Finanziato dall'Unione europea
NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Map-Level Emulator of CMB systematics

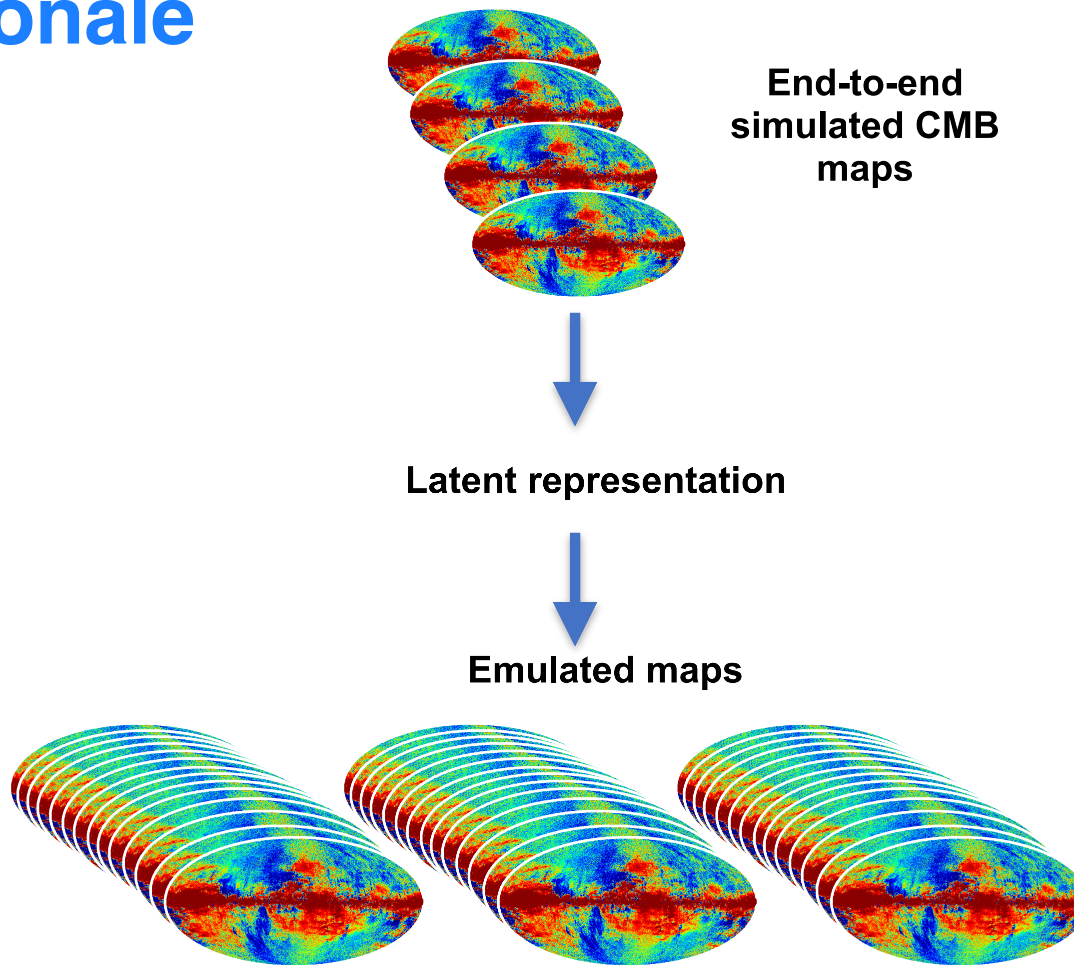
Paolo Campeti (INFN Ferrara)
In collaboration with: Martina Gerbino, Massimiliano Lattanzi, Luca Pagano

Spoke 3 Technical Workshop, Trieste October 9 / 11, 2023

Map-level emulator: Scientific Rationale

- End-to-end simulations of systematic effects in CMB experiments is highly computationally expensive
- Typically only ~few hundreds of simulated maps are available ($\mathcal{O}(10)$ kCPU hrs per simulation)
- High accuracy and ML studies need $\mathcal{O}(10^{4-5})$ or more simulations! Repeat for different sys. effects…

- Circumvent simulation using generative modelling to emulate maps!
- But…small training sets not sufficient for direct emulation of the fields
- Use a latent representation of the fields as intermediate step!
- Then emulate maps from this latent representation



Technical Objectives, Methodologies and Solutions

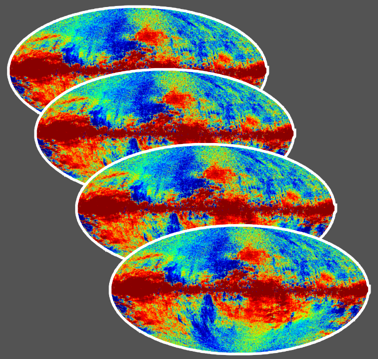
- **Choose a Latent representation: Wavelet Phase Harmonics (WPH)**
- **Why?**
  - **Highly sensitive to non-Gaussian information content (typical of systematics)**
  - **Doesn't require training!**
  - **Need to make sure that emulated observables are distributed correctly**
- **WPH can be contrasted with CNNs: WPH filters are defined by wavelets rather than learned from data → can be applied when number of simulations is small or just 1**
- **Used in recent literature in similar contexts:**
  - Generation of dust foregrounds maps from single training image *(Jeffrey + 2021, Regaldo-Saint Blancard + 2022)*
  - Dust polarisation de-noising for Planck data *(Delouis+2023)*
  - Emulating CMB anisotropies maps from cosmic strings *(Price + 2023)*
  - …

