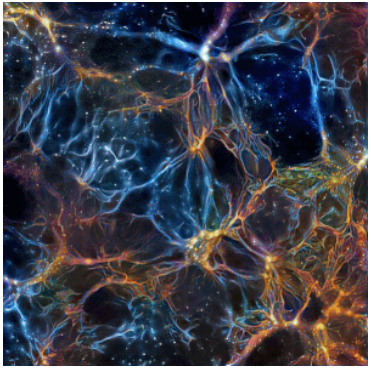
Tutorial for cosmological N-body simulations

Clément Stahl

Observatoire Astronomique de Strasbourg, Université de Strasbourg

27-28 November 2023



Credit: Quijote

Strasbourg

- 290 000 inhabitants (8th of France)
- Sucks at football
- City center is UNESCO
- Capital of french east region: Alsace
- Capital of Europe
- Bounced between France and Germany
  - 1262: German
  - 1681: French
  - 1871: German
  - 1918: French
  - 1940: German
  - 1945: French

Chile vs Alsace

Chile vs Alsace

Buildings of Strasbourg

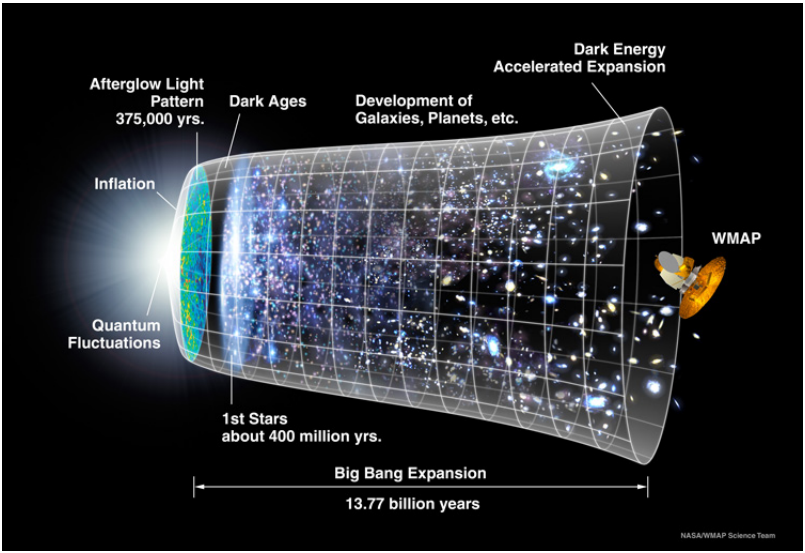
Notre Dame de Strasbourg

- 1176 $\rightarrow$ 1439: construction
- 1647 $\rightarrow$ 1874: Tallest building on earth (142m)
- 1842: Astronomical clock

Observatoire Astronomique de Strasbourg

- 1881: grand opening with largest refracting telescopes in Europe.
- today $\sim$ 80 people:
  - CDS: collection, curation & worldwide distribution of astronomical data
  - Galaxies, High Energy, Cosmology, Compact Objects & Stars

Cosmology in one slide

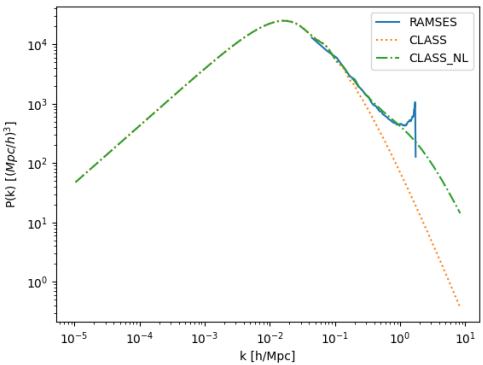


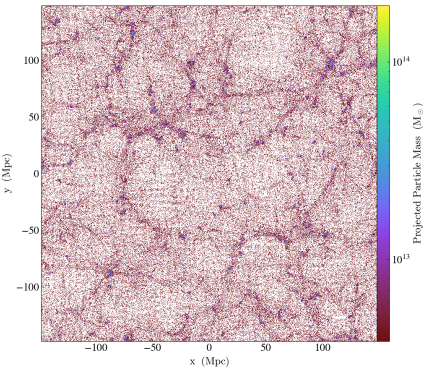
Cosmological structures formation

Fluids mechanics in an expanding universe.

Outline


linear
CLASS
evolution
Monofonic
realization
non-linear
RAMSES
evolution
visualization
YT
summary statistics


RAMSES
CLASS
CLASS_NL
P(k) [(Mpc/h)³]
k [h/Mpc]


y (Mpc)
x (Mpc)
Projected Particle Mass (M⊙)

Hands-on session

- **NOT** a course of cosmology, I'll give the basic framework and we dive into the practice
- Database of tools: HEP-ASTRO-COSMO and WG Tools and Methodology $\rightarrow$ **don't be afraid to ask** me, the community, chat GPT...
- Any idea how to do original research in that topic?

ChatGPT ChatGPT 3.5

Certainly! Merging the two code snippets to both calculate the lensed CMB spectrum using CLASS and then plot it using Matplotlib can be achieved as follows:

python Copy code

```python
import numpy as np
import matplotlib.pyplot as plt
from classy import Class

# Initialize Class
cosmo = Class()
params = {
    'output': 'tCl',
    'l_max_scalars': 5000,
    # Other cosmological parameters...
}
cosmo.set(params)
cosmo.compute()
```

1. Linear cosmology
   - General introduction
   - Einstein-Boltzmann solver: `CLASS`

2. Non-linear cosmology on a super-computer
   - Setting up initial condition: `Monofonic`
   - Non-linear dynamics: `RAMSES`
   - Diagnostic out the output: `YT, Pylians`

Standard model of cosmology

1. Gravity is described by General Relativity (GR)
2. Matter content is standard model of particle physics + dark matter + dark energy + radiation.
3. The cosmological principle
4. Initial conditions are nearly scale invariant, adiabatic, Gaussian...

Observational support

- Cosmic Microwave Background=CMB (temperature + polarization)
- Large Scale Structures=LSS (clustering + lensing)
- Supernovae
- Big-Bang Nucleosynthesis.....

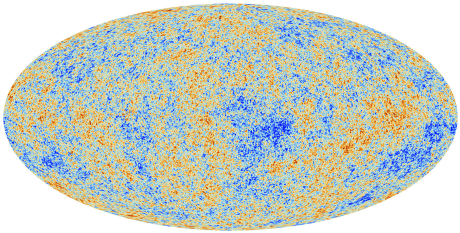


Image credit: Planck, ESA

Linear perturbations

Linear perturbation are the basic of modern cosmology. Direct observables:

- CMB: C_ℓ^{TT}, C_ℓ^{TE}, C_ℓ^{EE}, $C_\ell^{\phi\phi}$, C_ℓ^{BB}.
- LSS: $P_{\mathrm{lin}}(k,z)$, $\xi^{\pm}(z,\theta)$. 3×2-point cosmology.

