Sorbonne University

LISN – Laboratoire Interdisciplinaire des Sciences du Numérique, Université Paris-Saclay

Understanding Turbulence via Machine Learning

Master Thesis

Author: Valerio Actis Dato Casale

Supervisors: Sergio Chibbaro, Cyril Furtlehner, Alessandro Pelizzola

Laboratory: LISN, Université Paris-Saclay

Academic Year: 2024/2025

Introduction

Turbulence remains one of the most challenging and fundamental open problems in classical physics, governing processes from atmospheric circulation to industrial mixing, aerodynamics and active matter. Despite the compact look of the governing equations of turbulent flows, the Navier–Stokes equations, their nonlinear and multiscale nature gives rise to complex behaviors such as intermittency, non-Gaussian fluctuations, and anomalous scaling, making analytical treatment extremely difficult. A central feature of turbulent flows is the energy cascade: energy is transferred through swirls and vortex stretching from large to small scales , where it is eventually dissipated by viscosity. Understanding the underlying mechanisms at play in this phenomenon is one of the key steps in unraveling the mystery of Turbulence.

To make analytical and numerical progress, simplified models that retain the essential physical ingredients of turbulence are often employed. Among these, shell models have a particularly important role. These are systems of coupled ordinary differential equations designed to mimic the energy cascade across discretized scales (or shells) in Fourier space. Despite their simplicity, shell models successfully reproduce many statistical features of fully developed turbulence, including intermittency and multiscaling, while remaining computationally efficient and analytically more tractable.

In the last decade, machine learning (ML) has greatly renewed the research perspectives in many fields. ML's ability to extract patterns from complex data and identify structure in chaotic systems has opened new paths to understand physical phenomena, including turbulence modeling and reduced-order model discovery. The goal of this thesis is to test the capability of machine learning to identify and interpret the interaction structure underlying the energy cascade in turbulence. We employ sparse nonlinear regression techniques to extract relevant interactions from data generated by a shell model. This is just a first, elementary step towards the implementation of ML in the study of turbulent flows. A possible future development might make use of Restricted Boltzmann machines, inspired by the statistical nature of turbulence theory.

Phenomenological overview

Turbulent flow is supposedly completely described by the Navier-Stokes equation:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{1}$$

with the incompressibility condition

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

For a given geometrical shape of the boundaries The Reynolds number is the only control parameter of the flow [4]:

$$Re = \frac{VL}{\nu}, \tag{3}$$

where V is a characteristic velocity, L a characteristic length scale, and ν the kinematic viscosity. They are too complex to solve directly at high Reynolds numbers, where turbulent phenomena occur. Instead, the velocity field $\mathbf{u}(\mathbf{x}, t)$ is treated as a random variable, and statistical tools are employed to characterize its behavior.

Energy cascade. A central concept in fully developed turbulence is that of **energy cascade** [11]. Energy is injected at large scales through a force term creating disturbances ("eddies"), transferred across intermediate scales in the so called **inertial range** up to a viscous cutoff, the **Kolmogorov length** l_D, and finally dissipated into heat at small

scales by molecular friction. As we will see, this is correctly reproduced by Shell Models. it's important to note that the external forcing term is essential for the system to reach a stationary state as the injected energy is quickly dissipated in heat, and without a continuous source, the turbulent system would soon go back to a rest state.

Experimental results. From experimental data we can infer two laws valid, at least approximately, for almost any turbulent flow [4]:

1. **Two-thirds law.** In a turbulent flow at very high Reynolds number, the mean square velocity increment $\langle \delta u(l)^2 \rangle = \langle (\boldsymbol{u}(\boldsymbol{r}+\boldsymbol{l}) - \boldsymbol{u}(\boldsymbol{r}))^2 \rangle \sim l^{2/3}$ behaves approximately as the two-thirds power of the distance

2. **Law of finite energy dissipation** in a turbulent flow , keeping all experimental control parameters constant except for viscosity, which is lowered as much as possible, the energy dissipation per unit mass $\frac{dE}{dt} = \epsilon$ behaves in a way consistent with a finite positive limit.

Other statistical experimental findings concern the skewness and kurtosis of the velocity gradients [10]:

$$S = \frac{\langle (\delta u)^3 \rangle}{\langle (\delta u)^2 \rangle^{3/2}} \to \begin{cases} 0 & l \sim L \\ 0.4 & l \to 0 \end{cases}, \qquad K = \frac{\langle (\delta u)^4 \rangle}{\langle (\delta u)^2 \rangle^2} \to \begin{cases} 3 & l \sim L \\ 4 & l \to 0 \end{cases} \tag{4}$$

For large l these quantities approach values typical of a Gaussian distribution. However, as $l \to 0$, the kurtosis increases to $K \approx 4$, and the skewness becomes nonzero, signaling a deviation from Gaussian behavior and suggesting **intermittency** in the velocity gradients and an asymmetry in their probability distribution function.

The observed non-zero skewness linked to *vortex stretching*, a process where vortices become thinner in the direction perpendicular to the stretching due to the conservation of fluid volume. This leads to a reduction in the radial size of the vortices, causing larger flow structures to break apart into smaller ones. This cascade continues until the structures are sufficiently small for their kinetic energy to be dissipated as heat. Conversely, the high kurtosis indicates that the tails of the distribution are fatter than those of a Gaussian, suggesting that extreme values of velocity gradients are more likely to occur and thus that energy dissipation is highly *non-uniform*, with intense, localized bursts.

*Kolmogorov Theory.*At present days there exists no full theory of turbulence directly derived from the Navier-Stokes equations. However in 1941 Kolmogorov [5] derived one of the few exact results descending directly from (1) for $Re \to \infty$, the **four-fifth law**:

$$\langle [\delta u(l)]^3 \rangle = -\frac{4}{5} \varepsilon \, l, \tag{5}$$

under the assumptions of isotropy, homogeneity and finiteness of dissipation rate ϵ (**H1**).

In the same year Kolmogorov proposed a phenomenological theory (K41) based on **H1** and an assumption of **global scaling invariance** (**H2**) such that $l \to Al$:

$$x \to \Lambda \, x, \quad v \to \Lambda^h \, v, \quad t \to \Lambda^{1-h} \, t, \tag{6}$$

with A an arbitrary scaling factor and h the scaling exponent. From this and (5) we have:

$$\Lambda^{3h} \langle [\delta u(l)]^3 \rangle = -\Lambda \frac{4}{5} \varepsilon \, l \quad \to h = 1/3 \tag{7}$$

and thus, for higher order moments, in jargon **structure functions**:

$$S_p(l) = \langle |\delta v(l)|^p \rangle \sim l^{\zeta_p} \quad \zeta_p = p/3 \tag{8}$$

Finally, the famous inertial-range energy spectrum can be derived:

$$E(k) \sim C_K \, \varepsilon^{2/3} \, k^{-5/3}, \tag{9}$$

with C_K being the Kolmogorov constant. The invariance of the energy transfer rate under the scaling (7) is confirmed by its transformation law:

$$\varepsilon \to A^{3h-1} \, \varepsilon, \tag{10}$$

so that with $h = \frac{1}{3}$, ε remains invariant.

Unfortunately, experimental data and numerical simulations reveal that the exponents ζ_p deviate from the linear prediction of K41 for $p > 3$. This inadequacy is strictly related to the aforementioned phenomenon of intermittency, detectable, indeed, through higher order moments. The problem directly stems from assumption H2, as global scaling invariance implies uniform dissipation, a condition hardly encountered in real turbulent systems.

*Multifractality.*To describe the intermittent nature of turbulence the hypothesis that energy dissipation occurs uniformly on a *fractal* structure with fractal dimension $D_F < 3$ was firstly introduced, yet with unsatisfactory results. Successively, a **multifractal** model was proposed [8, 9], where, at odds with the K41 theory which assumes a single *global* scaling exponent h, only a *local* scaling invariance is assumed (**H3**): different regions of the turbulent flow experience different rates of energy transfer, leading to a continuous spectrum of scaling exponents instead of a single value. More formally $h < 1$ (for $h \geq 1$ the velocity gradients are not singular) is assumed on a fractal set F:

$$F = \bigcup_h \Omega(h)$$
$$\Omega(h) = \{x \mid \delta u(x, l) \sim l^z, \; z \in [h, h + dh]\}. \tag{11}$$

each subset $\Omega(h)$ is itself a fractal of dimension $D(h)$. being $P_l(h)$ the probability of having scaling exponent h proportional to the volume of $\Omega(h)$ over the total volume we have:

$$\langle |\delta v(l)|^p \rangle \sim \int l^{hp+3-D(h)} dh \tag{12}$$

and using a saddle point approximation

$$\zeta_p = \min_h \left[ph + 3 - D(h) \right]. \tag{13}$$

K41 is recovered for $P_l(h) = \delta(h - 1/3), D(h = 1/3) = 3$. The computation of $D(h)$ from the Navier-Stokes is the next big open question.

Shell models

Shell models are simplified dynamical systems designed to capture the essential features of turbulent energy cascades while remaining computationally tractable [3]. They replace the full Navier–Stokes equations with a system of coupled nonlinear ordinary differential equations that describe the evolution of velocity fluctuations across scales.Shell models follow the dynamics of complex-valued, scalar variables $u_n(t)$ (geometry and space are completely lost!), each associated with a discrete shell in Fourier space, corresponding to a wavenumber $k_n = k_0 \lambda^n$, where λ is the intershell ratio, typically set to 2. This implies

that the shells are logarithmically spaced, covering a wide range of scales with relatively few degrees of freedom.

The motivation behind shell models is clear: we need a simple, dimensionally consistent system that captures the multiscale, nonlinear, and chaotic nature of turbulence while being more analytically and computationally manageable than the full Navier–Stokes equations. Shell models aim to describe a deterministic dynamical evolution of a set of variables defined across a wide range of spatial scales, each associated with different characteristic times. In other words, the goal is to construct a model that embodies the phenomenological picture of the Richardson energy cascade, where energy flows from large to small scales, while retaining a well-defined deterministic time evolution.

The evolution equation for each shell variable typically takes the form:

$$\frac{du_n}{dt} = \nu k_n^2 u_n + \mathcal{G}_n[u,u] + f_n, \tag{14}$$

where $\nu k_n^2 u_n$ represents viscous dissipation, $\mathcal{G}_n[u,u]$ is a quadratic nonlinear coupling term that mediates energy transfer between shells, and f_n is an external forcing term.

The nonlinear term $\mathcal{G}_n[u,u]$ is constructed to preserve key physical invariants of the original Navier–Stokes dynamics—such as total energy and helicity—while restricting interactions to neighboring shells (nearest and next-nearest neighbors). This locality in Fourier space reflects the idea that most energy transfers in turbulent flows occur between adjacent scales. In the following we will focus on the so called Sabra model:

$$\frac{du_n}{dt} = \underbrace{i(k_{n+1}Au_{n+1}^*u_{n+2} + k_nBu_{n-1}^*u_{n+1} - k_{n-1}Cu_{n-2}u_{n-1})}_{\text{nonlinear}} - \underbrace{\nu k_n^2 u_n}_{\text{dissipation}} + \underbrace{f_n}_{\text{forcing}}$$

Such model exhibits a form of intermittency that is both qualitatively and quantitatively similar to that observed in real turbulent flows governed by the Navier–Stokes equations. In particular, it was showed that the anomalous scaling of velocity increments in turbulence is well reproduced [1]. This agreement supports the idea that shell models not only reproduce the correct energy cascade but also capture the intermittent nature of turbulence. The shell model thus provides a valuable tool to investigate the statistical features of turbulence without the full complexity of fluid dynamics in physical space.

setting A=1, the model contains two free parameters, B and C, which control the relative weights of the nonlinear interactions. In the inviscid ($\nu = 0$) and unforced ($f_n = 0$) limit these parameters are constrained by the requirement of conservation of two quadratic invariants:

$$E = \sum_n |u_n|^2, \quad \text{(total energy)} \tag{15}$$

$$Q_2 = \sum_n (-1)^n k_n^{-\log_\lambda |1-B|} |u_n|^2 \tag{16}$$

The energy conservation fixes: $C = -(1+B)$. A particularly popular choice consists in requiring the second invariant to have the same physical dimensions as a quantity called helicity in the Navier–Stokes equations. This is achieved by setting $-\log_\lambda |1-B| = 1$, which, for the commonly used intershell ratio $\lambda = 2$, leads to $B = -1/2$.