Technical Objectives, Methodologies and Solutions - 2

- **Workflow for *extreme data augmentation* as in *Price+2023***

A. Simulation

1. Create a small ensemble of (expensive) *end-to-end* simulations

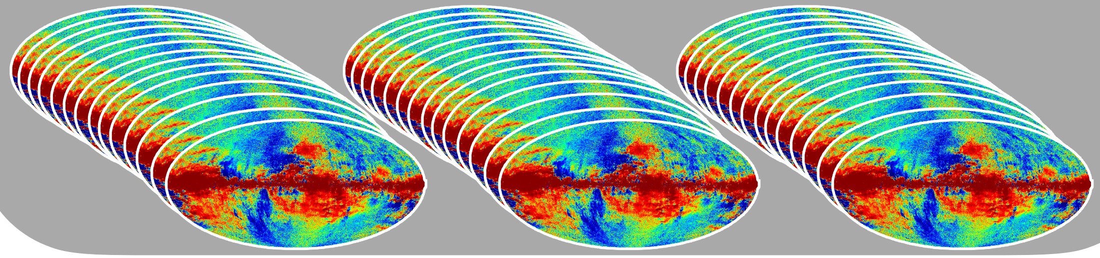


B. Compression Step

1. Draw uniform random simulation x_{sim} from the ensemble
2. Compute latent vector $z_{\text{sim}} = \Phi(x_{\text{sim}})$ where Φ is the WPH mapping

