Foundation models
for spatial-time series

Week 1

In December 2024, a NeurIPS workshop Foundational Models for Science

TheWell

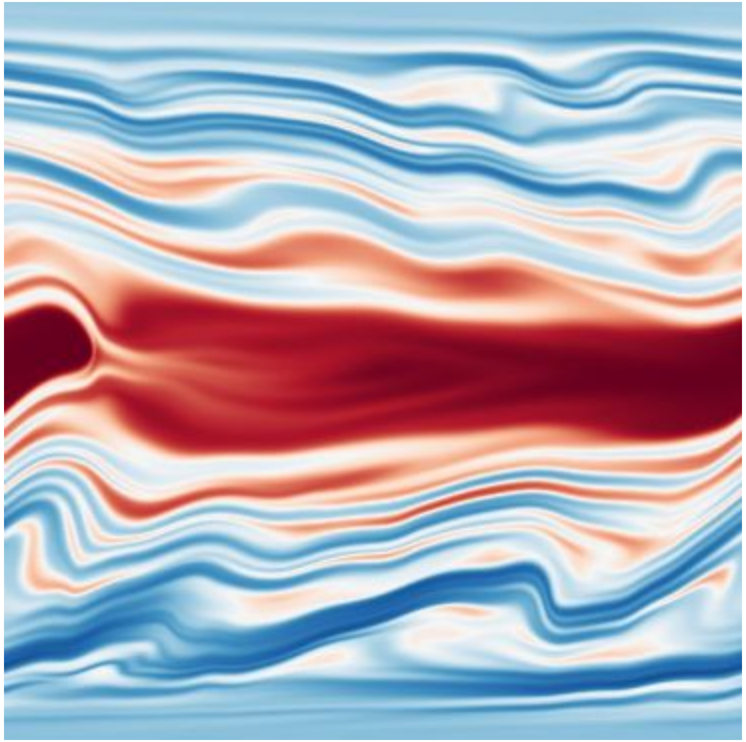
completely reflected our course “Functional Data Analysis” in September 2024

1. Foundation Models for Science: Progress, Opportunities, and Challenges URL
2. Foundation Models for the Earth system UPL, no paper
3. Foundation Methods for foundation models for scientific machine learning URL, no paper
4. AI-Augmented Climate simulators and emulators URL, no paper
5. Provable in-context learning of linear systems and linear elliptic PDEs with transformers NIPS
6. VSMNO: Solving PDE by Utilizing Spectral Patterns of Different Neural Operators NIPS

March 2025 Physics problem Simulations

1. The Well: a Large-Scale Collection of Diverse Physics Simulations for ML ArXiv, Code
2. Polymatic Advancing Science through Multi-Disciplinary AI blog
3. Long Term Memory: The Foundation of AI Self-Evolution ArXiv

Multistability of viscoelastic fluids in a 2D channel flow



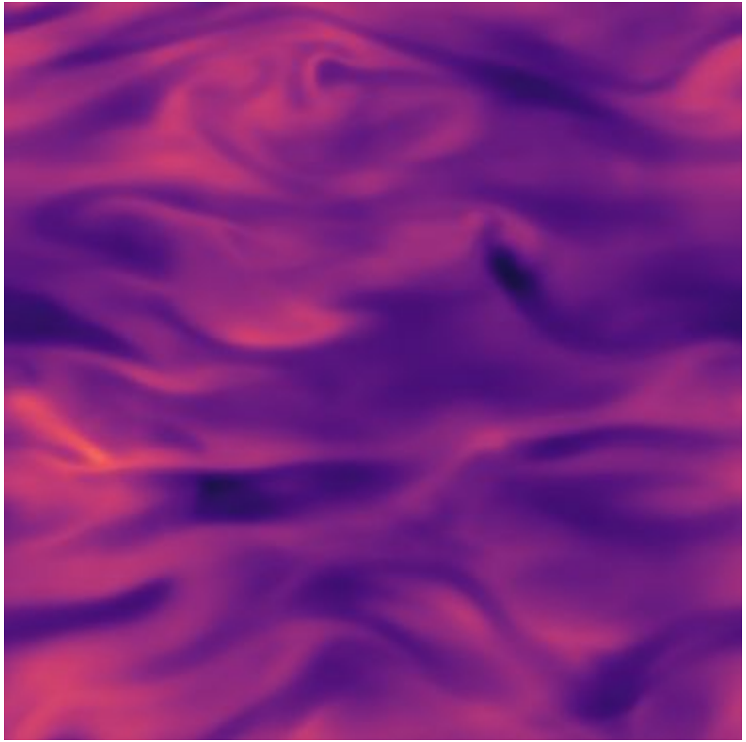
Multistability in viscoelastic flows, i.e. four different attractors (statistically stable states) are observed for the same set of parameters depending on the initial conditions.