Generating the Data

The initial step in our work was developing a flexible and extensible code capable of simulating a large number of complex-valued datasets from equation (15) under various parameters configurations. We based our implementation on existing reference codes, which we significantly modified to suit our objectives. This included generalizing the overall structure and expanding the range of tunable parameters. The resulting Fortran code, based on standard 4th order Runge-Kutta integration, embeds the external application of forcing to the system, and computes the energy spectra, structure functions up to any order (in principle), and the energy flux across individual shells. The number of shells used in the simulations will be indicated as nn, and it has to be finite. This means that the boundary shells must implement slightly different equations:

$$N(u_1) = ik_2 A u_2^* u_3 \qquad N(u_2) = i(k_3 A u_3^* u_4 + k_2 B u_1^* u_3 - k_1 C u_0 u_1)$$
$$N(u_{nn-1}) = i(k_{nn} B u_{nn-2}^* u_{nn} - k_{nn-1} C u_{nn-3} u_{nn-2}) \qquad N(u_{nn}) = -ik_{nn-1} C u_{nn-2} u_{nn-1}$$

For the forcing, we adopted this scheme: we calculate the total energy in the forcing range (defined as the range from shell n_{dwn} up to shell n_{upp} to which we apply the forcing) and determine a scaling factor α:

$$E_f = \sum_{n=1}^{nn} |u_n|^2 \cdot \Theta(n - n_{dwn}) \cdot \Theta(n_{upp} - n) = \sum_{n=dwn}^{upp} |u_n|^2 \qquad \alpha = \frac{\epsilon}{E_f}$$

and the forcing term is:

$$f_n = u_n(\alpha + i\phi_n) \cdot \Theta(n - n_{dwn}) \cdot \Theta(n_{upp} - n)$$

where: $\Theta(x)$ is the Heaviside step function, ϕ_n is a random phase.In the following we will always use $n_{dwn} = 1$,$n_{upp} = 3$. For the initial state of the system the variables u_n are randomly initialized. We used the parameters $A = 1$, $B = \frac{A}{r} - A$, and $C = -\frac{A}{r}$, r, the intershell ratio, can be arbitrarily chosen, we experimented with $r = 2$ and the golden ratio $r = \frac{1+\sqrt{5}}{2}$. Other key parameters include `step` (the total number of iterations), `sstep` (the interval between two samples of the spectra) and `fstep` (the interval between two samples of the fluxes). The timestep is defined as $dt = \beta\sqrt{\frac{\nu}{\epsilon}}$, where β, ν and ϵ are all tunable parameters. This choice ensures that dt remains smaller than the smallest timescale of the system, crucial for numerical stability and accurate temporal resolution. To validate the code and ensure it reproduces the expected dynamical behavior, we tested for the attainment of a statistically steady state. This was done by monitoring the running average of the total energy and of quantity eq. (16) and checking for convergence over time. An example of this process is shown in fig. 1, where also the time evolution of the quantity $\epsilon(t) = \sum_n k_n^2 |u_n|^2$ (enstrophy) is displayed, to clearly visualize the strong intermittent behavior. Once equilibrium was established, we used the subsequent time series data to compute various statistical observables, including the second-order structure function and the average energy dissipation per shell. These results are presented in fig. 19. As expected, the second-order structure function exhibits a scaling exponent close to 2/3, in agreement with the **Two-Thirds law**. Furthermore, the average energy flux remains nearly constant across the inertial range, confirming the consistency of the simulation with theoretical predictions. In figs. 14 and 15 we show similar plots for different nn and r values

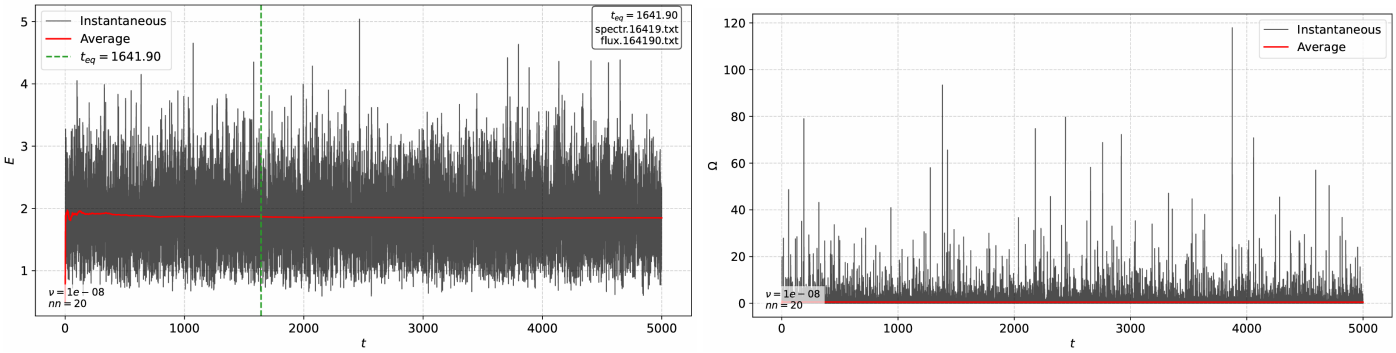

Figure 1: $r = 2$,$nn = 20$, $\nu = 10^{-8}$, $\epsilon = 1$. Energy (left) and $\epsilon(t)$ (right) timeseries. the red curves represent the average values and the dashed green line the timestep after which the system is considered at equilibrium.

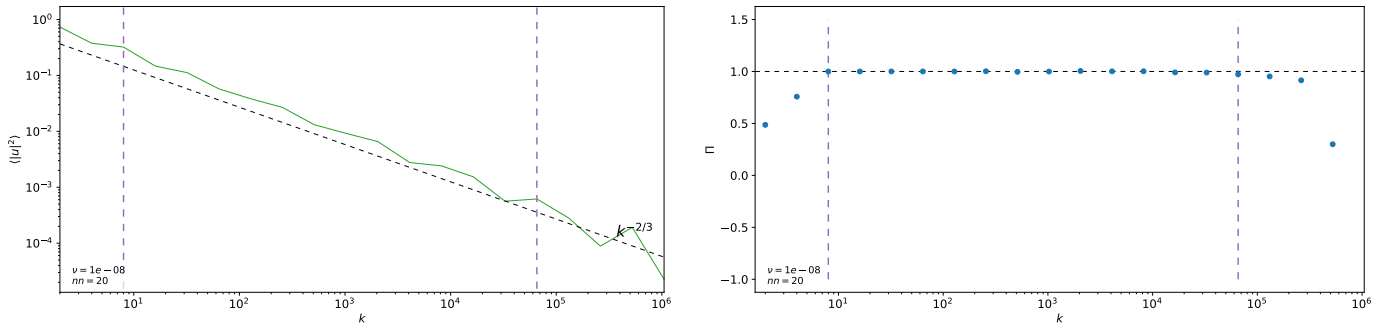

Figure 2: $r = 2$,$nn = 20$, $\nu = 10^{-8}$, and $\epsilon = 1$. **Left:** Second-order structure function (green line) compared to the theoretical scaling $k^{-2/3}$ (black dashed line), log-log scale. **Right:** Average energy flux per shell. The flux remains approximately constant in the inertial range, which lies between the two dashed purple lines. The forcing range is evident in the first three shells, while the dissipation range appears in the last three.

Learning from Data

The central goal of this part of the project is to explore whether the nonlinear structure of turbulence, as modeled by the SABRA shell model, can be recovered directly from the generated data using machine learning techniques.Several approaches are possible and we decided to start with a sparse Nonlinear Regression [2]. In particular, we aim to reconstruct the dynamical equations governing the evolution of each shell variable u_n by assuming that its time derivative $\dot{u}_n$ can be expressed as a sparse combination of nonlinear interactions. For this purpose, We extensively utilized Lasso regression.

Lasso Regression and Sparsity Promotion

Lasso (least absolute shrinkage and selection operator) regression is a popular technique for sparse modeling, where the goal is to balance prediction accuracy with model simplicity by penalizing the ℓ_1 norm of the coefficient vector [6]. Originally formulated for linear regression, the Lasso estimator solves the optimization problem:

$$\min_{W \in \mathbb{R}^p} \left\{ \frac{1}{N} \left\| y - W^T X \right\|_2^2 + \lambda \|W\|_1 \right\} \tag{17}$$

Here, $X \in \mathbb{R}^{p \times N}$ is the matrix containing the input terms evaluated at N time points, $y \in \mathbb{R}^N$ is the vector of time derivatives $\dot{u}_n$, W is the vector of coefficients ξ_j, and $\lambda > 0$

controls the strength of the sparsity penalty.

The solution $\hat{W}$ can be interpreted as a soft-thresholded version of the ordinary least squares (OLS) estimate:

$$\hat{W}_j = \mathrm{S}_{N,\lambda}\left(\hat{W}_j^{\mathrm{OLS}}\right) = \hat{W}_j^{\mathrm{OLS}} \cdot \max\left(0, 1 - \frac{N\lambda}{|\hat{W}_j^{\mathrm{OLS}}|}\right) \tag{18}$$

$$\hat{W}_j^{\mathrm{OLS}} = (X^\top X)^{-1} X^\top y \quad (\text{or simply } \hat{W}_j^{\mathrm{OLS}} = X^\top y \text{ if } X^\top X = I) \tag{19}$$

Here, $\mathrm{S}_{N,\lambda}$ denotes the *soft thresholding operator*, which sets small coefficients to zero and shrinks larger ones toward zero, thus promoting sparsity.

While Lasso uses an ℓ_1 penalty, Ridge regression instead penalizes the ℓ_2 norm of the coefficient vector:

$$\min_W \left\{ \frac{1}{N} \left\| y - W^T X \right\|_2^2 + \lambda \|W\|_2^2 \right\} \tag{20}$$

This results in small but non-zero coefficients for all features, promoting smoothness rather than sparsity. Unlike Lasso, Ridge cannot perform variable selection [7] (i.e., it does not drive coefficients to zero).

It is important to note that Lasso, while effective for identifying relevant interactions, can distort the true values of the coefficients. This is due to the ℓ_1 penalty shrinking all coefficients—including large, physically meaningful ones—toward zero. As a result, the learned coefficients may not faithfully reproduce the original magnitudes. In practice, a common workaround is to use Lasso for feature selection and then refit the selected model using unregularized (OLS) regression to recover unbiased coefficient estimates.

For SABRA model

We have access to time series of $\{u_n(t)\}$ and we can compute numerically their time derivatives, thus we represent the system state as a vector $\mathbf{U(t)} \in \mathbb{C}^{nn}$ (with nn number of shells) where each component is a shell velocity at time t:

$$\mathbf{U(t)} = [u_1(t), u_2(t), ..., u_{nn}(t)]^T \tag{21}$$

The SABRA model can be written in matrix form:

$$\frac{d\mathbf{U(t)}}{dt} = \mathbf{F}(\mathbf{U(t)}) = N(\mathbf{U(t)}) - D\mathbf{U(t)} + \mathbf{f(t)} \tag{22}$$

where: $N(\mathbf{U(t)})$ is the nonlinear term, D is the diagonal dissipation matrix with $D_{nn} = \nu k_n^2$, and $\mathbf{f(t)}$ is the forcing vector.To reproduce the Lasso regression structure, we construct a dictionary of functions $\Phi(\mathbf{U(t)})$, which consists of pre-computed features derived from the system state $\mathbf{U(t)}$. These features are assumed to capture the relevant interactions necessary to model the time evolution $\frac{d\mathbf{U(t)}}{dt}$. Adopting a Machine Learning perspective, we deliberately ignore any prior knowledge about the underlying equations or the model used to generate the data. Instead, we hypothesize a general class of functions that could potentially describe the dynamics, add them to the dictionary, and let the Lasso algorithm select the most informative ones based solely on the data. We consider two distinct scenarios for learning the dynamics:

Case 1: Subtracting Dissipation (SubDiss)

In this case, we assume that the dissipation matrix D is known[1]. We subtract both the dissipation and forcing terms from the target data[2], allowing us to focus exclusively on identifying the structure of the nonlinear interactions. The nonlinear terms for the i-th shell is then approximated as:

$$N(U_i(t)) \approx \sum_j W_{ij}\Phi_j(\mathbf{U(t)}) \tag{23}$$

Here, we assume that only **pairwise interactions** between shells are relevant. However, due to the complex-valued nature of the system, we must consider not only products of the form $u_i(t)u_j(t)$, but also combinations involving complex conjugates. Therefore, the dictionary vector includes the following types of interactions for all pairs of shells i, j:

$$\Phi_{ij}^{(1)}(t) = u_i(t)u_j(t), \quad \Phi_{ij}^{(2)}(t) = u_i^*(t)u_j(t), \quad \Phi_{ij}^{(3)}(t) = u_i(t)u_j^*(t), \quad \Phi_{ij}^{(4)}(t) = u_i^*(t)u_j^*(t) \tag{24}$$

Each of these terms is a candidate nonlinear interaction in the regression problem. The full dictionary $\mathbf{(\Phi(U(t)))}$ therefore consists of all such terms evaluated at each time t. For $nn = 20$, we get $\Phi(\mathbf{U(t)}) \in \mathbb{R}^{nn^2=1600}$ (this notation of $\Phi(\mathbf{U(t)})$ might appear confusing: $\Phi(\mathbf{U(t)})$ remains a vector for the single shell, we use double indexing only to identify the type of interaction).

