CAMELS

Release 0.1

CAMELS team

Apr 15, 2024

CAMELS

CAMELS stands for **C**osmology and **A**strophysics with **M**achin**E** **L**earning **S**imulations, and it is a project that aims at building bridges between cosmology and astrophysics through numerical simulations and machine learning. CAMELS contains 12,903 cosmological simulations –5,164 N-body and 7,712 state-of-the-art (magneto-)hydrodynamic– and more than 1 Petabyte of data. CAMELS is the largest set of cosmological hydrodynamic simulations ever run.

Type	Code	Subgrid model	Simulations
Hydrodynamic	Arepo	IllustrisTNG	3,167
	Gizmo	SIMBA	1,119
	MP-Gadget	Astrid	2,116
	OpenGadget	Magneticum	77
	Swift	EAGLE	1,088
	Ramses		166
	Enzo		6
N-body	Gadget-III	—	7,739

CHAPTER

ONE

NEWS

March 2024 A reorganization of the data has been performed in order to enhance its uniformity and simplicity. This will require slight changes to existing codes that access the data.

- Folders in the 1P sets are now named `1P_pX_Y` and each parameter only has 4 variations, rather than 10, such that `X` ranges from n2 to 2.
- Snapshot numbers in the IllustrisTNG and SIMBA suites, where simulations have only 34 snapshots, have been updated to match the numbering in the Astrid suite (and some TNG simulations) that have 91 snapshots. For example, where 33 used to be the z=0 snapshot, now it is 90 uniformly for all suites.
- Snapshot files have been renamed from `snap_###.hdf5` to `snapshot_###.hdf5` and fof/subfind files from `fof_subhalo_tab_###.hdf5` to `groups_###.hdf5`.

March 2024 A new simulation set, SB28, has been added to the CAMELS-TNG suite. This set comprises of 2,048 new simulations that sample a 28-dimensional parameter space of the IllustrisTNG galaxy formation model, including 5 cosmological parameters and 23 sub-grid physics parameters, which facilitates more comprehensive and robust studies of our uncertainties about cosmological modeling of galaxies and baryons. These simulations have the same volume and resolution as other CAMELS simulations. 91 snapshots were generated for each simulation, and group catalogs and merger trees are available as well. Associated new 1P simulations that sample the additional 22 parameters are also released, in this case both for the IllustrisTNG and SIMBA suites.

March 2024 A new suite, "GZ28," of 768 massive hydrodynamical zoom-in simulations spanning the IllustrisTNG parameter space has been added. GZ28 employs a novel parameter space sampling method, "CARPoolGP," that leverages correlations between the initial conditions of simulations. See *CAMELS-zoomGZ* for more details.

May 2023 The snapshots of the N-body simulations have been compressed. In order to read those snapshots with python, you need to use `import hdf5` and `import hdf5plugin`. See *Simulations* for details.

April 2023 All simulations in the Astrid suite are now publicly available. The CAMELS Multifield Dataset has also been updated and now incorporates 2D maps and 3D grids from the Astrid simulations. Halo and galaxy catalogues from both Subfind and Rockstar are also publicly available together with merger trees from SubLink and Consistent trees.

March 2023: A new set has been added to the IllustrisTNG suite: the BE set, for "Butterfly Effect". These 27 simulations all have the exact same initial conditions and are run with the fiducial TNG model, while capturing the intrinsic randomness of the simulation results.

March 2023: SubLink merger trees have been added for the IllustrisTNG suite.

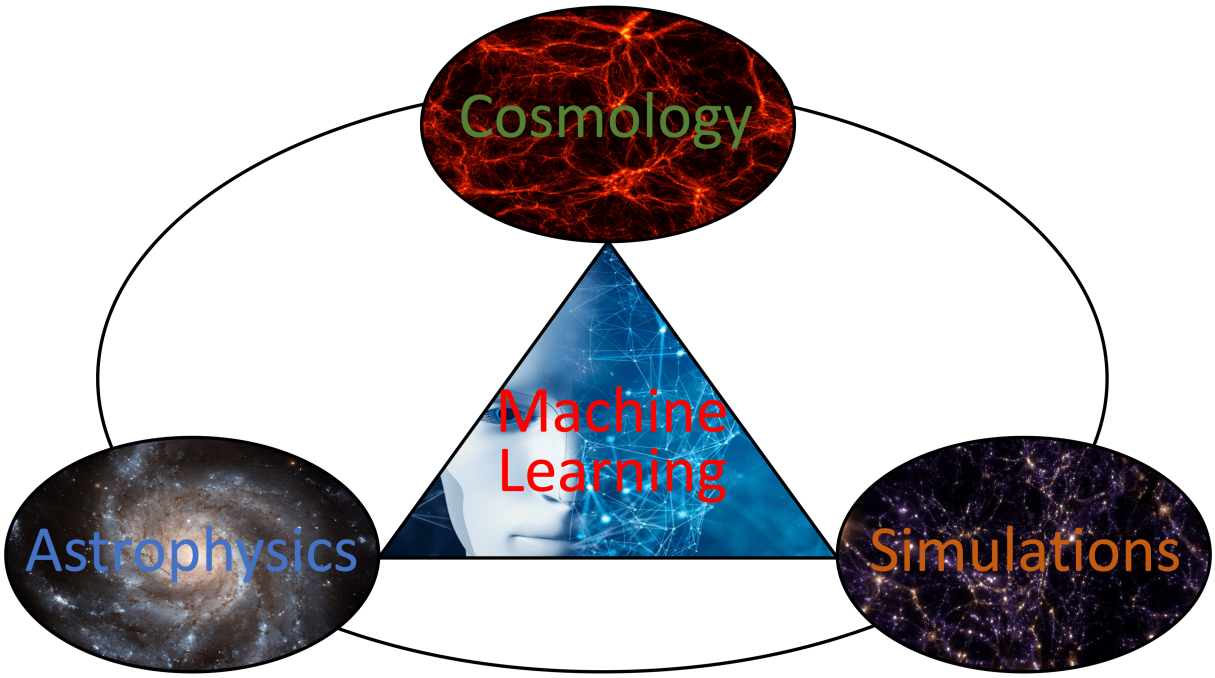
August 2022: The CAMELS binder now has Pylians3 and the CAMELS library preinstalled.

July 2022: CAMELS has grown! The Astrid suite – containing 1,092 state-of-the-art hydrodynamic simulations run with the MP-Gadget code employing the same subgrid model as the Astrid simulation – has been added to CAMELS. The N-body counterparts of these simulations are also available.

CHAPTER

TWO

SCIENTIFIC GOALS



In order to maximize the scientific return of cosmological missions such as CMB-S4, DESI, eROSITA, Euclid, Roman Observatory, Rubin Observatory, Simons Observatory, PFS, and SKA, accurate theoretical predictions in the non-linear regime, for generic statistics, are needed. Furthermore, these predictions need to incorporate the uncertainity associated to our poor knowledge of baryonic effects such as AGN feedback.

CAMELS is a large suite of N-body and state-of-the-art (magneto-)hydrodynamic simulations designed to tackle this problem. At its core, CAMELS represents a large dataset to train machine learning algorithms. The main scientific goals of CAMELS are these:

- Provide theory predictions for summary statistics and full 3D fields as a function of cosmology and astrophysics.
- Train neural networks to extract cosmological information while marginalizing over baryonic effects.
- Develop machine learning techniques to find the mapping between N-body simulations and hydrodynamic simulations with full baryonic physics.
- Quantify the dependence of galaxy formation and evolution on astrophysical and cosmological parameters.
- Use machine learning to efficiently calibrate subgrid parameters in cosmological hydrodynamic simulations to match a set of observations.

CHAPTER

THREE

PUBLICATIONS

1. **Parameter estimation from Ly forest in Fourier space using Information Maximising Neural Network**
   Soumak Maitra, Stefano Cristiani, Matteo Viel, Roberto Trotta, Guido Cupani
   2404.04327
2. **Debiasing with Diffusion: Probabilistic reconstruction of Dark Matter fields from galaxies with CAMELS**
   Victoria Ono, Core Francisco Park, Nayantara Mudur, Yueying Ni, Carolina Cuesta-Lazaro, Francisco Villaescusa-Navarro
   2403.10648
3. **Zooming by in the CARPoolGP lane: new CAMELS-TNG simulations of zoomed-in massive halos**
   Max E. Lee, Shy Genel, Benjamin D. Wandelt, Benjamin Zhang, Ana Maria Delgado, Shivam Pandey, Erwin T. Lau, Christopher Carr, Harrison Cook, Daisuke Nagai, Daniel Angles-Alcazar, Francisco Villaescusa-Navarro, Greg L. Bryan
   2403.10609
4. **Galaxy dispersion measured by Fast Radio Bursts as a probe of baryonic feedback models**
   Alexander Theis, Steffen Hagstotz, Robert Reischke, Jochen Weller
   2403.08611
5. **Probing the Circum-Galactic Medium with Fast Radio Bursts: Insights from the CAMELS Simulations**
   Isabel Medlock, Daisuke Nagai, Priyanka Singh, Benjamin Oppenheimer, Daniel Angles Alcazar, Francisco Villaescusa-Navarro
   2403.02313
6. **Cosmological multifield emulator**
   Sambatra Andrianomena, Sultan Hassan, Francisco Villaescusa-Navarro
   2402.10997
7. **A field-level emulator for modeling baryonic effects across hydrodynamic simulations**
   Divij Sharma, Biwei Dai, Francisco Villaescusa-Navarro, Uros Seljak
   2401.15891
8. **Modeling the Kinematics of Central and Satellite Galaxies Using Normalizing Flows**
   K.J. Kwon, ChangHoon Hahn
   2401.12318
9. **Cosmological Field Emulation and Parameter Inference with Diffusion Models**
   Nayantara Mudur, Carolina Cuesta-Lazaro, Douglas P. Finkbeiner
   2312.07534
10. **Towards out-of-distribution generalization in large-scale astronomical surveys: robust networks learn similar representations**

Yash Gondhalekar, Sultan Hassan, Naomi Saphra, Sambatra Andrianomena
2311.18007

11. **Learning an Effective Evolution Equation for Particle-Mesh Simulations Across Cosmologies**
Nicolas Payot, Pablo Lemos, Laurence Perreault-Levasseur, Carolina Cuesta-Lazaro, Chirag Modi, Yashar Hezaveh
2311.18017

12. **Taming assembly bias for primordial non-Gaussianity**
Emanuele Fondi, Licia Verde, Francisco Villaescusa-Navarro, Marco Baldi, William R. Coulton, Gabriel Jung, Dionysios Karagiannis, Michele Liguori, Andrea Ravenni, Benjamin D. Wandelt
2311.10088

13. **Probabilistic reconstruction of Dark Matter fields from biased tracers using diffusion models**
Core Francisco Park, Victoria Ono, Nayantara Mudur, Yueying Ni, Carolina Cuesta-Lazaro
2311.08558

14. **Baryonic Imprints on DM Halos: the concentration-mass relation and its dependence on halo and galaxy properties**
Mufan Shao, Dhayaa Anbajagane
2311.03491

15. **Domain Adaptive Graph Neural Networks for Constraining Cosmological Parameters Across Multiple Data Sets**
Andrea Roncoli, Aleksandra Ćiprijanović, Maggie Voetberg, Francisco Villaescusa-Navarro, Brian Nord
2311.01588

16. **HIDM: Emulating Large Scale HI Maps using Score-based Diffusion Models**
Sultan Hassan, Sambatra Andrianomena
2311.00833

17. **Latent space representations of cosmological fields**
Sambatra Andrianomena, Sultan Hassan
2311.00799

18. **Field-level simulation-based inference with galaxy catalogs: the impact of systematic effects**
Natalí S. M. de Santi, Francisco Villaescusa-Navarro, L. Raul Abramo, Helen Shao, Lucia A. Perez, Tiago Castro, Yueying Ni, Christopher C. Lovell, Elena Hernandez-Martinez, Federico Marinacci, David N. Spergel, Klaus Dolag, Lars Hernquist, Mark Vogelsberger
2310.15234

19. **Cosmology with Galaxy Photometry Alone**
ChangHoon Hahn, Francisco Villaescusa-Navarro, Peter Melchior, Romain Teyssier
2310.08634

20. **Exploring chemical enrichment of the intracluster medium with the Line Emission Mapper**
François Mernier, Yuanyuan Su, Maxim Markevitch, Congyao Zhang, Aurora Simionescu, Elena Rasia, Sheng-Chieh Lin, Irina Zhuravleva, Arnab Sarkar, Ralph P. Kraft, Anna Ogorzalek, Mohammadreza Ayromlou, William R. Forman, Christine Jones, Joel N. Bregman, Stefano Ettori, Klaus Dolag, Veronica Biffi, Eugene Churazov, Ming Sun, John ZuHone, Ákos Bogdán, Ildar I. Khabibullin, Norbert Werner, Nhut Truong, Priyanka Chakraborty, Stephen A. Walker, Mark Vogelsberger, Annalisa Pillepich, Mohammad S. Mirakhor
2310.04499

21. **Cosmology with multiple galaxies**
Chaitanya Chawak, Francisco Villaescusa-Navarro, Nicolas Echeverri Rojas, Yueying Ni, ChangHoon Hahn, Daniel Angles-Alcazar

2309.12048

22. **An Observationally Driven Multifield Approach for Probing the Circum-Galactic Medium with Convolutional Neural Networks**
Naomi Gluck, Benjamin D. Oppenheimer, Daisuke Nagai, Francisco Villaescusa-Navarro, Daniel Angles-Alcazar
2309.07912

23. **CASCO: Cosmological and AStrophysical parameters from Cosmological simulations and Observations – I. Constraining physical processes in local star-forming galaxies**
Valerio Busillo, Crescenzo Tortora, Nicola R. Napolitano, Leon V. E. Koopmans, Giovanni Covone, Fabrizio Gentile, Leslie K. Hunt
2308.14822

24. **Data Compression and Inference in Cosmology with Self-Supervised Machine Learning**
Aizhan Akhmetzhanova, Siddharth Mishra-Sharma, Cora Dvorkin
2308.09751

25. **Learnable wavelet neural networks for cosmological inference**
Christian Pedersen, Michael Eickenberg, Shirley Ho
2307.14362

26. **Cosmological baryon spread and impact on matter clustering in CAMELS**
Matthew Gebhardt, Daniel Angles-Alcazar, Josh Borrow, Shy Genel, Francisco Villaescusa-Navarro, Yueying Ni, Christopher Lovell, Daisuke Nagai, Romeel Dave, Federico Marinacci, Mark Vogelsberger, Lars Hernquist
2307.11832

27. **A Hierarchy of Normalizing Flows for Modelling the Galaxy-Halo Relationship**
Christopher C. Lovell, Sultan Hassan, Daniel Anglés-Alcázar, Greg Bryan, Giulio Fabbian, Shy Genel, ChangHoon Hahn, Kartheik Iyer, James Kwon, Natalí de Santi, Francisco Villaescusa-Navarro
2307.06967

28. **An Exploration of AGN and Stellar Feedback Effects in the Intergalactic Medium via the Low Redshift Lyman-α Forest**
Megan Taylor Tillman, Blakesley Burkhart, Stephanie Tonnesen, Simeon Bird, Greg L. Bryan, Daniel Anglés-Alcázar, Sultan Hassan, Rachel S. Somerville, Romeel Davé, Federico Marinacci, Lars Hernquist, Mark Vogelsberger
2307.06360

29. **Probabilistic matching of real and generated data statistics in generative adversarial networks**
Philipp Pilar, Niklas Wahlström
2306.10943

30. **Multi-Epoch Machine Learning 2: Identifying physical drivers of galaxy properties in simulations**
Robert McGibbon, Sadegh Khochfar
2306.07728

31. **Forecasting the power of Higher Order Weak Lensing Statistics with automatically differentiable simulations**
Denise Lanzieri, François Lanusse, Chirag Modi, Benjamin Horowitz, Joachim Harnois-Déraps, Jean-Luc Starck, The LSST Dark Energy Science Collaboration
2305.07531

32. **Interpreting Sunyaev-Zel'dovich observations with MillenniumTNG: Mass and environment scaling relations**

Boryana Hadzhiyska, Simone Ferraro, Rüdiger Pakmor, Sownak Bose, Ana Maria Delgado, César Hernández-Aguayo, Rahul Kannan, Volker Springel, Simon D. M. White, Lars Hernquist
2305.00992

33. **Cosmology with one galaxy? – The ASTRID model and robustness**
Nicolas Echeverri, Francisco Villaescusa-Navarro, Chaitanya Chawak, Yueying Ni, ChangHoon Hahn, Elena Hernandez-Martinez, Romain Teyssier, Daniel Angles-Alcazar, Klaus Dolag, Tiago Castro
2304.06084

34. **The CAMELS project: Expanding the galaxy formation model space with new ASTRID and 28-parameter TNG and SIMBA suites**
Yueying Ni, Shy Genel, Daniel Anglés-Alcázar, Francisco Villaescusa-Navarro, Yongseok Jo, Simeon Bird, Tiziana Di Matteo, Rupert Croft, Nianyi Chen, Natalí S. M. de Santi, Matthew Gebhardt, Helen Shao, Shivam Pandey, Lars Hernquist, Romeel Dave
2304.02096

35. **Invertible mapping between fields in CAMELS**
Sambatra Andrianomena, Sultan Hassan, Francisco Villaescusa-Navarro
2303.07473

36. **A universal equation to predict $\Omega_{\rm m}$ from halo and galaxy catalogues**
Helen Shao, Natali S. M. de Santi, Francisco Villaescusa-Navarro, Romain Teyssier, Yueying Ni, Daniel Angles-Alcazar, Shy Genel, Ulrich P. Steinwandel, Elena Hernandez-Martınez, Klaus Dolag, Christopher C. Lovell, Lehman H. Garrison, Eli Visbal, Mihir Kulkarni, Lars Hernquist, Tiago Castro, Mark Vogelsberger
2302.14591 | video |

37. **Robust field-level likelihood-free inference with galaxies**
Natalí S. M. de Santi, Helen Shao, Francisco Villaescusa-Navarro, L. Raul Abramo, Romain Teyssier, Pablo Villanueva-Domingo, Yueying Ni, Daniel Anglés-Alcázar, Shy Genel, Elena Hernandez-Martinez, Ulrich P. Steinwandel, Christopher C. Lovell, Klaus Dolag, Tiago Castro, Mark Vogelsberger
2302.14101 | video |

38. **Topological data analysis reveals differences between simulated galaxies and dark matter haloes**
Aaron Ouellette, Gilbert Holder, Ely Kerman
2302.01363

39. **Perturbation-theory informed integrators for cosmological simulations**
Florian List, Oliver Hahn
2301.09655

40. **On the choice of the most suitable indicator for the assembly state of dark matter haloes through cosmic time**
David Vallés-Pérez, Susana Planelles, Óscar Monllor-Berbegal, Vicent Quilis
2301.02253

41. **Predicting the impact of feedback on matter clustering with machine learning in CAMELS**
Ana Maria Delgado, Daniel Angles-Alcazar, Leander Thiele, Michelle Ntampaka, Shivam Pandey, Kai Lehman, Rachel S. Somerville, Shy Genel, Francisco Villaescusa-Navarro
2301.02231

42. **Inferring the impact of feedback on the matter distribution using the Sunyaev Zel'dovich effect: Insights from CAMELS simulations and ACT+DES data**
Shivam Pandey, Kai Lehman, Eric J. Baxter, Yueying Ni, Daniel Anglés-Alcázar, Shy Genel, Francisco Villaescusa-Navarro, Ana Maria Delgado, Tiziana di Matteo
2301.02186

43. **Baryonic Imprints on DM Halos: The concentration-mass relation in the CAMELS simulations**
Mufan Shao, Dhayaa Anbajagane, Chihway Chang
2212.05964

44. **Calibrating cosmological simulations with implicit likelihood inference using galaxy growth observables**
Yongseok Jo, Shy Genel, Benjamin Wandelt, Rachel Somerville, Francisco Villaescusa-Navarro, Greg L. Bryan, Daniel Angles-Alcazar, Daniel Foreman-Mackey, Dylan Nelson, Ji-hoon Kim
2211.16461

45. **X-ray Absorption Lines in the Warm-Hot Intergalactic Medium: Probing Chandra observations with the CAMEL simulations**
Amanda Butler Contreras, Erwin T. Lau, Benjamin D. Oppenheimer, Ákos Bogdán, Megan Tillman, Daisuke Nagai, Orsolya E. Kovács, Blakesley Burkhart
2211.15675

46. **HIGlow: Conditional Normalizing Flows for High-Fidelity HI Map Modeling**
Roy Friedman, Sultan Hassan
2211.12724

47. **Can denoising diffusion probabilistic models generate realistic astrophysical fields?**
Nayantara Mudur, Douglas P. Finkbeiner
2211.12444

48. **Emulating cosmological multifields with generative adversarial networks**
Sambatra Andrianomena, Francisco Villaescusa-Navarro, Sultan Hassan
2211.05000

49. **Evidence for efficient long-range AGN jet feedback from the low redshift Lyman- forest**
Megan Taylor Tillman, Blakesley Burkhart, Stephanie Tonnesen, Simeon Bird, Greg L. Bryan, Daniel Angles-Alcazar, Romeel Dave, Shy Genel
2210.02467

50. **Robust field-level inference with dark matter halos**
Helen Shao, Francisco Villaescusa-Navarro, Pablo Villanueva-Domingo, Romain Teyssier, Lehman H. Garrison, Marco Gatti, Derek Inman, Yueying Ni, Ulrich P. Steinwandel, Mihir Kulkarni, Eli Visbal, Greg L. Bryan, Daniel Angles-Alcazar, Tiago Castro, Elena Hernandez-Martinez, Klaus Dolag
2209.06843 | video |

51. **The SZ flux-mass (Y-M) relation at low halo masses: improvements with symbolic regression and strong constraints on baryonic feedback**
Digvijay Wadekar, Leander Thiele, J. Colin Hill, Shivam Pandey, Francisco Villaescusa-Navarro, David N. Spergel, Miles Cranmer, Daisuke Nagai, Daniel Anglés-Alcázar, Shirley Ho, Lars Hernquist
2209.02075 | video |

52. **Studying the Warm Hot Intergalactic Medium in emission: a reprise**
Gabriele Parimbelli, Enzo Branchini, Matteo Viel, Francisco Villaescusa-Navarro, John ZuHone
2209.00657

53. **Predictive uncertainty on improved astrophysics recovery from multifield cosmology**
Sambatra Andrianomena, Sultan Hassan
2208.08927

54. **Hybrid Physical-Neural ODEs for Fast N-body Simulations**
Denise Lanzieri, François Lanusse, Jean-Luc Starck
2207.05509

55. **The halo finding problem revisited: a deep revision of the ASOHF code**
David Valles-Perez, Susana Planelles, Vicent Quilis
2205.02245

56. **Learning cosmology and clustering with cosmic graphs**
Pablo Villanueva-Domingo, Francisco Villaescusa-Navarro
2204.13713

57. **Constraining cosmology with machine learning and galaxy clustering: the CAMELS-SAM suite**
Lucia A. Perez, Shy Genel, Francisco Villaescusa-Navarro, Rachel S. Somerville, Austen Gabrielpillai, Daniel Anglés-Alcázar, Benjamin D. Wandelt, L.Y. Aaron Yung
| 2204.02408 | video |

58. **Breaking baryon-cosmology degeneracy with the electron density power spectrum**
Andrina Nicola, Francisco Villaescusa-Navarro, David N. Spergel, Jo Dunkley, Daniel Anglés-Alcázar, Romeel Davé, Shy Genel, Lars Hernquist, Daisuke Nagai, Rachel S. Somerville, Benjamin D. Wandelt
| 2201.04142 | video |

59. **The Circumgalactic Medium from the CAMELS Simulations: Forecasting Constraints on Feedback Processes from Future Sunyaev-Zeldovich Observations**
Emily Moser, Nicholas Battaglia, Daisuke Nagai, Erwin Lau, Luis Fernando Machado Poletti Valle, Francisco Villaescusa-Navarro, Stefania Amodeo, Daniel Angles-Alcazar, Greg L. Bryan, Romeel Dave, Lars Hernquist, Mark Vogelsberger
| 2201.02708 | blog |

60. **Cosmology with one galaxy?**
Francisco Villaescusa-Navarro, Jupiter Ding, Shy Genel, Stephanie Tonnesen, Valentina La Torre, David N. Spergel, Romain Teyssier, Yin Li, Caroline Heneka, Pablo Lemos, Daniel Anglés-Alcázar, Daisuke Nagai, Mark Vogelsberger
| 2201.02202 | video | Quanta Magazine article | New Yorker article |

61. **The CAMELS project: public data release**
Francisco Villaescusa-Navarro, Shy Genel, Daniel Anglés-Alcázar, Lucia A. Perez, Pablo Villanueva-Domingo, Digvijay Wadekar, Helen Shao, Faizan G. Mohammad, Sultan Hassan, Emily Moser, Erwin T. Lau, Luis Fernando Machado Poletti Valle, Andrina Nicola, Leander Thiele, Yongseok Jo, Oliver H. E. Philcox, Benjamin D. Oppenheimer, Megan Tillman, ChangHoon Hahn, Neerav Kaushal, Alice Pisani, Matthew Gebhardt, Ana Maria Delgado, Joyce Caliendo, Christina Kreisch, Kaze W.K. Wong, William R. Coulton, Michael Eickenberg, Gabriele Parimbelli, Yueying Ni, Ulrich P. Steinwandel, Valentina La Torre, Romeel Dave, Nicholas Battaglia, Daisuke Nagai, David N. Spergel, Lars Hernquist, Blakesley Burkhart, Desika Narayanan, Benjamin Wandelt, Rachel S. Somerville, Greg L. Bryan, Matteo Viel, Yin Li, Vid Irsic, Katarina Kraljic, Mark Vogelsberger
| 2201.01300 | video | press release |

62. **Augmenting astrophysical scaling relations with machine learning : application to reducing the SZ flux-mass scatter**
Digvijay Wadekar, Leander Thiele, Francisco Villaescusa-Navarro, J. Colin Hill, David N. Spergel, Miles Cranmer, Nicholas Battaglia, Daniel Anglés-Alcázar, Lars Hernquist, Shirley Ho
| 2201.01305 | video | press release 1 | press release 2 |

63. **Percent-level constraints on baryonic feedback with spectral distortion measurements**
Leander Thiele, Digvijay Wadekar, J. Colin Hill, Nicholas Battaglia, Jens Chluba, Francisco Villaescusa-Navarro, Lars Hernquist, Mark Vogelsberger, Daniel Anglés-Alcázar, Federico Marinacci
| 2201.01663 | video | blog |

64. **Weighing the Milky Way and Andromeda with Artificial Intelligence**

Pablo Villanueva-Domingo, Francisco Villaescusa-Navarro, Shy Genel, Daniel Anglés-Alcázar, Lars Hernquist, Federico Marinacci, David N. Spergel, Mark Vogelsberger, Desika Narayanan
| 2111.14874 | video (26'-56') |

65. **Inferring halo masses with Graph Neural Networks**
Pablo Villanueva-Domingo, Francisco Villaescusa-Navarro, Daniel Anglés-Alcázar, Shy Genel, Federico Marinacci, David N. Spergel, Lars Hernquist, Mark Vogelsberger, Romeel Dave, Desika Narayanan
| 2111.08683 | video (26'-51') |

66. **HIFlow: Generating Diverse HI Maps Conditioned on Cosmology using Normalizing Flow**
Sultan Hassan, Francisco Villaescusa-Navarro, Benjamin Wandelt, David N. Spergel, Daniel Anglés-Alcázar, Shy Genel, Miles Cranmer, Greg L. Bryan, Romeel Davé, Rachel S. Somerville, Michael Eickenberg, Desika Narayanan, Shirley Ho, Sambatra Andrianomena
| 2110.02983 | video |

67. **The CAMELS Multifield Dataset: Learning the Universe's Fundamental Parameters with Artificial Intelligence**
Francisco Villaescusa-Navarro, Shy Genel, Daniel Angles-Alcazar, Leander Thiele, Romeel Dave, Desika Narayanan, Andrina Nicola, Yin Li, Pablo Villanueva-Domingo, Benjamin Wandelt, David N. Spergel, Rachel S. Somerville, Jose Manuel Zorrilla Matilla, Faizan G. Mohammad, Sultan Hassan, Helen Shao, Digvijay Wadekar, Michael Eickenberg, Kaze W.K. Wong, Gabriella Contardo, Yongseok Jo, Emily Moser, Erwin T. Lau, Luis Fernando Machado Poletti Valle, Lucia A. Perez, Daisuke Nagai, Nicholas Battaglia, Mark Vogelsberger
| 2109.10915 | website |

68. **Robust marginalization of baryonic effects for cosmological inference at the field level**
Francisco Villaescusa-Navarro, Shy Genel, Daniel Angles-Alcazar, David N. Spergel, Yin Li, Benjamin Wandelt, Leander Thiele, Andrina Nicola, Jose Manuel Zorrilla Matilla, Helen Shao, Sultan Hassan, Desika Narayanan, Romeel Dave, Mark Vogelsberger
| 2109.10360 | astrobites |

69. **Multifield Cosmology with Artificial Intelligence**
Francisco Villaescusa-Navarro, Daniel Anglés-Alcázar, Shy Genel, David N. Spergel, Yin Li, Benjamin Wandelt, Andrina Nicola, Leander Thiele, Sultan Hassan, Jose Manuel Zorrilla Matilla, Desika Narayanan, Romeel Dave, Mark Vogelsberger
| 2109.09747 | video (17'-38') |

70. **Inpainting hydrodynamical maps with deep learning**
Faizan G. Mohammad, Francisco Villaescusa-Navarro, Shy Genel, Daniel Angles-Alcazar, Mark Vogelsberger
2109.07070

71. **Finding universal relations in subhalo properties with artificial intelligence**
Helen Shao, Francisco Villaescusa-Navarro, Shy Genel, David N. Spergel, Daniel Angles-Alcazar, Lars Hernquist, Romeel Dave, Desika Narayanan, Gabriella Contardo, Mark Vogelsberger
| 2109.04484 | video | blog |

72. **Neural networks as optimal estimators to marginalize over baryonic effects**
Francisco Villaescusa-Navarro, Benjamin D. Wandelt, Daniel Anglés-Alcázar, Shy Genel, Jose Manuel Zorrilla Mantilla, Shirley Ho, David N. Spergel
2011.05992

73. **The CAMELS project: Cosmology and Astrophysics with MachinE Learning Simulations**
Francisco Villaescusa-Navarro, Daniel Anglés-Alcázar, Shy Genel, David N. Spergel, Rachel S. Somerville, Romeel Dave, Annalisa Pillepich, Lars Hernquist, Dylan Nelson, Paul Torrey, Desika Narayanan, Yin Li, Oliver Philcox, Valentina La Torre, Ana Maria Delgado, Shirley Ho, Sultan Hassan, Blakesley Burkhart, Digvijay Wadekar, Nicholas Battaglia, Gabriella Contardo

| 2010.00619 | video (0'-13') | podcast (in Italian) | blog | press release |

Attention: All the simulations in the IllustrisTNG, SIMBA, Astrid, and N-body suites are publicly available. To access other simulations (e.g. Magneticum, Swift-EAGLE, Ramses…etc) please fill up this form.

