Title: mochi_class: A tool to streamline cosmological analyses of Horndeski's gravity

Speakers: Matteo Cataneo

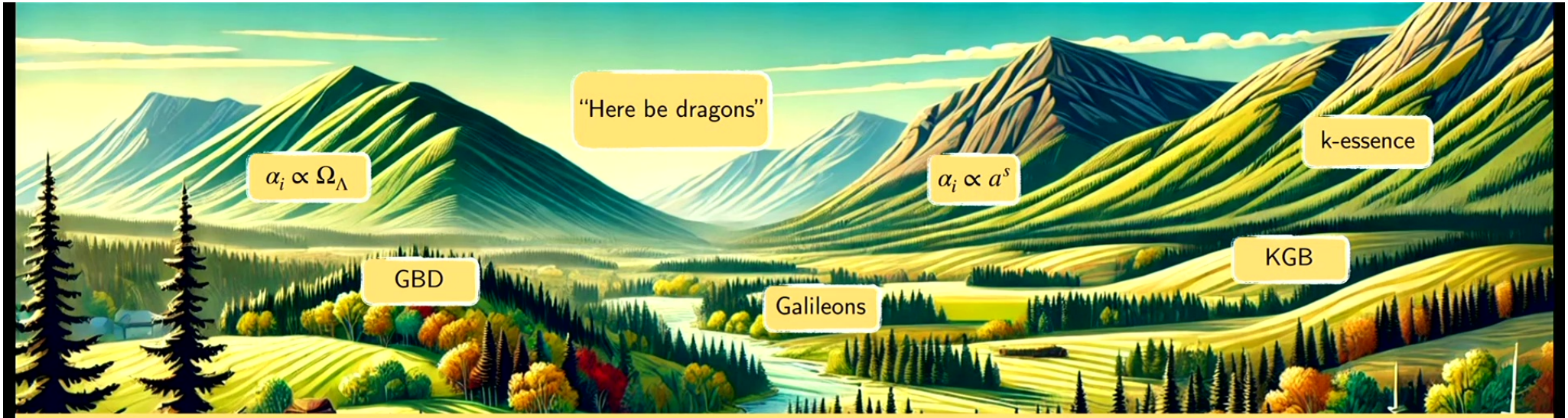
Collection/Series: 50 Years of Horndeski Gravity: Exploring Modified Gravity

Date: July 17, 2024 - 2:30 PM

URL: https://pirsa.org/24070068

Abstract:

I will introduce mochi_class, a refined version of the popular Einstein-Boltzmann solver hi_class optimised for calculations within Horndeski's gravity framework. Thanks to (i) a re-parametrisation of Horndeski functions, (ii) a numerically stable quasi-static approximation, and (iii) support for time-dependent inputs, mochi_class enhances hi_class capabilities and nicely complements other public Einstein-Boltzmann solvers. Additionally, I will present a non-parametric approach that, when integrated with Principal Component Analysis, can effectively reconstruct Horndeski functions from well-studied modified gravity models, extending the exploration of scalar-tensor theories beyond conventional parametrisations. I will conclude by highlighting practical applications where mochi_class can prove instrumental in analysing current and forthcoming large-scale structure data.



mochi_class:

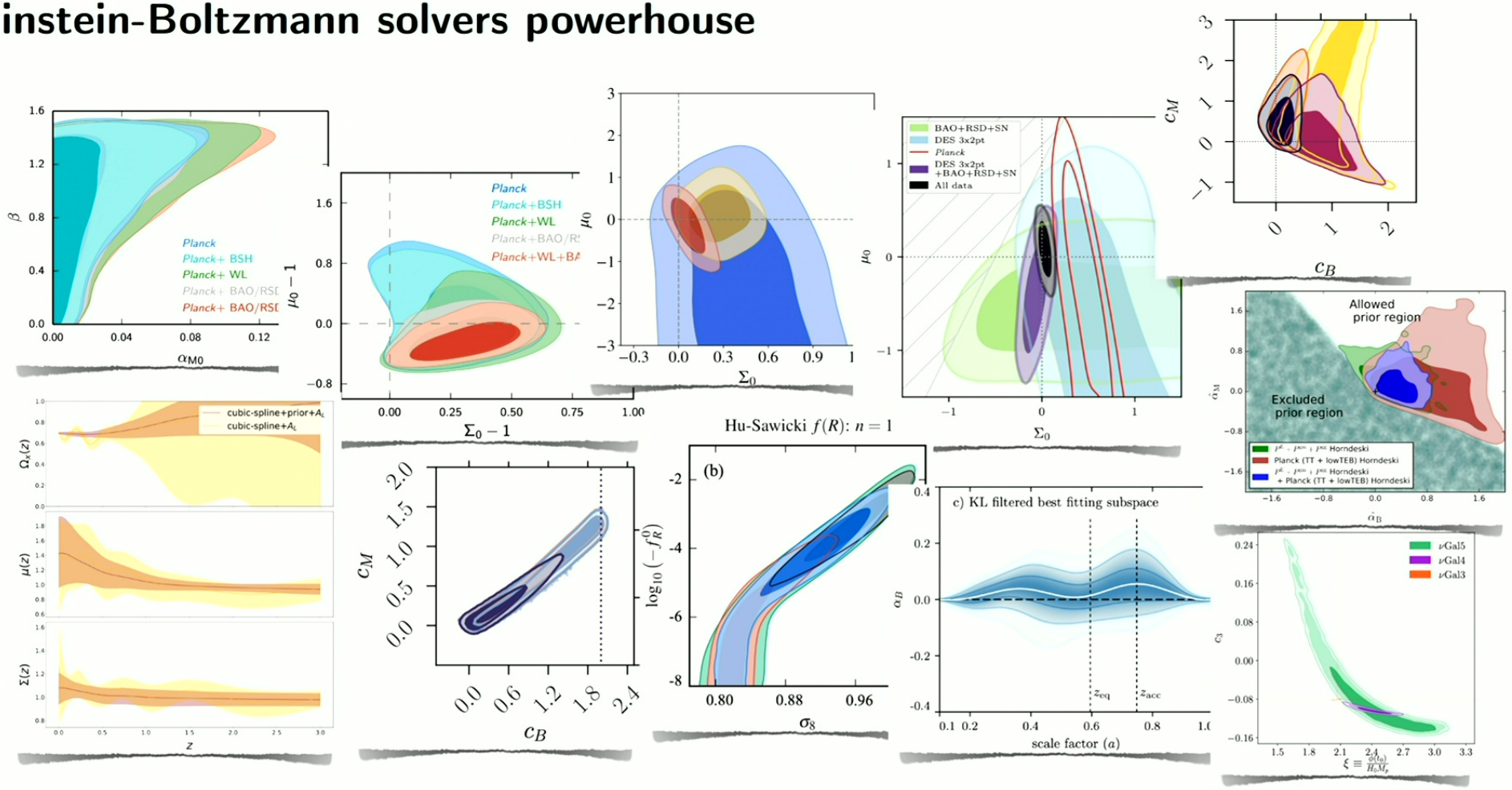
A numerical tool to streamline cosmological analyses of Horndeski gravity

arXiv:2407.11968

Matteo Cataneo, Perimeter Institute, 17/07/2024

In collaboration with: Emilio Bellini (SISSA)

Einstein-Boltzmann solvers powerhouse



Planck 2015, DES 2019, DES 2023, Seraille et al 2024, Noeller & Nicola 2018, Wang et al. 2023, Hu et al. 2016, Raveri 2019, Renk et al. 2017, Spurio Mancini et al. 2019...And many, many more!

The Einstein-Boltzmann solvers triad of marvels

- ✓ QSA
- ✓ Specific models
- ✓ Splined $\{\mu, \Sigma\}$
- ✓ $\{\mu, \Sigma\}, \{\mu, \gamma\}, \{Q, R\}$ parametrisations

- ✓ EFT of DE
- ✓ No QSA
- ✓ Specific models
- ✓ Lagrangian-to-EFT mapping
- ✓ EFT functions and α_i's parametrisations

- ✓ EFT of Horndeski w/ α_i's
- ✓ Selective QSA
- ✓ Specific models
- ✓ Lagrangian-to-EFT mapping
- ✓ α_i's parametrisations

But see also `ISiTGR`, `COOP`, `mgclass`...

Already got all we need, or do we?

	Horndeski gravity	EFT mapping	General parametrisation	Fully dynamical evolution	QSA	Maths stability	Python wrapper
`hi_class`	✔	(✔)(1)	✘	✔	(✔)(2)	✘	✔
`EFTCAMB`	✔	(✔)(1)	✘	✔(3)	✘	✔	✘
`MGCAMB`	(✔)(4)	✘	(✔)(5)	✘	✔	N/A	✔

(1) can lead to numerical instabilities
(2) can be inaccurate
(3) to speed up computation users can adjust two parameters, `EFTtoGR` and `EFTTurnOnPiInitial`, accuracy might be affected
(4) only a few particular models in the Horndeski class
(5) spline of phenomenological $\{\mu, \Sigma\}$

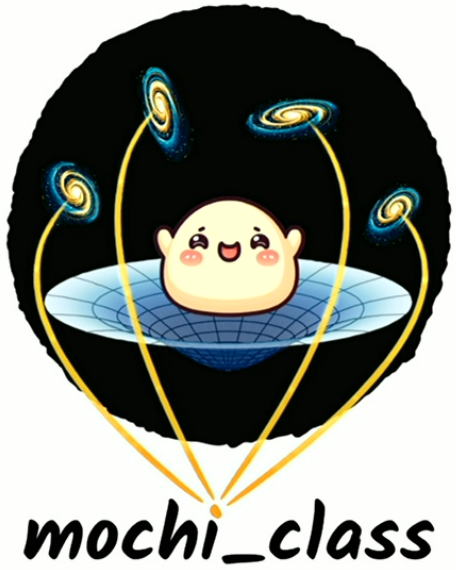
GAP: none of the public codes has all desired features

mochi_class: Modelling Optimisation to Compute Horndeski In CLASS

	Horndeski gravity	EFT mapping	General parametrisation	Fully dynamical evolution	QSA	Maths stability	Python wrapper
mochi_class	✔	✔	✔	✔	✔	✔	✔

