

TUTORIAL ON AXION COSMOLOGY AND COSMOLOGICAL BOUNDS

Mathieu Kaltschmidt

mkaltschmidt@unizar.es

1st Cosmic WISPers Training School
Lecce, September 13th 2023

Background: G. Pierobon

Outline

1. Introduction to some basic numerics
2. Coding Exercises: Create your (first?) own axion string simulations
3. Explore some of the physics that was discussed in today's lectures
4. Outlook on current research efforts in this area

Basic Numerics

- Our daily business as physicists: Solving differential equations!
- There is not always an analytical solution to every problem!



- Build an algorithm, that preserves the conservation laws! (Symplectic Algorithms)
- Basic example: Hamiltonian dynamics!

Numerics: Why and How?

- Discretize space and time and solve EoM on a discrete lattice
- Time evolution via **Timestepping**
- General scheme:

$$X(t_1) = M(X(t_0))$$

$$X(t_2) = M(X(t_1))$$

...

$$X(t_n) = M(X(t_{n-1}))$$

- **Goal:** Find Map M (= algorithm!) that is suited to deal with your problem!

Explicit vs. Implicit Methods

- **Explicit methods:** Compute state of a system at later times from current state
- **Implicit methods:** Compute state by solving equation involving both, current and later states of the system

Explicit	Implicit
+ Computationally cheaper	- More Expensive
- Less stable	+ Usually very stable

Order of Convergence

- Of course we need to find an algorithm, that converges with a certain level of accuracy to preserve the quantities we are simulating:

$$\mathcal{O}(\Delta t^n) \simeq X_{\text{exact}}(t) - X(t)$$

- It is often a crucial task to balance between optimal precision and performance of your code.

Overview of frequently used Integrators I

- **Forward Euler Method.** Finite difference def. of the derivative / Taylor expansion:

$$\frac{dX}{dt} \simeq \frac{X(t + \Delta t) - X(t)}{\Delta t} \implies X(t + \Delta t) \simeq X(t) + \frac{dX}{dt} \Delta t + \mathcal{O}(\Delta t^2)$$

- **Runge-Kutta.** Kick-Drift scheme. Implicit / Explicit versions. Standard RK4 Example:

$$X(t + \Delta t) = X(t) + \frac{1}{6} \left(P^{(1)} + 2P^{(2)} + 2P^{(3)} + P^{(4)} \right) \Delta t$$
$$P(t + \Delta t) = P(t) + \frac{1}{6} \left(K_1^P + 2K_2^P + 2K_3^P + K_4^P \right) \Delta t$$

Notation:

$$P^{(1)} = P(t), \quad P^{(2)} = P(t) + \frac{\Delta t}{2} K_1^P, \quad P^{(3)} = P(t) + \frac{\Delta t}{2} K_2^P, \quad P^{(4)} = P(t) + \Delta t K_3^P$$

$$K_1^P = K(X(t), P(t)), \quad K_2^P = K\left(X(t) + \frac{\Delta t}{2} P^{(1)}, P^{(1)}\right), \quad K_3^P = K\left(X(t) + \frac{\Delta t}{2} P^{(2)}, P^{(2)}\right), \quad K_4^P = K(X(t) + \Delta t P^{(3)}, P^{(3)})$$

Overview of frequently used Integrators II

- Position / Velocity **Verlet**:

$$\begin{aligned}X(t + \Delta t/2) &= X(t) + \frac{1}{2}P(t)\Delta t \\P(t + \Delta t) &= P(t) + X(t + \Delta t/2)\Delta t \\X(t + \Delta t) &= X(t + \Delta t/2) + \frac{1}{2}P(t + \Delta t)\Delta t\end{aligned}$$

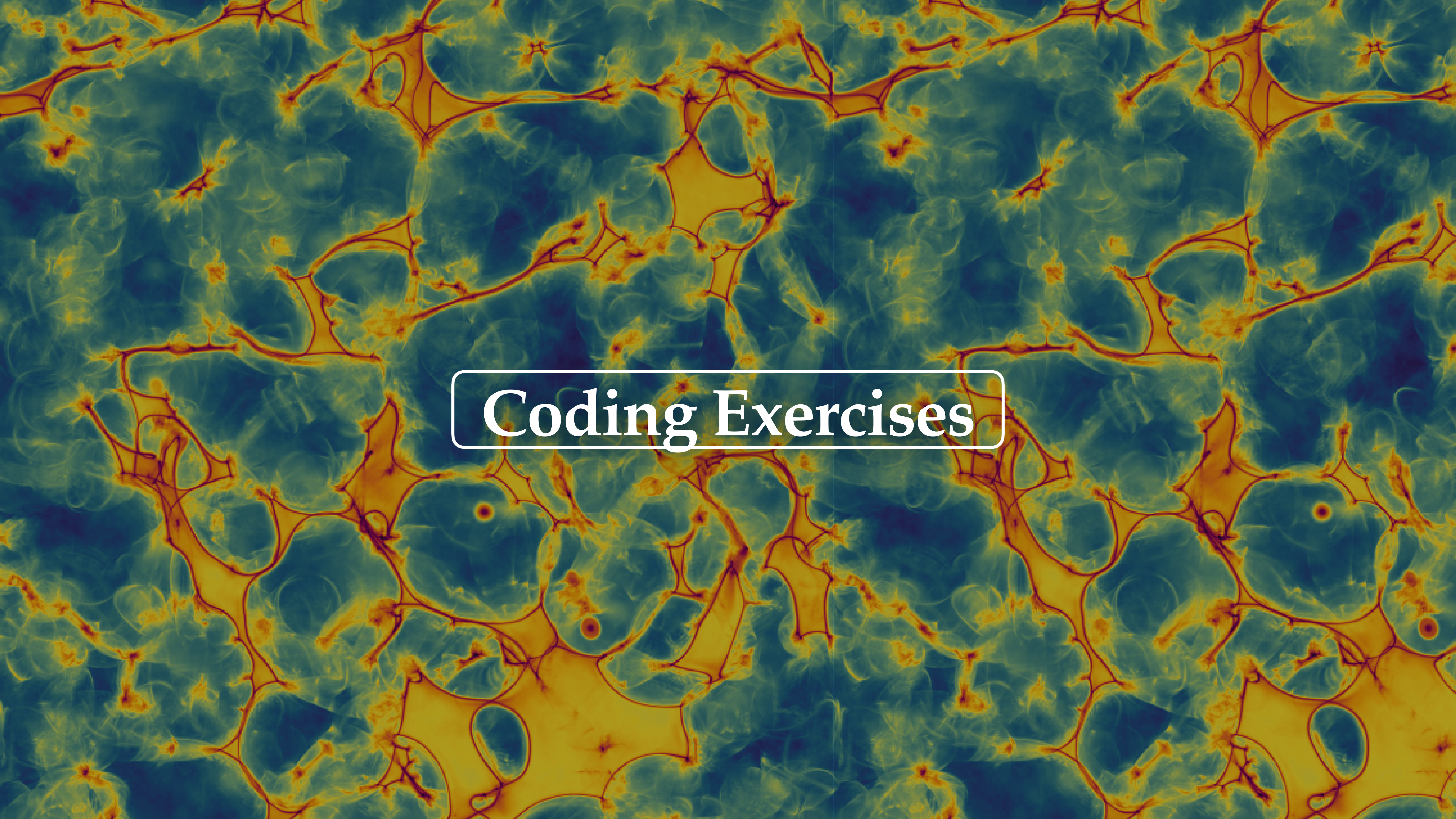
$$\begin{aligned}P(t + \Delta t/2) &= P(t) + \frac{1}{2}X(t)\Delta t \\X(t + \Delta t) &= X(t) + P(t + \Delta t/2)\Delta t \\P(t + \Delta t) &= P(t + \Delta t/2) + \frac{1}{2}X(t + \Delta t)\Delta t\end{aligned}$$

