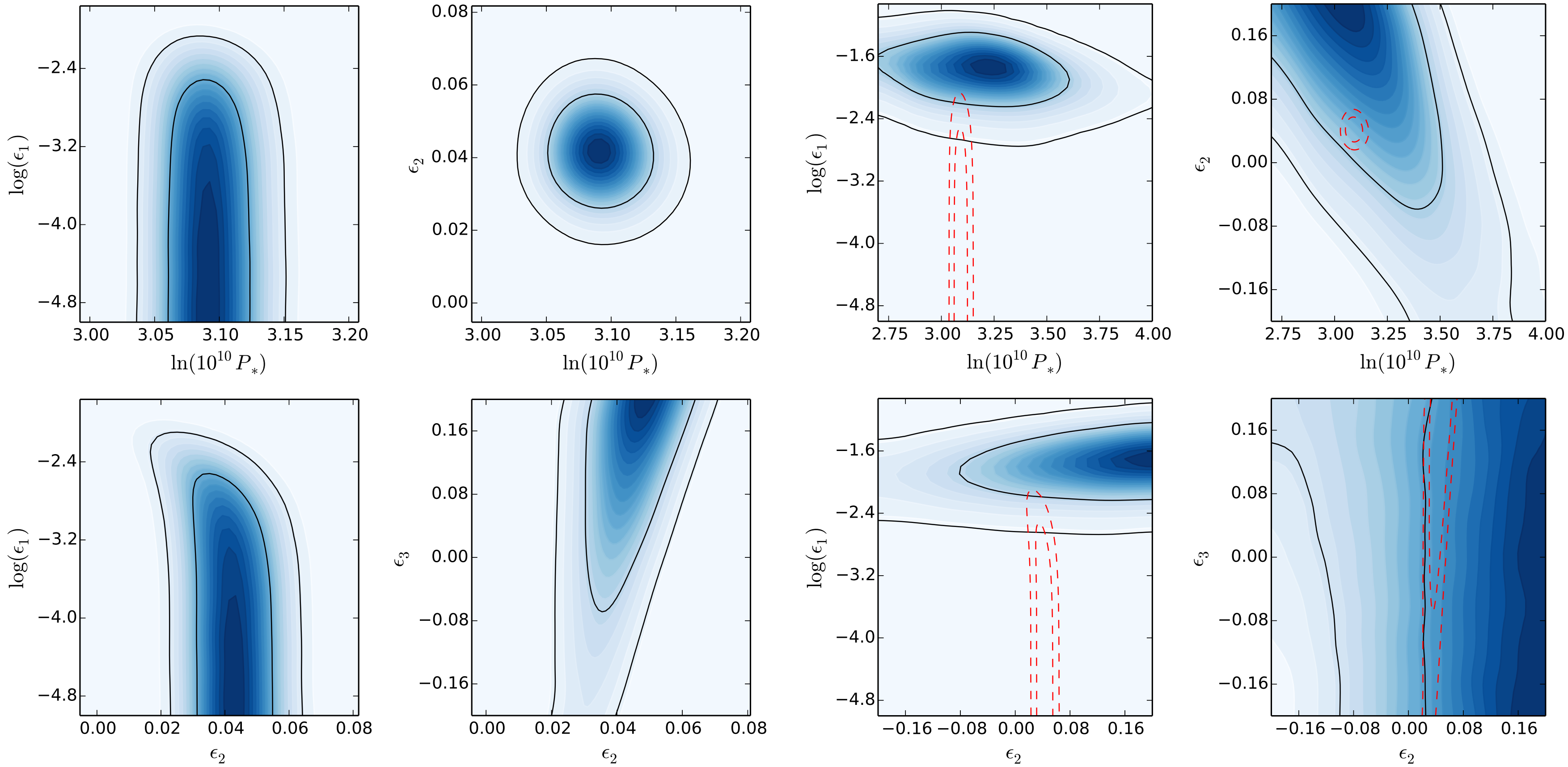
```
  function lfi_lnrhoreh_max(p,Pstar)
    implicit none
    real(kp) :: lfi_lnrhoreh_max
    real(kp), intent(in) :: p,Pstar

    real(kp) :: xEnd, potEnd, epsOneEnd
    real(kp) :: x, potStar, epsOneStar

    real(kp),parameter :: wrad=1._kp/3._kp
    real(kp),parameter :: junk=0._kp

    real(kp) :: lnRhoEnd


    xEnd = lfi_x_endinf(p)
    potEnd  = lfi_norm_potential(xEnd,p)
    epsOneEnd = lfi_epsilon_one(xEnd,p)

!   Trick to return x such that rho_reh=rho_end

    x = lfi_x_star(p,wrad,junk,Pstar)
    potStar = lfi_norm_potential(x,p)
    epsOneStar = lfi_epsilon_one(x,p)

    if (.not.slowroll_validity(epsOneStar)) stop 'lfi_lnrhoreh_max: slow-roll violated!'

    lnRhoEnd = ln_rho_endinf(Pstar,epsOneStar,epsOneEnd,potEnd/potStar)

    lfi_lnrhoreh_max = lnRhoEnd


  end function lfi_lnrhoreh_max



!return the unique p,x giving eps12 (and bfold if input)
  function lfi_xp_fromepsilon(eps1,eps2,bfold)
    implicit none
    real(kp), dimension(2) :: lfi_xp_fromepsilon
    real(kp), intent(in) :: eps1,eps2
    real(kp), intent(out), optional :: bfold

    real(kp) :: x, xEnd, p

    if (eps2.le.0._kp) then
       stop 'lfi_xp_fromepsilon: eps2<=0'
    endif

    p = 4._kp*eps1/eps2
    x = sqrt(8._kp*eps1/eps2/eps2)

    lfi_xp_fromepsilon(1) = x
    lfi_xp_fromepsilon(2) = p

    xEnd = lfi_x_endinf(p)

    if (present(bfold)) then
       bfold = -(lfi_efold_primitive(x,p)-lfi_efold_primitive(xEnd,p))
    endif

  end function lfi_xp_fromepsilon


!returns lnrhoreh from eps12, wreh and Pstar (and bfoldstar if
!present)
  function lfi_lnrhoreh_fromepsilon(w,eps1,eps2,Pstar,bfoldstar)
    implicit none
    real(kp) :: lfi_lnrhoreh_fromepsilon
    real(kp), intent(inout) :: w
    real(kp), intent(in) :: eps1,eps2,Pstar
    real(kp), intent(out), optional :: bfoldstar

    real(kp) :: p
    real(kp) :: xEnd, potEnd, epsOneEnd
    real(kp) :: x, potStar, epsOneStar
    real(kp) :: deltaNstar
    real(kp), dimension(2) :: lfistar

    logical, parameter :: printTest = .false.
    logical, parameter :: enforceWofp = .true.

    lfistar = lfi_xp_fromepsilon(eps1,eps2,bfoldstar)
    x = lfistar(1)
    p = lfistar(2)

    if (enforceWofp) then
       w = (p-2._kp)/(p+2._kp)
    endif

    xEnd = lfi_x_endinf(p)
    potEnd  = lfi_norm_potential(xEnd,p)
    epsOneEnd = lfi_epsilon_one(xEnd,p)

    potStar = lfi_norm_potential(x,p)
    epsOneStar = lfi_epsilon_one(x,p)

    if (.not.slowroll_validity(epsOneStar)) stop 'lfi_lnrhoreh: cannot trust slow-roll!'

    deltaNstar = lfi_efold_primitive(x,p) - lfi_efold_primitive(xEnd,p)

    if (printTest) then
       write(*,*)'eps1in= eps1comp= ',eps1, epsOneStar
       write(*,*)'eps2in= eps2comp= ',eps2, lfi_epsilon_two(x,p)
       write(*,*)'eps3in= eps3comp= ',eps2, lfi_epsilon_three(x,p)
    endif

    lfi_lnrhoreh_fromepsilon =  ln_rho_reheat(w,Pstar,epsOneStar,epsOneEnd,deltaNstar
 &
```

