Machine-Learning-Based Forecasting of Cosmological Simulations using Next Generation Reservoir Computing

Master's Thesis at the Faculty of Physics
of the
Ludwig-Maximilian-University Munich

submitted by

Ali Assadollah

born in Witten on 24 October 2001

Supervisor:
PD Dr. Christoph Räth

Munich, 3 March 2026

Machine-Learning-basierte Vorhersage kosmologischer Simulationen mithilfe von Next Generation Reservoir Computing

Masterarbeit an der Fakultät für Physik
der
Ludwig-Maximilians-Universität München

vorgelegt von
Ali Assadollah
geboren in Witten am 24.10.2001

Betreuer:
PD Dr. Christoph Räth

München, den 03.03.2026

Acknowledgements

Long before this thesis took its final shape, my supervisor, Dr. Christoph Räth, provided constant guidance, encouragement, and thoughtful feedback, for which I am deeply grateful.
Under his supervision, I was given the freedom to develop my own ideas while benefiting from insightful advice and well timed suggestions whenever challenges arose.
Close collaboration with Sebastian Baur and Daniel Koeglmayr, who introduced me to the group and carefully reviewed my results, as well as the openness and stimulating discussions within the entire group that had a goal oriented yet welcoming atmosphere that I truly appreciated.
Yet above all, I owe heartfelt thanks to my family for their unrelenting emotional and financial support, without which pursuing research in these fascinating topics would not have been possible.

Contents

1 List of Symbols **3**

2 Motivation **6**

3 Theory **8**
3.1 Foundations of Cosmology 8
3.2 Power Spectrum 10
3.3 Cosmic Time 12
3.4 Minkowski Functionals and Tensors 13

4 Methods **15**
4.1 Reservoir Computing 15
4.2 Next Generation Reservoir Computing 16
4.3 Local States 18
4.4 Healpy 20
4.5 Hyperparameter Optimization 22

5 Results **24**
5.1 Next Generation Reservoir Computing on the Sphere 24
5.2 Adaptive Local States 27
5.3 Magneticum Pathfinder 45
5.4 COMPASS Simulation 61

6 Summary and Outlook **74**

Literature **75**

1 List of Symbols

- $a(t)$: scale factor of the universe, normalised so that $a(t_0) = 1$.
- t: cosmic time.
- k: spatial curvature parameter.
- $\mathrm{d}s^2$: line element of the FLRW metric.
- c: speed of light.
- $H(t)$: Hubble parameter (expansion rate at time t), defined as $\dot{a}/a$.
- H_0: present-day Hubble constant ($H(t_0)$).
- ρ: total energy density of the universe.
- p: total pressure of the cosmic fluid.
- Λ: cosmological constant (dark energy).
- ρ_c: critical density, $\rho_c = 3H_0^2/(8\pi G)$.
- Ω_i: density parameter of component i, $\Omega_i = \rho_i/\rho_c$.
- Ω_m: matter density parameter (includes baryonic + dark matter).
- Ω_r: radiation density parameter.
- Ω_Λ: dark energy density parameter.
- Ω_k: curvature density parameter.
- z: cosmological redshift, related to the scale factor by $a = 1/(1+z)$.
- G: Newton's gravitational constant.
- $D_+(t)$: linear growth factor.
- V: volume.
- A: area or wave amplitude.
- $R(t)$: scale factor of the universe .
- σ: mass surface density.
- $\mathbf{x}$: position in real space.
- $\mathbf{k}$: wave vector in Fourier space.
- $\rho(\mathbf{x}, t)$: mass density .
- $\delta(\mathbf{x}, t)$: mass density fluctuation.
- $\xi(r)$: radial two-point correlation function.
- $P(k)$: radial power spectrum.
- n_s: spectral index of primordial power spectrum.

- θ: polar angle.
- φ: azimuthal angle.
- $Y_{\ell m}(\theta, \varphi)$: spherical harmonics function.
- $\mathrm{d}\Omega$: solid angle element.
- $\Delta T(\theta, \varphi)$: temperature field.
- ℓ: angular scale.
- C_ℓ: angular power spectrum coefficient.
- $a_{\ell m}$: spherical harmonic coefficient.
- $W_0^{2,0}$: Minkowski Tensor.
- α: net orientation index.
- β: intrinsic anisotropy index.
- i, j: index for the pixel number.
- t_i, i: current time step.
- $\vec{v}_i, \vec{x}_{i,t_i}$: multi-dimensional time series at time t_i.
- $\vec{r}_i$: reservoir state at time t_i.
- $\mathbf{W}_{in/out}$: input/output matrix of the reservoir.
- $\mathbb{O}_{\mathrm{lin/nonlin},i}$: linear/non-linear feature vector at time t_i.
- k: delay depth of the Next Gen Reservoir Computing instance.
- s: delay spacing of the Next Gen Reservoir Computing instance.
- α: regularization parameter in the regression.
- c: wave velocity.
- u: wave displacement.
- ∇^2: spatial Laplacian.
- N_t: total amount time steps of a simulation.
- $l_{\mathrm{lin/nonlin}}$: number of elements in the linear/non-linear feature vector.
- $\mathrm{SSP}(\mathbf{x}_1, \mathbf{x}_2)$: Surface Similarity Parameter between $\mathbf{x}_1$ and $\mathbf{x}_2$.
- $F_{\mathbf{x}}(\mathbf{k})$: Fourier transformation of field $\mathbf{x}$.
- m: mass (distribution).
- μ: mean of an observable.
- σ: standard deviation of an observable.
- $\mathrm{RMSE}(t_i)$: root mean square error at time step t_i.
- σ: surface mass density field-

- ϕ: phase of coefficient of Fourier transform.
- D: gradient field of the Fourier phases.
- $f(D)$: probability density function of the gradient values.
- $S(t_i)$: entropy at time step t_i.

The index 0 in mathematical symbols (Ω_o, t_0, etc) is referring to the present day value of that quantity.

2 Motivation

From the beginning of time, humanity's gaze has been pulled skyward not merely in search of physical horizons but in search of understanding, meaning, and perspective. Just as the European navigators of the fifteenth century ventured across uncharted seas to discover "new worlds," so too does modern science push outward, into realms far beyond the reaches of Earth, probing the cosmos for deeper truths. Today, cosmology stands at the intersection of observation, mathematics, and philosophy, asking fundamental questions about the origin, evolution, and ultimate fate of the universe, and in doing so reflecting back on our own place within it.

Over the past century, cosmology has advanced dramatically, advanced by increasingly precise observations of cosmic microwave background radiation, the distribution of galaxies, and the accelerating expansion of the universe. Both theory and observation have unveiled a universe filled with phenomena that challenge our intuition, from dark matter and dark energy to the very beginning of space and time itself. These discoveries, in turn, lead to philosophical questions: What does it mean to explain the universe? The aim of this thesis is to contribute its small part to the exploration of these grand themes, from the mathematical frameworks that describe cosmic evolution to the conceptual foundations that shape our interpretation of them. However, any attempt to provide answers to these questions inevitably involves engaging with complex systems in one form or another. Complex Systems occur across various fields. Be it in natural science like the non-laminal flow of a fluid, in economics (for example the trend development in a free market) or in social sciences like the settlement development of urban areas. And while there is not a single ubiquitous accepted definition for complex system, one can still identify a few common characteristics (Hasselblatt and Katok, 2003).

1. They consist of multiple reciprocally interacting elements or actors.
2. They show non-linear behaviour, meaning that in many cases small changes in the input suffice to cause large disproportionate changes in the general behaviour of the system. This can, for example, happen when there is a feedback loop implemented in the system, which means that the output of a smaller part of the system will get redirected back to itself as a new input.
3. They often show emergent behaviour, which means that the system as a whole has properties that each single element does not have. This can often allow these systems to be perceived as pseudo-intelligent, since they can adapt to some degree even though each of their elements is not conscious at all.

Because of their widespread presence, it is of great interest to be able to functionally describe those systems and predict their behaviour as well as possible.
For this thesis, the complex system of interest is the development of mass density fields in the universe over time. These fields form the largest known structures in the universe and take the form of a complicated network called the cosmic web (Bond et al., 1996). The cosmic web also clearly exhibit the aforementioned properties. Every particle acts as a separate element of the whole, interacting with other particles, following simple physical laws. And yet small perturbations in the mass of a cluster can lead to the cluster becoming much larger than this initial change. Since the accumulated mass is responsible for its gravitational force and thus its ability to accrete more mass, a feedback loop is also present.

Since the fundamental mechanism governing the interaction of matter on cosmological scales, namely gravity, is well understood, a natural approach to predicting the evolution of the mass density field

is to perform numerical simulations. In practice, this is achieved through N-body techniques, where the matter distribution is represented by a large number of particles that evolve under gravitational forces within a specified cosmological model. Such simulations are of utmost importance for interpreting cosmological observations and for connecting theoretical predictions with measurable quantities.

However, high-resolution N-body simulations are computationally expensive and often require substantial processing time and memory resources (Teyssier et al., 2009). Generating a single realization can require millions of CPU hours and robust cosmological analyses typically require many simulations spanning different cosmological parameters. These practical limitations may result in simulations that are incomplete, restricted in volume, or lacking sufficient temporal or spatial resolution.

These challenges motivate the development of fast surrogate approaches. In particular, it is desirable to construct mock simulation data from partially available or lower-resolution simulations while preserving the essential statistical and structural properties of the true mass density field. Machine learning methods provide a powerful framework for this task. Rather than explicitly solving the underlying physical equations at every step, they learn effective dynamical mappings directly from the data. Such approaches do not require full knowledge of the exact governing equations, yet they can still reproduce the key characteristics of the physical system with significantly reduced computational cost (Lu et al., 2017), (Soo et al., 2023), (Li et al., 2025).

In this thesis, cosmological simulations are predicted using machine learning methods and subsequently evaluated against the corresponding reference simulations to assess their validity. One particular approach is known as Next Generation Reservoir Computing (NGRC) (Gauthier et al., 2021), which emerged as a further development of the classical Reservoir Computing (RC) framework (Jaeger, 2001). This machine learning architecture offers several advantages: it is less energy-intensive and demonstrates improved efficiency in predicting spatio-temporal time series. In addition, unlike classical RC, NGRC does not require a dedicated warm-up phase during training and demands less training data to achieve high-quality predictions.

To the best of the author's knowledge, this work presents the first implementation of the NGRC framework for data defined on a spherical surface pixelization scheme, which is then validated using simple wave simulations. Afterwards, a novel extension of the local states method (Pathak et al., 2018), named adaptive local states, is developed and verified using wave simulations on a planar two-dimensional domain. The newly established spherical NGRC framework is then applied to predict cosmological simulations defined on a spherical surface, and the resulting predictions are compared with the corresponding reference simulations. Finally, two-dimensional planar cosmological simulations are predicted using NGRC, and the synthetic results are again compared to the true simulations with the focus on capturing their non-linear dynamics.

3 Theory

3.1 Foundations of Cosmology

In the following the basic cosmological concepts used in this thesis are explained following Carroll and Ostlie (1996). Cosmology is the scientific discipline that tries to understand the universe as a whole: its origin, its evolution over time, its current content and structure, and its eventual fate. In contrast to studies of individual astronomical objects like stars or galaxies, cosmology focuses on the largest structures the universe has.
Cosmology combines general relativity, particle physics, and statistical mechanics with observational evidence, such as the expansion of the universe, the cosmic microwave background radiation, and the distribution of matter on cosmic scales, to reconstruct models to understand the cosmic history. Modern scientific cosmology has established that the universe began in an extremely hot, dense state approximately 13.8 billion years ago and has since expanded and evolved into the structured cosmos we observe today.
The Lambda–Cold-Dark-Matter model (ΛCDM model) is the currently favoured theoretical framework that describes the dynamics, composition, and expansion history of the universe (Blanchard et al., 2024), (Turner, 2022). It is based on the cosmological principle namely the assumption that on sufficiently large scales the universe is homogeneous and isotropic, and its geometry is described by the Friedmann–Lemaître–Robertson–Walker (FLRW) metric,

$$ds^2 = -c^2\,dt^2 + a^2(t)\left[\frac{dr^2}{1-k\,r^2} + r^2\left(d\theta^2 + \sin^2\theta\,d\phi^2\right)\right]. \tag{3.1}$$

Here, $a(t)$ denotes the scale factor of the universe, normalized to unity at the present time t_0, i.e.,

$$a(t) := \frac{R(t)}{R(t_0)}. \tag{3.2}$$

The function $R(t)$ represents the (dimensional) cosmic scale factor, describing the overall expansion of the universe. Substituting this metric into Einstein's field equations

$$G_{\mu\nu} + \Lambda\,g_{\mu\nu} = \frac{8\pi G}{c^4}\,T_{\mu\nu}, \tag{3.3}$$

with

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2}\,R\,g_{\mu\nu}. \tag{3.4}$$

yields the Friedmann equations, which govern the expansion of the universe:

$$H^2(t) := \left(\frac{\dot{a}}{a}\right)^2 = \frac{8\pi G}{3}\,\rho_{m,r} - \frac{k\,c^2}{a^2} + \frac{\Lambda\,c^2}{3}, \tag{3.5}$$

$$\frac{\ddot{a}}{a} = -\frac{4\pi G}{3}\left(\rho + \frac{3p}{c^2}\right) + \frac{\Lambda\,c^2}{3}, \tag{3.6}$$

To express the relative contributions of different components of the universe, one defines dimensionless density parameters with the critical density ρ_c at the present epoch t_0:

$$\Omega_{0,i} \equiv \frac{\rho_i(t_0)}{\rho_c}, \qquad \text{with} \quad \rho_c \equiv \frac{3H_0^2}{8\pi G},$$

In this notation, the first Friedmann equation can be rewritten as

$$H^2(z) = H_0^2 \left[\Omega_r(1+z)^4 + \Omega_m(1+z)^3 + \Omega_k(1+z)^2 + \Omega_\Lambda\right], \tag{3.7}$$

where the redshift z and the energy densities scale with $(1+z)$ according to their equations of state (e.g., $\rho_m \propto (1+z)^3$ for matter and $\rho_r \propto (1+z)^4$ for radiation).

In the flat ΛCDM approximation ($\Omega_k = 0$), these density parameters satisfy

$$\Omega_m + \Omega_r + \Omega_\Lambda = 1, \tag{3.8}$$

with contemporary measurements indicating $\Omega_\Lambda \sim 0.7$ and $\Omega_m \sim 0.3$ (Aghanim et al., 2020). ΛCDM thus provides a minimal parametrisation of the universe's expansion history and matter content that is consistent with a wide range of cosmological observations.

Next it is examined how small initial density fluctuations evolve under gravity into the galaxies, clusters, and cosmic web observed today. These fluctuations are described by the density contrast

$$\delta(\mathbf{x}, t) := \frac{\rho(\mathbf{x}, t) - \bar{\rho}(t)}{\bar{\rho}(t)}, \tag{3.9}$$

where $\rho(\mathbf{x}, t)$ is the local matter density and $\bar{\rho}(t)$ is the mean density of the universe.

In the very beginning of the universe, the mass (both baryonic as well as dark matter) was distributed randomly in a Gaussian field. These initial fluctuations later grew rapidly during the inflationary phase (Linde). The dark matter component started earlier with the gravitational collapse, while the baryonic component started only after decoupling from photons when the last scattering surface was emitted (Padmanabhan, 1993).
In the linear regime ($|\delta| \ll 1$), perturbations grow according to a second-order differential equation that describes the competition between gravitational attraction and cosmic expansion. On sub-horizon scales in a matter-dominated universe, the linear growth equation is

$$\ddot{\delta} + 2H(t)\,\dot{\delta} - 4\pi G\,\bar{\rho}\,\delta = 0. \tag{3.10}$$

The term $2H\,\dot{\delta}$ represents the damping effect of cosmic expansion, while $4\pi G\,\bar{\rho}\,\delta$ drives gravitational growth.

In an Einstein–de Sitter universe (pure matter with $\Omega_m = 1$), the growing solution scales as

$$\delta_+(t) \propto a(t), \tag{3.11}$$

i.e., the density contrast grows in proportion to the scale factor $a(t)$. To characterise growth more generally, one defines the linear growth factor $D(a)$,

$$D(a) \equiv \frac{\delta(a)}{\delta(a_{\text{init}})}, \tag{3.12}$$

normalised so that $D(a = 1) = 1$ today. A related quantity is the growth rate

$$f(a) \equiv \frac{d \ln D}{d \ln a}, \tag{3.13}$$

which measures how fast structures are growing at a given epoch. Once dark energy dominates the cosmic energy budget, the Hubble expansion accelerates, increasing the friction term in the growth equation and suppressing the growth of perturbations relative to an Einstein–de Sitter universe.

3.2 Power Spectrum

Next the power spectrum is going to be introduced which is a important statistical tool used to characterise the distribution of matter across spatial scales in the universe. Starting from the matter overdensity defined in equation (3.9) its Fourier transform is defined as,

$$\tilde{\delta}(\mathbf{k}, t) = \int d^3x \, \delta(\mathbf{x}, t) \, e^{-i\mathbf{k}\cdot\mathbf{x}}, \tag{3.14}$$

where $\mathbf{k}$ is the comoving wavevector. Or in the case of finite data, i.e. without infinitely continuous or periodic data, the discrete Fourier transform is defined as

$$\tilde{\delta}(\mathbf{k}, t) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \delta(\mathbf{x}, t) \, e^{-2\pi i(\mathbf{k}\cdot\mathbf{x})} \tag{3.15}$$

In other words, one assume here that the distribution represents one period of periodic data. The data is wrapped around the x- and y-axes, so to speak.
Using the continous Fourier transform he matter power spectrum $P(k, t)$ is defined via

$$\left\langle \tilde{\delta}(\mathbf{k}, t) \, \tilde{\delta}^*(\mathbf{k}', t) \right\rangle = (2\pi)^3 \, \delta^{(3)}(\mathbf{k} - \mathbf{k}') \, P(k, t), \tag{3.16}$$

which quantifies the variance of the density field as a function of scale. The two-point correlation function $\xi(r)$ defined as

$$\xi(r) \;=\; \langle \delta(\mathbf{x}_1) \, \delta(\mathbf{x}_2) \rangle \quad \text{with} \quad r = |\mathbf{x}_1 - \mathbf{x}_2|.$$

relates to the power spectrum by

$$\xi(r) = \int \frac{d^3k}{(2\pi)^3} \, P(k) \, e^{i\mathbf{k}\cdot\mathbf{r}}, \tag{3.17}$$

and a dimensionless form is given by

$$\Delta^2(k) = \frac{k^3}{2\pi^2} \, P(k), \tag{3.18}$$

which represents the contribution to variance per logarithmic interval in k. There are also equivalent to the power spectrum functions to describe higher than two-point correlations like the bispectrum

(for three-point correlations). In linear theory, the time evolution of the matter power spectrum is governed by the growth factor $D_+(t)$,

$$P(k,t) = D_+^2(t)\, P(k,t_0), \tag{3.19}$$

where $P(k,t_0)$ is the spectrum at a reference epoch. A common parametrisation of the primordial spectrum is a power law,

$$P_0(k) \propto A_s\, k^{n_s}, \tag{3.20}$$

with amplitude A_s and spectral index n_s. The shape and amplitude of $P(k)$ are sensitive to cosmological parameters such as the matter density Ω_m and the spectral index n_s. By computing $P(k)$ for a given cosmological model and comparing to observational estimates from galaxy clustering, weak lensing, or other large-scale structure probes, one can perform a likelihood analysis to infer posterior constraints on the parameters (Bolliet et al., 2018)(Bridle et al., 2003). This makes the power spectrum a central observable in precision cosmology.

Now, in many cosmological observations (such as the CMB temperature anisotropy), the observed field is on a spherical manifold, motivating a different approach to calculate a power spectrum. The scalar field over the celestial sphere can be decomposed into spherical harmonics $Y_{\ell m}(\theta,\varphi)$, which form a complete and orthonormal set of functions, i.e., a basis:

$$\int_{S^2} Y_{\ell m}(\theta,\varphi)\, Y^*_{\ell' m'}(\theta,\varphi)\, \mathrm{d}\Omega = \delta_{\ell\ell'}\, \delta_{mm'}, \tag{3.21}$$

where $d\Omega = \sin\theta\, d\theta\, d\varphi$.

A scalar field on the sphere, for example the temperature fluctuations of the CMB $\Delta T(\theta,\varphi)$, can thus be expanded as

$$\Delta T(\theta,\varphi) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} a_{\ell m}\, Y_{\ell m}(\theta,\varphi), \tag{3.22}$$

with coefficients $a_{\ell m}$ that quantify the amplitude of each spherical harmonic mode. The angular power spectrum C_ℓ is defined as the variance of these coefficients at a fixed multipole moment:

$$C_\ell := \frac{1}{2\ell+1} \sum_{m=-\ell}^{\ell} \langle\, |a_{\ell m}|^2 \,\rangle, \tag{3.23}$$

and for a statistically isotropic field,

$$\langle\, a_{\ell m}\, a^*_{\ell' m'} \,\rangle = C_\ell\, \delta_{\ell\ell'}\, \delta_{mm'}. \tag{3.24}$$

The angular power spectrum characterises how the variance of the field is behaving as a function of angular scale, with lower ℓ corresponding to larger scale variations and higher ℓ to finer angular features. The two-point correlation function on the sphere,

$$C(\theta) := \langle\, \Delta T(\hat{n}_1)\, \Delta T(\hat{n}_2) \,\rangle, \tag{3.25}$$

with θ the angular separation between directions $\hat{n}_1$ and $\hat{n}_2$, is related to C_ℓ via

$$C(\theta) = \sum_{\ell=0}^{\infty} \frac{2\ell+1}{4\pi}\, C_\ell\, P_\ell(\cos\theta), \tag{3.26}$$

which shows that C_ℓ and the correlation function are spherical analogues of the Fourier power spectrum and two-point correlation in flat space. The statistical properties of a Gaussian random field on the sphere, such as the primary CMB anisotropies, can be fully described by the angular power spectrum C_ℓ.

3.3 Cosmic Time

Because light travels at a finite speed, one does not see distant astronomical objects as they are now, but as they were when the light one observes was emitted. This time delay means that the further away an object is, the further back in time we are looking when we observe it. To formalize this, cosmologists use the concept of lookback time, which quantifies exactly how far back in the history of the universe we are seeing an event. The lookback time $t_L(z)$ to an object at redshift z is defined as the difference between the present age of the universe t_0 and the age of the universe $t_e(z)$ when the light was emitted:

$$t_{\mathrm{L}}(z) := t_0 - t_{\mathrm{e}}(z). \tag{3.27}$$

Comsological simulations often times only give the redshift as the time stemp of their snapshots. To relate lookback time to redshift, one uses the fact that in an expanding FLRW universe the scale factor $a(t)$ and redshift z are related by

$$1 + z = \frac{1}{a(t)}, \tag{3.28}$$

where, by convention, the scale factor today is $a(t_0) = 1$. Because the expansion history of the universe determines the relationship between cosmic time and redshift, computing the lookback time requires integration over the expansion rate parameterised by the Hubble parameter $H(z)$.

Starting from the definition of cosmic time through the scale factor,

$$dt = \frac{da}{a\,H(a)}, \tag{3.29}$$

and substituting $a = 1/(1 + z)$, one obtains the differential relation

$$dt = -\frac{dz}{(1+z)\,H(z)}. \tag{3.30}$$

The lookback time to redshift z then follows from integrating this expression from $z = 0$ (today) to the redshift of interest z:

$$t_{\mathrm{L}}(z) = \int_0^z \frac{dz'}{(1+z')\,H(z')}. \tag{3.31}$$

This expression depends on the cosmological model through the functional form of the expansion rate $H(z)$. In general, this integral does not have a closed-form solution for arbitrary cosmological parameters and must be evaluated numerically. The result is that lookback time increases with redshift and asymptotes to the age of the universe as $z \to \infty$. The lookback time directly corresponds to the amount of cosmic time that has passed since the light was emitted. For example, a galaxy observed at $z \simeq 3$ has a lookback time of over $\sim$12 Gyr in typical flat ΛCDM cosmologies, meaning we are observing it as it existed more than 12 billion years ago when the universe was only a small fraction of its current age.
Because lookback time depends on the full expansion history $H(z)$, it provides a complementary probe to other cosmological measures such as teh luminosity distance or the angular diameter distance and is sensitive to the values of $\Omega_{m,0}$, $\Omega_{\Lambda,0}$, and H_0.

3.4 Minkowski Functionals and Tensors

There are many statistical descriptors beyond two-point statistics that are essential for capturing non-Gaussian features and the morphology of cosmic structures such as the skewness, which quantifies the asymmetry of a distribution (a non-zero skewness indicates that positive and negative fluctuations are not equally likely) or kurtosis, which describes the tailedness of a distribution (meaning that extreme values occur more frequently than in a Gaussian model) (Ben-David et al., 2015). Minkowski Functionals (MFs) and their tensorial generalisations, known as Minkowski Tensors (MTs), provide a mathematical framework to quantify the shape, connectivity, curvature, and anisotropy of spatial patterns in cosmological fields. These descriptors originate from integral geometry and have been applied to the analysis of large-scale structure, CMB maps, and weak gravitational lensing fields, such as for example the detection of bubble-like interstellar structures in optical emission line images (Collischon et al., 2021).
For a field defined on a domain, one constructs excursion sets by selecting regions where the field is above a threshold value. In d dimensions, Hadwiger's theorem guarantees the existence of exactly $d+1$ independent scalar Minkowski Functionals that completely characterise the morphology of the excursion sets. For example, in three dimensions there are four scalar functionals interpreted geometrically as volume, surface area, mean curvature, and Euler characteristic. For a two-dimensional field, the Minkowski Functionals are defined as

$$V_0 = \frac{1}{A}\int_{Q(\nu)} dA, \quad V_1 = \frac{1}{4A}\int_{\partial Q(\nu)} d\ell, \quad V_2 = \frac{1}{2\pi A}\int_{\partial Q(\nu)} \kappa\, d\ell, \tag{3.32}$$

where A is the domain area, $\partial Q(\nu)$ is the boundary of the region above threshold ν, and κ denotes curvature (Appleby et al., 2018).