The Lasso optimization problem becomes:

$$\min_W \left\{ \frac{1}{N_t} \left\| \frac{d\mathbf{U}_i}{dt} + D\mathbf{U}_i - \mathbf{f}_i - \mathbf{W}^T\Phi(U) \right\|_2^2 + \lambda\|\mathbf{W}\|_1 \right\} \tag{25}$$

where $\frac{d\mathbf{U}_i}{dt}$ is the vector whose components are values of the time derivatives of shell i at all the considered timesteps, $\mathbf{U}_i$ and $\mathbf{f}_i$ are the vectors containing the timeseries of shell $u_i(t)$ and forcing $f_i(t)$ and U is the matrix obtained stacking the vectors $\mathbf{U}_i$ for every shell i (note how $\Phi(U)$ is a matrix made up of vectors $\Phi(\mathbf{U(t}_1\mathbf{)})$, $\Phi(\mathbf{U(t}_2\mathbf{)})$,... stacked together) . In this formulation, the Lasso regression is responsible only for identifying the sparse set of nonlinear coefficients W_j, as the dissipation and forcing have already been removed from the target.

Case 2: Not Subtracting Dissipation (NoSubDiss)

In this case, we still subtract the forcing term $\mathbf{f}$, but we do not remove the dissipation one. Instead, we include the linear terms $\mathbf{U(t)}$ as additional features in our dictionary. This allows the regression to learn both nonlinear and linear (dissipative) terms directly from data.

The dictionary is expanded to include $\mathbf{U(t)}$ itself:

$$\Phi'(\mathbf{U(t)}) = [\Phi(\mathbf{U(t)}), \mathbf{U(t)}] \tag{26}$$

with, for $nn = 20$, $\Phi(\mathbf{U(t)}) \in \mathbb{R}^{nn^2+nn=1620}$ The model is now:

$$\frac{du_i(t)}{dt} - f_i(t) \approx \sum_j W_{ij}\Phi'_j(\mathbf{U(t)}) \tag{27}$$

[1]This assumption is not arbitrary: the quadratic dependence on k_n in the coefficients of the dissipation terms naturally arises from the Navier-Stokes equations (1) and can therefore be inferred.

[2]This procedure is also justifiable: since the forcing is external, it is not unreasonable to assume that it can be independently measured or controlled in an experimental setup.

and the optimization problem becomes:

$$\min_{W} \left\{ \frac{1}{N_t} \left\| \frac{d\mathbf{U}_i}{dt} - \mathbf{f}_i - \mathbf{W}^T \Phi'(U) \right\|_2^2 + \lambda \|\mathbf{W}\|_1 \right\} \tag{28}$$

In this formulation, the Lasso not only identifies the dominant nonlinear interactions but also recovers the dissipation structure by assigning appropriate weights to the linear terms in Φ.

Applying Lasso regression we expect to recover a good approximation of the structure of the interactions of the original model (15). in case 1 only 3 types of interactions are expected: *backward-forward* with coefficient iBk_n, *forward-forward* with coefficient iAk_{n+1}, and *backward-backward* with coefficient iCk_{n-1}. In case 2 we expect to recover the same interactions plus an additional coefficient νk_n^2 associated to the dissipation term. We would like to stress the fact that so far only *uncoupled* learning has been presented, i.e. each shell is assumed to have access to all interactions and the model selects *independently*, shell by shell, which interactions are relevant for the different shells.

LassoCV and Model Selection in Practice

In practice, we employed the `LassoCV` implementation from the `scikit-learn` Python library to perform automatic selection of the regularization parameter λ through cross-validation. The standard `Lasso` estimator solves the objective

$$\min_{W \in \mathbb{R}^p} \left\{ \frac{1}{2N} \left\| y - X^T W \right\|_2^2 + \lambda \|W\|_1 \right\}, \tag{29}$$

where λ controls the trade-off between data fidelity and model sparsity. Selecting an appropriate value of λ is crucial for obtaining a meaningful sparse model.

To automate this, `LassoCV` performs k-fold cross-validation: the dataset is split into k equally sized subsets (folds), and the model is trained on $k-1$ of them while evaluated on the remaining fold. This process is repeated k times, each time using a different fold as the validation set. The mean validation error across folds is computed for each candidate λ, and the value minimizing this error is selected.

This procedure helps prevent overfitting and underfitting by assessing generalization performance on unseen data. It is especially beneficial in our setting where the dictionary $\Phi(\mathbf{U(t)})$ may contain many irrelevant or redundant features.

The optimization is carried out via coordinate descent, and the regularization path is evaluated over a logarithmically spaced grid of λ values. The final model corresponds to the λ that yields the lowest average validation error.

The complex-valued nature of the data posed significant challenges for the implementation of the regression. An initial approach based on the modulus, $|\mathbf{U}(t)|$, was attempted but proved unsuccessful. Subsequently, we adopted a strategy that treats the real and imaginary components separately. Specifically, both the data and the dictionary functions were split into their real and imaginary parts, which were then stacked vertically to form an extended real-valued dataset. The imaginary unit multiplying the interaction coefficients was incorporated directly into the precomputed values of the dictionary functions.

We experimented with several Lasso-based regression strategies, including growing-window learning, batch-wise model aggregation, and greedy interaction selection. Each method sheds light on different aspects of learnability, model robustness, and the structure of turbulence.We chose to focus primarily on a dataset characterized by the parameters: $nn = 20$, $r = 2$, $sstep = 100{,}000$, and $step = 5 \times 10^9$, resulting in a total of 50,000 time samples for each shell. Unless otherwise specified, all subsequent analyses refer to

this dataset. In the following, we display selected example images, showing only a subset of shells and interaction coefficients to avoid excessive clutter. A more comprehensive collection of visualizations is provided in the Appendix.

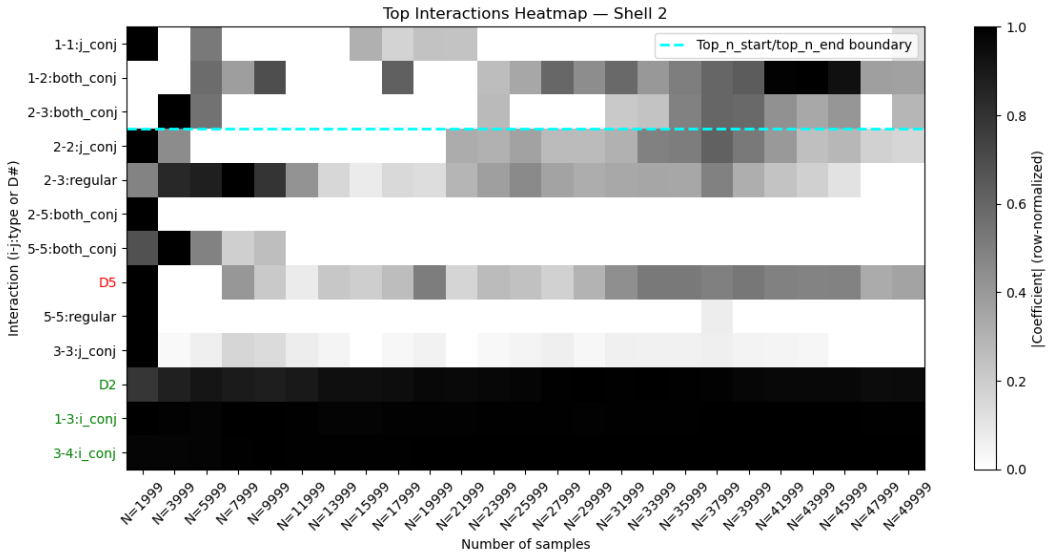

Figure 3: Evolution of NoSubDiss Lasso coefficients for selected interaction terms for shell 2, as a function of the number of training samples. Each row corresponds to an interaction, identified by the 2 numbers of the shell interacting and by the type of the interaction. "D" stands for dissipation. The values are row-normalized by the maximum value reached by the coefficient along its time evolution, to facilitate comparison in variability between interactions. in green the expected interactions/dissipations in black or red the unexpected interactions and dissipations.

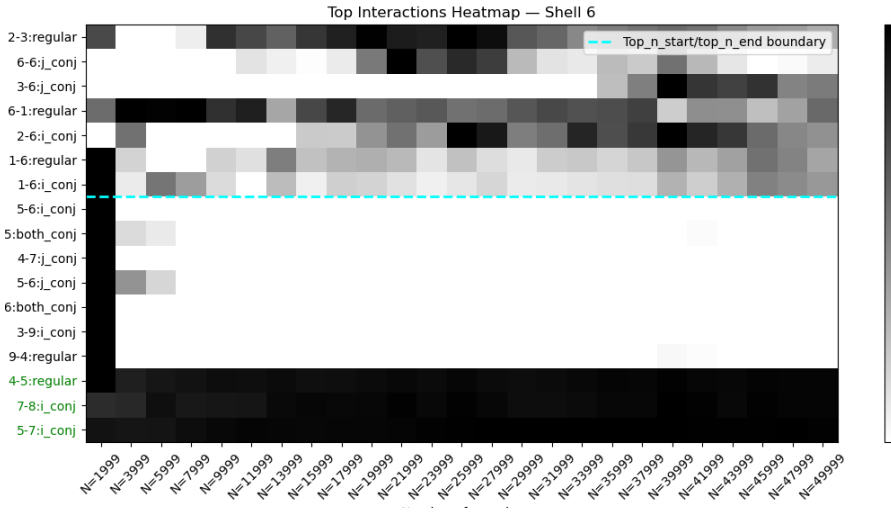

Figure 4: Evolution of SubDiss Lasso coefficients for selected interaction terms as a function of the number of training samples for shell 6.

Direct LassoCV on Growing Time Series

As a first approach, we applied `LassoCV` to each shell independently, using an increasing number of time samples from the generated time series. This setup allowed us to study the convergence behavior of the learned coefficients as more data became available. We recorded snapshots of the Lasso coefficients at various stages of training, producing a temporal map of how the learned interactions evolved. This analysis highlights which interactions emerge early, which stabilize over time, and how the identified models fluctuate with increasing data availability. Fig.(3) shows a heatmap illustrating the evolution of

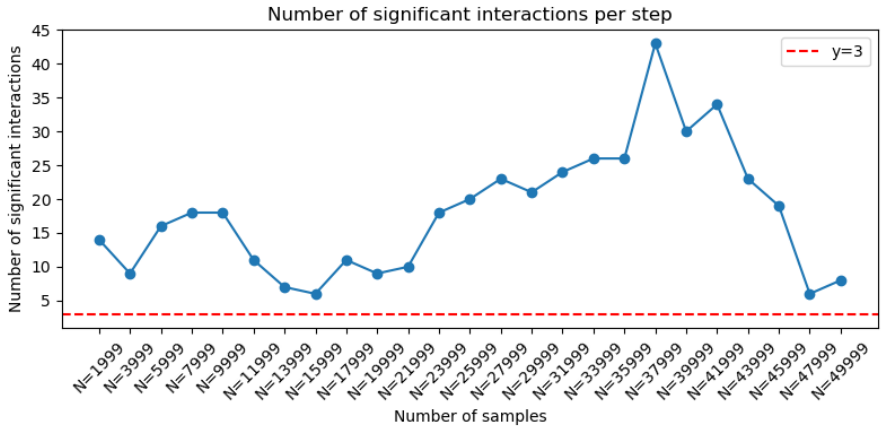

Figure 5: Evolution of number of nonzero NoSubDiss Lasso coefficients for selected interaction terms as a function of the number of training samples for shell 2.

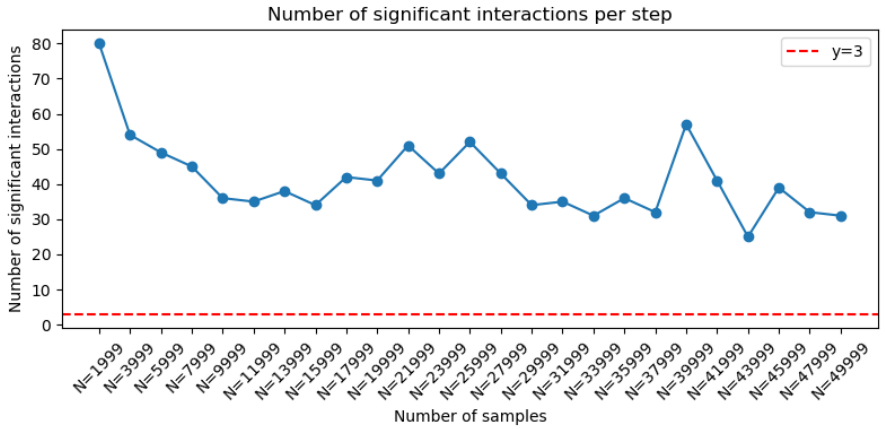

Figure 6: Evolution of number of nonzero SubDiss Lasso coefficients for selected interaction terms as a function of the number of training samples for shell 6.

the coefficients' magnitudes for **Shell 2** in the **NoSubDiss** case, while Fig.(4) presents the same for **Shell 6** in the **SubDiss** case.