Comparison with observations

Outline

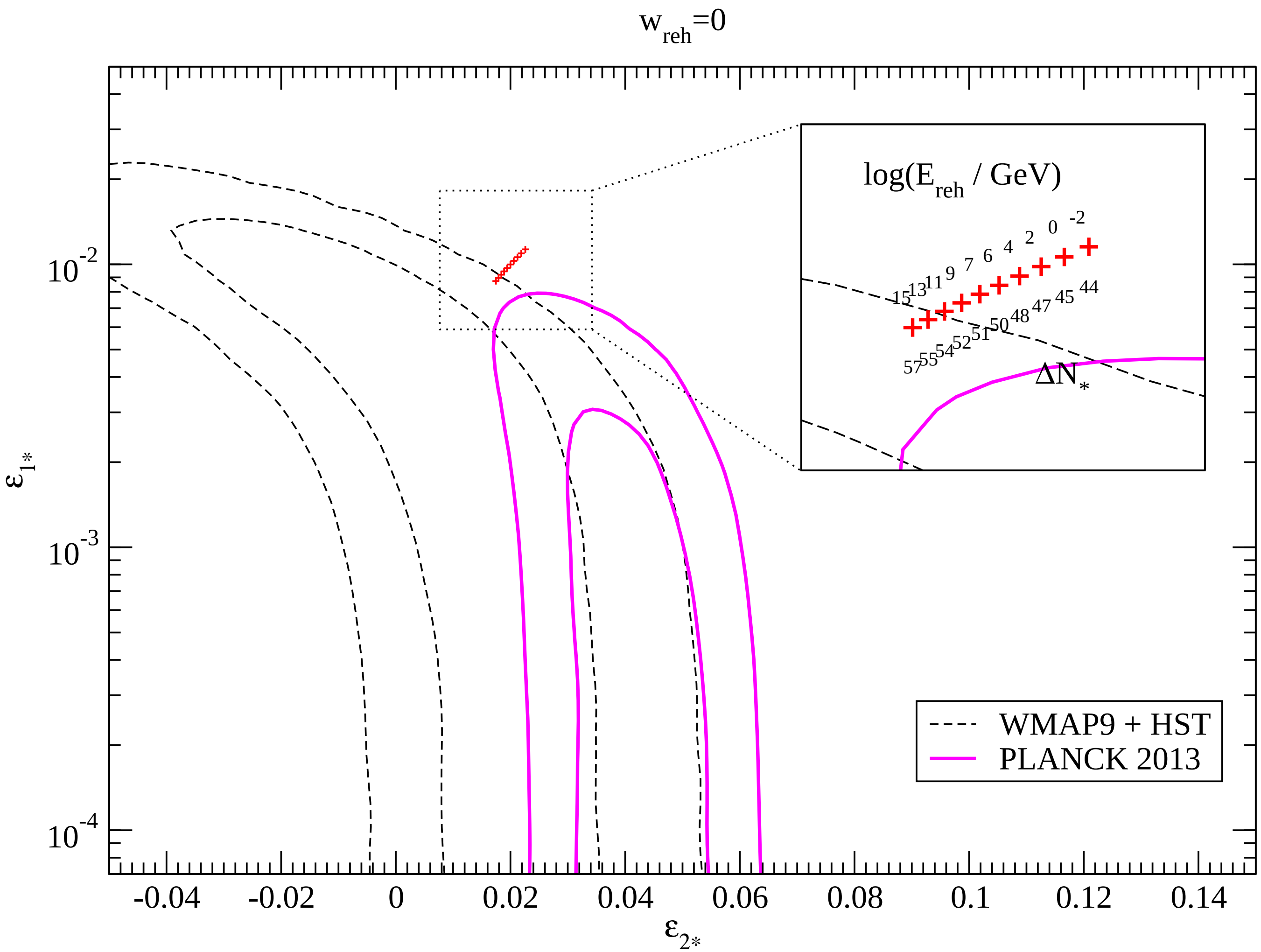
Constraints on the slow-roll parameters

- From the slow-roll expanded expression of $\mathcal{P}_\zeta(k)$ and $\mathcal{P}_h(k)$
  - ✦ Constraints on ϵ_{i*} and P_* (or H_*^2/ϵ_{1*})
  - ✦ Example from Planck 2013 and BICEP2



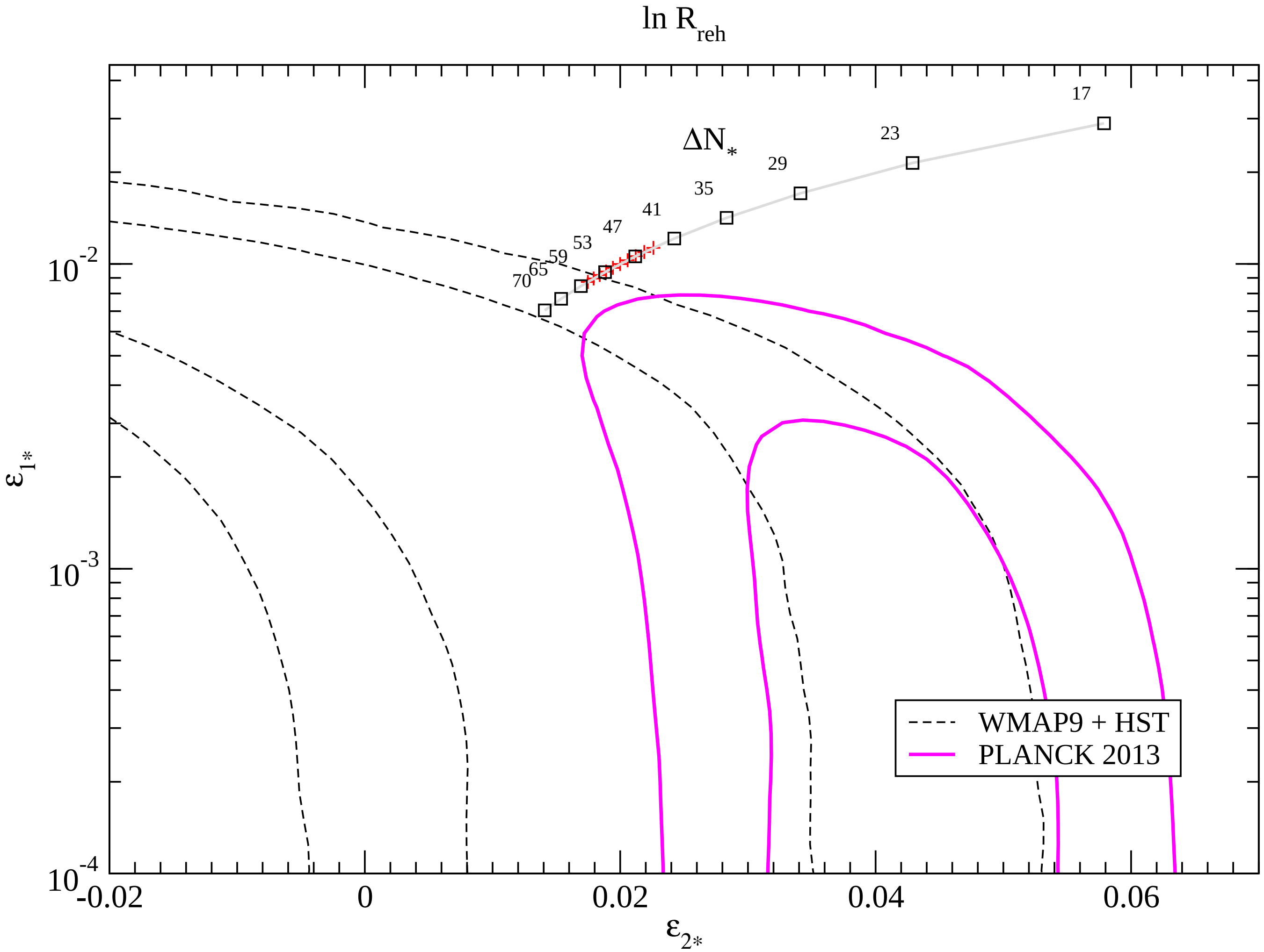
Comparison with model predictions

- Can only be done from the input of R_{reh}, or R_{rad}, or $(\bar{w}_{\mathrm{reh}}, \rho_{\mathrm{reh}})$
  - ✦ One can scan various reheating histories: ΔN_* is not arbitrary!
  - ✦ Example: LFI_2 with $\bar{w}_{\mathrm{reh}} = 0$ and $\rho_{\mathrm{nuc}} < \rho_{\mathrm{reh}} < \rho_{\mathrm{end}}$