One can further generalize Minkowski Functionals to Minkowski Tensors that capture not only the overall morphology of a spatial structure but also its orientation and anisotropy. Unlike scalar functionals, which integrate scalar geometry measures, Minkowski Tensors weight these measures with tensor products of vectors such as position ($\mathbf{r}$) and the normal ($\mathbf{n}$) on the boundary of the structure. In three dimensions, second-rank Minkowski Tensors in the Cartesian representation are defined as

$$W_0^{2,0} = \int_K \mathbf{r}\otimes\mathbf{r}\, dV, \tag{3.33}$$

$$W_1^{2,0} = \frac{1}{3}\int_{\partial K} \mathbf{r}\otimes\mathbf{r}\, d\mathcal{O}, \tag{3.34}$$

$$W_2^{2,0} = \frac{1}{3}\int_{\partial K} H\,\mathbf{r}\otimes\mathbf{r}\, d\mathcal{O}, \tag{3.35}$$

$$W_1^{0,2} = \frac{1}{3}\int_{\partial K} \mathbf{n}\otimes\mathbf{n}\, d\mathcal{O}, \tag{3.36}$$

$$W_2^{0,2} = \frac{1}{3}\int_{\partial K} H\,\mathbf{n}\otimes\mathbf{n}\, d\mathcal{O}, \tag{3.37}$$

where H is the mean curvature on the surface ∂K of the excursion set K, dV and $d\mathcal{O}$ are volume and surface area measures, and $\otimes$ denotes the symmetric tensor product. These definitions extend the

scalar measures such as volume, surface area, and integrated curvature to tensorial shape descriptors. The eigenvalues λ_i of a second-rank Minkowski Tensor yielded via

$$\begin{pmatrix} W^{00} & W^{01}\sin^2(\theta) \\ W^{10} & W^{11}\sin^2(\theta) \end{pmatrix} \begin{pmatrix} \nu^0 \\ \nu^1 \end{pmatrix} = \lambda \begin{pmatrix} \nu^0 \\ \nu^1 \end{pmatrix} \text{ with } W^{ij} = (W_1^{0,2})^{ij} \tag{3.38}$$

describe the relative spread of structural features along different directions, and thus encode information about the anisotropy of geometric features (Collischon et al., 2024). For instance, in two dimensions, one can define a net orientation index α and an intrinsic anisotropy index β via

$$\alpha \equiv \frac{\langle \lambda_{\max} \rangle}{\langle \lambda_{\min} \rangle}, \qquad \beta \equiv \left\langle \frac{\lambda_{\max}}{\lambda_{\min}} \right\rangle, \tag{3.39}$$

where angle brackets denote averaging over individual structures. Since $W_1^{0,2}$ is symmetric (i.e. its eigenvectors are orthogonal), for a statistically isotropic field, such as the cosmic microwave background under ΛCDM, one expects $\alpha \approx 1$ within uncertainty otherwise it hints at a preferred orientaiton of the structure.
Now that the basic theoretical concepts used in this work were introduced, the code and methods that are used to produce the results can be explained.

4 Methods

As a subdiscipline of artificial intelligence (AI), machine learning (ML) includes methods that try to make inferences from given data. Often, the advantage of these methods lies in the fact that one does not need to (fully) know a priori the exact model that describes the system Lu et al. (2017). With recent progress in machine learning, a new set of approaches has become available to analyse, predict, and control complex systems. One specific design of these machine learning algorithms is Reservoir Computing (RC), originally formulated in 2001 (Jaeger (2001)). Here, one makes use of a network, also called a "reservoir". Currently, Reservoir Computing is one of the machine learning algorithms best suited for dynamical systems using observed time-series data (Gauthier et al. (2021)).

4.1 Reservoir Computing

Although classical Reservoir Computing was not used for the results presented in this thesis, it is nonetheless helpful to familiarize ourselves with the computational principles of reservoir computing, as we will later introduce the Next-Generation Reservoir Computing framework which will be used for the results presented in Chapter 5. It is expected that less energy-demanding architectures, like Reservoir Computing, will play an increasingly important role in the future, especially in computationally expensive applications as well as applications where energy supply is heavily limited (for example in planes, satellites, etc.) (Meyer, 2024). Another advantage of Reservoir Computing is that it proves to be surprisingly efficient in predicting chaotic time series (Chattopadhyay et al., 2020). The RC algorithm works as follows: Given a multi-dimensional discrete time series (for demonstrative purposes we have chosen 3 dimensions):

$$\vec{v}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix},$$

where i is the time step index of the time series.

We proceed to introduce a network called the reservoir $\mathbf{A}$, in which the data is fed into, coupled by a input matrix $\mathbf{W}_{in}$.
Both the input matrix as well as the reservoir are static in the sense that they are generated once in the beginning and stay fixed throughout the training, which makes Reservoir Computing once again very computationally cheap. The reservoir state $\vec{r}_i$ for the next time step is then calculated according to

$$\vec{r}_{i+1} = \mathbf{A}\,\vec{r}_i + \mathbf{W}_{in}\,\vec{v}_i\ . \tag{4.1}$$

Now, this equation alone would not be able to capture non-linearities. For this reason, one usually takes the right-hand side of equation (4.1) as the argument of a non-linear function. The tanh activation function is particularly suitable in this context because it maps input values to the bounded interval $(-1, 1)$. This produces zero-centred outputs, which help prevent systematic bias in the activations and promote more stable and balanced learning dynamics.

$$\vec{r}_{i+1} = \tanh(\mathbf{A}\,\vec{r}_i + \mathbf{W}_{in}\,\vec{v}_i). \tag{4.2}$$

Usually, the adjacency matrix of a random network is used for the reservoir $\mathbf{A}$, with non-zero entries drawn randomly from the interval $[-b, b]$ with $b \in \mathbb{R}$, and the network connectivity fixed from the beginning. The reservoir states are then multiplied with the output matrix $\mathbf{W_{out}}$

$$\vec{v}_{i+1,RC} = \mathbf{W_{out}}\, \vec{r}_{i+1}, \tag{4.3}$$

and then the difference to the real time series $\vec{v}_{i+1}$ is minimized via Ridge Regression (van Wieringen) so that the output matrix is calculated via

$$\sum_{-T \leq i \leq 0} ||\vec{v}_{i,RC} - \vec{v}_i||^2 + \alpha ||\mathbf{W_{out}}||^2, \tag{4.4}$$

where $||\mathbf{W_{out}}||^2$ stands for the sum of the squares of all the matrix elements and the parameter α has the purpose of preventing overfitting (the larger the parameter the less accurate the fit becomes). Matrices $\mathbf{V}$ and $\mathbf{R}$ can be constructed by stacking all the vectors $\vec{v}_i$ and $\vec{r}_i$ within a training time interval $-T \leq i \leq 0$. Then $\mathbf{W_{out}}$ can be calculated via

$$\mathbf{W_{out}} = \mathbf{V}\,\mathbf{R}\left(R\,R^T + \alpha \mathbf{I}\right)^{-1}. \tag{4.5}$$

Additionally there is the possibility to concatenate a constant term to the reservoir vector

$$\tilde{\vec{r}}_i = \begin{pmatrix} \vec{r}_i \\ 1 \end{pmatrix}, \tag{4.6}$$

which updates equation (4.3) with a bias capturing feature to become

$$\vec{r}_{i+1,RC} = \mathbf{W_{out}}\, \tilde{\vec{r}}_i = \mathbf{W_{out}}\, \vec{r}_i + \vec{b}\,. \tag{4.7}$$

Breaking the symmetry ensures that the dynamics do not accidentally produce a reflected version of the trajectory in the opposite region of state space, a phenomenon known as a mirror attractor (Herteux and Räth, 2020).

4.2 Next Generation Reservoir Computing

In this thesis, the Next Generation Reservoir Computing (NGRC) machine learning framework is used for the results presented in Chapter 5. This choice was mainly due to the limited amount of data available for the systems analysed both in terms of spatial and temporal resolution. In contrast to classical Reservoir Computing, NGRC at its core creates features using a non-linear expansion of time-shifted input data, hence eliminating the need for randomly connected recurrent networks while require less hyperparameters to be optimized. Recent work has demonstrated that NGRC requires significantly less training data than traditional machine learning approaches while still capturing dynamical behavior effectively (Gauthier et al., 2021; Barbosa and Gauthier, 2022; Haluszczynski et al., 2023; Köglmayr and Räth, 2024). Furthermore, Next Generation Reservoir Computing offers

better interpretability in the sense that the features in combination with the optimized output layer, can be used for identifying the underlying equations of dynamical systems, via its trained weights.

This framework creates its reservoir state vectors, first through a linear time-delay feature vector $\mathbb{O}_{\text{lin}}$ analogous to takens time delay embedding (Takens, 1981) with hyperparameters k and s, where k stands for the number of past data points taking into this state vector, and s defining the time interval between those data points.

$$\mathbb{O}_{\text{lin}} = \begin{pmatrix} \vec{v}_i \\ \vec{v}_{i-1s} \\ \cdots \\ \vec{v}_{i-ks} \end{pmatrix}.$$

We will later see that an optimal choice of hyperparameters especially that of k and s are essential for accurate predictions as well a fast training and prediction of the NGRC. In NGRC, the linear feature vector is used to calculate an additional non-linear feature vector O_{nonlin} by calculating unique monomials of elements of the linear feature vector up to a given degree. For example, when the degree is set to two for a linear feature vector of a three dimensional system with hyperparameters $k = 2$ and $s = 1$ the non-linear feature vector contains all unique monomials of order two, which is equivalent to the Kronecker product of the linear feature vector

$$\mathbb{O}_{\text{nonlin},i} = \mathbb{O}_{2,i} = \mathbb{O}_{\text{lin},i} \otimes \mathbb{O}_{\text{lin},i} = \begin{pmatrix} x_i x_i \\ x_i x_{i-1} \\ x_i y_i \\ \cdots \\ x_{i-1} x_{i-1} \\ x_{i-1} y_i \\ z_i z_{i-1} \\ z_{i-1} z_{i-1} \end{pmatrix}. \tag{4.8}$$

To get the final feature vector, the linear and non-linear feature vectors are concatenated for each time step of the time series

$$\mathbb{O}_{tot,i} = \begin{pmatrix} \mathbb{O}_{\text{lin},i} \\ \mathbb{O}_{\text{nonlin},i} \end{pmatrix}.$$

Similar to the traditional Reservoir Computing this feature vector is trained to predict the next point in time using ridge regression, so that

$$\vec{v}_{i+1} = \vec{v}_i + \mathbf{W_{out}}\, \mathbb{O}_{tot,i}. \tag{4.9}$$

The matrix $\mathbf{W_{out}}$ is trained to learn the difference between the current and the next time step and it is determined via Tikhonov regularization (see equation (4.4)). Since in Next Generation Reservoir Computing, the model does not rely on a large random recurrent reservoir but instead builds an explicit set of polynomial features from time-delay embedded input data each weight of the output

Matrix $\mathbf{W_{out}}$ directly corresponds to a specific term in the feature library. This means that the learned weights can be interpreted as coefficients of a data-driven surrogate model that reflects the structure of the underlying dynamics, giving a rough sense of how the governing equations might look.
Throughout this thesis, the monomials used to construct the feature vectors are of first and second order (4.8). Additionally, a bias element is consistently included in the feature vector (4.7). Although these details will not be explicitly repeated later, they are always implicitly assumed.

4.3 Local States

As was illustrated with the three dimensional vector $\vec{v}_i$ the time series can generally be of any dimension. Put in other terms, one can also use normal Reservoir Computing as well as Next Generation Reservoir Computing to predict a (pixelized) map that is evolving in time where every pixel represents its own dimension. The computational resources needed for predicting such a map increase super-linearly with the number of pixels. How exactly the demanded resources increase is dependent on the degree of the monomials of $\mathbb{O}_{nonlin,i}$ as well as the exact choice of the hyperparameters k and s. To solve this problem, one can introduce an own NGRC instance for every dimension/pixel and its locally closest pixels (Pathak et al., 2018; Barbosa and Gauthier, 2022). Thus, for N dimensions we end up with N reservoirs, each of which only predicts the next time step of a single core dimension while being fed with the previous time steps of the core dimension/pixel and its neighbouring dimensions/pixels.

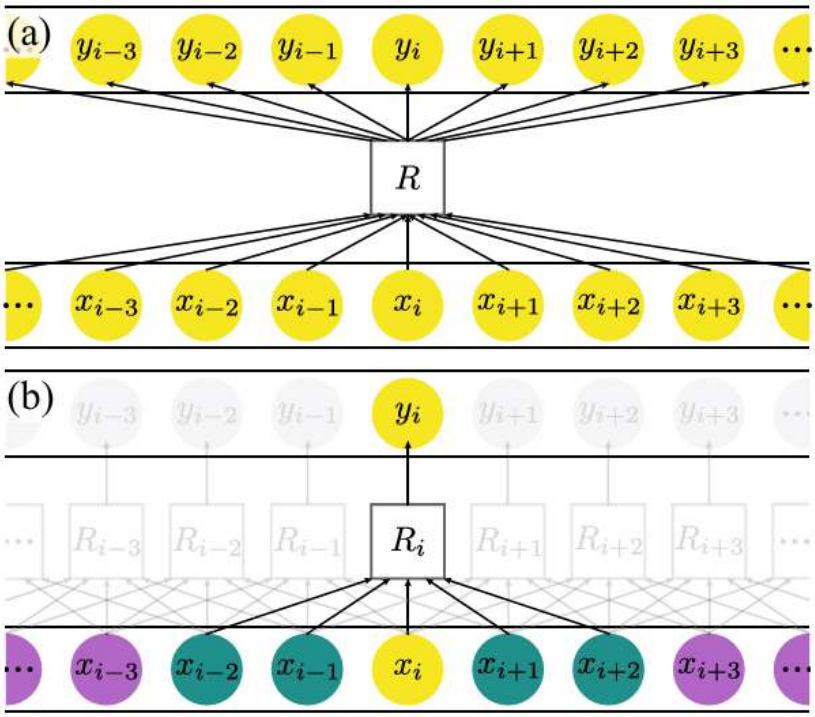


Fig. 1: Schematics showing the basic concept of normal Reservoir Computing (a) and Reservoir Computing using local states (b). In normal Reservoir Computing, the current time steps of all dimensions $(x_{i-3}, ..., x_i, ..., x_{i+3})$ are fed into a single reservoir to predict the next time step of all dimensions $(y_{i-3}, ..., y_i, ..., y_{i+3})$ together. In the local states approach, only the current time step of the core and its neighbours is fed into one of many reservoirs, each predicting the next time step of the corresponding core dimension. The schematic itself is sourced from (Baur and Räth, 2021)

The implementation of local states allows the required computational power to scale linearly with the number of dimensions or map pixels. Depending on the exact choice of neighbours for a given core, the quality of the prediction and the required computations can vary greatly. One can furthermore choose solely the neighbours of a core dimension based on the quality of prediction, regardless of their spatial proximity to the core; this would be called generalized local states (Baur and Räth, 2021). Since this approach was not used because all interaction all entirely spatial, no further detailed explanation is provided.
Since the reservoirs are enumerated "one dimensionally" with i as can be seen in 1 one only needs to find a function that maps the pixel index of a map progressing in time of any dimension to predict this map. In the classic flat two dimensional $(N_x \times N_y)$ pixel grid the task of finding such a function as well as the neighbouring pixels to a given pixel core is an easy one. The conversion of the two dimensional Index of a pixel $x_{i,j}(t_k)$ two a one dimensional one $x'_{i'}(t_k)$ is simply

$$i' = i + j \cdot N_y \tag{4.10}$$

As for the local states, the neighbouring pixel indices of a core index i, j can be determined via

$$Neighbour(i,j) = \{(i-1,j-1),(i,j-1),(i-1,j-1), \tag{4.11}$$
$$(i-1,j),(i-1,j),(i-1,j+1),(i,j+1),(i-1,j+1)\} \tag{4.12}$$

and then with (1) their one dimensional index can be obtained. The resulting neighbours can be represented by a matrix.

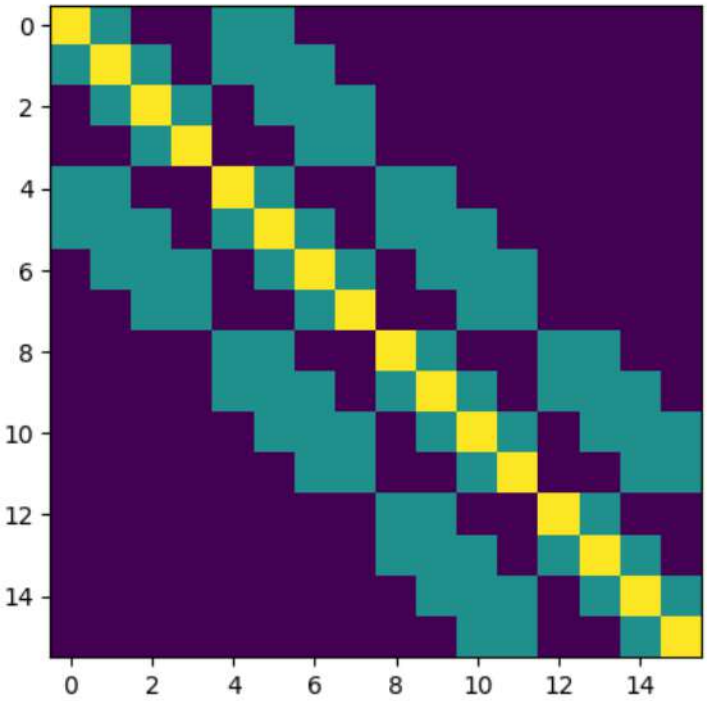


Fig. 2: Local states Matrix for a 4x4 pixel map. The axis show the pixel index of each pixel (going from 0 to 15). The Core pixel is shown as in yellow in each row while the turquoise pixel represent the neighbour pixel. Notably all pixels outside the boundaries of the 4×4 map are not taken into account.

For our purposes, it is of interest to enumerate all pixels of a spherical map, similar to above, with a one-dimensional index and to find the neighbours in spatial (or rather angular) proximity to a given core. Before presenting the solution used in this work, a brief introduction to `Healpy` is sensible.

4.4 Healpy

The study of functions defined on spherical domains plays a fundamental role in the physical sciences and engineering. This importance is especially evident in areas such as astronomy, cosmology, geophysics, and atomic as well as nuclear physics. `Healpy` offers a pixelation scheme the Hierarchical

Equal Area isoLatitude Pixelization (`HEALPix`) embedded in Python (`Healpy`) with which it is possible to visualize spherical maps as well to perform transformations on them (Górski et al. (2024)). As the name implies, `HEALPix` is hierarchically structured, meaning the full sky is recursively subdivided into smaller areas. This allows, first, easier downsampling of the resolution of a given map, and second, computational advantages, since spatially close areas are also close in the tree structure of the database.

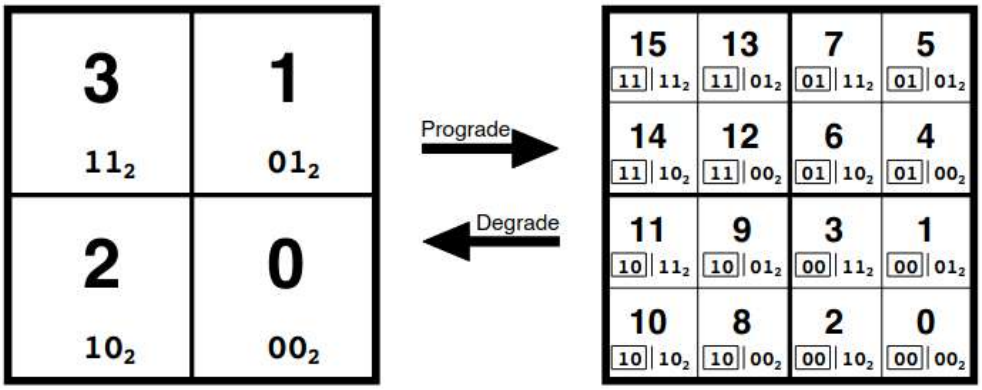


Fig. 3: Quadrilateral tree pixel indexing: The coarse coordinate patch on the left consists of four pixels, which can be labeled using just two bits. To achieve higher resolution, each pixel is subdivided into four "daughter" pixels, as shown on the right. These daughter pixels inherit the index of their parent (highlighted in a box) and are assigned two additional bits to form a new, unique index. Multiple such curvilinearly mapped patches, twelve in the case of `HEALPix`, are joined at their edges to cover the entire sphere. Each pixel's index also contains a prefix (not shown here) indicating the base-resolution pixel from which it originates. The schematic is sourced from https://healpix.sourceforge.io/pdf/intro.pdf.

The next important property of `HEALPix` is that every element of the full sky partition is of equal area. This ensures that no region on the sky is neither over nor under-represented though the shapes of the elements may differ from each other as this is unavoidable with every full sky partition.
Finally, the last relevant property is that the center of every pixel element share a latitude with other pixel center. The advantage of this setup is that later when calculating the spherical harmonics for a given pixel map it is no longer necessary to calculate the Legendre polynomial $P_l^m(\cos(\theta))$ of every pixel but only of the fewer latitudes on which several pixel center are lying.
The HEALPix pixelization starts off with 12 elements each of which can be further subdivided into a power 2.

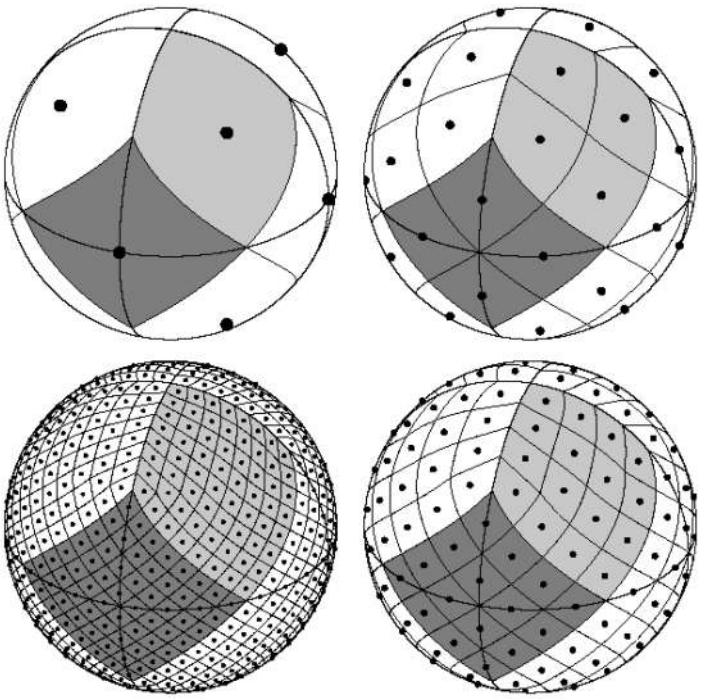


Fig. 4: Orthographic view of the `HEALPix` partition of the sphere: The overlaid equator and meridians illustrate the octahedral symmetry of `HEALPix`. Light-gray shading highlights one of the eight identical polar base-resolution pixels (four in the north, four in the south), while dark-gray shading highlights one of the four identical equatorial base-resolution pixels. Moving clockwise from the upper-left panel, the grid is hierarchically subdivided with the resolution parameter set to $N_{\text{side}} = 1, 2, 4, 8$, resulting in a total number of pixels $N_{\text{pix}} = 12 \times N_{\text{side}}^2 = 12, 48, 192, 768$. All pixel centers lie on $N_{\text{ring}} = 4 \times N_{\text{side}} - 1$ rings of constant latitude. Within each panel, all pixels have identical areas. The schematic is sourced from https://healpix.sourceforge.io/pdf/intro.pdf.

4.5 Hyperparameter Optimization

As was already mentioned, the chosen hyperparameters of the NGRC play an essential role in the quality of the prediction. The most naive way to determine these hyperparameters would be to define a loss function in order to quantify the quality of the prediction and then start a grid search, meaning one creates (in the case where we are looking for the hyperparameters k, s, α) a three-dimensional matrix with triples of hyperparameters in a given interval. Afterwards, one simply starts the training and prediction for every combination of hyperparameters and then takes the triple with the best result according to the cost function. This method has several disadvantages. For one, it requires a lot of computational resources/time, the higher dimensional the matrix of hyperparameters becomes, not to mention that a run with a single set of hyperparameters also takes a long time until it provides its prediction results. Furthermore, the sets of hyperparameters are distributed equally over the multidimensional space, meaning the promising areas are not scanned more thoroughly than areas in which it is unlikely to find suitable hyperparameters.

These problems can be avoided with more elaborate algorithms, which are implemented in various Python packages, one of them being `Optuna` (Akiba et al., 2019). It offers several methods for finding the optimal hyperparameters, but we will focus on the one used here, which is the Tree-structured Parzen Estimator (TPE). It works as follows: At the beginning, random sets of hyperparameters are generated and the cost function is applied to the predictions with these hyperparameters. Then the sets are assigned either to a suitable category of hyperparameters or a category for bad hyperparameters. For the good hyperparameters, a Gaussian kernel distribution is overlaid to get a density function describing the "landscape" of the loss function. The same is done with the bad hyperparameters. In the second phase, new sets of hyperparameters are generated in places where the distribution value of good hyperparameters is high and the distribution value of bad hyperparameters is low. This method allows a balanced approach in terms of exploitation (the information of the distribution of good and bad parameters is used to generate new hyperparameter sets) as well as exploration (in the initial phase, sets of hyperparameters are randomly generated without using prior knowledge). One last advantage `Optuna` offers is the possibility to run several hyperparameter sets in parallel, which again speeds up the whole optimisation process.

5 Results

After these introductory remarks, we can finally have a look at the results of our machine learning architecture on various examples.

5.1 Next Generation Reservoir Computing on the Sphere

Here we explain how it is possible to feed spatio-temporal maps on a spherical lattice into the next-generation Reservoir Computing framework and predict them afterwards. The first problem, namely, feeding two-dimensional maps into a one-dimensional NGRC, was already solved and explained. Conceptually, we assign every pixel in the map a unique number and then find all pixel indices that are in the spatial vicinity of a given core pixel to define our local states. Luckily, `Healpy` provides the function `hp.get_all_neighbours()`, which returns, for a given core pixel, the indices of all surrounding pixels. Using this, it is fairly straightforward to create the matrix in which the information about the local states is encoded. Once the matrix with the local states is created, one can proceed as described before with training and prediction.

Alternatively, one can also get the indices of all pixels within a radius of a direction vector using `hp.query_disc()`. This method also allows including more than just the direct neighbours for the local states. However, due to the extended computational cost, this method was not used.
Next, this new method of NGRC with spherical pixelisation scheme will be tested on non-linear synthetic data, namely a wave simulation created on the sphere (Faires and Burden, 1995). Consider the classical second-order wave equation:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u. \tag{5.1}$$

We can discretize equation (5.1) to get to first order

$$u^{t_i+1} = 2\,u^{t_i} - u^{t_i-1} + (c\,\Delta t)^2\,\nabla^2 u^{t_i}. \tag{5.2}$$

As the initial state, we place a Gaussian peak on the empty spherical surface and let the wave evolve according to equation (5.2). After a few time steps, we obtain the following state (see figure 5).