It is important to note that during the fitting, `LassoCV` selects many unexpected interaction terms that are not shown in the figures. The interactions displayed were selected using the following criterion: we retained the top n (here $n = 10$) largest interactions (in absolute value) from the **first** snapshot ($N = 1999$) and the top n from the **last** snapshot ($N = 49999$).These two sets are visually separated by a blue dashed line in the heatmaps. If fewer than $2n$ total interactions are displayed, it indicates that some interactions were among the largest both at the beginning and at the end. At this point, one might be concerned by the presence of several unexpected interactions identified by the regression. To address this, it is instructive to examine the magnitudes of the corresponding coefficients.

Figure 7 clearly shows that the coefficients associated with the expected interactions are several orders of magnitude larger than those of the unexpected ones, effectively rendering the latter irrelevant. In practice, such small-magnitude coefficients may result from overfitting to noise introduced by the numerical integration procedure. Plots(5,6) support this interpretation: in both cases, the number of identified interactions decreases as more samples are added, reaching a minimum around 14,000 samples. Beyond this point, the number either stabilizes or begins to increase again, which may signal the onset of overfitting. Assuming the underlying model is strongly sparse, one might consider manually increasing the regularization parameter λ beyond the value selected by `LassoCV`, in order to enforce a more stringent sparsity constraint. When doing so, the coefficient selection improves considerably, and in some cases becomes exact,as shown in fig. 16,17 in the Appendix. However, this approach is inherently heuristic and difficult to justify unless prior knowledge about the number of active interactions is available.

So far, several observations can be made. The model generally succeeds in identify-

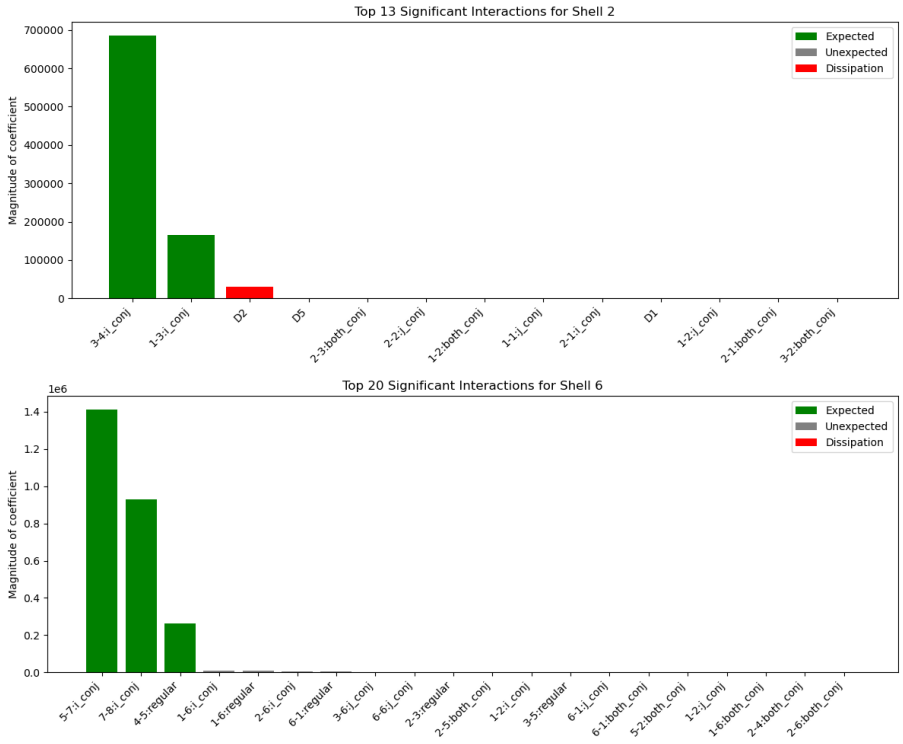

Figure 7: Magnitude of coefficients for NoSubDiss shell 2 and SubDiss shell 6 at the final snapshot. The histograms show a clear separation in scale between expected and unexpected interactions, with expected ones dominating by several orders of magnitude.

ing the expected interactions, which are consistently present throughout the increasing sample window and exhibit large magnitudes. Some unexpected interactions are also selected, but they typically appear with smaller magnitudes and less persistence.

However, the quality of learning degrades significantly under certain conditions. Notably, we observe a clear deterioration in performance for higher shell indices. While the lower shells—representing large-scale dynamics—tend to converge toward sparse and interpretable models, the higher shells exhibit noisier and less reliable results. This is likely due to a reduced signal-to-noise ratio at smaller scales, combined with the increasingly stiff dynamics of the system, which complicate the regression task.

As shown in the figures in the Appendix, and evident also from Fig. (9,10), the SubDiss case yields robust results across all shells: the correct interactions are recovered consistently. Still, both the number and magnitude of unexpected interactions increase with shell index.

In contrast, the NoSubDiss case presents a more fundamental difficulty. Although the regression still captures the expected dissipation terms, the majority of expected nonlinear interactions are no longer learned. As the shell number increases, the dissipation coefficients, which scale quadratically with k_n, begin to dominate, effectively disrupting

the identification of meaningful interactions, as shown in Fig.(8). A preliminary approach to face this problem consisted in normalizing both the derivatives and $\Phi(U)$ by the value k_n of the target shell, but it did not lead to a significant improvement. The magnitude of the coefficients, as expected, is not accurately recovered by `LassoCV`. However, from equation (15), we know that the true coefficients follow a specific order when sorted by magnitude. This raises the question of whether the regression procedure is at least capable of recovering the correct ranking. Unfortunately, as shown in figs. 22 and 23, in the Appendix, this is almost never the case.

Batch-wise Lasso and Interaction Consensus

We saw how in the first approach we encountered several difficulties: the inability to recover the correct magnitudes or ordering of the coefficients, the absence of interactions in

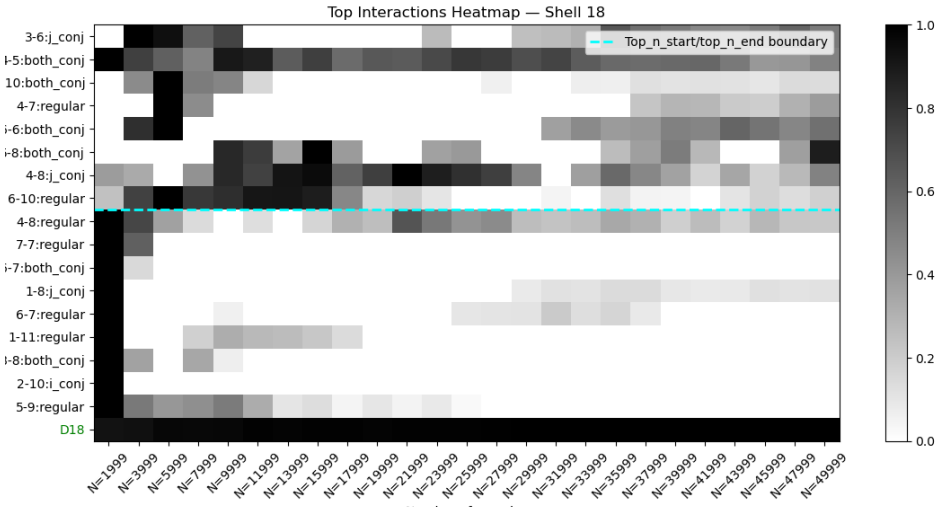

Figure 8: Evolution of SubDiss Lasso coefficients for shell 18: only the expected dissipation is found, no trace of the expected interactions.

the large-k shells for the NoSubDiss case, and the presence of many undesired interactions.

To address the latter issue, we explored alternative strategies aimed at improving interaction selection. Specifically, we partitioned the time series into several batches and applied independent Lasso regressions in parallel, one for each batch. For each regression, we recorded which interaction terms were selected. We then counted the number of times each interaction or dissipation term was selected across all batches. To retain only the most relevant terms, we introduced a threshold on the minimum number of occurrences required for a term to be considered significant.

From the set of selected terms, we constructed a final model $\mathbf{W}_{\text{final}}$ using only the retained coefficients. We experimented with various parameter combinations, including the number of batches ($n_{\text{batches}} = 5, 10$) and the occurrence threshold (minocc $= 5, 8, 9$).

This allowed us to identify interactions that appeared consistently across batches, indicating robustness and potential physical relevance. In the following, we present several figures to comprehensively visualize and compare the results of the increasing-window approach (from now on referred to as *single*) and the batch-wise approach.

Batch vs Increasing-Window Comparison

In Fig.(9), two panels are shown for the SubDiss case: the top one refers to the batch approach (5 batches, minocc=5), while the bottom corresponds to the single approach. Each panel includes four subplots:

The **top left** subplot shows the number of selected interaction terms per shell, separated into expected and unexpected. This highlights a clear improvement introduced by the batch approach: the number of unexpected interactions is significantly reduced, especially for large-k shells. (Note the difference in vertical scale between the batch and single approaches). The **bottom left** subplot displays the same data, but zooms in on the expected interactions only. Red dashed lines indicate the maximum number of possible expected interactions. This limit accounts for the fact that, although there are only three types of expected interactions, the dictionary also includes their symmetric counterparts (e.g., both $u_i^* u_j$ and $u_j u_i^*$, which are effectively the same), and can lead to apparent duplication of interactions. We observe that Lasso successfully selects nearly all expected coefficients for every shell. However, in the batch approach, the cross-checking procedure slightly impacts the recovery of expected terms, resulting in one missed interaction for shell 2 and two missed interactions for shell 19.The **top right** subplot shows the percentage contributions of expected and unexpected interactions to the total sum of coefficient magnitudes. This confirms the earlier observation: although unexpected interactions may be numerous, their magnitudes are generally much smaller than those of expected terms.

As a result, the main contribution to the total coefficient magnitude consistently comes from the expected interactions. The **bottom right** subplot displays how the total sum of coefficient magnitudes varies with the shell index. Figure(10) presents the same set of plots for the NoSubDiss case.

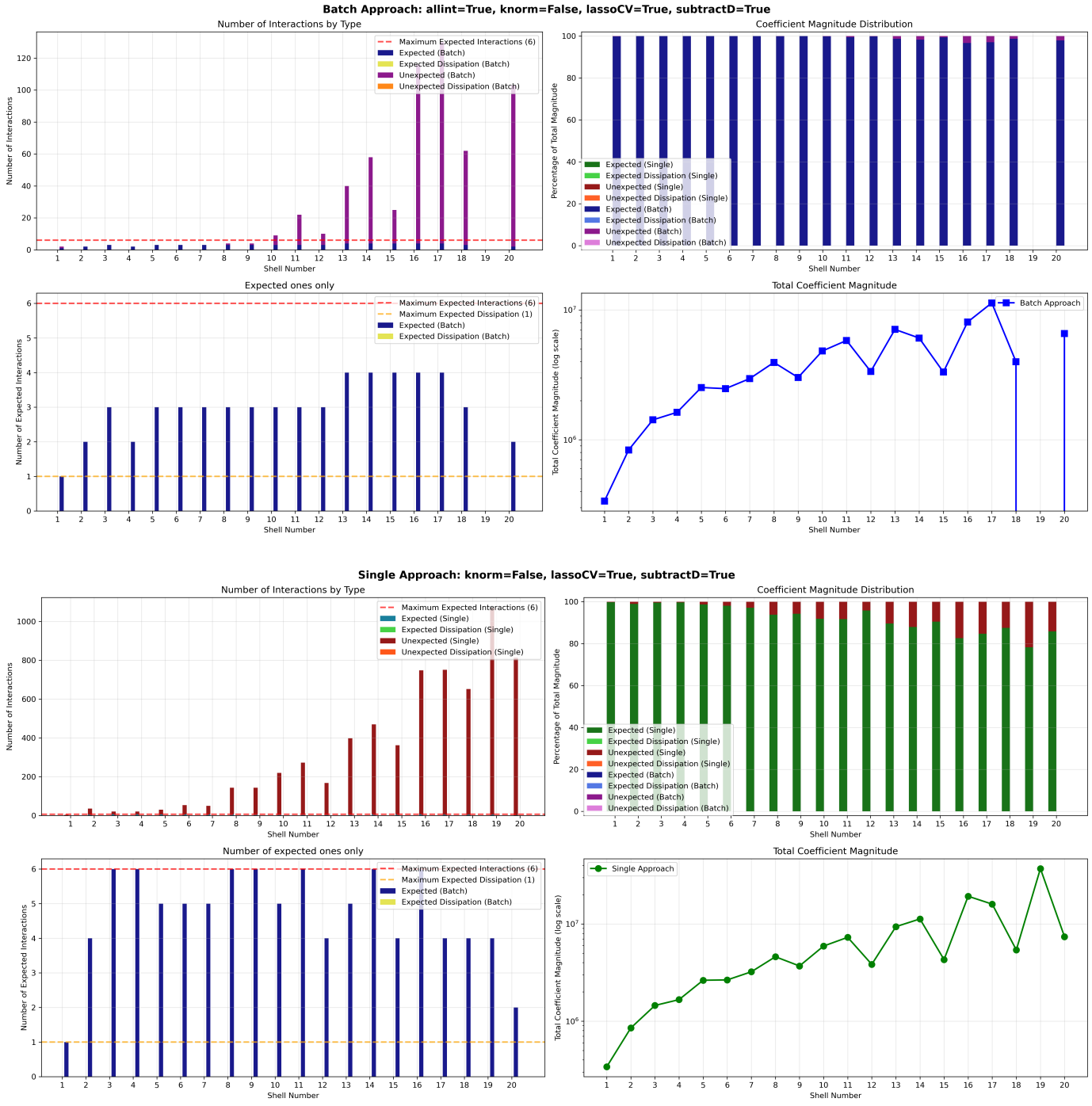

Figure 9: Summary of the two approaches for SubDiss. Top frame: batch approach with $n_{(batches)}$ and minocc=5. Top left: number of interactions per shell, split into expected and unexpected. Bottom left: same, zoomed in on expected only. Top right: percentage of expected vs. unexpected contribution to total coefficient magnitude per shell. Bottom right: total coefficient magnitude per shell.