- **Leapfrog**. Very similar to the Verlet method, but with evaluation at different times!

You will implement this algorithm today :-)

Summary

1. Break down your PDE / ODE down to the simplest possible form
2. Perform time evolution via discrete time stepping
3. Physics informs the choice of the algorithm
4. Accuracy of the solver matters
5. Memory can be an issue, especially for implicit methods



Coding Exercises

Exercise 1: First Simulation for a simple Toy Model

- In order to solve the respective equation of motion, we need to discretize it on a lattice. We start by implementing the discretised version of the Laplacian in **2D** with **periodic boundary conditions**!

$$\phi''(x_i) = \frac{1}{\Delta x^2} \sum_{j=1}^{N_g} C_j (\phi(x_{i+j}) + \phi(x_{i-j}) - 2\phi(x_i))$$

N_g	C_1	C_2	C_3	C_4
1	1	0	0	0
2	16/12	-1/12	0	0
3	3/2	-3/20	1/90	0
4	8/5	-1/5	8/315	-1/516

2 neighbours
in our case!

Coefficients for calculating the discretised Laplacian

Exercise 1: First Simulation for a simple Toy Model

- The next step is to define a function that computes the integration kernel at every timestep. For our case it should look like this:

$$K\left(\phi_i, \dot{\phi}_i\right) = \triangle(\phi_i) - \frac{2\pi^2}{w^2} \left(\phi_1^2 + \phi_2^2\right) \phi_i - \left(\frac{\alpha \cdot \text{era}}{t}\right) \dot{\phi}_i$$

- Note, that we have to compute the kernel for both, the real and the imaginary part of the field.

Exercise 1: First Simulation for a simple Toy Model

- The last step of this exercise is now to use these tools, to implement a simple Leapfrog integration scheme.
- You need to update the respective field values and derivatives according to the following scheme:

$$\begin{aligned}\phi^+ &= \phi + \Delta t \left(\dot{\phi} + \frac{1}{2} K(\phi, \dot{\phi}) \Delta t \right) \\ K^+ &= K(\phi^+, \dot{\phi}) \\ \dot{\phi}^+ &= \dot{\phi} + \frac{1}{2} (K + K^+) \Delta t\end{aligned}$$

Notation:

$$X \equiv X(t)$$

$$X^+ \equiv X(t + \Delta t)$$

- Now you can check your implementation by creating an animation with the code we prepared for you! Play around with the size of the timestep: What happens?

Exercise 2: Adding a Mass Term

- To study a case which is closer to the actual axion, case we now focus on a model with a mass term. Your task is to extend the previous implementation by adding the correct mass term to your definitions:

$$K(\phi_1, \dot{\phi}_1) = \triangle(\phi_1) - \frac{2\pi^2}{w^2} (\phi_1^2 + \phi_2^2) \phi_1 - \left(\frac{\alpha \cdot \text{era}}{t} \right) \dot{\phi}_1 - \frac{m^2 \phi_2^2}{(\phi_1^2 + \phi_2^2)^{3/2}}$$
$$K(\phi_2, \dot{\phi}_2) = \triangle(\phi_2) - \frac{2\pi^2}{w^2} (\phi_1^2 + \phi_2^2) \phi_2 - \left(\frac{\alpha \cdot \text{era}}{t} \right) \dot{\phi}_2 + \frac{m^2 \phi_1 \phi_2}{(\phi_1^2 + \phi_2^2)^{3/2}}$$

- What changes for the massive case? What do you observe?
- For the axion case, the mass is not constant, it strongly depends on the temperature! If you want you can play if a varying mass term.

Bonus: Try to implement more complex cases

Some suggestions for those of you who want to extend the general setup we just created:

1. Generalise the setup to 3 spatial dimensions
2. Study the effects of a non-constant mass term (i.e. play with the potential)
3. Get your hands on some of the current literature. You should be ready to understand the general framework now.
Suggestion: Buschmann, Foster & Safdi [[1906.00967](#)], especially the supplementary material!
4. Ask me about whatever interests you in this context!

The background of the slide is a Cosmic Microwave Background (CMB) fluctuation map. It displays a complex pattern of temperature variations across the sky, with warmer regions (yellow/orange) and cooler regions (blue). The pattern is characterized by a network of filaments and voids, typical of the large-scale structure of the universe.