CHAPTER

FOUR

DATA ACCESS

CAMELS data is stored at the Rusty cluster of the Flatiron Institute in New York City and its data can be accessed in different ways:

4.1 Binder

Binder is a system that allows users to read and manipulate data that is hosted at the Flatiron Institute through either a Jupyter notebook or a unix shell. The user can find some basic documentation here. All CAMELS data can be accessed, read, and manipulated through Binder.

Warning: Two important things need to be taken into account when using Binder. First, the Binder environment is ephemeral - after a few days of inactivity its contents are deleted, so one has to be vigilant about downloading any analysis results in time. Second, Binder is not designed to carry out long and heavy calculations. In this case we recommend the user to download the data and work with it locally.

4.2 Globus

The full CAMELS data can be downloaded via globus, an online system designed to efficiently transfer large amounts of data. This is the method we recommend to transfer the data.

Globus link

4.3 url

We provide access to the full CAMELS data via a simple uniform resource locator (url). We do not recommend downloading large amounts of data through this system, as can be slow and unstable. However, for small or individual files it may be convenient.

URL link

4.4 FlatHUB

FlatHUB is a platform that allows users to explore and compare data from different simulations by browsing and filtering the data, making simple preview plots, and downloading sub-samples of the data. We provide access to the SUBFIND halo and subhalo catalogues of the IllustrisTNG and SIMBA suites through this platform.

Link to FlatHUB

4.5 Rusty

Users with an account on the Flatiron Institute Rusty cluster, can find all CAMELS data in `/mnt/ceph/users/camels/PUBLIC_RELEASE`.

CHAPTER

FIVE

CITATION

Please consider citing the relevant papers if you use CAMELS data.

5.1 CAMELS

If you use CAMELS data you may consider citing the CAMELS presentation, first data release, and second data release papers:

```
@ARTICLE{CAMELS_presentation,
author = {{Villaescusa-Navarro}, Francisco and {Angl{\'e}s-Alc{\'a}zar}, Daniel and
↪{Genel}, Shy and {Spergel}, David N. and {Somerville}, Rachel S. and {Dave}, Romeel␣
↪and {Pillepich}, Annalisa and {Hernquist}, Lars and {Nelson}, Dylan and {Torrey}, Paul␣
↪and {Narayanan}, Desika and {Li}, Yin and {Philcox}, Oliver and {La Torre}, Valentina␣
↪and {Maria Delgado}, Ana and {Ho}, Shirley and {Hassan}, Sultan and {Burkhart},␣
↪Blakesley and {Wadekar}, Digvijay and {Battaglia}, Nicholas and {Contardo}, Gabriella␣
↪and {Bryan}, Greg L.},
title = "{The CAMELS Project: Cosmology and Astrophysics with Machine-learning␣
↪Simulations}",
journal = {\apj},
keywords = {Cosmology, Cosmological parameters from large-scale structure, Galaxy␣
↪formation, Astrostatistics, 343, 340, 595, 1882, Astrophysics - Cosmology and␣
↪Nongalactic Astrophysics, Astrophysics - Astrophysics of Galaxies, Astrophysics -␣
↪Instrumentation and Methods for Astrophysics},
year = 2021,
month = jul,
volume = {915},
number = {1},
eid = {71},
pages = {71},
doi = {10.3847/1538-4357/abf7ba},
archivePrefix = {arXiv},
eprint = {2010.00619},
primaryClass = {astro-ph.CO},
adsurl = {https://ui.adsabs.harvard.edu/abs/2021ApJ...915...71V},
adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}

@ARTICLE{CAMELS_DR1,
author = {{Villaescusa-Navarro}, Francisco and {Genel}, Shy and {Angl{\'e}s-Alc{\'a}zar},
↪ Daniel and {Perez}, Lucia A. and {Villanueva-Domingo}, Pablo and {Wadekar}, Digvijay␣
```

(continues on next page)

(continued from previous page)

```
↪and {Shao}, Helen and {Mohammad}, Faizan G. and {Hassan}, Sultan and {Moser}, Emily␣
↪and {Lau}, Erwin T. and {Machado Poletti Valle}, Luis Fernando and {Nicola}, Andrina␣
↪and {Thiele}, Leander and {Jo}, Yongseok and {Philcox}, Oliver H.~E. and {Oppenheimer},
↪ Benjamin D. and {Tillman}, Megan and {Hahn}, ChangHoon and {Kaushal}, Neerav and
↪{Pisani}, Alice and {Gebhardt}, Matthew and {Delgado}, Ana Maria and {Caliendo}, Joyce␣
↪and {Kreisch}, Christina and {Wong}, Kaze W.~K. and {Coulton}, William R. and
↪{Eickenberg}, Michael and {Parimbelli}, Gabriele and {Ni}, Yueying and {Steinwandel},␣
↪Ulrich P. and {La Torre}, Valentina and {Dave}, Romeel and {Battaglia}, Nicholas and
↪{Nagai}, Daisuke and {Spergel}, David N. and {Hernquist}, Lars and {Burkhart},␣
↪Blakesley and {Narayanan}, Desika and {Wandelt}, Benjamin and {Somerville}, Rachel S.␣
↪and {Bryan}, Greg L. and {Viel}, Matteo and {Li}, Yin and {Irsic}, Vid and {Kraljic},␣
↪Katarina and {Marinacci}, Federico and {Vogelsberger}, Mark},
title = "{The CAMELS Project: Public Data Release}",
journal = {\apjs},
keywords = {Cosmology, Hydrodynamical simulations, Astrostatistics, Galaxy formation,␣
↪Astrophysics - Cosmology and Nongalactic Astrophysics, Astrophysics - Astrophysics of␣
↪Galaxies, Astrophysics - Instrumentation and Methods for Astrophysics, Computer␣
↪Science - Artificial Intelligence, Computer Science - Machine Learning},
year = 2023,
month = apr,
volume = {265},
number = {2},
eid = {54},
pages = {54},
doi = {10.3847/1538-4365/acbf47},
archivePrefix = {arXiv},
eprint = {2201.01300},
primaryClass = {astro-ph.CO},
adsurl = {https://ui.adsabs.harvard.edu/abs/2023ApJS..265...54V},
adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}

@ARTICLE{CAMELS_DR2,
author = {{Ni}, Yueying and {Genel}, Shy and {Angl{\'e}s-Alc{\'a}zar}, Daniel and
↪{Villaescusa-Navarro}, Francisco and {Jo}, Yongseok and {Bird}, Simeon and {Di Matteo},
↪ Tiziana and {Croft}, Rupert and {Chen}, Nianyi and {de Santi}, Natal{\'\i} S.~M. and
↪{Gebhardt}, Matthew and {Shao}, Helen and {Pandey}, Shivam and {Hernquist}, Lars and
↪{Dave}, Romeel},
title = "{The CAMELS Project: Expanding the Galaxy Formation Model Space with New ASTRID␣
↪and 28-parameter TNG and SIMBA Suites}",
journal = {\apj},
keywords = {Large-scale structure of the universe, Hydrodynamical simulations, 902, 767,␣
↪Astrophysics - Cosmology and Nongalactic Astrophysics, Astrophysics - Astrophysics of␣
↪Galaxies, Computer Science - Machine Learning},
year = 2023,
month = dec,
volume = {959},
number = {2},
eid = {136},
pages = {136},
doi = {10.3847/1538-4357/ad022a},
archivePrefix = {arXiv},
```

(continues on next page)

(continued from previous page)

```
eprint = {2304.02096},
primaryClass = {astro-ph.CO},
adsurl = {https://ui.adsabs.harvard.edu/abs/2023ApJ...959..136N},
adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}
```

5.2 CAMELS-SAM

If you use data from CAMELS-SAM please consider citing the CAMELS-SAM paper

```
@ARTICLE{CAMELS-SAM,
author = {{Perez}, Lucia A. and {Genel}, Shy and {Villaescusa-Navarro}, Francisco and
↪{Somerville}, Rachel S. and {Gabrielpillai}, Austen and {Angl{\'e}s-Alc{\'a}zar},␣
↪Daniel and {Wandelt}, Benjamin D. and {Yung}, L.~Y. Aaron},
title = "{Constraining Cosmology with Machine Learning and Galaxy Clustering: The CAMELS-
↪SAM Suite}",
journal = {\apj},
keywords = {Large-scale structure of the universe, Neural networks, Cosmological␣
↪parameters from large-scale structure, N-body simulations, Galaxy formation, 902, 1933,
↪ 340, 1083, 595, Astrophysics - Astrophysics of Galaxies, Astrophysics - Cosmology and␣
↪Nongalactic Astrophysics},
year = 2023,
month = sep,
volume = {954},
number = {1},
eid = {11},
pages = {11},
doi = {10.3847/1538-4357/accd52},
archivePrefix = {arXiv},
eprint = {2204.02408},
primaryClass = {astro-ph.GA},
adsurl = {https://ui.adsabs.harvard.edu/abs/2023ApJ...954...11P},
adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}
```

5.3 CAMELS Multifield Dataset

If you use data from the CAMELS Multifield Dataset (CMD) consider citing the CMD paper

```
@ARTICLE{CMD,
author = {{Villaescusa-Navarro}, Francisco and {Genel}, Shy and {Angl{\'e}s-Alc{\'a}zar},
↪ Daniel and {Thiele}, Leander and {Dave}, Romeel and {Narayanan}, Desika and {Nicola},␣
↪Andrina and {Li}, Yin and {Villanueva-Domingo}, Pablo and {Wandelt}, Benjamin and
↪{Spergel}, David N. and {Somerville}, Rachel S. and {Zorrilla Matilla}, Jose Manuel␣
↪and {Mohammad}, Faizan G. and {Hassan}, Sultan and {Shao}, Helen and {Wadekar},␣
↪Digvijay and {Eickenberg}, Michael and {Wong}, Kaze W.~K. and {Contardo}, Gabriella␣
↪and {Jo}, Yongseok and {Moser}, Emily and {Lau}, Erwin T. and {Machado Poletti Valle},␣
↪Luis Fernando and {Perez}, Lucia A. and {Nagai}, Daisuke and {Battaglia}, Nicholas and
```

(continues on next page)

(continued from previous page)

```
↪{Vogelsberger}, Mark},
title = "{The CAMELS Multifield Data Set: Learning the Universe's Fundamental Parameters␣
↪with Artificial Intelligence}",
journal = {\apjs},
keywords = {Cosmological parameters from large-scale structure, Magnetohydrodynamical␣
↪simulations, Astrostatistics, N-body simulations, 340, 1966, 1882, 1083, Computer␣
↪Science - Machine Learning, Astrophysics - Cosmology and Nongalactic Astrophysics,␣
↪Astrophysics - Astrophysics of Galaxies, Astrophysics - Instrumentation and Methods␣
↪for Astrophysics, Computer Science - Computer Vision and Pattern Recognition},
year = 2022,
month = apr,
volume = {259},
number = {2},
eid = {61},
pages = {61},
doi = {10.3847/1538-4365/ac5ab0},
archivePrefix = {arXiv},
eprint = {2109.10915},
primaryClass = {cs.LG},
adsurl = {https://ui.adsabs.harvard.edu/abs/2022ApJS..259...61V},
adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}
```

CHAPTER

SIX

GENERAL DESCRIPTION

As of January 2024, CAMELS contains 12,903 cosmological simulations: 5,164 N-body and 7,739 hydrodynamic simulations.

All simulations follow the evolution of 256^3 dark matter particles and 256^3 initial gas resolution elements (only for the hydrodynamic) within a periodic comoving volume of $(25\ h^{-1}\mathrm{Mpc})^3$ from $z = 127$ down to $z = 0$.

Note: CAMELS is expanding and will soon include simulations with 512^3 dark matter particles and 512^3 gas resolution elements in periodic boxes of $(50\ h^{-1}\mathrm{Mpc})^3$. CAMELS has recently expanded to include a set of zoom-in simulations of massive halos, which are centered in periodic boxes of $(200\ h^{-1}\mathrm{Mpc})^3$.

For each simulation, we store multiple snapshots. We also keep a variety of post-processed data such as halo catalogs, power spectra…etc; see *Data organization*.

6.1 Organization

The simulations can be classified into suites, volumes, and sets, depending on the code used to run them, their volume, and how their parameters are organized. See *Organization* for further details. The table below shows the number of available hydrodynamic and N-body simulations:

Hydrodynamic simulations							
Suite	Set **SB**	**LH**	**1P**	**CV**	**EX**	**BE**	**Total**
IllustrisTNG	2048 *(28 parameters)*	1000	61	27	4	27	3167
SIMBA		1000	61	27	4	27	1119
Astrid	1024 *(7 parameters)*	1000	61	27	4		2116
Magneticum		50		27			77
Swift-EAGLE		1000	61	27			1088
Ramses	128 *(5 parameters)*	–	11	27			166
Enzo		–	5	1			6
All	3200	4050	260	163	12	54	**7739**

N-body simulations							
Suite	Set SB	LH	1P	CV	EX	BE	Total
IllustrisTNG	1024 *(5 parameters)*	1000	61	27	1	27	2140
SIMBA		1000					1089
Astrid	1024 *(3 parameters)*	1000					2113
Magneticum							89
Swift-EAGLE							89
Ramses		–					89
Enzo		–					89
All	2048	3000	61	27	1	27	**5164**

The value of the cosmological, astrophysical, and initial random seed of the different simulations can be found *Parameters*.

6.2 Codes

All N-body simulations have been run with the Gadget-III code, while the hydrodynamic simulations have been run with different codes: AREPO, GIZMO, MP-Gadget, OpenGadget, Swift, Ramses, and Enzo. See *Codes* for details on the different codes and subgrid physics models available in CAMELS.

6.3 Redshifts

Important: A reorganization of the data has been performed in 2024 in order to enhance its uniformity and simplicity. This will require slight changes to existing codes that access the data.

- Snapshot numbers in the IllustrisTNG and SIMBA suites, where simulations have only 34 snapshots, have been updated to match the numbering in the Astrid suite (and some TNG simulations) that have 91 snapshots. For example, where 33 used to be the z=0 snapshot, now it is 90 uniformly for all suites.

All simulations have 91 snapshots. This file contains the scale factors of the snapshots. The user can click below to see the redshift, scale factor, and snapshot number for the different snapshots:

Redshift	Scale Factor	34 snapshots	91 snapshots
14.99	0.063	—	000
13.34	0.070	—	001
11.98	0.077	—	002
11.20	0.082	—	003
10.48	0.087	—	004
9.69	0.094	—	005
9.00	0.100	—	006
8.49	0.105	—	007
8.01	0.111	—	008

continues on next page

Table 1 – continued from previous page

Redshift	Scale Factor	34 snapshots	91 snapshots
7.60	0.116	—	009
7.24	0.121	—	010
6.89	0.127	—	011
6.56	0.132	—	012
6.28	0.137	—	013
6.01	0.143	000	014
5.75	0.148	—	015
5.50	0.154	—	016
5.23	0.161	—	017
5.00	0.167	001	018
4.80	0.172	—	019
4.61	0.178	—	020
4.45	0.183	—	021
4.30	0.189	—	022
4.15	0.194	—	023
4.01	0.200	002	024
3.87	0.205	—	025
3.73	0.211	—	026
3.62	0.216	—	027
3.49	0.223	003	028
3.36	0.229	—	029
3.24	0.236	—	030
3.12	0.242	—	031
3.01	0.249	004	032
2.90	0.257	—	033
2.80	0.263	005	034
2.72	0.269	—	035
2.63	0.276	006	036
2.54	0.282	—	037
2.46	0.289	007	038
2.38	0.296	—	039
2.30	0.303	008	040
2.22	0.310	—	041
2.15	0.318	009	042
2.07	0.325	—	043
2.00	0.333	010	044
1.93	0.341	—	045
1.86	0.349	011	046
1.80	0.358	—	047
1.73	0.366	012	048
1.67	0.375	—	049
1.60	0.384	013	050
1.54	0.393	—	051
1.48	0.403	014	052
1.43	0.412	—	053
1.37	0.422	015	054
1.30	0.434	—	055
1.26	0.443	016	056
1.20	0.455	—	057
1.14	0.466	017	058

continues on next page

Table 1 – continued from previous page

Redshift	Scale Factor	34 snapshots	91 snapshots
1.09	0.478	—	059
1.05	0.489	018	060
1.00	0.501	—	061
0.95	0.513	019	062
0.90	0.525	—	063
0.86	0.538	020	064
0.82	0.550	—	065
0.77	0.564	021	066
0.73	0.577	—	067
0.69	0.591	022	068
0.65	0.605	—	069
0.61	0.620	023	070
0.58	0.635	—	071
0.54	0.650	024	072
0.50	0.665	—	073
0.47	0.681	025	074
0.43	0.698	—	075
0.40	0.714	026	076
0.37	0.731	—	077
0.34	0.749	027	078
0.30	0.771	—	079
0.27	0.789	028	080
0.24	0.808	—	081
0.21	0.827	029	082
0.18	0.847	—	083
0.15	0.867	030	084
0.13	0.888	—	085
0.10	0.910	031	086
0.07	0.931	—	087
0.05	0.954	032	088
0.02	0.977	—	089
0.00	1.000	033	090

Note: The exact redshifts of a given snapshot may be slighty different to the above ones. For instance, there are small differences between the exact redshifts of the snapshots in the Astrid and SIMBA suites. In the simulations of the IllustrisTNG suite these numbers can also be slightly different, since AREPO can only write snapshots in the highest time steps in the hierarchy.

Warning: We have moved some snapshots to tape. If you need these please reach out.

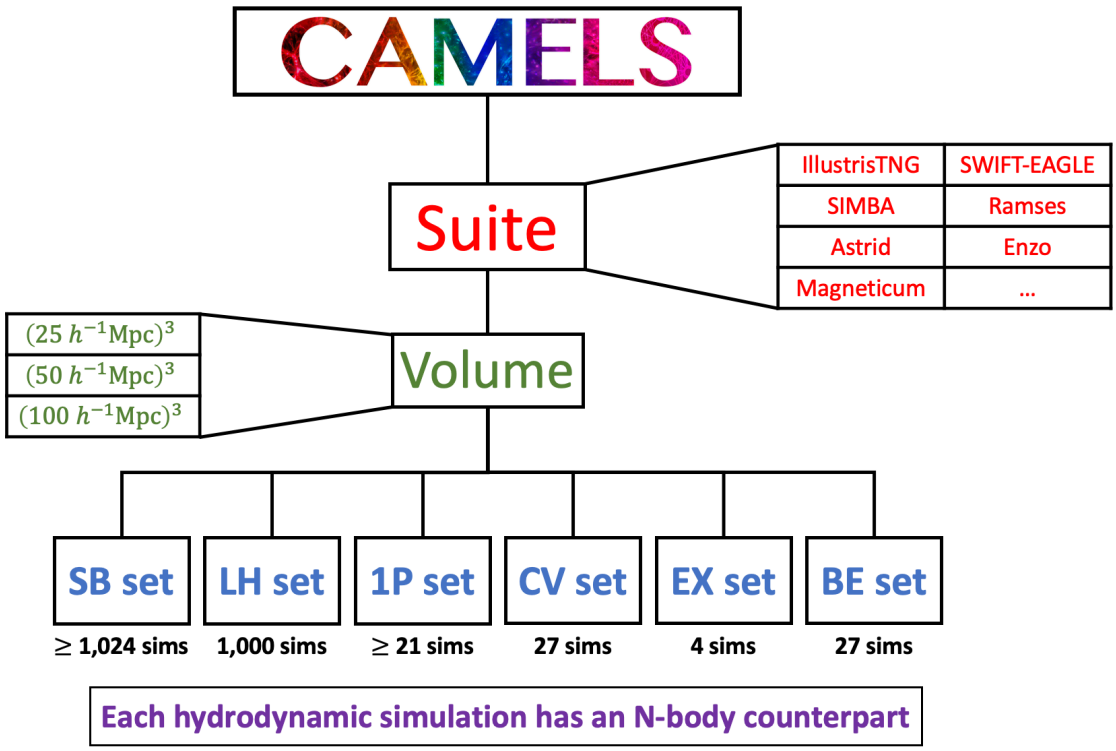
CHAPTER

SEVEN

ORGANIZATION

As of January 2024, CAMELS contains 12,903 cosmological simulations: 5,164 N-body and 7,739 hydrodynamic simulations.

The CAMELS simulations are organized into different suites, volumes, and sets depending on the code used to run them, their volume, and how the values of the cosmological parameters, the astrophysical parameters, and the initial random seeds are arranged. We refer the reader to *General description* for details on the number of simulations available per suite, volume, and set. The following scheme shows the way data is organized:



Knowing this structure is important to know where the data is located and which data to use for different tasks. For instance, the 25 Mpc/h IllustrisTNG simulations of the CV set are located in `/Sims/IllustrisTNG/L25n256/CV`.

7.1 Suites

The CAMELS simulations are organized in different *suites*:

- **IllustrisTNG**. This suite contains all hydrodynamic simulations that have been run with the AREPO code employing the same subgrid physics as the original IllustrisTNG simulations.
- **SIMBA**. This suite contains all hydrodynamic simulations that have been run with the GIZMO code employing the same subgrid physics as the original SIMBA simulation.
- **Astrid**. This suite contains all hydrodynamics simulations that have been run with the MP-Gadget code employing the same subgrid physics as the original Astrid simulation (see Ni et al. 2022 and Bird et al. 2022).
- **Magneticum**. This suite contains all hydrodynamic simulations that have been run with the Open-Gadget code using an updated version of the Magneticum subgrid model.
- **Swift-EAGLE**. This suite contains all hydrodynamic simulations that have been run with the Swift code using the EAGLE subgrid physics model (see Schaye et al. 2015 and Crain et al. 2015).
- **Ramses**. This suite contains all hydrodynamic simulations that have been run with the Ramses code using an state-of-the-art subgrid physics.
- **Enzo**. This suite contains all hydrodynamic simulations that have been run with the Enzo code using an state-of-the-art subgrid physics model.

We refer the reader to *Codes* for more details on the codes and subgrid physics models of the different suites.

Note: For every of the above suites, there is an collection of N-body simulations that represent the dark matter only counterparts of the above hydrodynamic simulations. For instance, **IllustrisTNG_DM** represents the N-body counterpart of the simulations in the **IllustrisTNG** suite.

7.2 Volumes

Each suite contains simulations run at different volumes and number of particles:

- **L25n256**. All simulations follow the evolution of 256^3 dark matter particles plus 256^3 initial gas elements in a periodic comoving volume of $(25\ h^{-1}\mathrm{Mpc})^3$.
- **L50n512**. All simulations follow the evolution of 512^3 dark matter particles plus 512^3 initial gas elements in a periodic comoving volume of $(50\ h^{-1}\mathrm{Mpc})^3$.
- **L100n1024**. All simulations follow the evolution of 1024^3 dark matter particles plus 1024^3 initial gas elements in a periodic comoving volume of $(100\ h^{-1}\mathrm{Mpc})^3$.

Note: As of December 2023, most of the simulations belong to the `L25n256` category. As the CAMELS project matures and expands it will incorporate simulations at larger volumes. If there is no *volume* folder present, then the data should be considered to belong to the `L25n256` category.

7.3 Sets

Each volume of each suite contains various simulation *sets*:

- **SB**. This set contains at least 128 simulations. For instance, the Ramses suite contains 128 simulations while the IllustrisTNG suite contains 2048 simulations. Each simulation has a different value of the cosmological and astrophysical parameters, that are arranged in a Sobol sequence with 2^N elements, where N is an integer number. Besides, each simulation has a different value of the initial random seed. This set will be named as `SBX`, where is the number of dimensions; for the instance, the `SB28` set of the IllustrisTNG suite. SB stands for Sobol sequence.
- **LH**. This set contains 1,000 simulations. Each simulation has a different value of the cosmological and astrophysical parameters, that are arranged in a latin-hypercube. Each simulation has a different value of the random seed used to generate the initial conditions. LH stands for Latin-Hypercube.
- **1P**. This set contains 4 simulations per parameter plus one fiducial. For example, in the IllustrisTNG suite we vary 28 parameters, so there are 113 simulations. For Ramses, where we only vary 5 parameters, we have 21 simulations. In this set, the value of the cosmological and astrophysical parameters in the simulations is varied only one at a time. The value of the random seed used to generate the initial conditions is the same in all simulations. This set is typically used to study the change induced by cosmology and astrophysics in a given quantity. 1P stands for 1-parameter at-a-time.
- **CV**. This set contains 27 simulations. All the simulations share the value of the cosmological and astrophysical parameters (that are set to their fiducial values), and they only differ in the value of their initial conditions random seed. This set is typically used to study the effect of cosmic variance. CV stands for Cosmic Variance.
- **EX**. This set contains 4 simulations. All simulations share the value of the cosmological parameters, but differ in the value of the astrophysical parameters. One simulation has fiducial values; the other three represent extreme cases with 1) very efficient AGN feedback, 2) very efficient supernova feedback, and 3) no feedback. All simulations share the value of the initial conditions random seed. This set can be used to study the maximum effect astrophysics can have on a given quantity. EX stands for Extreme.
- **BE**. This set contains 27 simulations and is currently available only for the IllustrisTNG suite. All of these simulations share the exact same initial conditions with the 1P set and all are run with the fiducial model, but they use different random number sequences for the evolution of the simulation (not to be confused with the random seed that is used to generate the initial conditions). Hence, the differences between them represent the intrinsic randomness of the simulation results, which can serve as a benchmark for the performance of various predictive models. BE stands for Butterfly Effect.

We refer the reader to *Parameters* for further details on the value and meaning of the varied parameters in the different sets.

Note: The SB and LH sets are very simular. The main difference is that SB uses a Sobol sequence to sample the parameter space while LH uses a Latin-hypercube. Given the fact that Sobol sequences have better properties to sample the parameter space than latin-hypercubes (e.g. it is very easy to expand the number of simulations, they sample the parameter space more uniformingly…etc), all new CAMELS simulations will use Sobol sequences instead of Latin-hypercubes. We keep the LH set for historical reasons, and while they are very useful for many things we encourage users to use SB sets (if available) instead of LH sets.

CHAPTER

EIGHT

CODES

As discussed in *Organization*, the CAMELS simulations can be classified into different suites depending on the code and subgrid model used to run them. Below we provide some details on the codes, physics, and subgrid models employed.