Input for other codes: N-body, 21cm, Lyman-α forest.

Einstein-Boltzmann equations

$$G_{\mu\nu} = T_{\mu\nu} \tag{1}$$

$$\mathcal{L}[f] = \mathcal{C}[f] \tag{2}$$

Linear matter perturbations:

Vlasov-Poisson equation

$$\mathcal{L}[f] = 0 \tag{3}$$

$$\Delta\phi = 4\pi G\bar{\rho}a^2\delta \tag{4}$$

Zeroth moment: conservation of mass:

$$\dot{\delta}(\vec{x},t) + \frac{1}{a(t)}\vec{\nabla}\cdot\vec{v} = 0 \tag{5}$$

First moment: Euler equation:

$$\dot{\vec{v}}(\vec{x},t) + H\vec{v} + \frac{1}{a(t)}\vec{\nabla}\phi = 0 \tag{6}$$

$$\ddot{\delta}(\vec{x},t) + (\text{expansion})\,\dot{\delta}(\vec{x},t) - (\text{gravity})\,\delta(\vec{x},t) = 0 \tag{7}$$

$$\ddot{\delta}(\vec{x},t) + 2H\dot{\delta}(\vec{x},t) - \frac{3}{2}H^2\delta(\vec{x},t) = 0 \tag{8}$$

$\delta \equiv \frac{\rho-\bar{\rho}}{\bar{\rho}}$ = density contrast ; $a(t)$ =scale factor ; $H \equiv \dot{a}/a$ = Hubble parameter
$\vec{v}$: velocity of the dark matter fluid ; ϕ = gravitational potential.

Cosmological observables

Solve **linear** Vlasov-Boltzmann system within seconds, eg. (`CLASS`, `CAMB`).
Possible to sample parameter space → MCMC.

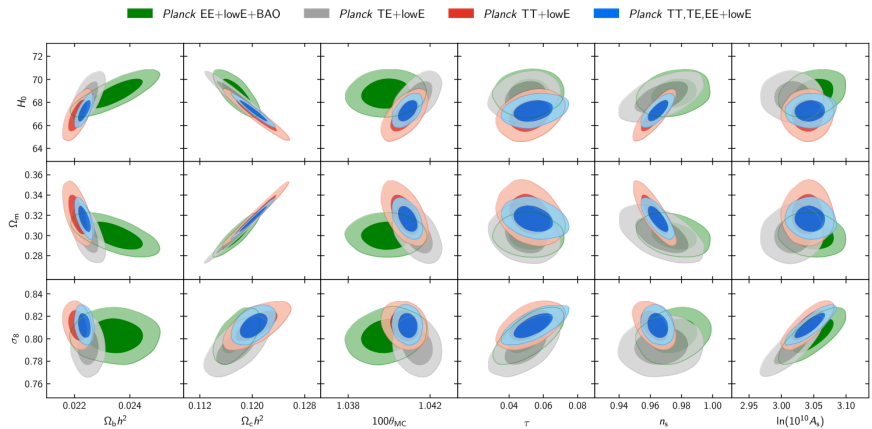


Image credit: Planck 18

Queen cosmological observable: Matter power spectrum

$$\langle \delta(\vec{k}_1, t)\delta(\vec{k}_2, t)\rangle = (2\pi)^3 \delta_D(\vec{k}_1 + \vec{k}_2) P_m(k_1, t) \quad (9)$$

Fourier transform of two-point correlation fct: $\xi(|\vec{x}_2 - \vec{x}_1|) = \langle \delta(\vec{x}_1, t)\delta(\vec{x}_2, t)\rangle$

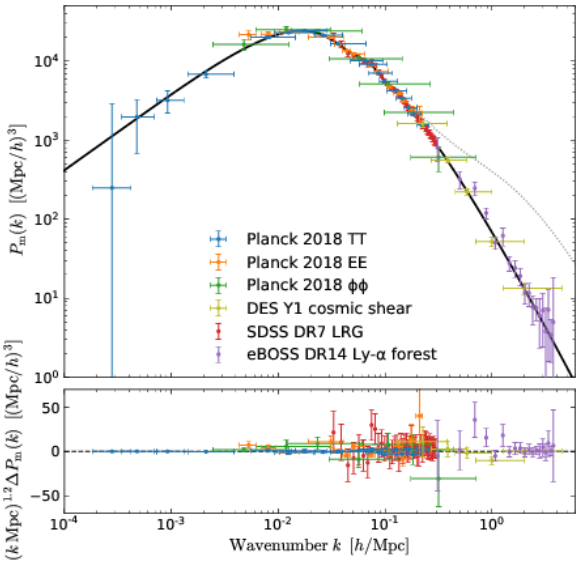
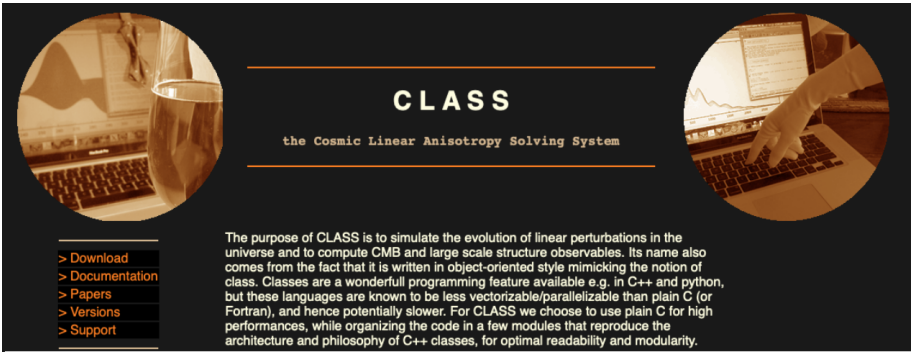
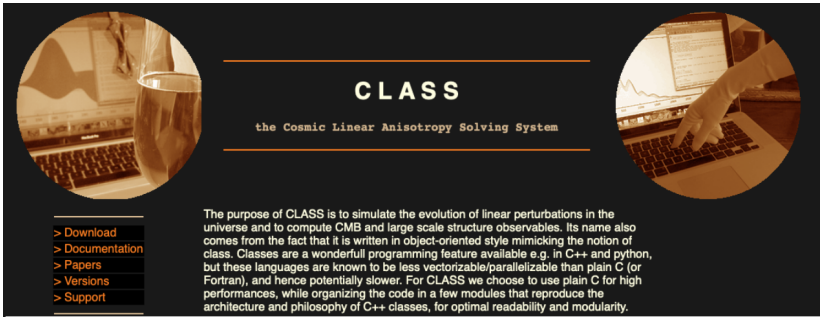


Image credit: Chabanier 19

1. Linear cosmology
   - General introduction
   - Einstein-Boltzmann solver: `CLASS`
2. Non-linear cosmology on a super-computer
   - Setting up initial condition: `Monofonic`
   - Non-linear dynamics: `RAMSES`
   - Diagnostic out the output: `YT, Pylians`



`CLASS` written in `C`.
Python wrapper=`classy` (a bit less stable, especially for Windows/MAC).