Outlook: Jaxions code

Jaxions Code

- State-of-the-art, highly parallelised, (static-grid) code to simulate the evolution of the axion dark matter field in the early Universe
- Publicly available on Github: <https://github.com/veintemillas/jaxions>

jaxions v0

**Mathieu Kaltschmidt,^a Giovanni Pierobon,^b Javier Redondo,^{a,c}
Kenichi Saikawa,^{c,d} Alejandro Vaquero^a**

^aUniversity of Zaragoza, P. Cerbuna 12, 50009 Zaragoza, Spain

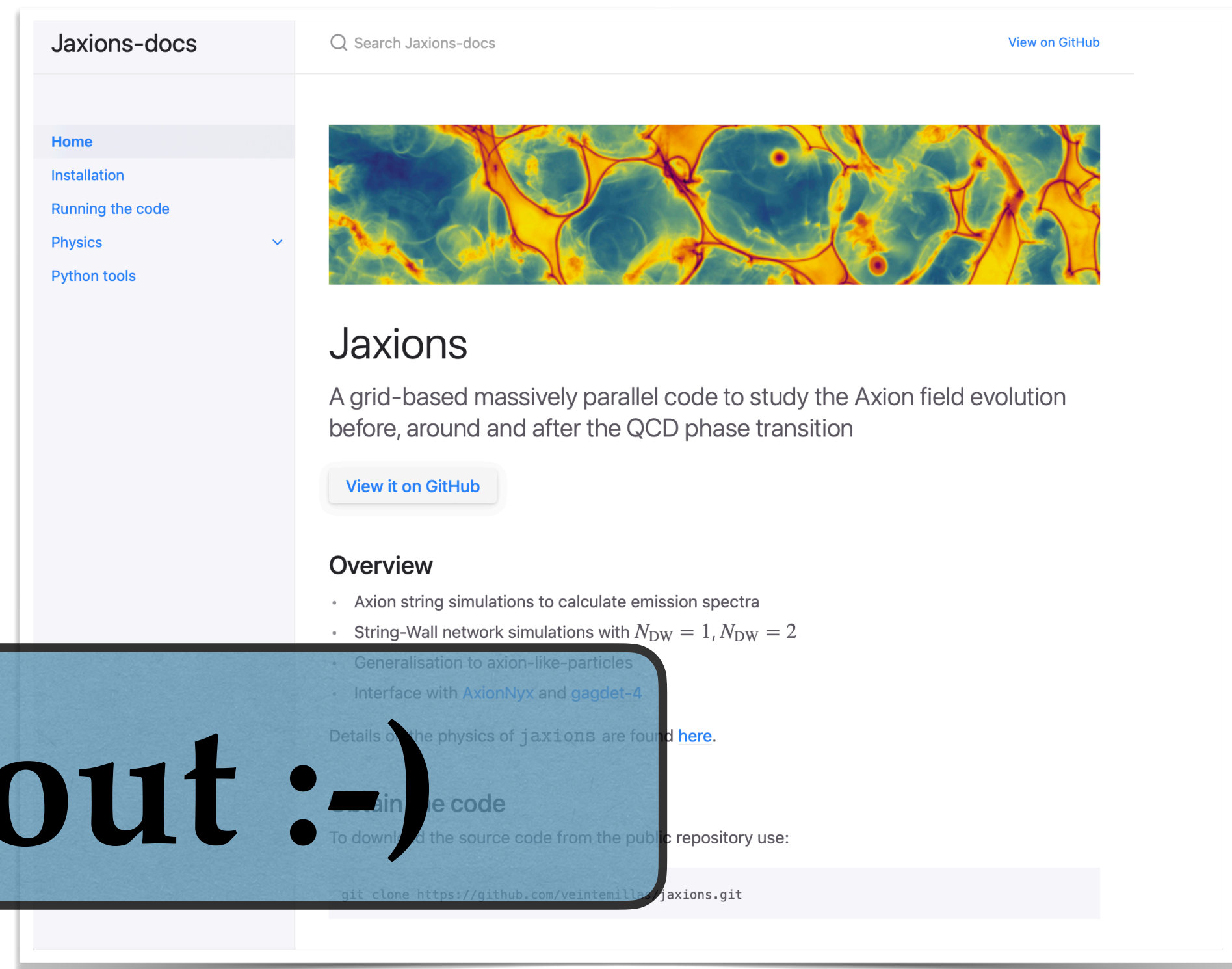
^bSchool of Physics, The University of New South Wales, NSW 2052 Kensington, Sydney, Australia

^cMax-Planck-Institut für Physik (Werner-Heisenberg-Institut), Föhringer Ring 6, 80805 München, Germany

^dInstitute for Theoretical Physics, Kanazawa University, Kakuma-machi, Kanazawa, Ishikawa 920-1192, Japan

E-mail: jredondo@unizar.es

Abstract. We describe the `jaxions` numerical code to simulate the evolution and properties of the axion dark matter field.



Jaxions-docs

Search Jaxions-docs

View on GitHub

Home

Installation

Running the code

Physics

Python tools

Jaxions

A grid-based massively parallel code to study the Axion field evolution before, around and after the QCD phase transition

View it on GitHub

Overview

- Axion string simulations to calculate emission spectra
- String-Wall network simulations with $N_{\text{DW}} = 1, N_{\text{DW}} = 2$
- Generalisation to axion-like-particles
- Interface with `AxionNyx` and `gadget-4`

Details of the physics of `jaxions` are found [here](#).