Here, one can clearly observe the emergence of expected dissipative terms and their dominant contribution to the overall magnitude. More importantly, however, we note

a stark difference between the two approaches: while the single-window method results in partial recovery of expected interactions, the batch approach leads to their complete disappearance at higher shell indices.

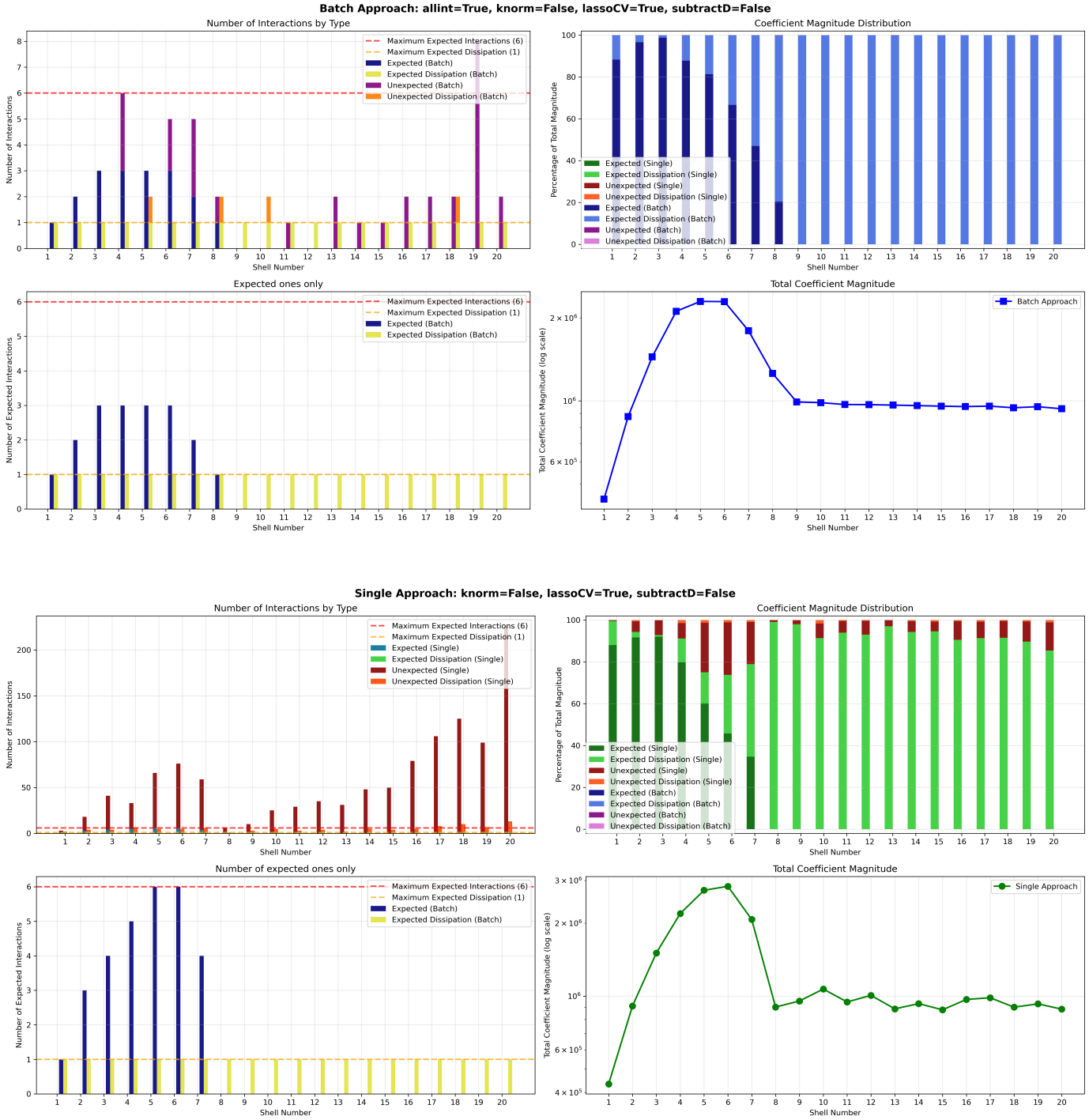

Figure 10: Same summary, but for NoSubDiss case.

In Figure(12), we propose a further global visualization of the regression results.For each shell, we selected the 10 largest interactions/dissipations, expected or not.We then filled a global list of all $10 \cdot nn = 200$ selected interactions and sorted it by magnitude. From this sorted list, we extracted the top 90 terms and plotted their occurrence in an interaction matrix. In each matrix, both the x- and y-axes correspond to shell indices. A filled square indicates that the corresponding shell pair engages in an interaction or dissipation. The colormap encodes only the presence or absence of a selected interaction, and

thus any overlap between terms is not explicitly shown unless originated by a dissipation and an interaction.

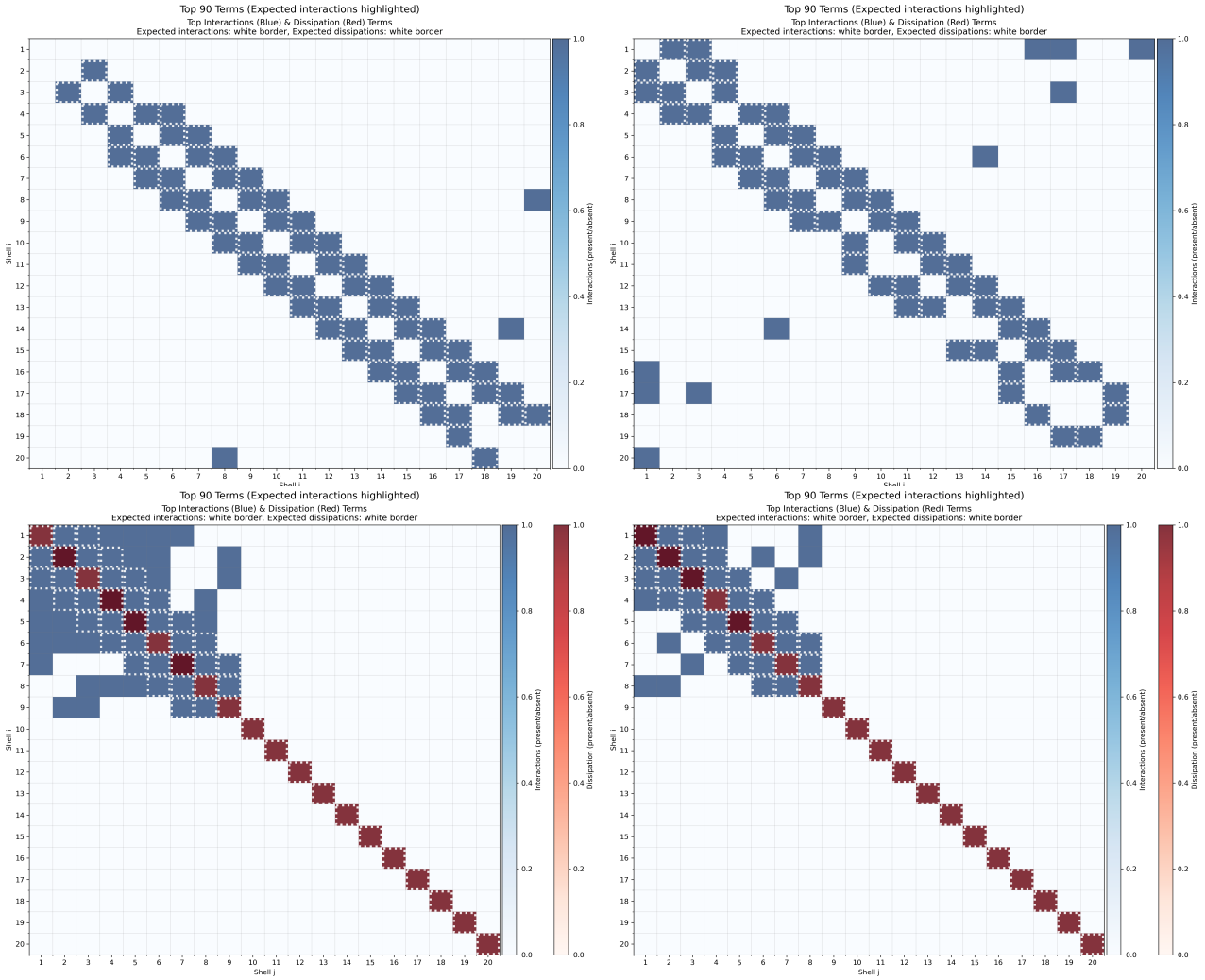

Figure 12: Matrix of all interactions/dissipations learnt by each shell. top left and right: single and batch ($n_{\text{batches}} = 5$, minocc=5) in SubDiss. Bottom left and right: single and batch ($n_{\text{batches}} = 5$, minocc=5) in NoSubDiss. The dashed white borders are present if the interaction is expected.

Greedy Selection of Interactions by Importance

Lastly, we reversed the standard Lasso approach. Instead of starting with a full dictionary of interactions and selecting a sparse subset, we masked all interactions but one, applied `LassoCV` (Also Ridge and OLS have been attempted, leading to similar results), and evaluated the resulting loss. Repeating this for each candidate interaction, we ranked them by their individual ability to reduce prediction error. This procedure might help reduce overfitting by avoiding situations where many small, unexpected interactions combine to explain the data. By testing one interaction at a time, we limit the chance of the model relying on noisy or spurious terms. We then built a sparse model incrementally by adding interactions one by one in the order of decreasing effectiveness. At each iteration, we measured the loss, selecting the interaction that provided the most significant improvement. In Fig.(13), we present the results of this procedure when a maximum of three interactions is selected per shell for the SubDiss case (shell indices are 0-based). The resulting interaction matrix closely resembles that of Fig.(12). Similarly, the behavior of the final R^2 scores shows a familiar pattern: a systematic decrease in fitting performance

with increasing shell number. The slight increase in R^2 observed beyond shell 17 might be explained by the fact that fitting shells 18 and 19 requires only two and one interactions, respectively.

An additional observation, less evident in Fig.(13), becomes more apparent in Figs.(21,20) in the appendix, where up to five or seven interactions per shell are allowed.In the evolutions of R^2 value we can clearly see that (particularly for small-k shells), the R^2 curves often exhibit an "elbow" at step 2—corresponding to the addition of the third interaction. After this point, the increase in R^2 becomes significantly slower. This suggests that the first three interactions contribute most of the predictive power, while additional interactions provide diminishing returns. This suggests the right number of expected interactions per shell.