$$
\begin{aligned}
Re(\partial_t \mathbf{u}^* + (\mathbf{u}^* \cdot \nabla)\mathbf{u}^*) + \nabla p^* &= \beta \Delta \mathbf{u}^* + (1-\beta)\nabla \cdot \mathbf{T}(\mathbf{C}^*), \\
\partial_t \mathbf{C}^* + (\mathbf{u}^* \cdot \nabla)\mathbf{C}^* + \mathbf{T}(\mathbf{C}^*) &= \mathbf{C}^* \cdot \nabla \mathbf{u}^* + (\nabla \mathbf{u}^*)^T \cdot \mathbf{C}^* + \epsilon \Delta \mathbf{C}^*, \\
\nabla \mathbf{u}^* &= 0,
\end{aligned}
$$

$$
\text{with} \quad \mathbf{T}(\mathbf{C}^*) = \frac{1}{\mathrm{Wi}}(f(\mathrm{tr}(\mathbf{C}^*))\mathbf{C}^* - \mathbf{I}),
$$

$$
\text{and} \quad f(s) := \left(1 - \frac{s-3}{L_{max}^2}\right)^{-1}.
$$

Magnetohydrodynamics (MHD) compressible turbulence

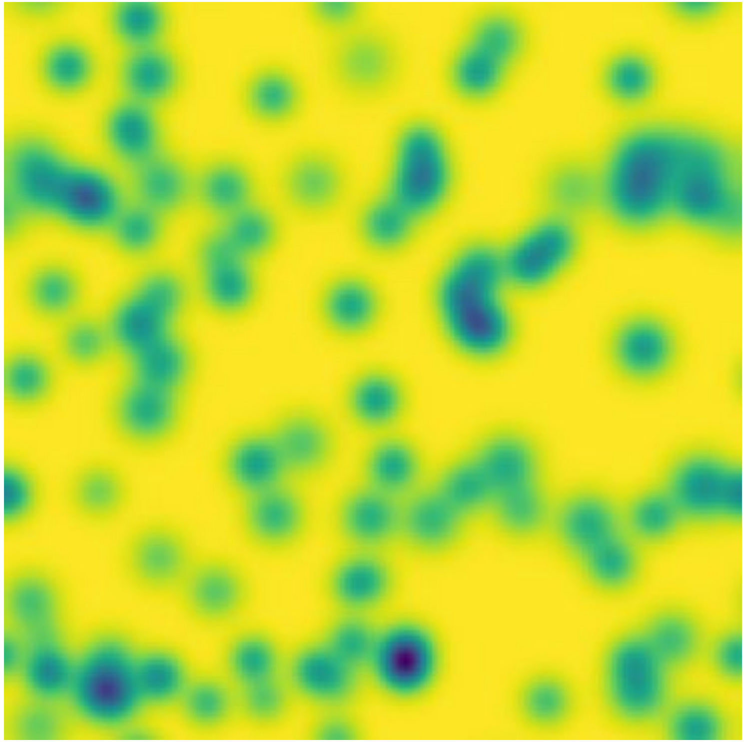


This is an MHD fluid flows in the compressible limit (subsonic, supersonic, sub-Alfvenic, super-Alfvenic).

$$
\begin{aligned}
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) &= 0 \\
\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v} - \mathbf{B}\mathbf{B}) + \nabla p &= 0 \\
\frac{\partial \mathbf{B}}{\partial t} - \nabla \times (\mathbf{v} \times \mathbf{B}) &= 0
\end{aligned}
$$

where ρ is the density, $\mathbf{v}$ is the velocity, $\mathbf{B}$ is the magnetic field, $\mathbf{I}$ the identity matrix and p is the gas pressure.

Gray-Scott reaction-diffusion equations



Stable Turing patterns emerge from randomness, with drastic qualitative differences in pattern dynamics depending on the equation parameters.

$$\frac{\partial A}{\partial t} = \delta_A \Delta A - AB^2 + f(1 - A)$$
$$\frac{\partial B}{\partial t} = \delta_B \Delta B - AB^2 - (f + k)B$$

Functional Data Analysis

m1p.org/fda

1. Multimodal data
2. Continuous time and space models
3. State spaces and convolution
4. Physics-informed models
5. Multilinear models
6. Riemannian spaces

The paradigm

The paradox of time series forecasting is that a simple model and a complex model (like SSA and LSTM) deliver the same or better accuracy of forecasting. An LLM-class model delivers poor accuracy. So the foundation model shall process the optimal pair (data, local model), acting as a mixture of experts for various models.

The problem and a possible architecture

1. For a
   a. set of time series and a context, we have to return
   b. an optimal mathematical model,
   c. an optimal local model with
   d. the optimal state space, and
   e. the accuracy of forecasting.
2. The architecture of the foundation model is a collection of the local models.
3. It constructs various phase spaces, learn the operator parameters, and compares the models
4. It learns relations or links between the operators.

Physics-informed learning is a type of mixture of experts. Since there are a number of vector field transformations, we learn not only the operations, but as a sequence the transformations between the operators.