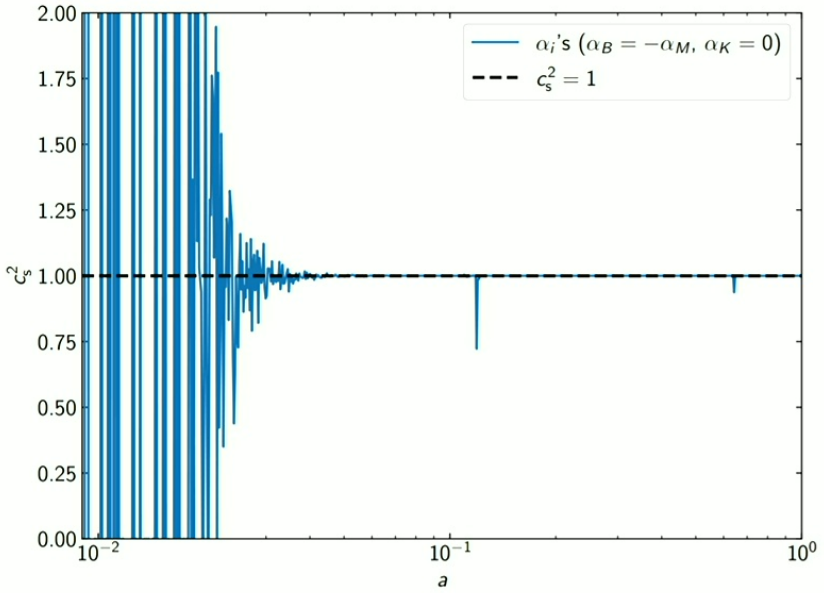
Restricted to theories with $\alpha_T = 0$:

$$\mathcal{L}_{\rm SH} = K(\phi, X) + G_4(\phi)R - G_3(\phi, X)\Box\phi$$



`mochi_class`: MODELLING OPTIMISATION TO COMPUTE HORNDESKI IN CLASS

1) Implements manifestly stable parametrisation (Kennedy et al. 2018) & 2) Avoids numerical instabilities in c_s^2



Stability conditions:

$$M_*^2 > 0$$

$$D_{\rm kin} = \alpha_{\rm K} + 3\alpha_{\rm B}^2/2 > 0$$

$$c_s^2 = \frac{1}{D_{\rm kin}}\left[(2-\alpha_{\rm B})\left(-\frac{H'}{aH^2} + \frac{1}{2}\alpha_{\rm B} + \alpha_{\rm M}\right) - \frac{3(\rho_{\rm m} + p_{\rm m})}{H^2 M_*^2} + \frac{\alpha_{\rm B}'}{aH}\right] > 0$$

Solution: $\alpha_{\rm K}, \alpha_{\rm B}, \alpha_{\rm M} \Longrightarrow M_*^2, D_{\rm kin}, c_s^2$

mochi_class: Modelling Optimisation to Compute Horndeski In CLASS

3) Takes general functions of time as input

```
else if(pba->gravity_model_smg == nkgb){

  /* Action is

    -X + 1/n * g^[(2n-1)/2] Lambda (X/Lambda^4)^n box(phi)

    g was picked like this so that it approx. remains g*Omega_smg0 ~ O(1) for all n
    Note that for n=1/4 the Lambda mass scales cancels out, so we set it to 1.
  */

  double g = pba->parameters_smg[0];
  double npow = pba->parameters_smg[1];
  double ngpow = copysign(1.,g)*pow(fabs(g),(2.*npow-1.)/2.)/npow;
  double H0=pba->H0;

  pgf->G2    = -X;
  pgf->G2_X  = -1.;

  // pgf->G3 = 1/n g^[(2n-1)/2] Lambda (X/Lambda^4)^n

  pgf->G3_X = npow*ngpow*pow(X,npow-1)/pow(H0,2*npow);
  pgf->G3_XX = npow*(npow-1.)*ngpow*pow(X,npow-2)/pow(H0,2*npow);

}
```

```
if (pba->gravity_model_smg == propto_omega) {

  double c_k = pba->parameters_2_smg[0];
  double c_b = pba->parameters_2_smg[1];
  double c_m = pba->parameters_2_smg[2];
  double c_t = pba->parameters_2_smg[3];

  pvecback[pba->index_bg_kineticity_smg] = c_k*Omega_smg;
  pvecback[pba->index_bg_braiding_smg] = c_b*Omega_smg;
  pvecback[pba->index_bg_tensor_excess_smg] = c_t*Omega_smg;
  pvecback[pba->index_bg_mpl_running_smg] = c_m*Omega_smg;
  pvecback[pba->index_bg_delta_M2_smg] = delta_M_pl; //M2-1
  pvecback[pba->index_bg_M2_smg] = 1.+delta_M_pl;
}
```