Most generic reheating parametrization

- In the absence of any information on the reheating, one should use R_{reh} (or R_{rad})
- Same example: LFI_2 without assuming $\bar{w}_{\mathrm{reh}} = 0$

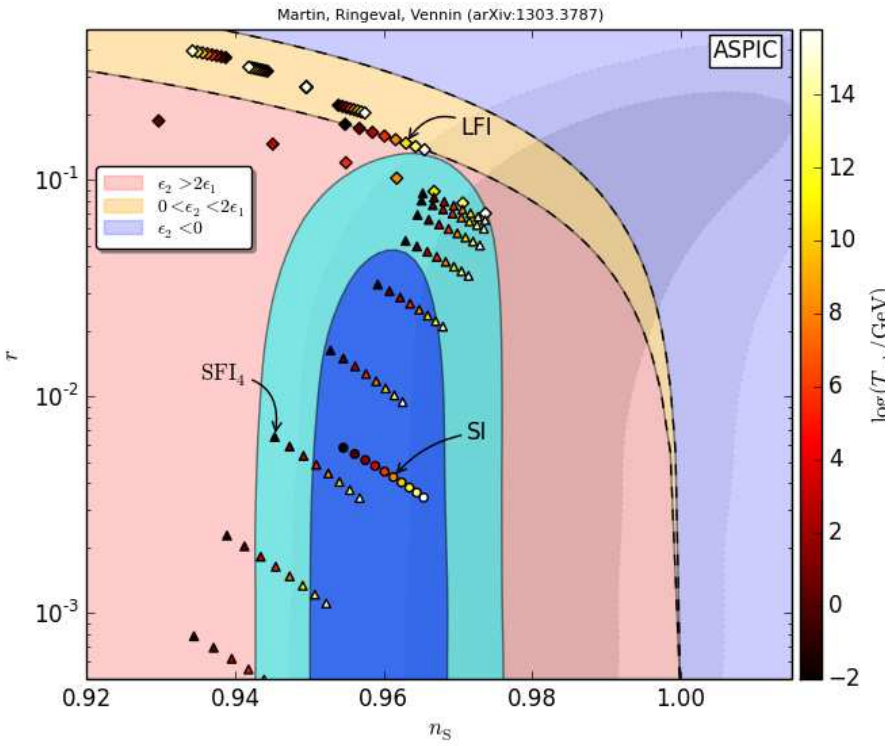


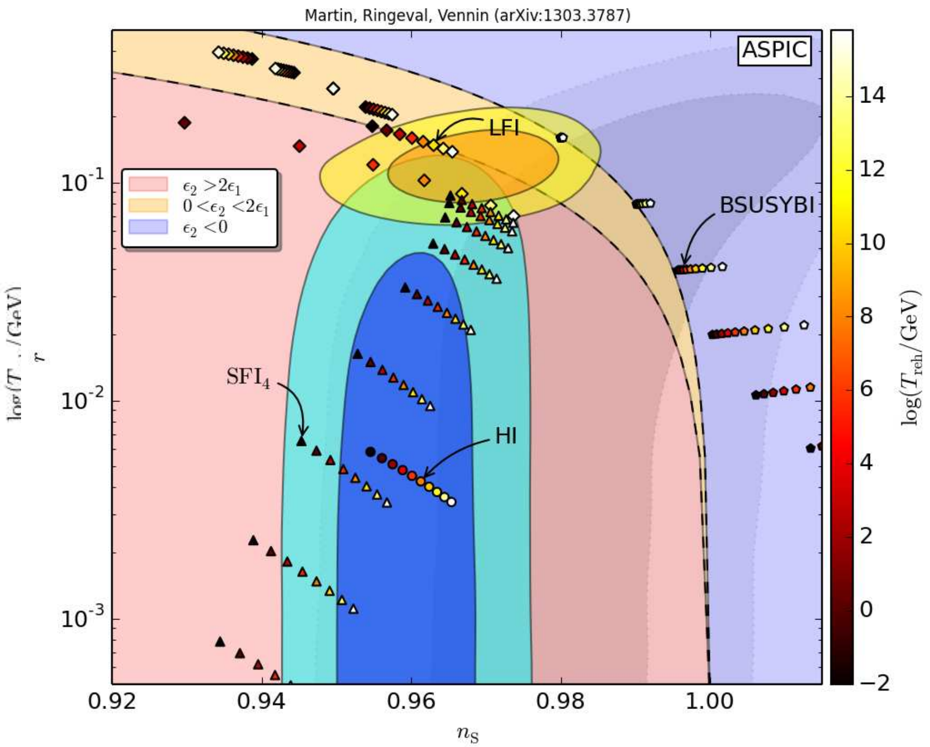
Model predictions with ASPIC

- For all *Encyclopædia Inflationaris* models

$$\text{potential parameters} + \text{reheating} \longrightarrow \epsilon_{i*} \longrightarrow n_{\mathrm{S}},\ r,\ \alpha_{\mathrm{S}} \ldots \text{(with consistency relations)}$$

- Easy to check for which reheating history a model is compatible with the data

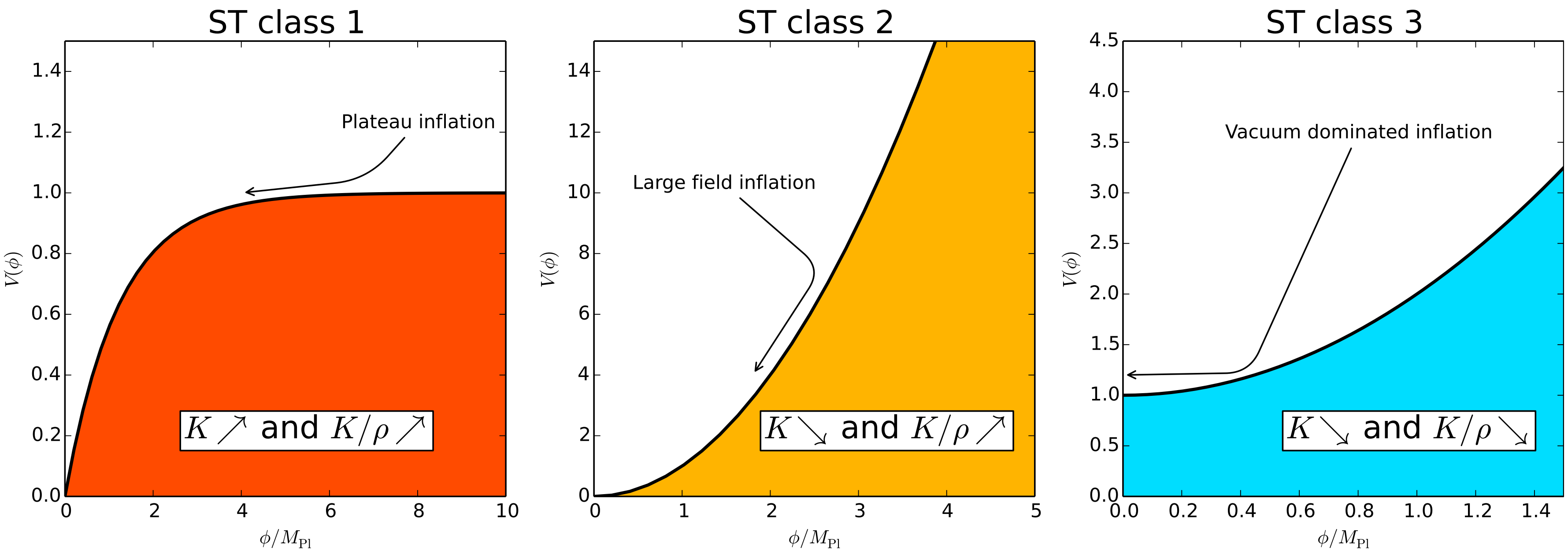




Schwarz Terrero-Escalante classification

- Based on the relative energy evolution at the pivot scale (ϕ_*)

$$K = \frac{1}{2}\dot{\phi}^2 \qquad \rho = K + V \qquad P = K - V \simeq -\rho$$