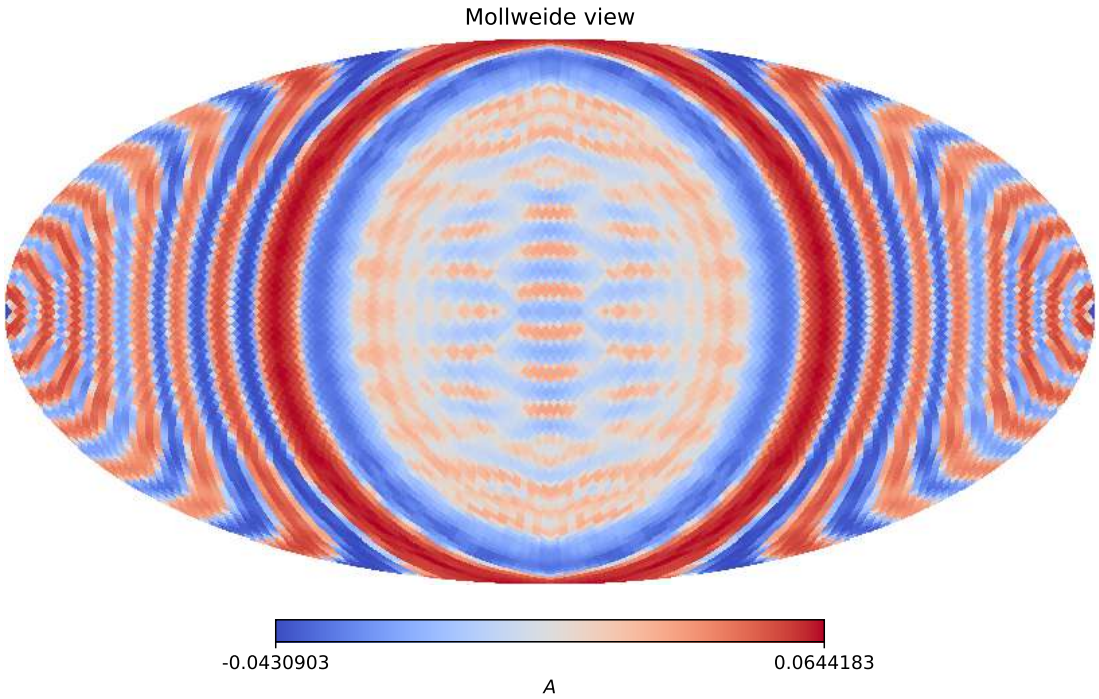


Fig. 5: Mollweide projection of a snapshot of a wave simulation generated with the discretized wave equation. The colour bar is showing the amplitude of the wave A. The simulation has a spatial resolution of N_{side}=32 and a temporal resolution of $N_t = 3000$.

The Simulation has in total $N_t = 3000$ time steps of which 2700 were used for the training and 300 in the prediction. The optimal hyperparameters were determined using `Optuna` with a smaller excerpt of the full sphere surface of size 1000 pixels.

Hyperparameter	Value
α	0.00624
k	15
s	50

We can then start the prediction of the spherical wave and get the following result (see figure 6).

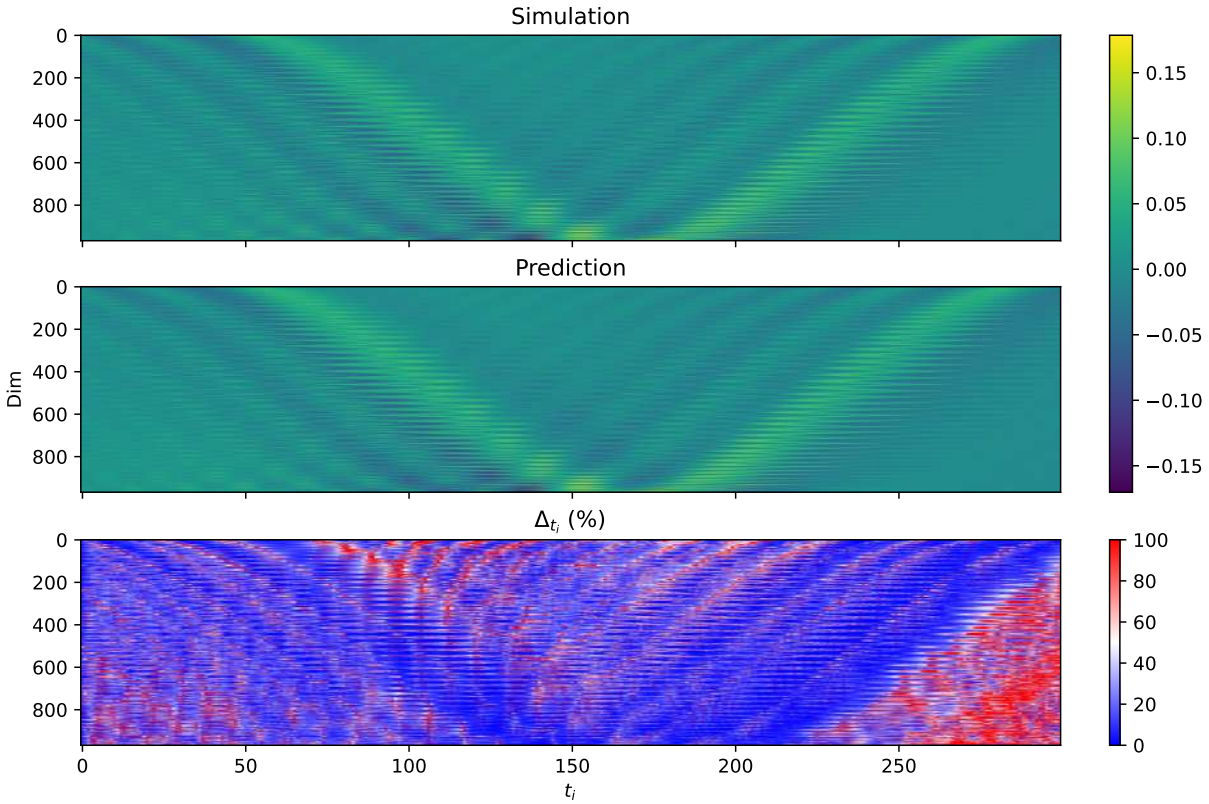


Fig. 6: Representation of the real and predicted wave simulation on the surface of the sphere. The horizontal axis shows the time step after the beginning of the prediction phase while the vertical axis shows the index of the pixels of the 1000 pixel field. At the bottom the absolute of the relative difference between the prediction and the real simulation is shown in percent.

As we can see, the prediction is in very good agreement with the actual simulation. In particular, the wave crest of the original simulation is captured very well, which is the important behaviour to predict. Deviations in predicting the flat areas with values close to zero are naturally more

pronounced, since the bottom plot is normalized by the pixel values of the test simulation. These results show that the implementation of the NGRC on the sphere was successful, and we can continue using it for predicting cosmological simulations on the sphere.

5.2 Adaptive Local States

Here, a new method is presented as an improvement to the classical local states approach called adaptive local states, that allows faster and higher quality prediction of simulation data. To illustrate the problem and motivate this method more vividly, we work for now with simulations of water waves presented as a two-dimensional time series since they exhibit a clear preferred orientation throughout the whole simulation domain.

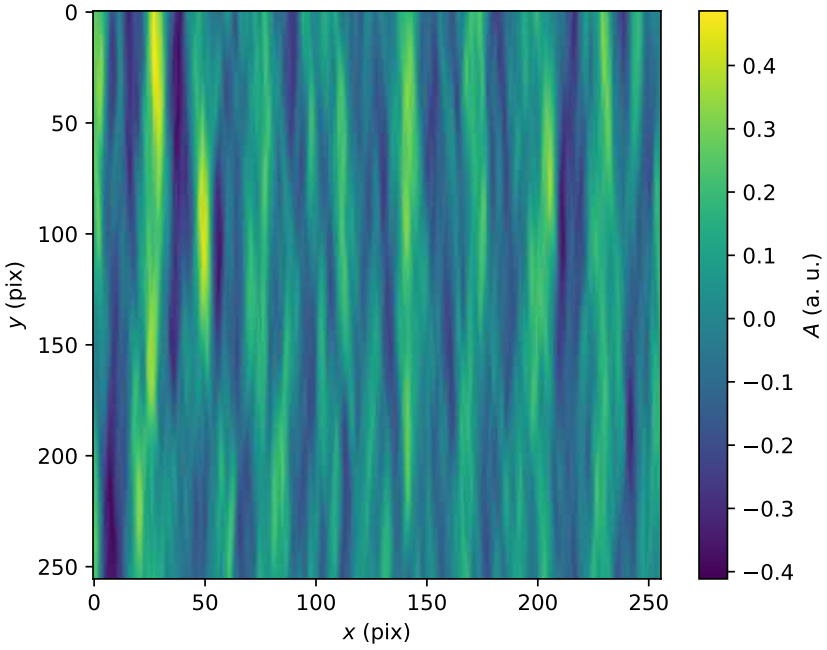


Fig. 7: Snapshot of the wave simulation at the 500th time step. The wave crest and trough of the amplitude A are clearly visible, propagating from left to right. The axes denote the pixel indices (X, Y), yielding a total spatial resolution of 256×256 pixels. In physical units, 256 pixels correspond to a length of 1000 meters in real space. Each time step represents 0.1 seconds, and the simulation has a temporal resolution of $N_t = 1000$.

We can again employ our local states NGRC machine learning architecture to predict the behaviour of the wave simulation.
In the case of this 2D-flat data, for every core pixel, we take the 8 surrounding neighbouring pixels into account in both the training and the prediction phase. In the case of a core pixel lying at the edge of the 2D patch, we only consider the neighbouring pixels that are within the 2D patch. Thus, in the case of a core pixel being at the edge, there are 5 neighbouring pixels, while there are 3 pixels

in the case where the core pixel is located in one of the four corners of the patch.

For the training of the output matrix using the wave simulation data both during hyperparameter optimization and for all the subsequent full predictions, a total of $N_{train} = 850$ time steps were used. The prediction phase begins immediately after the 850th time step. Note, however, that the time step length varies among predictions, which can be directly inferred from the corresponding plots.

A smaller patch of 16 $\times$ 16 pixels was taken from the simulation to optimize the parameters for the included time steps k and s, as well as the regularization parameter α. Since only the direct neighbours were used for the local states of a given core, one could hypothetically also choose a smaller patch. However, at that point, the effect of the ratio between the number of core pixels at the edge of a patch and the number of core pixels inside the patch was not examined, which is why it was decided to use a not too small patch. For optimization, the package `Optuna` was used as described in the earlier chapter.

Both the predicted $\tilde{\vec{x}}_t$ as well as the original time series $\vec{x}_t$ can be represented as a vector at time step t, where each component $x_{i,t}$ represents a pixel i. The cost function then works as follows: First, we calculate the absolute of the relative difference between the original time series and the predicted time series

$$\Delta_{t^*} = \sum_i \left| \frac{\tilde{\vec{x}}_{i,t^*} - \vec{x}_{i,t^*}}{\vec{x}_{i,t^*}} \right|. \tag{5.3}$$

Then, the average over several time steps is calculated to account for the possibility that the deviation may be exceptionally high at a single time step, while it is fairly small at the surrounding time steps. Empirically averaging over 10 time steps appears to be good compromise between robustness and precision, as averaging over all time steps would not give any new information.

$$avg_{t^*} = \frac{\sum_{t=t^*-9}^{t*} \Delta_{t^*}}{10}. \tag{5.4}$$

The first t^* where avg_{t^*} over 50% is the returned value of the reward function, meaning we try to maximize this value for various test hyperparameters.

With this method and after testing 100 sets of hyperparameters in the intervals

$$\begin{aligned} \alpha &\in [10^{-4}; 1] \\ k &\in [2; 6] \\ s &\in [1; 9] \end{aligned}$$

we finally get the best hyperparameters

Hyperparameter	Value
α	0.00367
k	4
s	6

Tab. 1: Parameters obtained through optimization with the python package `Optuna`. In the optimization a 16 times 16 pixels patch was used.

After that the best parameters were used over for the prediction of the full 256×256 pixel patch. The results can be seen in figure 8.

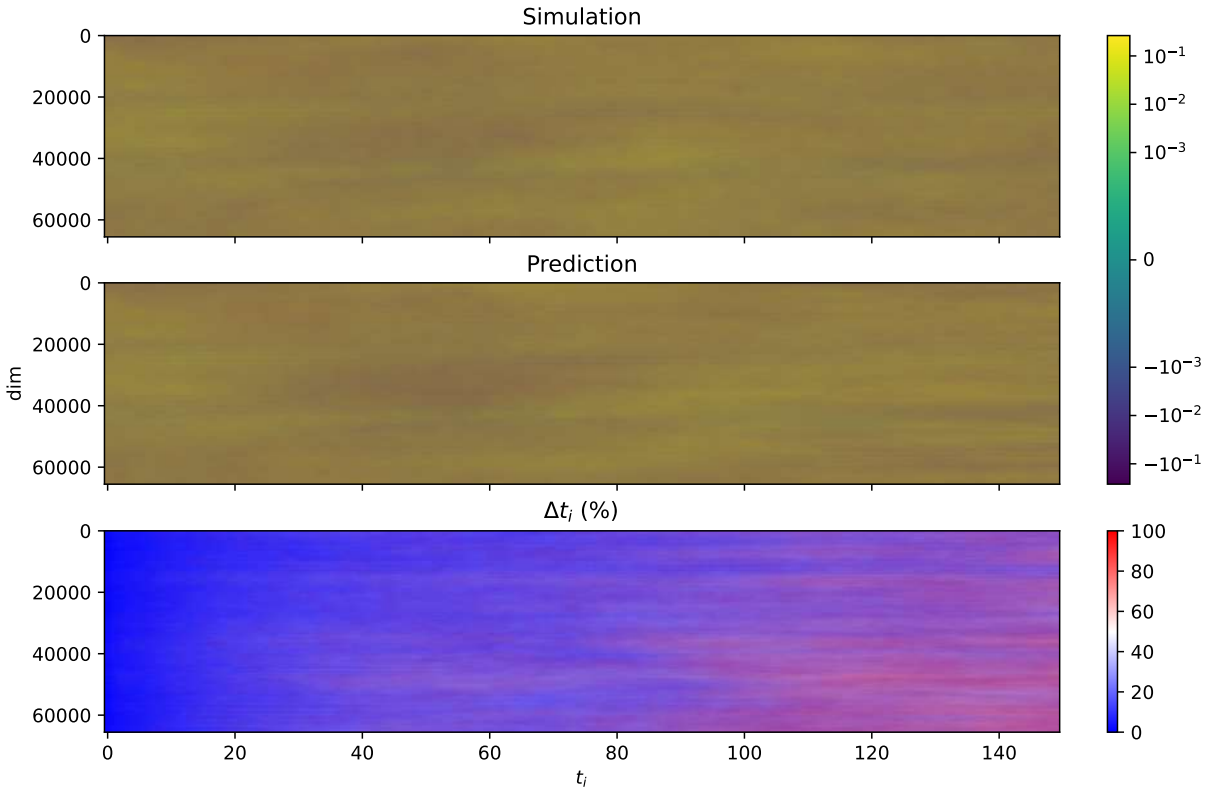


Fig. 8: Representation of the original and predicted wave simulation. The x-axis shows the time step after the beginning of the prediction phase, while the y-axis shows the index of the pixels of the 256 × 256 pixel field. At the bottom, the absolute value of the relative difference between the prediction and the original simulation is shown in percent.

If we calculate the average deviation between the original and predicted simulation over 10 time steps we get a value of 45%.

We can also observe that the deviation does not grow uniformly across all pixels. Instead, the error first emerges in a single pixel and as a result propagates to the surrounding pixels. This behavior is expected, since the local states are defined through the spatial neighborhood, consequently, inaccuracies in neighboring pixels affect the prediction of the central pixel.

One issue with simply considering at the relative difference is that we still do not have a performance baseline to compare it to. The easiest way to predict the evolution of the waves without considering any prior knowledge would be to simply assume that the waves remain constant. Therefore, we can now compare how much better our elaborate NGRC prediction performs compared to this naive prediction.

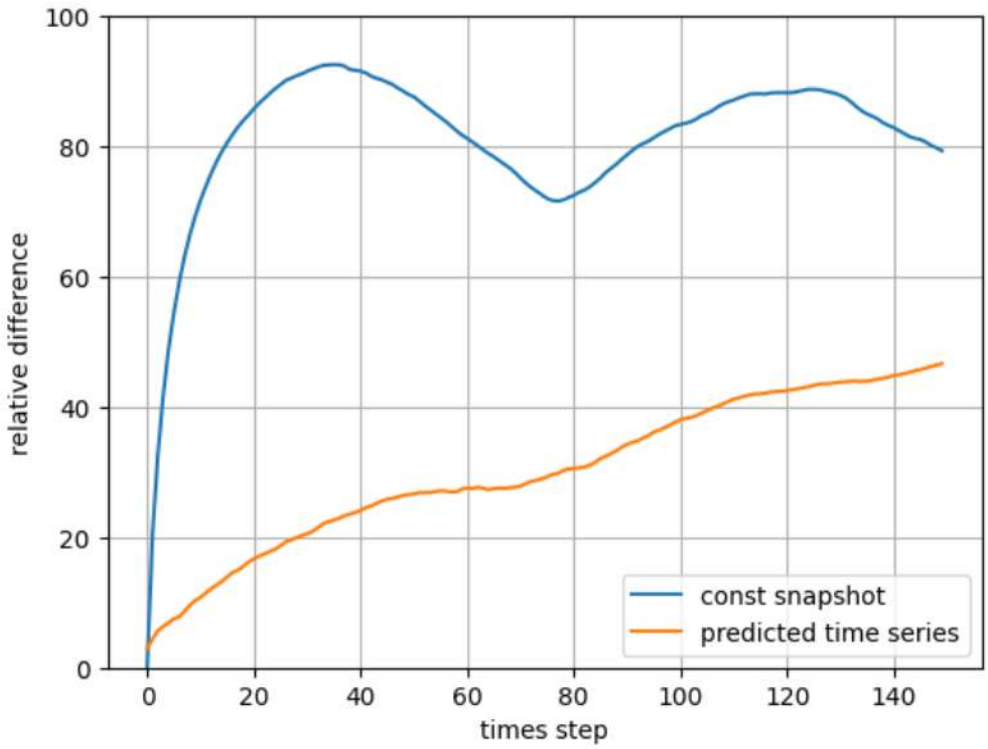


Fig. 9: Relative difference between the original wave simulation and one of two prediction methods. For each time step, the absolute value of the difference across all pixels was calculated. The first method simply assumes that the waves freeze at the first time step, so a constant snapshot is compared to a dynamic wave simulation. The second method is the prediction made by the NGRC local states method.

In the calculations, the difference was clipped at 100%, causing it to gradually saturate at this value; without this clipping, the difference would be expected to exceed 100%. As noted earlier, the average difference between the original simulation and the NGRC prediction barely reaches 50% by the 150th time step. In contrast, if the waves are assumed to remain constant in time for the prediction, the difference does not simply keep increasing; instead, it begins to oscillate once it approaches roughly 80%, likely reflecting the waves inherent periodicity.

We can verify this possible explanation by calculating the average time a wave needs to fully pass a pixel. The wave simulation consists of several waves with different periods, but the peak of the Fourier spectrum T_p, as well as the real time corresponding to a single time step Δt, are given. The number of time steps until a wave fully passes a pixel is then roughly

$$N_t = T_p \times \Delta t = 80.04 \ , \tag{5.5}$$

which matches the observed periodicity in the graph.

Another interesting observation is that the relative difference at the first time step does not start at 0% but instead at around 5%, and then increases much more slowly than 5% per time step.

Now one can also examine what influence the selection of local states has on the prediction quality of the core pixel. As previously explained in subsection 4.3, choosing the correct local states can greatly speed up the training as well as the prediction process. The reason why this is demonstrated using wave simulations is that wave dynamics show a clear preferred orientation, namely the direction in which the flat wave front propagates (from left to right). The existence of a preferred direction motivates investigating whether the orientation of the neighbouring local-state pixels has an influence on the quality of the prediction.

The following cases are studied: taking the neighbouring horizontal (left and right) pixels as local states; taking the neighbouring vertical (up and down) pixels as local states; and taking both vertical and horizontal, but not the diagonally neighbouring, pixels as local states. For the solely horizontal or vertical local states, not only the direct neighbours but also the second-nearest neighbours were chosen.

The matrices for the local states can be found in figure 10. Using these matrices, we can then finally start the predictions of the wave simulations. Since we are only interested in how well the different local states perform relative to eachother we can also simply predict an 8×8 pixel excerpt of the whole simulation to speed up the process.

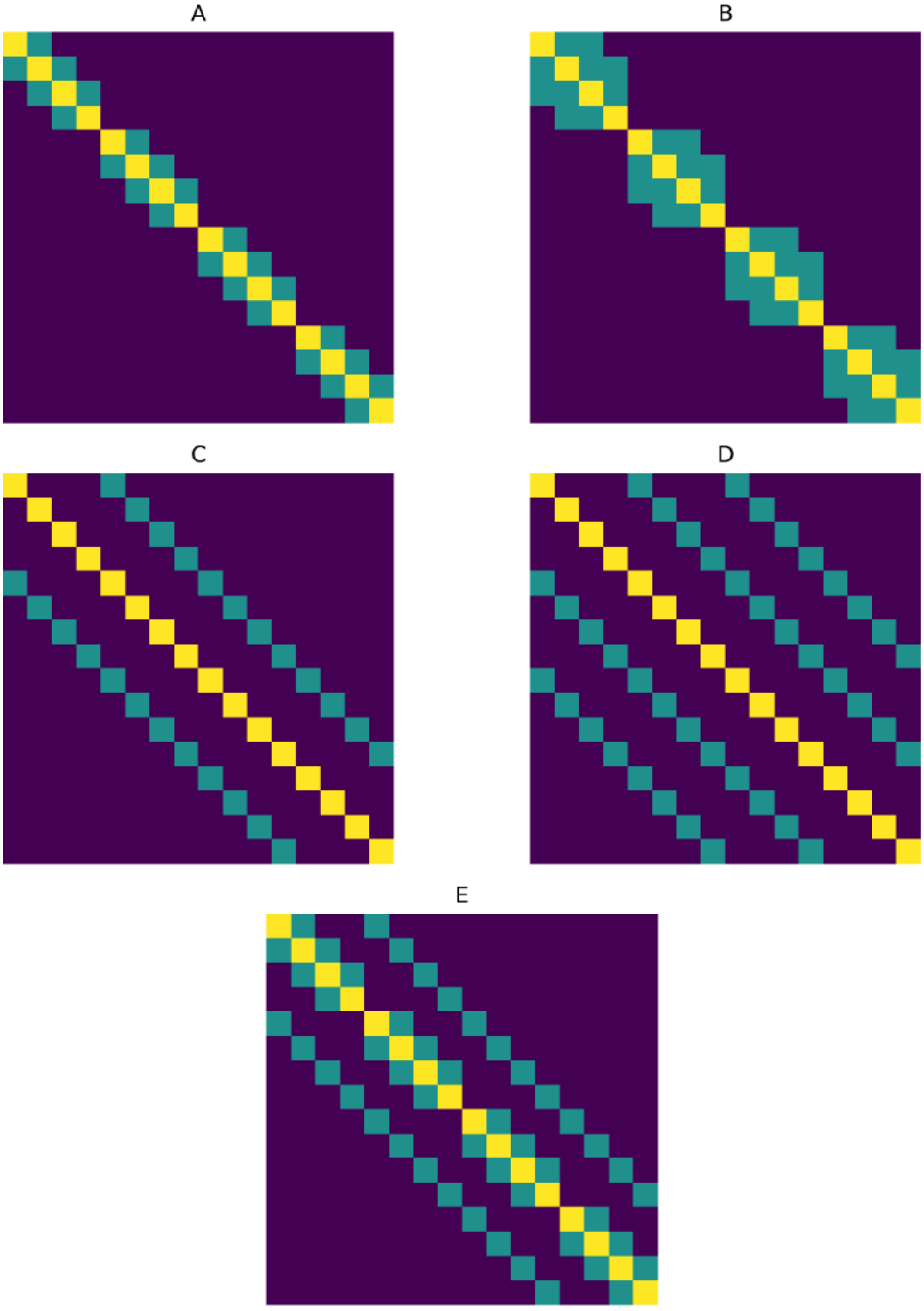
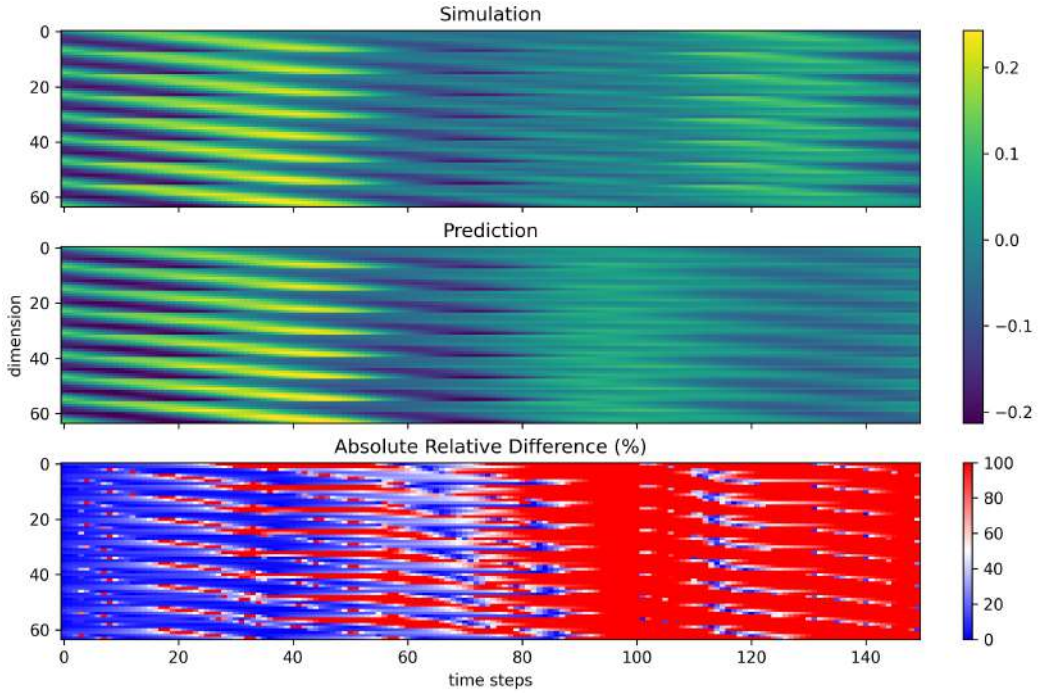
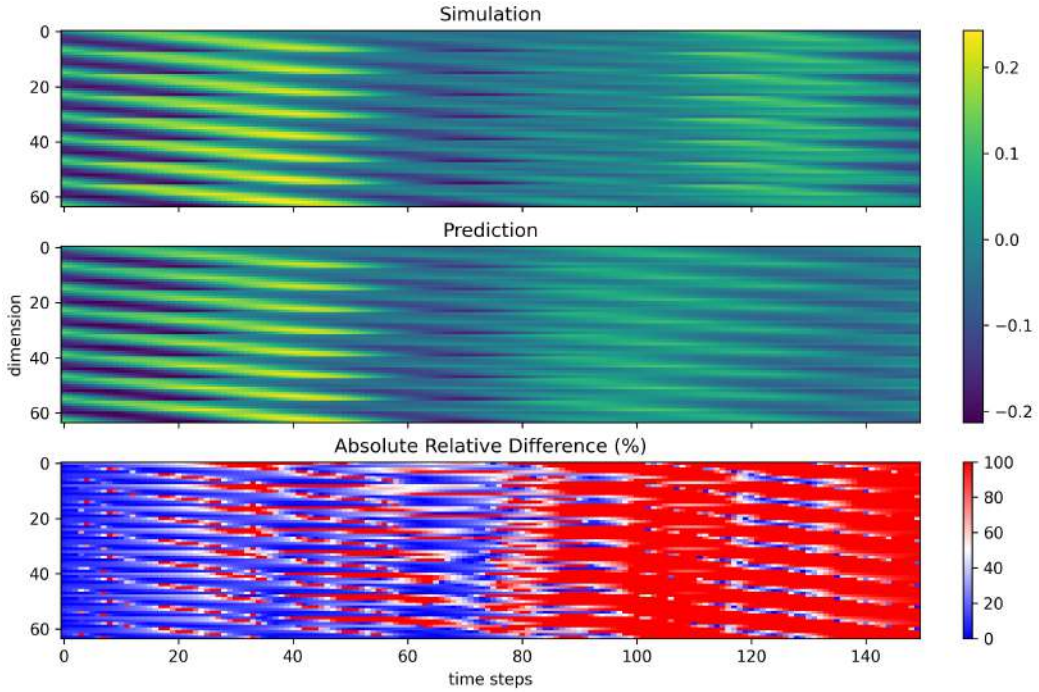


Fig. 10: Local states used for predicting the 2D wave simulation on a 4×4 pixel field. The axes show the pixel indices; yellow marks the core pixel, and blue marks the pixels used as local states. Plot A uses only the left and right neighbours, Plot B includes the second-nearest horizontal neighbours, Plots C and D show the same for vertical neighbours, and Plot E uses all surrounding neighbours except the diagonals, effectively combining horizontal and vertical neighbours.