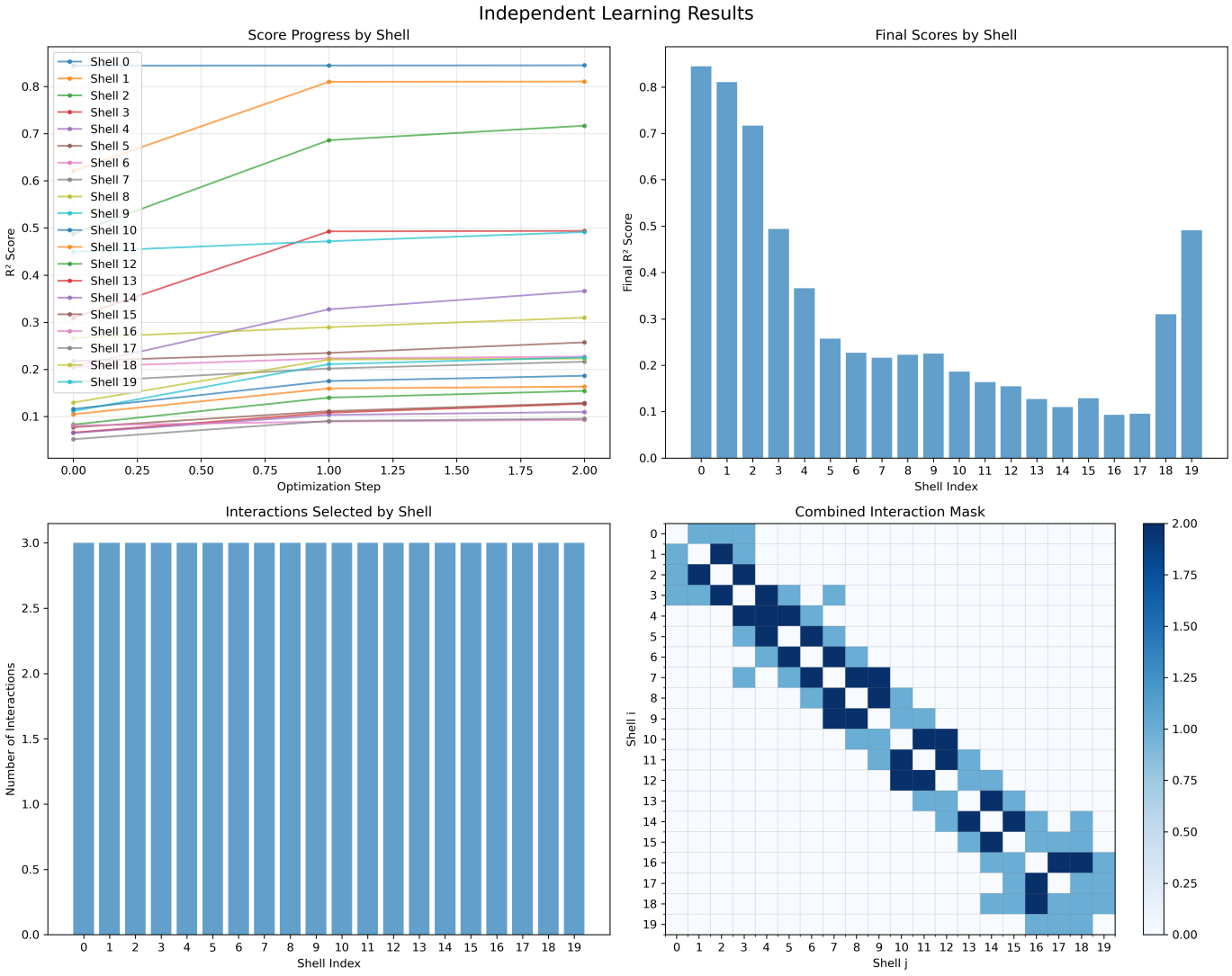

Figure 13: Greedy selection. top left: R^2 scores as a function of the steps for all shells, top right: final R^2 scores by shell, bottom left: number of selected interactions by shell, bottom right: interaction matrix (note that now the shells are indexed with a 0-based convention).

Synthesis and Interpretation of Results

Each of the above methods offers complementary insights into the regression of interaction terms from dynamical data. The growing-window Lasso approach reveals how learning

stabilizes as the data volume increases. In particular, it confirms the **consistency and persistence of the expected interactions** across increasingly larger sample sizes, while also indicating the approximate number of samples required to identify the relevant coefficients. This insight is valuable beyond the first method itself: for instance, if around 14,000 samples are necessary for stable identification, then in the batch-wise setting using 5 batches of 50,000 samples each (i.e., batches of 10,000 samples) already brings us reasonably close to that target. Taking significantly more batches would reduce the batch size and risk moving further from this ideal, potentially weakening the identification.

The second, batch-wise approach provides a **refined mechanism for filtering out spurious interactions**, improving sparsity and interpretability. By cross-verifying which interactions appear consistently across multiple independently trained models, it becomes possible to rule out most of the unexpected terms. Interestingly, even when the batch data are randomly shuffled in time—thus destroying temporal continuity—**the expected interactions remained among the most frequently recovered**, further supporting their robustness and likely physical relevance.

The third method, based on a **greedy ranking of interactions by individual effectiveness**, complements the previous approaches by offering a reverse, bottom-up perspective. It does not assume sparsity from the beginning but builds it progressively, selecting at each step the interaction that most improves predictive performance. The results from this method **confirm the relevance and dominance of the expected interactions**, as these are typically selected in the first few steps. The appearance of a clear "elbow" in the R^2 growth curve—especially prominent in small-k shells—after the inclusion of the third interaction, **strongly suggests that three interactions are sufficient** to capture the essential dynamics per shell. This aligns perfectly with the theoretical expectation and the patterns identified by the previous methods.

Across all approaches, a consistent theme emerges: **the expected interactions are always the most relevant**—either by being selected first in greedy procedures, by dominating the coefficient magnitudes by several orders, or by persisting across different subsets of data. Taken together, these findings build a coherent picture where, despite variations in strategy, the most meaningful interactions stand out both statistically and dynamically. This suggests that even in noisy or partial data regimes, a combination of thoughtful regression strategies can yield insight into the structure of the underlying physical system.

Conclusion and Future Perspectives

The analysis presented in this work demonstrates how different regression strategies can complement one another in uncovering the underlying interaction structure of the SABRA shell model. The growing-window approach helped establish the stability and consistency of expected interactions as more data are added, and even informed the selection of batch sizes for the second method. The batch-wise strategy improved model robustness by filtering out many spurious terms while retaining the key interactions. The greedy approach, in turn, confirmed from a reverse perspective that only a small number of interaction terms—typically three per shell—are needed to achieve high predictive accuracy, a fact reflected in the shape of the R^2 score curves.

A common feature across all methods was the dominance of expected interactions in terms of both frequency of selection and magnitude. These terms consistently appear as the most relevant, which reinforces their physical credibility and suggests a stable underlying structure in the system dynamics.

The next objective is to improve the estimation of the actual values of the interaction coefficients, Something at which `LassoCV` fails. Ridge regression has been tested, giving better results in terms of accuracy of the coefficients' values but paired with a less sharp feature selection. The crucial step is now to retain only the features selected by `LassoCV`

and perform ordinary least square (OLS) regression to recover their associated coefficients, this time with the exact numerical value. Unfortunately, such procedure, apparently trivial, has turned out to be problematic and unable to correctly identify the expected coefficients.

In parallel, we propose a possible future coarse-grained approach exploiting again the Lasso feature selection. Rather than focusing on individual interaction terms, we group them into *interaction classes* based on the shell separation distance $d = |i - j|$ between the indices of the interacting shells i and j. For each shell n, we retain only those dictionary terms previously identified by the `LassoCV` approach in which shell n interacts with shell $n \pm d$, and aggregate them into a single class indexed by d. Each class therefore captures all interactions occurring at a fixed shell distance. Using the coefficients estimated by the initial regression, we weight each term accordingly and construct a new dictionary whose entries are class-wise aggregated functions, this will be our new ϕ matrix. The associated new, shorter coefficient vector $\mathbf{W}$ will thus regulate the relative importance of each interaction class. This class-based reformulation may facilitate the identification of dominant interaction ranges—such as classes c_1 and c_2 in our case—and could enhance physical interpretability. Specifically, if we assume that interaction dynamics do not vary significantly across shells, we can encourage structural consistency by coupling the regressions performed for different shells. One strategy is to introduce an *adaptive regularization term* in the per-shell loss function: if, for example, class c_2 is found to be important for shell 1, we can reward its selection in subsequent shells by adding a term whose strength grows with the number of prior shells that identified c_2 as relevant. This history-aware regularization helps reinforce coherent model structure and may even stabilize learning for large-index shells, where the fitting has proven less effective.

Alternatively, we propose a more integrated approach via *joint learning*. Instead of fitting each shell independently, we concatenate all shell time series into a single target vector $\mathbf{y}$, and we stack the class-based interaction dictionaries for each shell horizontally into a global matrix Φ. In this coupled setting, the regression yields a single coefficient vector $\mathbf{W}$ that governs the importance of each interaction class across all shells simultaneously. This strategy extracts information globally, ensuring that the relative relevance of each class is inferred from the collective behavior of the system.

Finally, we plan to explore the use of data-driven generative models. In particular, we already implemented a first Restricted Boltzmann Machine (RBM) on the time series of shell variables. The RBM will be used to model the statistical structure of the system, potentially allowing us to generate synthetic but physically consistent dynamics. The team has also developed a theoretical framework to extract causality relationships from the structure of a trained RBM. Such framework could be applied also in our case, to push forward the power of pattern recognition and dynamics modelization in Turbulence.

References

[1] Luca Biferale. Shell models of energy cascade in turbulence. *Annual Review of Fluid Mechanics - ANNU REV FLUID MECH*, 35:441–468, 01 2003.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[3] Tomas Bohr, Mogens H. Jensen, Giovanni Paladin, and Angelo Vulpiani. *Dynamical Systems Approach to Turbulence*. Cambridge Nonlinear Science Series. Cambridge University Press, 1998.

[4] Uriel Frisch. *Turbulence: The Legacy of A. N. Kolmogorov*. Cambridge University Press, 1995.

[5] Andrei Nikolaevich Kolmogorov, V. Levin, Julian Charles Roland Hunt, Owen Martin Phillips, and David Williams. Dissipation of energy in the locally isotropic turbulence. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 434(1890):15–17, 1991.

[6] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, 2019. A high-bias, low-variance introduction to Machine Learning for physicists.

[7] L.E. Melkumova and S.Ya. Shatskikh. Comparing ridge and lasso estimators for data analysis. *Procedia Engineering*, 201:746–755, 2017. 3rd International Conference "Information Technology and Nanotechnology", ITNT-2017, 25-27 April 2017, Samara, Russia.

[8] Giovanni Paladin and Angelo Vulpiani. Anomalous scaling laws in multifractal objects. *Physics Reports*, 156(4):147–225, 1987.

[9] G. Parisi and Uriel Frisch. On the singularity structure of fully developed turbulence in turbulence and predictability in geophysical fluid dynamics and climate dynamics. *NTurbulence and Predictability of Geophysical Flows and Climate Dynamics*, 88, 01 1985.

[10] A. R. Paterson. Statistical fluid mechanics: Mechanics of turbulence. vol 2. a. s. monin and a. m. yaglom. mit press, cambridge, massachusetts. 1975. 874 pp. £25.00. *The Aeronautical Journal*, 80(781):44–44, 1976.

[11] Lewis Fry Richardson. *Weather Prediction by Numerical Process*. Cambridge Mathematical Library. Cambridge University Press, 2 edition, 2007.