Outputs of CLASS:

- CMB anisotropy spectra: C_ℓ + decomposition in intrinsic, Sachs-Wolfe, Doppler, ISW.
- Matter (linear) power spectrum + decomposition in density, RSD, lensing + emulated (HaloFit) non-linear corrections
- Transfert functions (position, velocity, potentials), and their time evolution
- Matter density number count or lensing (C_ℓ)
- Background evolution of a cosmological model: characteristic redshifts, physical and comoving scales, angles.
- Thermal history of a cosmological model

Models implemented in CLASS

- Vanilla ΛCDM
- primordial perturbations: calculate inflation perturbation for a given $V(\phi)$, isocurvature
- neutrinos: chemical potentials, arbitrary phase-space distributions, flavor mixing, interacting neutrinos
- Dark radiation: fluid/self-interacting dark radiation, viscous dark radiation
- Dark matter: warm, anhilating, decaying, interacting, warm+interacting, generalized dark matter.
- Dark energy: fluid with $w(a)$ + c_s, quitessence $V(\phi)$, Early Dark Energy (AxiCLASS and TriggerCLASS to be merged)
- modified gravity: hi_class
- Relativistic effects: (More on this on thursday)
  - different gauge choices (first order): synchronous, Newtonian, N-body
  - Second order: SONG
- CLASS SZ, MultiClass

oodles amount of information online (+community)

- file `explanatory.ini`
- http://class-code.net
- Online course: (slides and video)
- online html documentation

The CLASS Tour: CCA, Simons Foundation, New York City, 15-16 July 2019

CLASS + SONG workshop

by Christian Fidler and Julien Lesgourgues

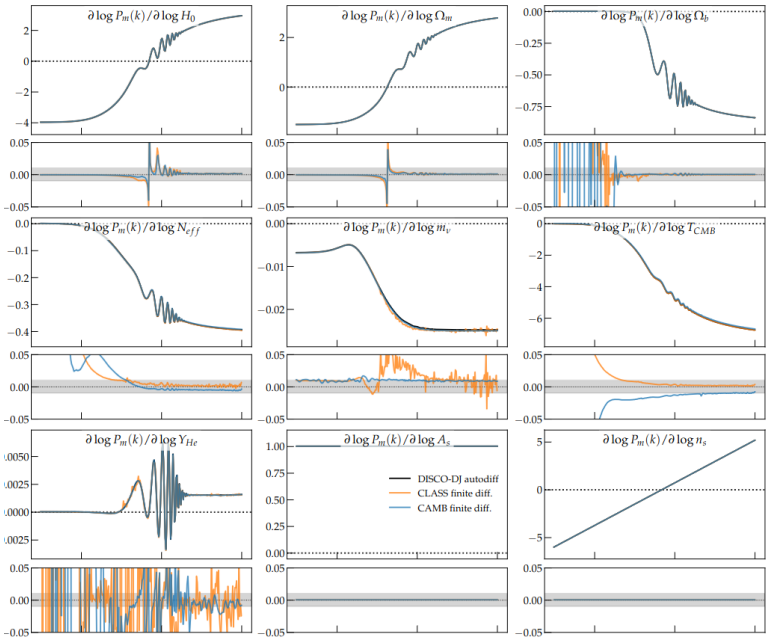
Overall program and schedule.

Lectures and exercises on CLASS:

- CLASS Basics: Coding spirit and general rules [slides], [video]
- CLASS Theory: Less trivial aspect of the implemented linear perturbation theory [slides], [video]
- CLASS Usage I: From interactive runs to python notebooks [slides], [video]
- CLASS Usage II: Exploring the code's possibilities through python notebooks [slides], [video]
- CLASS Coding I: Essential rules and conventions specific to the code [slides], [video]
- CLASS Coding II: How to implement new physics and new ingredients [slides], [video]
- CLASS Exercises: [slides], [video]

CLASS and machine learning: DISCO-DJ

DIfferentiable Simulations for COsmology - Done with JAX, Hahn 23



Usage of CLASS

Command line

`./class explanatory.ini`

`explanatory.ini` very well documented with all input possibilities.

Examples

`parameter_name = value:`	`Omega_cdm = 0.25`
`option = yes / no:`	`lensing = yes`
`selection = list, of, things:`	`output = tCl, pCl, lCl`
Comments:	`#` This variable is as stubborn as a mule: refuses to change its value!

- Outputs files written to `output/<filename>_xxx.dat`
- Command line file outputs: `write_background`, `write_thermodynamics`

Python wrapper

- `import classy`
- `cosmo = classy.Class()`
- `cosmo.set(input_dictionary)`
- `cosmo.compute()`
- `cosmo.lensed_cl()`
- `cosmo.get_pk_all(k,z)`
- `cosmo.get_background()`
- `cosmo.Hubble(z)`

eg. Matter budget

`Omega_m=0.3`	Ω_m
`omega_m=0.15`	$\Omega_m h^2$
`Omega_cdm=0.25`	$\Omega_{\rm cdm}$
`omega_cdm=0.125`	$\Omega_{\rm cdm} h^2$

Usage of CLASS

Output for CLASSY (python)

- `lensed_cl(lmax=-1)`: $C_\ell^{\rm lensed}$; `raw_cl(lmax=-1)`: $C_\ell^{\rm unlensed}$
- `get_pk_all(k,z, nonlinear=True, cdmbar=False)`: $P(k,z)$
- `get_pk_and_k_and_z(nonlinear=True, only_clustering_species=False, h_units=False)`
- `get_transfer_and_k_and_z(output_format='class', h_units=False)`: $\delta(k,z)$, $\theta(k,z)$
- `get_background()`, `get_thermodynamics()`, `get_primordial()`, `get_perturbations()`, `get_transfer(z)`
- `h()`, `n_s()`, `z_reio()` `Omega_m()` `Omega_r()`, `Omega_Lambda()`, `Omega_b()`, `keq()`, `age()`, `rs_drag()`
- `luminosity_distance(z)`, `angular_distance(z)`, `angular_distance_from_to(z1,z2)`, `comoving_distance(z)`
- `scale_independent_growth_factor(z)`: $D(z)$
- `sigma(R,z,hunits=False)`: σ_8 =`sigma(8,0,hunits=True)`
- `Hubble(z)`: $H(z)$

CLASS and machine learning:

A precise symbolic emulator of the linear matter power spectrum

Deaglan J. Bartlett ⋆[1], Lukas Kammerer[2], Gabriel Kronberger[2], Harry Desmond[3], Pedro G. Ferreira[4], Benjamin D. Wandelt[1,5], Bogdan Burlacu[2], David Alonso[4], and Matteo Zennaro[4]