8.1 IllustrisTNG

The simulations in the IllustrisTNG suite have been run with the Arepo code using the same subgrid physics as the IllustrisTNG simulation. Arepo uses TreePM to solve for gravity and a moving Voronoi mesh to solve for ideal magnetohydrodynamics (MHD). The IllustrisTNG galaxy formation physics implementation includes sub-grid models for star-formation, stellar evolution and galactic winds, as well as supermassive black hole (SMBH) seeding, merging, accretion and feedback. The latter operates in two modes, selected based on SMBH mass and Eddington ratio, where the high-accretion mode is thermal and the low accretion mode is kinetic and is the more efficient one in ejecting gas and quenching massive galaxies. The galactic winds feedback is kinetic, implemented via briefly hydrodynamically decoupled wind particles, with energy and mass loading factors that are prescribed based on local velocity dispersion and metallicity. Much more detail can be found on the IllustrisTNG project website.

The video below shows an example of a CAMELS-IllustrisTNG simulation:

8.2 SIMBA

The simulations in the SIMBA suite have been run with the GIZMO code using the same subgrid physics as the original SIMBA simulation, presented in Davé et al. 2019. SIMBA builds on the MUFASA model (Dave et al. 2016) and the gravitational torque accretion and kinetic AGN feedback implementation of Anglés-Alcázar et al. 2017a. SIMBA uses the "Meshless Finite Mass" hydrodynamics mode of GIZMO (Hopkins 2015) and computes gravitational forces using a modified version of the TreePM algorithm of Springel 2005, including adaptive gravitational softenings. Radiative cooling and photoionization are implemented using Grackle-3.1 (Smith et al. 2017), stars form from molecular gas following Krumholz & Gnedin (2011), and dust formation and evolution is modelled on-the-fly (Li et al. 2019). Stellar feedback includes two-phase galactic winds with their mass loading factor and velocity taken from the FIRE simulations (Muratov et al. 2015; Angles-Alcazar et al. 2017b). SMBHs grow through cold gas accretion driven by gravitational torques (Hopkins & Quataert 2011; Angles-Alcazar et al. 2017a) and hot gas accretion following Bondi 1952. Kinetic AGN feedback follows a two-mode approach with quasar winds transitioning into high-speed collimated jets at low Eddington accretion rates, and X-ray radiative feedback follows Choi et al. (2012).

The video below shows an example of a CAMELS-SIMBA simulation:

8.3 Astrid

The simulations in the ASTRID suite have been run with the MP-Gadget code, a massively scalable version of the cosmological structure formation code Gadget-3, to solve the gravity (with TreePM), hydrodynamics (using entropy-conserving formulation of Smoothed Particle Hydrodynamics method). ASTRID models galaxy formation physics including sub-grid models for multi-phase ISM and star-formation (Springel & Hernquist 2003), stellar evolution and metal enrichment (Vogelsberger et al. 2014), galactic winds, as well as supermassive black hole (SMBH) seeding, merging, accretion and feedback (Di Matteo et al. 2005). The details of ASTRID subgrid model are described in Bird et al. 2022 and Ni et al. 2022.

The video below shows an example of a CAMELS-Astrid simulation:

8.4 Magneticum

The simulations in the Magneticum suite have been run with the parallel cosmological Tree-PM code OpenGadget3. The code uses an entropy-conserving formulation of Smoothed Particle Hydrodynamics (SPH) (Springel & Hernquist 2002), with SPH modifications according to Dolag et al. (2004), Dolag et al. (2005), and Dolag et al. (2006). It includes also prescriptions for multiphase interstellar medium based on the model by Springel & Hernquist (2003) as well as Tornatore et al. (2007) for the metal enrichment prescription. The model follows the growth and evolution of BHs and their associated AGN feedback based on the model presented by Springel et al. (2005) and Di Matteo et al. (2005), but includes modifications based on Fabjan et al. (2011), Hirschmann et al. (2014b), and Steinborn et al. (2016).

The video below shows an example of a CAMELS-Magneticum simulation:

8.5 Swift-EAGLE

The simulations in the Swift-EAGLE suite have been run with the smoothed particle hydrodynamics and gravity code Swift. Swift is a parallel, open-source, versatile and modular code, with a range of hydrodynamics solvers, gravity solvers, and sub-grid models for galaxy formation (see Swift website). In this suite we use the SPHENIX flavour of SPH, coupled with a modified version of the Evolution and Assembly of GaLaxies and their Environments (EAGLE) subgrid model for galaxy formation and evolution (see Schaye et al. 2015 and Crain et al. 2015). This includes element-by-element radiative cooling and heating rates from Ploeckinger & Schaye 2020, stellar evolution and enrichment from Wiersma et al. 2009, and single thermal-mode feedback from massive stars and accreting AGN (see Schaye & Dalla Vecchia, Booth & Schaye 2009, Rosas-Guevara et al. 2015).

8.6 Ramses

The simulations in the RAMSES suite have been run with the RAMSES code using the same subgrid physics as in Kretschmer & Teyssier (2021) and Teyssier et al. (2011). RAMSES uses Adaptive Particle Mesh to solve for gravity and the Godunov Finite Volume Constrained Transport method to solve for ideal magnetohydrodynamics (MHD). The galaxy formation physics implementation includes a multi-freefall sub-grid model for star-formation and supernovae momentum feedback as in Kretschmer and Teyssier (2021), as well as supermassive black hole (SMBH) seeding, merging, accretion and feedback as in Teyssier et al. (2011) and Pellissier et al. (2023). RAMSES also models metallicity dependent radiative cooling, as well as radiation heating from a self-shielded UV background consistent with standard reionization models.

The video below shows an example of a CAMELS-Ramses simulation:

8.7 Enzo

The simulations in the Enzo suite have been run with the Enzo code.

8.8 N-body

All the N-body simulations hve been run with the TreePM code Gadget-III code. The number of voxels in the PM grid is typically set to be 8 times that of the number of particles. The gravitational softening is set to $1/40$ of the mean inter-particle distance.

The video below shows an example of a CAMELS-Nbody simulation:

CHAPTER

NINE

PARAMETERS

As discussed in *Organization*, the CAMELS simulations can be classified into different sets, depending on how their parameters (cosmological, astrophysical, and initial random seed) are organized. Here we provide details about this.

Suite	Set CV	1P	EX	LH	BE	SB
IllustrisTNG	params fiducial	params info	params info	params info	params fiducial	params info
SIMBA	params fiducial	params info	params info	params info	params fiducial	
Astrid	params fiducial	params info	params info	params info		params info
Magneticum	params fiducial			params		
Swift-EAGLE	params fiducial	params info		params info		
Ramses	params fiducial	params info				params info
Enzo						

9.1 Cosmological parameters

All simulations share the value of these cosmological parameters:

w	M_ν	Ω_k
-1	0.0 eV	0.0

For the other cosmological parameters, the different sets vary them differently:

	$\Omega_{\rm m}$	σ_8	$\Omega_{\rm b}$	h	n_s
CV	0.3	0.8	0.049	0.6711	0.9624
BE	0.3	0.8	0.049	0.6711	0.9624
EX	0.3	0.8	0.049	0.6711	0.9624
LH	0.1 - 0.5	0.6 - 1.0	0.049	0.6711	0.9624
1P	0.1 - 0.5	0.6 - 1.0	0.029 - 0.069	0.4711 - 0.8711	0.7624 - 1.1624
SB	0.1 - 0.5	0.6 - 1.0	0.029 - 0.069	0.4711 - 0.8711	0.7624 - 1.1624

Attention: In the case of the Astrid SB7 set, $\Omega_{\rm b}$ is varied from 0.01 to 0.09.

9.2 Astrophysical parameters

We emphasize that every subgrid physics model is different, and the parameters of one model does not mean anything in another one. Thus, we will describe these parameters for each suite and what is varied.

9.2.1 IllustrisTNG

The IllustrisTNG suite contains all sets: 1P, CV, LH, EX, BE, and SB. This table shows which parameters are varied in each set:

	Astrophysical parameters
CV	*fiducial* IllustrisTNG
BE	*fiducial* IllustrisTNG
LH	*standard* 4 astrophysical parameters
EX	*standard* 4 astrophysical parameters
1P	*extended* 23 astrophysical parameters
SB28	*extended* 23 astrophysical parameters

The meaning and range of variation of the 4 *standard* IllustrisTNG parameters are these:

- $A_{\rm SN1}$: it represents the energy per unit SFR of the galactic winds. It can vary from 0.25 to 4. Fiducial value is 1.
- $A_{\rm SN2}$: it represents the wind speed of the galactic winds. It can vary from 0.5 to 2. Fiducial value is 1.
- $A_{\rm AGN1}$: it represents the energy per unit blach-hole accretion rate. It can vary from 0.25 to 4. Fiducial value is 1.
- $A_{\rm AGN2}$: it represents the ejection speed/burstiness of the kinetic mode of the black-hole feedback. It can vary from 0.5 to 2. Fiducial value is 1.

Note: A value of 1 in the astrophysical parameters $A_{\rm SN1}$, $A_{\rm SN2}$, $A_{\rm AGN1}$, $A_{\rm AGN2}$, represents the value chosen in the original flagship simulation of each suite. For instance, the original IllustrisTNG simulation has $A_{\rm SN1}=1$, $A_{\rm SN2}=1$, $A_{\rm AGN1}=1$, $A_{\rm AGN2}=1$.

The *extended* 23 IllustrisTNG parameters represent an almost-complete set of the IllustrisTNG model, i.e. it contains almost all parameters in the subgrid physics model. The range of the parameters was chosen to cover a very wide range of effects. For SB28, the user can find the list of parameters, their range, and meaning here.

9.2.2 SIMBA

The SIMBA suite contains 4 different sets: 1P, CV, LH, and EX. This table shows which parameters are varied in each set:

	Astrophysical parameters
CV	*fiducial* SIMBA
BE	*fiducial* SIMBA
LH	*standard* 4 astrophysical parameters
EX	*standard* 4 astrophysical parameters
1P	*extended* 23 astrophysical parameters

The meaning and range of variation of the 4 *standard* SIMBA parameters are these:

- $A_{\rm SN1}$: it represents the mass loading of the galactic winds. It can vary from 0.25 to 4. Fiducial value is 1.
- $A_{\rm SN2}$: it represents the wind speed of the galactic winds. It can vary from 0.5 to 2. Fiducial value is 1.
- $A_{\rm AGN1}$: it represents the momentum flux of the QSO & jet-mode black-hole feedback. It can vary from 0.25 to 4. Fiducial value is 1.
- $A_{\rm AGN2}$: it represents the jet speed of the jet-mode black-hole feedback. It can vary from 0.5 to 2. Fiducial value is 1.

Note: A value of 1 in the astrophysical parameters $A_{\rm SN1}$, $A_{\rm SN2}$, $A_{\rm AGN1}$, $A_{\rm AGN2}$, represents the value chosen in the original flagship simulation of each suite. For instance, the original SIMBA simulation has $A_{\rm SN1}=1$, $A_{\rm SN2}=1$, $A_{\rm AGN1}=1$, $A_{\rm AGN2}=1$.

Important: While we call these parameters in the same way as the ones of IllustrisTNG, we emphasize that they are completely independent of each other. For instance, a neural network trained to predict $A_{\rm SN1}$ from IllustrisTNG simulation should fail if tested on SIMBA.

9.2.3 Astrid

The Astrid suite contains 5 different sets: 1P, CV, LH, EX, and SB. This table shows which parameters are varied in each set:

	Astrophysical parameters
CV	*fiducial* Astrid
LH	*standard* 4 astrophysical parameters
EX	*standard* 4 astrophysical parameters
1P	*standard* 4 astrophysical parameters
SB7	*standard* 4 astrophysical parameters

The meaning and range of variation of the 4 *standard* Astrid parameters are these:

- $A_{\rm SN1}$: it represents the energy per SFR of the galactic winds. It can vary from 0.25 to 4. Fiducial value is 1.
- $A_{\rm SN2}$: it represents the wind speed of the galactic winds. It can vary from 0.5 to 2. Fiducial value is 1.

- $A_{\rm AGN1}$: it represents the energy per black-hole accretion rate of the kinetic black-hole feedback. It can vary from 0.25 to 4. Fiducial value is 1.
- $A_{\rm AGN2}$: it represents the energy per unit black-hole accretion rate of the thermal model of the black-hole feedback. It can vary from 0.25 to 4. Fiducial value is 1.

Note: A value of 1 in the astrophysical parameters $A_{\rm SN1}$, $A_{\rm SN2}$, $A_{\rm AGN1}$, $A_{\rm AGN2}$, represents the value chosen in the original flagship simulation of each suite. For instance, the original Astrid simulation has $A_{\rm SN1} = 1$, $A_{\rm SN2} = 1$, $A_{\rm AGN1} = 1$, $A_{\rm AGN2} = 1$.

Note: The SB7 suite of Astrid varies $\Omega_{\rm m}$, σ_8, the above four standard astrophysical parameters and $\Omega_{\rm b}$, that varies from 0.01 to 0.09.

Important: While we call these parameters in the same way as the ones of IllustrisTNG and SIMBA, we emphasize that they are completely independent of each other. For instance, a neural network trained to predict $A_{\rm SN1}$ from IllustrisTNG simulation should fail if tested on Astrid.

9.2.4 Magneticum

This table shows which parameters are varied in each set:

	Astrophysical parameters
CV	*fiducial* Magneticum
LH	*standard* 4 astrophysical parameters

The meaning and range of variation of the 4 *standard* Magneticum parameters are these:

- $A_{\rm SN1}$ represents the energy per unit of SFR of the galactic winds. It can vary from 0.25 to 4.
- $A_{\rm SN2}$ represents the wind speed of the galactic winds. It can vary from 0.5 to 2.
- $A_{\rm AGN1}$ represents the rate of energy injection from AGN into the ISM,. It can vary from 0.25 to 4.
- $A_{\rm AGN2}$ represents the threshold for switching to radio mode. It can vary from 0.5 to 2.

Attention: What we call here *fiducial* Magneticum does not correspond exactly with the original Magneticum simulation, but with its updated model. See *Codes* for more details.

9.2.5 Swift-EAGLE

The Swift-EAGLE suite contains 3 different sets: CV, 1P, and LH. This table shows which parameters are varied in each set:

	Astrophysical parameters
CV	*fiducial* Ramses
1P	*standard* 4 astrophysical parameters
LH	*standard* 4 astrophysical parameters

The meaning and range of variation of the 4 *standard* EAGLE parameters are these:

- $A_{\rm SN1}$ represents the thermal energy injected in each SNII event. It can vary from 0.25 to 4.
- $A_{\rm SN2}$ represents the metallicity dependence of the stellar feedback fraction per unit stellar mass. It can vary from 0.5 to 2.
- $A_{\rm AGN1}$ represents the scaling of the black hole Bondi accretion rate. It can vary from 0.25 to 4.
- $A_{\rm AGN2}$ represents the temperature jump of gas particles in AGN feedback events. It can vary from 0.5 to 2.

9.2.6 Ramses

The Ramses suite contains 3 different sets: CV, 1P, and SB. This table shows which parameters are varied in each set:

	Astrophysical parameters
CV	*fiducial* Ramses
1P	*standard* 4 astrophysical parameters
SB5	*standard* 4 astrophysical parameters

The meaning and range of variation of the 4 *standard* Ramses parameters are these:

- $A_{\rm SN1}$: this parameter controls the amplitude of the supernovae mechanical energy. It can vary from 0.1 to 10. Fiducial value is 1.
- $A_{\rm SN2}$: this parameter controls the amplitude of the star-formation efficiency of the Ramses multi-free-fall subgrid model. It can vary from 0.05 to 5. Fiducial value is 0.5.
- $A_{\rm AGN1}$: this parameter represents the size of the accretion and feedback region around the sink particles (representing SMBH in Ramses). Sizes are in units of the cell size (usually held quasi-constant in physical scale). It can vary from 2 to 8. Fiducial value is 4.
- $A_{\rm AGN2}$: this parameter represents the gravitational softening of the sink particles (representing SMBH in Ramses). Sizes are in units of the cell size (usually held quasi-constant in physical scale). It can vary from 1 to 4. Fiducial value is 2.

Important: The value of $A_{\rm AGN2}$ in Ramses is set to $A_{\rm AGN1}/2$ in all Ramses simulations. Thus, in SB5 there are only two free cosmological parameters ($\Omega_{\rm m}$ and σ_8) and three free astrophysical parameters ($A_{\rm SN1}$, $A_{\rm SN2}$, and $A_{\rm AGN1}$).

Note: There will not be a LH set of Ramses and only Sobol sequences.

9.2.7 Enzo

Note: We remind the user that for each hydrodynamic simulation there is an N-body counterpart with the same value of the cosmological parameters and of the initial random seed. Thus, the value of the cosmological parameters and of the initial random seed for the N-body simulations can be found in the above links. For instance, for the N-body simulation `Astrid_DM/LH/LH_345` the value of $\Omega_{\rm m}$, σ_8, and the initial random seed is 0.4714, 0.689, and 10350, respectively (the same as the simulation `Astrid/LH/LH_345`).

CHAPTER

TEN

DATA ORGANIZATION

CAMELS data is organized in a hierarchical way.

10.1 Type folders

At the highest level, CAMELS contains different *type folders*:

- `Sims`. This folder contains the raw output from the simulations.
- `FOF_Subfind`. This folder contains the Subfind halo and subhalo catalogues.
- `Rockstar`. This folder contains the Rockstar halo and subhalo catalogues.
- `Caesar`. This folder contains the CAESAR halo and subhalo catalogues.
- `AHF`. This folder contains the AMIGA Halo Finder (AHF) halo and subhalo catalogues.
- `Pk`. This folder contains the power spectra measurements.
- `Bk`. This folder contains the bispectra measurements.
- `PDF`. This folder contains the PDF measurements.
- `VIDE_Voids`. This folder contains the void catalogues.
- `Lya`. This folder contains the Lyman-α spectra.
- `X-rays`. This folder contains the X-rays files.
- `Profiles`. This folder contains the halo radial profiles.
- `CMD`. This folder contains the CAMELS Multifield Dataset (CMD).
- `SCSAM`. This folder contains the data from CAMELS-SAM.

When possible, we have tried to organize the data inside the *type folders* in a self-similar way.

10.2 Suite folders

Inside the considered *type folder*, there are different *suite folders*:

- `IllustrisTNG`. This folder contains the data generated from the simulations of the IllustrisTNG suite
- `IllustrisTNG_DM`. This folder contains the data generated from the N-body counterparts of the simulations in the IllustrisTNG suite.
- `SIMBA`. This folder contains the data generated from the simulations of the SIMBA suite

- `SIMBA_DM`. This folder contains the data generated from the N-body counterparts of the simulations in the SIMBA suite.
- `Astrid`. This folder contains the data generated from the simulations of the Astrid suite.
- `Astrid_DM`. This folder contains the data generated from the N-body counterparts of the simulations in the Astrid suite.
- `Magneticum`. This folder contains the data generated from the simulations of the Magneticum suite.
- `Magneticum_DM`. This folder contains the data generated from the N-body counterparts of the simulations in the Magneticum suite.
- `EAGLE`. This folder contains the data generated from the simulations of the Swift-EAGLE suite.
- `EAGLE_DM`. This folder contains the data generated from the N-body counterparts of the simulations in the Swift-EAGLE suite.
- `Ramses`. This folder contains the data generated from the simulations of the Ramses suite.
- `Ramses_DM`. This folder contains the data generated from the N-body counterparts of the simulations in the Ramses suite.
- `Enzo`. This folder contains the data generated from the simulations of the Enzo suite.
- `Enzo_DM`. This folder contains the data generated from the N-body counterparts of the simulations in the Enzo suite.

Note: For some data products some *suite folders* may be missing. For instance, Lyman-α spectra are not generated from N-body simulations. Thus, in that case, only the IllustrisTNG and SIMBA *suite folders* are present.

10.3 Volume folders

Inside a *suite folder* there three different *volume folders*:

- `L25n256`. This folder contains the data from the simulations with 256^3 dark matter particles (plus 256^3 initial gas elements if hydrodynamic) in a periodic box of $25\ h^{-1}\mathrm{Mpc}$ side.
- `L50n512`. This folder contains the data from the simulations with 512^3 dark matter particles (plus 512^3 initial gas elements if hydrodynamic) in a periodic box of $50\ h^{-1}\mathrm{Mpc}$ side.
- `L100n1024`. This folder contains the data from the simulations with 1024^3 dark matter particles (plus 1024^3 initial gas elements if hydrodynamic) in a periodic box of $100\ h^{-1}\mathrm{Mpc}$ side.

Note: Note that not all simulations, and its associated data, are available. For instance, as of April 2024, for most of the suites, `L25n256` is pretty complete, while `L50n512` only contains IllustrisTNG simulations and there are no simulations in the `L100n1024` volume for any suite.

10.4 Set folders

Inside a *volume folder* there are several *set folders*:

- `SB`. This folder contains the data from the simulations of the SB set. Typically, we will add a number of this set to identify the number of parameters varied. For instance, `SB28` in the case of IllustrisTNG to denote that 28 parameters are varied in a sobol sequence. Inside this folder, there are subfolders named `SBN_X`, where `N` is the number of parameters varied (e.g. 28 in the case of IllustrisTNG), and `XX` ranges from `0` to a multiple of 2 (e.g. 2048 for IllustrisTNG-SB28, 256 for Ramses-SB5...etc).
- `LH`. This folder contains the data from the simulations of the LH set. Inside this folder, there are subfolders named `LH_X`, where where `X` ranges from `0` to `999`.
- `1P`. This folder contains the data from the simulations of the 1P set. Inside this folder, there are subfolders named `1P_pX_`, where `X` ranges from `1` to `N` while `Y` goes from `n2` (-2) to `2`. Where `N` is the number of parameters (e.g. 28 in IllustrisTNG, 6 in Astrid...etc).
- `CV`. This folder contains the data from the simulations of the CV set. Inside this folder, there are subfolders named `CV_X`, where `X` ranges from `0` to `26`.
- `EX`. This folder contains the data from the simulations of the EX set. Inside this folder, there are subfolders named `EX_X`, where `X` ranges from `0` to `3`.
- `BE`. This folder contains the data from the simulations of the BE set. Inside this folder, there are subfolders named `BE_X`, where `X` ranges from `0` to `26`.

Note: The IllustrisTNG suite folder contains a new set of zoom-in simulations under the simulation folder `zoom`. See *CAMELS-zoomGZ* for more details.

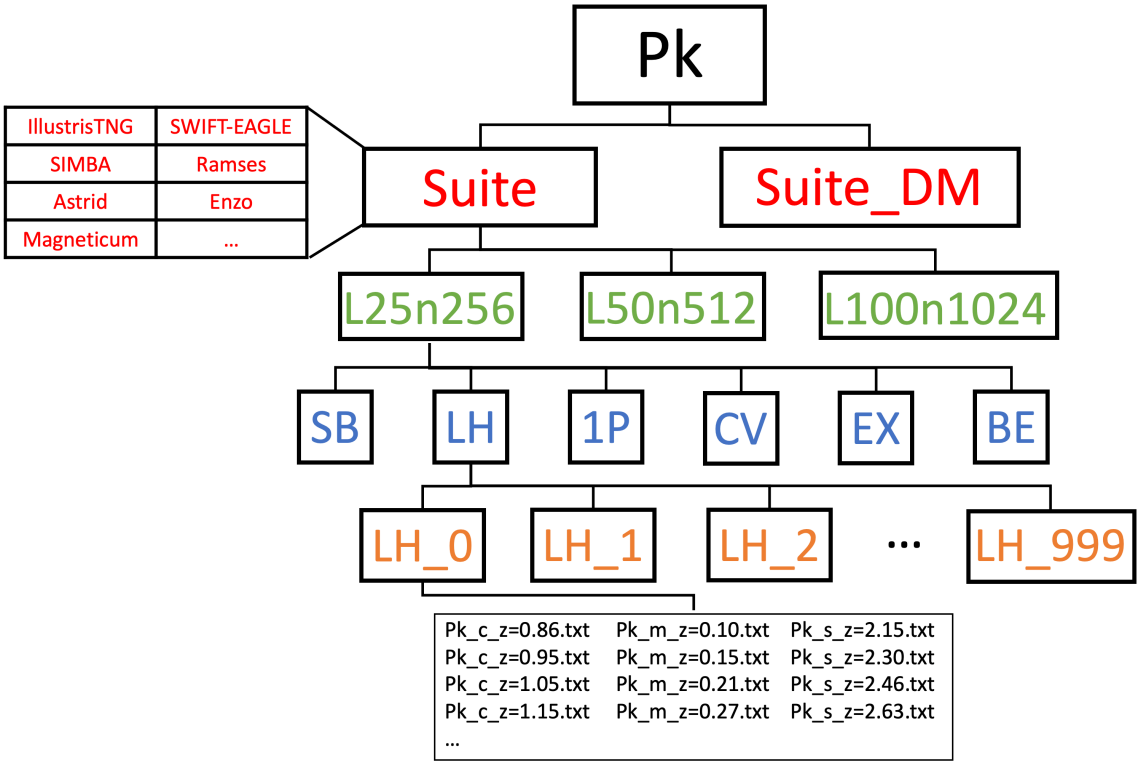
As can be seen, the name of the folder can be used to identify the simulation set and its parameters.

Note: The numeric scheme of the 1P set labels was chosen to help the user to identify which parameter and its variation is the one considered. This may be more useful than just listing the simulations from 0 to, e.g., 65. We refer the reader to *Parameters* for the actual value of the parameters in each simulation.

10.5 Actual data

Finally, inside a *set folder* the user can find the associated data for that particular simulation. We note that these folders can contain multiple files, e.g. the power spectra of the considered simulations at all redshifts.

The image below shows an scheme with the generic data structure for the case of the power spectra:



Warning: There are some data products that are organized in a different way to the one outlined above. For instance, the CAMELS Multifield Dataset (CMD) follows a different data structure. In these cases we describe in detail the structure of those data products.

CHAPTER

ELEVEN

SIMULATIONS

Important: A reorganization of the data has been performed in 2024 in order to enhance its uniformity and simplicity. This will require slight changes to existing codes that access the data.

- Folders in the 1P sets are now named `1P_pX_Y` and each parameter only has 4 variations, rather than 10, such that `X` ranges from n2 to 2.
- Snapshot numbers in the IllustrisTNG and SIMBA suites, where simulations have only 34 snapshots, have been updated to match the numbering in the Astrid suite (and some TNG simulations) that have 91 snapshots. For example, where 33 used to be the z=0 snapshot, now it is 90 uniformly for all suites.
- Snapshot files have been renamed from `snap_###.hdf5` to `snapshot_###.hdf5` and fof/subfind files from `fof_subhalo_tab_###.hdf5` to `groups_###.hdf5`.

The `Sims` folder contains the raw data from all CAMELS simulations. The data is organized following the general hierarchical structure described in *Suite folders*. Thus, the simulations are organized by:

1. the code used to run the simulations: *suite folder*
2. the volume of the simulations: *volume folder*
3. the way simulations parameters are arranged: *set folder*

11.1 Suite folders

The `Sims` folder contains different *suite folders*, that contain all simulations run with a given code:

- `IllustrisTNG`. This folder contains all simulations of the IllustrisTNG suite.
- `SIMBA`. This folder contains all simulations of the SIMBA suite.
- `Astrid`. This folder contains all simulations of the Astrid suite.
- `Magneticum`. This folder contains all simulations of the Magneticum suite.
- `Ramses`. This folder contains all simulations of the Ramses suite.
- `EAGLE`. This folder contains all simulations of the Swift-EAGLE suite.
- `Enzo`. This folder contains all simulations of the Enzo suite.

Each of the above suites have an associated N-body suite. Those simulations are located in a folder with the name of the suite followed by `DM`. For instance, the folder `IllustrisTNG_DM` contains all the N-body simulations that are the counterpart of the full hydrodynamic simulations contained in the `IllustrisTNG` folder.

Note: The value of the cosmological, astrophysical, and initial random seed for simulations in the different suites can be found in *Parameters*. Those files also hold for the N-body simulations; each N-body simulation has the same value of the cosmological parameters and initial random seed as their hydrodynamic counterpart.

11.2 Volume folders

The vast majority of the CAMELS simulations follow the evolution of 256^3 dark matter and 256^3 initial fluid elements in a periodic $(25\ h^{-1}\mathrm{Mpc})^3$ volume. As the CAMELS project evolves and matures, it will contain simulations with larger volumes while keeping the same mass and spatial resolution. As of January 2024, only simulations in the IllustrisTNG suite have simulations at two different volumes: $(25\ \mathrm{Mpc/h})^3$ and $(50\ \mathrm{Mpc/h})^3$.