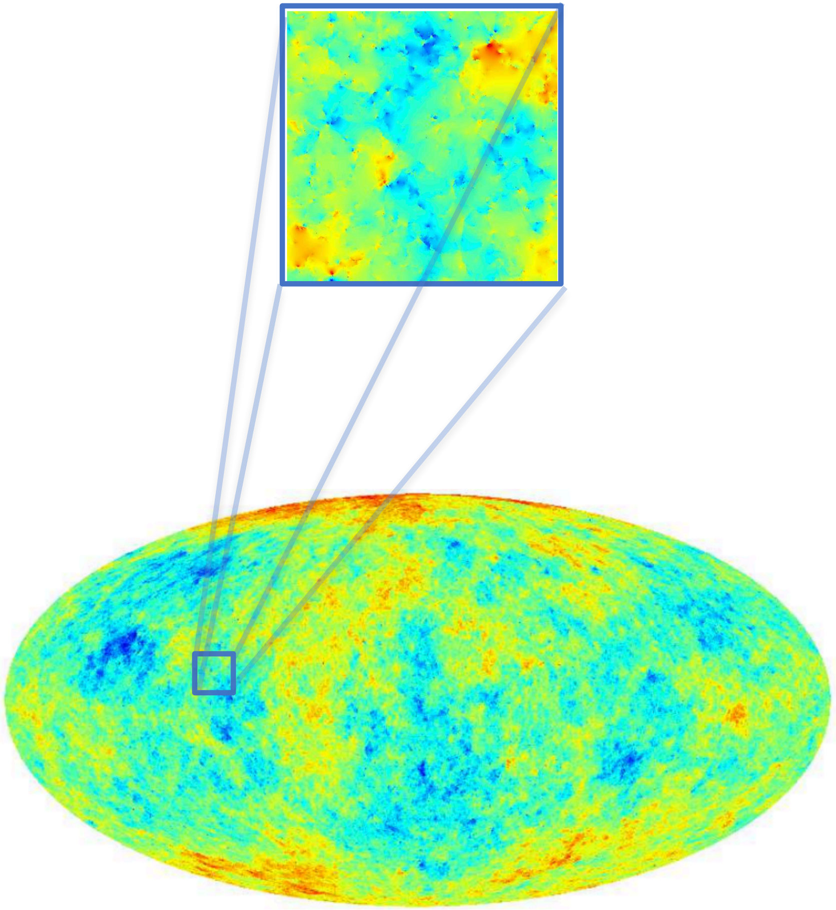
C. Synthesis Step

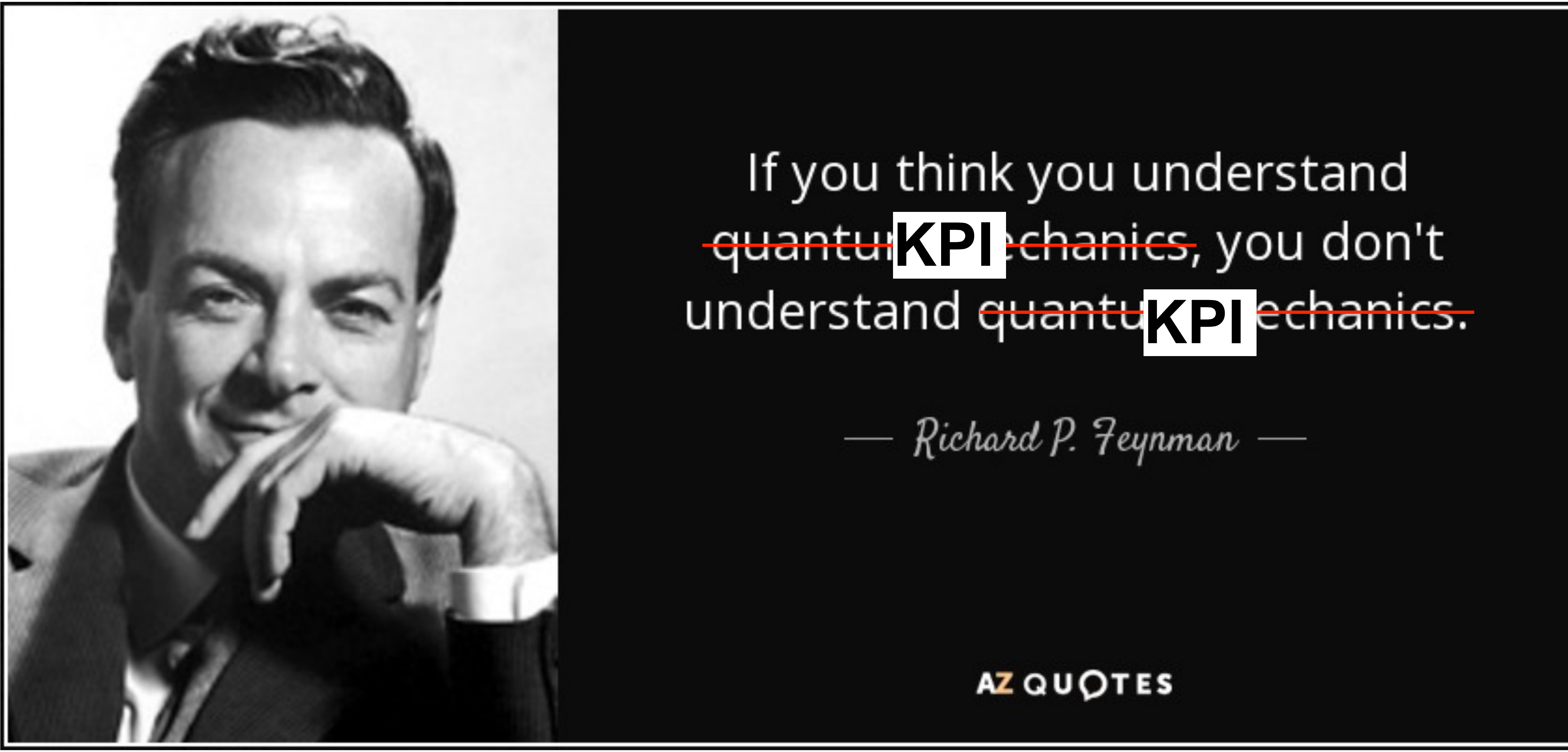
1. Take random Gaussian realisation x_0
2. Using autodiff property of compression mapping Φ, do iterative minimisation of ℓ_2 -loss $\mathcal{L}(x) = ||\Phi(x) - z_{\text{sim}}||_2^2$
3. Obtain x_{emu} such that $\Phi(x_{\text{emu}}) \simeq z_{\text{sim}}$
4. repeat steps B and C to generate many emulated maps