- In terms of slow-roll parameters

$$\text{ST1: } \epsilon_{2*} > 2\epsilon_{1*}, \quad \text{ST2: } 0 < \epsilon_{2*} < 2\epsilon_{1*}, \quad \text{ST3: } \epsilon_{2*} < 0$$

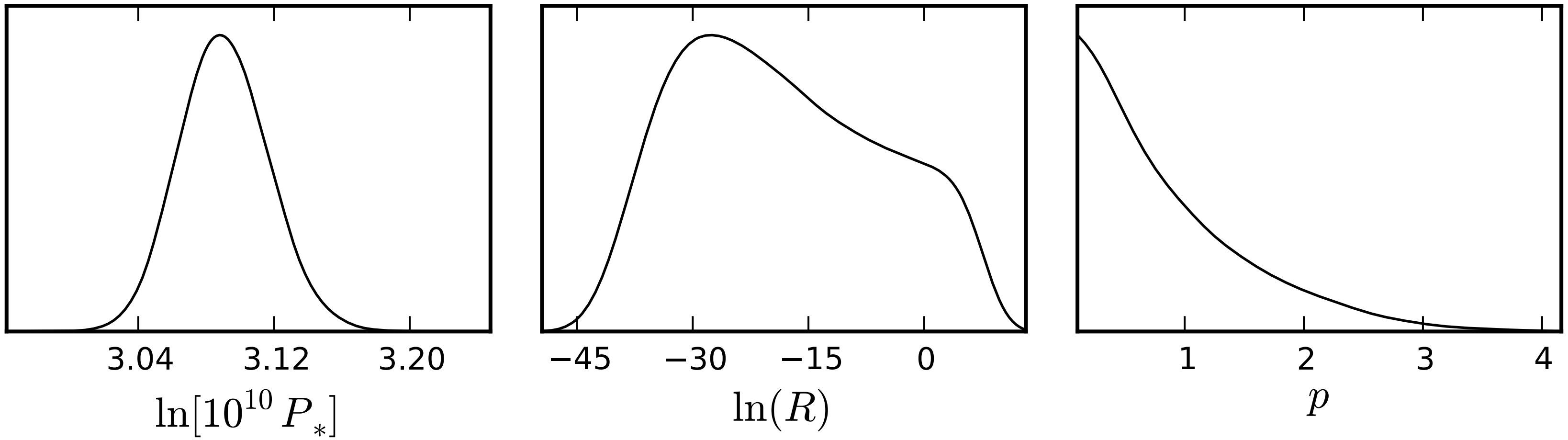
- This is not exactly the color of $\mathcal{P}_\zeta$: $n_{\rm S} - 1 = -2\epsilon_{1*} - \epsilon_{2*} + \mathcal{O}(\epsilon^2)$

Using the slow-roll approximation as a proxy

- To constrain the fundamental inflationary parameters: $\boldsymbol{\theta}_{\mathrm{inf}}$

$$(\boldsymbol{\theta}_{\mathrm{inf}}, R_{\mathrm{reh}}) \longrightarrow \texttt{ASPIC} \longrightarrow \epsilon_{i*} \longrightarrow \begin{cases} \mathcal{P}_\zeta(k) \\ \mathcal{P}_h(k) \end{cases} \longrightarrow \texttt{CAMB} \longleftrightarrow \text{CMB data}$$

- Example: Planck 2013 data analysis with LFI

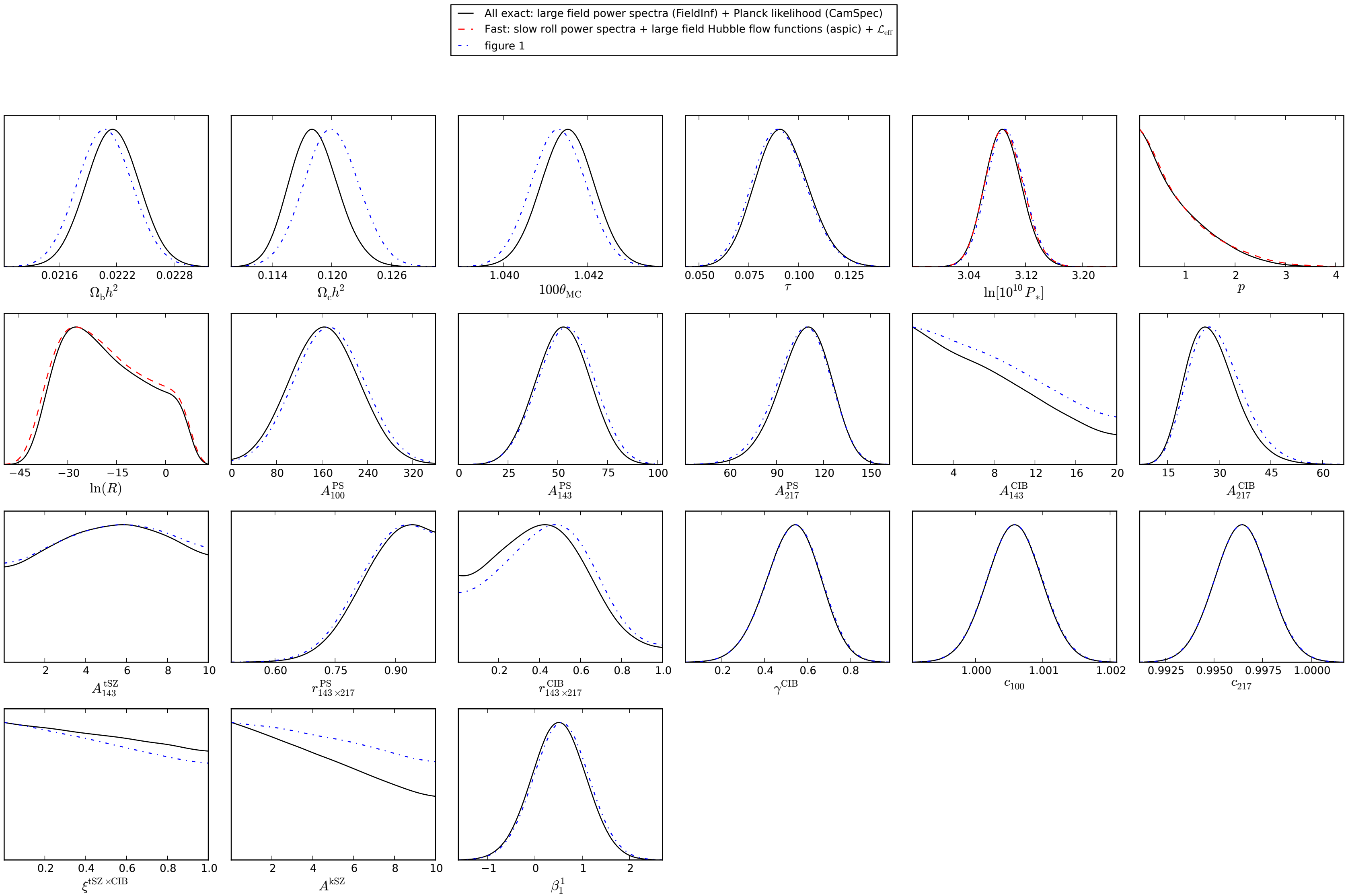


- Confidence intervals are on the relevant parameters (95% CL)

$$p < 2.3, \qquad -37 < \ln R_{\mathrm{reh}} < 6$$

Accuracy of the slow-roll approximation

- First order quantities marginalized over second order



Bayesian model comparison

- Bayesian evidence
  - For each model $\mathcal{M}$, marginalisation over all parameters

$$\mathcal{E}(D|\mathcal{M}) = \int \mathrm{d}\boldsymbol{\theta}\mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\mathcal{M})$$

  - Gives the posterior probability of $\mathcal{M}$ to explain the data D

$$p(\mathcal{M}|D) = \frac{\mathcal{E}(D|\mathcal{M})\pi(\mathcal{M})}{p(D)} \qquad \textit{where} \qquad p(D) = \sum_i \mathcal{E}(\mathcal{M}_i|D)\pi(\mathcal{M}_i)$$