In general, between the *suite* and the *set folders*, there may be some *volume folders* indicating the volume and number of particles of the simulations. For the foreseable future, only these three folders will exist:

- `L25n256`. This folder contains the simulations that follow the evolution of 256^3 dark matter particles and 256^3 initial fluid elements in a $(25\ \mathrm{Mpc/h})^3$ volume.
- `L50n512`. This folder contains the simulations that follow the evolution of 512^3 dark matter particles and 512^3 initial fluid elements in a $(50\ \mathrm{Mpc/h})^3$ volume.
- `L100n1024`. This folder contains the simulations that follow the evolution of 1024^3 dark matter particles and 1024^3 initial fluid elements in a $(100\ \mathrm{Mpc/h})^3$ volume.

In general, the volume folders will follow the convention `LXnY`, where `X` is the box size in Mpc/h and Y is the cubic root of the number of dark matter particles in the simulations.

Warning: If no volume folder is present in a given simulation suite, it means that all simulations are simulations are standard, i.e. they follow the evolution of 256^3 dark matter particles and 256^3 initial fluid elements in a $(25\ \mathrm{Mpc/h})^3$ volume.

11.3 Set folders

Inside each *suite folder* (or *volume folder*) there are the *set folders* (see *Organization* for details):

- `1P`. This folder constains the simulations of the 1P set. Inside this folder, there are subfolders named `1P_pX_Y` that contain the different simulations in the 1P set. `X` ranges from `1` to `N`, where `N` is the number of parameters while `Y` goes from `n2` (-2) to `2` and denotes the variation of the parameter where 0 is the fiducial value. See *Set folders* for details about the naming of the simulations in the 1P set.
- `CV`. This folder contains the simulation of the CV set. The subfolders in this folder are named `CV_X`, where X goes from 0 to 26.
- `LH`. This folder contains the simulation of the LH set. The subfolders in this folder are named `LH_X` where X goes from 0 to 999.
- `EX`. This folder contains the simulation of the EX set. The subfolders in this folder are named `EX_X` where X goes from 0 to 3.
- `BE`. This folder contains the simulation of the BE set. The subfolders in this folder are named `BE_X`, where X goes from 0 to 26.

- `SB`. This folder contains the simulation of the SB set. In general, this set is named as `SBY`, where Y is the number of dimensions sampled in the Sobol Sequence (e.g. SB28 for IllustrisTNG). The subfolders in this folder are named `SBY_X`, where X goes from 0 to N-1, where N is the number of simulations in the Sobol sequence.
- `zoom`. This folder contains sets of zoom-in simulations. The subfolders correspond to the halo type of zoom-in simulations, e.g. `GZY` representing Group Zoom, and Y the number of parameter space dimensions sampled. The individual zoom-in simulations are in the corresponding subfolders with GZY_X where X goes from 0 to N-1 with N being the number of simulations.
- `CosmoAstroSeed_<suitname>_<volumename>_<setname>.txt`. This file contains the value of the cosmological and astrophysical parameter, together with the value of the random seed, for each simulation in the set. The format of the file is: simulation_name [parameter1 parameter2 ... parameterN] seed.

Besides the above, the *set folders* may also contain some files with the value of the cosmological and astrophysical parameters for the Sobol sequences.

Note: The structure and organization of the N-body simulations (e.g. `IllustrisTNG_DM`) is the same as their full hydrodynamic counterparts.

11.4 Simulation folders

The subfolders inside the *set folders* are *simulations folders*, and they contain the actual simulations:

```
>> ls Sims/IllustrisTNG/L25n256/CV/CV_0
blackhole_details         fof_subhalo_tab_021.hdf5  snap_011.hdf5
blackhole_mergers         fof_subhalo_tab_022.hdf5  snap_012.hdf5
CosmoAstro_params.txt     fof_subhalo_tab_023.hdf5  snap_013.hdf5
extra_files               fof_subhalo_tab_024.hdf5  snap_014.hdf5
fof_subhalo_tab_000.hdf5  fof_subhalo_tab_025.hdf5  snap_015.hdf5
fof_subhalo_tab_001.hdf5  fof_subhalo_tab_026.hdf5  snap_016.hdf5
fof_subhalo_tab_002.hdf5  fof_subhalo_tab_027.hdf5  snap_017.hdf5
fof_subhalo_tab_003.hdf5  fof_subhalo_tab_028.hdf5  snap_018.hdf5
fof_subhalo_tab_004.hdf5  fof_subhalo_tab_029.hdf5  snap_019.hdf5
fof_subhalo_tab_005.hdf5  fof_subhalo_tab_030.hdf5  snap_020.hdf5
fof_subhalo_tab_006.hdf5  fof_subhalo_tab_031.hdf5  snap_021.hdf5
fof_subhalo_tab_007.hdf5  fof_subhalo_tab_032.hdf5  snap_022.hdf5
fof_subhalo_tab_008.hdf5  fof_subhalo_tab_033.hdf5  snap_023.hdf5
fof_subhalo_tab_009.hdf5  ICs                       snap_024.hdf5
fof_subhalo_tab_010.hdf5  snap_000.hdf5             snap_025.hdf5
fof_subhalo_tab_011.hdf5  snap_001.hdf5             snap_026.hdf5
fof_subhalo_tab_012.hdf5  snap_002.hdf5             snap_027.hdf5
fof_subhalo_tab_013.hdf5  snap_003.hdf5             snap_028.hdf5
fof_subhalo_tab_014.hdf5  snap_004.hdf5             snap_029.hdf5
fof_subhalo_tab_015.hdf5  snap_005.hdf5             snap_030.hdf5
fof_subhalo_tab_016.hdf5  snap_006.hdf5             snap_031.hdf5
fof_subhalo_tab_017.hdf5  snap_007.hdf5             snap_032.hdf5
fof_subhalo_tab_018.hdf5  snap_008.hdf5             snap_033.hdf5
fof_subhalo_tab_019.hdf5  snap_009.hdf5
fof_subhalo_tab_020.hdf5  snap_010.hdf5
```

The most relevant ones are these:

- `ICs`. This folder contains the initial conditions of the simulations. See *Initial conditions* for further details.

- `snapshot_0XY.hdf5`. These are the simulation snapshots. Numbers go from 000 (corresponding to $z = 15$) to 090 (corresponding to $z = 0$). See *Redshifts* to know the redshifts associated to the different numbers. These files contain the positions, velocities, IDs and other properties of the dark matter particles and the fluid resolution elements of the simulation. See *Simulations* for details on how to read these files.
- `groups_0XY.hdf5`. These files contain the halo/galaxy catalogues. Numbers go from 000 (corresponding to $z = 15$) to 090 (corresponding to $z = 0$). See *Redshifts* to know the redshifts associated to the different numbers. These files contain the properties of the halos and subhalos identified by SUBFIND. See *SUBFIND catalogs* to see how to read these files.

There are many other files in a simulation folder that we do not describe as they are barely used. Reach out to us if you need help with those.

11.5 Snapshots

CAMELS snapshots are stored as single hdf5 files. In order to read them in python, you will need `h5py`. The simplest way to inspect the content of a snapshot is this:

```
>> h5ls -r Sims/IllustrisTNG/L25n256/CV/CV_14/snapshot_024.hdf5
/                        Group
/Config                  Group
/Header                  Group
/Parameters              Group
/PartType0               Group
/PartType0/Coordinates   Dataset {15879574, 3}
/PartType0/Density       Dataset {15879574}
/PartType0/ElectronAbundance Dataset {15879574}
/PartType0/EnergyDissipation Dataset {15879574}
/PartType0/GFM_AGNRadiation Dataset {15879574}
/PartType0/GFM_CoolingRate Dataset {15879574}
/PartType0/GFM_Metallicity Dataset {15879574}
/PartType0/GFM_Metals    Dataset {15879574, 10}
/PartType0/GFM_MetalsTagged Dataset {15879574, 6}
/PartType0/GFM_WindDMVelDisp Dataset {15879574}
/PartType0/GFM_WindHostHaloMass Dataset {15879574}
/PartType0/InternalEnergy Dataset {15879574}
/PartType0/Machnumber    Dataset {15879574}
/PartType0/MagneticField Dataset {15879574, 3}
/PartType0/MagneticFieldDivergence Dataset {15879574}
/PartType0/Masses        Dataset {15879574}
/PartType0/NeutralHydrogenAbundance Dataset {15879574}
/PartType0/ParticleIDs   Dataset {15879574}
/PartType0/Potential     Dataset {15879574}
/PartType0/StarFormationRate Dataset {15879574}
/PartType0/SubfindDMDensity Dataset {15879574}
/PartType0/SubfindDensity Dataset {15879574}
/PartType0/SubfindHsml   Dataset {15879574}
/PartType0/SubfindVelDisp Dataset {15879574}
/PartType0/Velocities    Dataset {15879574, 3}
/PartType1               Group
/PartType1/Coordinates   Dataset {16777216, 3}
/PartType1/ParticleIDs   Dataset {16777216}
/PartType1/Potential     Dataset {16777216}
```

(continues on next page)

(continued from previous page)

```
/PartType1/SubfindDMDensity Dataset {16777216}
/PartType1/SubfindDensity Dataset {16777216}
/PartType1/SubfindHsml   Dataset {16777216}
/PartType1/SubfindVelDisp Dataset {16777216}
/PartType1/Velocities    Dataset {16777216, 3}
/PartType4               Group
/PartType4/BirthPos      Dataset {524754, 3}
/PartType4/BirthVel      Dataset {524754, 3}
/PartType4/Coordinates   Dataset {524754, 3}
/PartType4/GFM_InitialMass Dataset {524754}
/PartType4/GFM_Metallicity Dataset {524754}
/PartType4/GFM_Metals    Dataset {524754, 10}
/PartType4/GFM_MetalsTagged Dataset {524754, 6}
/PartType4/GFM_StellarFormationTime Dataset {524754}
/PartType4/GFM_StellarPhotometrics Dataset {524754, 8}
/PartType4/Masses        Dataset {524754}
/PartType4/ParticleIDs   Dataset {524754}
/PartType4/Potential     Dataset {524754}
/PartType4/SubfindDMDensity Dataset {524754}
/PartType4/SubfindDensity Dataset {524754}
/PartType4/SubfindHsml   Dataset {524754}
/PartType4/SubfindVelDisp Dataset {524754}
/PartType4/Velocities    Dataset {524754, 3}
/PartType5               Group
/PartType5/BH_BPressure  Dataset {1257}
/PartType5/BH_CumEgyInjection_QM Dataset {1257}
/PartType5/BH_CumEgyInjection_RM Dataset {1257}
/PartType5/BH_CumMassGrowth_QM Dataset {1257}
/PartType5/BH_CumMassGrowth_RM Dataset {1257}
/PartType5/BH_Density    Dataset {1257}
/PartType5/BH_HostHaloMass Dataset {1257}
/PartType5/BH_Hsml       Dataset {1257}
/PartType5/BH_Mass       Dataset {1257}
/PartType5/BH_Mdot       Dataset {1257}
/PartType5/BH_MdotBondi  Dataset {1257}
/PartType5/BH_MdotEddington Dataset {1257}
/PartType5/BH_Pressure   Dataset {1257}
/PartType5/BH_Progs      Dataset {1257}
/PartType5/BH_U          Dataset {1257}
/PartType5/Coordinates   Dataset {1257, 3}
/PartType5/Masses        Dataset {1257}
/PartType5/ParticleIDs   Dataset {1257}
/PartType5/Potential     Dataset {1257}
/PartType5/SubfindDMDensity Dataset {1257}
/PartType5/SubfindDensity Dataset {1257}
/PartType5/SubfindHsml   Dataset {1257}
/PartType5/SubfindVelDisp Dataset {1257}
/PartType5/Velocities    Dataset {1257, 3}
```

As can be seen, the snapshots contain different groups and blocks:

- `Header`. This group contains different properties of the simulations such as its box size, number of particles, value of the cosmological parameters…etc.

- `PartType0`. This group contains the properties of the gas particles.
- `PartType1`. This group contains the properties of the dark matter particles.
- `PartType2`. This group contains low-resolution dark matter particles, only relevant in zoom-in simulations.
- `PartType4`. This group contains the properties of the star particles.
- `PartType5`. This group contains the properties of the black hole particles.

For instance, the block `/PartType4/Coordinates` contains the coordinates of the star particles. A detailed description of the different blocks can be found here.

Note: While the format of the snapshots in the different suites is almost identical, there are a few differences. See *Suite differences* for more information.

Note: The zoom-in simulations contain snapshot directories as opposed to individual files.

Reading the snapshot header and blocks can be done as follows:

```
import numpy as np
import h5py
import hdf5plugin

# snapshot name
snapshot = 'Sims/IllustrisTNG/L25n256/CV/CV_14/snapshot_014.hdf5'

# open file
f = h5py.File(snapshot, 'r')

# read different attributes of the header
BoxSize      = f['Header'].attrs[u'BoxSize']/1e3 #Mpc/h
redshift     = f['Header'].attrs[u'Redshift']
h            = f['Header'].attrs[u'HubbleParam']
Masses       = f['Header'].attrs[u'MassTable']*1e10 #Msun/h
Np           = f['Header'].attrs[u'NumPart_Total']
Omega_m      = f['Header'].attrs[u'Omega0']
Omega_L      = f['Header'].attrs[u'OmegaLambda']
Omega_b      = f['Header'].attrs[u'OmegaBaryon']
scale_factor = f['Header'].attrs[u'Time'] #scale factor

# read gas positions
pos_g = f['PartType0/Coordinates'][:]/1e3  #positions in Mpc/h

# read dark matter velocities; need to multiply by sqrt(a) to get peculiar velocities
vel_c = f['PartType1/Velocities'][:]*np.sqrt(scale_factor) #velocities in km/s

# read star masses
mass_s = f['PartType4/Masses'][:]*1e10  #Masses in Msun/h

# read black hole positions and the gravitational potential at their locations
pos_bh       = f['PartType5/Coordinates'][:]/1e3  #positions in Mpc/h
potential_bh = f['PartType5/Potential'][:]/scale_factor #potential in (km/s)^2
```

(continues on next page)

(continued from previous page)

```
# close file
f.close()
```

Warning: To read the hdf5 files you need to do both `import hdf5` and `import hdf5plugin`. This is because the CAMELS N-body simulations have been compressed in a way that requires an additional library: `hdf5plugin`. We recommend loading that library always as its usage is transparent and will work with both compressed and uncompressed snapshots. If you don't have it already, you can install it with `python -m pip install hdf5plugin`. Note that the hdf5plugin library is already installed on binder.

Note: Note that the N-body simulations only contain the positions, velocities and IDs of the dark matter particles.

11.6 Initial conditions

The initial conditions of all simulations were generated at $z = 127$ using second order lagrangian perturbation theory (2LPT). The same transfer function (total matter) was used for the gas and dark matter components. Particles were initially laid down in a regular grid: one grid for the dark matter particles and another grid, offset by half a grid cell, for the gas.

The initial condition files can be found inside each simulation folder. For instance, to access the initial conditions of the LH_156 simulation of the SIMBA suite:

```
>> ls Sims/SIMBA/L25n256/LH/LH_156/ICs
2LPT.param   ics.1  ics.4  ics.7              Pk_m_z=0.000.txt
CAMB.params  ics.2  ics.5  inputspec_ics.txt
ics.0        ics.3  ics.6  logIC
```

There are different files:

- `2LPT.param`. This is the 2LPT parameter file used to generate the simulation initial conditions.
- `CAMB.params`. This CAMB parameter file used to generate the $z = 0$ matter power spectrum needed to generate the initial conditions.
- `ics.X`. These files contain the positions, velocities, and IDs of the particles in the initial conditions. They can be Gadget Format I (for the hydrodynamic simulations) or hdf5 format (for the N-body simulations). In both cases, the data can be read with Pylians3 as shown below. The hdf5 files can also be read as standard snapshots (see *read_snaps*).
- `inputspec_ics.txt`. A file generated by 2LPT with the input power spectrum. Only needed for debugging.
- `logIC`. This file contains the output generated by 2LPT when generating the initial conditions. One useful for internal debugging.
- `Pk_m_z=0.000.txt`. The linear matter power spectrum at $z = 0$ for the simulation. This file is generated by running the `CAMB` code with the `CAMB.params` parameter file. This file is used in `2LPT.param` to generate the initial conditions.

The files with the initial conditions can be read as follows:

```
import numpy as np
import readgadget

# name of the snapshot
snapshot = '/mnt/ceph/users/camels/Sims/Astrid/L25n256/LH/LH_156/ICs/ics'

# read snapshot header
header   = readgadget.header(snapshot)
BoxSize  = header.boxsize/1e3  #Mpc/h
Nall     = header.nall         #Total number of particles
Masses   = header.massarr*1e10 #Masses of the particles in Msun/h
Omega_m  = header.omega_m      #value of Omega_m
Omega_l  = header.omega_l      #value of Omega_l
h        = header.hubble       #value of h
redshift = header.redshift     #redshift of the snapshot
Hubble   = 100.0*np.sqrt(Omega_m*(1.0+redshift)**3+Omega_l)#Value of H(z) in km/s/(Mpc/h)

# read positions, velocities and IDs of the gas particles
ptype = [0] #gas is particle type 0
pos_g = readgadget.read_block(snapshot, "POS ", ptype)/1e3 #positions in Mpc/h
vel_g = readgadget.read_block(snapshot, "VEL ", ptype)     #peculiar velocities in km/s
ids_g = readgadget.read_block(snapshot, "ID  ", ptype)-1   #IDs starting from 0

# read positions, velocities and IDs of the dark matter particles
ptype = [1] #dark matter is particle type 1
pos_c = readgadget.read_block(snapshot, "POS ", ptype)/1e3 #positions in Mpc/h
vel_c = readgadget.read_block(snapshot, "VEL ", ptype)     #peculiar velocities in km/s
ids_c = readgadget.read_block(snapshot, "ID  ", ptype)-1   #IDs starting from 0
```

Warning: The format of the ICs of the N-body simulations is hdf5 instead of Gadget format I. These files can be read in the same way as above or can be read as hdf5 files; see *read_snaps*. Keep in mind that those files have been compressed, so you need to use load the hdf5plugin library with `import hdf5plugin`.

Note: When using the `readgadget` library, the particle velocities automatically incorporate the $\sqrt{a}$ Gadget factor.

Note: When reading initial conditions of N-body simulations, only positions, velocities, and IDs for dark matter particles are present, not for gas.

11.7 Suite differences

The simulations from the different suites are very different: they solve the hydrodynamic equations using completely different methods and the subgrid models employed are distinct. However, the format of the data is similar among the sets. The main differences are these:

- The format of the metallicity array is slightly different. In SIMBA, `Metallicity` is an 11-element array where the n=0 component is the *total* metal mass fraction (everything not H, He), and the remaining elements contain the mass fraction in [He,C,N,O,Ne,Mg,Si,S,Ca,Fe].
- Particle positions are saved in single precision in SIMBA, while in IllustrisTNG are stored in double precision.
- The SIMBA simulations track `Dust_Masses` and `Dust_Metallicity` (that are not available in IllustrisTNG), while IllustrisTNG simulations contain magnetic fields (not available in SIMBA).
- In the SIMBA simulations the masses of the dark matter particles are listed individually in `PartType1/Masses`. In the IllustrisTNG simulations the dark matter particle mass is only stored in the header.
- The hydrodynamics methods are different and so the sizes (and shapes) that gas elements represent are different in IllustrisTNG and SIMBA.

11.8 Compression

The snapshots of the CAMELS simulations are compressed to best utilize the available resources. The data is compressed using two different schemes: lossless and lossy. Currently, the snapshots of the hydrodynamic simulations are compressed using a lossless scheme, whereas the snapshots of the N-body simulations are compressed using a lossy method (see table below). We do this because N-body simulations are much faster to (re)run and also because this compression scheme has been well tested with other N-body simulations like Abacus and Quijote.

Simulation type		Compression type
N-body	Snapshots	lossy
	ICs	lossless
Hydrodynamic	Snapshots	lossless
	ICs	none

Lossy compression

This compression allows us to shrink the size of the files by a factor of ~ 2.5. The details of the lossy compression are the following. The snapshots are compressed with a Blosc filter, as implemented in the hdf5plugin Python package. Blosc compression applies a transpose to the data then passes it to zstandard, all of which is lossless and transparent to the user. As a preconditioning step to increase the Blosc compression ratio, we manually null out some bits of the positions and velocities to increase the compression ratio. This step is lossy.

In more detail, the positions are stored as absolute coordinates in float32 precision. The lossy preconditioning we apply is to set several of the low bits in the float32 significand to zero. The number of bits nulled out is B=6 for the 1024^3 simulations, B=7 for 512^3, and B=8 for 256^3. This introduces a fractional error of $2^{(-24+B)}$, which is 1.5×10^{-5} for simulations with 256^3 particles. Thus, for traditional CAMELS boxes of 25 Mpc/h size, the worst-case is translated into an error of 0.38 kpc/h, smaller than the softening length of these simulations. Thus, this should have minimal impact on science projects. Likewise, we null out 11 low bits of the velocities, for a fractional precision of 0.01%. The velocity rarely goes above 6000 km/s in LCDM N-body simulations, so this is a worst case of 0.6 km/s precision. The particle IDs are compressed in a lossless manner.

The HDF5 compressed in this way contains a new group called `/CompressionInfo` whose attributes contain a JSON string describing the exact compression options used. The scripts used to do the compression are here: https://github.

com/lgarrison/quijote-compression. We thank Lehman Garrison for setting this up.

CHAPTER

TWELVE

SUBFIND CATALOGS

Important: A reorganization of the data has been performed in 2024 in order to enhance its uniformity and simplicity. This will require slight changes to existing codes that access the data.

- Folders in the 1P sets are now named `1P_pX_Y` and each parameter only has 4 variations, rather than 10, such that `X` ranges from n2 to 2.
- Snapshot numbers in the IllustrisTNG and SIMBA suites, where simulations have only 34 snapshots, have been updated to match the numbering in the Astrid suite (and some TNG simulations) that have 91 snapshots. For example, where 33 used to be the z=0 snapshot, now it is 90 uniformly for all suites.
- Snapshot files have been renamed from `snap_###.hdf5` to `snapshot_###.hdf5` and fof/subfind files from `fof_subhalo_tab_###.hdf5` to `groups_###.hdf5`.

The `FOF_Subfind` folder contains the FOF+SUBFIND halo/subhalo/galaxy catalogs. The data is organized following the general hierarchical structure described in *Suite folders*.

The catalogs are stored as hdf5 files. `h5py` is needed in order to read these files using python. CAMELS provides a halo/subhalo catalog for each snapshot. The halos are identified through FOF and subhalos are identified through SUBFIND.

The easiest way to inspect the content of these files is:

```
>> h5ls -r SIMBA/CV_5/groups_031.hdf5
/                        Group
/Config                  Group
/Group                   Group
/Group/GroupBHMass       Dataset {32272}
/Group/GroupBHMdot       Dataset {32272}
/Group/GroupCM           Dataset {32272, 3}
/Group/GroupFirstSub     Dataset {32272}
/Group/GroupGasMetalFractions Dataset {32272, 11}
/Group/GroupGasMetallicity Dataset {32272}
/Group/GroupLen          Dataset {32272}
/Group/GroupLenType      Dataset {32272, 6}
/Group/GroupMass         Dataset {32272}
/Group/GroupMassType     Dataset {32272, 6}
/Group/GroupNsubs        Dataset {32272}
/Group/GroupPos          Dataset {32272, 3}
/Group/GroupSFR          Dataset {32272}
/Group/GroupStarMetalFractions Dataset {32272, 11}
/Group/GroupStarMetallicity Dataset {32272}
```

(continues on next page)

(continued from previous page)

```
/Group/GroupVel          Dataset {32272, 3}
/Group/GroupWindMass     Dataset {32272}
/Group/Group_M_Crit200   Dataset {32272}
/Group/Group_M_Crit500   Dataset {32272}
/Group/Group_M_Mean200   Dataset {32272}
/Group/Group_M_TopHat200 Dataset {32272}
/Group/Group_R_Crit200   Dataset {32272}
/Group/Group_R_Crit500   Dataset {32272}
/Group/Group_R_Mean200   Dataset {32272}
/Group/Group_R_TopHat200 Dataset {32272}
/Header                  Group
/IDs                     Group
/IDs/ID                  Dataset {14575639}
/Parameters              Group
/Subhalo                 Group
/Subhalo/SubhaloBHMass   Dataset {22315}
/Subhalo/SubhaloBHMdot   Dataset {22315}
/Subhalo/SubhaloBfldDisk Dataset {22315}
/Subhalo/SubhaloBfldHalo Dataset {22315}
/Subhalo/SubhaloCM       Dataset {22315, 3}
/Subhalo/SubhaloGasMetalFractions Dataset {22315, 11}
/Subhalo/SubhaloGasMetalFractionsHalfRad Dataset {22315, 11}
/Subhalo/SubhaloGasMetalFractionsMaxRad Dataset {22315, 11}
/Subhalo/SubhaloGasMetalFractionsSfr Dataset {22315, 11}
/Subhalo/SubhaloGasMetalFractionsSfrWeighted Dataset {22315, 11}
/Subhalo/SubhaloGasMetallicity Dataset {22315}
/Subhalo/SubhaloGasMetallicityHalfRad Dataset {22315}
/Subhalo/SubhaloGasMetallicityMaxRad Dataset {22315}
/Subhalo/SubhaloGasMetallicitySfr Dataset {22315}
/Subhalo/SubhaloGasMetallicitySfrWeighted Dataset {22315}
/Subhalo/SubhaloGrNr     Dataset {22315}
/Subhalo/SubhaloHalfmassRad Dataset {22315}
/Subhalo/SubhaloHalfmassRadType Dataset {22315, 6}
/Subhalo/SubhaloIDMostbound Dataset {22315}
/Subhalo/SubhaloLen      Dataset {22315}
/Subhalo/SubhaloLenType  Dataset {22315, 6}
/Subhalo/SubhaloMass     Dataset {22315}
/Subhalo/SubhaloMassInHalfRad Dataset {22315}
/Subhalo/SubhaloMassInHalfRadType Dataset {22315, 6}
/Subhalo/SubhaloMassInMaxRad Dataset {22315}
/Subhalo/SubhaloMassInMaxRadType Dataset {22315, 6}
/Subhalo/SubhaloMassInRad Dataset {22315}
/Subhalo/SubhaloMassInRadType Dataset {22315, 6}
/Subhalo/SubhaloMassType Dataset {22315, 6}
/Subhalo/SubhaloParent   Dataset {22315}
/Subhalo/SubhaloPos      Dataset {22315, 3}
/Subhalo/SubhaloSFR      Dataset {22315}
/Subhalo/SubhaloSFRinHalfRad Dataset {22315}
/Subhalo/SubhaloSFRinMaxRad Dataset {22315}
/Subhalo/SubhaloSFRinRad Dataset {22315}
/Subhalo/SubhaloSpin     Dataset {22315, 3}
/Subhalo/SubhaloStarMetalFractions Dataset {22315, 11}
```

(continues on next page)

(continued from previous page)

```
/Subhalo/SubhaloStarMetalFractionsHalfRad Dataset {22315, 11}
/Subhalo/SubhaloStarMetalFractionsMaxRad Dataset {22315, 11}
/Subhalo/SubhaloStarMetallicity Dataset {22315}
/Subhalo/SubhaloStarMetallicityHalfRad Dataset {22315}
/Subhalo/SubhaloStarMetallicityMaxRad Dataset {22315}
/Subhalo/SubhaloStellarPhotometrics Dataset {22315, 8}
/Subhalo/SubhaloStellarPhotometricsMassInRad Dataset {22315}
/Subhalo/SubhaloStellarPhotometricsRad Dataset {22315}
/Subhalo/SubhaloVel      Dataset {22315, 3}
/Subhalo/SubhaloVelDisp  Dataset {22315}
/Subhalo/SubhaloVmax     Dataset {22315}
/Subhalo/SubhaloVmaxRad  Dataset {22315}
/Subhalo/SubhaloWindMass Dataset {22315}
```

The catalogs contain two main groups:

- `Group`. This group contains the properties of the halos.
- `Subhalos`. This group contains the properties of the subhalos. Galaxies are generally considered to be subhalos with stellar mass larger than 0.

A detailed description of the different blocks in the catalogs can be found here.