Appendix: some additional figures

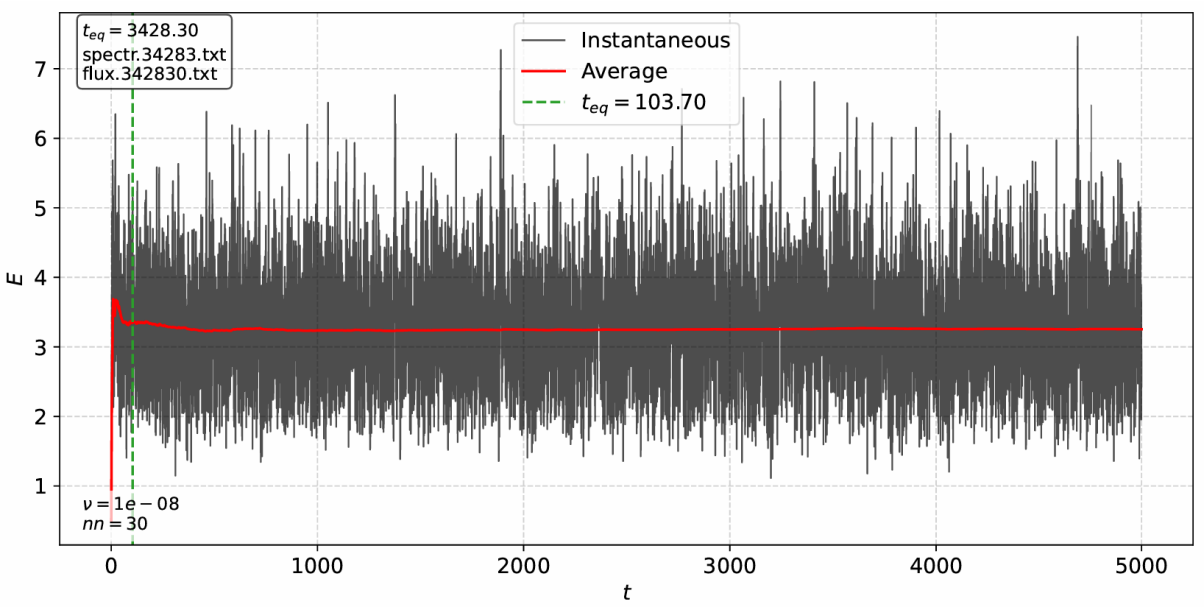

Figure 14: $r = \frac{1+\sqrt{5}}{2}, nn = 30, \nu = 10^{-8}, \epsilon = 1$. same plot as fig. 1, for a different set of parameters. The t_{eq} in the top left corner is the equilibration time found for quantity eq. (16)

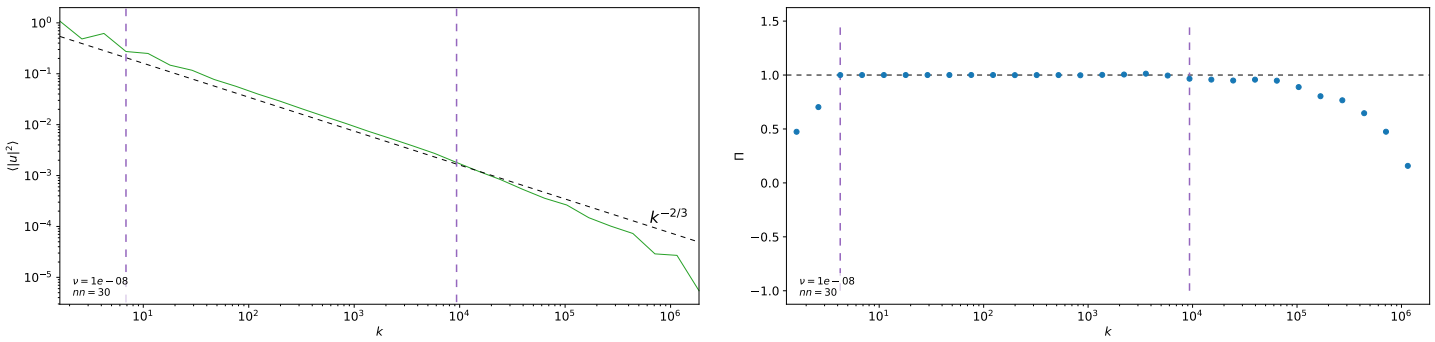

Figure 15: $r = \frac{1+\sqrt{5}}{2}, nn = 30, \nu = 10^{-8}, \epsilon = 1$. same plot as fig. 19, for a different set of parameters

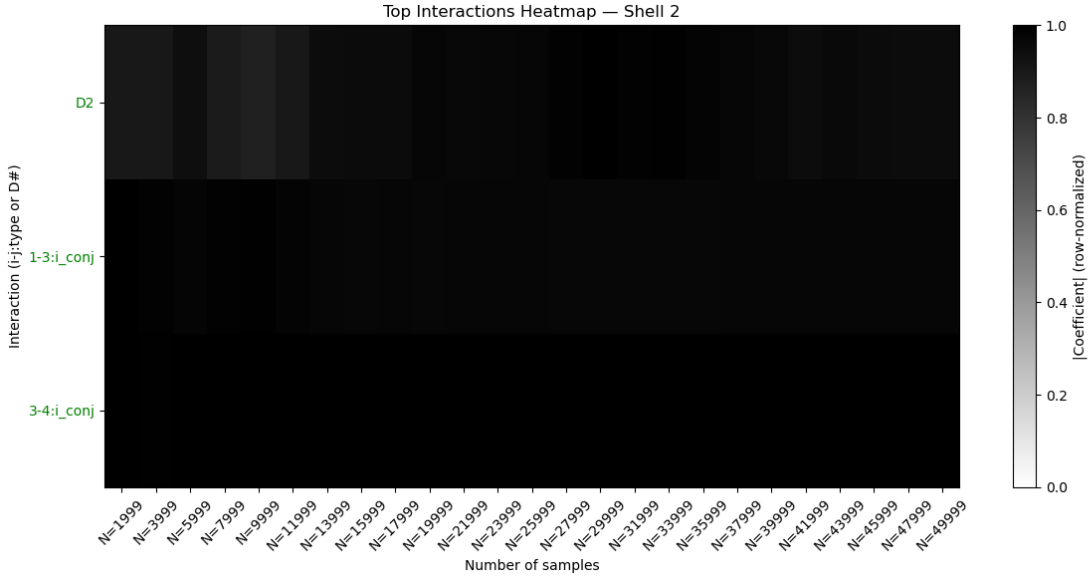

Figure 16: Evolution of SubDiss coefficients for Shell 2 as a function of number of samples obtained using `Lasso` instead of `LassoCV`, with fixed $\lambda = 500$. The only nonzero coefficients are the expected ones.

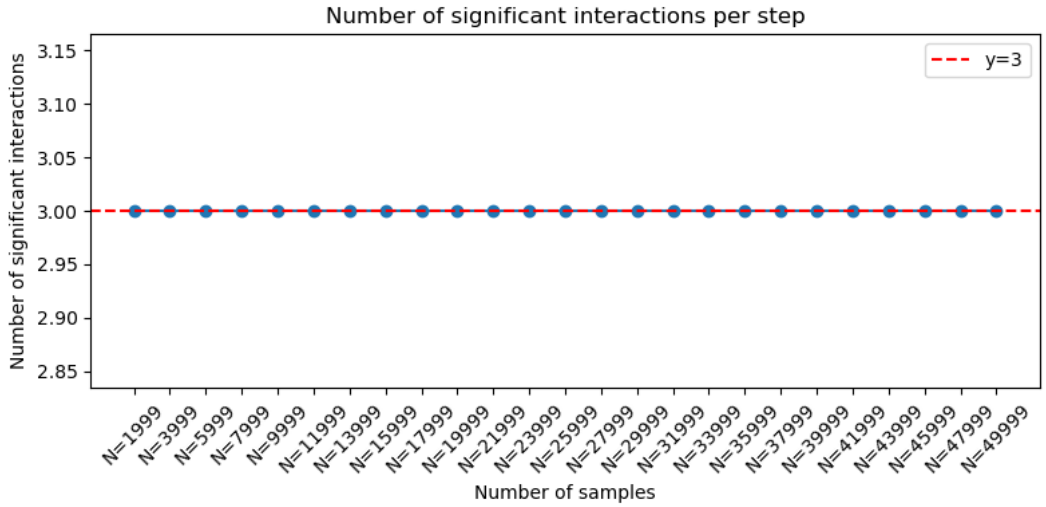

Figure 17: number of nonzero SubDiss coefficients for shell 2 as a function number of samples obtained using `Lasso` instead of `LassoCV`, with fixed $\lambda = 500$. The number of coefficients is 3 (corresponding to the 3 expected interactions) already after 2000 samples, and remain constant all the way.

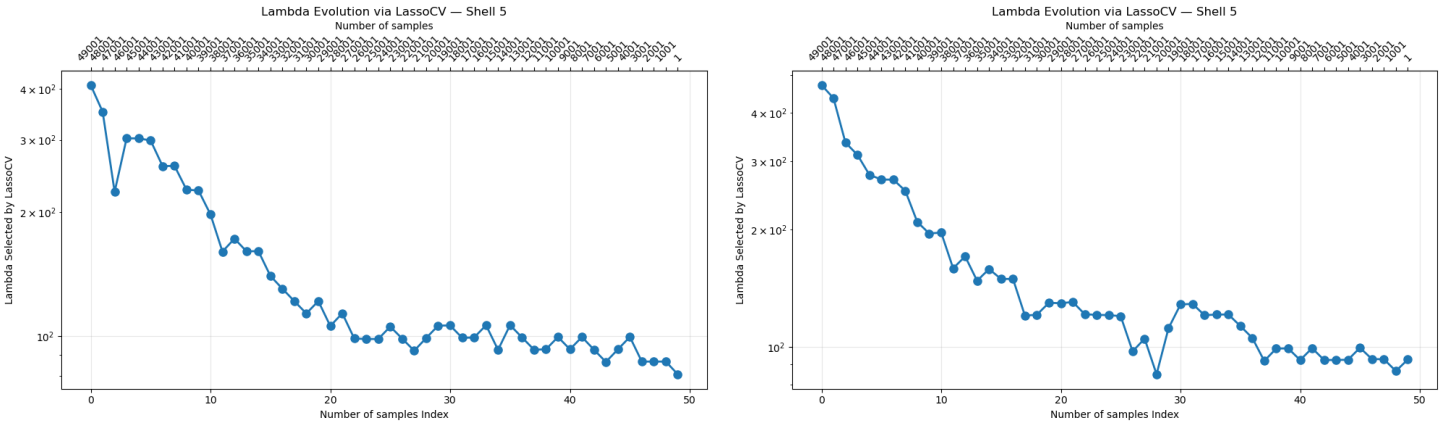

Figure 18: Evolution of values of λ selected by LassoCV as a function of the number of samples for shell 4 SubDiss (right) and NoSubDiss (left). The x-axis is reversed: tick 49001 means that 999 samples have been used.

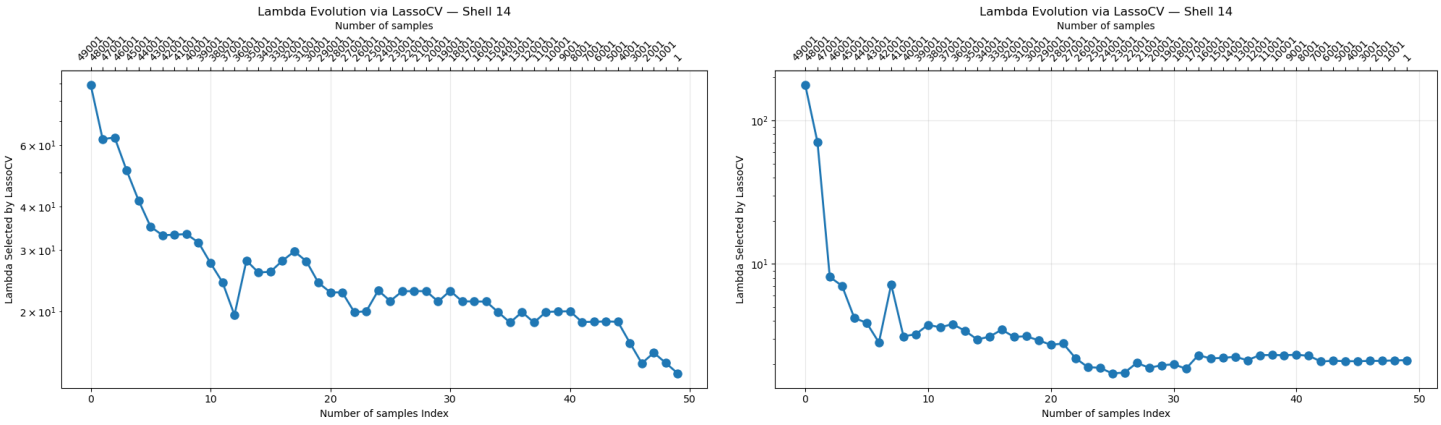

Figure 19: Evolution of values of λ selected by LassoCV as a function of the number of samples for shell 13 SubDiss (right) and NoSubDiss (left).