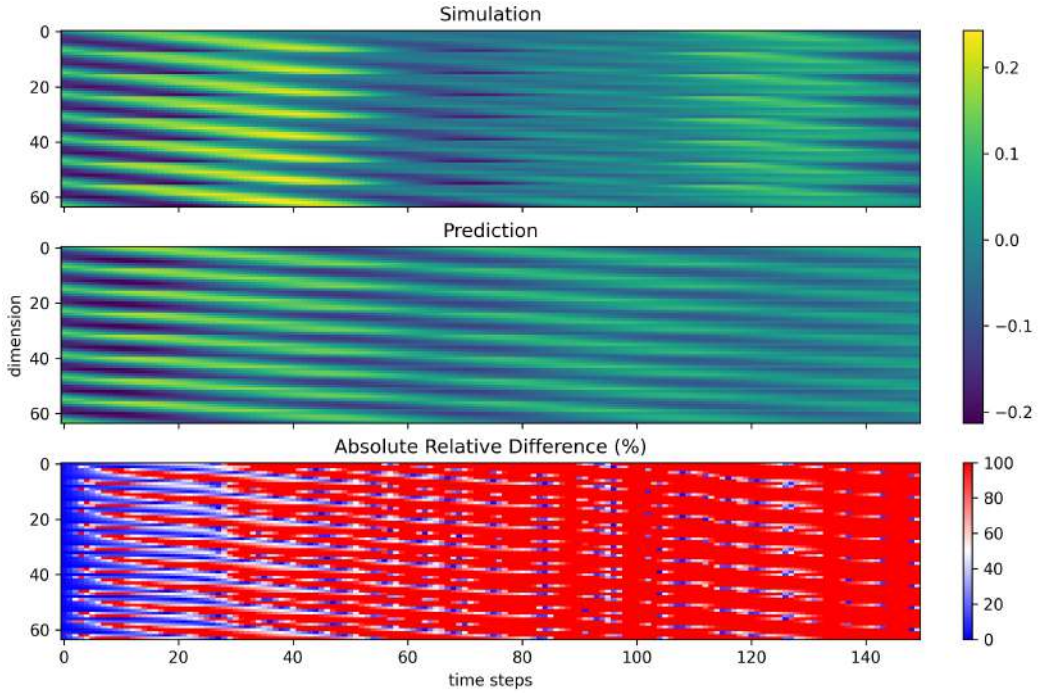


(a) Horizontal neighbours

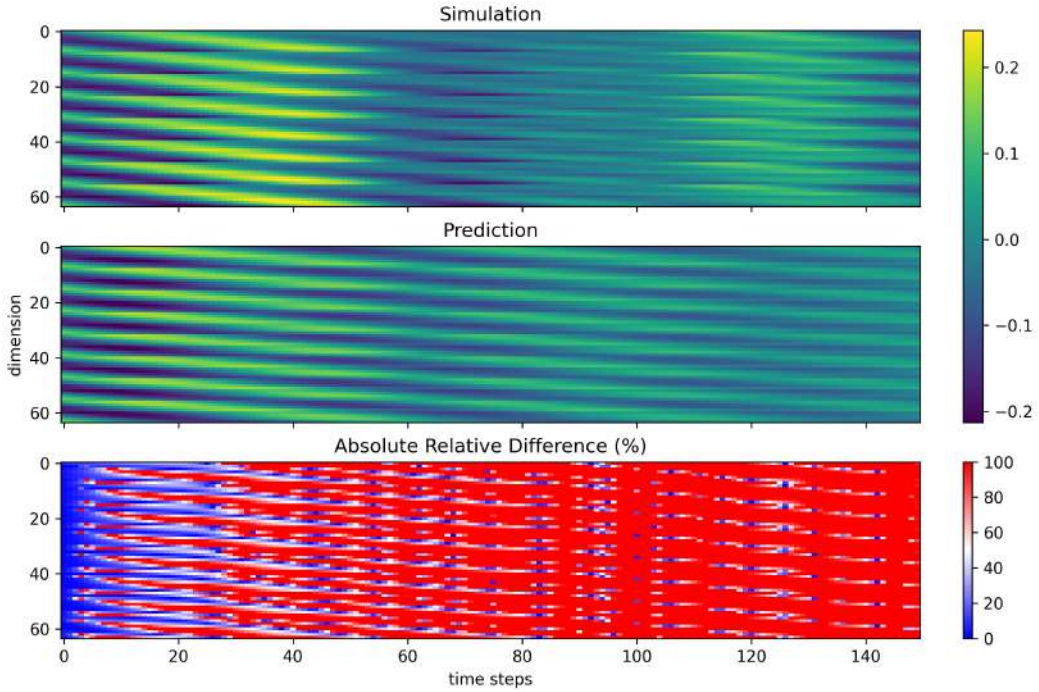


(b) Horizontal neighbours with 2 to the left and right

Fig. 11: Comparison between the predicted and original behaviour of a simulated wave. (a) is showing the case where the pixels to left and right where taken as local states while (b) is showing the case where the two pixels to the left and the two pixels two the right of the core pixel were taken for the local states.

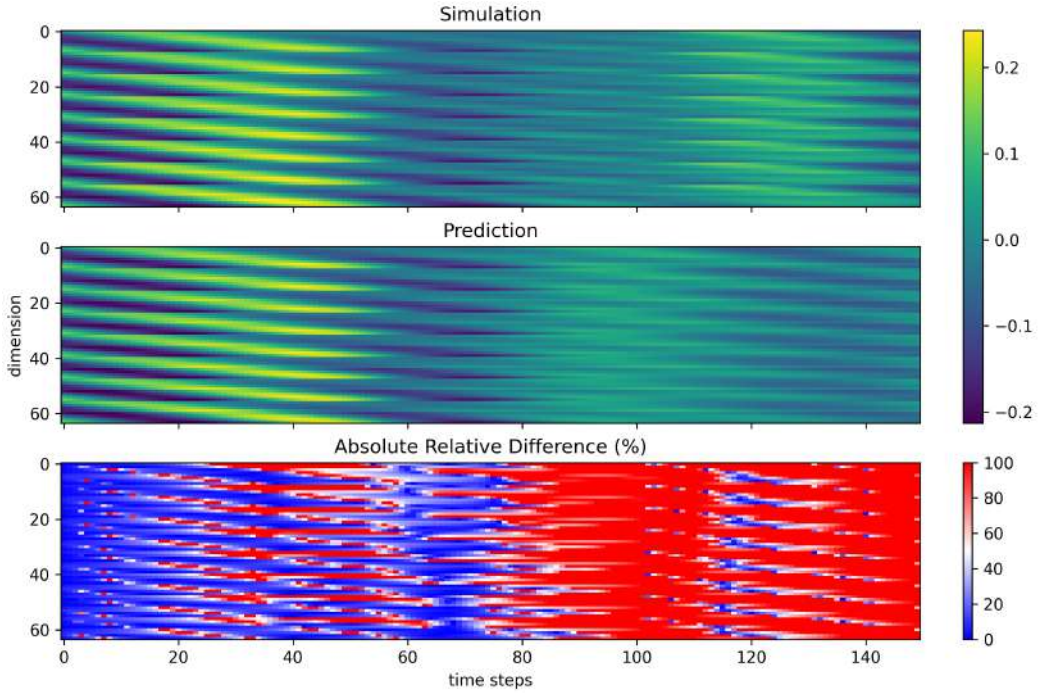


(a) Vertical neighbours

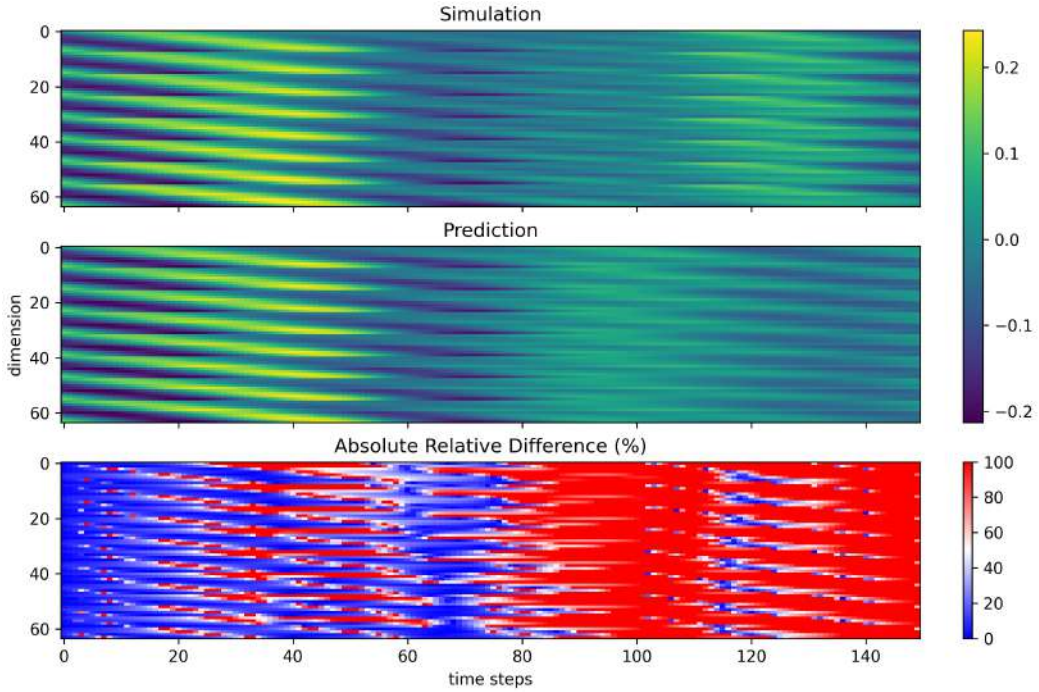


(b) Vertical neighbours with 2 above and below

Fig. 12: Comparison between the predicted and original behaviour of a simulated wave. (a) is showing the case where the pixels above and below were taken as local states while (b) is showing the case where the two pixels above and the two pixels below the core pixel were taken for the local states.



(a) All surrounding neighbours



(b) Horizontal and vertical neighbours

Fig. 13: Comparison between the predicted and original behaviour of a simulated wave. (a) is showing the case where all the 8 surrounding pixels were taken as local states while (b) is showing the case where the vertical and horizontal neighbours of the core pixel were taken for the local states.

We can observe in the plots 11, 12 and 13 showing the absolute of the relative difference how the difference for certain time steps increases even above 50%, only to decrease in the following time steps, showing once again the importance of including several time steps when checking whether the average of the relative difference is above 50%.

If we now take into account at which time step the average over the next time steps exceeds 50%, we obtain the following values (see table 2).

Setup	Reward Function Value
Normal	82
Cross	81
Vertical	26
Vertical 2	28
Horizontal	59
Horizontal 2	83

Tab. 2: Measured values for different local states

These values generally also reflect the visual impression one gets regarding the quality of the prediction from the plots. We can see that the worst-performing local states setup is the use of vertical neighbours. This congruent with our expectations, insofar as, since the wave is moving from left to right, there is very little information encoded in the pixels above and below the core pixel to predict its value at the next time step, and the pixels above and below have little influence on how the core pixel will behave in the future.
Nonetheless, taking more than just the pixel immediately above and below, by also including the second pixels above and below, increases the "forecast horizon," simply because more information is fed into the training.
Next in the performance hierarchy is the horizontal setup, which is partly due to the fact that it only considers two pixels, which is relatively few compared to the other setups. However, if we take not only the very next horizontal neighbour but also the second horizontal neighbour for the local state (Horizontal 2), the forecast horizon increases much more than the increase observed from the vertical to the vertical2 setup. This is likely due to the direction of wave propagation along the horizontal axis, which makes every additional pixel in the horizontal direction provide additional information, whereas additional pixels along the vertical axis do not add as much.
If we take both the vertical and horizontal neighbours for the local states (cross), we again observe a rather large increase in the forecast horizon. This is again due to the fact that the horizontal and vertical axes are not correlated, so they encode “different information,” so to speak.
To conclude, the largest forecast horizons are observed with the cross setup, the normal setup (which includes all 8 surrounding neighbours of the core pixel), and the horizontal2 setup.
Now, looking solely at the reward function values, one might get the impression that, since the normal setup performs more or less as well as the cross or horizontal2 setup, the choice of local states has minimal influence on the prediction quality. However, the number of pixels taken into account also affects the time required for training and prediction. While a total of 9 pixels are included in the construction of the feature vector for the normal setup, we only take 5 for the cross or horizontal2 setups. The corresponding feature vectors $\mathbb{O}_{\text{lin}}$ and $\mathbb{O}_{\text{nonlin}}$ thus have the following lengths:

$$l_{\text{cross,horizontal2}} = l_{\text{lin}} + l_{\text{nonlin}} \ , \tag{5.6}$$

where the length of the non linear feature vector l_{nonlin} with the number of local states (and with the core) N_{LS} and a monomial of order 2 and the hyperparameter value k is calculated as follows

$$l_{\text{nonlin}} = \binom{k \times N_{LS}}{2} = \frac{(k \times N_{LS})!}{(k \times N_{LS} - 2)!(2)!}. \tag{5.7}$$

The calculation for the linear feature vector l_{lin} is simply

$$l_{\text{lin}} = k \times N_{LS}, \tag{5.8}$$

with $k = 4$ according to table 1 and $N_{LS} = 5$ in the cross and the horizontal2 or $N_{LS} = 9$ in the normal the corresponding total feature vectors have the length

$$\begin{aligned} l_{\text{cross,horizontal2}} &= 210 \\ l_{\text{normal}} &= 666\ . \end{aligned}$$

This means that our feature vector is a factor of $l_{\text{normal}}/\text{l}_{\text{cross,horizontal2}} = 3.17$ longer compared to the cross or horizontal2 setup. Our forecast horizon with the cross or horizontal2 setup is just as good as in the normal case with all surrounding pixels as local states, while requiring much fewer computational resources!
In this wave simulation, the dominant orientation remains constant over time. However, there are many cases where the directions of the waves change over time. If we were to set our local states to be constant in time, the local states could align with the wave orientation for some time steps but not for others, leading to suboptimal predictions. Ideally, we would be able to automatically determine the "direction of the flow" in our simulations and then choose our local states dynamically according to that direction.
The requirements become even more complicated when we expect the simulation data not only to change direction over time but also to show different directions in different regions of the simulation. In that case, the local states are not only changing over time but also need to differ for different core pixels at a given time step.
While previously the direction of flow was intuitively clear from looking at the wave simulation, we now need to operationalize finding the flow direction in order to automate the process for each time step. One method that works particularly well with wave simulations is to take the direction with the highest gradient at the position of the core pixel as the flow direction of the wave. To obtain "adaptive" local states for a core pixel, the indices of all eight surrounding pixels are determined. Then the two pixels with the highest difference/gradient relative to the core pixel are chosen as the local states. In this way, the flow direction can be individually determined for every core pixel.
To dynamically change the local states over time, one simply predicts one time step ahead before recalculating the new local states. The challenge here is that, for predicting a new time step, all the $\{1s, 2s, ..., k \cdot s\}$ previous time steps are used to construct the feature vector. This means that up to $k \cdot s$ time steps before the predicted step, the local states cannot be changed, and thus a flow direction averaged over this period must be used. As long as the simulation changes slowly compared to this $k \cdot s$ period, this is not a problem. For fast-changing simulations, this limitation must be kept

in mind.
We can now check whether this method of using the gradient to determine the flow direction works well. For example, the local states matrix would look as follows when using the local states method (see figure 14).

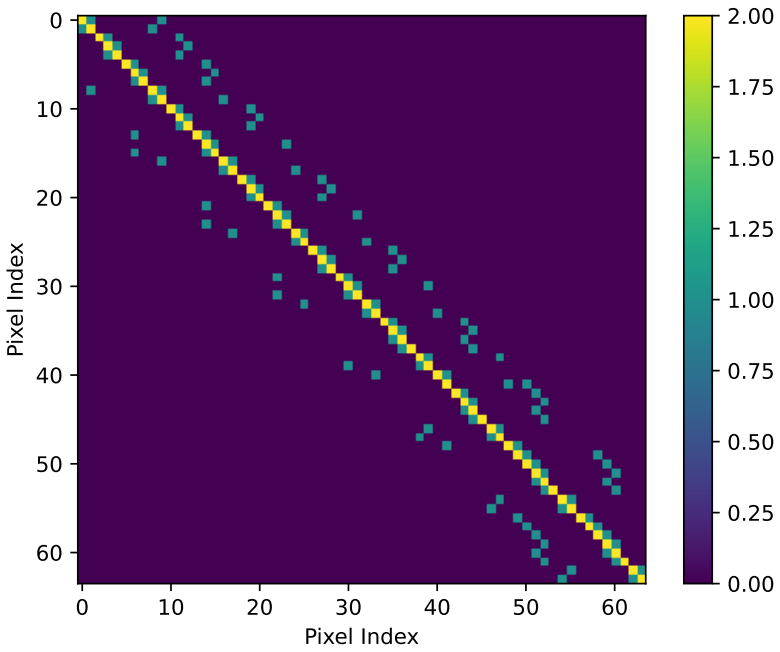


Fig. 14: Neighbourhood matrix being generated by using the new adaptive local states method which incorporates considering the two pixel with the highest gradient towards the core pixel as local states pixel.

Interestingly, we get a different impression than what we would obtain if we simply assumed that the entire wave is moving horizontally to the right. Comparing the adaptive local states matrix with the one generated by taking only the horizontal neighbours (see 10), we see that only one of the two surrounding pixels with the highest gradient lies horizontally next to the core pixel. A clearer picture emerges if we choose the two pixels with the highest gradient out of all the neighbours forming a cross around the core pixel (see 10). We then obtain the following neighbourhood matrix in figure 15.

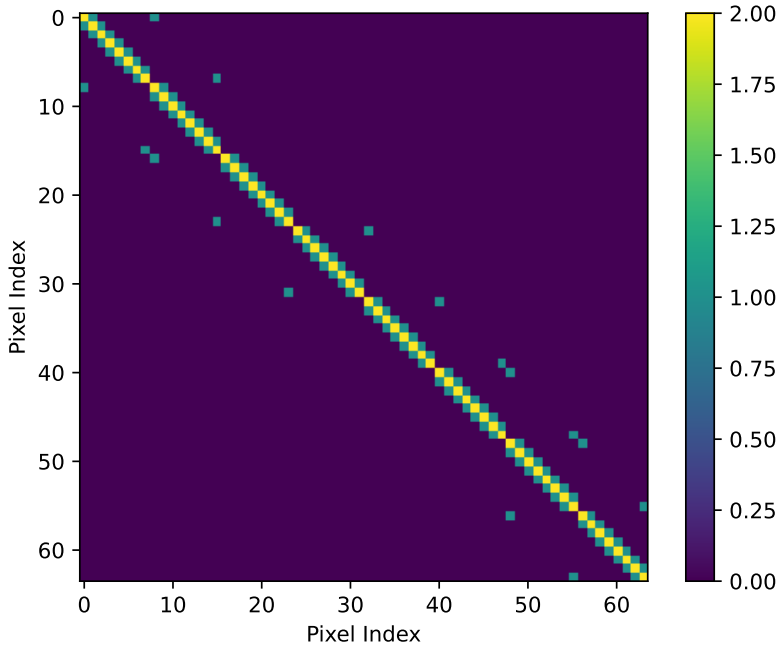


Fig. 15: Neighbourhood matrix being generated by using a modified version of the new adaptive local states method which incorporates considering the two pixel with the highest gradient towards the core pixel as local states pixel taken from the set of neighbours laying crosswise next to the core pixel.

As we can see, the local states pixels now lie to the left and right of the core pixel, meaning that in the classical adaptive local states matrix, the second neighbour lies diagonally to the core pixel. This is reasonable because, although the wave as a whole may move to the right, there can be local regions where the flow direction shifts slightly upward or downward. What would be problematic is if the pixels with the highest gradient were aligned directly above and below the core pixel along the vertical axis, but this was not observed.
With all the preliminaries discussed, we can now examine the results generated using the adaptive local states method.

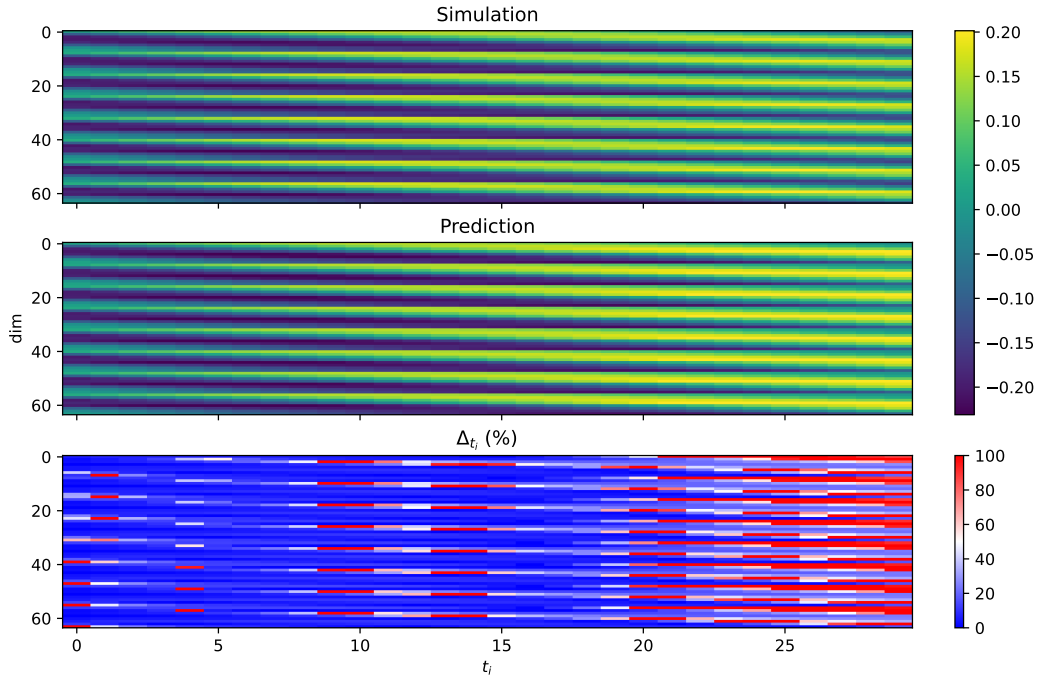


Fig. 16: Prediction made with the help of the analytical local states method. In the $k \cdots$ time steps preceeding the current time step the two neighbouring pixels with the highest gradient to the core pixel will be chosen as local states of that time step.

Comparing the the results of the adaptive local states method with that of the classical constant local methods we see that the forecasting horizon of the adaptive local states method is much higher than that of the constant one.
A limitation of the adaptive local states method is that the weights in the output matrix W_{out} must be retrained at each predicted time step. This retraining uses a segment of the full training time series spanning from the current time step back to $k \cdot s$ steps prior. Although the reduced length of the training segment accelerates individual training iterations, this procedure must be repeated for every time step, resulting in a higher computational cost for generating complete predictions compared to the conventional constant local states approach.
Motivated by this problem, one can instead train the output matrix with the full training time series for all surrounding neighbours (i.e., using the classical constant local states). Then, during prediction, one considers only two pixels for the local states, exactly as in the adaptive local states approach. The information from the trained weights of the other six neighbours is, for the current time step, effectively discarded. However, it can be used again in the prediction of the following time step if the corresponding pixel indeed had a high gradient relative to the core pixel in that step. This semi-adaptive approach mitigates the significant speed loss of the fully adaptive method while still ensuring better prediction quality than the constant local states approach.

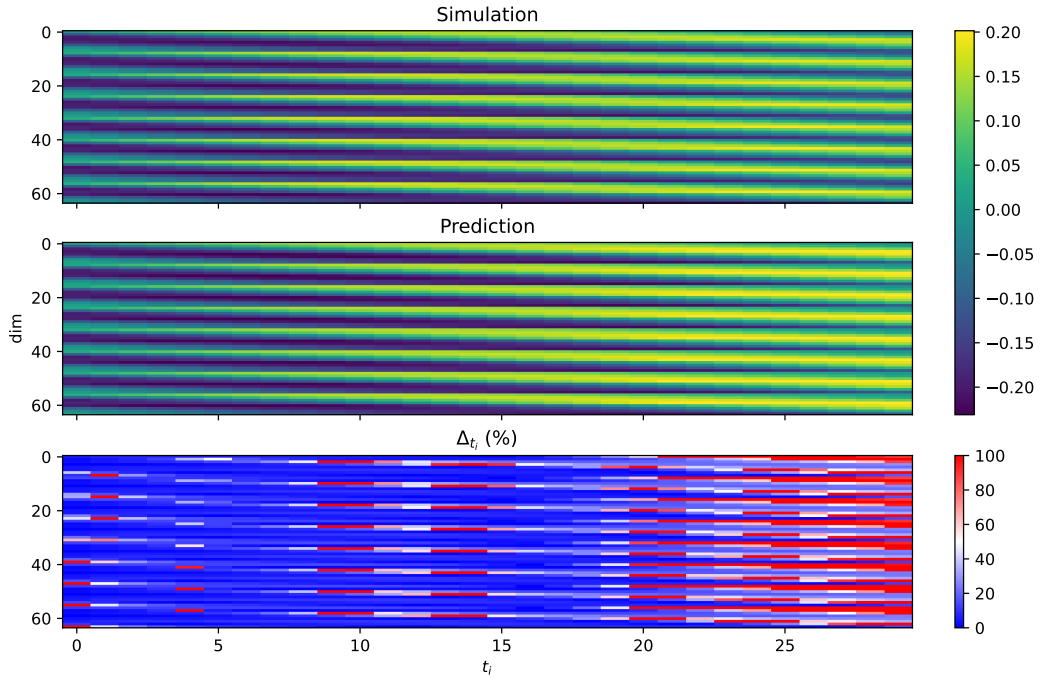


Fig. 17: Prediction made with the help of semi analytical local states method. While the training is running with constant local states the prediction choses only the two pixels with the highest gradient similar to the adaptive local states method.