Solution: $M_*^2, D_{\rm kin}, c_{\rm s}^2, \tilde{\rho}_\phi$ or w_ϕ can be read from file or input as arrays

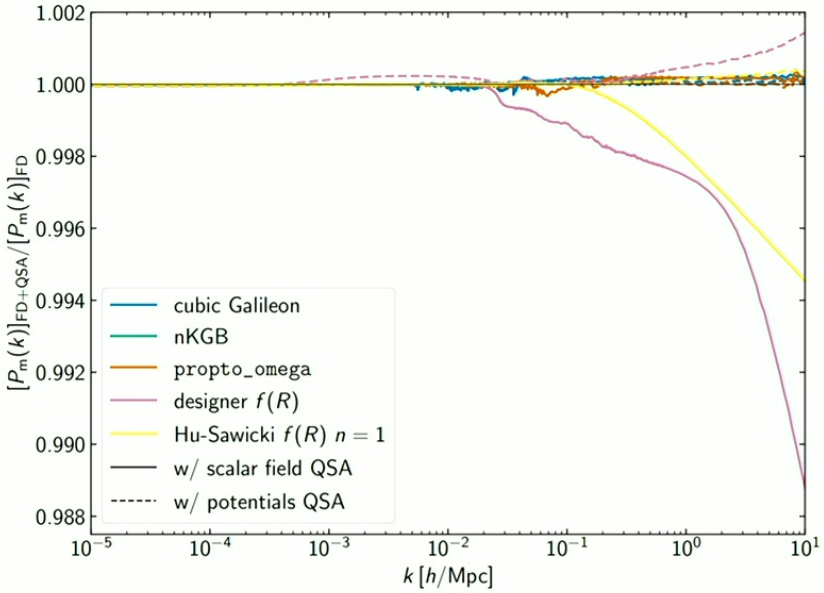
`mochi_class`: MODELLING OPTIMISATION TO COMPUTE HORNDESKI IN CLASS

4) Replaces `hi_class` QSA with $\{\mu, \gamma\}$ à la `MGCAMB`, and connects it to the EFT functions (Pogosian & Silvestri 2016)

In `hi_class`:

$$V_X \approx \frac{\mathcal{S}(\tau, k, \eta, \delta\rho, \delta p)}{a^2 H^2 \mu_{\rm eff}^2(\tau, k)}$$

$$V_X' \approx \frac{\mathrm{d}}{\mathrm{d}\tau}\left[\frac{\mathcal{S}(\tau, k, \eta, \delta\rho, \delta p)}{a^2 H^2 \mu_{\rm eff}^2(\tau, k)}\right]$$