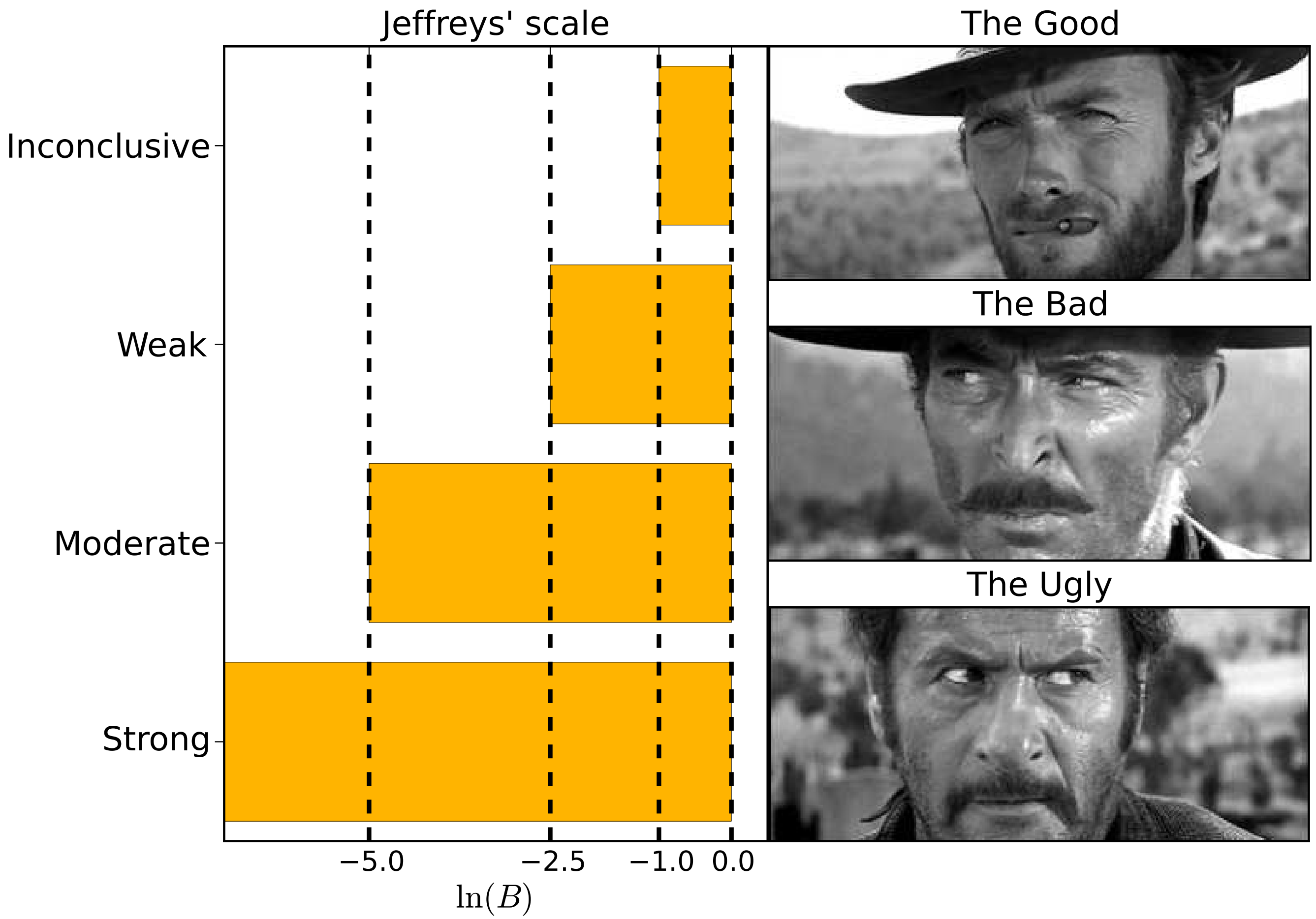
- Bayes' factor
  - Gives the posterior odds between $\mathcal{M}$ and a reference model $\mathcal{M}_0$

$$\frac{p(\mathcal{M}|D)}{p(\mathcal{M}_0|D)} = B\frac{\pi(\mathcal{M})}{\pi(\mathcal{M}_0)} \Rightarrow B = \frac{\mathcal{E}(D|\mathcal{M})}{\mathcal{E}(D|\mathcal{M}_0)}$$

  - Measure of how much the prior information has been updated

Jeffreys' scale

- Strength of evidence of $\mathcal{M}$ compared to $\mathcal{M}_0$



- ASPIC allows to fastly do that for all the *Encyclopædia Inflationaris* models

Bayes factor for hundred of models

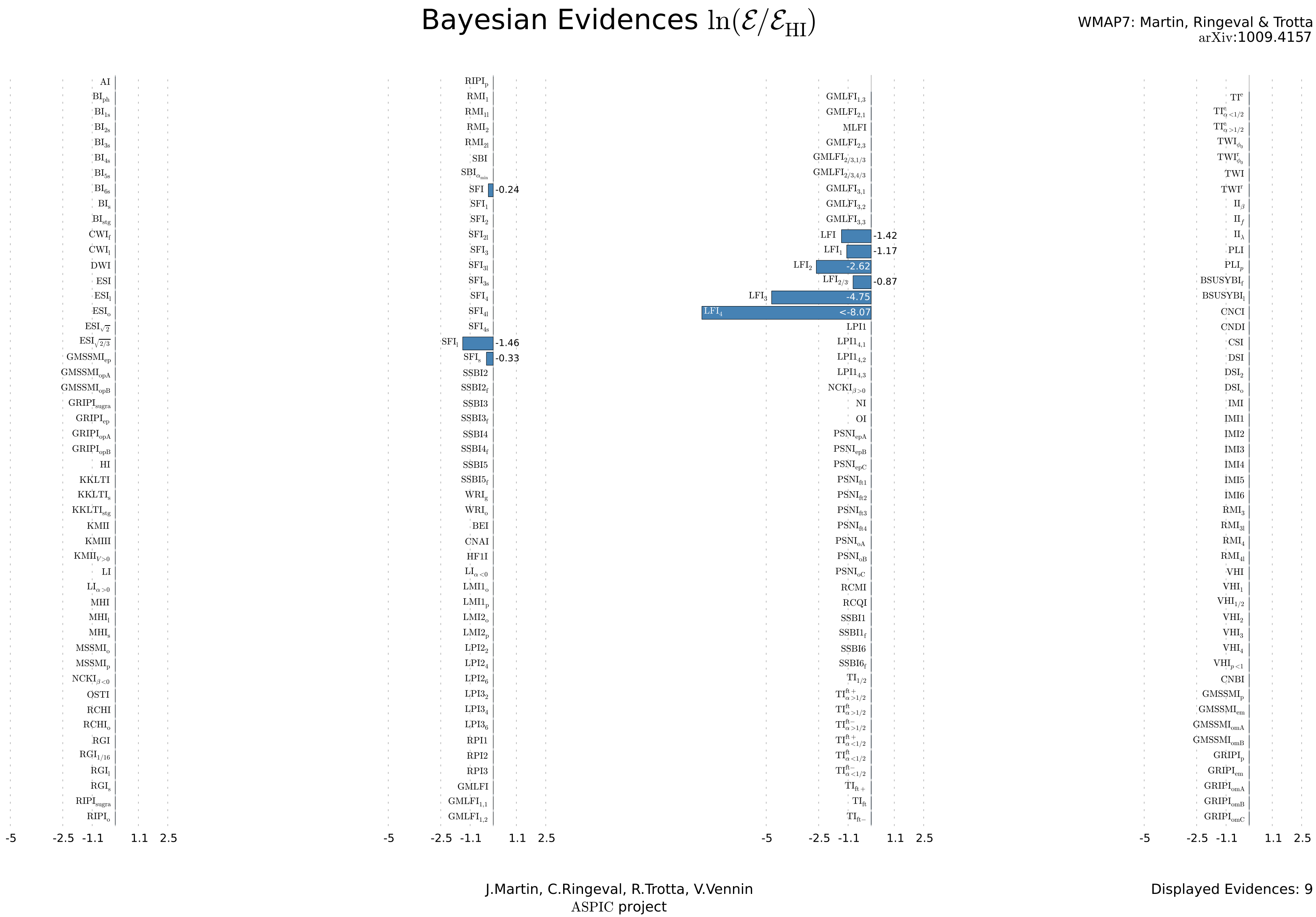
Bayesian Evidences $\ln(\mathcal{E}/\mathcal{E}_{\rm HI})$