Get the code

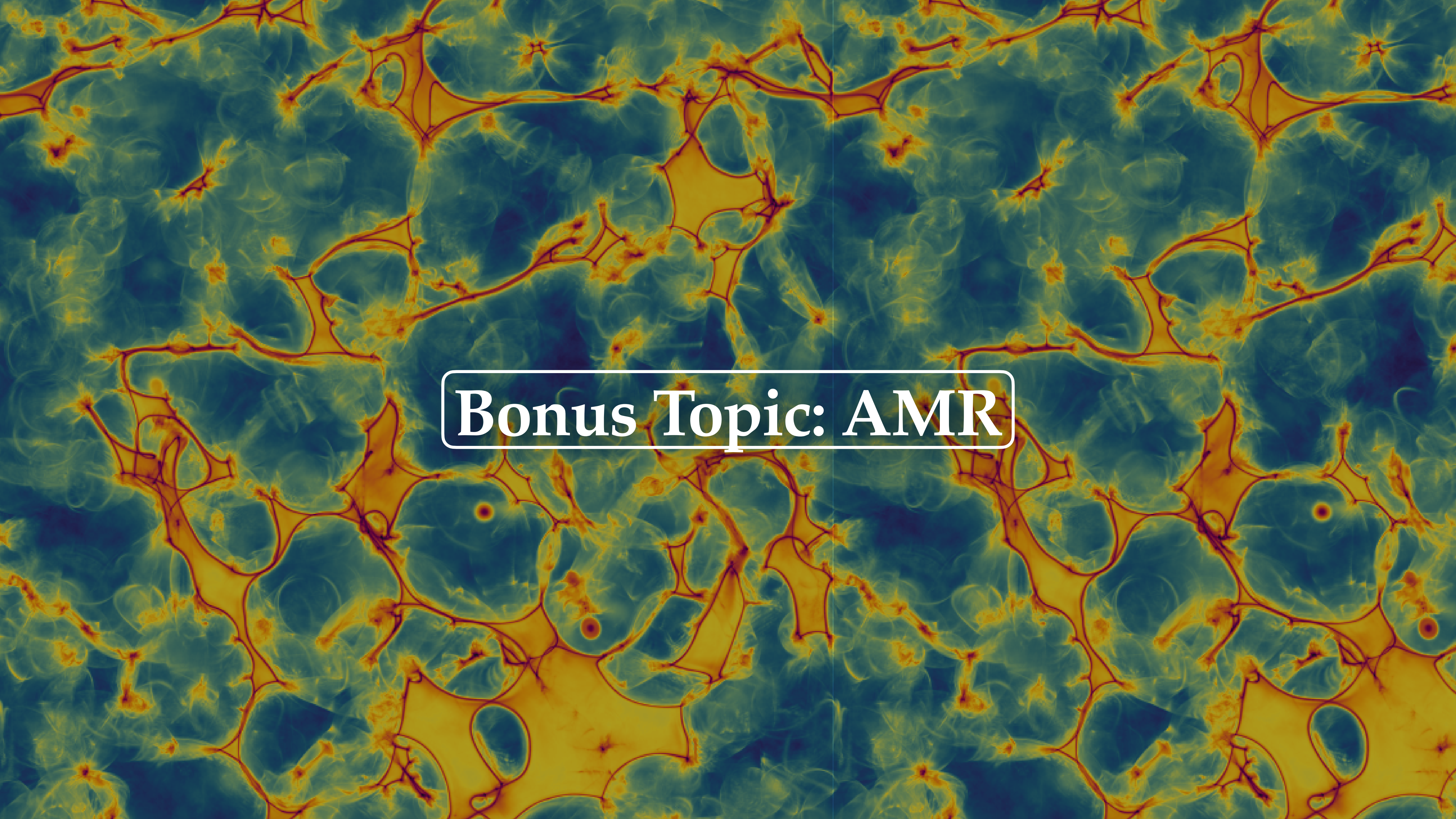
To download the source code from the public repository use:

```
git clone https://github.com/veintemillas/jaxions.git
```

Check it out :-)

References using the Jaxions Code

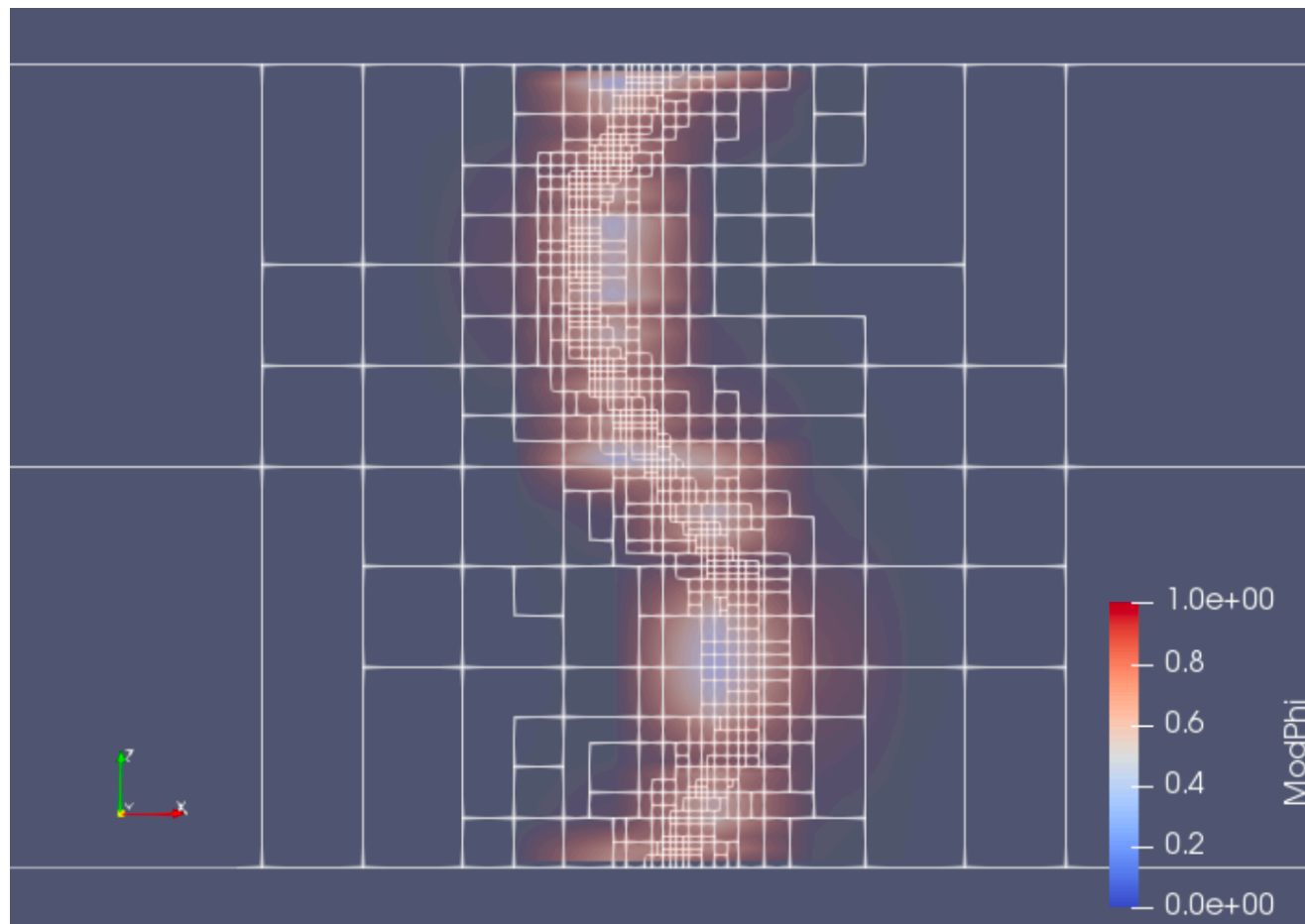
1. A. Vaquero, J. Redondo, J. Stadler (2018): Early seeds of axion miniclusters, [1809.09241](#)
2. B. Eggemeier, J. Redondo, K. Dolag, J. Niemeyer, A. Vaquero (2019): First Simulations of Axion Minicluster Halos, [1911.09417](#)
3. C. O'Hare, G. Pierobon, J. Redondo, Y. Wong (2021): Simulations of axionlike particles in the post-inflationary scenario, [2112.05117](#)
4. B. Eggemeier, C. O'Hare, G. Pierobon, J. Redondo, Y. Wong (2022): Axion minivoids and implications for direct detection, [2212.00560](#)
5. G. Pierobon, J. Redondo, K. Saikawa, A. Vaquero, G. D. Moore (2023): Miniclusters from axion string simulations, [2307.09941](#)