Note: For the IllustrisTNG suite, the particles in the snapshots are organized according to their FOF/Subfind group membership, as described here. However, for the snapshots in the other suites (e.g. IllustrisTNG_DM, SIMBA, Astrid), that is not the case. In those cases, instead, in order to access the particles of a specific FOF group or Subfind subhalo, a special hdf5 group called /IDs that exists in the group catalog files (as appears above for example for the SIMBA CV_5 case) needs to be used. This is a list of particle IDs (not ordered by type – all types mixed together) that is ordered according to the group membership in a similar way to how the particles are ordered in the native IllustrisTNG files. Namely, if one reorders the particles from e.g. an Astrid snapshot such that their IDs in the reordered list is the same as the IDs/ dataset from the corresponding group catalog, and then separates them by type, then by working with this reordered sets of particles, one can assign particles to groups in the standard IllustrisTNG-like approach. Note that there is an exception to the above with regards to SIMBA snapshots, which typically have duplicate IDs. There is no way to distinguish which of the particles with duplicate IDs truly belongs to a particular group except by sanity checks. For example, one in a pair of such particles may be physically too far away from the group center to plausibly truly belong to it. It is the user's responsibility to apply such sanity checks and filtering.

Reading these files with python is straightforward:

```
import numpy as np
import h5py

# catalog name
catalog = 'SIMBA/CV_5/groups_090.hdf5'

# value of the scale factor
scale_factor = 1.0

# open the catalogue
f = h5py.File(catalog, 'r')

# read the positions, velocities and masses of the FoF halos
pos_h  = f['Group/GroupPos'][:]/1e3           #positions in Mpc/h
```

(continues on next page)

(continued from previous page)

```
vel_h  = f['Group/GroupVel'][:]/scale_factor  #velocities in km/s
mass_h = f['Group/GroupMass'][:]*1e10         #masses in Msun/h

# read the positions, black hole masses and stellar masses of the subhalos/galaxies
pos_g  = f['Subhalo/SubhaloMass'][:]/1e3        #positions in Mpc/h
BH_g   = f['Subhalo/SubhaloBHMass'][:]*1e10     #black-hole masses in Msun/h
M_star = f['Subhalo/SubhaloMassType'][:,4]*1e10 #stellar masses in Msun/h

# close file
f.close()
```

Note: Differently to the snapshots, the format of these files is identical across the simulations in the IllustrisTNG and SIMBA suites.

12.1 Suite differences

The halo/subhalo catalogs are designed to be as uniform as possible across the two suites. Thus, the metallicity field in the subfind catalogs of SIMBA differ from the metallicity field of the SIMBA snapshots. The `Metallicity` and `MetalFraction` fields in the subfind catalogs follow the same convention as those from the IllustrisTNG catalogs, except that the elements are the same as in the SIMBA snapshots.

In particular:

- In IllustrisTNG snapshots and group catalogs, `Metallicity` is the total content of elements heavier than H & He, and `Metals` or `MetalFractions` is a 10-element array with the elements in this order: [H, He, C, N, O, Ne, Mg, Si, Fe, other metals]
- In SIMBA snapshots, `Metallicity` is an 11-element array with the elements in this order: [the total content of elements heavier than H & He, He,C,N,O,Ne,Mg,Si,S,Ca,Fe].
- In SIMBA FOF+Subfind catalogs, the structure is similar to IllustrisTNG: `Metallicity` is the total content of elements heavier than H & He, and `Metals` or `MetalFractions` is a 11-element array with the elements in this (SIMBA-snapshot-like) order: [H,He,C,N,O,Ne,Mg,Si,S,Ca,Fe]

In the SIMBA catalogs, the `SubhaloStellarPhotometrics` and `WindMass` fields contain some irrelevant numbers as those quantities are not calculated within the SIMBA simulations.

Please also note the differences with respect to the ordering of the particles in the snapshots and its relation to the group catalogs, which are detailed in a blue Note box above in this page.

CHAPTER

THIRTEEN

SUBLINK CATALOGS

The folders `SubLink` and `SubLink_gal` contain the SUBFIND-based SubLink and SubLink_gal merger trees. The data is organized following the general hierarchical structure described in *Suite folders*.

The format of the catalogs is the traditional one from SubLink, and we refer the user to the SubLink documentation for further details. We note that these trees are not yet available for SIMBA, but for now only for IllustrisTNG and Astrid.

We make available both SubLink and SubLink_gal versions generated by the SubLink algorithm. SubLink is the dark matter-based version, which corresponds exactly to the method used in the public data release of the IllustrisTNG project, while SubLink_gal is the baryonic-based version, which tracks the baryons across snapshots rather than the dark matter. The differences between the two are modest.

CHAPTER

FOURTEEN

ROCKSTAR CATALOGS

The folder `Rockstar` contains the Rockstar halo and subhalo catalogs. The data is organized following the general hierarchical structure described in *Suite folders*.

The format of the catalogs is the traditional one from Rockstar, and we refer the user to the Rockstar documentation for further details. We note that for the hydrodynamic simulations, Rockstar has been run using the version that account for components other dark matter such as gas, stars, and black-holes.

We also release the merger trees constructed through consistent trees. We refer the reader to the consistent trees documentation for details on the format of the files.

CHAPTER

FIFTEEN

AHF CATALOGS

The folder `AHF` contains the Amiga Halo Finder (AHF) halo and subhalo catalogs. The data is organized following the general hierarchical structure described in *Suite folders*.

The format of the catalogs is the traditional one from AHF, and we refer the user to the AHF documentation for further details.

The data is organized following the generic scheme outlined in *Data organization*.

CHAPTER

SIXTEEN

CAESAR CATALOGS

The folder `Caesar` contains the CAESAR halo and galaxy catalogs. The data is organized following the general hierarchical structure described in *Suite folders*.

CAESAR computes a large number of properties for each halo and galaxy, including physical properties such as masses and sizes of each component, dynamical properties such as velocity dispersions, and photometric properties using the FSPS package. The structure of the data is the generic one outlined in *Data organization*. There is one CAESAR catalog for each snapshot of each hydrodynamic simulation in CAMELS.

For each snapshot CAESAR generates two files:

- `fof6d_newsnaps_0XX`. This file contains the galaxies identified using the 6D Friends-of-Friends (FoF) algorithm. `XX` represents the snapshot number (from 00 to 33).
- `caesar_newsnaps_0XX.hdf5`. This file is the actual CAESAR catalog. `XX` represents the snapshot number (from 00 to 90).

The user can find details on how to read and manipulate CAESAR catalogs in the CAESAR documentation.

CHAPTER

SEVENTEEN

POWER SPECTRA

The `Pk` folder contains the power spectra. The data is organized following the general hierarchical structure described in *Suite folders*.

For each simulation we povide the power spectrum of the total matter at all available redshifts. For the hydrodynamic simulations we have also provide power spectra for the gas, dark matter, stars, and black-holes components at all redshifts. The files containing the power spectra are named as:

`Pk_type_z=Z.ZZ.txt`

where `type` can be `m` (for total matter), `g` (for gas), `c` (for dark matter), `s` (for stars), or `bh` (for black-holes). `Z.ZZ` is the redshift of the snapshot.

The data, a simple txt file, can be read as follows

```
import numpy as np

# get the name of the file with the power spectrum
f_Pk = 'CAMELS/Pk/SIMBA/LH_456/Pk_g_z=0.00.txt'

# read the data
k, Pk = np.loadtxt(f_Pk, unpack=True)

# k  contains the wavenumbers in h/Mpc
# Pk contains the power spectra measurements in (Mpc/h)^3
```

Sometimes it is useful to know the number of modes in each k-bin (e.g. if one wants to rebin the measured power spectrum). The file `Number_of_modes.txt` contains that information:

```
import numpy as np

# get the name of the file containing the number of modes
f_in = 'CAMELS/Pk/Number_of_modes.txt'

# read the file
k, Nmodes = np.loadtxt(f_in, unpack=True)

# k      contains the wavenumbers in h/Mpc
# Nmodes contains the number of modes in each k-bin
```

For snapshots at redshift zero, we additionally compute the power spectrum on all scales up to k = 1000 h/Mpc, with the small-computations performed using the HIPSTER code. This computes the Legendre multipoles of the power spectrum, P_ell(k), with a linear binning for k < 25 h/Mpc, and a logarithmic binning beyond.

All-scale spectra are computed for each LH simulation from the IllustrisTNG, IllustrisTNG_DM and SIMBA suites in both real- and redshift-space and take the form `Pk_type_allk_rsd_z=Z.ZZ.txt`, where `rsd` specifies the treatment of redshift-space distortions, either unspecified (real-space), or `RSDX` for distortions added along axis X. The file headers contain a variety of useful information about the sample and method hyperparameters.

As an example, the all-scale power spectrum of gas from simulation 42 in redshift-space is read in as follows:

```
import numpy as np

# get the name of the file with the power spectrum
f_Pk = 'CAMELS/Pk/IllustrisTNG/LH_42/Pk_g_allk_RS2_z=0.00.txt'

# read the data
k, Pk0, Pk2, Pk4 = np.loadtxt(f_Pk, unpack=True)

# k  contains the wavenumbers in h/Mpc
# Pk0 contains the power spectra monopole measurements in (Mpc/h)^3
# Pk2 contains the power spectra quadrupole measurements in (Mpc/h)^3
# Pk4 contains the power spectra hexadecapolef measurements in (Mpc/h)^3
```

All scale spectra can be computed for additional samples upon request.

CHAPTER

EIGHTEEN

BISPECTRA

The `Bk` folder contains the bispectra measurements. The data is organized following the general hierarchical structure described in *Suite folders*.

For each simulation of the latin hypercube sets we have computed the bispectrum of the total matter at redshift zero. For the hydrodynamic simulations we have also compute bispectra for the gas and dark matter at redshift zero. We do not compute bispectra of the stellar or black-hole components, due to the small number of particles in those datasets, hence the low signal-to-noise.

Bispectra are computed using two approaches. At low-k, the spectra are computed using an FFT-based approach, as implemented in the Pylians code. This estimates B(k1, k2, mu), where mu is the angle between k1 and k2. Data are assigned to a 128 x 128 x 128 grid, using triangular-shaped-cloud interpolation, before the bispectra are computed for a range of k values up to k = 5 h/Mpc.

At high-k, the FFT-based approach is inefficient, since it requires a large FFT grid. In this case, we compute bispectra using the HIPSTER code, which uses a configuration-space estimator to extend to small scales without computational penalties (strictly via a convolution with a smooth window which is of negligible importance on small scales). This estimates the Legendre multipoles of the bispectrum, B_ell(k1, k2), for a range of ell up to ell = 5 and k from 0 to 50 h/Mpc. Additional details of the method hyperparameters can be found in the bispectrum headers.

The files containing the power spectra are named as:

`Pk_type_method_rsd_z=0.00.txt`

where `type` can be `m` (for total matter), `g` (for gas), or `c` (for dark matter). `method'' is either ``lowk` or `highk`, indicating the two regimes given above, and `rsd` specifies the treatment of redshift-space distortions, either unspecified (real-space), or `RSDX` for distortions added along axis X. The file headers contain a variety of useful information about the sample.

The data is simply given as a set of text files. For the low-k spectra:

```
import numpy as np

# get the name of the file with the low-k bispectrum (here for CDM in real-space)
f_Bk = 'CAMELS/Bk/IllustrisTNG/LH_456/Bk_c_lowk_z=0.00.txt'

# read the angular bins
mu_arr =  np.loadtxt(f_Bk,max_rows=1)
# read the bispectrum
data = np.loadtxt(f_Bk,skiprows=15)
k1, k2, Bk = data[:,0], data[:,1], data[:,2:]

# k1, k2  contain the wavenumbers in h/Mpc
# Bk contains the bispectrum measurements in (Mpc/h)^6 for each k1, k2 pair (row) and mu␣
↪bin (column)
```

Whilst for the high-k spectra:

```
import numpy as np

# get the name of the file with the low-k bispectrum (here for gas using redshift-space␣
↪axis 2)
f_Bk = 'CAMELS/Bk/IllustrisTNG/LH_456/Bk_g_lowk_RS2_z=0.00.txt'

# read the bispectrum
data = np.loadtxt(f_Bk)
k1, k2, Bk = data[:,0], data[:,1], data[:,2:]
# load the angular multipoles
ells = np.arange(len(Bk[0]))

# k1, k2  contain the wavenumbers in h/Mpc
# Bk contains the bispectrum multipole measurements in (Mpc/h)^6 for each k1, k2 pair␣
↪(row) and multipole ell (column).
```

CHAPTER

NINETEEN

PROBABILITY DISTRIBUTION FUNCTIONS

The `PDF` folder contains the estimated probability distribution functions (PDFs). The data is organized following the general hierarchical structure described in *Suite folders*.

We provide PDFs for all the 3D grid files in the CAMELS Multifield Dataset (CMD) (See *CAMELS Multifield Dataset*), each containing 1,000 grids from the LH set.

The PDFs are stored as `.npy` files and they are named as `hist_Grids_prefix_sim_LH_grid_z=redshift.npy`, where `prefix` is the word identifying each field (`HI`, `Vgas`, etc.), `sim` can be `IllustrisTNG`, `SIMBA`, `Nbody_IllustrisTNG`, or `Nbody_SIMBA`, `grid` can be `128`, `256`, or `512` and `redshift` can be 0, 0.5, 1, 1.5 or 2. These files can be read with numpy as follows:

```
import numpy as np

# name of the pdf file
pdfgrids = 'hist_Grids_HI_SIMBA_LH_128_z=0.00.npy'

# read the data
pdf = np.load(pdfgrids) #--shape of pdf is (1000,500)

# get the bin counts for the 4th grid
print(pdf[3])
```

The file contains 1,000 entries with 500 values per entry denoting the number of counts in 500 bins for that entry. See the CAMELS data release for more details on how the PDFs are calculated.

CHAPTER

TWENTY

VIDE VOIDS

The `VIDE_Voids` contains the void catalogues. The data is organized following the general hierarchical structure described in *Suite folders*.

For each hydrodynamic simulation we provide voids identified in the galaxy distribution at redshift 0 with the VIDE void finder.

For each void catalogue we provide a folder with the standard VIDE output that is described here.

CHAPTER

TWENTYONE

LYMAN-ALPHA SPECTRA

The `Lya` folder contains the mock Lyman-α spectra. The data is organized following the general hierarchical structure described in *Suite folders*.

For each simulation snapshot we provide Lyman-alpha spectra for 5,000 random sightlines. These spectral lines are contained within an hdf5 file and can be read using the same code they were generated with.

The data is organized such that the spectra corresponding to a snapshot resides in an appropriately named folder within the simulation folder. The spectra files are located at

`suite/sim/SPECTRA_0##/Lya-spectra.hdf5`

where `suite` can be either `IllustrisTNG` or `SIMBA`, `sim` is the simulation of interest, and `##` is the snapshot number (for snapshot numbers 0 through 9, a 0 must pad the front). For example `IllustrisTNG/1P_1_n5/SPECTRA_001/Lya-spectra.hdf5` is the Lyman-alpha spectra for the IllustrisTNG suite `1P_1_n5` simulation at redshift 5.

The easiest way to read the data is using the same code that generated the spectra; see the Fake Spectra GitHub for instructions on how to install it. The data can then be read as follows:

```
from fake_spectra.plot_spectra import PlottingSpectra

# We use their plotting routine to read in the data.
fs = PlottingSpectra(num=1, base="suite/sim", savefile="Lya-spectra.hdf5", label="My␣
↪label")

# num       the simulation snapshot number as an integer (you should not pad numbers 0␣
↪through 9 for num)
# base      the directory that the SPECTRA_0## directory is located in
# savefile  the name of the spectra file (they are all "Lya-spectra.hdf5")
# label     an identifier for your spectra (this is usually unimportant)
```

It is important to note that when given the information above, the code is looking for `base/SPECTRA_0num/savefile` (where `num` is padded automatically with a 0 by the code if num is 0 through 9). Because of this the user is cautioned to ensure that the spectra file remains in a directory called `SPECTRA_0num`. This will not be a problem unless you restructure the snapshot data after download or downloaded just the spectra file.

Once the data is loaded in, the user can retrieve optical depths for one or all of the spectral lines in the file:

```
from fake_spectra.plot_spectra import PlottingSpectra
import numpy as np

# Loading in spectra for the IllustrisTNG suite, 1P_4_3 simulation, and snapshot 001.
fs = PlottingSpectra(num=1, base="IllustrisTNG/1P_4_3", savefile="Lya-spectra.hdf5",␣
↪label="My label")
```

(continues on next page)

(continued from previous page)

```
 # To retrieve optical depths for all 5000 spectral lines:
 taus = fs.get_tau("H", 1, 1215)
 # The arguments entered above correspond to hydrogen Lyman-alpha. Changing these would␣
↪result in errors.

 # To retrieve optical depths for a single spectral line:
 tau = fs.get_tau("H", 1, 1215, 0)
 # The last argument indicates which line you want tau for. This can be anywhere from 0␣
↪to 4999.

 # Flux is proportional to e^-tau:
 flux = np.exp(-tau)

# To plot flux as a function of relative velocity, calculate the relative velocity array:
 rel_v = ( np.arange(0, np.size(tau)) ) * fs.dvbin
```

The spectra is available for both the IllustrisTNG and SIMBA suites for all the simulation sets at all redshifts. See the CAMELS data structure documentation for details on available simulations.

Additional spectra samples can be calculated for the simulations using the Fake Spectra code; the code also has an assortment of useful plotting methods, see the documentation for further details.

CHAPTER

TWENTYTWO

X-RAYS

The `X-rays` folder contains the X-rays data. The data is organized following the general hierarchical structure described in *Suite folders*.

We provide X-ray SIMPUT files for every halo above 10^12 M_solar for the 032 snapshot at z=0.05. Within each subdirectory containing a single simulation (e.g. `LH_100`) is a directory for the snapshot, of which there is currently only one `snap_032`. For every FoF halo above 10^12 M_solar, there is a SIMPUT fits file of the form:

`IllustrisTNG.LH_100.snap_032.halo_010.100ks.z0.05.z_simput.fits`

in the directory structure:

`IllustrisTNG/LH_100/snap_032/`

within which halos are ordered by the FoF group ID in the group finder catalogue (e.g. `halo_010`).

There exist a total of 160,693 halos:

- 5,939 IllustrisTNG_1P
- 2,003 IllustrisTNG_CV
- 342 IllustrisTNG_EX
- 75,340 IllustrisTNG_LH
- 5,157 SIMBA_1P
- 1,706 SIMBA_CV
- 70,206 SIMBA_LH
- 160,693 total

Each halo SIMPUT file consists of the main *simput* file, which is always the same size 11,520 bytes and represents the fits header, and a *phlist* file, which holds the photons and represents the fits data file:

`IllustrisTNG.LH_100.snap_032.halo_010.100ks.z0.05.z_simput.fits`

and

`IllustrisTNG.LH_100.snap_032.halo_010.100ks.z0.05.z_phlist.fits`

The pair of files, referred to as SIMPUT files, hold a Monte-Carlo-generated sample of photons produced by the pyXSIM software package that would be collected in an aperture of 3,000 cm^2 over 100,000 seconds for the object placed at z=0.05. The SIMPUT file format is the standard file format used in simulations of X-ray observations, and serves as the input into other software that generates mock observations for a specific telescope, such as SOXS and SIXTE.

It is helpful to have a background in X-ray simulation software to use these files. The limited collecting aperture in exposure time (3,000 x 100,000 = 3e+08 cm^2 s) means that mock observation software cannot simulate longer observations, but it is very rare for a telescope to observe deeper than this (e.g. eROSITA has an effective area of 2000

cm^2 and scans the sky to an average exposure of 2,000 seconds). While projected at z=0.05, it should be possible to scale the observation to be closer in the low redshift regime where received flux scales as z^-2. Note that the cosmology sets the angular size and luminosity distances, which in this case is affected by the Omega_M parameter.

Because the analysis of the SIMPUT files is rather complex, we also provide a reduced data format in the form of a single file in the base X-rays directory:

`CAMELS.Xray.hdf5`

This file has hold projected soft X-ray (0.5-2.0 keV) surface brightness profiles in units of ergs s^-1 kpc^-2 in 7 logarithmic radial bins ranging from 10-1,280 kpc. Each halo is in a hierarchical directory structure.

CHAPTER

TWENTYTHREE

CAMELS CGM PROFILES

The folder `Profiles` contains the CGM profiles. The data is organized following the general hierarchical structure described in *Suite folders*.

For each snapshot of each simulation, we provide three-dimensional spherically-averaged profiles of gas density, thermal pressure, gas mass-weighted temperature, and gas mass-weighted metallicity for the simulations of the 1P, LH, and CV sets of both the IllustrisTNG and SIMBA suites.

Specifically, we use illstack_CAMELS, a CAMELS-specific version of the original, more general code illstack to generate the three-dimensional profiles, extending radially from 0.01-10 Mpc in 25 log10 bins. The profiles are stored in hdf5 format which can be read with the python script provided in the repository.

The profiles are located as

`Profiles/suite/sim/suite_sim_0##.hdf5`

where `suite` is either `IllustrisTNG` or `SIMBA`, `sim` is the simulation of interest, e.g. `1P_4_n5`, `LH_42`, `CV_130`, and `0##` is the snapshot number, ranging from `000` to `033`.

Below is an example python script for extracting the profile data from the hdf5 file:

```
import matplotlib.pyplot as plt
import numpy             as np
import h5py

#-------------------------input section---------------
suite='SIMBA'
sim='CV_0'
snap='024'
#-----------------------------------------------------
data_dir='/mnt/ceph/users/camels/PUBLIC_RELEASE/Sims'
prof_dir='/mnt/home/elau/ceph/illstack_CAMELS/Profiles/'

def extract(simulation,snap):

    '''
    Return values of the CGM profiles from the CAMELS simulation

    Inputs:
      simulation: string, name of the simulation, e.g., 1P_5_2, LH_123, CV_12
      snap: string, number of the snapshot, from '000' to '033', '033' being the last␣
↪snapshot corresponding to z=0

    Outputs:
```

(continues on next page)

(continued from previous page)

```
      z: float, redshift
      r: np array, radial bin on kpc
      val_dens: np array, density profile in g/cm^3
      val_pres: np array, volume-weighted thermal pressure profile in erg/cm^3
      val_temp_mw: np array, mass-weighted temperature in K
      val_metals_mw: np array, mass-weighted metallcity in Zsun
      mh: np array, halo mass (M200c) in Msun
      rh: np array, halo radius (R200c) in kpc

   '''

   h=0.6711
   omegab=0.049
   omegam,sigma8=np.loadtxt(data_dir+'/'+suite+'/'+simulation+'/CosmoAstro_params.txt',
↪usecols=(1,2),unpack=True)
   omegalam=1.0-omegam

   kb = 1.38e-16 # erg/K
   erg_to_keV = 6.242e+8
   K_to_keV = kb * erg_to_keV
   m_e = 9.11e-28 # electron mass in g
   m_p = 1.6726e-24 # in g
   XH = 0.76 #primordial hydrogen fraction
   mu = 0.58824; # X=0.76 assumed
   mu_e = mue = 2.0/(1.0+XH); # X=0.76 assumed
   Msun = 1.989e33
   kpc = 3.0856e21

   data_file= data_dir+'/'+suite+'/'+simulation+'/snap_'+snap+'.hdf5'
   profile_file = prof_dir+'/'+suite+'/'+simulation+'/'+suite+'_'+simulation+'_'+snap+'.
↪hdf5'
   b=h5py.File(data_file,'r')
   z=b['/Header'].attrs[u'Redshift']

   comoving_factor = (1.0+z)

   density_conversion_factor = Msun*kpc**(-3) * 1e10 * h**2 * comoving_factor**3
   #from 1e10Msol/h*(km/s)**2 ckpc^{-3} to keV cm^{-3}
   pressure_conversion_factor = density_conversion_factor * 1e10 * erg_to_keV
   temperature_conversion_factor = (1e5)**2 * kb * erg_to_keV

   stacks=h5py.File(profile_file,'r')
   val            = stacks['Profiles']
   val_dens       = np.array(val[0,:,:]) * density_conversion_factor #density in g cm^3
   val_pres       = np.array(val[1,:,:]) * pressure_conversion_factor  #thermal␣
↪pressure in keV cm^-3
   val_metals_mw  = np.array(val[2,:,:])/Zsun #mass-weighted metallicity in solar units
   val_temp_mw    = np.array(val[3,:,:]) * temperature_conversion_factor #mass-weighted␣
↪temperature in keV
   bins           = np.array(stacks['nbins']) #number of radial bins
   r              = np.array(stacks['r']) / h / comoving_factor #radial bins in␣
↪comoving kpc
```

(continues on next page)

(continued from previous page)

```
    nprofs          = np.array(stacks['nprofs']) #number of halos
    m200c           = np.array(stacks['Group_M_Crit200'])*1e10 / h #M200c in Msol
    r200c           = np.array(stacks['Group_R_Crit200']) / h / comoving_factor #R200c in␣
↪kpc

    return z, r, val_dens, val_pres, val_temp_mw, val_metals_mw, m200c, r200c
```

CHAPTER

TWENTYFOUR

CAMELS MULTIFIELD DATASET

The `CMD` folder contains the CAMELS Multifield Dataset, CMD, a collection of hundreds of thousands of 2D maps and 3D grids generated from the simulations of the CAMELS project.

We note that the data structure in CMD is different to the generic one described in *Suite folders*. We refer the user to the CDM documentation for details on the data structure and how to read and manipulate this data.

CHAPTER

TWENTYFIVE

CAMELS-SAM

The `SCSAM` folder contains the CAMELS-SAM data. CAMELS-SAM is set of collection of more than 1,000 galaxy catalogues, each of them from a different cosmological and astrophysical model spanning a periodic comoving volume of $(100\ h^{-1}\mathrm{Mpc})^3$. Each catalogue was constructed by running an N-body simulation and applying the Santa Cruz Semi-Analytic Model on top of it.

We refer the user to the CAMELS-SAM website for further details on it, its data structure (that is different to the generic one described in *Suite folders*), and how to read and manipulate the data.

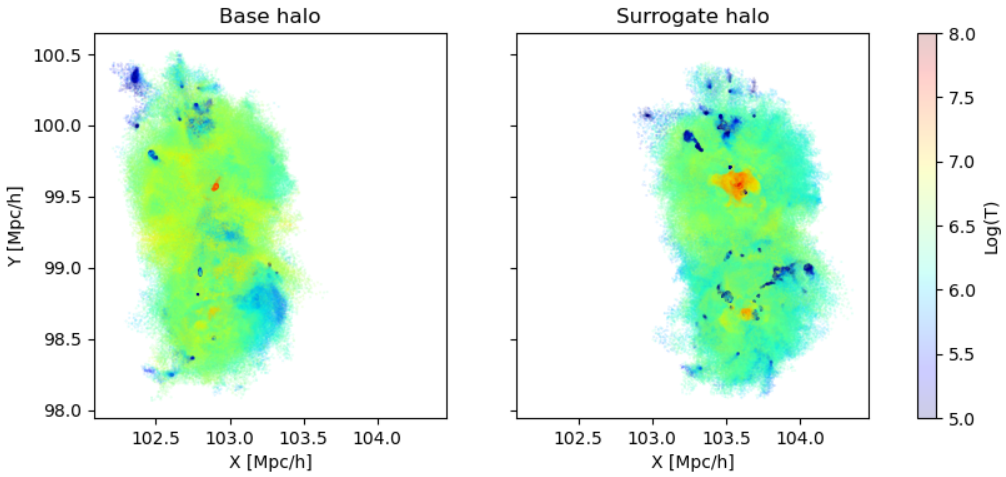
CHAPTER

TWENTYSIX

CAMELS-ZOOMGZ

CAMELS-zoomGZ is a suite of 768 zoom-in simulations of halos between the mass range of $M^{13} M_\odot\, h^{-1}$ – $M^{14.5} M_\odot\, h^{-1}$, and spanning 28 astrophysical and cosmological parameters in the IllustrisTNG galaxy formation model. The IllustrisTNG parameter space was sampled using a novel reduced variance sampling and emulation method called CARPoolGP, which differs from the standard Latin hypercube or Sobol sequence approaches.

We refer users to the CAMELS-zoomGZ website for more details on the simulations.



CHAPTER

TWENTYSEVEN

TUTORIALS

We provide multiple tutorials showing how to read and manipulate CAMELS data. Inside each tutorial there is a link to open the notebook in our binder, that contains all CAMELS data.