ABSTRACT

Context. Computing the matter power spectrum, $P(k)$, as a function of cosmological parameters can be prohibitively slow in cosmological analyses, hence emulating this calculation is desirable. Previous analytic approximations are insufficiently accurate for modern applications, so black-box, uninterpretable emulators are often used.
Aims. To construct an efficient, differentiable, interpretable, symbolic emulator for the redshift zero linear matter power spectrum which achieves sub-percent level accuracy. We also wish to obtain a simple analytic expression to convert A_s to σ_8 given the other cosmological parameters.
Methods. We utilise an efficient genetic programming based symbolic regression framework to explore the space of potential mathematical expressions which can approximate the power spectrum and σ_8. We learn the ratio between an existing low-accuracy fitting function for $P(k)$ and that obtained by solving the Boltzmann equations and thus still incorporate the physics which motivated this earlier approximation.
Results. We obtain an analytic approximation to the linear power spectrum with a root mean squared fractional error of 0.2% between $k = 9 \times 10^{-3} - 9\,h\,\mathrm{Mpc}^{-1}$ and across a wide range of cosmological parameters, and we provide physical interpretations for various terms in the expression. We also provide a simple analytic approximation for σ_8 with a similar accuracy, with a root mean squared fractional error of just 0.4% when evaluated across the same range of cosmologies. This function is easily invertible to obtain A_s as a function of σ_8 and the other cosmological parameters, if preferred.
Conclusions. It is possible to obtain symbolic approximations to a seemingly complex function at a precision required for current and future cosmological analyses without resorting to deep-learning techniques, thus avoiding their black-box nature and large number of parameters. Our emulator will be usable long after the codes on which numerical approximations are built become outdated.

$$(a_0 A_s + a_1 n_s)(a_2 \Omega_b + \log(a_3 \Omega_M)) \log(a_4 h) + a_5 \tag{10}$$

$$\text{with } \vec{a} = [1.61 \times 10^8, 0.343, -7.86, 18.2, 3.67, 0.00336] \tag{11}$$

1. Linear cosmology
   - General introduction
   - Einstein-Boltzmann solver: `CLASS`
2. Non-linear cosmology on a super-computer
   - Setting up initial condition: `Monofonic`
   - Non-linear dynamics: `RAMSES`
   - Diagnostic out the output: `YT, Pylians`

Why a (super-)computer?

- Linear perturbation theory: a (very good) tool when

$$\frac{\rho - \bar{\rho}}{\bar{\rho}} \equiv \delta \ll 1 \tag{12}$$

- Quasi linear scales: $\delta \sim 1$: analytical approaches [NOT the topic today]
  - regularized perturbation theory (RegPT, Bernardeau 15)
  - renormalization group flow (Pietroni 08)
  - kinetic field theory (Bartelmann 19)
  - effective field theories (Baumann 10)
    - `CLASS-PT`
    - `PyBird` (Pierre Zhang)

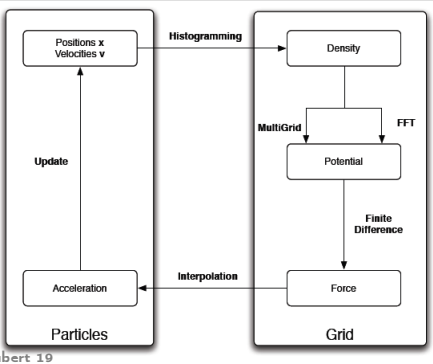
  $\rightarrow k \sim 0.3h/\mathrm{Mpc}$
- Strong non-linear regime: $\delta \gg 1$, no analytical approximations for gravitational evolution of matter density perturbations $\rightarrow$ **numerical approach** [topic of today :)]

Numerical approach

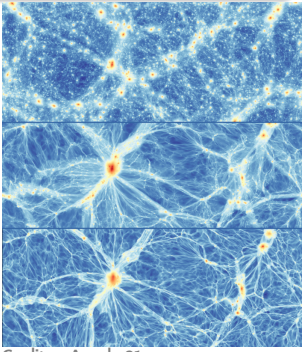
- Large number of elements (stars, dark matter particles) $\rightarrow$ statistical approach: **distribution function in phase space**: f_{m}
- Dark matter assumed to be composed of $\mathcal{O}(10^6 - 10^{12})$ particles gravitationally interacting.
- Vlasov-Poisson system : ($\phi \sim 10^{-5}$, $v \sim 10^{-2}$)

$$\frac{\partial f_{\mathrm{m}}}{\partial t} + \frac{v_i}{a^2}\frac{\partial f_{\mathrm{m}}}{\partial x^i} - \frac{\partial \phi}{\partial x^i}\frac{\partial f_{\mathrm{m}}}{\partial v_i} = 0 \tag{13}$$

$$\Delta\phi = 4\pi G\bar{\rho}a^2\delta \tag{14}$$



Credits : Aubert 19



Credits : Angulo 21

Review: Angulo & Hahn 21

Different gravity solvers

- Particle-Particle (Aarseth03)
- Hierarchical Tree (Barnes & Hut 86).
  Exemples of (public) tree codes: PKDGRAV3 (Potter 16) GADGET4 (Springel 20), ChaNGa (Quinn 15).
- Particle-Mesh (Hockney & Eastwood 81).
  Exemples of (public) mesh codes: FASTPM (Feng 16), ENZO (Bryan 97), **RAMSES** (Teyssier 02), AMIGA (Knebe 01)

Time evolution

- Once you have calculated the gravitational force for every particle, needs to move the particle according to its dynamics (Newton's law)
- Second order ODE: Runge-Kutta4, or LeapFrog
- Symplectic structure: Kick-Drift-Kick or Drift-Kick-Drift.

Supercalculators

- HPC infrastructure = paralellism. MPI library to communicate among different nodes. OpenMP to speed up directives in nodes computations.
- Computational domain (particles or particles+grid) decomposed into smaller pieces distributed over the nodes.
- Hilbert curve decomposition (note: for very cluttered situation `MULTIPLEDOMAINS` helps in GADGET4.)