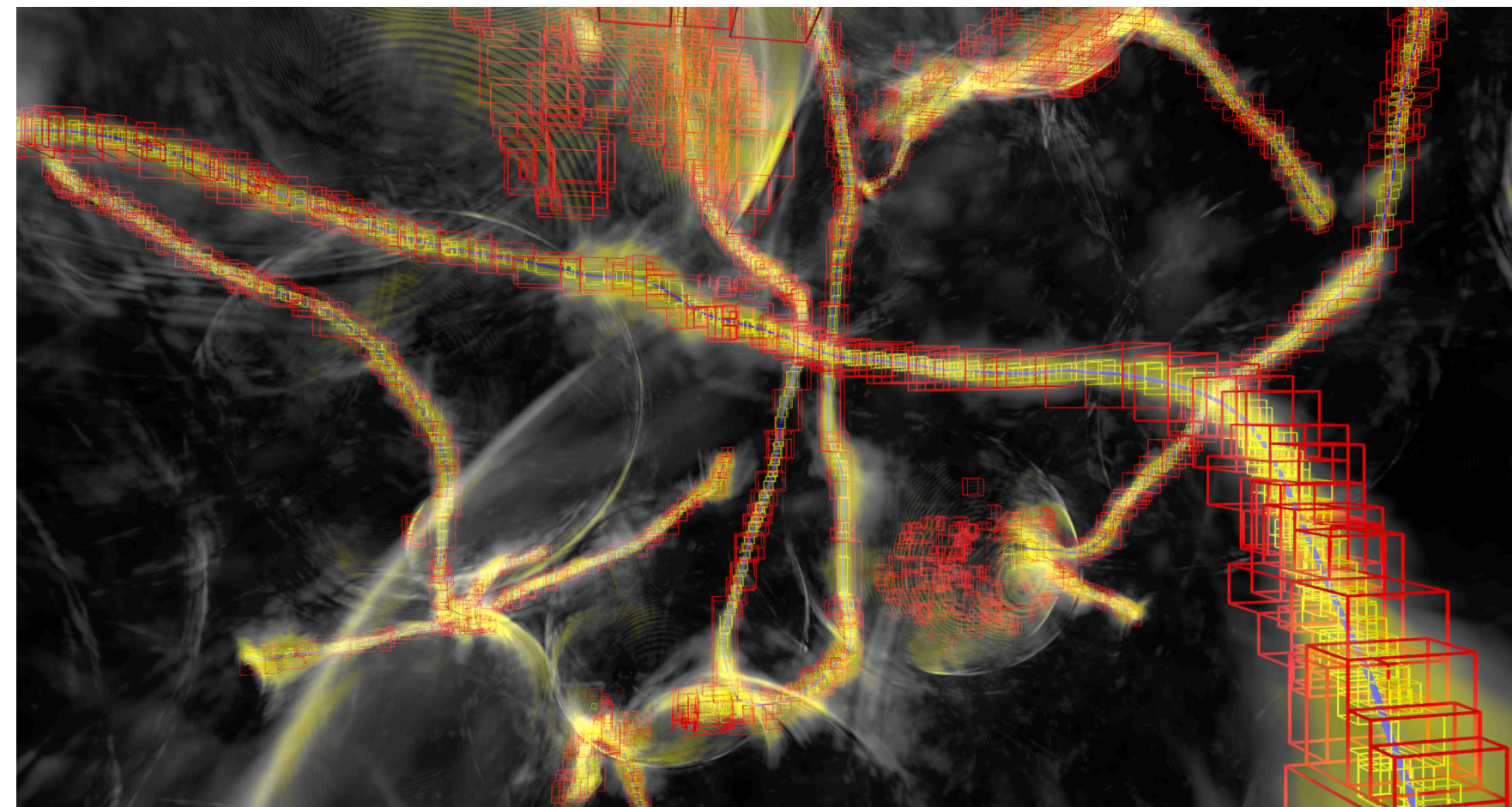
Bonus Topic: AMR

Adaptive Mesh Refinement

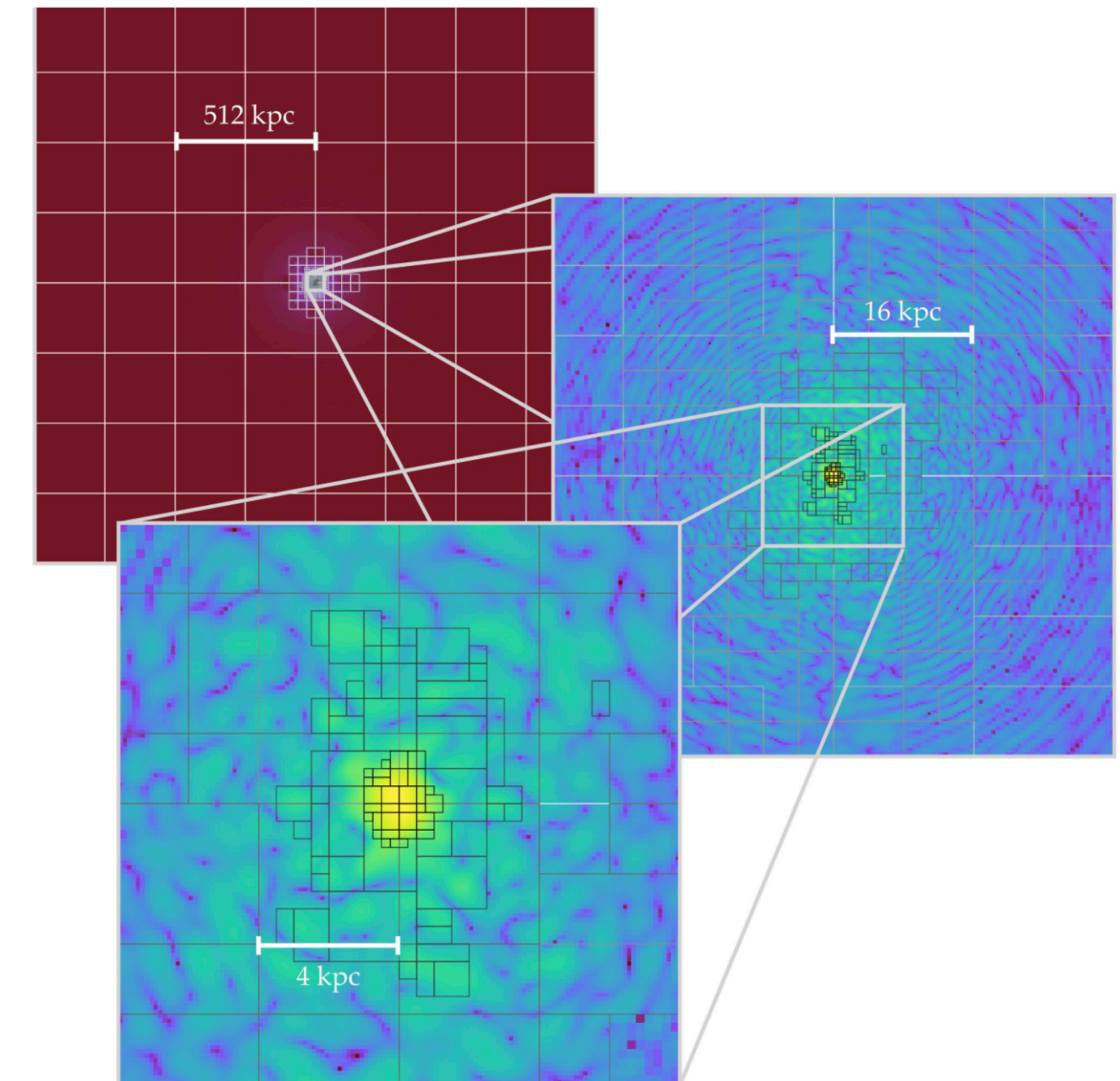
- **Idea:** Focus computational power on specific parts
- Nowadays widely used in cosmological simulation codes, engineering applications **and** recently in axion string simulations
- Current codes mostly based on AMReX:
<https://amrex-codes.github.io/amrex/>



Drew & Shellard [[1910.01718](#)]



Benabou *et al.* [[2308.01334](#)]



Schwabe *et al.* [[2007.08256](#)]

Our current AMR Framework

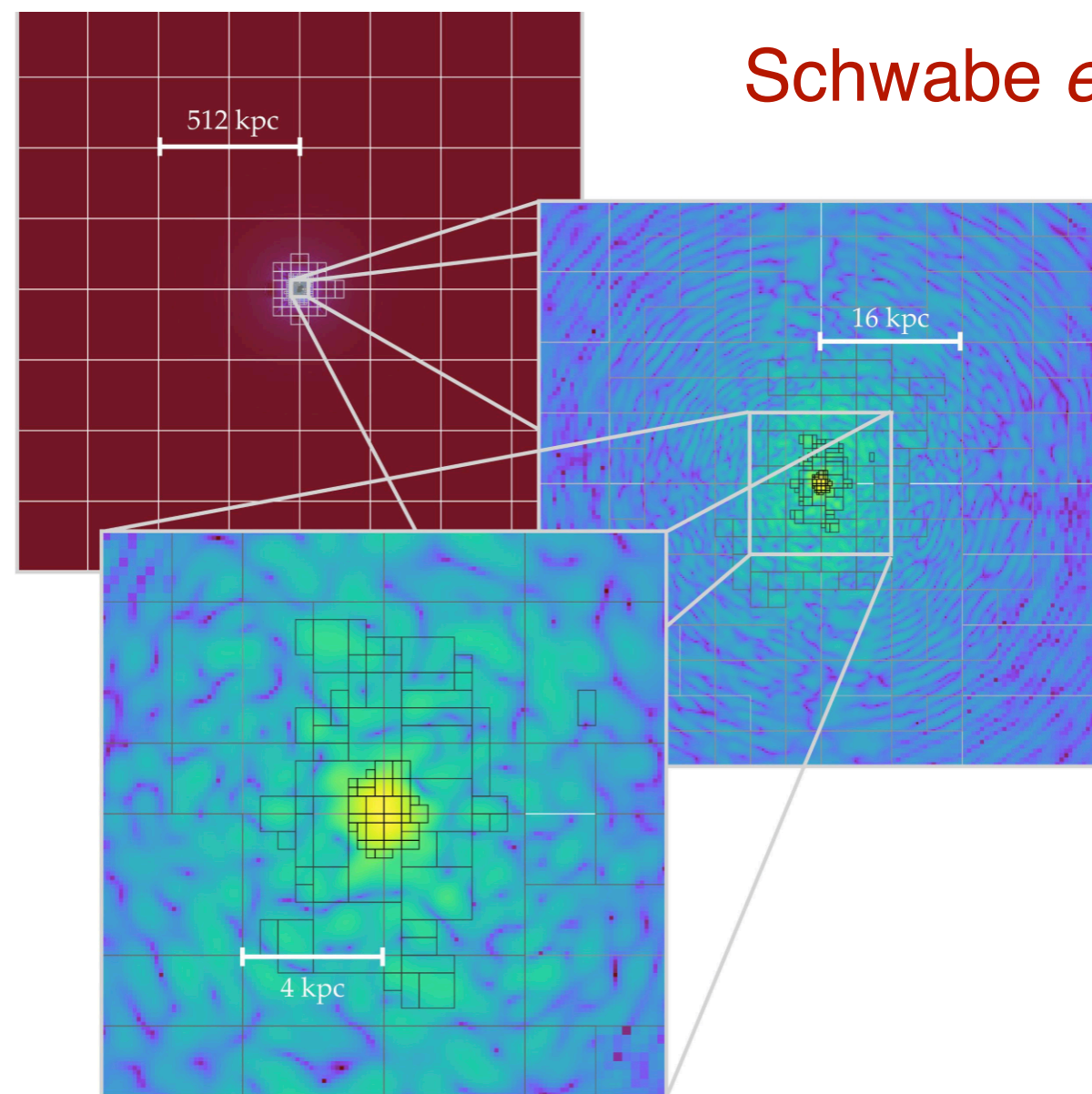
Nyx

an adaptive mesh, cosmological hydrodynamics simulation code

Almgren *et al.* [ApJ 765 39, 2013]