27.1 Reading and manipulating snapshots

Example of how to read and manipulate CAMELS snapshots

[1]:
```
import numpy as np
import h5py
import hdf5plugin
```

Get the name of the snapshot

[2]:
```
f_snap = '/home/jovyan/Data/Sims/IllustrisTNG/CV/CV_0/snap_033.hdf5'
```

Open the file

[3]:
```
data = h5py.File(f_snap, 'r')
```

Read the header of the snapshot

[4]:
```
BoxSize      = data['Header'].attrs[u'BoxSize']/1e3    #size of the snapshot in comoving␣
↪Mpc/h
redshift     = data['Header'].attrs[u'Redshift']       #reshift of the snapshot
scale_factor = data['Header'].attrs[u'Time']           #scale factor
h            = data['Header'].attrs[u'HubbleParam']    #value of the hubble parameter in␣
↪100 km/s/(Mpc/h)
Masses       = data['Header'].attrs[u'MassTable']*1e10 #masses of the particles in Msun/h
Np           = data['Header'].attrs[u'NumPart_Total']  #total number of particles for␣
↪specie
Omega_m      = data['Header'].attrs[u'Omega0']         #Omega_matter
Omega_L      = data['Header'].attrs[u'OmegaLambda']    #Omega_baryon
Omega_b      = data['Header'].attrs[u'OmegaBaryon']    #Omega_Lambda
```

[5]:
```
print('Box size:                %.2f Mpc/h'%BoxSize)
print('snapshot redshift:       %.2f'%redshift)
print('Number of gas  particles: %d'%Np[0])
print('Number of star particles: %d'%Np[4])
```

(continues on next page)

(continued from previous page)

```
print('Omega_m:                  %.3f'%Omega_m)
print('Omega_b:                  %.3f'%Omega_b)
print('Omega_L:                  %.3f'%Omega_L)
```

```
Box size:                 25.00 Mpc/h
snapshot redshift:        0.00
Number of gas  particles: 15695635
Number of star particles: 636474
Omega_m:                  0.300
Omega_b:                  0.049
Omega_L:                  0.700
```

Read the positions of the gas, dark matter, stars, and black-holes particles

[6]:
```
pos_gas   = data['PartType0/Coordinates'][:]/1e3 #Mpc/h
pos_dm    = data['PartType1/Coordinates'][:]/1e3 #Mpc/h
pos_stars = data['PartType4/Coordinates'][:]/1e3 #Mpc/h
pos_bh    = data['PartType5/Coordinates'][:]/1e3 #Mpc/h
```

[7]:
```
print('%.2f < pos_gas_X < %.2f'%(np.min(pos_gas[:,0]), np.max(pos_gas[:,0])))
print('%.2f < pos_gas_Y < %.2f'%(np.min(pos_gas[:,1]), np.max(pos_gas[:,1])))
print('%.2f < pos_gas_Z < %.2f'%(np.min(pos_gas[:,2]), np.max(pos_gas[:,2])))
```

```
0.00 < pos_gas_X < 25.00
0.00 < pos_gas_Y < 25.00
0.00 < pos_gas_Z < 25.00
```

[8]:
```
print('Number of star particles: %d'%pos_stars.shape[0])
```

```
Number of star particles: 636474
```

27.2 Computing power spectra

Example of how to compute a power spectrum from CAMELS data

[1]:
```
import numpy as np
import h5py
import hdf5plugin
import MAS_library as MASL
import Pk_library as PKL
import matplotlib.pylab as plt
```

Get the name of the snapshot

[2]:
```
f_snap = '/home/jovyan/Data/Sims/SIMBA/LH/LH_367/snap_020.hdf5'
```

Open the snapshot file

[3]:
```
data = h5py.File(f_snap, 'r')
```

Read the snapshot header

[4]:
```
BoxSize      = data['Header'].attrs[u'BoxSize']/1e3    #size of the snapshot in comoving␣
↪Mpc/h
redshift     = data['Header'].attrs[u'Redshift']       #reshift of the snapshot
scale_factor = data['Header'].attrs[u'Time']           #scale factor
h            = data['Header'].attrs[u'HubbleParam']    #value of the hubble parameter in␣
↪100 km/s/(Mpc/h)
Masses       = data['Header'].attrs[u'MassTable']*1e10 #masses of the particles in Msun/h
Np           = data['Header'].attrs[u'NumPart_Total']  #total number of particles for␣
↪specie
Omega_m      = data['Header'].attrs[u'Omega0']         #Omega_matter
Omega_L      = data['Header'].attrs[u'OmegaLambda']    #Omega_Lambda
```

Read the positions and the masses of the gas particles

[5]:
```
pos_gas   = data['PartType0/Coordinates'][:]/1e3 #Mpc/h
mass_gas  = data['PartType0/Masses'][:]*1e10     #Msun/h
```

Check the masses positions and masses of the gas particles

[6]:
```
print('Box size: %.2f Mpc/h'%BoxSize)
print('Snapshot redshift: %.2f'%redshift)
print('%.2f < pos_gas_X < %.2f'%(np.min(pos_gas[:,0]), np.max(pos_gas[:,0])))
print('%.3e < mass_gas  < %.3e'%(np.min(mass_gas), np.max(mass_gas)))
```

```
Box size: 25.00 Mpc/h
Snapshot redshift: 0.86
0.00 < pos_gas_X < 25.00
1.219e+07 < mass_gas  < 3.811e+07
```

Make sure the positions are a numpy float32 array

[7]:
```
pos_gas = pos_gas.astype(np.float32)
```

Define array the will contain the gas density field

[8]:
```
grid  = 512
delta = np.zeros((grid,grid,grid), dtype=np.float32) #grid will have (512,512,512) voxels
```

Assign gas particle positions and masses to the regular grid We will use the Cloud-in-Cell mass assignment scheme

[9]:
```
MAS     = 'CIC'
verbose = True
MASL.MA(pos_gas, delta, BoxSize, MAS, W=mass_gas, verbose=verbose)
```

```

Using CIC mass assignment scheme with weights
Time taken = 1.496 seconds

```

Check that the mass in the grid is the same as the mass of all particles

[10]:
```
print('Sum of all particle masses:         %.3e Msun/h'%np.sum(mass_gas, dtype=np.
↪float64))
print('Sum of the mass in all grid voxels: %.3e Msun/h'%np.sum(delta, dtype=np.float64))
```

```
Sum of all particle masses:         2.062e+14 Msun/h
Sum of the mass in all grid voxels: 2.062e+14 Msun/h
```

calculate the gas overdensity field delta = mass_gas / < mass_gas > - 1

[11]:
```
delta /= np.mean(delta, dtype=np.float32)
delta -= 1.0
```

Check that mean is zero, and minimum cant be smaller than -1

[12]:
```
print('%.2f < delta < %.2f'%(np.min(delta), np.max(delta)))
print('< delta > = %.2e'%(np.mean(delta, dtype=np.float64)))
```

```
-1.00 < delta < 11278.17
< delta > = -2.41e-06
```

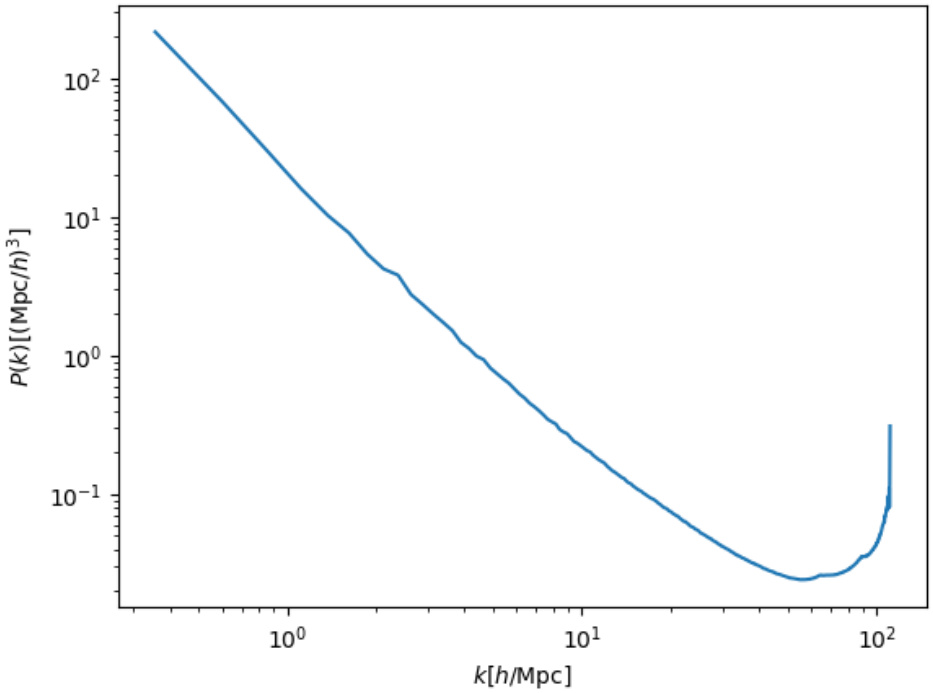
Calculate the power spectrum

[13]:
```
axis     = 0     #specifies the axes along which redshift-space distortions are place.␣
↪Not used in real-space
MAS      = 'CIC' #specifies the mass assignment scheme used to construct the density␣
↪field
threads  = 1     #number of openmp threads
verbose  = True
Pk_class = PKL.Pk(delta, BoxSize, axis, MAS, threads, verbose)
k        = Pk_class.k3D
Pk       = Pk_class.Pk[:,0]
```

```

Computing power spectrum of the field...
Time to complete loop = 7.01
Time taken = 12.10 seconds
```

Plot power spectrum

[14]:
```
plt.xlabel(r'$k [h/{\rm Mpc}]$')
plt.ylabel(r'$P(k) [({\rm Mpc}/h)^3]$')
plt.xscale('log')
plt.yscale('log')
plt.plot(k, Pk)
plt.show()
```



nbsphinx-code-borderwhite

[ ]:

27.3 Creating images from snapshots

Here we will show how to create an image of the gas surface density from a CAMELS snapshot

[1]:
```
import numpy as np
import h5py
import hdf5plugin
import camels_library as CL
import MAS_library as MASL
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
```

Get the name of the snapshot

[2]:
```
f_snap = '/home/jovyan/Data/Sims/IllustrisTNG/1P/1P_1_3/snap_033.hdf5'
```

Open file and read the header

[3]:
```
data         = h5py.File(f_snap, 'r')
BoxSize      = data['Header'].attrs[u'BoxSize']/1e3    #size of the snapshot in comoving␣
↪Mpc/h
redshift     = data['Header'].attrs[u'Redshift']       #reshift of the snapshot
scale_factor = data['Header'].attrs[u'Time']           #scale factor
h            = data['Header'].attrs[u'HubbleParam']    #value of the hubble parameter in␣
```

(continues on next page)

(continued from previous page)

```
↪100 km/s/(Mpc/h)
Masses       = data['Header'].attrs[u'MassTable']*1e10 #masses of the particles in Msun/h
Np           = data['Header'].attrs[u'NumPart_Total']  #total number of particles for␣
↪specie
Omega_m      = data['Header'].attrs[u'Omega0']         #Omega_matter
Omega_L      = data['Header'].attrs[u'OmegaLambda']    #Omega_Lambda
```

Read the positions and masses of the gas particles

[4]:
```
pos_gas  = data['PartType0/Coordinates'][:]/1e3 #Mpc/h
mass_gas = data['PartType0/Masses'][:]*1e10     #Msun/h
```

We now assume that each gas particle represents an uniform sphere where its radius is the distance to its closest 32th neighbor. In this cases we are taking into account periodic boundary conditions by setting the size of the box to be BoxSize. Note that we are multiplying that value by a very small number to avoid problems if there are particles just at the edge.

[5]:
```
k          = 32   #will compute distance to 32th closest neighbor
threads    = 1    #number of openmp threads
verbose    = True #whether to print some information
radius_gas = CL.KDTree_distance(pos_gas,  pos_gas, k, BoxSize*(1.0+1e-8), threads,␣
↪verbose) #Mpc/h
```

```
Time to build KDTree = 15.161 seconds
Time to find k-neighbors = 231.620 seconds
```

Lets define the map that will contain the data

[6]:
```
grid_size = 256 #the map will contain (grid_size x grid_size) pixels
map_gas   = np.zeros((grid_size, grid_size), dtype=np.float64)
```

Now lets define the boundaries of our slice

[7]:
```
x_min, x_max = 0., BoxSize  #Mpc/h
y_min, y_max = 0., BoxSize  #Mpc/h
z_min, z_max = 0., 5.0      #Mpc/h
```

Find the particles that are inside the slice

[8]:
```
indexes = np.where((pos_gas[:,0]>=x_min) & (pos_gas[:,0]<x_max) &
                   (pos_gas[:,1]>=y_min) & (pos_gas[:,1]<y_max) &
                   (pos_gas[:,2]>=z_min) & (pos_gas[:,2]<z_max))
pos_gas_    = pos_gas[indexes]
mass_gas_   = mass_gas[indexes]
radius_gas_ = (radius_gas[indexes])
print('%.3f < X slice < %.3f'%(np.min(pos_gas_[:,0]), np.max(pos_gas_[:,0])))
print('%.3f < Y slice < %.3f'%(np.min(pos_gas_[:,1]), np.max(pos_gas_[:,1])))
print('%.3f < Z slice < %.3f'%(np.min(pos_gas_[:,2]), np.max(pos_gas_[:,2])))
```

```
0.000 < X slice < 25.000
0.000 < Y slice < 25.000
0.000 < Z slice < 5.000
```

Project the particles along the axis perpendicular to the image

[9]:
```
pos_gas_ = pos_gas_[:,[0,1]] #image will show XY plane, so we project along Z axis
pos_gas_ = np.ascontiguousarray(pos_gas_) #This is required for Pylians to speed the␣
↪calculation up
pos_gas_ = pos_gas_.astype(np.float32)
print(pos_gas_.shape)
print('%.3f < X slice < %.3f'%(np.min(pos_gas_[:,0]), np.max(pos_gas_[:,0])))
print('%.3f < Y slice < %.3f'%(np.min(pos_gas_[:,1]), np.max(pos_gas_[:,1])))
```

```
(5555960, 2)
0.000 < X slice < 25.000
0.000 < Y slice < 25.000
```

[10]:
```
tracers     = 1000
r_divisions = 20
periodic    = True
verbose     = True
pos_gas_    = pos_gas_.astype(np.float32)
mass_gas_   = mass_gas_.astype(np.float32)
radius_gas_ = radius_gas_.astype(np.float32)
MASL.projected_voronoi(map_gas, pos_gas_, mass_gas_, radius_gas_, x_min, y_min, BoxSize,
                       tracers, r_divisions, periodic, verbose)
```

```
Calculating projected mass of the voronoi tracers...
Time taken = 96.141 s
```

Check that the mass in the image is the same as in the slice

[11]:
```
print('Mass in image: %.3e Msun/h'%(np.sum(map_gas)))
print('Mass in slice: %.3e Msun/h'%(np.sum(mass_gas_)))
print('%.3e < Mass in image < %.3e'%(np.min(map_gas), np.max(map_gas)))
```

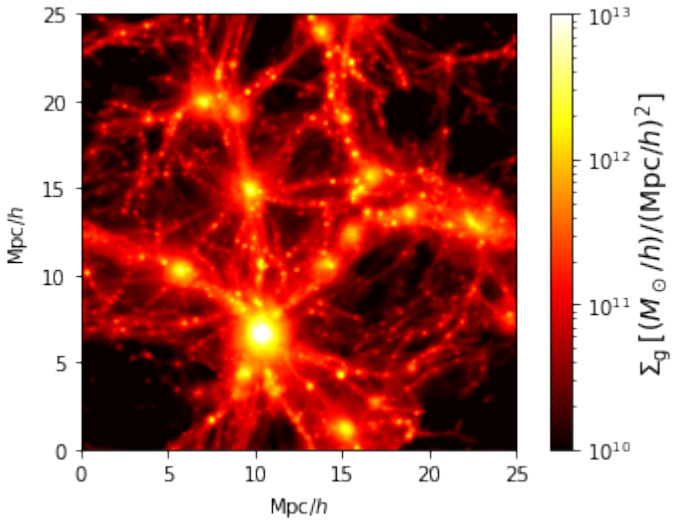
```
Mass in image: 7.384e+13 Msun/h
Mass in slice: 7.384e+13 Msun/h
2.699e+07 < Mass in image < 4.089e+11
```

At this point the map contains the gas mass in every pixel. To compute mass surface density divide by area

[12]:
```
area_pixel = (BoxSize/grid_size)**2 #(Mpc/h)^2
map_gas /= area_pixel #(Msun/h)/(Mpc/h)^2
```

Show the image

[13]:
```
plt.xlabel(r'${\rm Mpc}/h$')
plt.ylabel(r'${\rm Mpc}/h$')
cax = plt.imshow(map_gas,cmap=plt.get_cmap('hot'),origin='lower',
                     interpolation='bicubic',
                     extent=[x_min, x_max, y_min, y_max],
                     norm = LogNorm(vmin=1e10,vmax=1e13))
cbar = plt.colorbar(cax) #in ax2 colorbar of ax1
cbar.set_label(r"$\Sigma_{\rm g}\,[(M_\odot/h)/({\rm Mpc}/h)^2]$",fontsize=14,labelpad=0)
```



nbsphinx-code-borderwhite

27.4 Working with particles in halos and subhalos/galaxies

Here we show four different cases: - Selecting the particles inside IllustrisTNG halos - Selecting the particles inside halos of simulations that are not IllustrisTNG - Selecting the particles inside IllustrisTNG subhalos/galaxies - Selecting the particles inside subhalos/galaxies from simulatons that are not IllustrisTNG

The reason why the procedure for IllustrisTNG and non-IllustrisTNG simulations is slighty different is because the snapshots of the IllustrisTNG simulations are sorted such as the first particles in the snapshots are the one that belong to the first halo, the next particles in the simulations are the ones that belong to the second halos...etc. For other simulations, this is not the case so we need to read an additional block that contains the IDs of the particles in the halos/subhalos/galaxies. For details on how the data is organized in the IllustrisTNG simulations please check this.

Note that the Nbody counterparts of the IllustrisTNG simulations fall into non-IllustrisTNG simulations.

[1]:
```
import numpy as np
import h5py
```

27.4.1 Particles inside FoF halos: IllustrisTNG

We first showing how to identify the particles that belong to a given FoF halo in IllustrisTNG

[2]:
```
# get the name of the snapshot and its corresponding Subfind halo catalog
f_snapshot = '/home/jovyan/Data/Sims/IllustrisTNG/CV/CV_0/snap_033.hdf5'
f_catalog  = '/home/jovyan/Data/FOF_Subfind/IllustrisTNG/CV/CV_0/fof_subhalo_tab_033.hdf5
↪'
```

Lets read the halo catalog

[3]:
```
f = h5py.File(f_catalog, 'r')
f.keys()
```

[3]:
```
<KeysViewHDF5 ['Config', 'Group', 'Header', 'IDs', 'Parameters', 'Subhalo']>
```

Lets see what is in the Group block

[4]:
```
f['Group'].keys()
```

[4]:
```
<KeysViewHDF5 ['GroupBHMass', 'GroupBHMdot', 'GroupCM', 'GroupFirstSub',
↪'GroupGasMetalFractions', 'GroupGasMetallicity', 'GroupLen', 'GroupLenType', 'GroupMass
↪', 'GroupMassType', 'GroupNsubs', 'GroupPos', 'GroupSFR', 'GroupStarMetalFractions',
↪'GroupStarMetallicity', 'GroupVel', 'GroupWindMass', 'Group_M_Crit200', 'Group_M_
↪Crit500', 'Group_M_Mean200', 'Group_M_TopHat200', 'Group_R_Crit200', 'Group_R_Crit500',
↪ 'Group_R_Mean200', 'Group_R_TopHat200']>
```

Now, lets read the positions, velocities, masses, and lenghts of the halos

[5]:
```
pos_h     = f['Group/GroupPos'][:]/1e3   #Mpc/h
vel_h     = f['Group/GroupVel'][:]       #km/s
SFR_h     = f['Group/GroupSFR'][:]       #Msun/yr
mass_h    = f['Group/GroupMass'][:]*1e10 #Msun/h
len_h     = f['Group/GroupLen'][:]       #the total number of particles in the halo␣
↪(gas+dm+stars+black_holes)
lentype_h = f['Group/GroupLenType'][:]   #the number of particles in a halo by particle␣
↪type
f.close()
```

Lets print the position, velocity, mass and length of a given halo

[6]:
```
index = 167 #index of the halo
print('position:',pos_h[index],'Mpc/h')
print('velocity:',vel_h[index],'km/s')
print('mass: %.3e Msun/h'%mass_h[index])
print('total length:',len_h[index],'particles')
print('number of particles by type:',lentype_h[index])
```

```
position: [ 9.327968  17.098047   3.0652359] Mpc/h
velocity: [-51.31196  -29.259886 -33.310223] km/s
mass: 3.354e+11 Msun/h
total length: 7542 particles
number of particles by type: [2976 4543    0    0   20    3]
```

Now imagine we want to select the gas particles that belong to this halo. For this, we need to know where the particles start in the list. This can be done as follows:

[7]:
```
offset = np.sum(lentype_h[:index], axis=0) #this is the sum of the lengths of all FoF␣
↪halos previous to the one we consider
```

We can now read the particles in the snapshot and identify the ones in this halo as

[8]:
```
f = h5py.File(f_snapshot, 'r')
print(f.keys())
```

```
<KeysViewHDF5 ['Header', 'PartType0', 'PartType1', 'PartType4', 'PartType5']>
```

Lets see what gas properties do we have

[9]:
```
print(f['PartType0'].keys())
```

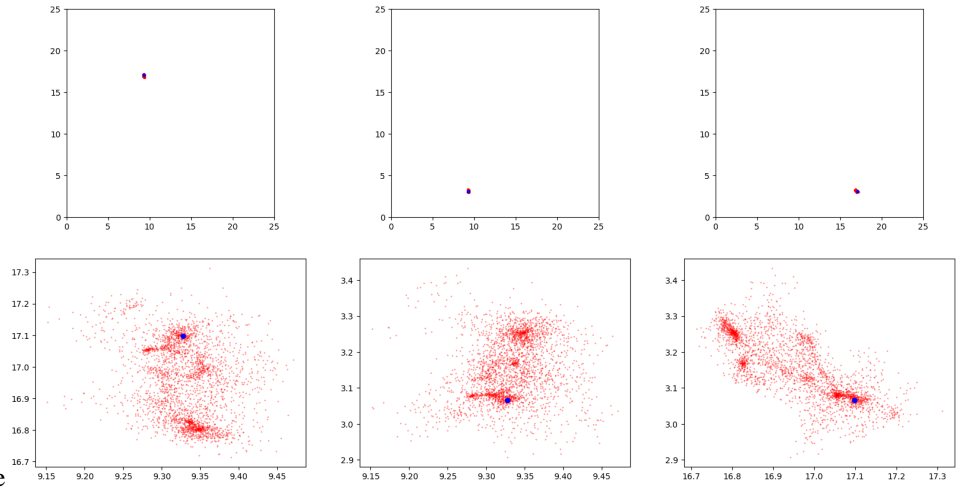
```
<KeysViewHDF5 ['Coordinates', 'Density', 'ElectronAbundance', 'EnergyDissipation', 'GFM_
↪AGNRadiation', 'GFM_CoolingRate', 'GFM_Metallicity', 'GFM_Metals', 'GFM_MetalsTagged',
↪'GFM_WindDMVelDisp', 'GFM_WindHostHaloMass', 'InternalEnergy', 'Machnumber',
↪'MagneticField', 'MagneticFieldDivergence', 'Masses', 'NeutralHydrogenAbundance',
↪'ParticleIDs', 'Potential', 'StarFormationRate', 'SubfindDMDensity', 'SubfindDensity',
↪'SubfindHsml', 'SubfindVelDisp', 'Velocities']>
```

Now, lets read the positions, velocities, and star formation rates of the gas particles in the above halo

[10]:
```
# the gas particles of this halos will start at offset[0] and will contain lentype_h[0]␣
↪particles
start = offset[0]
end   = offset[0]+lentype_h[index,0]
pos_gas = f['PartType0/Coordinates'][start:end]/1e3   #Mpc/h
vel_gas = f['PartType0/Velocities'][start:end]        #km/s
sfr_gas = f['PartType0/StarFormationRate'][start:end] #Msun/yr
f.close()

#pos_dm = f['PartType1/Coordinates'][:]/1e3   #Mpc/h
#vel_dm = f['PartType1/Velocities'][:]        #km/s
```

Lets check if the positions of the particles are around the center of the halo

[11]:
```
import matplotlib.pyplot as plt
from pylab import *
fig = figure(figsize=(20,10))
ax1 = fig.add_subplot(231)
ax2 = fig.add_subplot(232)
ax3 = fig.add_subplot(233)
ax4 = fig.add_subplot(234)
ax5 = fig.add_subplot(235)
ax6 = fig.add_subplot(236)
for ax in [ax1,ax2,ax3]:
    ax.set_aspect('equal')
    ax.set_xlim([0,25])
    ax.set_ylim([0,25])
ax1.scatter(pos_gas[:,0], pos_gas[:,1], s=0.1,c='r')
ax2.scatter(pos_gas[:,0], pos_gas[:,2], s=0.1,c='r')
ax3.scatter(pos_gas[:,1], pos_gas[:,2], s=0.1,c='r')
ax1.scatter(pos_h[index,0], pos_h[index,1], s=10, c='b')
ax2.scatter(pos_h[index,0], pos_h[index,2], s=10, c='b')
ax3.scatter(pos_h[index,1], pos_h[index,2], s=10, c='b')
# now make a zoom-in
ax4.scatter(pos_gas[:,0], pos_gas[:,1], s=0.1,c='r')
ax5.scatter(pos_gas[:,0], pos_gas[:,2], s=0.1,c='r')
ax6.scatter(pos_gas[:,1], pos_gas[:,2], s=0.1,c='r')
ax4.scatter(pos_h[index,0], pos_h[index,1], s=30, c='b')
ax5.scatter(pos_h[index,0], pos_h[index,2], s=30, c='b')
ax6.scatter(pos_h[index,1], pos_h[index,2], s=30, c='b')
plt.show()
```



nbsphinx-code-borderwhite

Lets check if the star-formation rate of this group is equal to the star-formation rate of the gas particles on it

[12]:
```
SFR1 = np.sum(sfr_gas)
print('SFR from the gas particles = %.3e Msun/yr'%SFR1)
print('SFR of the FoF group = %.3e Msun/yr'%SFR_h[index])
```

```
SFR from the gas particles = 5.332e-02 Msun/yr
SFR of the FoF group = 5.332e-02 Msun/yr
```

In a similar way, we can read the properties of the dark matter particles in a FoF halo

[13]:
```
# the dark matter particles of this halos will start at offset[1] and will contain␣
↪lentype_h[1] particles
f = h5py.File(f_snapshot, 'r')
start = offset[1]
end   = offset[1]+lentype_h[index,1]
pos_dm = f['PartType1/Coordinates'][start:end]/1e3   #Mpc/h
vel_dm = f['PartType1/Velocities'][start:end]        #km/s
f.close()

# lets print the positions of the dark matter particles to see that they are around the␣
↪halo center
print(pos_dm)
```

```
[[ 9.327825  17.098303   3.0644984]
 [ 9.32777   17.098856   3.0653036]
 [ 9.328103  17.097187   3.065979 ]
 ...
 [ 9.272596  16.833157   3.1447148]
 [ 9.27645   16.907875   3.1141644]
 [ 9.411737  16.957142   3.074211 ]]
```

[ ]:

27.4.2 Particles inside FoF halos: SIMBA, Astrid, Magneticum

We now show to read the particles inside halos for simulations that are not IllustrisTNG.