Jean Zay @ IDRIS

Joliot Curie @ TGCC

Jean Zay (IDRIS, Saclay)

- CSL: 86 344 CPU cores
- V100: 2448 V100 GPUs
- A100: 416 A100 GPUs

Joliot Curie (TGCC, Bruyères-le-Châtel)

- Irene SKL: 79 488 CPU cores
- Irene Rome: 293 376 CPU cores
- Irene V100: 128 V100 GPUs

State of the art simulations

Credits : Angulo 21

Year	Simulation	Code, Algorithm	Supercomputer, Location	Cores $[10^3]$	N_p $[10^{12}]$	Box $[h^{-1}\mathrm{Gpc}]$	ϵ $[h^{-1}\mathrm{kpc}]$
2014	Dark Sky (Skillman et al. 2014)	2HOT FMM	Titan USA	20	1.1	8	36.8
2017	TianNu (Emberson et al. 2017)	CUBEP^{3}M PM-PM-PP	Tianhe-2 China	331	2.97	1.2	13
2017	Euclid Flagship (Potter et al. 2017)	PKDGRAV3 Tree-FMM	PizDaint Switzerland	4	2.0	3.	4.8
2019	Outer Rim (Heitmann et al. 2019)	HACC Tree-PM	Mira USA	524	1.07	3.0	2.84
2019	Cosmo-π (Cheng et al. 2020)	CUBE PM-PM	π 2.0 China	20	4.39	3.2	195
2020	Uchuu (Ishiyama et al. 2021)	GREEM Tree-PM	ATERUI-II Japan	<40	2.0	2.0	4.3
2020	Last Journey (Heitmann et al. 2021)	HACC Tree-PM	Mira USA	524	1.24	3.4	3.14
2021	Far Point (Frontiere et al. 2021)	HACC Tree-PM	Summit USA	?	1.86	1	0.8

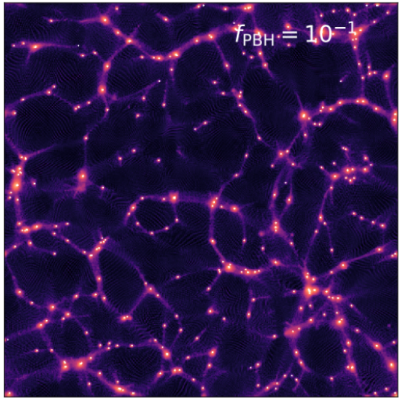
Table 1 List of cosmological simulations with a particle number in excess of 1 trillion (10^{12})

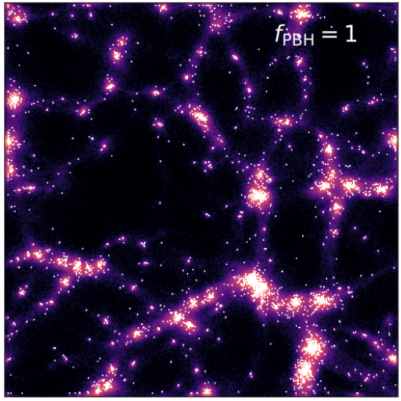
Cosmic Dawn III: 131 072 CPU + 24 576 GPUs (Lewis 22)
Fugaku 4 Euclid: 720 M CPU-hours x 3 years (2023-2025), Top 4 HPC, 30 MW!
Gordon Bell Prize

'Modified' gravity

Some selected codes (often written on the top of RAMSES or GADGET)

- Axion/Fuzzy Dark matter: `SCALAR` (Mina 19), May 21
- $f(R)$ (Zhao 10), `ECOSMOG` (Li 11, Brax 12), Modified Gravity-GADGET (Puchwein 13), (Arnold 19), `MG-GLAM` (Ruan 21 & Hernández-Aguayo 21)
- `ISIS` (Llinares 13), Scalar dark matter: (Hopkins 18)
- MOND: Phantom of RAMSES, Dipolar dark matter, `Bi-Poisson`
- ΛPBH (Inmar 19)





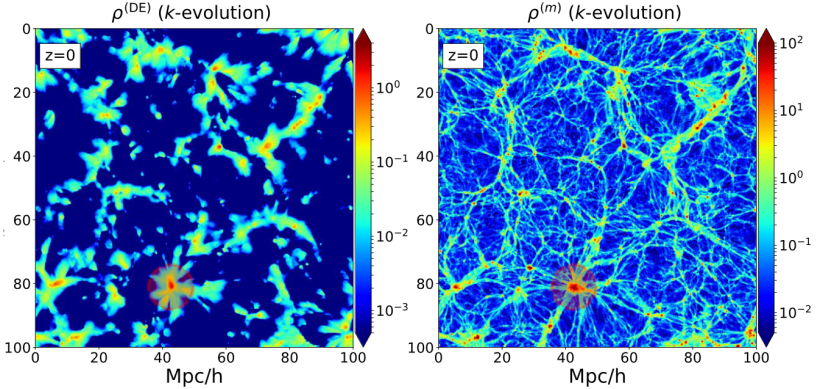
General Relativity (more in my seminar of thursday)

Instead of Poisson, solve

$$-3\mathcal{H}(\phi' + \mathcal{H}\psi) + \Delta\phi = 4\pi G\bar{\rho}a^2\delta \tag{15}$$

+ 5 other degrees of freedom of GR

- RayGal (Breton 21)
- GRAMSES (Barrera-Hinojosa 19)
- `RELIC` (Adamek 20), `gevolution` (Adamek 16)
- `k-evolution` (Hassani 19), Asevolution (Christiansen 23)



1. Linear cosmology
   - General introduction
   - Einstein-Boltzmann solver: `CLASS`
2. Non-linear cosmology on a super-computer
   - Setting up initial condition: `Monofonic`
   - Non-linear dynamics: `RAMSES`
   - Diagnostic out the output: `YT, Pylians`

General description

Goal

Convert the (linear) power spectrum into a list of particles' positions + velocities

Draw a realisation of a 3D Gaussian Random field

$$\delta(\vec{x}) \propto \int \frac{d^3\vec{k}}{(2\pi)^3} \sqrt{P_L(k)} \mathcal{N}_k(0,1) e^{i\vec{k}\cdot\vec{x}}, \tag{16}$$

where $\mathcal{N}(0,1)$ is drawn from a Gaussian random field and P_L is the linear power spectrum (`CLASS`, `CAMB`)
$\rightarrow$ velocities obtained with linear perturbation theory

Example of initial condition generators

- mpgrafic
- 2LPTIC
- FASTPM
- N-GENICS (with GADGET)
- **MUSIC** and **Monofonic**
- ginnungagap
- genetIC
- Panphasia

```
################################################################################################
# Example config file for MUSIC2 - monofonIC single resolution simulation ICs
#                                                                    version 1 from 2020/08/23
################################################################################################

################################################################################################
[setup]

GridRes         = 128      # number of grid cells per linear dimension for calculations
                           #   = particles for sc initial load
BoxLength       = 300      # length of the box in Mpc/h
zstart          = 24.0     # starting redshift
```

Comments: on a laptop, `GridRes` ~ 128
good to start at early enough time for linear theory to be valid

```
[cosmology]
## transfer = ... specifies the Einstein-Boltzmann plugin module

ParameterSet    = Planck2018EE+BAO+SN  # specify a pre-defined parameter set,
                                       # or set to 'none' and set manually below

## cosmological parameters, to set, choose ParameterSet = none,
## default values (those not specified) are set to the values
## from 'Planck2018EE+BAO+SN', we currently assume flatness
# Omega_m         = 0.3158
# Omega_b         = 0.0494
# Omega_L         = 0.6842
# H0              = 67.321
# n_s             = 0.9661
# sigma_8         = 0.8102
# A_s             = 2.148752e-09  # can use A_s instead of sigma_8 when using CLASS
# Tcmb            = 2.7255
# k_p             = 0.05
# N_ur            = 2.046
# m_nu1           = 0.06
# m_nu2           = 0.0
# m_nu3           = 0.0
```