Using the reward function defined earlier in equation (5.4) to quantify the difference we get the following "forecast horizon":

Setup	Reward Function Value
Adaptive	32
Semi Adaptive	31
Constant	29

Tab. 3: Forecast horizon determined via reward function (5.4) for different local states methods.

As expected, the forecast horizon of the semi-adaptive method lies between that of the constant local states and the adaptive local states approaches.
Since we are currently examining data consisting of a superposition of waves, there is the possibility to use a different metric more suitable for oscillatory spatio-temporal data, namely the Surface Similarity Parameter (SSP). The SSP is defined as follows: (Perlin and Bustamante, 2016)

$$\mathrm{SSP}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\sqrt{\int \left|F_{\mathbf{x}_1}(\mathbf{k}) - F_{\mathbf{x}_2}(\mathbf{k})\right|^2 \mathrm{d}\mathbf{k}}}{\sqrt{\int \left|F_{\mathbf{x}_1}(\mathbf{k})\right|^2 \mathrm{d}\mathbf{k}} + \sqrt{\int \left|F_{\mathbf{x}_2}(\mathbf{k})\right|^2 \mathrm{d}\mathbf{k}}}, \tag{5.9}$$

where $F_{\mathbf{x}}(\mathbf{k})$ denotes the Fourier transformation of $\mathbf{x}(t)$. The advantage of this metric over other

methods, such as calculating the RMSE or using the previously introduced reward function, is that the SSP is more uniformly sensitive to deviations in the phase, wavelength, and amplitude of a wave composition (Wedler et al., 2022). In contrast, the RMSE or simply calculating the relative difference in percent is oversensitive to deviations in the amplitude, as these deviations are not bounded. It is therefore of interest to apply the SSP to the results produced so far for the wave data and to check whether this new metric qualitatively changes the observations made.
We calculate the SSP for each time step $\mathrm{SSP}(t_i)$ for every local states setup to get qualitatively the following plot (see figure 18).

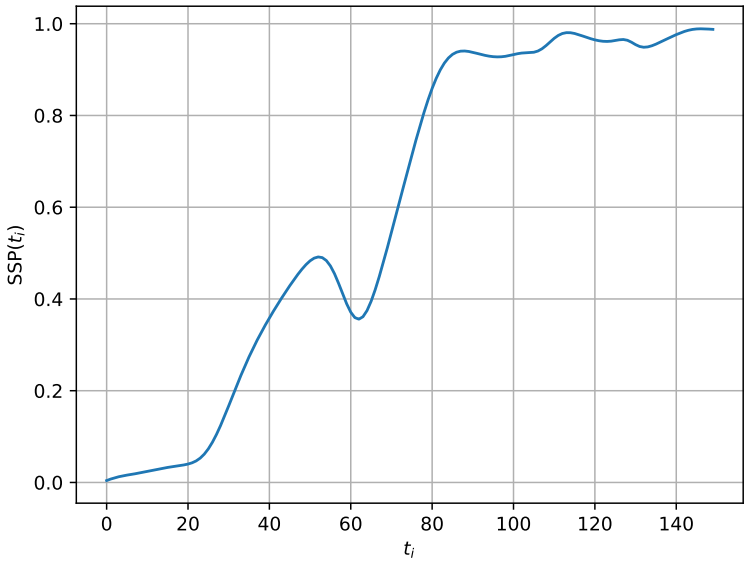


Fig. 18: SSP for each time step $\mathrm{SSP}(t_i)$ for the classical local states method which includes all surrounding pixels of the core pixel for neighbourhood matrix.

We can observe how the SSP exhibits now monotone behavior which stems again from the oscillatory nature of the simulation data.
Then we calculate the average SSP over all time steps:

$$\overline{\mathrm{SSP}} = \frac{1}{N_t} \sum_{t_i=1}^{N_t} \mathrm{SSP}(t_i). \tag{5.10}$$

The average SSP is calculated for every local states setup and presented in table 4.

Setup	$\overline{\text{SSP}}$
Normal	0.0251
Cross	0.0338
Vertical	0.0844
Vertical 2	0.0798
Horizontal	0.0535
Horizontal 2	0.0231

Tab. 4: Measured $\overline{\text{SSP}}$ values for different local states.

We see that the overall picture has not changed and the ranking of the different local states set ups regarding their $\overline{\text{SSP}}$ performance stays the same though the relative difference between the best and the worst performing local states set up is now slightly larger.

5.3 Magneticum Pathfinder

We now want to apply all the aforementioned methods to astrophysical simulations, starting with the `Magneticum Pathfinder` Simulation Suite (Dolag et al. (2025)). The Magneticum simulations are designed to track the formation of cosmological structures across a wide range of scales. They consist of a series of hydrodynamical simulations of varying cosmological volumes, each sampled with a very large number of particles, providing high spatial resolution. Numerous physical processes are included.
For our analysis, we used the mass maps of Box2 and Box2b. The mass maps were created by considering the mass distribution in full-sky cones within a given redshift interval. The mass within each cone was then projected along the radial axis, resulting in a two-dimensional distribution in the Mollweide projection.

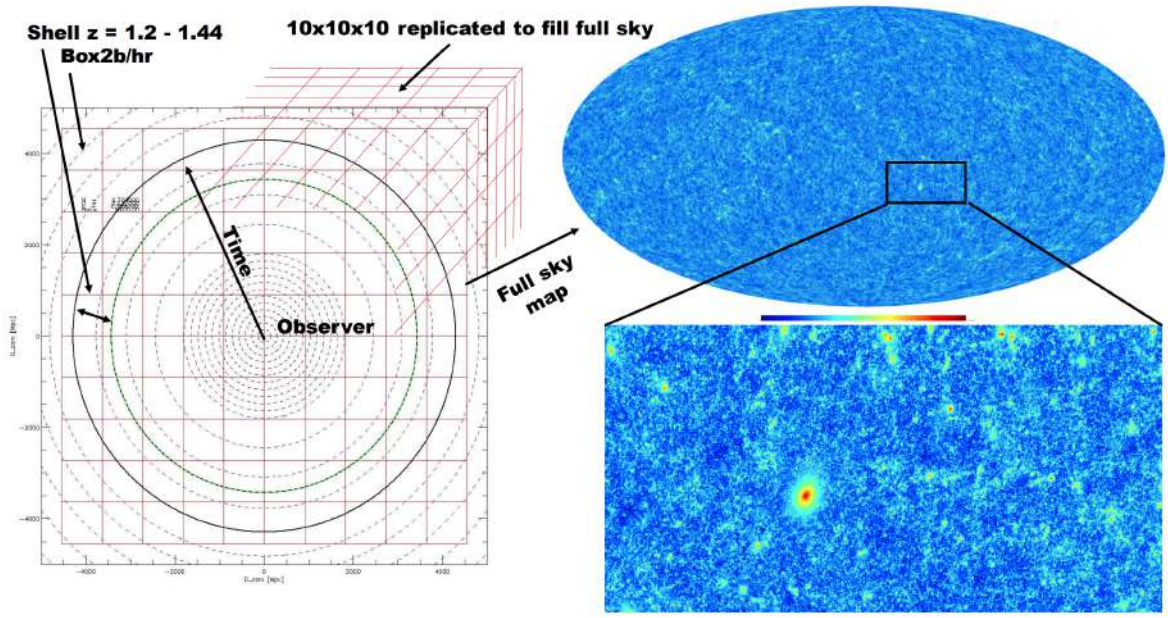


Fig. 19: Illustration showing the geometry of one slice for the full tSZ sky light-cone from Box2b/hr on the left. The right panel shows the full sky map of the whole light-cone with a zoom onto one galaxy cluster.

The simulation is using about $2 \cdot 2000^3$ particles with particle masses for dark mater and gas of $m_{DM} = 6.9 \cdot 10^8 \mathrm{M}_\odot/h$ and $m_{gas} = 1.4 \cdot 10^8 \mathrm{M}_\odot/h$. For the simulation the WMAP7 cosmology (Komatsu et al., 2011) was used with cosmological parameters in table 5.

Cosmological Parameter	Value
Ω_m	0.272
Ω_Λ	0.728
H_0	70.4
n	0.963
σ_8	0.809

Tab. 5: WMAP7 cosmological parameters used for simulation

In total there are 14 full sky light cones each with a given average redshift z_i. Assuming a WMAP7 cosmology equation 3.31 for the age of the universe can be specified to

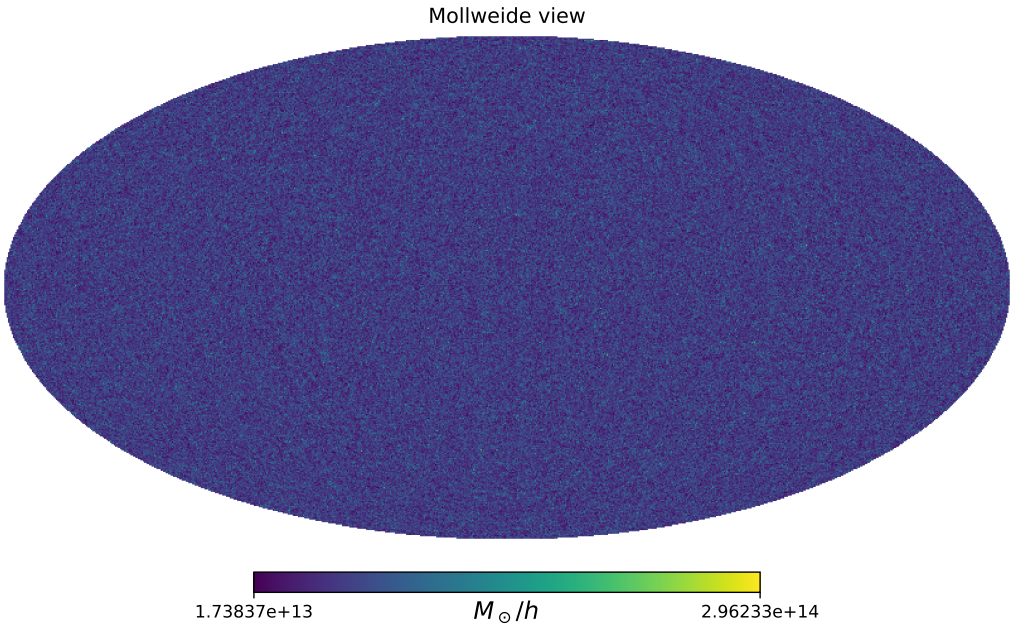
$$t(z) = \int_z^\infty \frac{1}{(1+z')\ H(z')}\,\mathrm{d}z' = \int_z^\infty \frac{1}{(1+z')\ H_0\ \sqrt{\Omega_m(1+z')^3+\Omega_\Lambda}}\,\mathrm{d}z'. \tag{5.11}$$

This integral has no analytical solution, however one can calculate a value numerically. Using the python library `Astropy` one can calculate with the list of the redshift the corresponding age of the universe at that redshift. The results can be found in table 6.

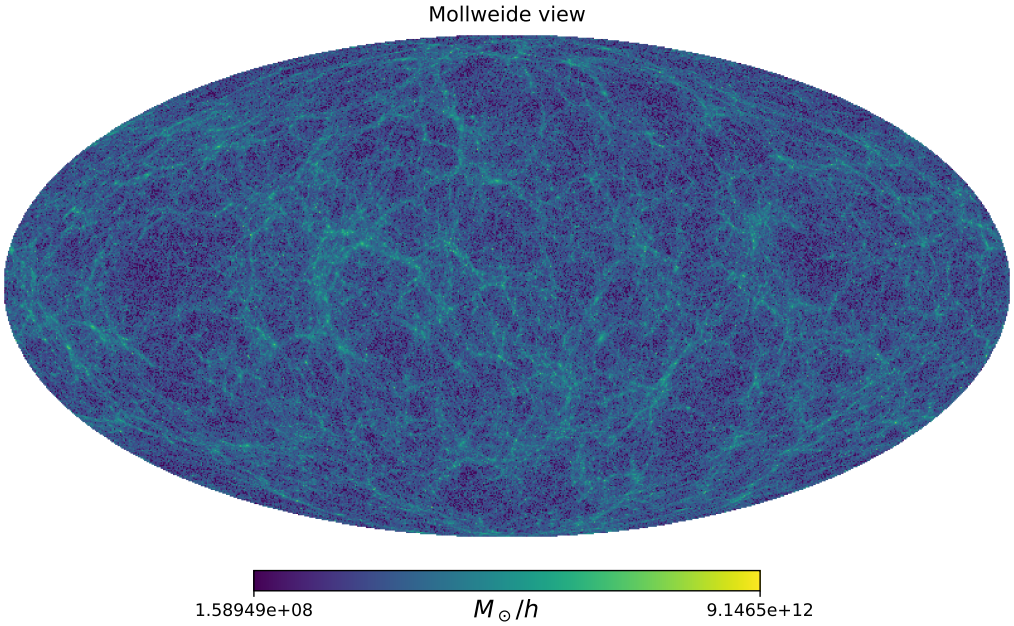
Cone	Redshift z	Age of Universe (Gyr)
1	0.03367115	13.29683801
2	0.06740759	12.86013230
3	0.10224509	12.42995381
4	0.17536824	11.59050129
5	0.21372931	11.18187268
6	0.25334240	10.78105554
7	0.29424834	10.38834261
8	0.33648937	10.00400833
9	0.38010904	9.62830739
10	0.42515233	9.26147130
11	0.47166573	8.90370584
12	0.67340373	7.56648266
13	0.90279626	6.38328163
14	1.18107610	5.29361009

Tab. 6: Redshifts and corresponding ages of the Universe for each cone.

We can also view the first and last snapshot of the simulation data to get a first impression how the time series evolves with time.



(a) First time step



(b) Last time step

Fig. 20: First and last time step of the projected mass distribution of the `Magneticum Pathfinder` Simulation Suite in Mollweide projection. The original data had $N_t = 14$ and after interpolation $N_t = 161$.

Which may appear surprising at first glance are the lower masses at later times. If we take a closer look we get following behaviour (see figure 21).

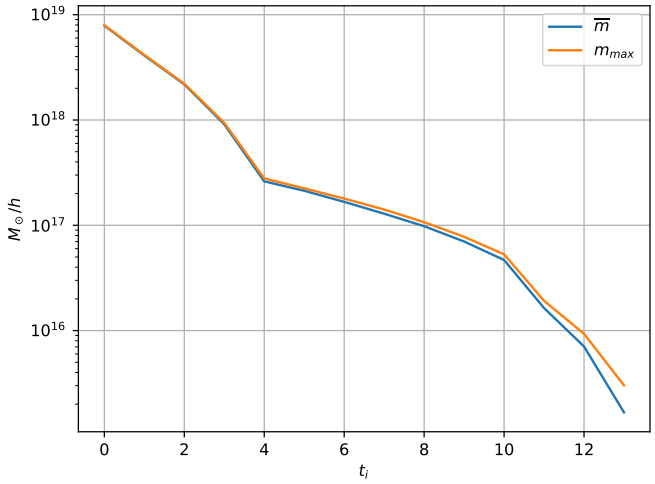


Fig. 21: Mean $\overline{m}$ and maximum m_{max} mass for each time step of the original data from the spherical cosmological simulation.

The plot is plausible insofar as the mean and maximum of the mass distribution are initially the same and start deviating from each other over time. However, even the structures formed at late times counterintuitively have less mass than small mass fluctuations in the early phase. This can be explained by the different redshifts of the full-sky time cones: the "slices" come not only from different ages of the Universe but also from different positions, meaning that the same pixel represents a larger cube in real space the higher the redshift/radial distance to us is. Feeding this time series of mass distribution into the NGRC setup presents several problems.
The first problem is that there are simply not enough time steps to train the weights in the output matrix W_{out}, which becomes even more critical considering that some of the 15 time steps must be reserved to compare the synthetic simulations with the original simulation. One way to address this problem is to interpolate between the time steps to ensure smooth transitions, allowing the output matrix to better adjust.
Naively, one could assume that, given 14 time steps, a polynomial of order 13 would suffice, where for each pixel the polynomial would be fitted to pass through each value at its corresponding time step. One could also take a polynomial of lower order if one accepts that the resulting fit will not pass exactly through every data point. The problem with this approach is that using a polynomial of any order would make the system, by definition, non-chaotic. The NGRC setup would then have no problem predicting such a simple system.
Instead, to better preserve the chaotic nature of the sparsely time-resolved original data, one can take advantage of Piecewise Cubic Hermite Interpolating Polynomials (PCHIPs). While PCHIPs use polynomials of order 3, similar to cubic splines, they have the advantage of being shape-preserving: they do not introduce artificial changes in the monotonicity of the original data and do not create new local maxima or minima (Rabbath and Corriveau, 2019). Granted, no interpolation method can

fully preserve the nonlinear dynamics of the original system (after all, these are still polynomials). However, this effect is mitigated the more data points the original dataset contains.
Moreover, this prediction should be considered a proof of concept, as it will later be applied to data with sufficient temporal resolution, making interpolation unnecessary.

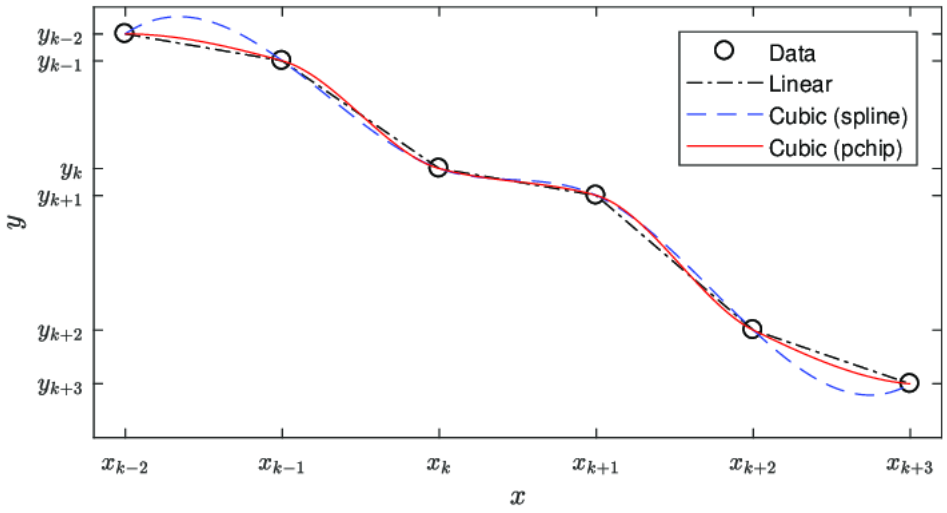


Fig. 22: Example of interpolating data points using different methods. The simplest method is linear interpolation, which has the clear disadvantage of not maintaining smooth transitions in the slope of the resulting curve. Interpolation with cubic splines or higher-order polynomials ensures smoothness in the first and even higher derivatives but can create artificial extrema. PCHIPs ensure smoothness in the first derivative while preserving the dynamics, in the sense that they do not create new extrema. The figure was taken from (Youssef et al., 2020).

Now that we have interpolated the original data, the next step is to decrease the resolution of each snapshot. Currently, each snapshot is saved as a Healpy file with $N_{\text{side}} = 4096$, which is equivalent to about 200 million pixels. With the given hardware limitations, prediction for more than 100,000 pixels is not feasible. For this reason, the data for each time step will be reduced to $N_{\text{side}} = 128$, meaning that we now have about 200,000 pixels per time step. Thankfully, `Healpy` provides the function `pixelfunc.ud_grade()`, which allows for an easy computation of a lower-resolution frame. Each frame still represents the integrated mass along the radial axis, meaning that the observable scales linearly with the number of pixels, which in turn scales with the square of N_{side}. Thus, to ensure that the total mass in each frame remains invariant after the transformation, we need to normalize each pixel value by $\left(\frac{N_{\text{side},out}}{N_{\text{side},in}}\right)^{-2}$.

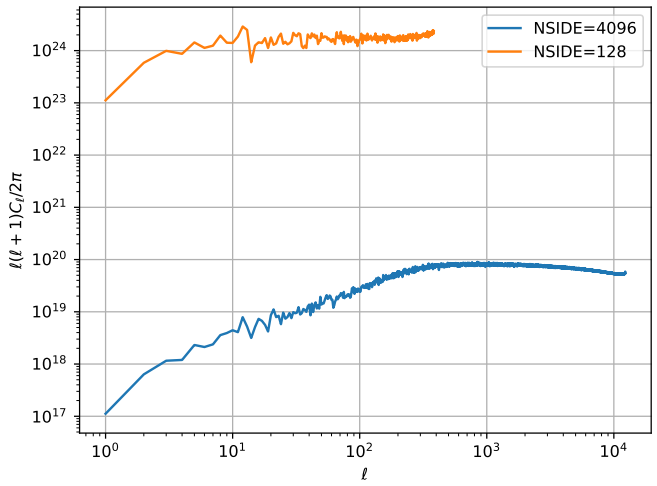


Fig. 23: Power spectrum of the first snapshot of the cosmological simulation calculated both for the original resolution as well as the lower resolution which was created by normalizing each pixel with an additional term to ensure mass conservation.

Unfortunately, normalizing each pixel value after the transformation using this term will lead to a shifted power spectrum (see equation 3.23), since multiplying each pixel value also results in uniformly higher or lower variations at all scales, as can be seen in Figure 23. One way to mitigate this problem, at least for large scales in real space, is to omit the normalization.

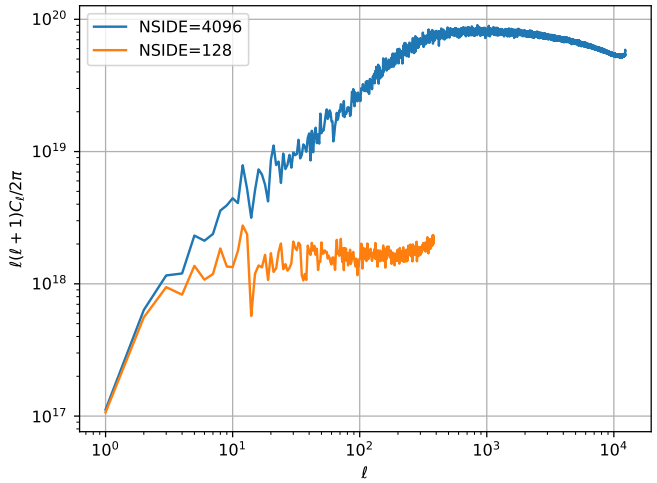


Fig. 24: Power spectrum of the first snapshot of the cosmological simulation calculated both for the original resolution as well as the lower resolution which was created without normalizing.

While mass conservation is no longer ensured, the resulting power spectrum is no longer orders of magnitude larger than the power spectrum at higher resolution. However, it begins to deviate from the higher-resolution power spectrum at larger ℓ. This is simply due to the fact that all small-scale structures (smaller than the pixel size at the lower resolution) corresponding to large ℓ are necessarily removed when decreasing the resolution.
An approximate estimation can be derived as follows: In `Healpy`, the maximum number of samples around a ring (i.e., at constant latitude) is $4{\cdot}N_{\mathrm{side}}$. To resolve all the m modes of spherical harmonics, one needs at least $2l + 1$ samples. Equating both terms and dividing by two gives then

$$l_{\mathrm{max}} = 2 \cdot \mathrm{NSIDE} = 256$$

So we can expect to see increasingly large differences emerging between the new and old power spectra at values larger than $l_{\mathrm{max}} = 256$. Non-conserved power spectra (and also higher-order point correlations) further limit the extent to which the true nonlinear interactions of the original simulation are preserved.
With the caveat in mind that lowering the resolution will always lead to different power spectra, we continue by taking the logarithm of the pixel values, since within a time step, pixel values span a wide range over many orders of magnitude, which would negatively influence prediction fidelity when used for training.
Finally, since the 200,000 pixels are still too many to be fed into the NGRC setup, a small patch of the whole sphere is extracted to further reduce the number of pixels being predicted. Hypothetically, it would also be conceivable to reduce the resolution of the whole sphere even further instead of taking a patch. However, this would only remove additional small-scale structures. The main concern with predicting several patches instead of the whole sphere, namely that the higher boundary-area-to-inner-area ratio reduces prediction quality will be examined later.
Lastly, all pixel values are normalized so that the standard deviation becomes 1, and the smallest pixel value is then subtracted from each pixel so that the minimum value becomes 0. One could also subtract the mean instead, but having both negative and positive pixel values reduces prediction quality. The final result can be seen in figure 25.

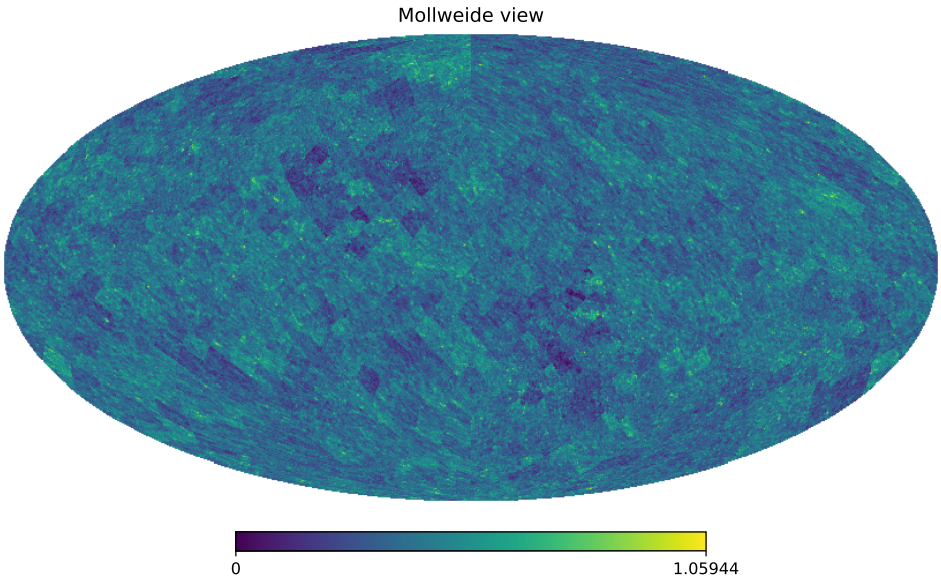


Fig. 25: Mass ditribution after lowering the resolution to $N_{\text{side}} = 128$, taking the logarithm and normalizing and shifting all pixel values so that that the standard deviation is 1 and the smallest pixel values is 0. This snapshot represents the last time step (see plot b) in 20)

We see that, as intended, the smallest pixel value is 0, and that smaller structures that could previously be resolved are now somewhat smoothed out. Interestingly, sharp and straight edges appear on the map after lowering the resolution, these result from the pixelization in `Healpy`.
For the training of the output matrix using the `Magneticum` simulation data—both during hyperparameter optimization and for all the subsequent full predictions, a total of $N_{train} = 129$ time steps were used. The prediction phase begins immediately after the 850th time step and it has a length of $N_{pred} = 32$. The following results were using the classical local states method.
Now that all necessary transformations have been applied to our original data, we can determine the hyperparameters required for optimal predictions. To do this, we proceed as described earlier using `Optuna`, noting that the optimization was performed on a small patch of the sphere rather than the full sphere. We then obtain the following hyperparameters, which will be used from this point onward.

Hyperparameter	Value
α	0.112
k	8
s	9

Tab. 7: Optimal hyperparameters for the prediction of the spherical `Magneticum` cosmological simulation.