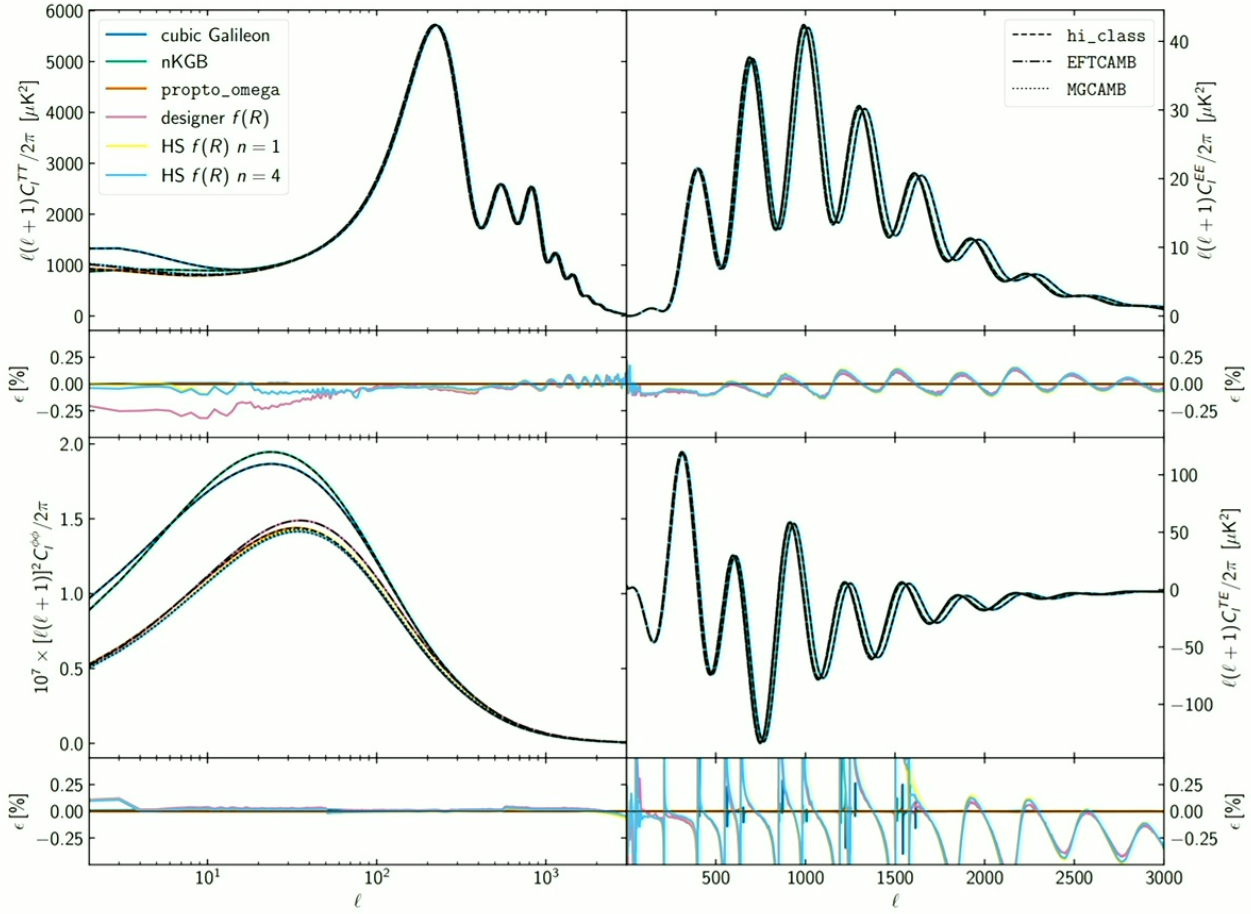


Solution:

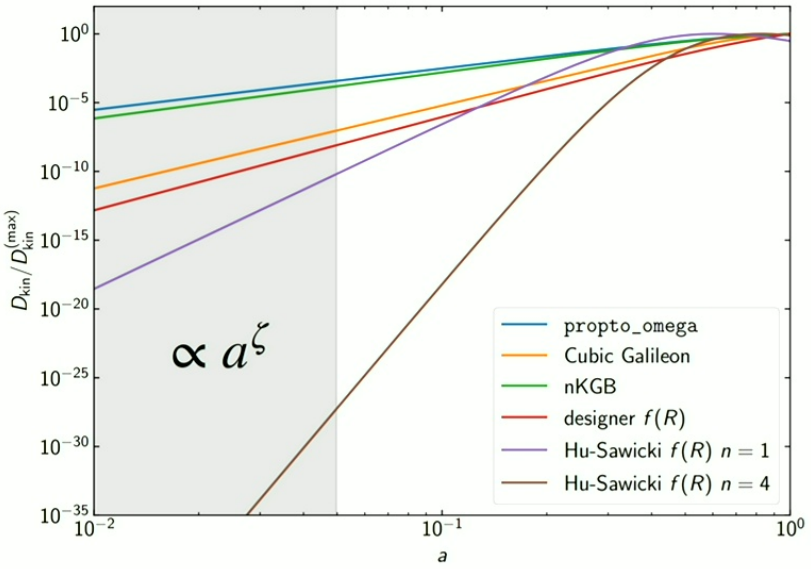
$$k^2\Psi = -4\pi G a^2 \mu(a, k)\rho_{\rm m}\Delta_{\rm m}$$

$$\gamma(a, k) = \Phi/\Psi$$

Code comparison (1)



Extending common parametrisations



$$\Delta M_*^2 \approx A_M e^{\zeta_M x + \Delta_1(x)} \qquad c_{\rm s}^2 \approx (A_{c_{\rm s}} e^{\zeta_{c_{\rm s}} x} + C_{c_{\rm s}}) e^{\Delta_3(x)}$$

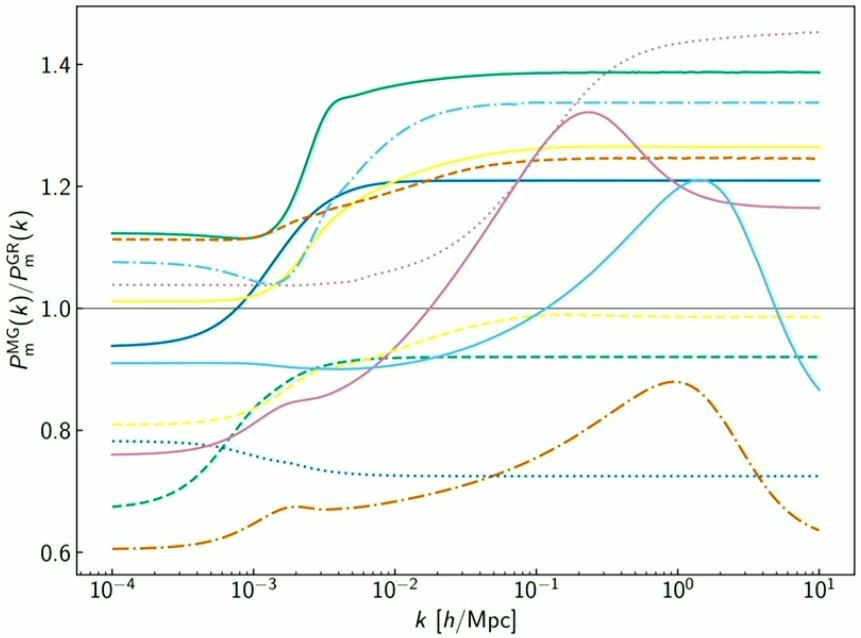
$$D_{\rm kin} \approx A_D e^{\zeta_D x + \Delta_2(x)} \qquad \tilde{\rho}_\phi \approx (A_\rho e^{\zeta_\rho x} + C_\rho)\Delta_4(x)$$