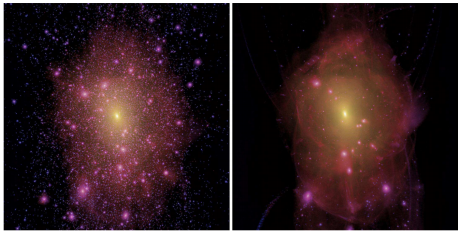
Physics beyond ΛCDM in Monofonic

Dark Matter

- Warm Dark Matter: Vanilla (public) monofonic
- ETHOS-like parametrization (Cyr-Racine 15, Murgia 17):

$$P_X/P_{\rm CDM} = \left(1 + (\alpha k)^\beta\right)^{-2\gamma} \tag{17}$$

(WDM: $\beta = \gamma = 2.23$)
describe decay of the power spectrum of fuzzy dark matter, sterile ν...
Implemented in a separate branch of monofonic (available upon request)



Credits : Angulo 21

Dark Energy

Scalar dark energy and w_0, w_a in branch of monofonic (available upon request)

Physics beyond ΛCDM in Monofonic

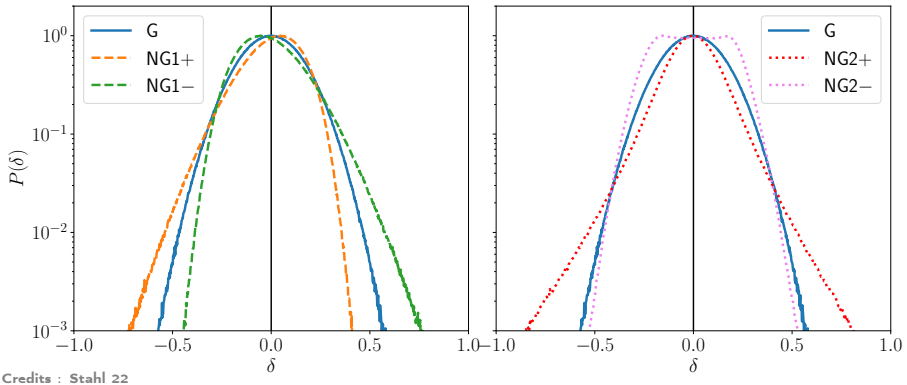
Primordial non-Gaussianities (PNG)

$$\Phi = \Phi_G + f_{\rm NL}\Phi_G^2 + g_{\rm NL}\Phi_G^3 \tag{18}$$

Scale dependant PNG:

$$f_{\rm NL}(k) = f_{\rm NL}^0 \left(\frac{k}{k_0}\right)^{n_{f_{\rm NL}}} \tag{19}$$

(available upon request)

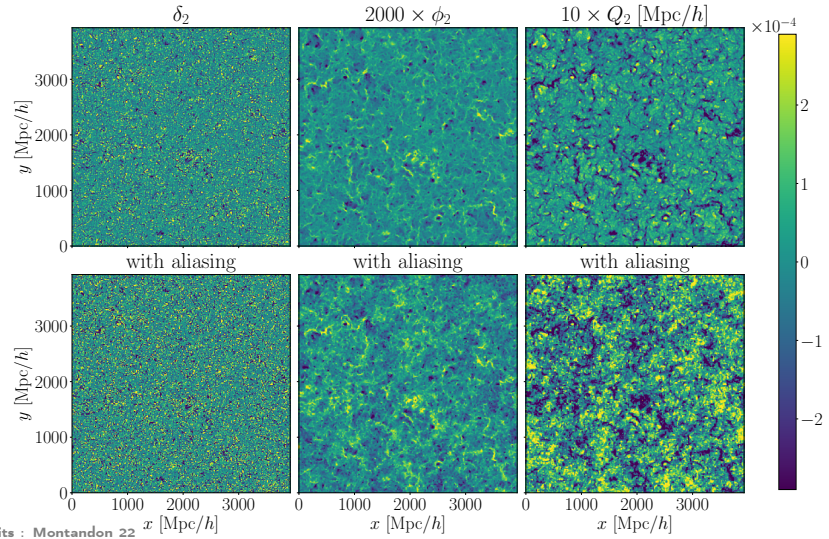


Credits : Stahl 22

Nice features of Monofonic: De-aliasing (*)

Aliasing:

numerical error due to sampling of the fields
→ most important when multiplying two fields (PNG, nLPT...)

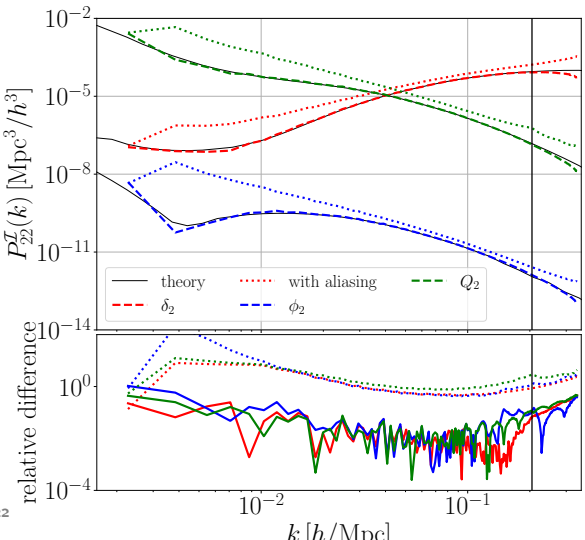


Credits : Montandon 22

Nice features of Monofonic: De-aliasing (*)

Orszag's 3/2 rule

implemented in monofonic (Orszag 71)



Credits : Montandon 22

Nice features of Monofonic (*)

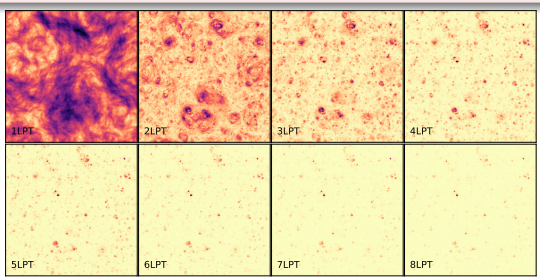
Transiants

Linear perturbation theory has two solutions: the growing modes ($\propto a(t)$ in matter domination) and the decaying mode ($\propto a^{-3/2}$).
Numerical error to neglect decaying mode at early time $\rightarrow$ transiant Crocce 06.

Push perturbation theory to higher order, to start simulation later, so that decaying modes had time to decay.