[14]:
```
# get the name of the snapshot and its corresponding Subfind halo catalog
f_snapshot = '/home/jovyan/Data/Sims/SIMBA/1P/1P_2_4/snap_033.hdf5'
f_catalog  = '/home/jovyan/Data/FOF_Subfind/SIMBA/1P/1P_2_4/fof_subhalo_tab_033.hdf5'
```

Lets read the halo catalog

[15]:
```
f = h5py.File(f_catalog, 'r')
print(f.keys())
```

```
<KeysViewHDF5 ['Config', 'Group', 'Header', 'IDs', 'Parameters', 'Subhalo']>
```

Lets see what is in the Group and IDs blocks

[16]:
```
print(f['Group'].keys())
print(f['IDs'].keys())
```

```
<KeysViewHDF5 ['GroupBHMass', 'GroupBHMdot', 'GroupCM', 'GroupFirstSub',
↪'GroupGasMetalFractions', 'GroupGasMetallicity', 'GroupLen', 'GroupLenType', 'GroupMass
↪', 'GroupMassType', 'GroupNsubs', 'GroupPos', 'GroupSFR', 'GroupStarMetalFractions',
↪'GroupStarMetallicity', 'GroupVel', 'GroupWindMass', 'Group_M_Crit200', 'Group_M_
↪Crit500', 'Group_M_Mean200', 'Group_M_TopHat200', 'Group_R_Crit200', 'Group_R_Crit500',
↪ 'Group_R_Mean200', 'Group_R_TopHat200']>
<KeysViewHDF5 ['ID']>
```

Now, lets read the positions, velocities, masses, and lenghts of the halos. For simulations other than IllustrisTNG, we can also also read a block with the particles IDs

[17]:
```
pos_h      = f['Group/GroupPos'][:]/1e3   #Mpc/h
vel_h      = f['Group/GroupVel'][:]       #km/s
SFR_h      = f['Group/GroupSFR'][:]       #Msun/yr
mass_h     = f['Group/GroupMass'][:]*1e10 #Msun/h
len_h      = f['Group/GroupLen'][:]       #the total number of particles in the halo␣
↪(gas+dm+stars+black_holes)
lentype_h  = f['Group/GroupLenType'][:]   #the number of particles in a halo by particle␣
↪type
IDs_h      = f['IDs']['ID'][:]
f.close()
```

Lets take a look at the IDs of the particles in the halos. Lets seen that it contains as many IDs as particles in halos:

[18]:
```
print('The number of IDs is:',IDs_h.shape)
print('The total number of particles in all halos is: %d'%np.sum(len_h))
```

```
The number of IDs is: (14546093,)
The total number of particles in all halos is: 14546093
```

Lets choose a given halo and print its position, velocity, mass, and length

[19]:
```
index = 351 #index of the halo
print('position:',pos_h[index],'Mpc/h')
print('velocity:',vel_h[index],'km/s')
print('mass: %.3e Msun/h'%mass_h[index])
```

(continues on next page)

(continued from previous page)

```
print('star-formation rate: %.3e'%SFR_h[index])
print('total length:',len_h[index],'particles')
print('number of particles by type:',lentype_h[index])
```

```
position: [22.522245 14.857593  4.033678] Mpc/h
velocity: [ 76.06228  -34.665913 -98.167984] km/s
mass: 1.575e+11 Msun/h
star-formation rate: 1.196e+00
total length: 3440 particles
number of particles by type: [1036 2183    0    0  221    0]
```

We first need to get the IDs of the particles belonging to this halo with

[20]:
```
start = np.sum(len_h[:index])
end   = start+len_h[index]
indexes_h = IDs_h[start:end]
print('This halo contains %d particles'%indexes_h.shape[0])
```

```
This halo contains 3440 particles
```

indexes represent the IDs of the particles belonging to that halo. We can now read the property we want from the simulation and use those IDs to identify the particles in that halo

[21]:
```
f = h5py.File(f_snapshot, 'r')
print(f.keys())
print(f['PartType0'].keys(),'\n')
print(f['PartType1'].keys(),'\n')
print(f['PartType4'].keys(),'\n')
print(f['PartType5'].keys(),'\n')
```

```
<KeysViewHDF5 ['Header', 'PartType0', 'PartType1', 'PartType4', 'PartType5']>
<KeysViewHDF5 ['AGS-Softening', 'Coordinates', 'DelayTime', 'Density', 'Dust_Masses',
↪'Dust_Metallicity', 'ElectronAbundance', 'FractionH2', 'GrackleHI', 'GrackleHII',
↪'GrackleHM', 'GrackleHeI', 'GrackleHeII', 'GrackleHeIII', 'HaloID', 'ID_Generations',
↪'InternalEnergy', 'Masses', 'Metallicity', 'NWindLaunches', 'NeutralHydrogenAbundance',
↪ 'ParticleIDs', 'Potential', 'Sigma', 'SmoothingLength', 'StarFormationRate',
↪'Velocities']>

<KeysViewHDF5 ['AGS-Softening', 'Coordinates', 'HaloID', 'ID_Generations', 'Masses',
↪'ParticleIDs', 'Potential', 'Velocities']>

<KeysViewHDF5 ['AGS-Softening', 'Coordinates', 'Dust_Masses', 'Dust_Metallicity', 'HaloID
↪', 'ID_Generations', 'Masses', 'Metallicity', 'ParticleIDs', 'Potential',
↪'StellarFormationTime', 'Velocities']>

<KeysViewHDF5 ['AGS-Softening', 'BH_AccretionLength', 'BH_Mass', 'BH_Mass_AlphaDisk',
↪'BH_Mdot', 'BH_NProgs', 'Coordinates', 'HaloID', 'ID_Generations', 'Masses',
↪'ParticleIDs', 'Potential', 'StellarFormationTime', 'Velocities']>
```

Lets read the header of this simulation and the total number of particles on it

[22]:
```
BoxSize      = f['Header'].attrs[u'BoxSize']/1e3    #size of the snapshot in comoving␣
↪Mpc/h
```

(continues on next page)

(continued from previous page)

```
redshift     = f['Header'].attrs[u'Redshift']       #reshift of the snapshot
scale_factor = f['Header'].attrs[u'Time']           #scale factor
h            = f['Header'].attrs[u'HubbleParam']    #value of the hubble parameter in␣
↪100 km/s/(Mpc/h)
Masses       = f['Header'].attrs[u'MassTable']*1e10 #masses of the particles in Msun/h
Np           = f['Header'].attrs[u'NumPart_Total']  #total number of particles for specie
Omega_m      = f['Header'].attrs[u'Omega0']         #Omega_matter
Omega_L      = f['Header'].attrs[u'OmegaLambda']    #Omega_baryon
print('Total number of particles in the snapshot per type',Np)
```

```
Total number of particles in the snapshot per type [15783459 16777216        0        0 ␣
↪ 989418      976]
```

Lets read the positions and velocities of the particles belonging to this halos

[23]:
```
pos_gas   = f['PartType0/Coordinates'][:]/1e3   #Mpc/h
pos_dm    = f['PartType1/Coordinates'][:]/1e3   #Mpc/h
pos_stars = f['PartType4/Coordinates'][:]/1e3   #Mpc/h
pos_bh    = f['PartType5/Coordinates'][:]/1e3   #Mpc/h

vel_gas   = f['PartType0/Velocities'][:]        #km/s
vel_dm    = f['PartType1/Velocities'][:]        #km/s
vel_stars = f['PartType4/Velocities'][:]        #km/s
vel_bh    = f['PartType5/Velocities'][:]        #km/s

IDs_gas   = f['PartType0/ParticleIDs'][:]
IDs_dm    = f['PartType1/ParticleIDs'][:]
IDs_stars = f['PartType4/ParticleIDs'][:]
IDs_bh    = f['PartType5/ParticleIDs'][:]

f.close()
```

Now we want to match the IDs of the particles belonging the halo that we have stored in the indexes variable with the IDs of the different particle types from the snapshot. Lets first do this for the gas particles

[24]:
```
# This routine will find the common IDs and also return their location in each array
common_indexes, indexes1, indexes2 = np.intersect1d(IDs_gas, indexes_h, assume_
↪unique=False, return_indices=True)
pos_gas_halo = pos_gas[indexes1]
vel_gas_halo = vel_gas[indexes1]
```

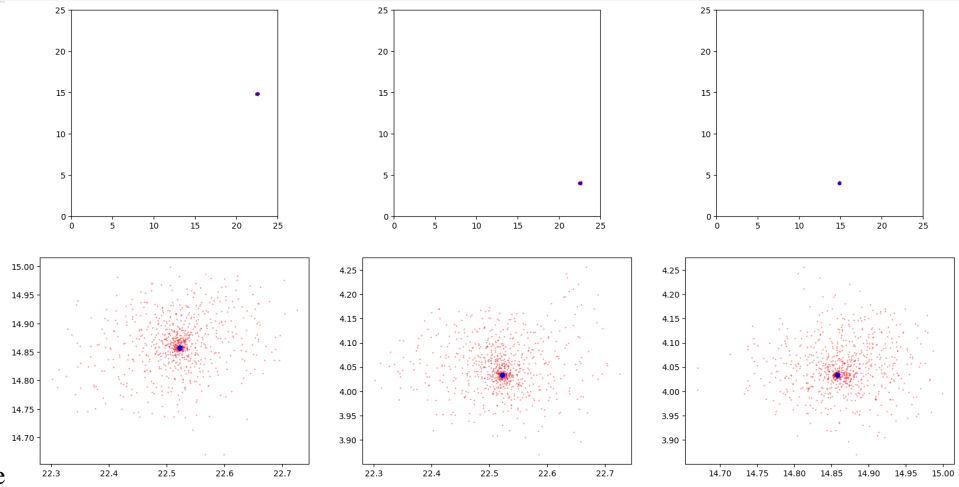
Lets make a plot with the positions of the particles to see where they are

[25]:
```
import matplotlib.pyplot as plt
from pylab import *
fig = figure(figsize=(20,10))
ax1 = fig.add_subplot(231)
ax2 = fig.add_subplot(232)
ax3 = fig.add_subplot(233)
ax4 = fig.add_subplot(234)
ax5 = fig.add_subplot(235)
ax6 = fig.add_subplot(236)
for ax in [ax1,ax2,ax3]:
```

(continues on next page)

(continued from previous page)

```
    ax.set_aspect('equal')
    ax.set_xlim([0,25])
    ax.set_ylim([0,25])
ax1.scatter(pos_gas_halo[:,0], pos_gas_halo[:,1], s=0.1,c='r')
ax2.scatter(pos_gas_halo[:,0], pos_gas_halo[:,2], s=0.1,c='r')
ax3.scatter(pos_gas_halo[:,1], pos_gas_halo[:,2], s=0.1,c='r')
ax1.scatter(pos_h[index,0], pos_h[index,1], s=10, c='b')
ax2.scatter(pos_h[index,0], pos_h[index,2], s=10, c='b')
ax3.scatter(pos_h[index,1], pos_h[index,2], s=10, c='b')
# now make a zoom-in
ax4.scatter(pos_gas_halo[:,0], pos_gas_halo[:,1], s=0.1,c='r')
ax5.scatter(pos_gas_halo[:,0], pos_gas_halo[:,2], s=0.1,c='r')
ax6.scatter(pos_gas_halo[:,1], pos_gas_halo[:,2], s=0.1,c='r')
ax4.scatter(pos_h[index,0], pos_h[index,1], s=30, c='b')
ax5.scatter(pos_h[index,0], pos_h[index,2], s=30, c='b')
ax6.scatter(pos_h[index,1], pos_h[index,2], s=30, c='b')
plt.show()
```



nbsphinx-code-borderwhite

Lets check that the star-formation rate of this halo is equal to the star-formation rate of the gas particles on it

[26]:
```
# lets read the SFR of the gas particles in the snapshot
f = h5py.File(f_snapshot, 'r')
sfr_gas = f['PartType0/StarFormationRate'][:] #Msun/yr
IDs_gas = f['PartType0/ParticleIDs'][:]
f.close()

common_indexes, indexes1, indexes2 = np.intersect1d(IDs_gas, indexes_h, assume_
↪unique=False, return_indices=True)
SFR = sfr_gas[indexes1]

# lets check the total star-formation rates
print('Total star-formation rate from the gas particles in the halo: %.3e'%np.sum(SFR))
print('Total star-formation rate from the halo catalog: %.3e'%SFR_h[index])
```

```
Total star-formation rate from the gas particles in the halo: 1.196e+00
Total star-formation rate from the halo catalog: 1.196e+00
```

[ ]:

27.4.3 Particles in a subhalo/galaxy: IllustrisTNG

[27]:
```
# get the name of the snapshot and its corresponding Subfind catalog
f_snapshot = '/home/jovyan/Data/Sims/IllustrisTNG/LH/LH_50/snap_033.hdf5'
f_catalog  = '/home/jovyan/Data/FOF_Subfind/IllustrisTNG/LH/LH_50/fof_subhalo_tab_033.
↪hdf5'
```

Lets start reading the halo/subhalo catalog

[28]:
```
f = h5py.File(f_catalog, 'r')
print(f.keys())
```

```
<KeysViewHDF5 ['Config', 'Group', 'Header', 'IDs', 'Parameters', 'Subhalo']>
```

We need to read data from both the Group and the Subhalo blocks

[29]:
```
print(f['Group'].keys(),'\n')
print(f['Subhalo'].keys())
```

```
<KeysViewHDF5 ['GroupBHMass', 'GroupBHMdot', 'GroupCM', 'GroupFirstSub',
↪'GroupGasMetalFractions', 'GroupGasMetallicity', 'GroupLen', 'GroupLenType', 'GroupMass
↪', 'GroupMassType', 'GroupNsubs', 'GroupPos', 'GroupSFR', 'GroupStarMetalFractions',
↪'GroupStarMetallicity', 'GroupVel', 'GroupWindMass', 'Group_M_Crit200', 'Group_M_
↪Crit500', 'Group_M_Mean200', 'Group_M_TopHat200', 'Group_R_Crit200', 'Group_R_Crit500',
↪ 'Group_R_Mean200', 'Group_R_TopHat200']>

<KeysViewHDF5 ['SubhaloBHMass', 'SubhaloBHMdot', 'SubhaloBfldDisk', 'SubhaloBfldHalo',
↪'SubhaloCM', 'SubhaloGasMetalFractions', 'SubhaloGasMetalFractionsHalfRad',
↪'SubhaloGasMetalFractionsMaxRad', 'SubhaloGasMetalFractionsSfr',
↪'SubhaloGasMetalFractionsSfrWeighted', 'SubhaloGasMetallicity',
↪'SubhaloGasMetallicityHalfRad', 'SubhaloGasMetallicityMaxRad',
↪'SubhaloGasMetallicitySfr', 'SubhaloGasMetallicitySfrWeighted', 'SubhaloGrNr',
↪'SubhaloHalfmassRad', 'SubhaloHalfmassRadType', 'SubhaloIDMostbound', 'SubhaloLen',
↪'SubhaloLenType', 'SubhaloMass', 'SubhaloMassInHalfRad', 'SubhaloMassInHalfRadType',
↪'SubhaloMassInMaxRad', 'SubhaloMassInMaxRadType', 'SubhaloMassInRad',
↪'SubhaloMassInRadType', 'SubhaloMassType', 'SubhaloParent', 'SubhaloPos', 'SubhaloSFR',
↪ 'SubhaloSFRinHalfRad', 'SubhaloSFRinMaxRad', 'SubhaloSFRinRad', 'SubhaloSpin',
↪'SubhaloStarMetalFractions', 'SubhaloStarMetalFractionsHalfRad',
↪'SubhaloStarMetalFractionsMaxRad', 'SubhaloStarMetallicity',
↪'SubhaloStarMetallicityHalfRad', 'SubhaloStarMetallicityMaxRad',
↪'SubhaloStellarPhotometrics', 'SubhaloStellarPhotometricsMassInRad',
↪'SubhaloStellarPhotometricsRad', 'SubhaloVel', 'SubhaloVelDisp', 'SubhaloVmax',
↪'SubhaloVmaxRad', 'SubhaloWindMass']>
```

[30]:
```
# lets read the length of the FoF groups and the number of subhalos they contain
lentype_h = f['Group/GroupLenType'][:]
Nsub_h = f['Group/GroupNsubs'][:]

# for the subhalos lets read their position, mass, velocity, and SFR
pos_sh     = f['Subhalo/SubhaloPos'][:]/1e3        #Mpc/h
```

(continues on next page)

(continued from previous page)

```
vel_sh     = f['Subhalo/SubhaloVel'][:]            #km/s
mass_sh    = f['Subhalo/SubhaloMassType'][:]*1e10  #Msun/h
SFR_sh     = f['Subhalo/SubhaloSFR'][:]            #Msun/yr
lentype_sh = f['Subhalo/SubhaloLenType'][:]
index_h_sh = f['Subhalo/SubhaloGrNr']
```

Now lets take a given subhalo and prints its properties

[31]:
```
index_sh = 972
print('position:',pos_sh[index_sh],'Mpc/h')
print('velocity:',vel_sh[index_sh],'km/s')
print('mass:',mass_sh[index_sh],'Msun/h')
print('total length:',len_h[index_sh],'particles')
print('number of particles by type:',lentype_sh[index_sh])
```

```
position: [22.246033  16.84598    3.4119482] Mpc/h
velocity: [ 582.90076   -35.248653 -111.824135] km/s
mass: [4.0012946e+09 6.6489463e+10 0.0000000e+00 0.0000000e+00 8.6655034e+08
 1.8247212e+07] Msun/h
total length: 878 particles
number of particles by type: [265 814   0   0  89   1]
```

First, we need to know which halo this subhalo/galaxy belongs to, and we can get that with

[32]:
```
index_h = index_h_sh[index_sh] #index of the halos where this subhalo belongs to
print('This subhalo belongs to group %d'%index_h)
```

```
This subhalo belongs to group 5
```

Now we need to know how many subhalos/galaxies are before this one in the catalog. For that, lets sum all the subhalos in the halos that are before the halo this subhalo belong to

[33]:
```
Nsub_prev_groups = np.sum(Nsub_h[:index_h]) #This is the number of subhalos that the␣
↪preceeding halos
print('The first %d halos contain %d subhalos'%(index_h,Nsub_prev_groups))

# The number of subhalos that preceed our subhalo in the halo it belongs to is
preceeding_subhalos_in_halo = index_sh - Nsub_prev_groups
print('There are %d subhalos that preceed our subhalo in the halo it belongs to'
↪%preceeding_subhalos_in_halo)
```

```
The first 5 halos contain 970 subhalos
There are 2 subhalos that preceed our subhalo in the halo it belongs to
```

Now we can finally compute the start and end of the indexes of the particles

[34]:
```
start = np.sum(lentype_h[:index_h],axis=0) + np.sum(lentype_sh[Nsub_prev_groups:index_
↪sh],axis=0)
end = start + lentype_sh[index_sh]
print(start)
print(end)
```

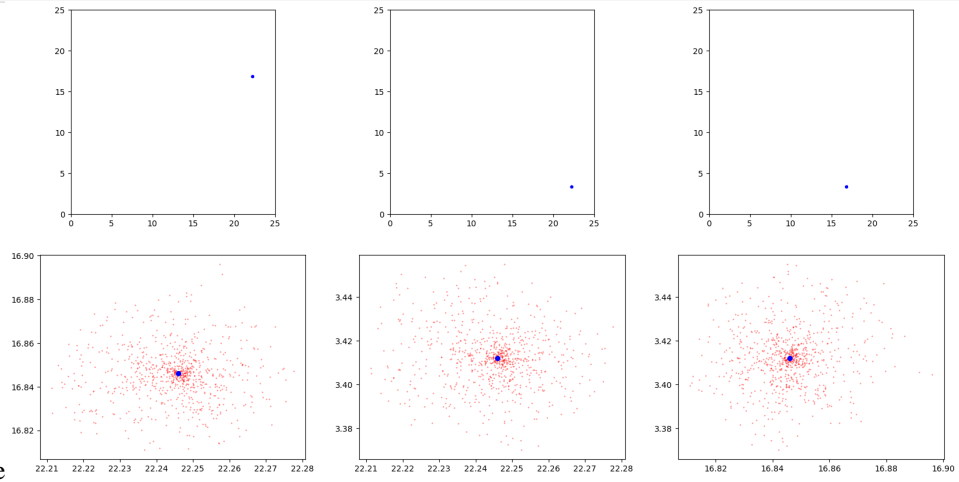
```
[1237429 2179886       0       0  210331     118]
[1237694 2180700       0       0  210420     119]
```

Now, lets read the positions and velocities of the dark matter of this subhalo/galaxy

[35]:
```
f = h5py.File(f_snapshot,'r')
pos_dm = f['PartType1/Coordinates'][start[1]:end[1]]/1e3 #Mpc/h
vel_dm = f['PartType1/Velocities'][start[1]:end[1]]      #km/s
f.close()
```

Lets plot the positions of the dark matter particles and also the position of the subhalo center

[36]:
```
import matplotlib.pyplot as plt
from pylab import *
fig = figure(figsize=(20,10))
ax1 = fig.add_subplot(231)
ax2 = fig.add_subplot(232)
ax3 = fig.add_subplot(233)
ax4 = fig.add_subplot(234)
ax5 = fig.add_subplot(235)
ax6 = fig.add_subplot(236)
for ax in [ax1,ax2,ax3]:
    ax.set_aspect('equal')
    ax.set_xlim([0,25])
    ax.set_ylim([0,25])
ax1.scatter(pos_dm[:,0], pos_dm[:,1], s=0.1,c='r')
ax2.scatter(pos_dm[:,0], pos_dm[:,2], s=0.1,c='r')
ax3.scatter(pos_dm[:,1], pos_dm[:,2], s=0.1,c='r')
ax1.scatter(pos_sh[index_sh,0], pos_sh[index_sh,1], s=10, c='b')
ax2.scatter(pos_sh[index_sh,0], pos_sh[index_sh,2], s=10, c='b')
ax3.scatter(pos_sh[index_sh,1], pos_sh[index_sh,2], s=10, c='b')
# now make a zoom-in
ax4.scatter(pos_dm[:,0], pos_dm[:,1], s=0.1,c='r')
ax5.scatter(pos_dm[:,0], pos_dm[:,2], s=0.1,c='r')
ax6.scatter(pos_dm[:,1], pos_dm[:,2], s=0.1,c='r')
ax4.scatter(pos_sh[index_sh,0], pos_sh[index_sh,1], s=30, c='b')
ax5.scatter(pos_sh[index_sh,0], pos_sh[index_sh,2], s=30, c='b')
ax6.scatter(pos_sh[index_sh,1], pos_sh[index_sh,2], s=30, c='b')
plt.show()
```



nbsphinx-code-borderwhite

Lets check that the stellar mass of this subhalo is equal to the sum of the masses of its stars

[37]:
```
print('This subhalo has a stellar mass of %.3e Msun/h'%mass_sh[index_sh,4])
```

(continues on next page)

(continued from previous page)

```
# now select the masses of the stars belonging to this subhalo/galaxy
f = h5py.File(f_snapshot,'r')
masses = f['PartType4/Masses'][start[4]:end[4]]*1e10 #Msun/h
f.close()
print('The sum of the masses of the stars of this subhalo is %.3e Msun/h'%np.sum(masses))
```

```
This subhalo has a stellar mass of 8.666e+08 Msun/h
The sum of the masses of the stars of this subhalo is 8.666e+08 Msun/h
```

In general, any field contained in the snapshot, can be read for the particles that belong to this subhalo as field = f['PartTypeX/property'][start[X]:end[X]]

[ ]:

27.4.4 Particles in a subhalo/galaxy: SIMBA, Astrid, Magneticum

We now show to read the particles inside subhalos/galaxies for simulations that are not IllustrisTNG.

[38]:
```
# get the name of the snapshot and its corresponding Subfind catalog
f_snapshot = '/home/jovyan/Data/Sims/Astrid/LH/LH_401/snap_090.hdf5'
f_catalog  = '/home/jovyan/Data/FOF_Subfind/Astrid/LH/LH_401/fof_subhalo_tab_090.hdf5'
```

Lets start reading the halo/subhalo catalog

[39]:
```
f = h5py.File(f_catalog, 'r')
print(f.keys())
print(f['IDs'].keys())
```

```
<KeysViewHDF5 ['Config', 'Group', 'Header', 'IDs', 'Parameters', 'Subhalo']>
<KeysViewHDF5 ['ID']>
```

We need to read data from both the Group and the Subhalo blocks

[40]:
```
print(f['Group'].keys(),'\n')
print(f['Subhalo'].keys(),'\n')
print(f['IDs'].keys())
```

```
<KeysViewHDF5 ['GroupBHMass', 'GroupBHMdot', 'GroupCM', 'GroupFirstSub',
↪'GroupGasMetalFractions', 'GroupGasMetallicity', 'GroupLen', 'GroupLenType', 'GroupMass
↪', 'GroupMassType', 'GroupNsubs', 'GroupPos', 'GroupSFR', 'GroupStarMetalFractions',
↪'GroupStarMetallicity', 'GroupVel', 'GroupWindMass', 'Group_M_Crit200', 'Group_M_
↪Crit500', 'Group_M_Mean200', 'Group_M_TopHat200', 'Group_R_Crit200', 'Group_R_Crit500',
↪ 'Group_R_Mean200', 'Group_R_TopHat200']>

<KeysViewHDF5 ['SubhaloBHMass', 'SubhaloBHMdot', 'SubhaloBfldDisk', 'SubhaloBfldHalo',
↪'SubhaloCM', 'SubhaloGasMetalFractions', 'SubhaloGasMetalFractionsHalfRad',
↪'SubhaloGasMetalFractionsMaxRad', 'SubhaloGasMetalFractionsSfr',
↪'SubhaloGasMetalFractionsSfrWeighted', 'SubhaloGasMetallicity',
↪'SubhaloGasMetallicityHalfRad', 'SubhaloGasMetallicityMaxRad',
↪'SubhaloGasMetallicitySfr', 'SubhaloGasMetallicitySfrWeighted', 'SubhaloGrNr',
↪'SubhaloHalfmassRad', 'SubhaloHalfmassRadType', 'SubhaloIDMostbound', 'SubhaloLen',
↪'SubhaloLenType', 'SubhaloMass', 'SubhaloMassInHalfRad', 'SubhaloMassInHalfRadType',
```

(continues on next page)

(continued from previous page)

```
↪'SubhaloMassInMaxRad', 'SubhaloMassInMaxRadType', 'SubhaloMassInRad',
↪'SubhaloMassInRadType', 'SubhaloMassType', 'SubhaloParent', 'SubhaloPos', 'SubhaloSFR',
↪ 'SubhaloSFRinHalfRad', 'SubhaloSFRinMaxRad', 'SubhaloSFRinRad', 'SubhaloSpin',
↪'SubhaloStarMetalFractions', 'SubhaloStarMetalFractionsHalfRad',
↪'SubhaloStarMetalFractionsMaxRad', 'SubhaloStarMetallicity',
↪'SubhaloStarMetallicityHalfRad', 'SubhaloStarMetallicityMaxRad',
↪'SubhaloStellarPhotometrics', 'SubhaloStellarPhotometricsMassInRad',
↪'SubhaloStellarPhotometricsRad', 'SubhaloVel', 'SubhaloVelDisp', 'SubhaloVmax',
↪'SubhaloVmaxRad', 'SubhaloWindMass']>

<KeysViewHDF5 ['ID']>
```

[41]:
```
# lets read the length of the FoF groups and the number of subhalos they contain
len_h  = f['Group/GroupLen'][:]
Nsub_h = f['Group/GroupNsubs'][:]

# for the subhalos lets read their position, mass, velocity, and SFR
pos_sh     = f['Subhalo/SubhaloPos'][:]/1e3        #Mpc/h
vel_sh     = f['Subhalo/SubhaloVel'][:]            #km/s
mass_sh    = f['Subhalo/SubhaloMassType'][:]*1e10  #Msun/h
SFR_sh     = f['Subhalo/SubhaloSFR'][:]            #Msun/yr
len_sh     = f['Subhalo/SubhaloLen'][:]
lentype_sh = f['Subhalo/SubhaloLenType'][:]
index_h_sh = f['Subhalo/SubhaloGrNr']

# lets also read the IDs block
IDs_sh = f['IDs/ID'][:]
```

Now lets take a given subhalo and print its properties

[42]:
```
index_sh = 589
print('position:',pos_sh[index_sh],'Mpc/h')
print('velocity:',vel_sh[index_sh],'km/s')
print('mass:',mass_sh[index_sh],'Msun/h')
print('total length:',len_sh[index_sh],'particles')
print('number of particles by type:',lentype_sh[index_sh])
```

```
position: [ 3.00393  14.963539 22.125805] Mpc/h
velocity: [-98.59593  -42.708527 332.98932 ] km/s
mass: [1.68489472e+08 1.12988455e+11 0.00000000e+00 0.00000000e+00
 3.46133024e+08 0.00000000e+00] Msun/h
total length: 1478 particles
number of particles by type: [  14 1170    0    0  294    0]
```

First, we need to know which halo this subhalo/galaxy belongs to, and we can get that with

[43]:
```
index_h = index_h_sh[index_sh] #index of the halos where this subhalo belongs to
print('This subhalo belongs to group %d'%index_h)
```

```
This subhalo belongs to group 3
```

Now we need to know how many subhalos/galaxies are before this one in the catalog. For that, lets sum all the subhalos in the halos that are before the halo this subhalo belong to

[44]:
```
Nsub_prev_groups = np.sum(Nsub_h[:index_h]) #This is the number of subhalos that the␣
↪preceeding halos
print('The first %d halos contain %d subhalos'%(index_h,Nsub_prev_groups))

# The number of subhalos that preceed our subhalo in the halo it belongs to is
preceeding_subhalos_in_halo = index_sh - Nsub_prev_groups
print('There are %d subhalos that preceed our subhalo in the halo it belongs to'
↪%preceeding_subhalos_in_halo)
```

```
The first 3 halos contain 584 subhalos
There are 5 subhalos that preceed our subhalo in the halo it belongs to
```

Now we can finally compute the start and end of the indexes of the particles

[45]:
```
start = np.sum(len_h[:index_h]) + np.sum(len_sh[Nsub_prev_groups:index_sh])
end = start + len_sh[index_sh]
print(start)
print(end)
```

```
2925850
2927328
```

We can get the IDs of the particles that belong to this subhalo/galaxy as this

[46]:
```
indexes = IDs_sh[start:end]
print(indexes)
```

```
[         20338487          20367737          20409329 ...
 72057594058295854 72057594058300274 72057594058283564]
```