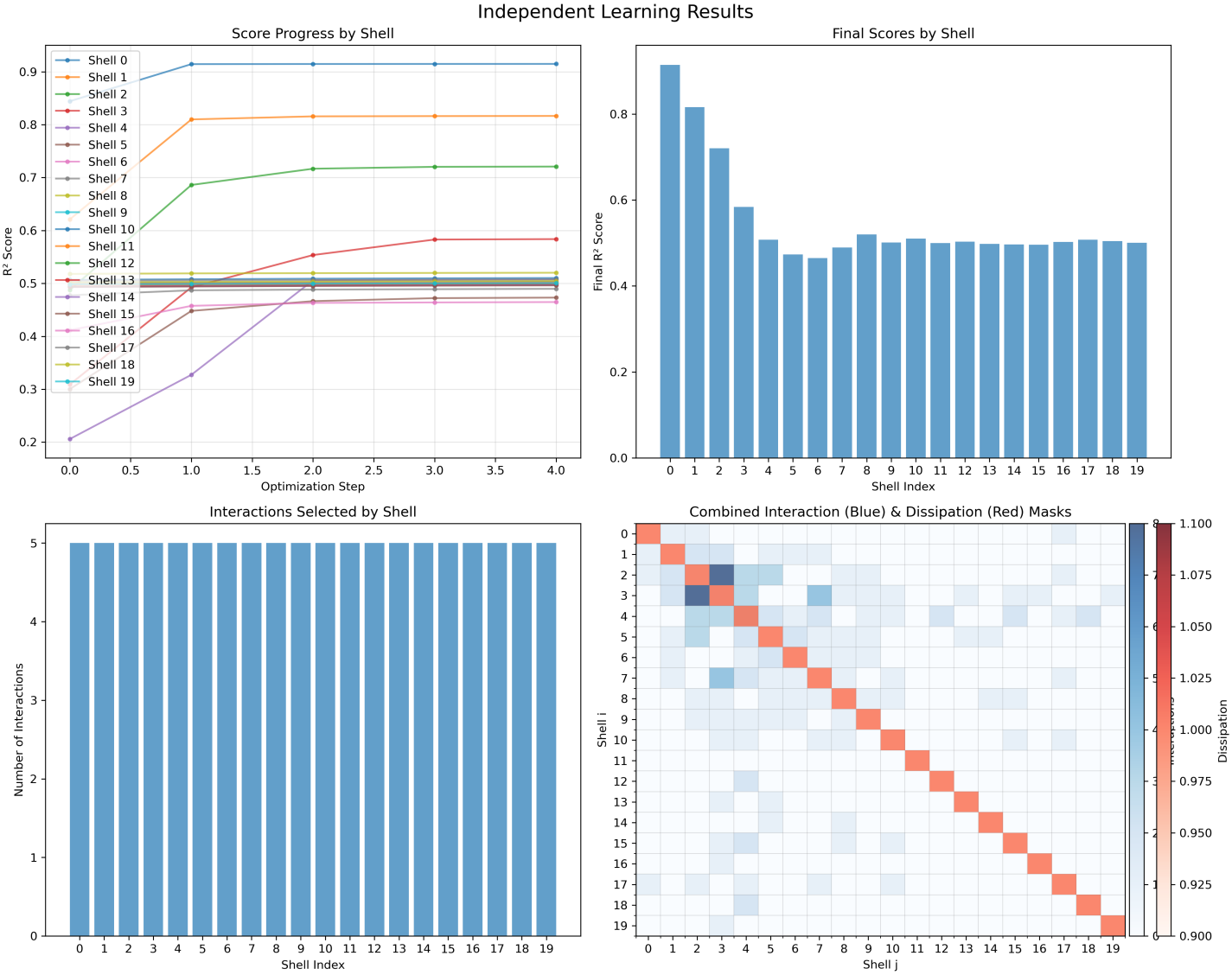

Figure 20: Greedy selection allowing for 5 interactions per shell, NoSubDiss.

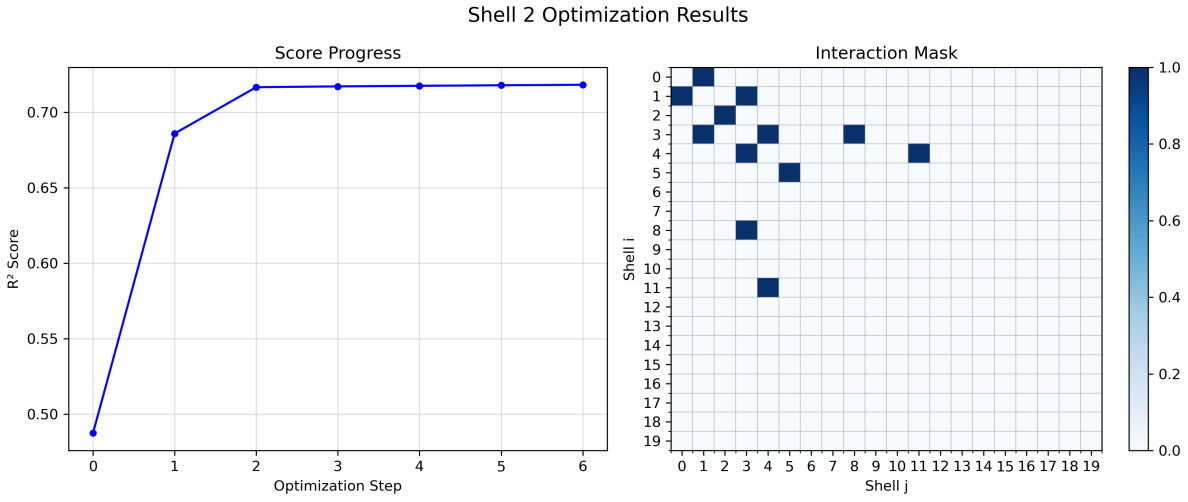

Figure 21: Greedy selection for a single shell SubDiss. Also here one can clearly observe the "elbow" at step 2

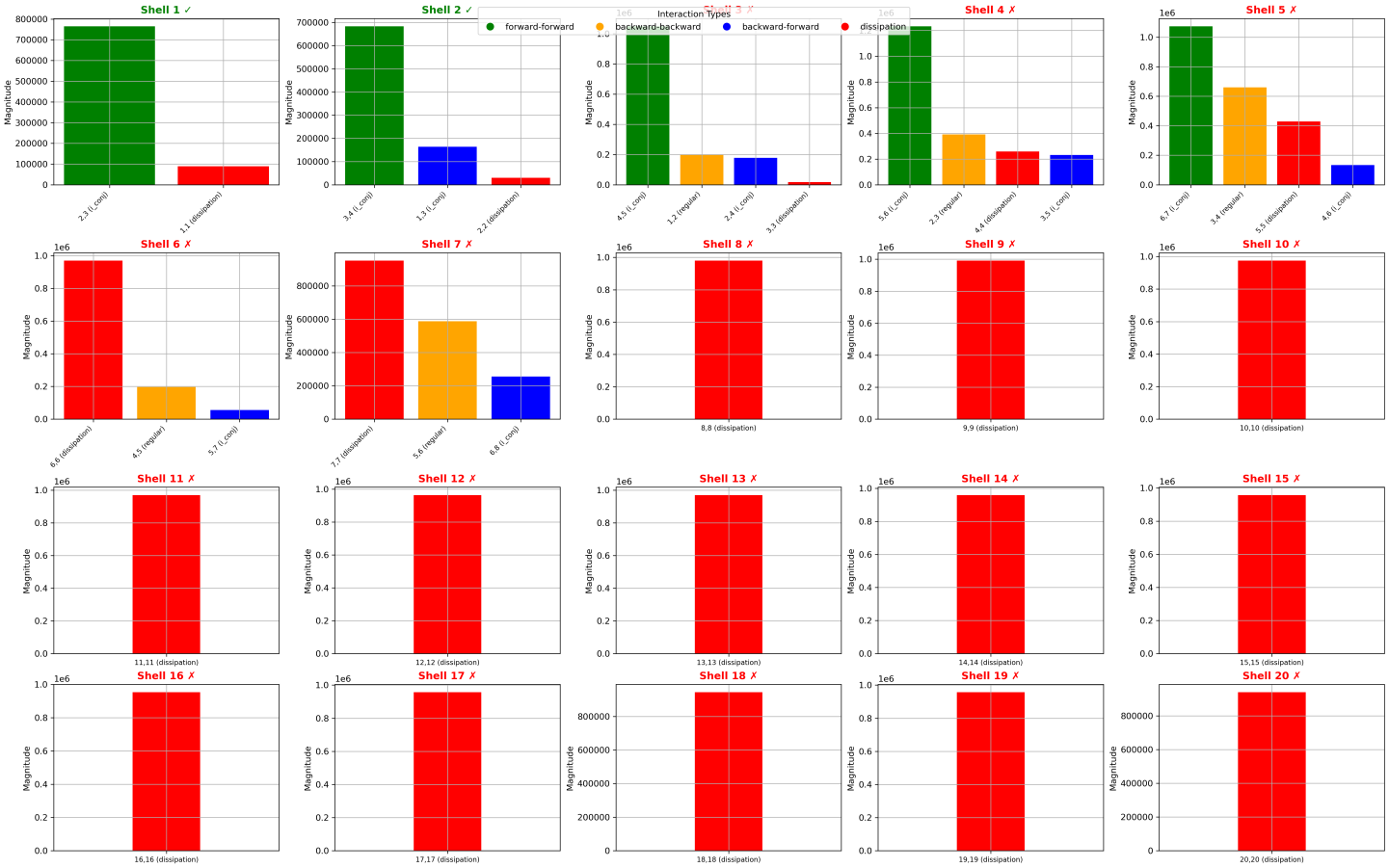

Figure 22: Ranking of magnitudes of expected coefficients NoSubDiss learnt by the model, different colors of the bins indicates different types of interactions. The tick or the cross on top of each graph indicate if the relative ordering in the coefficients' magnitude is correctly learnt.

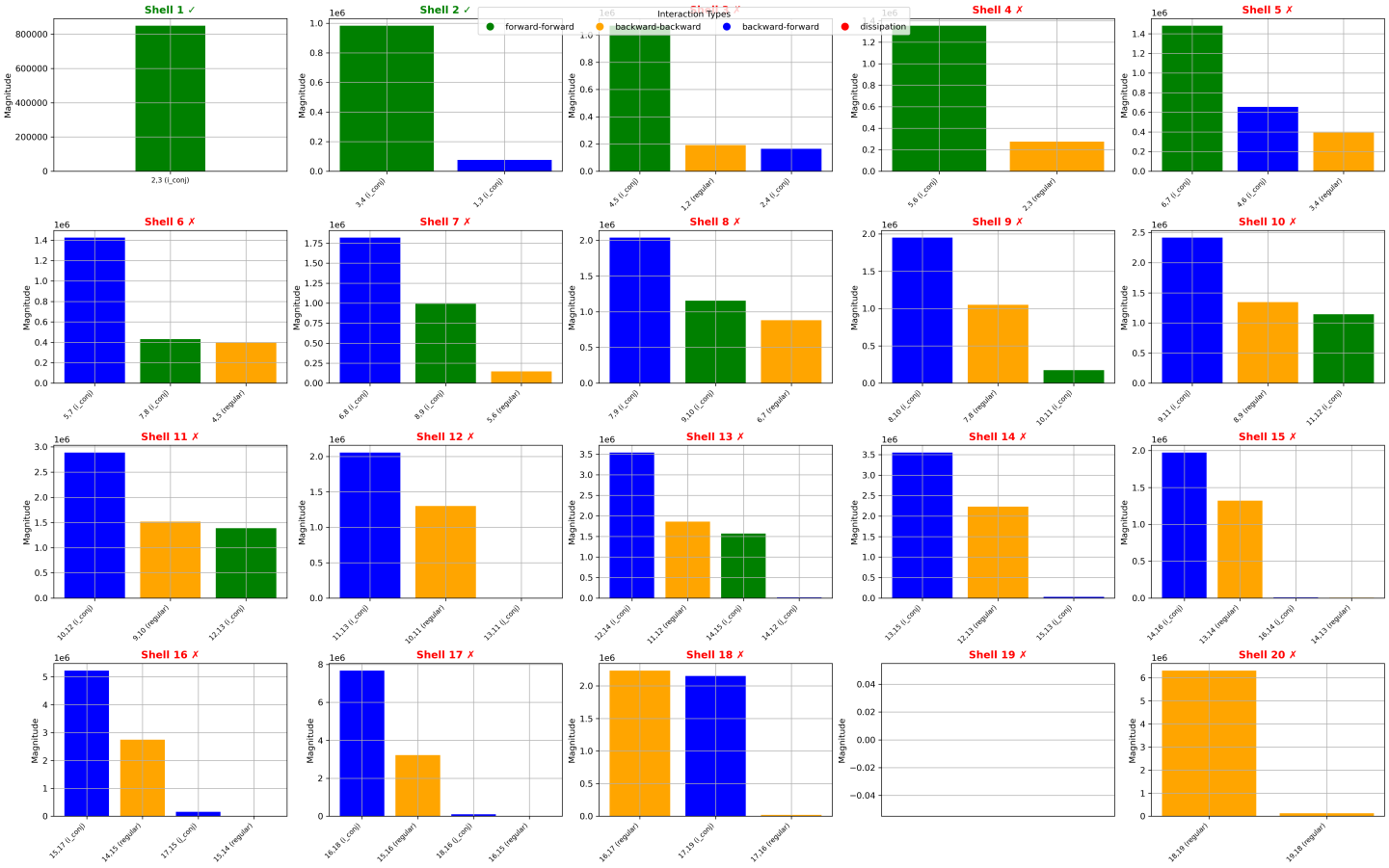

Figure 23: Ranking of magnitudes of expected coefficients SubDiss learnt by the model, different colors of the bins indicates different types of interactions. The tick or the cross on top of each graph indicate if the relative ordering in the coefficients' magnitude is correctly learnt.