- 2LPTIC code
- 3LPT (start simulation at $z = 7$) in vanilla Monofonic and recursion relations nLPT implemented in a branch of Monofonic (Rampf 20)

```
LPTorder        = 3         # order of the LPT to be used (1,2 or 3)
```



Nice features of Monofonic (*)

```
DoFixing        = no      # do mode fixing à la Angulo&Pontzen (https://arxiv.org/abs/1603.05253)
DoInversion     = no       # invert phases (for paired simulations)
```

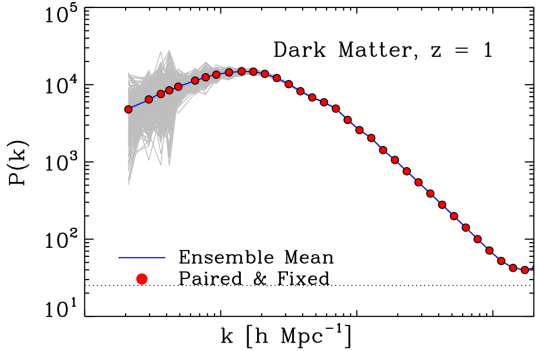
Fixing: draw the amplitude from a Dirac-delta

the phase θ_k is drawn from $[0, 2\pi]$

$$\delta(\vec{x}) \propto \int \frac{d^3\vec{k}}{(2\pi)^3} \sqrt{P_L(k)} e^{i\theta_k} e^{i\vec{k}\cdot\vec{x}}, \tag{20}$$

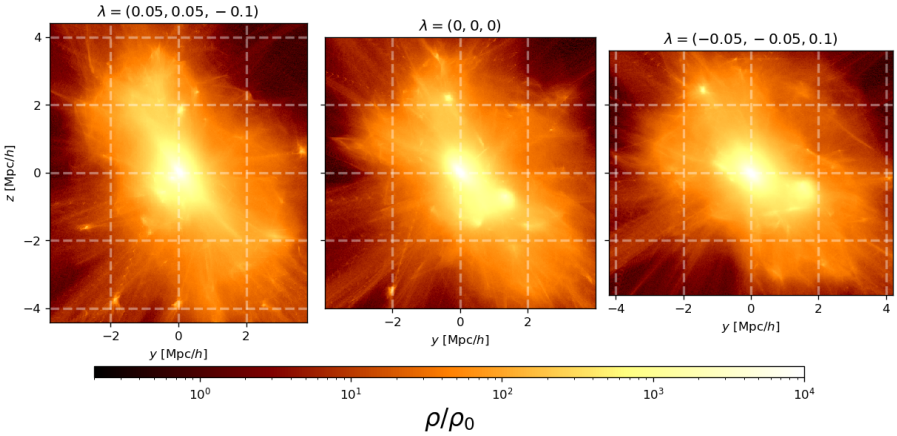
no more Gaussian random field realisation!

Paired = take the same simulation with $\theta_k + \pi$



Anisotropic simulations (*)

```
## Use below for anisotropic large scale tidal field ICs up to 2LPT
## see Stuecker+2020 (https://arxiv.org/abs/2003.06427)
# LSS_aniso_lx     = +0.1
# LSS_aniso_ly     = +0.1
# LSS_aniso_lz     = -0.2
```



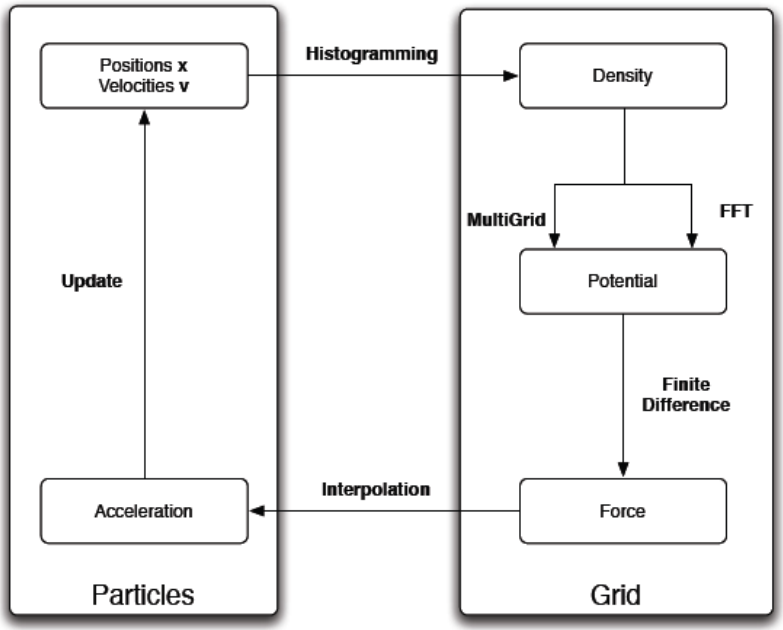
Possible outputs

```
[output]
## format = .... specifies the output plugin module

##> RAMSES / GRAFIC2 compatible format
# format                = grafic2
# filename        = ics_ramses
# grafic_use_SPT  = no # if no then uses PPT, otherwise linear SPT

##> Gadget-2/3 'fortran unformatted binary'-style format
# format          = gadget2
# filename        = ics_gadget.dat
# UseLongids      = false

##> Gadget-2/3 HDF5 format
# format          = gadget_hdf5
# filename        = ics_gadget.hdf5

##> Arepo HDF5 format (virtually identical to gadget_hdf5)
# format          = AREPO
# filename        = ics_arepo.hdf5

##> HACC compatible generic-io format
# format          = genericio
# filename        = ics_hacc

##> SWIFT compatible HDF5 format. Format broadly similar to gadget_hdf5 but in a single
##> file even when using MPI. No h-factors for position and masses and no sqrt(a)-factor for the velocities.
##> IDs are stored using 64-bits unless UseLongids is set to false.
# format          = SWIFT
# filename        = ics_swift.hdf5
# UseLongids      = true

##> Generic HDF5 output format for testing or PT-based calculations
# format          = generic
# filename        = debug.hdf5
# generic_out_eulerian = yes  # if yes then uses PPT for output
```

don't forget to uncomment the desired output (RAMSES, for this tutorial)

1. Linear cosmology
   - General introduction
   - Einstein-Boltzmann solver: `CLASS`
2. Non-linear cosmology on a super-computer
   - Setting up initial condition: `Monofonic`
   - Non-linear dynamics: `RAMSES`
   - Diagnostic out the output: `YT, Pylians`

RAMSES in one slide



Credits : Aubert 19

Example of input file for RAMSES

```
&RUN_PARAMS
cosmo=.true.
pic=.true.
poisson=.true.
hydro=.false.
nrestart=0
nremap=1
nsubcycle=1,2
ncontrol=1
verbose=.false.
/

&AMR_PARAMS
levelmin=7
levelmax=12
nexpand=1
ngridmax=2000000
npartmax=3000000
/
```

```
&INIT_PARAMS
filetype='grafic'
initfile(1)='../music/ics_ramses/level_007/'
/

&OUTPUT_PARAMS
foutput=1000
noutput=1
/

&REFINE_PARAMS
m_refine=6*8.,
/
```

`levelmin` and `levelmax` control the minimal and maximal level of refinement of the RAMSES grid.