As can be seen, differently to the SIMBA simulations, the IDs of Astrid can be very large. For dealing with this we will need to explictly match IDs. Lets now read the snapshot and some properties of it

[47]:
```
f = h5py.File(f_snapshot, 'r')
print(f.keys())
print(f['PartType0'].keys(),'\n')
print(f['PartType1'].keys(),'\n')
print(f['PartType4'].keys(),'\n')
print(f['PartType5'].keys(),'\n')
```

```
<KeysViewHDF5 ['Header', 'PartType0', 'PartType1', 'PartType4', 'PartType5']>
<KeysViewHDF5 ['Coordinates', 'Density', 'ElectronAbundance', 'GFM_Metallicity', 'GFM_
↪Metals', 'InternalEnergy', 'Masses', 'NeutralHydrogenAbundance', 'ParticleIDs',
↪'SmoothingLength', 'StarFormationRate', 'Velocities']>

<KeysViewHDF5 ['Coordinates', 'Masses', 'ParticleIDs', 'Velocities']>

<KeysViewHDF5 ['Coordinates', 'GFM_Metallicity', 'GFM_Metals', 'GFM_StellarFormationTime
↪', 'Masses', 'ParticleIDs', 'Velocities']>

<KeysViewHDF5 ['BH_Density', 'BH_Mass', 'BH_Mdot', 'Coordinates', 'GFM_
↪StellarFormationTime', 'Masses', 'ParticleIDs', 'SmoothingLength', 'Velocities']>
```

Lets read the positions and masses of the gas, dark matter, and stars in the snapshot

[48]:
```
pos_gas    = f['PartType0/Coordinates'][:]/1e3   #Mpc/h
pos_dm     = f['PartType1/Coordinates'][:]/1e3   #Mpc/h
pos_stars  = f['PartType4/Coordinates'][:]/1e3   #Mpc/h
mass_gas   = f['PartType0/Masses'][:]*1e10       #Msun/h
mass_dm    = f['PartType1/Masses'][:]*1e10       #Msun/h
mass_stars = f['PartType4/Masses'][:]*1e10       #Msun/h
IDs_gas    = f['PartType0/ParticleIDs'][:]
IDs_dm     = f['PartType1/ParticleIDs'][:]
IDs_stars  = f['PartType4/ParticleIDs'][:]
f.close()
```

Now lets see the locations where the IDs of the particles belonging to the subhalo match those of the gas and star particles in the snapshot

[49]:
```
common_indexes, indexes1, indexes2 = np.intersect1d(IDs_dm, indexes, assume_unique=False,
↪ return_indices=True)
print('The IDs that are in both the gas and all particles in the subhalo are', common_
↪indexes)
print('The indexes of the common IDs in the gas array', indexes1)
```

```
The IDs that are in both the gas and all particles in the subhalo are [1570675 2009055␣
↪2017184 ... 3632115 4064806 4122391]
The indexes of the common IDs in the gas array [15181907 15179506 15179189 ... 15179209␣
↪15179808 15181651]
```

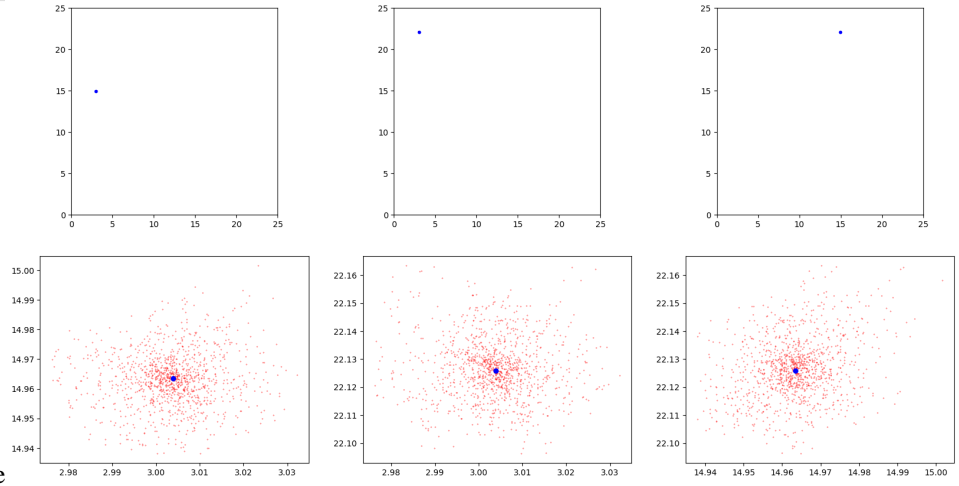
Lets plot the positions of the gas particles belonging to this subhalo/galaxy together with the subhalo center

[50]:
```
pos_dm_subhalo = pos_dm[indexes1]

import matplotlib.pyplot as plt
from pylab import *
fig = figure(figsize=(20,10))
ax1 = fig.add_subplot(231)
ax2 = fig.add_subplot(232)
ax3 = fig.add_subplot(233)
ax4 = fig.add_subplot(234)
ax5 = fig.add_subplot(235)
ax6 = fig.add_subplot(236)
for ax in [ax1,ax2,ax3]:
    ax.set_aspect('equal')
    ax.set_xlim([0,25])
    ax.set_ylim([0,25])
ax1.scatter(pos_dm_subhalo[:,0], pos_dm_subhalo[:,1], s=0.1,c='r')
ax2.scatter(pos_dm_subhalo[:,0], pos_dm_subhalo[:,2], s=0.1,c='r')
ax3.scatter(pos_dm_subhalo[:,1], pos_dm_subhalo[:,2], s=0.1,c='r')
ax1.scatter(pos_sh[index_sh,0], pos_sh[index_sh,1], s=10, c='b')
ax2.scatter(pos_sh[index_sh,0], pos_sh[index_sh,2], s=10, c='b')
ax3.scatter(pos_sh[index_sh,1], pos_sh[index_sh,2], s=10, c='b')
# now make a zoom-in
ax4.scatter(pos_dm_subhalo[:,0], pos_dm_subhalo[:,1], s=0.1,c='r')
ax5.scatter(pos_dm_subhalo[:,0], pos_dm_subhalo[:,2], s=0.1,c='r')
ax6.scatter(pos_dm_subhalo[:,1], pos_dm_subhalo[:,2], s=0.1,c='r')
ax4.scatter(pos_sh[index_sh,0], pos_sh[index_sh,1], s=30, c='b')
ax5.scatter(pos_sh[index_sh,0], pos_sh[index_sh,2], s=30, c='b')
```

(continues on next page)

(continued from previous page)

```
ax6.scatter(pos_sh[index_sh,1], pos_sh[index_sh,2], s=30, c='b')
plt.show()
```



nbsphinx-code-borderwhite

Lets now check that the stellar mass of this galaxy is the same as the sum of the masses of its stars

[51]:
```
common_indexes, indexes1, indexes2 = np.intersect1d(IDs_stars, indexes, assume_
↪unique=False, return_indices=True)
SM = mass_stars[indexes1]
print('Sum of the masses of the stars of this galaxy: %.3e'%np.sum(SM))
print('Stellar mass of the galaxy: %.3e'%mass_sh[index_sh,4])
```

```
Sum of the masses of the stars of this galaxy: 3.461e+08
Stellar mass of the galaxy: 3.461e+08
```

In general, given a property that wants to be known for the particles in a subhalo, one needs to read it together with the particle ids of that type. E.g. to read the metallicities of the gas particles in a subhalo/galaxy, one needs to read all gas particle metallicities and their IDs. Then, matching IDs between the particles in the halo and the read particles can be done using the above intersect1d routine. Finally, one can just take the value of the considered property in the particles belonging to the subhalo as:

property = property_read_from_snapshot[indexes1]

[ ]:
```
```

CHAPTER

TWENTYEIGHT

POST-PROCESSING: IMAGES

There are two different ways to create images from the simulations:

28.1 Column density

The first option is to create images by computing the column density along the center of each pixel. The next script computes the column density of the gas temperature as

$$\bar{T}(x,y) = \frac{\int m(x,y,z)T(x,y,z)dz}{\int m(x,y,z)dz}$$

where $m(x,y,z)$ and $T(x,y,z)$ are the gas mass and temperature at position (x,y,z).

```
import numpy as np
import MAS_library as MASL
import camels_library as CL
import h5py

####################################### INPUT ##########################################
# input and output files
snapshot = '/mnt/ceph/users/camels/Sims/IllustrisTNG/LH_0/snap_033.hdf5'
f_out    = 'gas_temperature.npy'

# region over which make the image (should be squared)
x_min, x_max = 0.0, 25.0 #Mpc/h
y_min, y_max = 0.0, 25.0 #Mpc/h
z_min, z_max = 0.0, 5.0 #Mpc/h
grid         = 250      #image will have grid x grid pixels

# parameters to compute column density
plane    = 'XY'  #plane to project the region: 'XY', 'YZ', 'XZ'
periodic = True  #whether treat image as periodic in the considered plane

# KDTree parameters
k       = 32 #number of neighborghs
threads = -1
########################################################################################

# read gas position and masses
f       = h5py.File(snapshot, 'r')
```

(continues on next page)

(continued from previous page)

```
BoxSize  = f['Header'].attrs[u'BoxSize']/1e3  #Mpc/h
redshift = f['Header'].attrs[u'Redshift']
pos_g    = f['PartType0/Coordinates'][:]/1e3  #Mpc/h
pos_g    = pos_g.astype(np.float32)           #positions as float32
Mg       = f['PartType0/Masses'][:]*1e10      #Msun/h
f.close()
T        = CL.temperature(snapshot)           #K
Rg       = CL.KDTree_distance(pos_g, pos_g, k, BoxSize*(1.0+1e-8), threads,␣
↪verbose=False) #Mpc/h
Rg       = Rg.astype(np.float32)              #radii as float32

# select the particles in the considered region
indexes = np.where((pos_g[:,0]>x_min) & (pos_g[:,0]<x_max) &
                   (pos_g[:,1]>y_min) & (pos_g[:,1]<y_max) &
                   (pos_g[:,2]>z_min) & (pos_g[:,2]<z_max))[0]
pos_g_ = pos_g[indexes]
T_     = T[indexes]
Mg_    = Mg[indexes]
Rg_    = Rg[indexes]

if   plane=='XY':  axis_x, axis_y, width = 0, 1, x_max-x_min
elif plane=='YZ':  axis_x, axis_y, width = 1, 2, y_max-y_min
elif plane=='XZ':  axis_x, axis_y, width = 0, 2, z_max-z_min

# project gas mass*temperatures into a 2D map
TM = np.zeros((grid,grid), dtype=np.float64)
MASL.voronoi_RT_2D(TM, pos_g_, T_*Mg_, Rg_, x_min, y_min,
                   axis_x, axis_y, width, periodic, verbose=True)

# project gas mass into a 2D map
M = np.zeros((grid,grid), dtype=np.float64)
MASL.voronoi_RT_2D(M, pos_g_, Mg_, Rg_, x_min, y_min,
                   axis_x, axis_y, width, periodic, verbose=True)

# compute mean temperature
T = TM/M
print('%.3e < T < %.3e'%(np.min(T), np.max(T)))

# save image to file
np.save(f_out, T)
```

The image can be plotted with something like this:

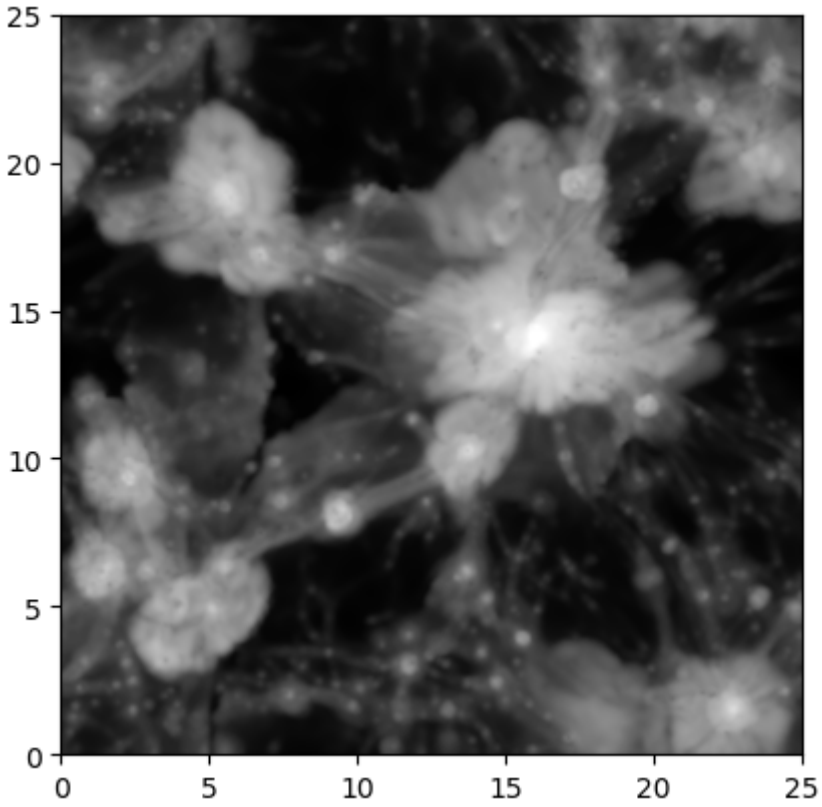
```
import numpy as np
from pylab import *
from matplotlib.colors import LogNorm

image = np.load('gas_temperature.npy')
f_out = 'gas_temperature.png'

fig = figure()
ax1 = fig.add_subplot(111)
```

(continues on next page)

(continued from previous page)

```
ax1.imshow(image.T, cmap=get_cmap('binary_r'), origin='lower', interpolation='bicubic',
           extent=[0,25,0,25], norm = LogNorm(vmin=2e3,vmax=1e7))
savefig(f_out, bbox_inches='tight')
close(fig)
```

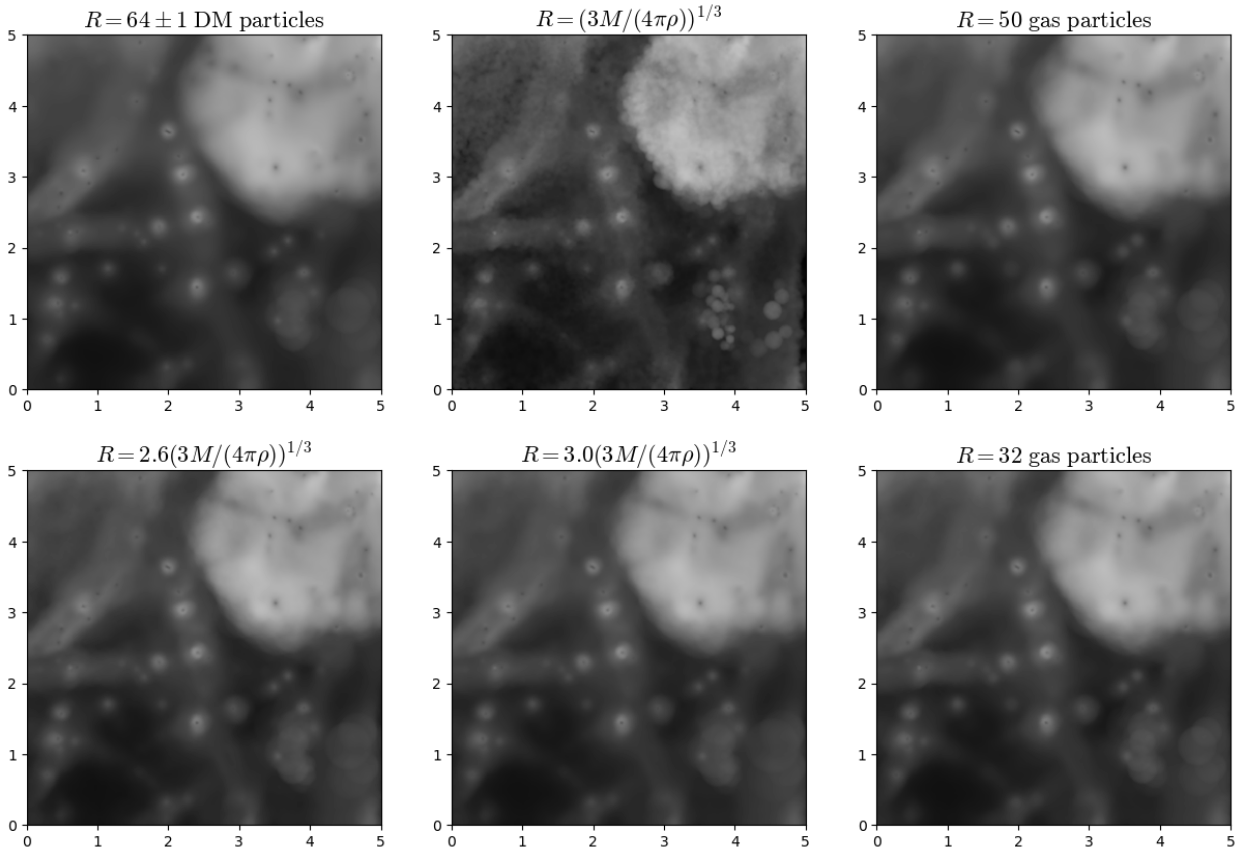
Producing this image:



The most important thing is to define the radius of the gas particles. There are multiple possibilities for this:

- For IllustrisTNG, each gas particle can be approximated as sphere with the same volume as the one of its voronoi cell, i.e., $R = (3M/(4\pi\rho))^{1/3}$.
- The above radius definition can be expanded by an overll factor to avoid empty regions.
- The radius can be considered as the distance to the k nearest gas particle.

The below image shows how different definitions led to different results:



We recommend using as radius of a gas particle the distance to its 32th nearest gas particle.

28.2 3D fields slices

CHAPTER

TWENTYNINE

CAMELS LIBRARY

The CAMELS library is a set python routines written to simplify the analysis of CAMELS data.

29.1 Installation

For the stable version do:

```
python -m pip install CAMELS-library
```

For the development version do:

```
git clone https://github.com/franciscovillaescusa/CAMELS
cd CAMELS
python -m pip install .
```

Note: The CAMELS library is already installed in our CAMELS binder.

29.2 Routines

29.2.1 Distance to k nearest neighbors

The routine `KDTree_distance` can be used to compute the distance to the k nearest neighbors of a set of particles. Note that the neighbors can be a different type of particle as the one considered. For instance, it can be used to compute the distance to the k nearest gas particles from the positions of star particles. The ingredients needed for this routine are:

- `pos1`. The positions of the particles over which compute its nearest neighbors. In the above example, these would be the positions of the gas particles.
- `pos2`. The positions of the particles over which compute the distances. In the above example, these would be the positions of the start particles. Note that `pos2` can be the same as `pos1`: e.g. to compute the distance to the k nearest neighborgs from a set of dark matter particles.
- `k`. The number of neighbors to consider.

- `BoxSize`. To account for periodic boundary conditions, set this number to the size of the simulation box. Note that this number may need to be just slightly larger than that to avoid problems with particles in the edge. E.g. set it to `BoxSize*(1.0+1e-8)`.
- `threads`. Number of openmp threads to use in the calculation. Set to `-1` to use all the available threads.
- `verbose`. Whether to print some information on the progress.

An example of how to use this routine is this

```
import numpy as np
import camels_library as CL
import h5py

################## INPUT ##################
# snapshot name
snapshot = '/mnt/ceph/users/camels/Sims/IllustrisTNG/LH_0/snap_033.hdf5'

# KDTree parameters
k       = 32    #number of neighbors
threads = -1    #number of openmp threads
verbose = True  #whether print some information on calculation progress
###########################################

# read positions of gas and star particles
f         = h5py.File(snapshot, 'r')
BoxSize   = f['Header'].attrs[u'BoxSize']/1e3  #Mpc/h
pos_gas   = f['PartType0/Coordinates'][:]/1e3  #Mpc/h
pos_stars = f['PartType4/Coordinates'][:]/1e3  #Mpc/h
f.close()

# compute distance of each star particle to its k nearest gas particle
# d is a 1D numpy array with the distance of each star particle to its
# k nearest neighborghs
d = CL.KDTree_distance(pos_gas, pos_stars, k, BoxSize*(1.0+1e-8), threads, verbose) #Mpc/
↪h
```

29.2.2 Gas temperature

The routine `temperature` can be used to compute the temperature of the gas particles in a snapshot. One example is this:

```
import numpy as np
import CAMELS_library as CL

# snapshot name
snapshot = '/mnt/ceph/users/camels/Sims/SIMBA/1P_5/snap_033.hdf5'

# get gas temperature in Kelvin
T = CL.temperature(snapshot)
```

29.2.3 Gas pressure

The routine `pressure` returns the gas pressure of the gas particle of a given snapshot, in units of $(M_\odot/h)(\mathrm{km/s})^2/(\mathrm{kpc}/h)^3$

```
import numpy as np
import CAMELS_library as CL

# snapshot name
snapshot = '/mnt/ceph/users/camels/Sims/SIMBA/CV_12/snap_020.hdf5'

# compute gas pressure in unit of (Msun/h)*(km/s)^2/(kpc/h)^3
P = CL.pressure(snapshot)
```

29.2.4 Electron density

The routine `electron_density` computes the electron number density of the gas particles of a snapshot. This routine assumes that star-forming particles are fully neutral and therefore their electron number density is equal to 0. The units of the output are $h^2\mathrm{cm}^{-3}$. Its usage is as follows:

```
import numpy as np
import CAMELS_library as CL

# snapshot name
snapshot = '/mnt/ceph/users/camels/Sims/IllustrisTNG/EX_0/snap_030.hdf5'

# compute electron number density in 1e20 electrons*h^2/cm^3 units
n_e = CL.electron_density(snapshot)
```

CHAPTER

THIRTY

PYLIANS3

Many analyses of the CAMELS simulations have been carried out using the Pylians3 libraries. Furthermore, the *CAMELS library* requires Pylians3, so we encourage its usage to analyze CAMELS data.

30.1 Installation

For the stable version do:

```
python-m pip install Pylians
```

For the development version do:

```
git clone https://github.com/franciscovillaescusa/Pylians3.git
cd Pylians3
python -m pip install .
```

Note: Pylians3 is already installed in our CAMELS binder.

CHAPTER

THIRTYONE

TEAM

- Francisco Villaescusa-Navarro * (Simons Foundation & Princeton)
- Daniel Angles-Alcazar * (UConn & CCA)
- Shy Genel * (CCA & Columbia)
- IllustrisTNG team
- SIMBA team
- Astrid team
- SMAUG collaboration
- Nicholas Battaglia (Cornell)
- Simeon Bird (Riverside)
- Greg L. Bryan (Columbia)
- Blakesley Burkhart (Rutgers & CCA)
- Joyce Caliendo (UMass)
- William R. Coulton (CCA)
- Rupert Croft (Carnegie Mellon)
- Romeel Dave (Edinburg)
- Ana Maria Delgado (Harvard)
- Tiziana Di Matteo (Carnegie Mellon)
- Yu Feng (Berkeley)
- ChangHoon Hahn (Princeton)
- Sultan Hassan (CCA)
- Lars Hernquist (Harvard)
- Michael Eickenberg (CCM)
- Matthew Gebhardt (UConn)
- Vid Irsic (Cambridge)
- Yongseok Jo (CTP/Seoul)
- Neerav Kaushal (Michigan Tech)
- Katarina Kraljic (LAM/Marseille)

- Christina Kreisch (Princeton)
- Valentina La Torre (Tufts)
- Erwin T. Lau (Harvard)
- Yin Li (CCA/CCM)
- Luis Fernando Machado Poletti Valle (Zurich)
- Faizan G. Mohammad (Waterloo)
- Emily Moser (Cornell)
- Daisuke Nagai (Yale)
- Desika Narayanan (Florida)
- Yueying Ni (Carnegie Mellon/Harvard)
- Andrina Nicola (Princeton)
- Benjamin D. Oppenheimer (Colorado)
- Gabriele Parimbelli (Rome)
- Lucia A. Perez (Arizona)
- Oliver H. E. Philcox (Princeton)
- Alice Pisani (CCA/Cooper Union)
- Helen Shao (Princeton)
- Rachel S. Somerville (CCA)
- David N. Spergel (Simons Foundation)
- Ulrich P. Steinwandel (CCA)
- Leander Thiele (Princeton)
- Megan Tillman (Rutgers)
- Matteo Viel (SISSA)
- Pablo Villanueva-Domingo (IFIC/Valencia)
- Mark Vogelsberger (MIT)
- Digvijay Wadekar (IAS)
- Benjamin Wandelt (IAP & CCA)
- Kaze W.K. Wong (CCA)

* core team

CHAPTER

THIRTYTWO

CONTACT

For problems, bugs, questions,…etc please contact us as camel.simulations@gmail.com.

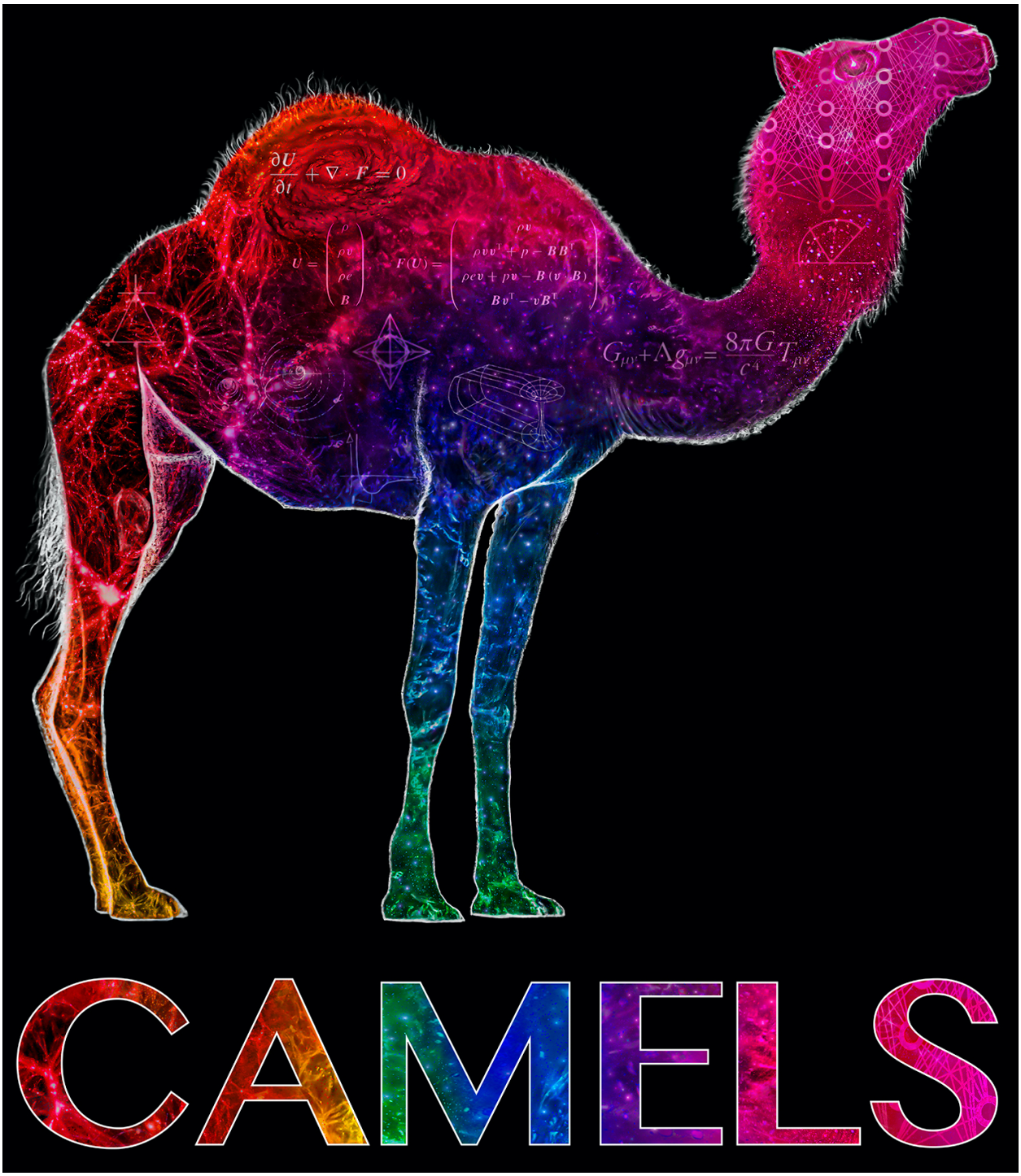
CHAPTER

THIRTYTHREE

LOGO

The CAMELS logo was designed taking into account its three main components:

- **Cosmology**: showing the large-scale structure of the Universe on the left.
- **Astrophysics**: represented by the galaxy and the stars in the front legs.
- **Machine Learning**: illustrated as an artificial neural network in the camel head.


CAMELS