J.Martin, C.Ringeval, R.Trotta, V.Vennin
ASPIC project

Displayed Evidences: 0

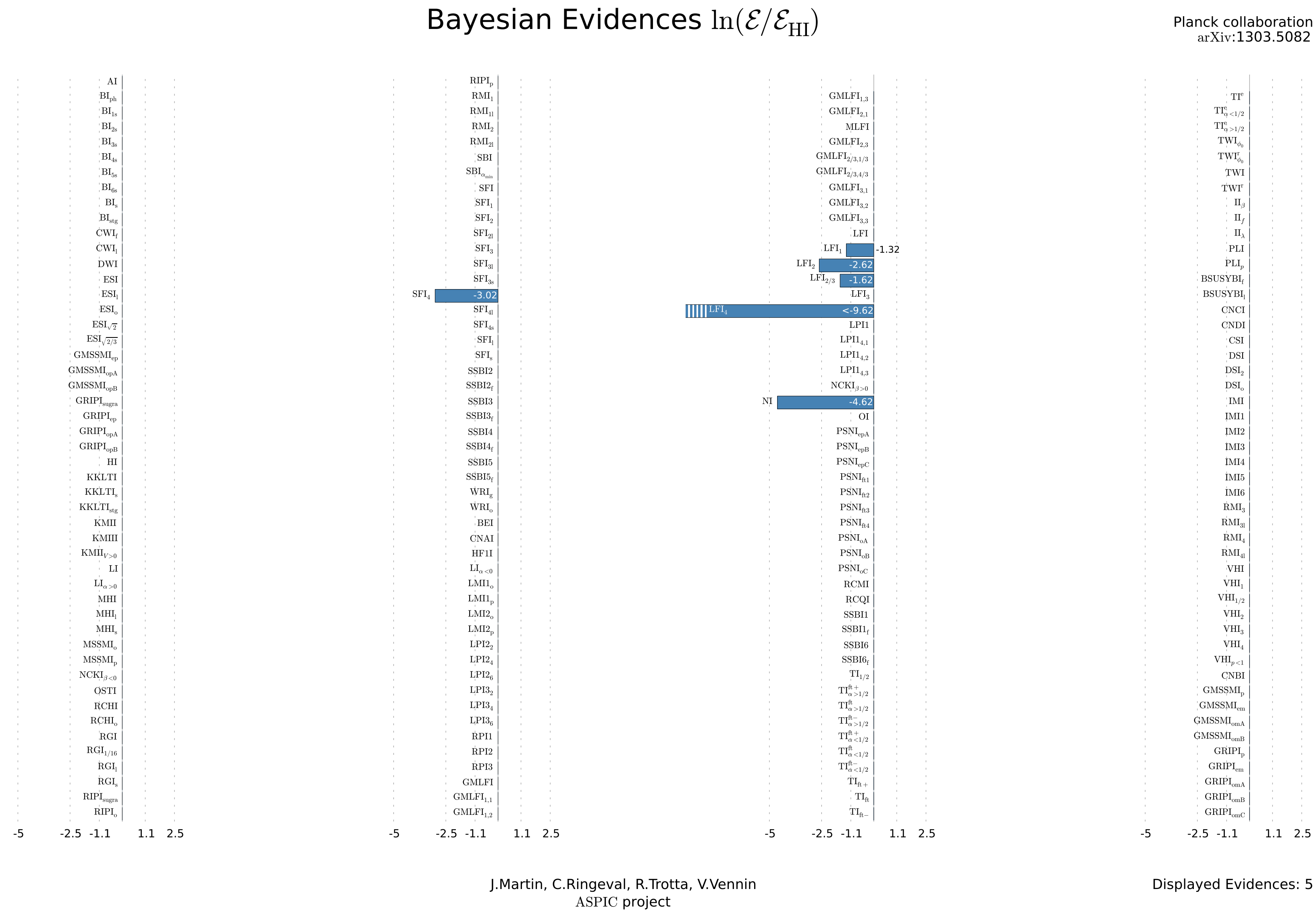
Bayes factor for hundred of models

WMAP7



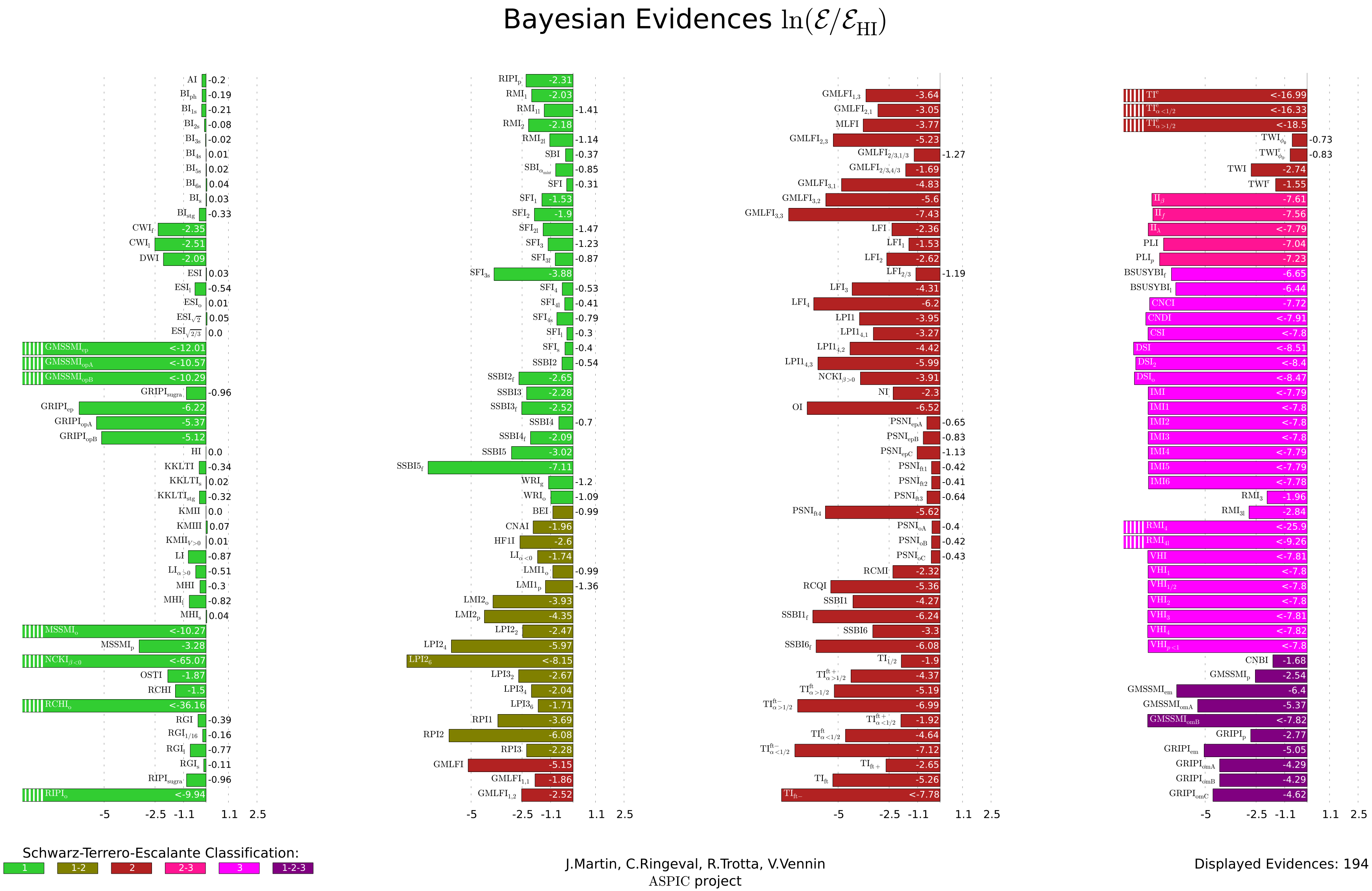
Bayes factor for hundred of models

PLANCK



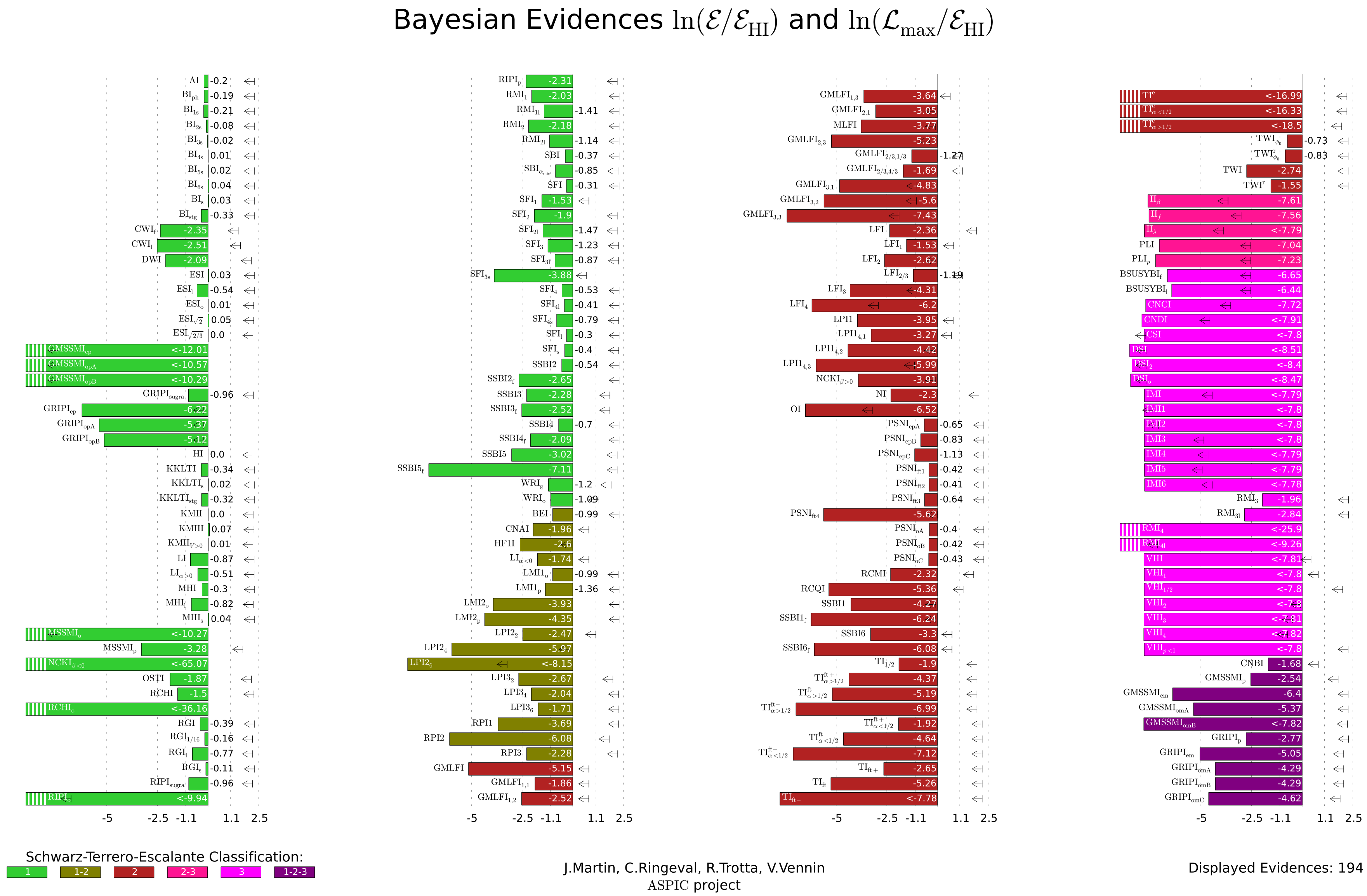
Bayes factor for hundred of models

PLANCK



Bayes factor for hundred of models

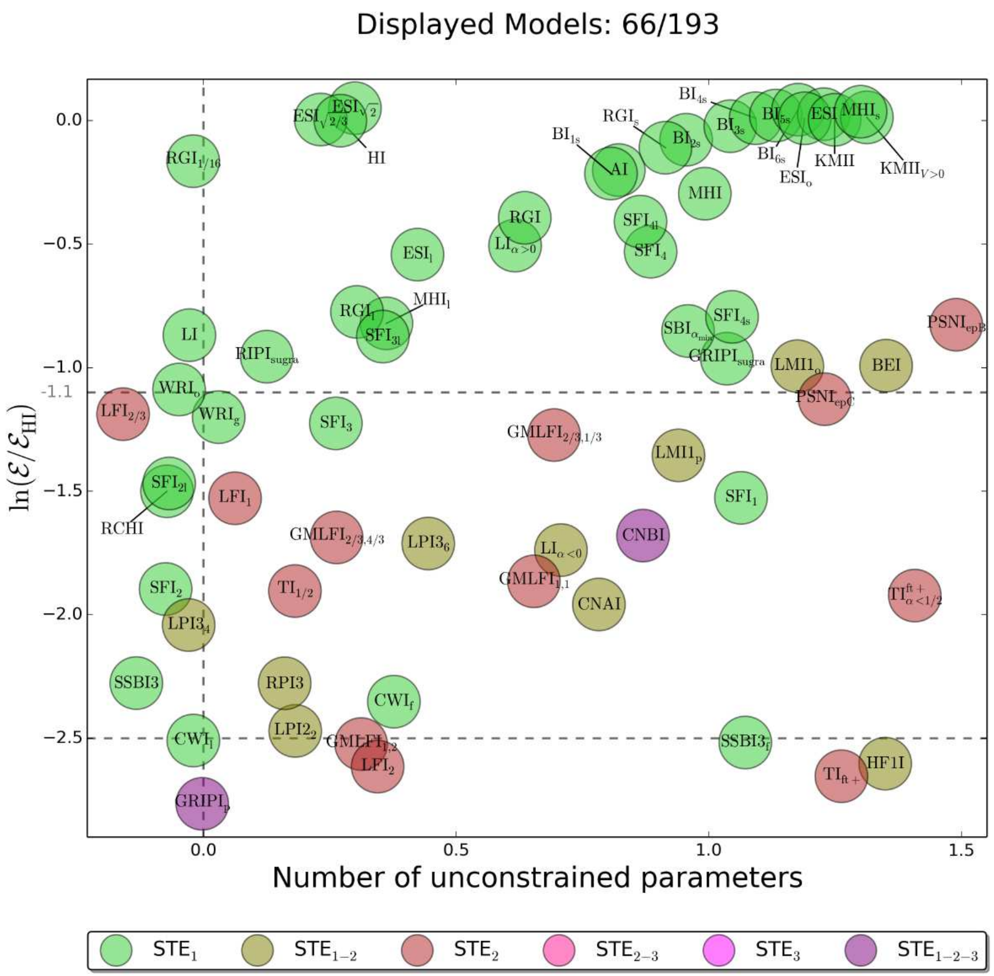
PLANCK



Narrowing down the simplest with complexity

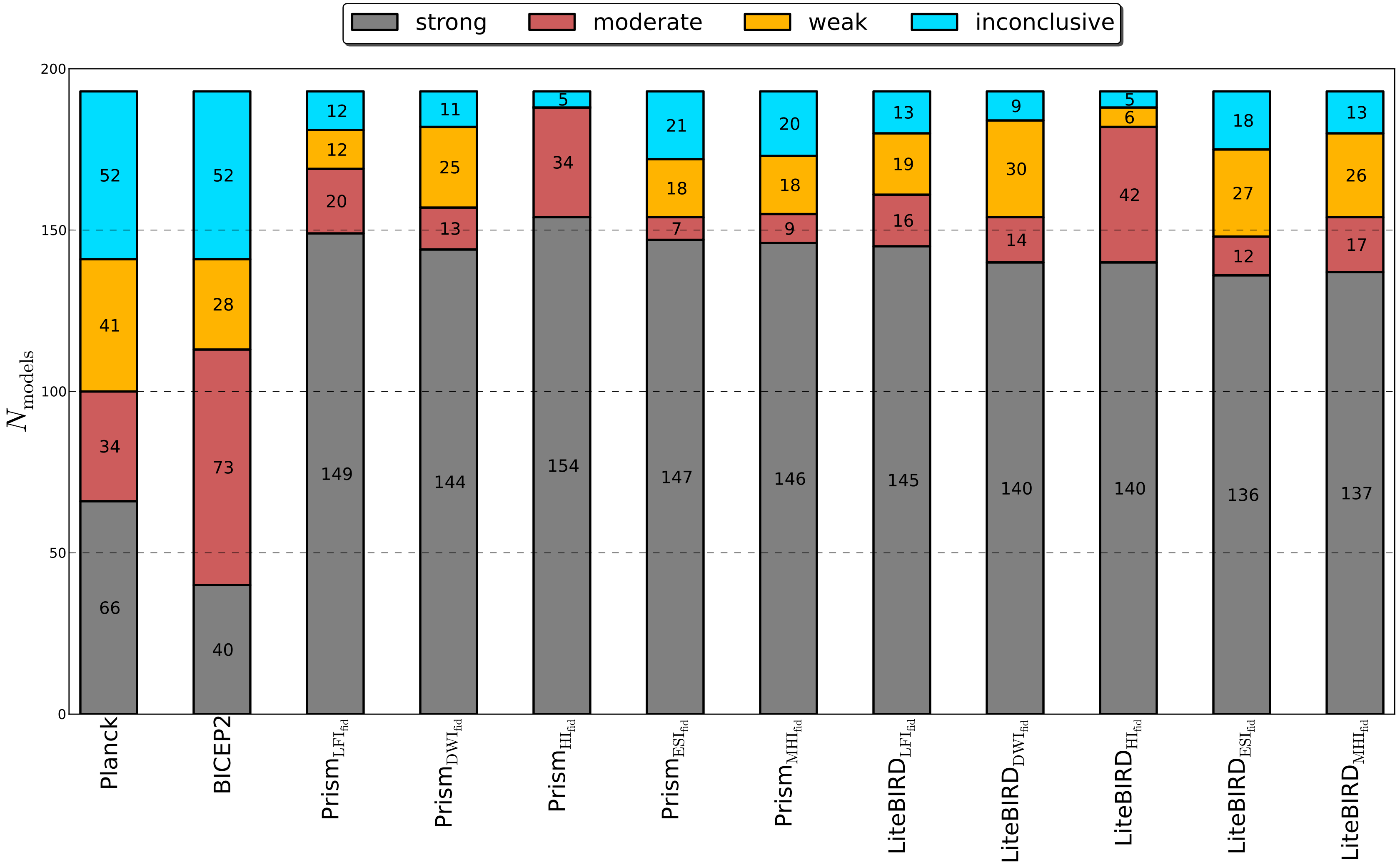
- Bayesian complexity $\simeq$ the number of constrained parameters

$$\mathcal{C} = \langle -2\ln\mathcal{L}\rangle + 2\ln\mathcal{L}_{\max} \quad \Rightarrow \quad N_{\mathrm{unconstrained}} = N_{\mathrm{param}} - \mathcal{C}$$



Data constraining power

- Comparison between PLANCK and future CMB experiments



Using the ASPIC library

Outline

Using the ASPIC library

Automated installation
Importing the library
More options
A toy program with LFI

Automated installation

- Released as a GNU software (license GPLv3)
  - Requirements: an Unix-like system (Linux, Mac,. . .) + fortran 08 compiler
  - Download the source code at: `http://cp3.irmp.ucl.ac.be/~ringeval/aspic.html`
  - Unpack the archive, configure, compile and install in `PREFIX`

```
tar -zxvf ./aspic-0.3.1.tar.gz
cd aspic-0.3.1/
./configure --prefix=/home...
make
make install
```

- Within `PREFIX`, standard Unix tree
  - Library in `lib/`
  - Include files in `include/` and documentation in `man/`

Importing the library

- Into your source code
  - ✦ Import everything or particular modules and functions

```
include 'aspic.h'
```

```
use lfisr, only : lfi_epsilon_one,