And we want

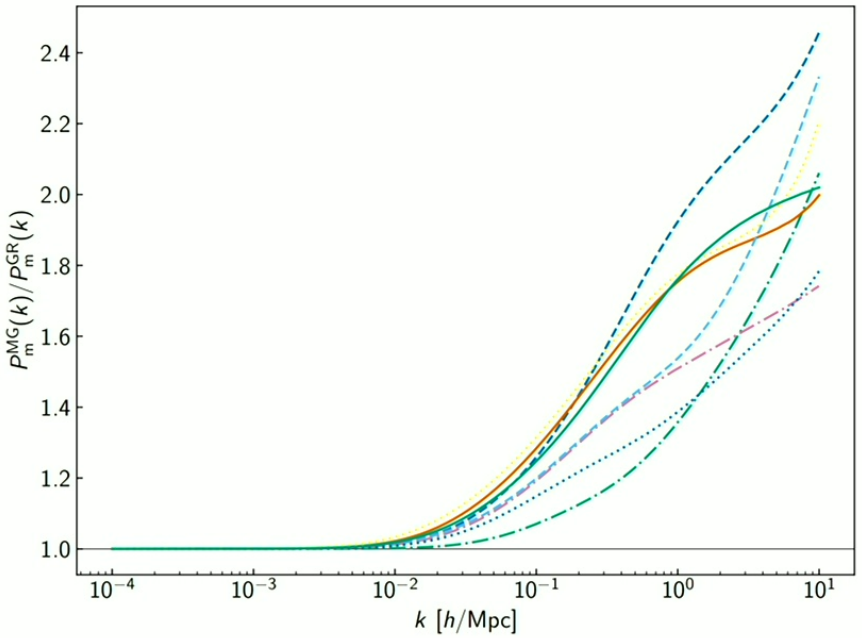
1) $\Delta_{1\text{-}3}(x) \longrightarrow 0$, $\Delta_4(x) \longrightarrow 1$ for $x \ll 0$
2) All $\Delta_i(x)$ to be varying smoothly