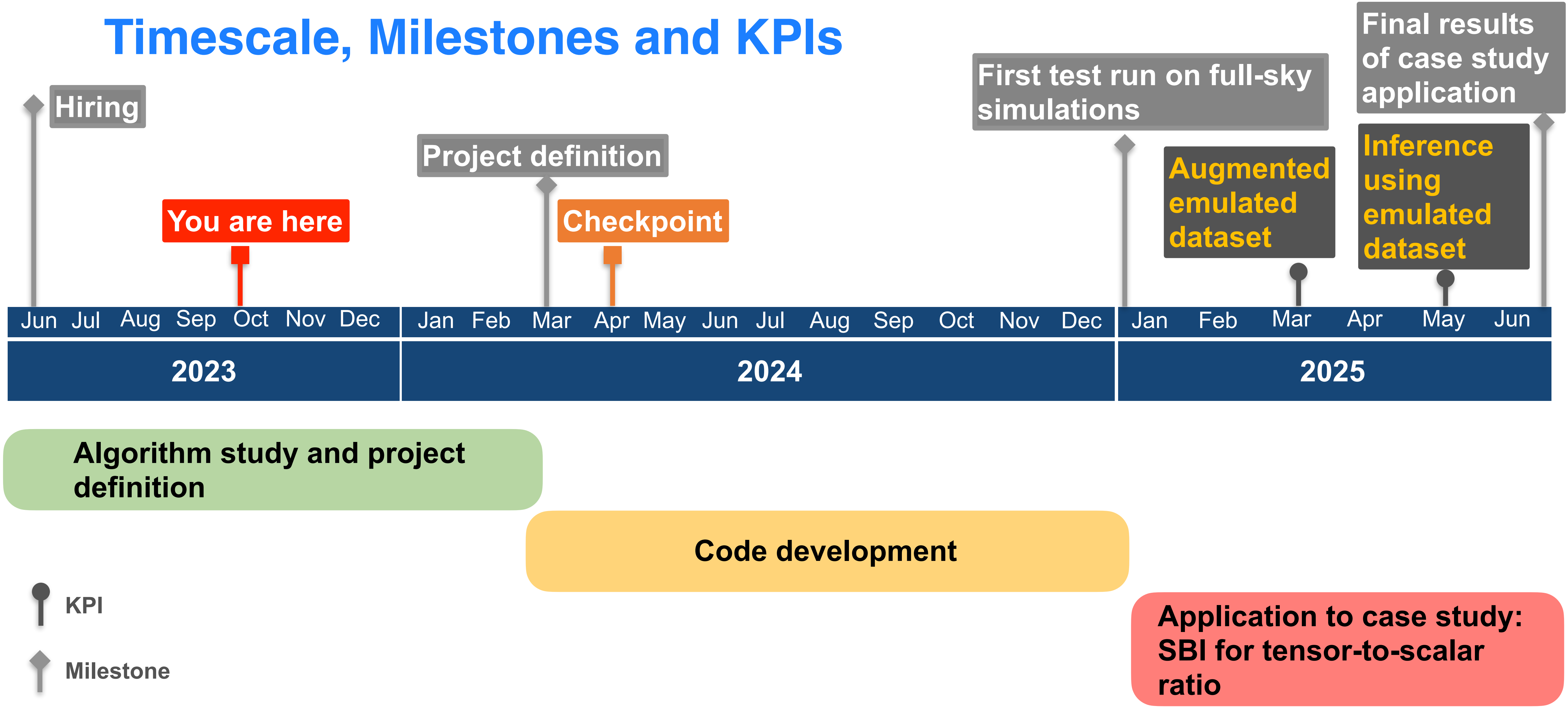
Technical Objectives, Methodologies and Solutions - 3

- **Public codes apply only to *small flat-sky patches* suitable for ground-based experiments (e.g. SO/ CMB-S4)**
- **We want to apply this to CMB systematics in full-sky suitable for satellites such as LiteBIRD: spherical scattering transform** (*Delouis+2022, McEwen+2021*)
- **Training set for LiteBIRD is already available**
- **Need to retain the correlations in the systematics among different frequency channels of the experiment (***Regaldo-Saint Blancard + 2022, Delouis+2022***)**
- **Complete our study by using the emulated training set for Simulation Based Inference for cosmological parameters (e.g. tensor-to-scalar ratio)**




If you think you understand
~~quantu~~ KPI ~~echanics~~, you don't
understand ~~quantu~~ KPI ~~echanics.~~
— Richard P. Feynman —
AZ QUOTES

Timescale, Milestones and KPIs



Accomplished Work, Results

- **Initial goal before hiring was just broadly defined**
- **I studied recent literature in order to find an appropriate problem to tackle**
- **I'm currently studying literature to identify the best algorithm for our needs for our preliminary project definition**
- **I'm gathering the most suitable available codes as a baseline to modify for our specific needs**

Next Steps and Expected Results (by next checkpoint: April 2024)

- Identification and definition of the algorithms suitable for our case study
- The algorithm presented here is very promising: latent space defined via WPH filter + maximum likelihood algorithm for emulation + SBI algorithm for inference on cosmological parameters
- Study and gather the codes already available in the literature
- By next checkpoint we aim to:

    1. Have project and algorithms defined
    2. Have started code development

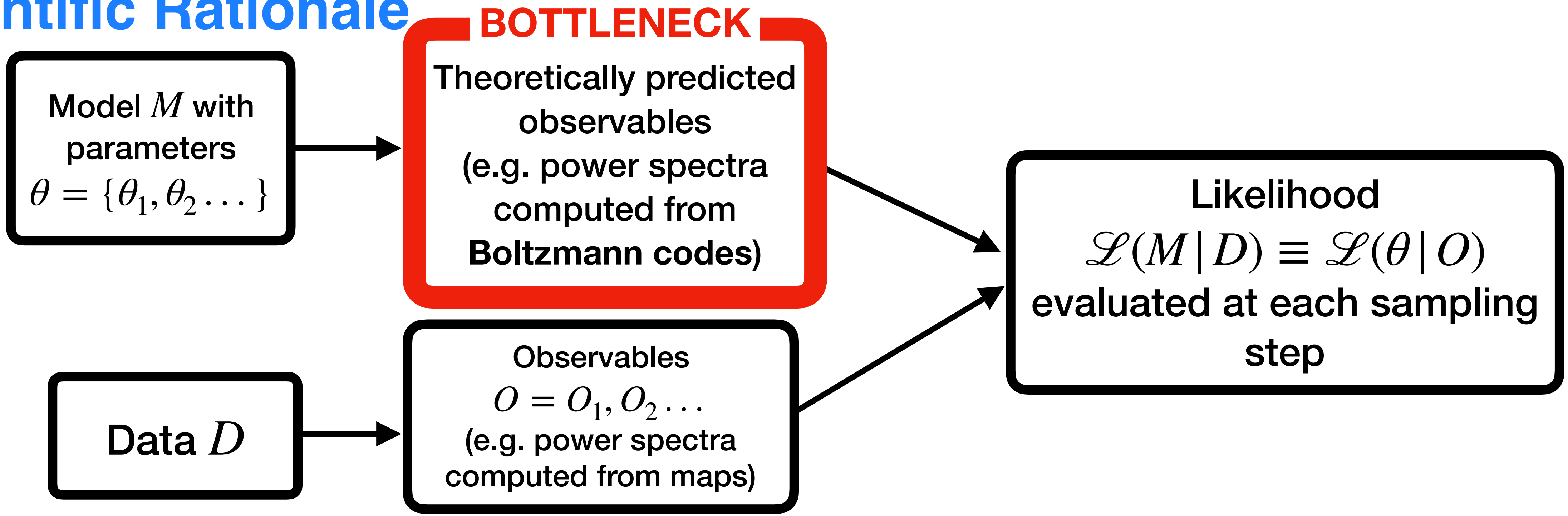
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

GPU-Porting of Boltzmann Codes

Paolo Campeti (INFN Ferrara)
In collaboration with: Martina Gerbino, Massimiliano Lattanzi

Spoke 3 Technical Workshop, Trieste October 9 / 11, 2023

Scientific Rationale



- **Bottleneck can be overridden by NN emulators! Factor $\mathcal{O}(10^{3-6})$ faster than Boltzmann solvers**
- **But…NN emulators require re-training for each cosmological model being compared to data (e.g. beyond ΛCDM models)**
- **Re-training can be very expensive! $\mathcal{O}(10^5 - 6)$ spectra needed, $\mathcal{O}(10)$ mins per spectrum**
- **Solution: porting Boltzmann codes Camb or Class to GPU!**

Technical Objectives, Methodologies and Solutions

- **The two main bottlenecks in Einstein-Boltzmann Solvers are:**

  1. **The integration of cosmological perturbations over time that provides all transfer functions and related source functions.**

  2. **The line-of-sight integrals which project the transfer function from Fourier space to harmonic space**

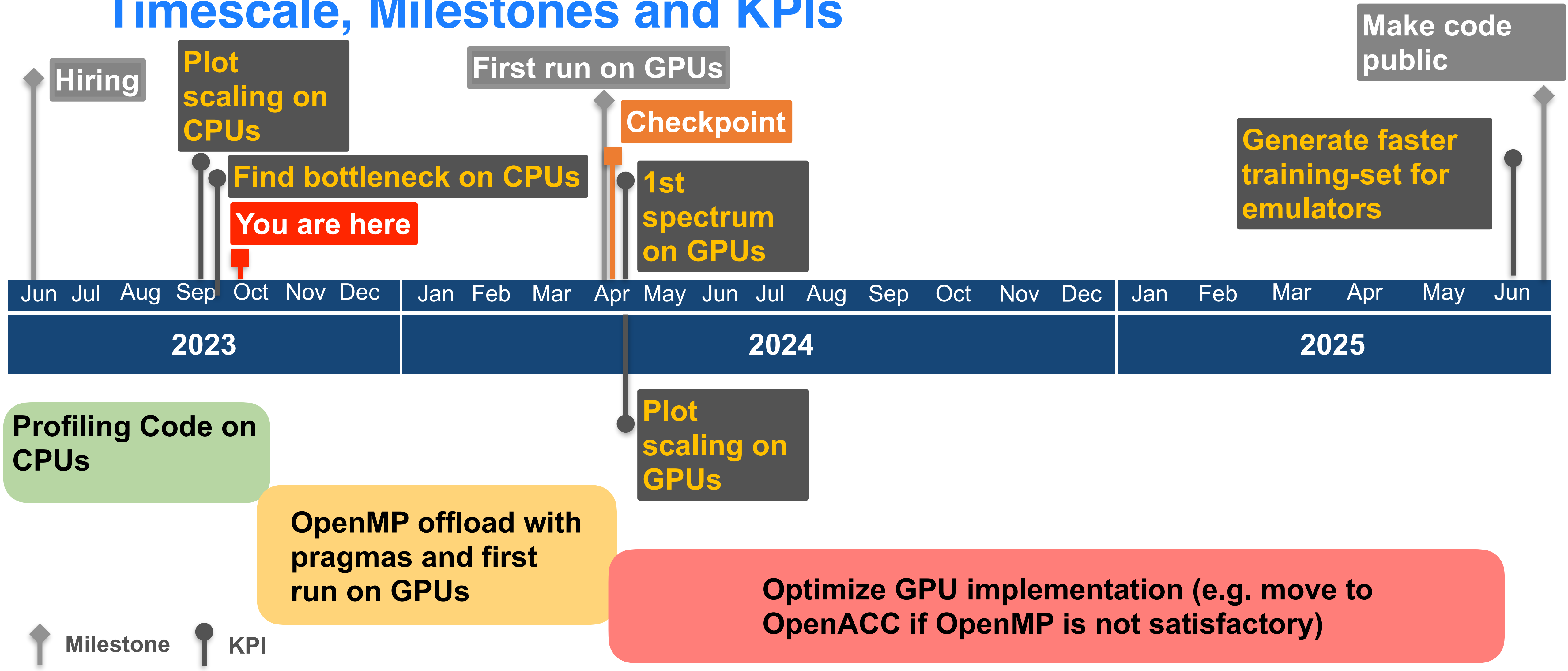
$$C_\ell^{XY} = 4\pi \int \frac{dk}{k} \mathcal{P}_\mathcal{R}(k) \Delta_\ell^X(k) \Delta_\ell^Y(k) \qquad \Delta_\ell^E(k) = \int d\tau S_P(k,\tau) \epsilon_\ell(k[\tau_0 - \tau]) \ .$$