lfi_epsilon_two
use srflow, only : scalar_spectral_index
use srflow, only : tensor_to_scalar_ratio
```

- Link your code `test.f90` to the (already) installed library
  - ✦ ASPIC is in the library path of your system

```
gfortran -c test.f90
gfortran test.o -o test -laspic
```

  - ✦ ASPIC installed in PREFIX=/home/...

```
gfortran -I/home/.../include/aspic -c test.f90
gfortran test.o -o test -L/home.../lib -laspic
```

More options

- Exhaustive documentation
  - ✦ `http://cp3.irmp.ucl.ac.be/~ringeval/man/libaspic.html`
  - ✦ Installed on your system: `man libaspic` and `man aspic_???`

liblpi(3) Module convention liblpi(3)

NAME

lpi1 lpi2 lpi3 - the logarithmic potential inf ation modules

SYNOPSIS

Physical potential	V(phi) = Mˆ4 xˆp log(x)ˆq
Routine units	*real(kp)* :: **x** = phi/phi0
Parameters	*real(kp)* :: **p,q,phi0**

DESCRIPTION

The *lpi1* module is used for the logarithmic inf ation at large f eld values, namely in the region for which 'x > 1'. In this regime, inf ation proceeds at decreasing f eld values and naturally ends at 'xend' returned by the function **lpi1_x_endinf**(p,q,phi0).

The *lpi2* module is used for the logarithmic inf ation at intermediate f eld values, namely in the region for which 'xVmax < x < 1'. In this regime, inf ation proceeds at increasing f eld values and naturally ends at 'xend' returned by the function **lpi2_x_endinf**(p,q,phi0).

Finally, the *lpi3* module is used for the logarithmic inf ation at small f eld values, namely in the region for which '0 < x < xVmax'. In this regime, inf ation proceeds at decreasing f eld values and naturally ends at 'xend' returned by the function **lpi3_x_endinf**(p,q,phi0).