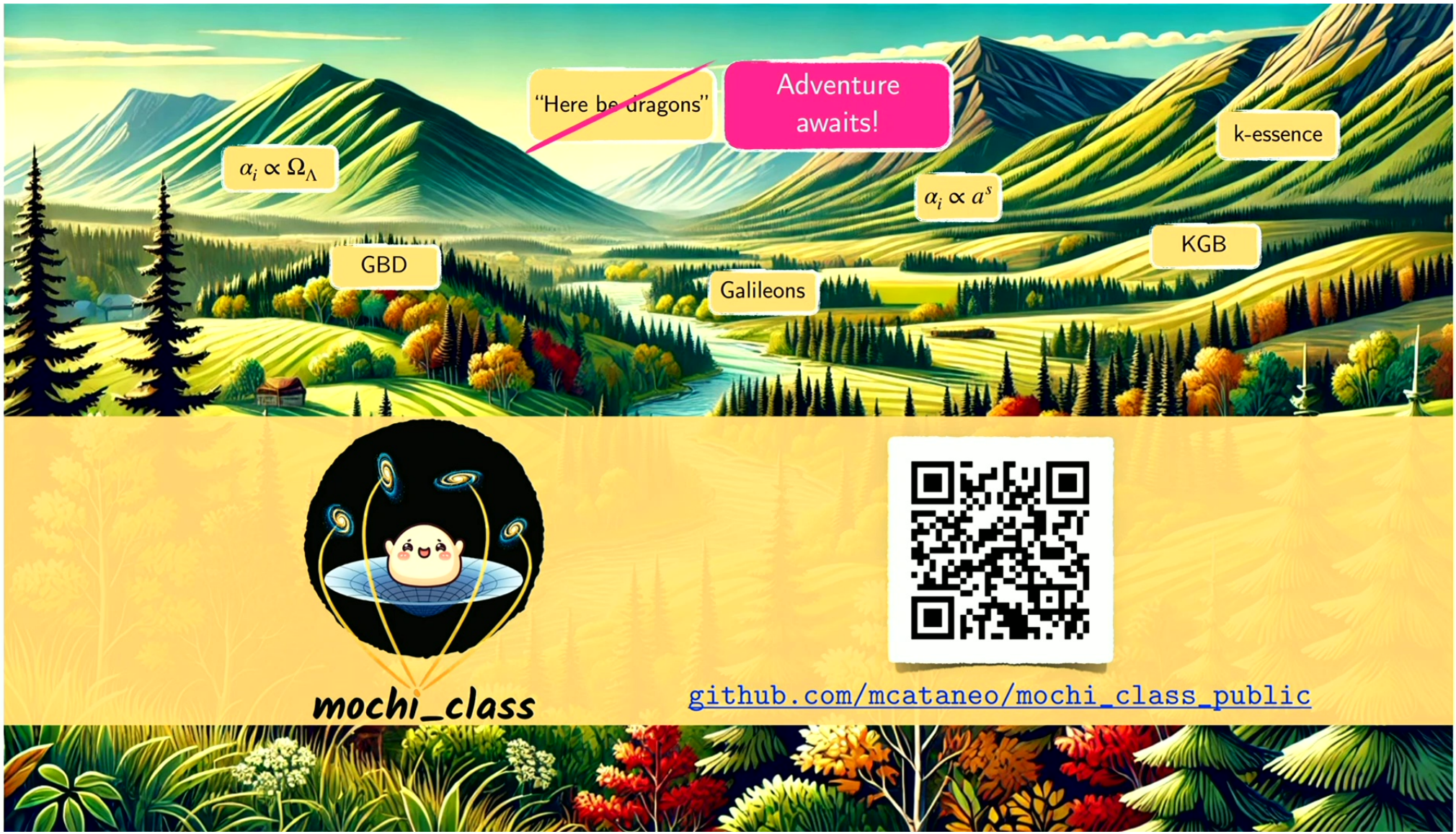
Let's generate new models!

Scalar Horndeski (31 parameters) - 5% 'survival rate'



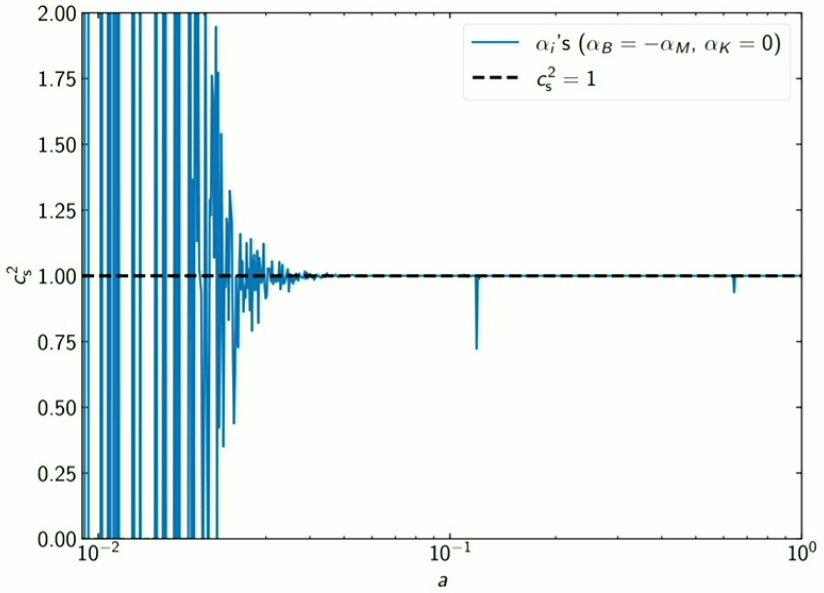
f(R)-like (16 parameters) - 20% 'survival rate'




"Here be dragons"
Adventure awaits!
$\alpha_i \propto \Omega_\Lambda$
$\alpha_i \propto a^s$
k-essence
GBD
Galileons
KGB
mochi_class
github.com/mcataneo/mochi_class_public

`mochi_class`: MODELLING OPTIMISATION TO COMPUTE HORNDESKI IN CLASS

1) Implements manifestly stable parametrisation (Kennedy et al. 2018) & 2) Avoids numerical instabilities in c_s^2