- **Step 1: perturbation module ideally suited for an NN approach (e.g. ClassNet), very difficult to optimize with HPC techniques: ODE algorithms are sequential in nature. Speedup factor ~ 3.**

- **Step 2: harmonic transfer module, can be tackled with GPU porting! Large number of independent integrals.**

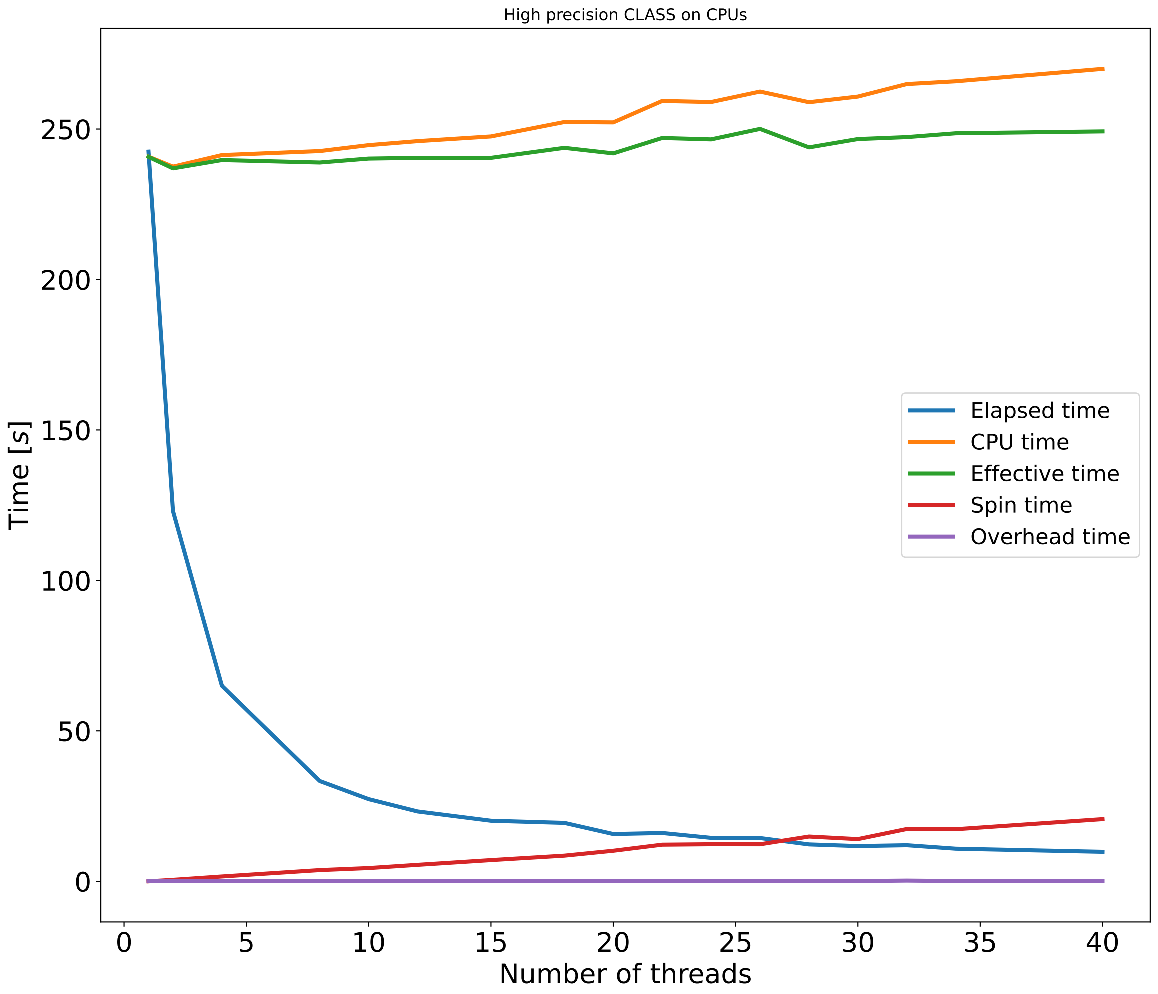
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Timescale, Milestones and KPIs