Finally we can predict a large patch of the simulation of the sphere in figure 26.

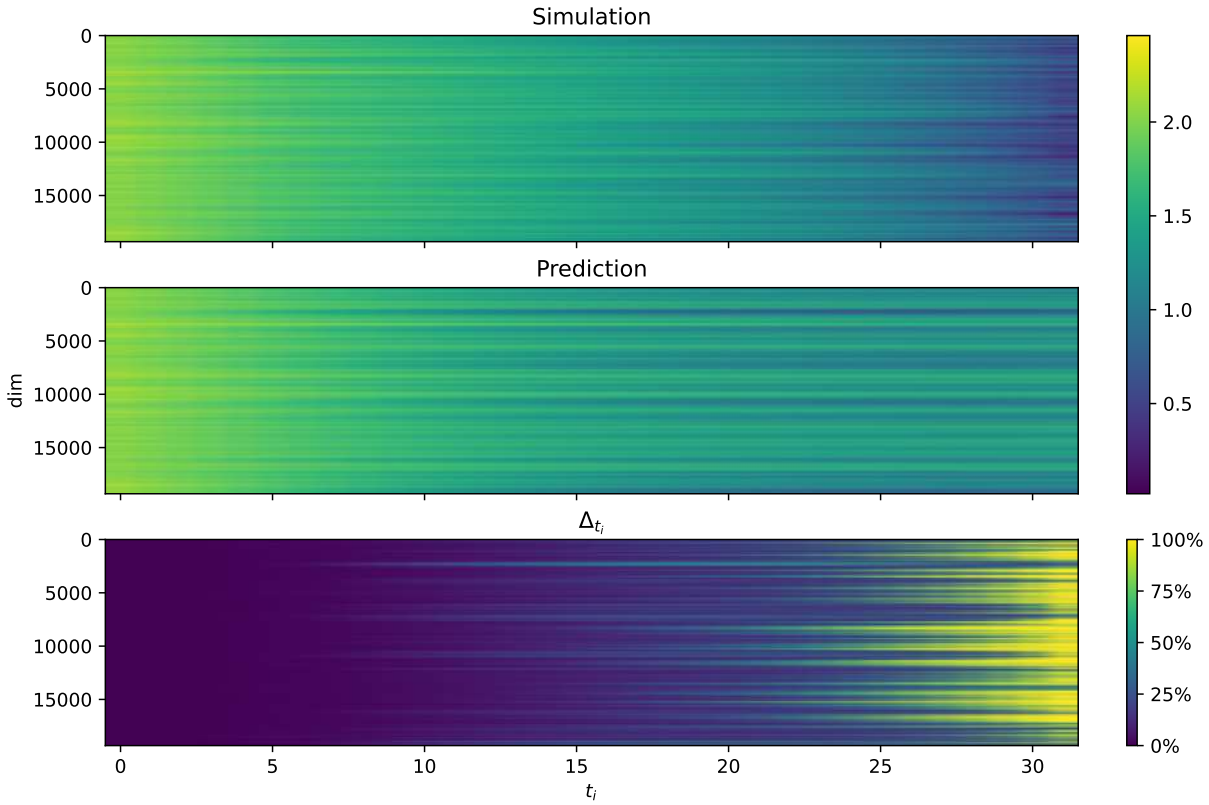


Fig. 26: Representation of the original and predicted patch of the spherical simulation. The X-axis shows the time step after the beginning of the prediction phase, while the Y-axis shows the index of the pixels in the 20,000-pixel field. At the bottom, the absolute relative difference between the prediction and the original simulation is shown in percent.

We can see that after 25 time steps, the deviation from the original simulation starts becoming increasingly large. Specifically, using equation 5.4, we find that the forecast horizon is reached at 32 time steps. However, this provisional reward function was introduced only to develop the necessary code. For a more detailed analysis, we will now compare using the root mean square error (RMSE):

$$\mathrm{RMSE}(t) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i^{\mathrm{pred}}(t) - y_i^{\mathrm{true}}(t)\right)^2} \tag{5.12}$$

where t indexes the current time step and i is the pixel index. This measure is independent of the resolution (i.e., the number of pixels per time step) and of the mean and standard deviation of the data (which is why we standardized the mean and standard deviation beforehand). We can then calculate the RMSE for each time step (see figure 27).

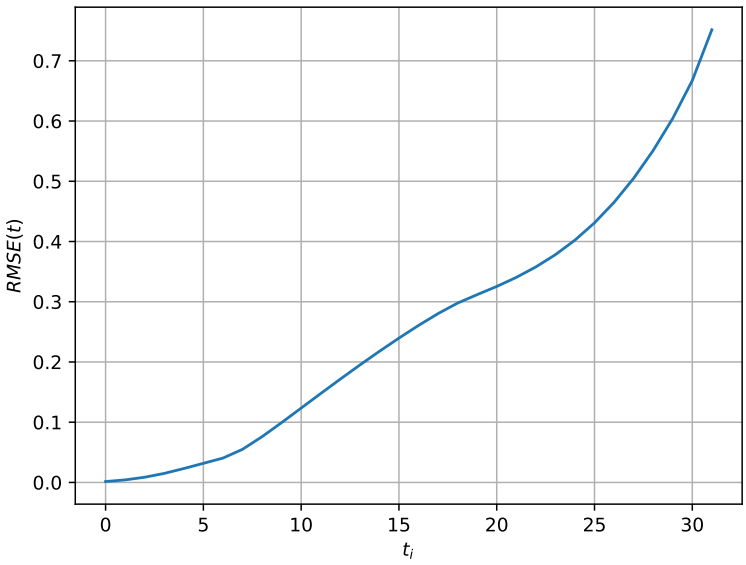


Fig. 27: RMSE per time step of a patch of the full spherical simulation after the logarithm of the original data was transformed to have suitable mean and standard deviation.

Qualitatively, the overall impression did not change. However, since the calculated RMSE(t) is now independent of the number of pixels in each snapshot/time step, we can compare how the prediction quality varies with the size of the excerpt taken from the full sphere. This is particularly interesting because it allows us to estimate how much better the prediction quality would be if we directly predicted the whole sphere at a lower resolution, instead of the current procedure where the full sphere has higher resolution but we perform several predictions on patches/excerpts taken from the sphere.
When comparing predictions of patches with varying pixel numbers/sizes, it should be considered that if the patches are too small, differences in prediction quality might simply arise because larger patches include information that is generally harder to detect. For example, a smaller patch might include only a void, while the largest patch includes several large-scale structures, which naturally leads to different prediction quality. To address this problem, we chose our smallest patch such that key statistical metrics, like the mean and standard deviation, no longer change with increasing patch size.

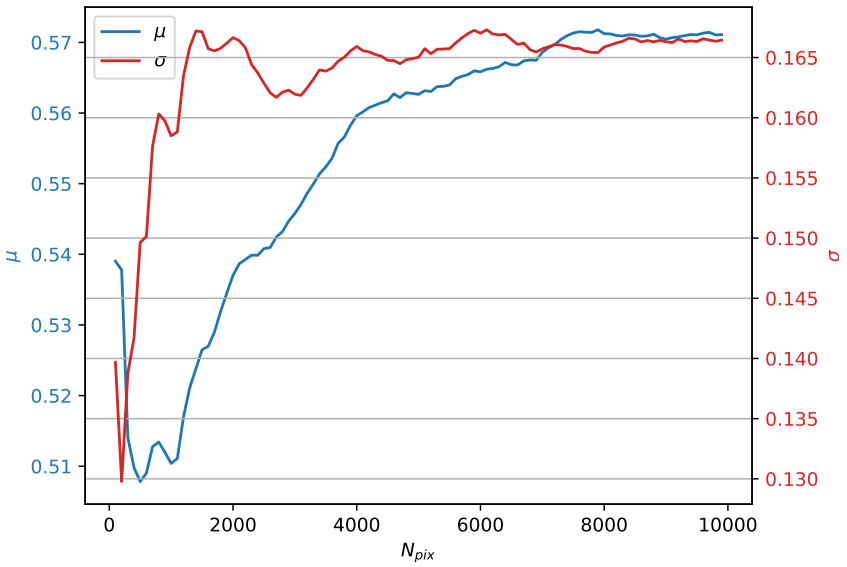


Fig. 28: The mean and standard deviation of all pixel inside of excerpt of the whole sphere as function of the size of the excerpt.

We see that at a patch size of about 4000 pixels the mean and the standard deviation we reach a plateau. From there on we start several predictions where the patches get larger by 200 pixels with every step. If we now compare how the RMSE(t) evolves in time for the different patches (see figure 29).

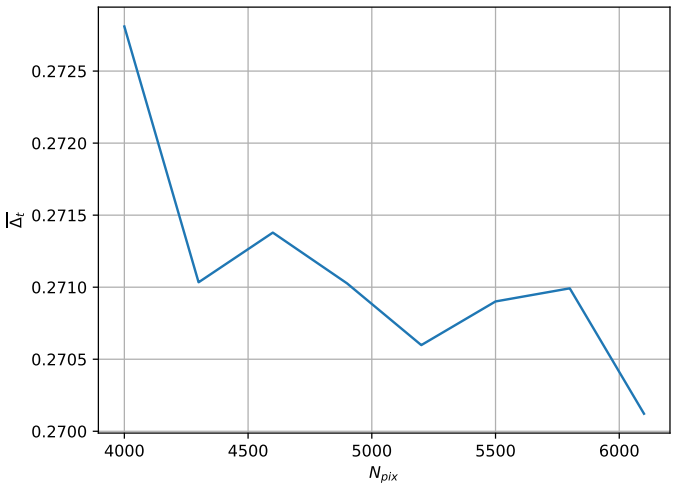


Fig. 29: RMSE(t) between the prediction and the original simulation for patches of the sphere that get larger sizes in 200 pixel increments.

As we can see, the error does decrease with increasing patch size, meaning that indeed the ratio between the number of core pixels in the boundary zone and the ones inside the patch has an influence on the prediction quality. Thus we can not use several smaller patches instead of the whole sphere to predict its behavior and expect the same forecasting fidelity.
Another reason why it is of interest to predict the full sphere is the qualitative difference of not having periodic boundaries when predicting a patch.
Now, the resolution of the sphere will be reduced to $N_{\text{side}} = 32$, corresponding to about 13,000 pixels. The result of the prediction can be seen in 30.

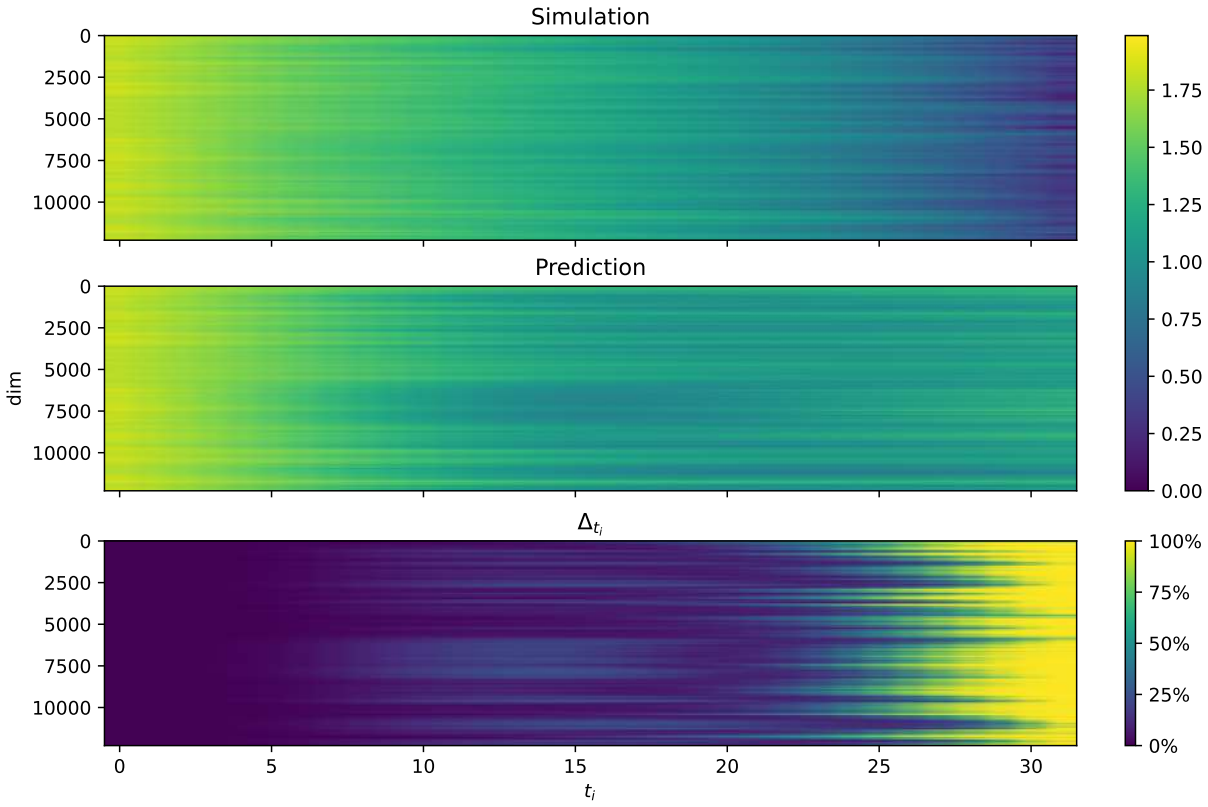


Fig. 30: Representation of the original and predicted full spherical simulation with $N_{\text{side}} = 32$. The X-axis shows the time step after the beginning of the prediction phase, while the Y-axis shows the index of the pixels. At the bottom, the absolute relative difference between the prediction and the original simulation is shown in percent.

Again, the prediction and the original simulation are in agreement up to the 27th time step. From there on, the original and predicted time series evolve increasingly differently, but this alone does not mean that the predictions are invalid. It may still be the case that the prediction conserves the values of some summary statistics, such as the power spectrum (encoding the 2-point correlation), the bispectrum (encoding the 3-point correlation function), or other higher-order point correlations. There is also a plethora of other summary statistics to compare the distribution of cosmological simulations such as a mass or peak histogram (Perraudin et al., 2019).

For now we start by calculating the power spectrum using equation 3.23 for each time step, both for the prediction and for the original simulation, and then compute the difference between the two.

$$\Delta P_{\ell,t_i} = \frac{\ell(\ell+1)}{2\pi} \frac{C_{\ell,t_i,pred} - C_{\ell,t_i,test}}{C_{\ell,t_i,test}} \tag{5.13}$$

While this power spectrum does not contain physical information per se, since we applied transfor-

mations to the original data before feeding it to the reservoir training, we can still compare the two power spectra, as both have undergone the same transformations.

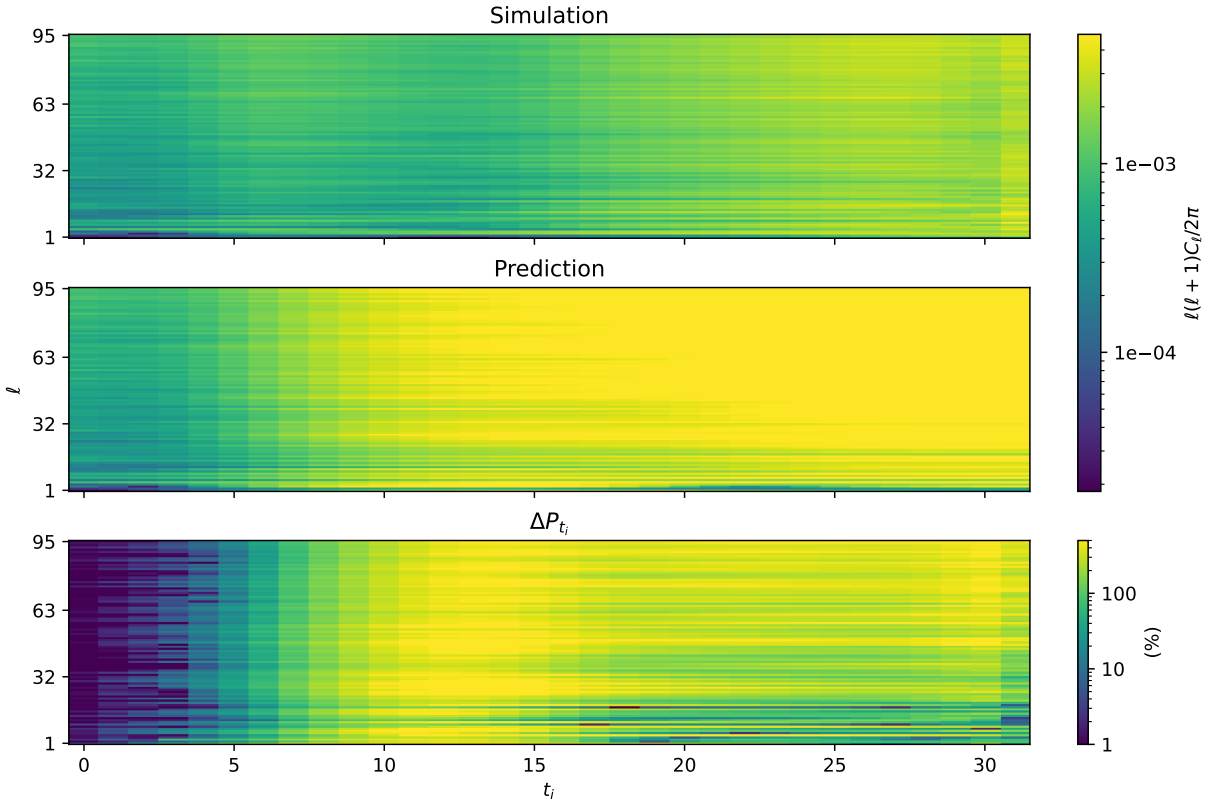


Fig. 31: Power spectra of the original and predicted simulation with $N_{\text{side}} = 32$ for each time step. The vertical axis shows the values of ℓ, which scale inversely with length in real space. The horizontal axis represents the time step t_i. The bottom plot shows the calculated difference between the two.

Interestingly, even though the pixel-wise differences in the raw values are very small and below 50% for the most part (figure 30), the corresponding power spectra show deviations of over 100%, indicating that after a short amount of time, the two-point correlations—especially on small scales—start deviating from the original simulation.

We could now continue calculating and comparing the bispectrum and examine higher-order correlations, but this becomes increasingly computationally expensive. One way to address this problem is by using Minkowski Functionals and Minkowski Tensors. Specifically, we calculate the ratio of the Eigenvalues of $W_1^{0,2}$ (see equation (3.39)) to obtain information about a preferred orientation in the structures. For this we use the python package `Litchi` (Collischon, 2024) and binarize the spherical maps with a threshold to then calculate $W_1^{0,2}$ 13 times each with a different threshold ranging from the 1st to the 99th percentile pixel value. Afterwards we take the average of those 13 Minkowski maps to get the final Minkowski map α_i, where i denotes the pixel index. Then we calculate the pixel wise relative difference between the Minkowski map of the original simulation

and the Minkowski map of the predicted behavior

$$\Delta\alpha_i = \left| \frac{\alpha_{test,i} - \alpha_{pred,i}}{\alpha_{test,i}} \right|. \tag{5.14}$$

The result can seen in figure 32.

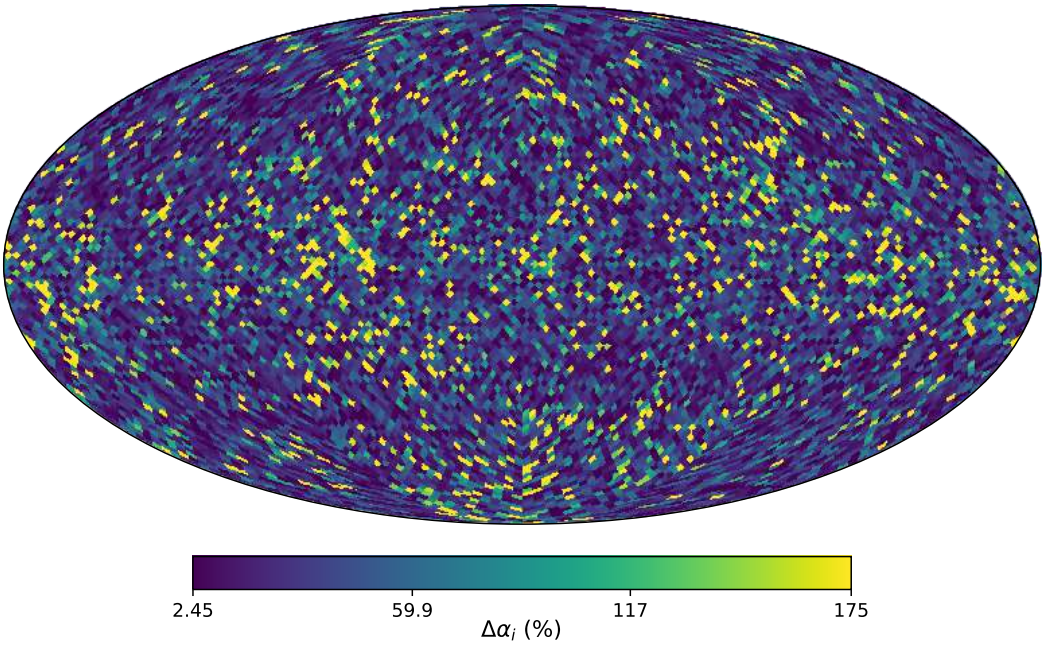
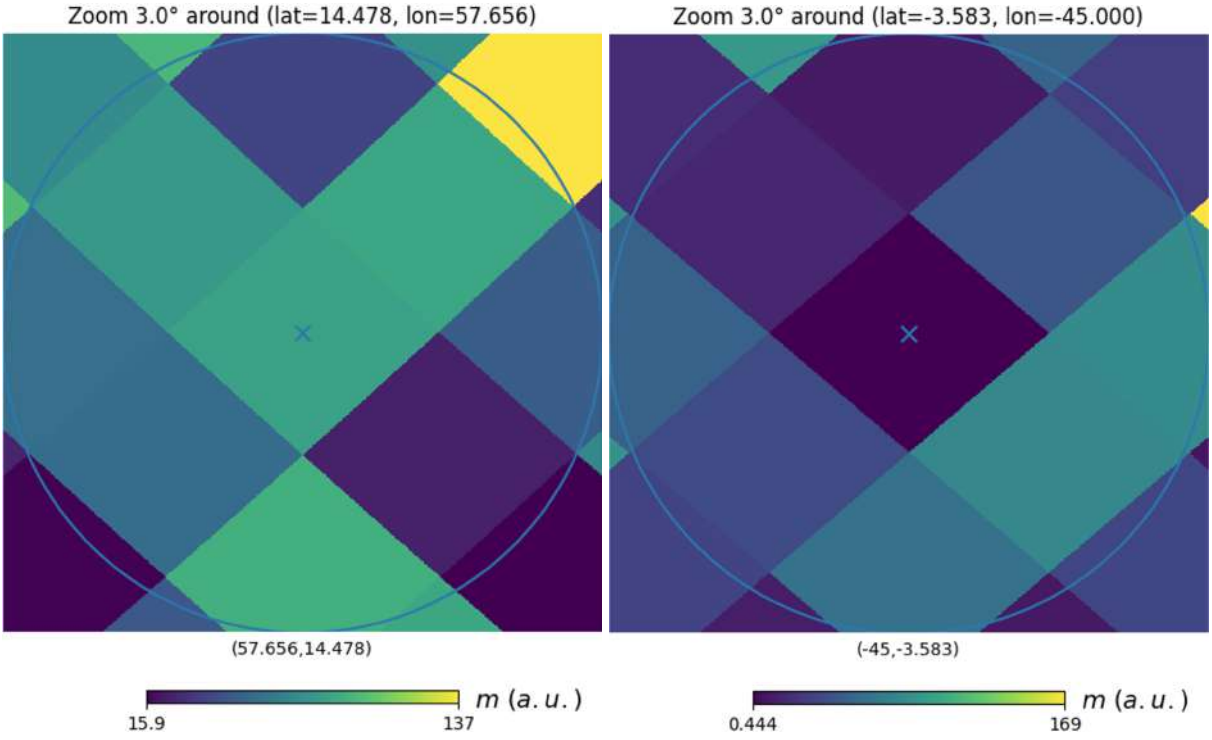


Fig. 32: Relative difference in the net orientation field $\Delta\alpha_i$ between the the net orientation of the original and of the predicted simulation for the time step $t_i = 135$.

We can see that for the most part the percentage difference in the net orientation is in the single digit range and it is only for small minority of pixels on the sphere where the deviation is larger than 100%. One possible explanation is that the large differences are the result of areas with very high or very low mass leading to difficulties in the prediction for the NGRC setup. We want to test this hypothesis and examine these areas in more detail by first calculating the 30 positions of the spots with angular radius of three degrees (the size of the spots in the net orientaion map with high deviation) in the spherical mass maps that have the highest $\{\vec{v}_{cluster,i}\}$ and also the lowest mass $\{\vec{v}_{void,i}\}$. So now we have five clusters as well as five voids with a radius of three degrees.



(a) (b)

Fig. 33: Zoom in on a three degree cluster on the left (a) and on the right the three degree large void (b). The excerpt are both from the original mass distribution m after taking the logarithm, lowering the resolution and normalizing and shifting the data.

Finally we calculate the average of the deviation in all of the clusters as well as in all of the voids and get results that are effectively equal and, more notably, exhibit a very large spread/variance.

$$\Delta\alpha_{cluster} = 60 \pm 50$$
$$\Delta\alpha_{void} = 60 \pm 50$$

and indeed we can observe if we plot the rank (starting from most dense to least dense cluster and vice versa for the void) that that the two lines are from the beginning close to each other as can be seen in 34.

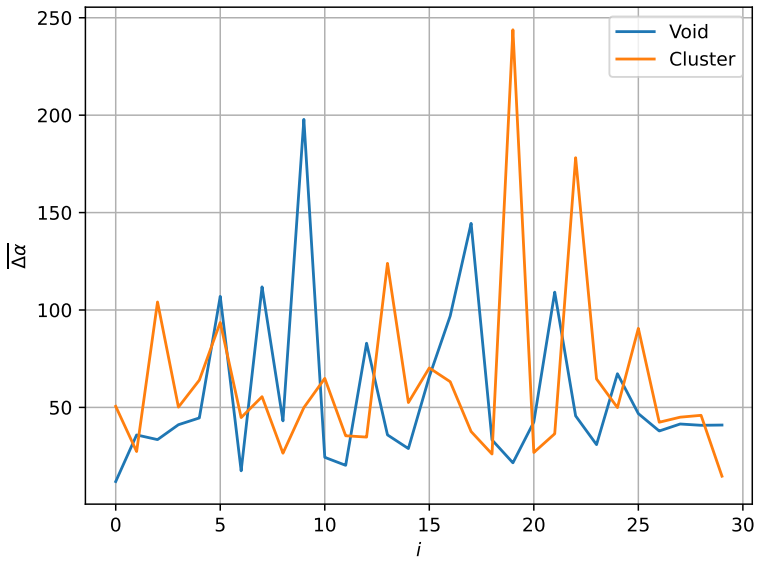


Fig. 34: Relative difference in the net orientation field $\Delta\alpha_i$ between the the net orientation of the original and of the predicted simulation for the time step $t_i = 135$.