Hydrodynamics in RAMSES (*)

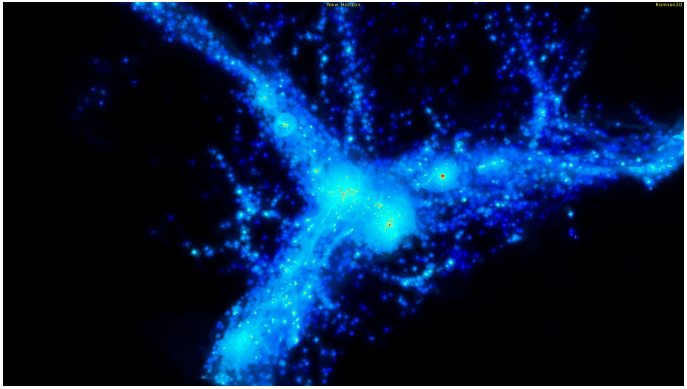
```
&RUN_PARAMS
cosmo=.true.
pic=.true.
poisson=.true.
ordering='hilbert'
nremap=20
nsubcycle=1,30*2
nrestart=55
hydro=.true.
ncontrol=1
sink=.true.
sinkprops=.true.
/

&HYDRO_PARAMS
gamma=1.6666667
courant_factor=0.8
slope_type=1
pressure_fix=.true.
scheme='muscl'
riemann='hllc'
/
```

NewHorizon
El Gordo cluster

```
&PHYSICS_PARAMS
haardt_madau=.true.
cooling=.true.
z_reion=10.0
z_ave=1d-3
n_star=0.1d0
eps_star=0.02
del_star=50d0
T2_star=1d4
g_star=1.3333334
f_ek=1d0
Mseed=1d5
sink_AGN=.true.
bondi=.true.
drag=.true.
X_floor=1d-2
eAGN_K=1d0
eAGN_T=0.15d0
TAGN=0.0
r_gal=50.0d0
sigmav_max=1d15
T2maxAGN=1d10
boost_acc=2d0
boost_drag=2d0
metal=.true.
eta_sn=0.3d0
yield=0.05d0
f_w=10d0
t_delay=10d0
/
```

Things to bear in mind when running a simulation

- Distances larger than ~ 0.33 box size L affected by periodic boundary condition
- Cosmic variance effects due to box size L (pairing...)
- Check gravitational softening length ϵ (typically $\frac{1}{30} L/N^{1/3}$)
- Distances smaller than 2-3 cells size or $3 \times \epsilon$ affected by numerical error (unresolved force)
- Too large time step can introduce error in particle trajectories



Credits : NewHorizon

Machine learning and cosmological simulations

CAMELS: Cosmology and Astrophysics with MachinE Learning Simulations.
700 Tb of data, 53 papers since october 2020! Astro and Cosmo together!

Applications include:

- Non-linear interpolation
- Parameter estimation
- **Symbolic regression**
- Generative models
- Anomaly detection

Type	Code	Subgrid model	Simulations
Hydrodynamic	Arepo	IllustrisTNG	2,143
	Gizmo	SIMBA	1,092
	MP-Gadget	Astrid	2,116
	OpenGadget	Magneticum	77
	SWIFT	EAGLE	77
	Ramses		5
	Enzo		6
N-body	Gadget-III	–	5,164

$$\log_{10}(\mathrm{SFRD}) = -\frac{1.777 + \sigma_8 A_{\mathrm{SN2}}}{1+z} - (1+z)\left(\frac{0.365}{\sigma_8} - 0.559\Omega_{\mathrm{m}} + 3.57\times10^{-3}\frac{A_{\mathrm{SN1}}}{\Omega_{\mathrm{m}}\sigma_8}\right) + 0.39^{A_{\mathrm{SN1}}}$$

$$\log_{10}(\mathrm{SFRD}) = \frac{0.696}{(1+z)A_{\mathrm{SN2}}} - (1+z)\left(\frac{0.0389}{\Omega_{\mathrm{m}}} + \frac{0.379}{\sigma_8}\right) - 1.333 + 2.317\log(1+z) - A_{\mathrm{SN1}}^{0.391}$$

$$\log_{10}(\mathrm{SFRD}) = \frac{0.692A_{\mathrm{SN1}}^{0.458}}{(1+z)A_{\mathrm{SN2}}} - (1+z)\left(\frac{0.038}{\Omega_{\mathrm{m}}} + \frac{0.425}{\sigma_8}\right) + \frac{0.36}{\sigma_8} - 1.631 + 2.534\log(1+z) - 1.21A_{\mathrm{SN1}}^{0.406}$$

For this tutorial:

- Visualization with YT[a]
- Power spectrum with Pylians[b]
- Halo Mass Function with YT theoretical line with colossus[c]

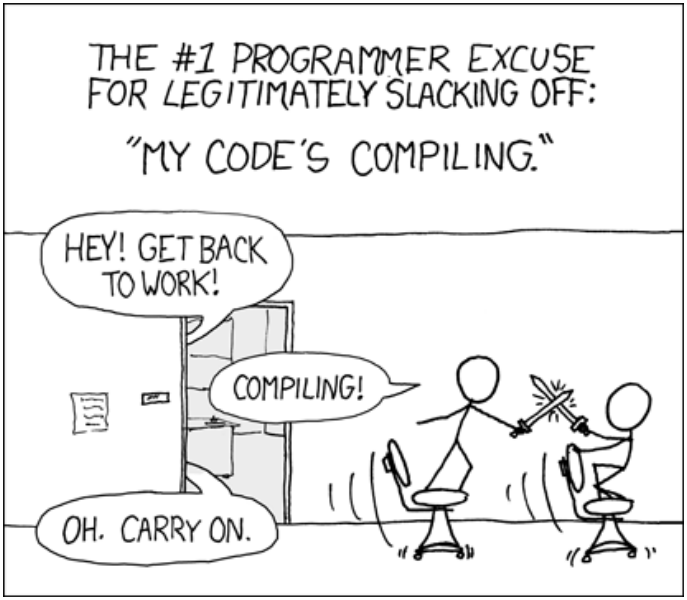
They are very well documented and with a large community helping and developing!

[a] can also give you access to data including underlying gravity fields (rotation curves, modified gravity...), 3D visualization (volume rendering), derived quantities...

[b] can also give you density field, real space correlation function, bispectrum, voids, redshift space distorsion...

[c] can also give you halo mass function and halo bias, density profiles, splashback radius, concentration

Other tools include POWMES, Pynbody, Nbodykit, bskit, Hipster, Corrfunc, HALOTOOLS, ytree, Glnemo, uns_io...
Database of tools for cosmology on the Action Dark Energy WG webpage.


THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.