Shared functions can be found in a module named **lpicommon**. The value of 'xvMax' is returned by **lpi_x_potmax**(p,q,phi0) accessible through

use lmicommon, only : lmi_x_potmax

AUTHORS

Jerome Martin, Christophe Ringeval, Vincent Vennin

- Checkout the `README` file for more options and troubleshootings

A toy program with LFI


```
program toy
  use infprec, only : kp
  use lfisr, only : lfi_epsilon_one, lfi_epsilon_two
  use lfisr, only : lfi_epsilon_three, lfi_x_endinf
  use lfireheat, only : lfi_x_rreh, lfi_x_star
  use srflow, only : scalar_spectral_index, tensor_to_scalar_ratio
  use cosmopar, only : lnMpinGeV, PowerAmpScalar
  implicit none

  real(kp) :: lnR
  real(kp), dimension(3) :: eps

  real(kp) :: DeltaN
  real(kp) :: p, xstar, xend
  real(kp) :: ns, r

  real(kp) :: ErehGeV, wreh,lnRhoReh

  p=2

!radiation-like reheating
  lnR = 0._kp

  xend = lfi_x_endinf(p)
  xstar = lfi_x_rreh(p,lnR,DeltaN)

  print *,'xend= xstar= DeltaN= ',xend,xstar,DeltaN

  eps(1) = lfi_epsilon_one(xstar,p)
  eps(2) = lfi_epsilon_two(xstar,p)
  eps(3) = lfi_epsilon_three(xstar,p)

  ns = scalar_spectral_index(eps)
  r = tensor_to_scalar_ratio(eps)

  print *,'ns= r= ',ns,r

  read(*,*)

!matter like reheating at Ereh=10^8 GeV
  ErehGeV = 1e8
  wreh = 0

  lnRhoReh = 4._kp*(log(ErehGev)-lnMpinGev)

  xstar = lfi_x_star(p,wreh,lnRhoReh,PowerAmpScalar,DeltaN)

  print *,'xend= xstar= DeltaN= ',xend,xstar,DeltaN

  eps(1) = lfi_epsilon_one(xstar,p)
  eps(2) = lfi_epsilon_two(xstar,p)
  eps(3) = lfi_epsilon_three(xstar,p)

  ns = scalar_spectral_index(eps)
  r = tensor_to_scalar_ratio(eps)

  print *,'ns= r= ',ns,r

end program toy
```