Furthermore they both neither decrease nor increase. From this we can reject our hypothesis as it is: it is neither areas with high nor areas with very low pixel values that show high deviations in the net orientation between surrogate and original simulation. Thus we can reject earlier stated hypothesis that there is the pattern that extrema in the mass distribution are responsible for deviations in the net orientation.

5.4 COMPASS Simulation

Next, we want to apply our NGRC setup to the zoom-in simulation data from COMPASS (Steinwandel et al., 2023). The simulation that is being predicted is a cube with dimensions $3.59 \cdot 10^4 \times 3.59 \cdot 10^4 \times 1.79 \cdot 10^4$ kpc^3 evolving in time. For each time step, the mass distribution of the cube is summed along the z-axis to obtain a surface density. As a result the data is formatted as a flat 2D plane with a resolution of 2048×2048 pixels.

Another major difference compared to the spherical `Magneticum` simulation is that here it is always the same cube used for the simulation data, whereas in the `Magneticum` simulation, changes in the mass distribution arise not only from gravitational interactions between particles but also because the same pixels at different time steps correspond to volumes in real space at different positions and with different sizes. Thus, with this new dataset, we can examine in an isolated manner the prediction quality on non-linear physical processes free from effects of how the mass distribution was discretized.

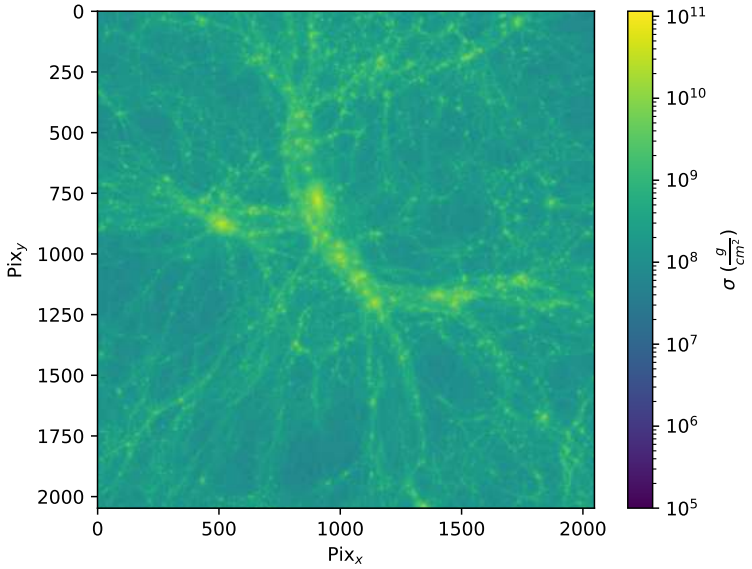


Fig. 35: Projected mass density distribution along the z axis for one snapshot of the `Compass` simulation. The x and y axis show the pixel indices along the axis. The Simulation has a spatial resolution of 2048×2048 pixels and a temporal resolution of $N_t = 780$.

Before we can feed the data into our algorithm, we need to apply a few transformations, similar to those used for the spherical simulation data: First, we take the logarithm of the original data. Then, we scale down the resolution of the data to 64×64 pixels. We do so by averaging the density over the original pixels corresponding to one new, lower resolution, pixel as the density is an intrinsive whose sum over the whole domain is not being conserved. Finally, we normalize by the standard deviation of the pixel values and subtract the smallest pixel value from all pixels so that the standard deviation is now one and the smallest value of the data set is zero.

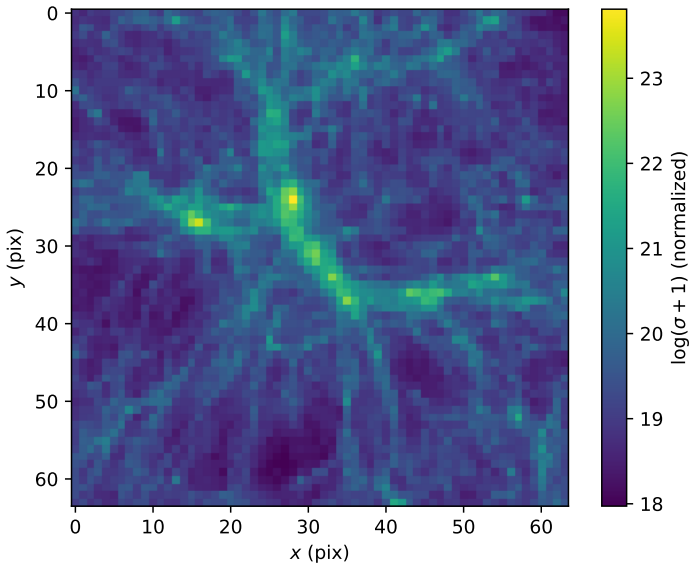


Fig. 36: Projected mass density distribution along the z axis for one snapshot of the `Compass` simulation, after decreasing the resolution, taking the logarithm and normalizing and shifting the pixel values. The x and y axis show the pixel indices along the axis.

For the training of the output matrix using the COMPASS simulation data, both during hyperparameter optimization and for all the subsequent full predictions, a total of $N_{train} = 850$ time steps were used. The prediction phase begins immediately after the 850th time step with varying time step lengths. The following results were using the classical local states method.
Again before starting the prediction on the full simulation we optimize the hyperparameters on a smaller patch, here with dimensions 32×32 pixel.

Hyperparameter	Value
α	0.0301
k	10
s	17

Tab. 8: Optimal hyperparameters for the prediction of the planar 2D cosmological COMPASS simulation.

Using the hyperparameters in table 8 we can now predict the full 2D flat plane.

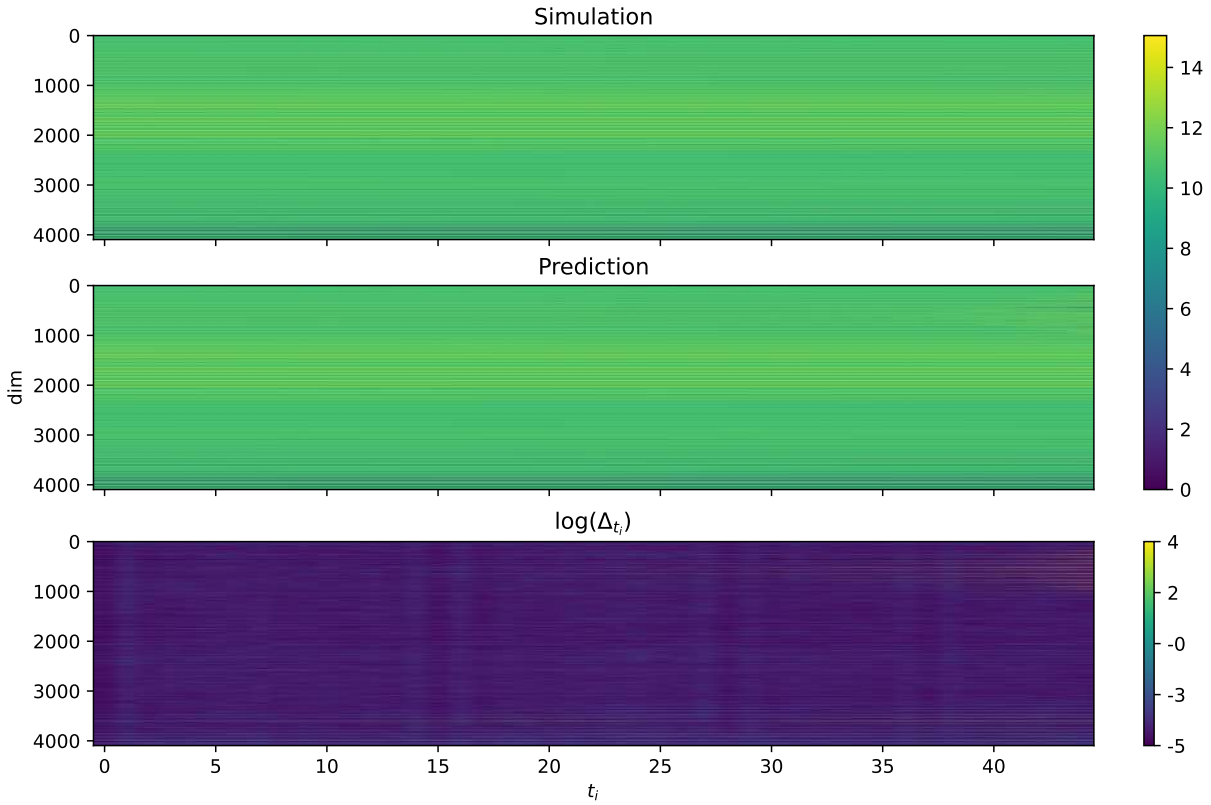


Fig. 37: original and predicted evolution of the `Compass` simulation after taking the logarithm, lowering the resolution to 64×64 pixel and normalizing and shifting the pixel values.

Here too the prediction agrees with the original simulation up to the 40th (for most pixels the difference between original and prediction is below 10%), before they start deviating from each other. Interestingly, this time the error starts to grow rapidly and much faster compared to the prediction of the spherical `Magneticum` simulation. Specifically, the pixel indices around the 500th pixel deviate from the original simulation by 10,000% earlier than the other pixels. This is further illustrated by examining the RMSE(t) shown in figure 38.

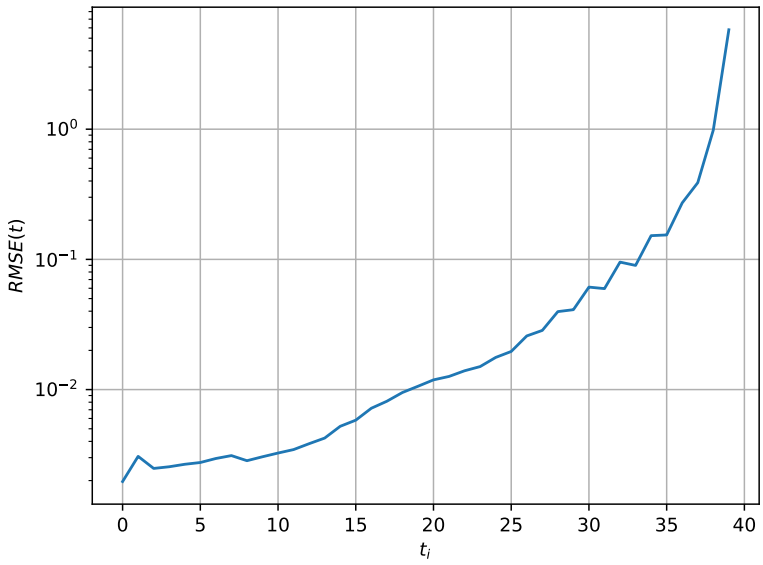


Fig. 38: RMSE per time step of the prediction of the scaled and normalized full 2D flat `COMPASS` simulation.

Once again, we see that while the RMSE between the prediction and the original is lower in the first time steps compared to the spherical simulation (27), it also grows much faster over different magnitudes, possibly indicating that the hyperparameters of the NGRC are not yet perfectly optimized. Note that these surrogate simulations were created considerably faster than the original simulations. In the case of the COMPASS simulation the production of the data required 3,000,000 CPU hours while the surrogate created with NGRC framework would require only 30,000 CPU hours meaning that this new methods improves the speed by a factor of 100. The 30,000 CPU hours of the NGRC were obtained by extrapolating from the amount of computational resources that were used to predict the excerpt of the full simulation in figure 37. Future efforts could focus on predicting the full simulation on super computers and compare them with the extrapolated value of 30,000 CPU hours. In addition to short term congruence we are interested in preserving the statistical properties of the original simulation in our prediction. We can again calculate the power spectrum and its difference between the original simulation and the prediction. This time, we use equation 3.15, since we are no longer dealing with periodic data.

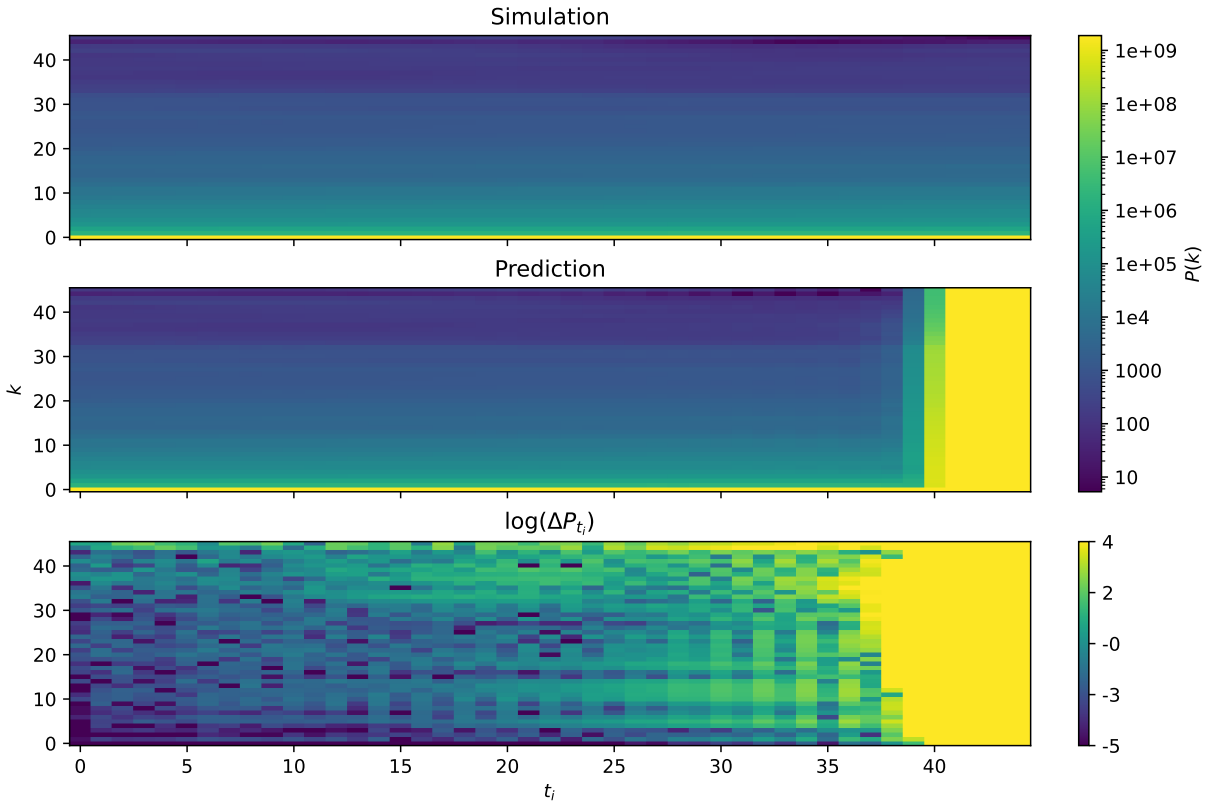
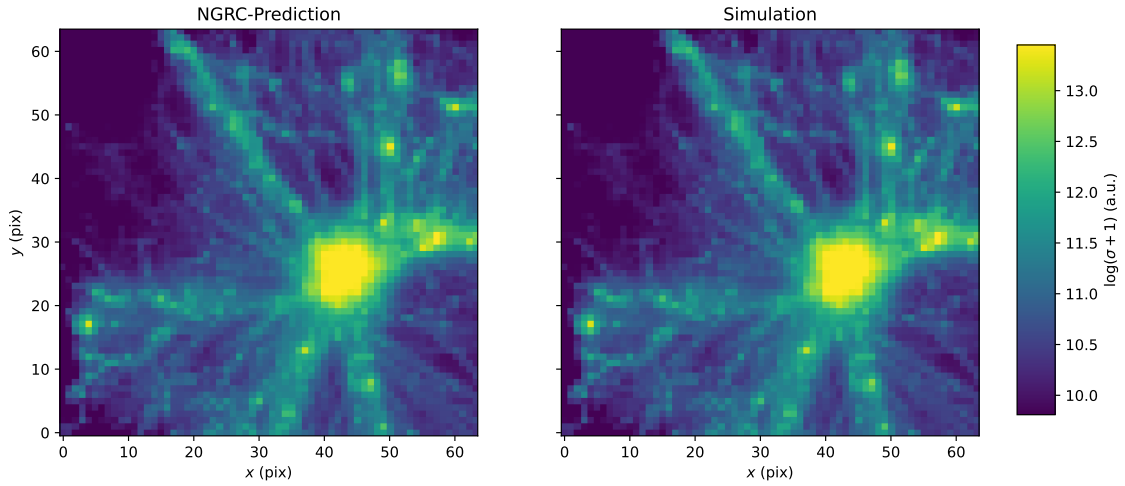
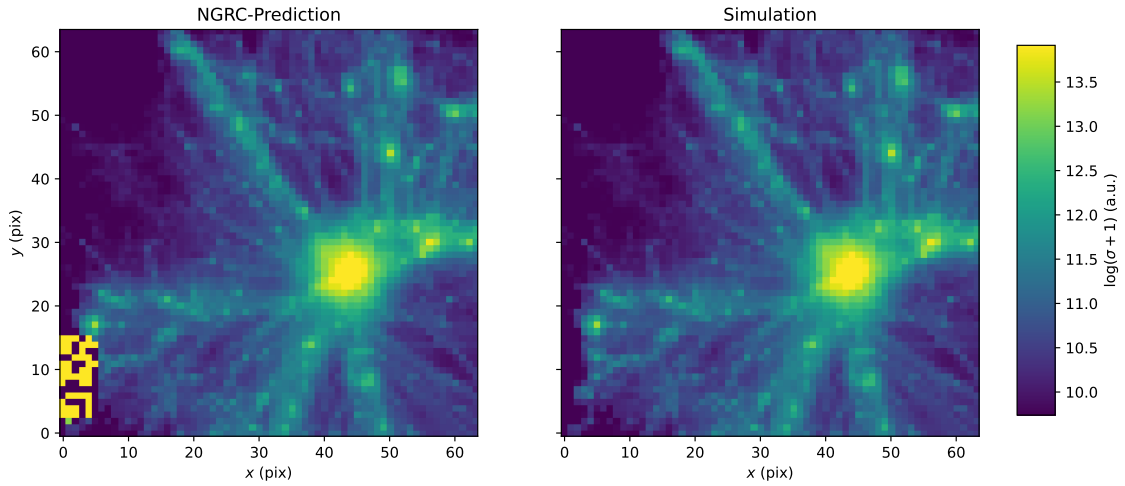


Fig. 39: Power spectra of the original and predicted `COMPASS` simulation after taking the logarithm, lowering the resolution to 64×64 pixels, and normalizing the pixel values.

We now see again that the error between the original and predicted power spectrum is initially small but grows rapidly, becoming several orders of magnitude larger (a factor of roughly 10^9) than the error in the first time steps. Another observation is that the prediction is better at conserving the statistical properties for small k values, corresponding to large scales in real space. This makes using the NGRC method for generating and analysing small scale structures less favourable than for analysing characteristics on large scales.
The incongruity between the results for the prediction of the power spectrum and raw pixel value difference can be explained when we have a look at the reconstructed mass density distributions at a time step where differences in the power spectra are small ($t_i = 10$) and a time step where the power spectra of original and synthetic simulation deviate by orders of magnitudes ($t_i = 44$).



(a)

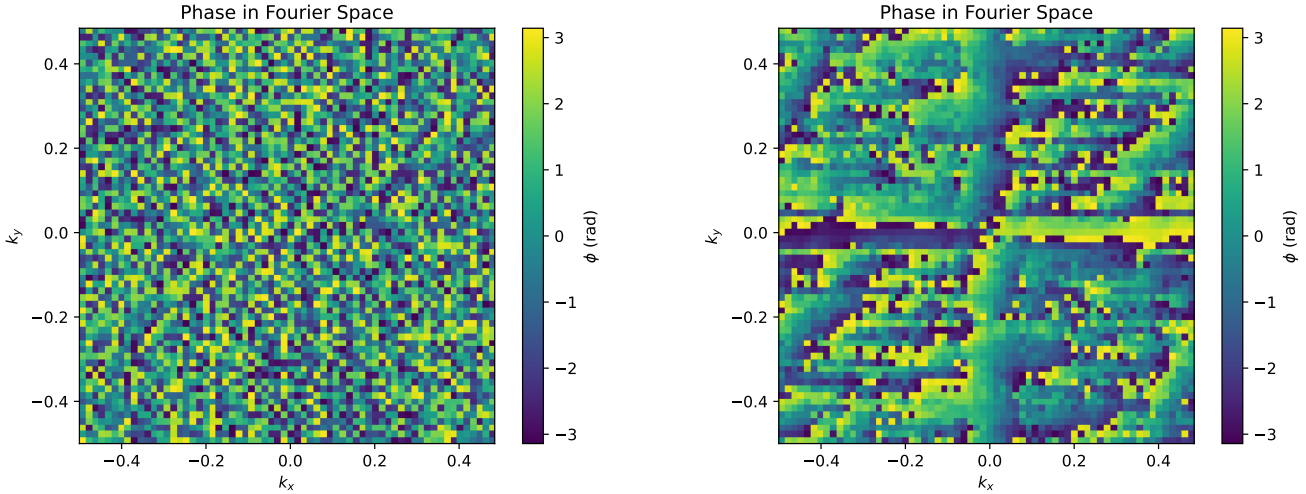


(b)

Fig. 40: Comparison of the transformed mass density distribution between the predicted and original simulation at time steps $t_i = 10$ (a) and $t_i = 44$ (b).

As can be seen in figure 40 the large deviation in the predicted power spectrum is primarily driven by extreme errors in a small number of pixels located in the bottom-left region of the domain. Although the number of affected pixels is small and their spatial extent is limited, their disproportionately large deviation is sufficient to significantly distort the power spectrum. The origin of this sudden error can be traced back to the boundary conditions of the simulation. In the original simulation, new material continuously flows into the domain from the edges. However, at approximately 40 time steps, the mass density entering from the bottom-left corner abruptly assumes a value of exactly

zero. This stands in strong contrast to the typical pixel values at all other times, which are strictly positive and significantly larger than zero. The resulting sharp discontinuity and the emergence of an extended region of zero-valued pixels create a discontinuous spot in the data. This abrupt cut acts as a seed for rapidly growing prediction errors, which subsequently propagate and manifest as large deviations in the raw pixel value and finally the power spectrum.
Another aspect we want to examine is how well our NGRC method performs at varying stages of the simulation. As was already explained in Chapter 3.1, initial Gaussian fluctuations started collapsing in a first linear and then non-linear manner.
To determine the degree of this non-linearity following (Coles and Chiang, 2000), we can calculate the Fourier transform of each time step and examine the phases for each (k_x, k_y) pair.



(a) First time step

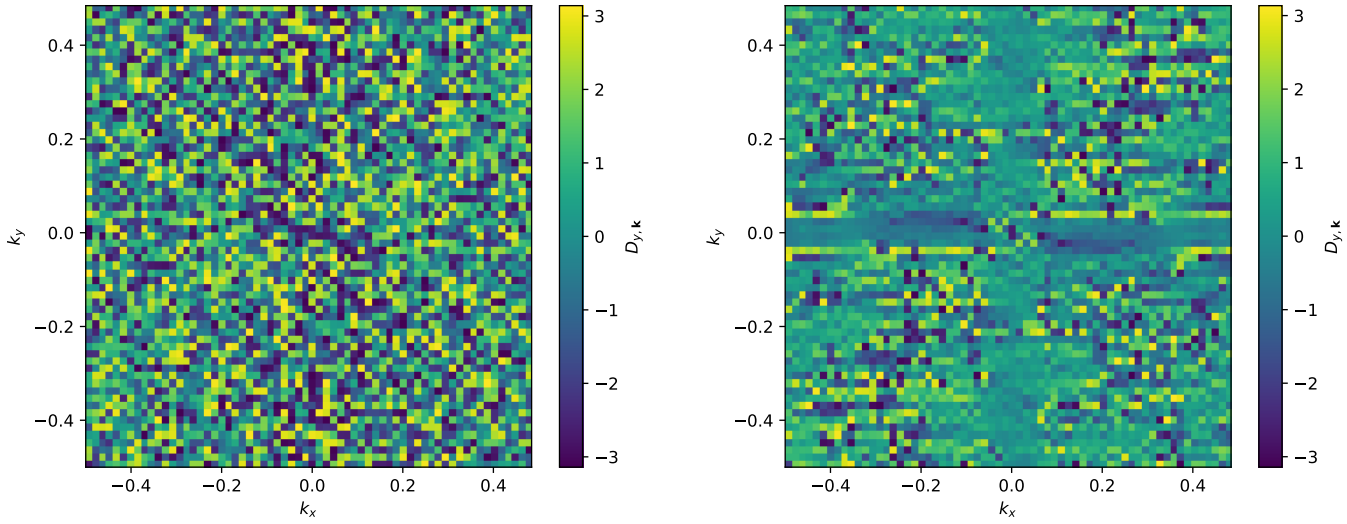
(b) Last time step

Fig. 41: Phases of the Fourier transform for two different time steps. (a) Shows the first time step in the evolution of the mass density distribution, where the Gaussian random field exhibits no pattern, compared to (b), which represents the last time step at which clear structures have formed.

As we can see in figure 41, as soon as structures emerge in the data, the phases of their corresponding Fourier transform start becoming increasingly correlated. To quantify the level of correlation, we first calculate the gradient along both the x and y axes for every k value.

$$D_{x,\mathbf{k}} = \phi_{k_x+1,k_y} - \phi_{k_x,k_y} \tag{5.15}$$
$$D_{x,\mathbf{k}} = \phi_{k_x,k_y+1} - \phi_{k_x,k_y} \ . \tag{5.16}$$

The resulting gradient field for the aforementioned time steps appear in figure 42.



(a) First time step

(b) Last time step

Fig. 42: Gradient field $D_{y,\mathbf{k}}$ of the phases of the Fourier transform for two different time steps. (a) shows the first time step in the evolution of the mass density distribution, where the Gaussian random field exhibits no pattern, compared to (b), which represents the last time step at which clear structures have formed.

With the probability density function (PDF) of the values in the gradient field $f(D_{x/y,\mathbf{k}})$ we can calculate an entropy $S_{x/y}(t_i)$ for both axes for each time step

$$S_x(t_i) = -\int_{-2\pi}^{2\pi} f(D_{x,\mathbf{k}}) \log\left[f(D_{x,\mathbf{k}})\right] dD_{x,\mathbf{k}} \tag{5.17}$$

$$S_y(t_i) = -\int_{-2\pi}^{2\pi} f(D_{y,\mathbf{k}}) \log\left[f(D_{y,\mathbf{k}})\right] dD_{y,\mathbf{k}} \ . \tag{5.18}$$

The smallest entropy one can get is 0 while the largest entropy corresponds to a random Gaussian field. In such a field there is correlation between phases and so the PDF is

$$f_{Gau}(D) = \frac{1}{4\pi}. \tag{5.19}$$

Plugging this PDF into the equation 5.17 yields

$$S_{max}(t_i) = -\int_{-2\pi}^{2\pi} \frac{1}{4\pi} \log\left(\frac{1}{4\pi}\right) dD_{x,\mathbf{k}} = \log(2\pi) \approx 1.84 \ . \tag{5.20}$$

Since we are not specifically interested in the entropy along a specific axis the arithmetic mean was calculated for every time step and for the entropy of a Gaussian field

$$S(t_i) = \sqrt{S_x(t_i)^2 + S_y(t_i)^2} \text{ with } S_{max} = \sqrt{2\log(2\pi)^2} \approx 2.6 \ . \tag{5.21}$$

Figure 43 illustrates the time evolution of the entropy in the original simulation.