AxioNyx

Extended for Fuzzy DM
Simulations + String Networks



Schwabe *et al.* [2007.08256]

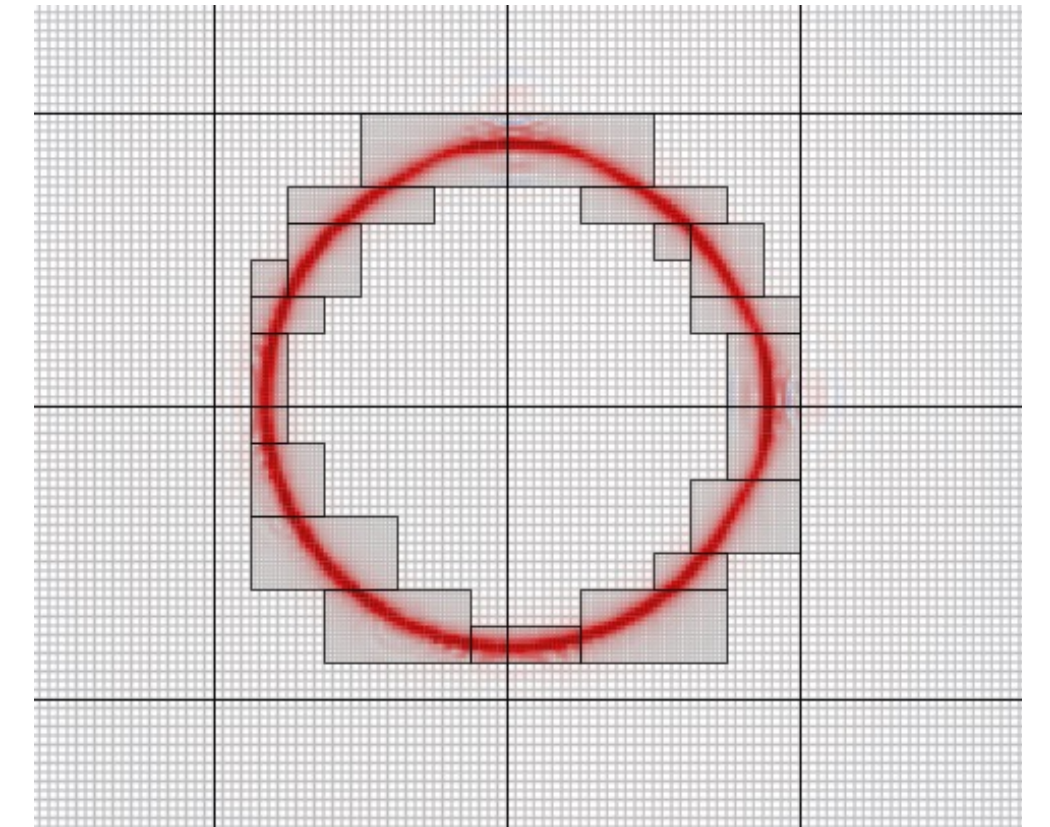
Generate ICs with Jaxions

Convert grid data to boxlib format

Read IC and interpolate
on grid(s)

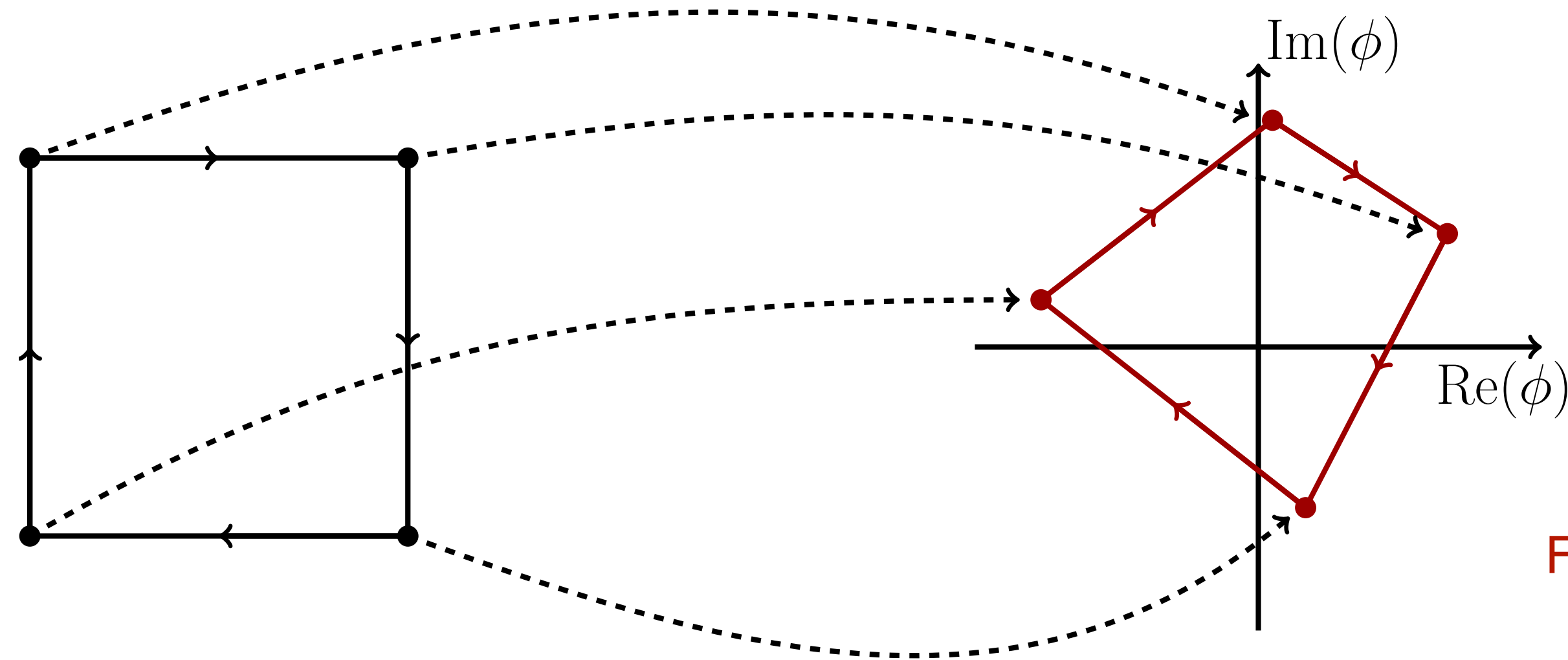
Evolve System with
AxioNyx (with AMR)