Accomplished Work, Results

- **Elapsed time:** wall time from the beginning to the end of collection
- **CPU time:** time during which the CPU is actively executing your application
- **Effective time:** CPU time spent in the user code. Does not include Spin and Overhead time.
- **Spin time:** Wait Time during which the CPU is busy
- **Overhead time:** CPU time spent on the overhead of known synchronization and threading libraries, such as OpenMP
- **CPU time = Effective time + Spin time + Overhead**



Accomplished Work, Results - 2

- **Profiling of the CLASS code using VTune tool**

Hotspots

Analysis Configuration | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Flame Graph | Platform

Grouping: Call Stack

Function Stack	CPU Time: Total	CPU Time: Self	Module	Function (Full)	Source File
Total	100.0%	0s			
_start	97.2%	0s	class	_start	start.S
__libc_start_main	97.2%	0s	libc.so.6	__libc_start_main	
main	97.2%	0s	class	main	class.c
perturbations_init	51.4%	0s	class	perturbations_init	perturbations.c
transfer_init	43.4%	0s	class	transfer_init	transfer.c
lensing_init	1.1%	0s	class	lensing_init	lensing.c
fourier_init	1.0%	0.012s	class	fourier_init	fourier.c
harmonic_init	0.4%	0s	class	harmonic_init	harmonic.c
output_init	0.0%	0s	class	output_init	output.c
thermodynamics_init	0.0%	0s	class	thermodynamics_init	thermodynamics.c
input_init	0.0%	0s	class	input_init	input.c
background_init	0.0%	0s	class	background_init	background.c
__clone	2.4%	0s	libc.so.6	__clone	
_INTERNAL1311483b::__kmp_wait_template<kmp_flag_64<(bool)0, (bool)1>, (bool)1, (bool)0, (b	0.3%	0.784s	libiomp5.so	_INTERNAL1311483b::__kmp_wait_template<kmp_fla...	kmp_wait_release.h
kmp_flag_native<unsigned long long, (flag_type)1, (bool)1>::notdone_check	0.0%	0.088s	libiomp5.so	kmp_flag_native<unsigned long long, (flag_type)1, (bo...	kmp_wait_release.h
sched_yield	0.0%	0.056s	libc.so.6	sched_yield	

Accomplished Work, Results - 3

- **Specifically in the transfer module we find:**

Hotspots

Analysis Configuration | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Flame Graph | Platform

Grouping: Call Stack

Function Stack	CPU Time: Total	CPU Time: Self	Module	Function (Full)	Source File
Total	100.0%	0s			
_start	97.2%	0s	class	_start	start.S
__libc_start_main	97.2%	0s	libc.so.6	__libc_start_main	
main	97.2%	0s	class	main	class.c
perturbations_init	51.4%	0s	class	perturbations_init	perturbations.c
transfer_init	43.4%	0s	class	transfer_init	transfer.c
[OpenMP fork]	42.6%	0s	libiomp5.so	__kmpc_fork_call	kmp.h
__kmp_fork_call	42.6%	0s	libiomp5.so	__kmp_fork_call	kmp_runtime.cpp
[OpenMP dispatcher]	42.6%	0s	libiomp5.so	__kmp_invoke_task_func	kmp_runtime.cpp
transfer_initompparallel@316	42.6%	0s	class	transfer_initompparallel@316	transfer.c
transfer_compute_for_each_q	42.6%	0.320s	class	transfer_compute_for_each_q	transfer.c
transfer_compute_for_each_l	41.6%	0.528s	class	transfer_compute_for_each_l	transfer.c
transfer_integrate	41.0%	1.852s	class	transfer_integrate	transfer.c
transfer_radial_function	38.5%	8.984s	class	transfer_radial_function	transfer.c
hyperspherical_Hermite4_interpolation_vector_Phi	15.8%	39.208s	class	hyperspherical_Hermite4_interpolation_vector_Phi	hyperspherical.c
hyperspherical_Hermite4_interpolation_vector_Phid2Phi	10.9%	26.912s	class	hyperspherical_Hermite4_interpolation_vector_Phid2Phi	hyperspherical.c
hyperspherical_Hermite4_interpolation_vector_dPhi	6.5%	16.003s	class	hyperspherical_Hermite4_interpolation_vector_dPhi	hyperspherical.c

Next Steps and Expected Results (by next checkpoint: April 2024)