Stability conditions:

$$M_*^2 > 0$$

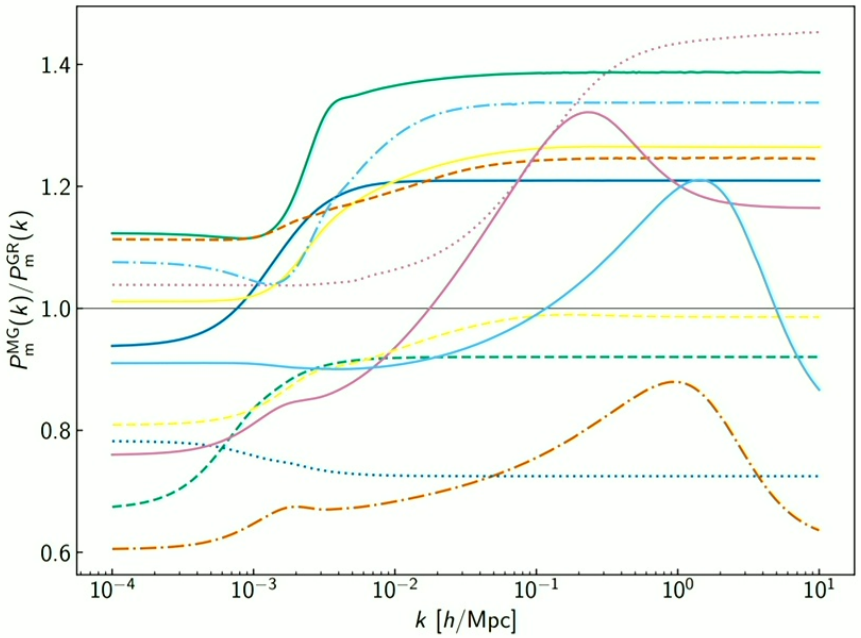
$$D_{\rm kin} = \alpha_{\rm K} + 3\alpha_{\rm B}^2/2 > 0$$

$$c_s^2 = \frac{1}{D_{\rm kin}}\left[(2-\alpha_{\rm B})\left(-\frac{H'}{aH^2} + \frac{1}{2}\alpha_{\rm B} + \alpha_{\rm M}\right) - \frac{3(\rho_{\rm m} + p_{\rm m})}{H^2 M_*^2} + \frac{\alpha_{\rm B}'}{aH}\right] > 0$$

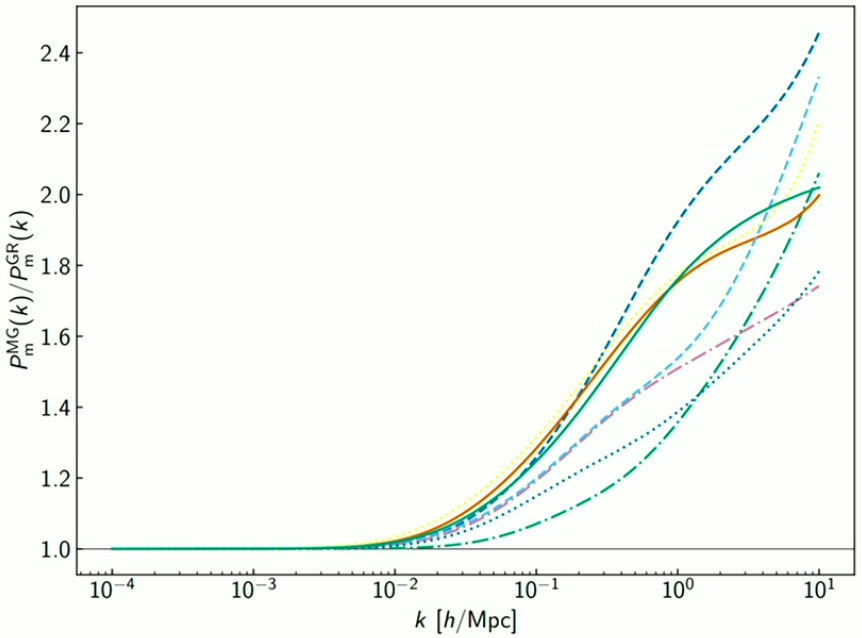
Solution: $\alpha_{\rm K}, \alpha_{\rm B}, \alpha_{\rm M} \Longrightarrow M_*^2, D_{\rm kin}, c_s^2$

Let's generate new models!

Scalar Horndeski (31 parameters) - 5% 'survival rate'



f(R)-like (16 parameters) - 20% 'survival rate'



Mathematical stability

- Scalar field fluctuations EOM

$$\mathcal{A}(\tau)V_X'' + \mathcal{B}(\tau)V_X' + \left[\mathcal{C}(\tau) + k^2\mathcal{D}(\tau)\right] V_X = \mathcal{E}(\tau, k)$$

- If $\mathcal{B}^2(\tau) - 4\mathcal{A}(\tau)[\mathcal{C}(\tau) + k^2\mathcal{D}(\tau)] > 0$:

$$\frac{-\mathcal{B}(\tau) \pm \sqrt{\mathcal{B}^2(\tau) - 4\mathcal{A}(\tau)[\mathcal{C}(\tau) + k^2\mathcal{D}(\tau)]}}{2\mathcal{A}(\tau)} < \xi H_0$$

- If $\mathcal{B}^2(\tau) - 4\mathcal{A}(\tau)[\mathcal{C}(\tau) + k^2\mathcal{D}(\tau)] < 0$:

$$-\frac{\mathcal{B}(\tau)}{2\mathcal{A}(\tau)} < \xi H_0$$