Assumptions on the time series

1. There is a set of time series
2. This set is carried by a single timeline
3. This set is declared as a spatial time series
4. There is a relation between time series expressed by a metric tensor
5. Time series transforms to its phase trajectory
6. We forecast targets $y_{t+1} = f(x_t, y_t)$ there are two different phase spaces: for x and for y
7. There is a context of time series, unchanged in time

Initial models

1. Models
    a. Direct models: AR, ARIMA, GRU, LSTM
    b. Metric models: LLE, DM, GH, RBF
    c. Non-parametric: GPR
2. Ways to construct state spaces
    a. SSM models: S4, S5, Hippo
    b. Kalman
    c. SSA, SSM
3. Ways to transform state spaces
    a. FT, OL, ODE
    b. CCA, CCM

The time series has two domains: the time domain and the frequency domain. The spatial time series also has a metric space or metric tensor that changes in time.

Crawler and Language

Datasets: any that fit the assumptions below

Models: any model mentioning the interface of FM-wrappers

Context: for data and the models

1. BNCI Horizon 2020: open access BCI data sets
2. Climate Prediction Center: wind, sea level, and sea temperature for years
3. NFDA Book time series: satellite, spectrometric, phoneme, electricity consumption, El Niño

The plan and the scoring system

1.	Form your group and select a large model you will modify	
2.	Deploy the code without modification	
3.	Present the strategy of modification	(group evaluation)
4.	Modify and run examples	
5.	Present the intermediate model	(group evaluation)
6.	Select a physics-informed model, run the code	
7.	Present the math and the source code	(personal evaluation)
8.	Embed the model and present the test	(personal evaluation)
9.	Present the workflow	(group evaluation)

Comparative scoring (expert estimation of the idea quality and code evaluation)

Deliveries for the next

1) a comparative analysis of the foundation models (as a paper),
2) computational experiments with the foundation model comparison (.ipynb),
3) a selected foundation model of optimal architecture (described in a paper),
4) the model is developed as a software (pytorch or jax pipeline),
5) a basic database of the time series to train the foundation model,
6) a deployment setup (flask, aws),
7) use cases and examples (.ipynb), and
8) theoretical research of the model properties (a submitted paper).

References

1. Mahoney, M.W. (2024). Foundation Models for Science: Progress, Opportunities, and Challenges. Advances in Neural Information Processing Systems (NeurIPS).
2. Perdikaris, P. (2024). Foundation models for the Earth system. Advances in Neural Information Processing Systems (NeurIPS).
3. Zanna, L. (2024). AI-Augmented Climate Simulators and Emulators. Advances in Neural Information Processing Systems (NeurIPS).
4. Ohana, R. et al (2025). The Well: A Large-Scale Collection of Diverse Physics Simulations for Machine Learning. arXiv.
5. Cole, F. (2024). Provable in-context learning of linear systems and linear elliptic PDEs with transformers. Advances in Neural Information Processing Systems (NeurIPS).
6. Jing F. et. al. (2024). VSMNO: Solving PDE by Utilizing Spectral Patterns of Different Neural Operators. Advances in Neural Information Processing Systems (NeurIPS).

Method	Training Needed?	Modular Tool Add?	Industry Use	Best For
ToolkenGPT	Train embeddings	✅ Yes	Research	Modular, low-cost tool use
Toolken+	Train embeddings	✅ Yes	Research	More robust tool selection
Toolformer	Self-supervised	❌ No	Research	Multi-tool API calls
ReAct	No (prompting)	✅ Yes	Academic + Frameworks	Reasoning + acting
Function Calling	Light fine-tune	✅ Yes	Widely used	Structured API calls
Agent Frameworks	No	✅ Yes	Industry + OSS	Multi-step planning
RAG	No	✅ Yes	Very common	Knowledge injection
Code-as-Tool	No	✅ Yes	Niche use	Math, logic, execution

Name / Project	Type	Open Weights?	External Code Execution in Pipeline?	How Execution Works
LLaMA 2 / 3 (Meta)	Base/general LLM	✅	❌	Pure transformer inference
Falcon (TII)	Base/general LLM	✅	❌	Pure inference
MPT (MosaicML)	Base/general LLM	✅	❌	Pure inference
Mistral / Mixtral	Base/general LLM	✅	❌	Pure inference
OPT (Meta)	Base/general LLM	✅	❌	Pure inference
Pythia / GPT-NeoX / GPT-J (EleutherAI)	Base/general LLM	✅	❌	Pure inference
StarCoder / SantaCoder (BigCode)	Code LLM	✅	⚠️ Only if wrapped	Model itself just generates code; execution comes from eval harness or agent

smol-developer / GPT Engineer	Code agent framework	✅	✅	LLM generates code, runs locally, feeds errors/output back
OpenDevin	Agent framework	✅	✅	Executes Python + shell commands in controlled env
AutoGPT / BabyAGI / AgentGPT	Agent frameworks	✅	✅	Model proposes actions (Python, shell, API), host executes
LangChain Agents (open LLM backends)	Orchestration framework	✅	✅	Supports Python REPL, retrievers, APIs
LlamaIndex Agents	Orchestration framework	✅	✅	Similar: allows retrieval + code execution
Haystack Agents	Orchestration framework	✅	✅	Open agent system, Python + tool execution
Toolformer (reimplementations)	Research agent model	✅	✅	Augmented LLaMA-style models that can call APIs/tools
SymPy + LLM integrations (research demos)	Hybrid math agents	✅	✅	LLM delegates algebra/calculus to SymPy