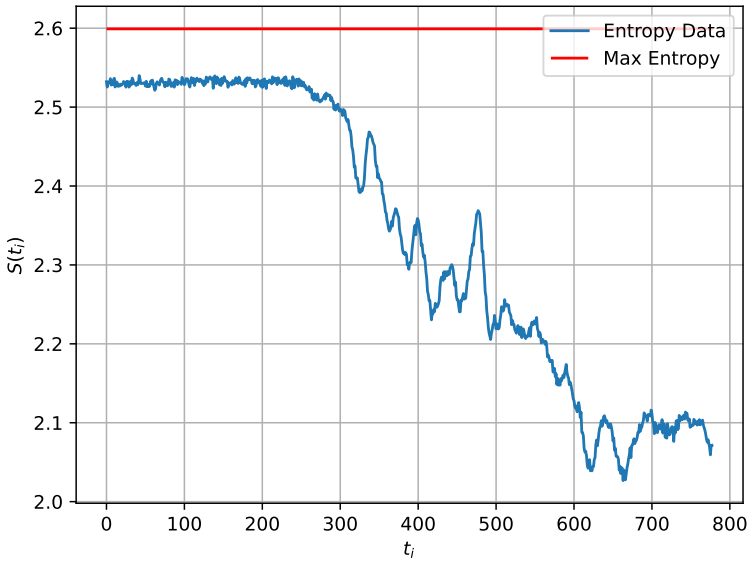


Fig. 43: The blue line shows the entropy for each time $S(t_i)$ step while the red line representing the theoretical maximum value of the entropy as a reference.

As expected, at the beginning of the simulation, when we are in the linear regime, the calculated entropy has its highest value. This aligns with our expectations because, without any emerging structures, the amount of information we can extract from the data is at its lowest (Lairez, 2022). Over time, we enter the non-linear regime, where structures start forming, and thus the amount of information we can extract from the data increases, causing the entropy to decrease accordingly.

Now that we have established entropy as a proxy for non-linearity, we perform several predictions starting at different time steps of the original data

$$t_{start} \in \{160; 215, 270; 325; 380; 435; 490; 545; 600; 655; 710\}$$

with a length of 40 time steps. We can then evaluate how fast the error between the prediction and the original simulation grows per time step within this prediction time window

$$\partial_t \Delta = \left[\sum_{t_i=21}^{40} \left(\sum_j \left| \frac{y_{pred,j}(t_i) - y_{test,j}(t_i)}{y_{test,j}(t_i)} \right| \right) \right] - \left[\sum_{t_i=1}^{20} \left(\sum_j \left| \frac{y_{pred,j}(t_i) - y_{test,j}(t_i)}{y_{test,j}(t_i)} \right| \right) \right], \quad (5.22)$$

where j is the pixel index of the pixels of one time step. We decided to consider a time window of 40 steps, as this is the duration after which the power spectrum between original and prediction starts to deviate considerably (see figure 39). We can then plot the growth of the error against the entropy (see figure 44).

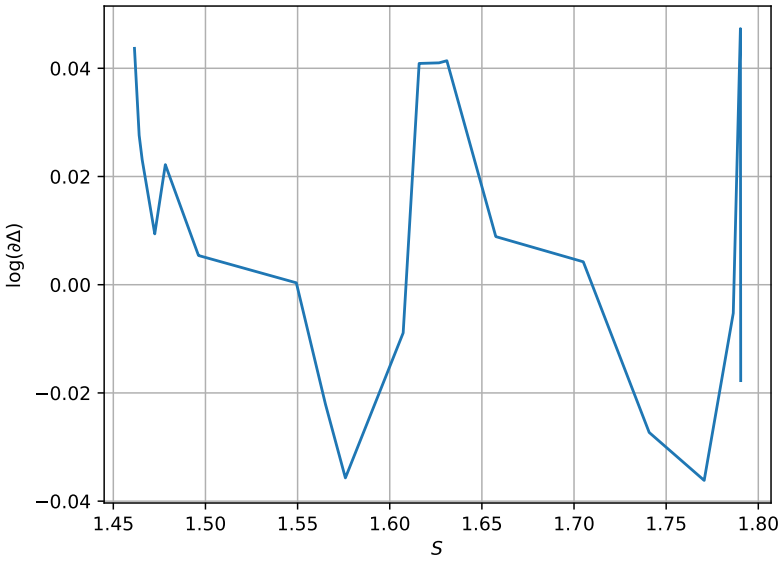


Fig. 44: Error growth per unit time between the original and predicted simulations, $\log(\partial\Delta)$, within a given time window plotted against the average entropy in that time window, $\overline{S}$.

As we can see, there is no observable tendency for the error growth per time step, $\partial_t\Delta$, to increase with decreasing entropy. However, a more definitive answer would require additional data points. Continuing in the same manner by comparing the power spectra.

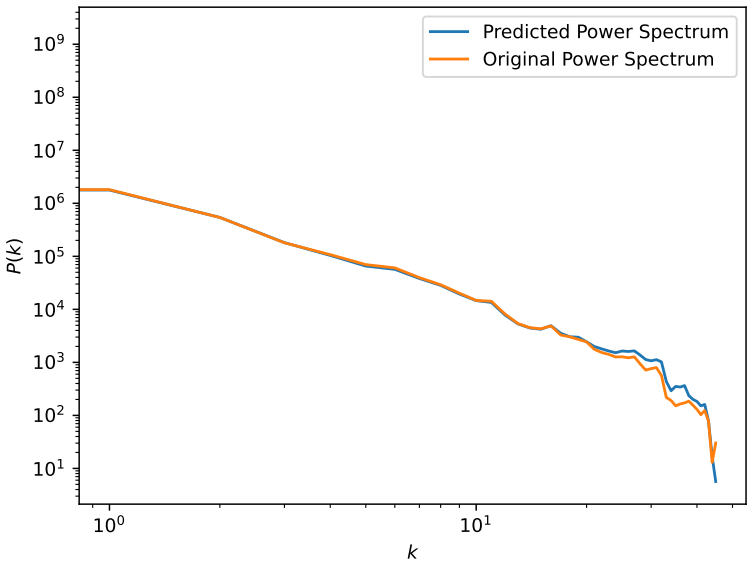


Fig. 45: Comparison of the radial power spectra $P(k)$ between the original and predicted simulation for a given time step.

to equation 5.23 for the power spectra and plot the growth in the error per time step against the entropy of the data

$$\partial_t \Delta P = \left[\sum_{t_i=21}^{40} \left(\sum_k \left| \frac{P_{pred,j}(k) - P_{test,j}(k)}{P_{test,j}(k)} \right| \right) \right] - \left[\sum_{t_i=1}^{20} \left(\sum_k \left| \frac{P_{pred,j}(k) - P_{test,j}(k)}{P_{test,j}(k)} \right| \right) \right], \quad (5.23)$$

where here k is referring to the radial wave vector which is independent of direction. We then can plot the error growth in the power spectrum against the entropy.

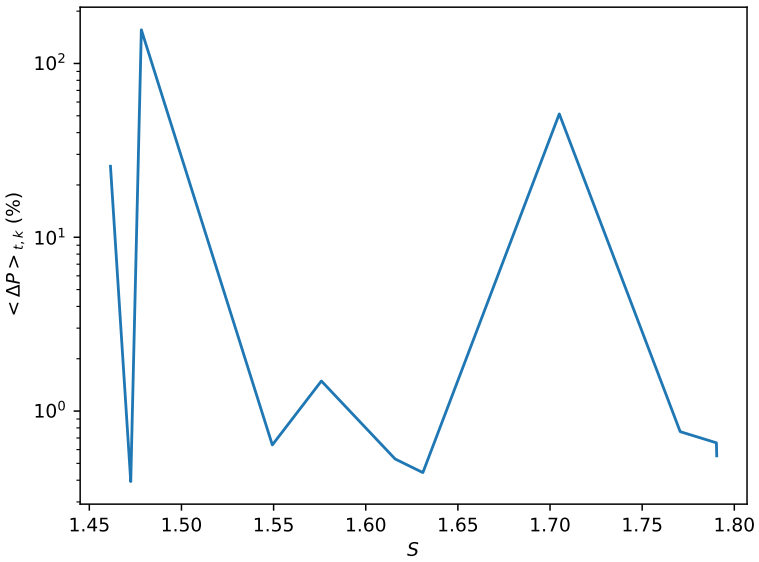


Fig. 46: Power spectrum error growth per unit time between the original and predicted simulations, $< \Delta P >_{t,k}$, within a given time window plotted against the average entropy in that time window S.

Again we can not see any tendency that the error growth of the power spectrum increases with decreasing entropy further confirming earlier observation. We can conclude that the non-linearity of the original simulation does not influence the quality of the predictions neither for pixel wise difference nor for the power spectrum as summary statistic. However it should be noted that there are currently only 18 data points so more data would give more significant results.

6 Summary and Outlook

To summarize, we started by investigating how different compositions of local states affect the prediction quality of dynamical systems with a preferred orientation. In this context, we introduced and validated the novel adaptive local states method, which enables both faster and more accurate predictions.
Furthermore, we developed a framework for predicting spatio-temporal dynamics defined on a spherical surface with the `HEALPix` pixelisation scheme. The functionality of this framework was verified using synthetic chaotic wave simulations on the sphere using the scalar wave equation. We then applied the framework for spherical data to predict the projected mass density of cosmological simulations. While the prediction of the raw pixel values performed better than the baseline (i.e., assuming the simulation remains frozen at the final training time step), other statistical measures, such as the power spectrum, showed earlier and more pronounced deviations from the ground truth. Using the RMSE as an evaluation metric, we additionally examined how the ratio between boundary area and inner area of spatial region of the full spherical data influences prediction quality.
We applied a NGRC instance to predict simulations of flat, two-dimensional cosmological mass distribution evolutions. In this case, the pixel-wise error showed significantly faster growth compared to the spherical simulations. We analysed both the temporal evolution of the pixel-wise error and the deviation in the power spectra, relating them to the entropy at different simulation times. From this, we inferred that the degree of nonlinearity in the structure formation does not appear to significantly affect prediction quality. However, a more accurate study of this effect would require simulations covering a longer time span.
It should also be noted that all NGRC instances in this work were trained using a loss function that minimizes the pixel-wise error between prediction and simulation. Therefore, an interesting direction for future research would be to investigate training strategies that prioritize the preservation of statistical properties over long time spans, even if individual pixel-wise deviations remain large or even both where both short term pixel wise difference and long term deviations in statistical properties are included.
Finally, it would be helpful to repeat the same analysis for the planar two-dimensional cosmological simulations with new simulation data where the original data do not exhibit an abrupt decrease of values to zero at certain regions after a few time steps in order obtain predicted power spectra that are not by orders of magnitudes deviating from the original power spectrum and subsequently better assess how non linearity influence the prediction quality.
Looking forward, further development of the NGRC framework could accelerate and enhance simulation based cosmological inference by reducing the computational cost of large-scale simulations and enabling rapid generation of mock simulations for comparison with observational data. Also such surrogate or emulator-based prediction strategies could help focus costly high-precision simulations on narrower regions of parameter space that are most relevant for specific observational targets by creating mock simulations on a wide range of parameters and discarding the ranges deemed not useful. In this way the can contribute to refine constraints on cosmic structure, dark matter, and cosmic microwave background features subsequently on cosmological parameters.

References

N. Aghanim, Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A. J. Banday, R. B. Barreiro, N. Bartolo, S. Basak, R. Battye, K. Benabed, J.-P. Bernard, M. Bersanelli, P. Bielewicz, J. J. Bock, J. R. Bond, J. Borrill, F. R. Bouchet, F. Boulanger, M. Bucher, C. Burigana, R. C. Butler, E. Calabrese, J.-F. Cardoso, J. Carron, A. Challinor, H. C. Chiang, J. Chluba, L. P. L. Colombo, C. Combet, D. Contreras, B. P. Crill, F. Cuttaia, P. de Bernardis, G. de Zotti, J. Delabrouille, J.-M. Delouis, E. Di Valentino, J. M. Diego, O. Doré, M. Douspis, A. Ducout, X. Dupac, S. Dusini, G. Efstathiou, F. Elsner, T. A. Enßlin, H. K. Eriksen, Y. Fantaye, M. Farhang, J. Fergusson, R. Fernandez-Cobos, F. Finelli, F. Forastieri, M. Frailis, A. A. Fraisse, E. Franceschi, A. Frolov, S. Galeotta, S. Galli, K. Ganga, R. T. Génova-Santos, M. Gerbino, T. Ghosh, J. González-Nuevo, K. M. Górski, S. Gratton, A. Gruppuso, J. E. Gudmundsson, J. Hamann, W. Handley, F. K. Hansen, D. Herranz, S. R. Hildebrandt, E. Hivon, Z. Huang, A. H. Jaffe, W. C. Jones, A. Karakci, E. Keihänen, R. Keskitalo, K. Kiiveri, J. Kim, T. S. Kisner, L. Knox, N. Krachmalnicoff, M. Kunz, H. Kurki-Suonio, G. Lagache, J.-M. Lamarre, A. Lasenby, M. Lattanzi, C. R. Lawrence, M. Le Jeune, P. Lemos, J. Lesgourgues, F. Levrier, A. Lewis, M. Liguori, P. B. Lilje, M. Lilley, V. Lindholm, M. López-Caniego, P. M. Lubin, Y.-Z. Ma, J. F. Macías-Pérez, G. Maggio, D. Maino, N. Mandolesi, A. Mangilli, A. Marcos-Caballero, M. Maris, P. G. Martin, M. Martinelli, E. Martínez-González, S. Matarrese, N. Mauri, J. D. McEwen, P. R. Meinhold, A. Melchiorri, A. Mennella, M. Migliaccio, M. Millea, S. Mitra, M.-A. Miville-Deschênes, D. Molinari, L. Montier, G. Morgante, A. Moss, P. Natoli, H. U. Nørgaard-Nielsen, L. Pagano, D. Paoletti, B. Partridge, G. Patanchon, H. V. Peiris, F. Perrotta, V. Pettorino, F. Piacentini, L. Polastri, G. Polenta, J.-L. Puget, J. P. Rachen, M. Reinecke, M. Remazeilles, A. Renzi, G. Rocha, C. Rosset, G. Roudier, J. A. Rubiño-Martín, B. Ruiz-Granados, L. Salvati, M. Sandri, M. Savelainen, D. Scott, E. P. S. Shellard, C. Sirignano, G. Sirri, L. D. Spencer, R. Sunyaev, A.-S. Suur-Uski, J. A. Tauber, D. Tavagnacco, M. Tenti, L. Toffolatti, M. Tomasi, T. Trombetti, L. Valenziano, J. Valiviita, B. Van Tent, L. Vibert, P. Vielva, F. Villa, N. Vittorio, B. D. Wandelt, I. K. Wehus, M. White, S. D. M. White, A. Zacchei, and A. Zonca. Planck 2018 results: Vi. cosmological parameters. *Astronomy amp; Astrophysics*, 641:A6, Sept. 2020. ISSN 1432-0746. doi: 10.1051/0004-6361/201833910. URL http://dx.doi.org/10.1051/0004-6361/201833910.

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019. URL https://arxiv.org/abs/1907.10902.

S. Appleby, P. Chingangbam, C. Park, K. P. Yogendran, and P. K. Joby. Minkowski tensors in three dimensions: Probing the anisotropy generated by redshift space distortion. *The Astrophysical Journal*, 863(2):200, Aug. 2018. ISSN 1538-4357. doi: 10.3847/1538-4357/aacf8c. URL http://dx.doi.org/10.3847/1538-4357/aacf8c.

W. A. S. Barbosa and D. J. Gauthier. Learning spatiotemporal chaos using next-generation reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(9), Sept. 2022. ISSN 1089-7682. doi: 10.1063/5.0098707. URL http://dx.doi.org/10.1063/5.0098707.

S. Baur and C. Räth. Predicting high-dimensional heterogeneous time series employing generalized local states. *Physical Review Research*, 3(2), June 2021. ISSN 2643-1564. doi: 10.1103/physrevresearch.3.023215. URL http://dx.doi.org/10.1103/PhysRevResearch.3.023215.

A. Ben-David, S. v. Hausegger, and A. D. Jackson. Skewness and kurtosis as indicators of non-gaussianity in galactic foreground maps. *Journal of Cosmology and Astroparticle Physics*, 2015 (11):019–019, Nov. 2015. ISSN 1475-7516. doi: 10.1088/1475-7516/2015/11/019. URL http://dx.doi.org/10.1088/1475-7516/2015/11/019.

A. Blanchard, J.-Y. Héloret, S. Ilić, B. Lamine, and I. Tutusaus. cdm is alive and well. *The Open Journal of Astrophysics*, 7, May 2024. ISSN 2565-6120. doi: 10.33232/001c.117170. URL http://dx.doi.org/10.33232/001c.117170.

B. Bolliet, B. Comis, E. Komatsu, and J. F. Macías-Pérez. Dark energy constraints from the thermal sunyaev–zeldovich power spectrum. *Monthly Notices of the Royal Astronomical Society*, 477(4): 4957–4967, Mar. 2018. ISSN 1365-2966. doi: 10.1093/mnras/sty823. URL http://dx.doi.org/10.1093/mnras/sty823.

J. R. Bond, L. Kofman, and D. Pogosyan. How filaments of galaxies are woven into the cosmic web. *Nature*, 380(6575):603–606, Apr. 1996. ISSN 1476-4687. doi: 10.1038/380603a0. URL http://dx.doi.org/10.1038/380603a0.

S. L. Bridle, A. M. Lewis, J. Weller, and G. Efstathiou. Reconstructing the primordial power spectrum. *Monthly Notices of the Royal Astronomical Society*, 342(4):L72–L78, July 2003. ISSN 1365-2966. doi: 10.1046/j.1365-8711.2003.06807.x. URL http://dx.doi.org/10.1046/j.1365-8711.2003.06807.x.

B. W. Carroll and D. A. Ostlie. *An Introduction to Modern Astrophysics*. Addison-Wesley, San Francisco, 2nd edition, 1996.

A. Chattopadhyay, P. Hassanzadeh, and D. Subramanian. Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, 27(3):373–389, July 2020. ISSN 1607-7946. doi: 10.5194/npg-27-373-2020. URL http://dx.doi.org/10.5194/npg-27-373-2020.

P. Coles and L.-Y. Chiang. Characterizing the nonlinear growth of large-scale structure in the universe. *Nature*, 406(6794):376–378, July 2000. ISSN 1476-4687. doi: 10.1038/35019009. URL http://dx.doi.org/10.1038/35019009.

C. Collischon. ccollischon/litchi: Initial release. https://doi.org/10.5281/zenodo.11940174, 2024. Zenodo.

C. Collischon, M. Sasaki, K. Mecke, S. D. Points, and M. A. Klatt. Tracking down the origin of superbubbles and supergiant shells in the magellanic clouds with minkowski tensor analysis. *Astronomy amp; Astrophysics*, 653:A16, Aug. 2021. ISSN 1432-0746. doi: 10.1051/0004-6361/202040153. URL http://dx.doi.org/10.1051/0004-6361/202040153.

C. Collischon, M. A. Klatt, A. J. Banday, M. Sasaki, and C. Räth. Morphometry on the sphere: Cartesian and irreducible minkowski tensors explained and implemented. *Communications Physics*, 7(1), July 2024. ISSN 2399-3650. doi: 10.1038/s42005-024-01751-1. URL http://dx.doi.org/10.1038/s42005-024-01751-1.

K. Dolag, R.-S. Remus, L. M. Valenzuela, L. C. Kimmig, B. Seidel, S. Fortune, J. Stoiber, A. Ivleva, T. Hoffmann, V. Biffi, I. Marini, P. Popesso, and S. Vladutescu-Zopp. Encyclopedia magneticum: Scaling relations from cosmic dawn to present day, 2025. URL https://arxiv.org/abs/2504.01061.

J. D. Faires and R. L. Burden. *Numerische Methoden: Näherungsverfahren und ihre praktische Anwendung*. Spektrum Akademischer Verlag, 1995. ISBN 3-86025-332-8.

D. Gauthier, E. Bollt, A. Griffith, and W. Barbosa. Next generation reservoir computing. *Nature Communications*, 12(1):5564, Sept. 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-25801-2. URL https://rdcu.be/eBAJK.

K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. *HEALPix: Framework for high-resolution discretization and fast analysis of data distributed on the sphere*, 2024. URL https://healpix.sourceforge.io/pdf/intro.pdf. Introduction to HEALPix.

A. Haluszczynski, D. Köglmayr, and C. Räth. Controlling dynamical systems to complex target states using machine learning: next-generation vs. classical reservoir computing. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2023.

B. Hasselblatt and A. Katok. *A First Course in Dynamics: with a Panorama of Recent Developments*. Cambridge University Press, 2003.

J. Herteux and C. Räth. Breaking symmetries of the reservoir equations in echo state networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(12), Dec. 2020. ISSN 1089-7682. doi: 10.1063/5.0028993. URL http://dx.doi.org/10.1063/5.0028993.

H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. *GMD-Report 148, German National Research Institute for Computer Science*, 01 2001.

D. Köglmayr and C. Räth. Extrapolating tipping points and simulating non-stationary dynamics of complex systems using efficient machine learning. *Scientific Reports*, 14(1):507, 2024.

E. Komatsu, K. M. Smith, J. Dunkley, C. L. Bennett, B. Gold, G. Hinshaw, N. Jarosik, D. Larson, M. R. Nolta, L. Page, D. N. Spergel, M. Halpern, R. S. Hill, A. Kogut, M. Limon, S. S. Meyer, N. Odegard, G. S. Tucker, J. L. Weiland, E. Wollack, and E. L. Wright. Seven-yearwilkinson microwave anisotropy probe(wmap) observations: Cosmological interpretation. *The Astrophysical Journal Supplement Series*, 192(2):18, Jan. 2011. ISSN 1538-4365. doi: 10.1088/0067-0049/192/2/18. URL http://dx.doi.org/10.1088/0067-0049/192/2/18.

D. Lairez. What entropy really is : the contribution of information theory, 2022. URL https://arxiv.org/abs/2204.05747.

G. Li, Z. Lu, J. Wang, and Z. Wang. Machine learning in stellar astronomy: Progress up to 2024, 2025. URL https://arxiv.org/abs/2502.15300.

A. Linde. *Inflationary Cosmology*, page 1–54. Springer Berlin Heidelberg. ISBN 9783540743521. doi: 10.1007/978-3-540-74353-8_1. URL http://dx.doi.org/10.1007/978-3-540-74353-8_1.

Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos*, 27(4):041102, Apr. 2017. ISSN 1448-6083. doi: 10.1063/1.4979665. URL https://pubmed.ncbi.nlm.nih.gov/28456169/.

D. Meyer. Openai reportedly wants to build 5-gigawatt data centers, and nobody knows who could supply that much power. https://fortune.com/2024/09/27/openai-5gw-data-centers-altman-power-requirements-nuclear/, Sept. 2024. Accessed: 2026-02-27.

T. Padmanabhan. *Structure Formation in the Universe*. 1993.

J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.*, 120:024102, Jan 2018. doi: 10.1103/PhysRevLett.120.024102. URL https://link.aps.org/doi/10.1103/PhysRevLett.120.024102.

M. Perlin and M. D. Bustamante. A robust quantitative comparison criterion of two signals based on the sobolev norm of their difference. *Journal of Engineering Mathematics*, 101(1):115–124, Mar. 2016. ISSN 1573-2703. doi: 10.1007/s10665-016-9849-7. URL http://dx.doi.org/10.1007/s10665-016-9849-7.

N. Perraudin, A. Srivastava, A. Lucchi, T. Kacprzak, T. Hofmann, and A. Réfrégier. Cosmological n-body simulations: a challenge for scalable generative models, 2019. URL https://arxiv.org/abs/1908.05519.

C. Rabbath and D. Corriveau. A comparison of piecewise cubic hermite interpolating polynomials, cubic splines and piecewise linear functions for the approximation of projectile aerodynamics. *Defence Technology*, 15(5):741–757, 2019. ISSN 2214-9147. doi: https://doi.org/10.1016/j.dt.2019.07.016. URL https://www.sciencedirect.com/science/article/pii/S2214914719301187. SI: 2019 International Symp. Ballistics.

J. Y. H. Soo, I. Y. K. A. Shuaili, and I. M. Pathi. Machine learning applications in astrophysics: Photometric redshift estimation. In *FIRST INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE amp; DATA ANALYTICS: Incorporating the 1st South-East Asia Workshop on Computational Physics and Data Analytics (CPDAS 2021)*, volume 2756, page 040001. AIP Publishing, 2023. doi: 10.1063/5.0140152. URL http://dx.doi.org/10.1063/5.0140152.

U. P. Steinwandel, K. Dolag, L. Böss, and T. Marin-Gilabert. Towards cosmological simulations of the magnetized intracluster medium with resolved coulomb collision scale, 2023. URL https://arxiv.org/abs/2306.04692.

F. Takens. Detecting strange attractors in turbulence. In D. Rand and L.-S. Young, editors, *Lecture Notes in Mathematics, Berlin Springer Verlag*, volume 898, page 366. 1981. doi: 10.1007/BFb0091924.

R. Teyssier, S. Pires, S. Prunet, D. Aubert, C. Pichon, A. Amara, K. Benabed, S. Colombi, A. Refregier, and J.-L. Starck. Full-sky weak-lensing simulation with 70 billion particles. *Astronomy amp; Astrophysics*, 497(2):335–341, Feb. 2009. ISSN 1432-0746. doi: 10.1051/0004-6361/200810657. URL http://dx.doi.org/10.1051/0004-6361/200810657.

M. S. Turner. The road to precision cosmology. *Annual Review of Nuclear and Particle Science*, 72(1):1–35, Sept. 2022. ISSN 1545-4134. doi: 10.1146/annurev-nucl-111119-041046. URL http://dx.doi.org/10.1146/annurev-nucl-111119-041046.

W. N. van Wieringen. Lecture notes on ridge regression.

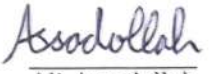
M. Wedler, M. Stender, M. Klein, S. Ehlers, and N. Hoffmann. Surface similarity parameter: A new machine learning loss metric for oscillatory spatio-temporal data. *Neural Networks*, 156:123–134, Dec. 2022. ISSN 0893-6080. doi: 10.1016/j.neunet.2022.09.023. URL http://dx.doi.org/10.1016/j.neunet.2022.09.023.

A. Youssef, B. Murmann, and H. Omran. Analog ic design using precomputed lookup tables: Challenges and solutions. *IEEE Access*, PP:1–1, 07 2020. doi: 10.1109/ACCESS.2020.3010875.

Declaration:

I hereby declare that this thesis is my own work, and that I have not used any sources and aids other than those stated in the thesis.

Munich, 3 March 2026



Ali Assadollah