Analyse Data



yt (Python), ParaView (in principle all programs that can handle the boxlib format)

Practical Guide: Tagging Strings



Fleury and Moore [[1509.00026](#)]

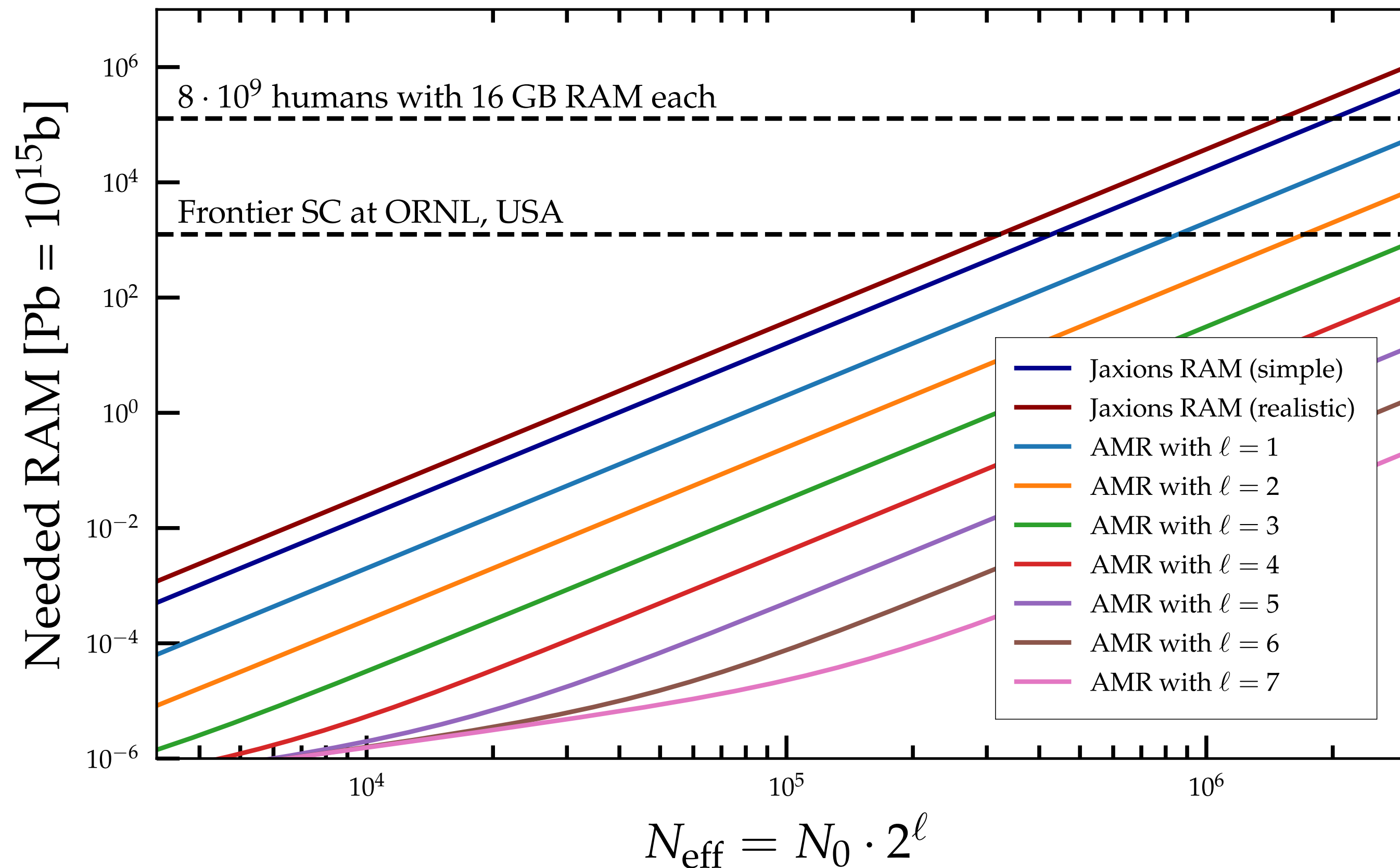
- Map four points around a plaquette to four points on the complex ϕ plane
- If the resulting tetragon encloses the origin, identify plaquette as pierced by a string:

$$N_{\text{plaquettes}} = 6\xi \left(\frac{L}{N_0\tau} \right)^2 N_0^3$$

RAM Estimates for String Simulations with AMR

- Need $\text{RAM}(N_0, \ell) = 2 \times 2 \times 4 \text{ bytes} \times \left(N_0^3 + \pi \cdot 2^7 (2^\ell - 1) 6\xi \left(\frac{L}{\tau} \right)^2 N_0 \right)$

Contribution from each level of refinement!



References for Axion String Simulations with AMR

1. A. Drew, E. P. S. Shellard (2019): Radiation from Global Topological Strings using Adaptive Mesh Refinement: Methodology and Massless Modes, [1910.01718](#)
2. A. Drew, E. P. S. Shellard (2022): Radiation from Global Topological Strings using Adaptive Mesh Refinement: Massive Modes, [2211.10184](#)
3. M. Buschmann et al. (2021): Dark Matter from Axion Strings with Adaptive Mesh Refinement, [2108.05368](#)
4. J. Benabou et al. (2023): Signatures of Primordial Energy Injection from Axion Strings, [2308.01334](#)
5. Lots of work in progress ...

Stay tuned!