- **Milestones:**
  1. First CLASS Run on GPUs using OpenMP pragmas
  2. First attempt at profiling CLASS on GPUs
- **KPIs:**
  3. First spectrum computed with GPUs
  4. Plot CLASS scaling on GPUs

Wavelet Phase Harmonics statistic

A.1. Bump-steerable wavelets

The WPH statistics rely on a bank of wavelets that are dilations and rotations of a mother bump-steerable wavelet $\boldsymbol{\psi}$ defined in Fourier space as follows (Mallat et al. 2020):

$$\hat{\boldsymbol{\psi}}(\boldsymbol{k}) = \exp\left(\frac{-(k-\xi_0)^2}{\xi_0^2-(k-\xi_0)^2}\right)\cdot \mathbf{1}_{[0,2\xi_0]}(k) \times \cos^{L-1}(\arg(\boldsymbol{k}))\cdot \mathbf{1}_{[0,\pi/2]}(|\arg(\boldsymbol{k})|), \quad \text{(A.1)}$$

with $k = \|\boldsymbol{k}\|$, $\mathbf{1}_A(x)$ being the indicator function that returns 1 if $x \in A$ and 0 otherwise, and $\xi_0 = 0.85\pi$ being the central wavenumber of the mother wavelet. The dilated and rotated wavelets are defined as follows:

$$\boldsymbol{\psi}_{\boldsymbol{\xi}_{j,\theta}}(\boldsymbol{r}) = 2^{-j}\boldsymbol{\psi}\left(2^{-j}\mathrm{rot}_{-\theta}[\boldsymbol{r}]\right), \quad \text{(A.2)}$$

with j being the index of the dilation by a factor 2^j and θ being the angle of rotation. We call $\boldsymbol{\xi}_{j,\theta} = 2^{-j}\xi_0\boldsymbol{u}_\theta$ with $\boldsymbol{u}_\theta = \cos\theta\,\boldsymbol{u}_x + \sin\theta\,\boldsymbol{u}_y$ the central wavevector of the wavelet obtained by such a transform. In practice, we consider J dilation indices j ranging from 0 to $J-1$, and we divide 2π into $2L$ evenly spaced rotation angles θ. The bank of wavelets is thus the set of $2JL$ wavelets $\{\boldsymbol{\psi}_{\boldsymbol{\xi}_{j,\theta}} : j = 0,\dots,J-1, \theta = 0, \frac{\pi}{L},\dots,\frac{(2L-1)\pi}{L}\}$, and these wavelets cover most of Fourier space with their respective band-passes.

In this Letter, we work with 512×512 maps and choose $J = 8$ and $L = 8$. We show in Fig. A.1 one example wavelet from the bank.

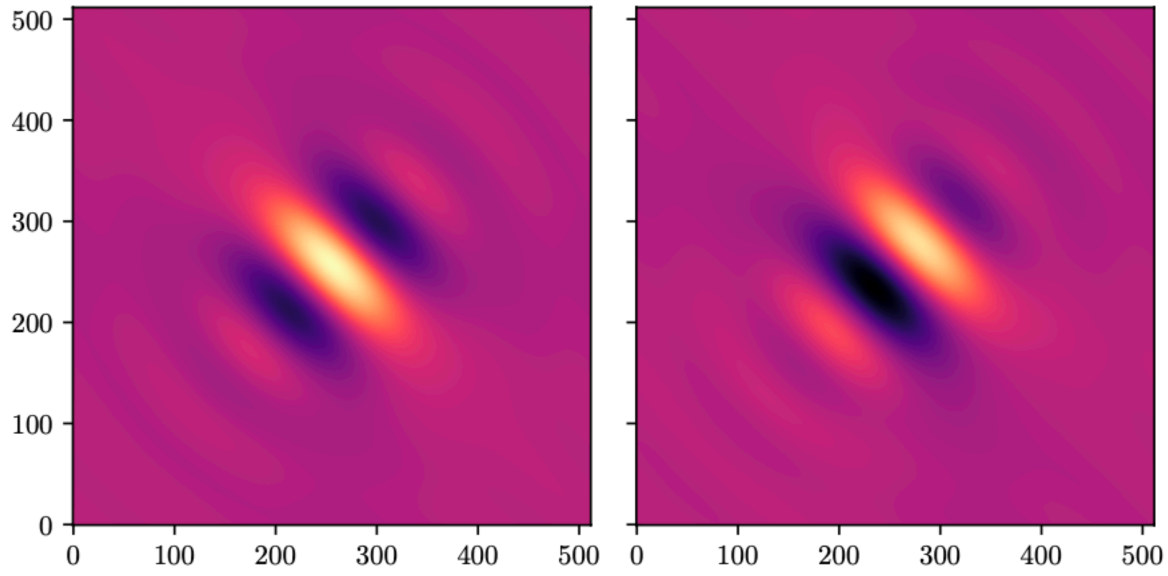


Fig. A.1. Real part (*left*) and imaginary part (*right*) of a 512×512 bump-steerable wavelet $\boldsymbol{\psi}_{6,\pi/4}$. The wavelet is centered in the middle of the map for a better visualization.