PLAID: A Unified Data Model for Machine Learning on Heterogeneous Physics Simulations

Anonymous Authors[1]

Abstract

Machine learning-based surrogate models have emerged as a powerful tool to accelerate simulation-driven scientific workflows, but their adoption is limited by the lack of large-scale, diverse, and standardized datasets for physics-based simulations. Existing benchmarks often focus on narrow domains or rely on simplified data models, and fail to capture the heterogeneity arising from variable geometries, meshes, and topologies, which is critical for assessing generalization in realistic settings. We introduce PLAID (Physics-Learning AI Data model), a unified and extensible data layer for heterogeneous physics simulations. It preserves the full complexity of simulation data while enabling efficient and scalable machine learning workflows, together with a library for dataset construction and manipulation [LINK REMOVED]. We release six datasets covering structural mechanics and computational fluid dynamics, designed to reflect realistic industrial scenarios and provide standardized benchmarks. The framework includes reproducible evaluation protocols and is integrated with Hugging Face to enable open, community-driven benchmarking with active user participation [LINK REMOVED].

1. Introduction

Numerical simulation is central to scientific and engineering research, providing insights into complex physical phenomena across many domains (Zienkiewicz & Morice, 1971; Zienkiewicz & Taylor, 2005; Press, 2007; Samanidou et al., 2007; Viceconti & Hunter, 2016; Brotzge et al., 2023). These simulations typically rely on solving partial differential equations using large-scale numerical solvers, and are often computationally expensive, with single high-fidelity runs requiring hours or days. In many-query settings such as design exploration, optimization, or uncertainty quantification, this cost becomes prohibitive. Surrogate modeling techniques have therefore been developed to approximate simulation outputs at a fraction of the cost.

Classical surrogate models perform non-linear regression over parametric spaces using statistical learning techniques, such as polynomial regression, nearest neighbors, support vector machines, random forests (Breiman, 2001), and Gaussian processes (Williams & Rasmussen, 2006). These models are widely supported by software libraries such as UQLab (Marelli & Sudret, 2015), OpenTURNS (Baudin et al., 2016), Dakota (Adams et al., 2024) and Lagun (Da Veiga et al., 2025). However, they are typically restricted to low-dimensional, tabular parameter spaces and cannot be directly used in more complex simulation setups. In contrast, many modern applications involve richer input configurations, including unstructured meshes, spatially varying fields, and complex boundary or material conditions. These settings require learning from heterogeneous, high-dimensional data with nonparametric variability.

Recent advances in physics machine learning have begun to address these challenges. One line of work, often referred to as physics-based model reduction, builds surrogates that approximate the solution of the governing equations directly (Amsallem et al., 2015; Casenave et al., 2020; Daniel et al., 2020; Lee & Carlberg, 2020; Kim et al., 2022; Barral et al., 2024b). Other approaches have also been proposed using non-parametric methods based on the use of morphing (Casenave et al., 2024; Kabalan et al., 2025b;a) or optimal transport (Perez et al., 2024; 2025), and have the advantage of requiring a smaller number of design points. Deep learning methods have recently shown strong potential for modeling the spatiotemporal dynamics of physical systems. Graph Neural Networks (GNNs) build on the message-passing paradigm introduced by Gilmer et al. (2017): architectures such as MeshGraphNets (Pfaff et al., 2021) extend GNNs to general mesh-based simulations. Hierarchical versions like MultiScale MeshGraphNets (Fortunato et al., 2022) enhance scalability and accuracy, while recent works demonstrate effectiveness in inverse (Allen et al., 2022) and steady-state problems (Harsch & Riedelbauch, 2021). Other

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the AI4Physics Workshop at the International Conference on Machine Learning (AI4Physics@ICML 2026). Do not distribute.

developments include geodesic convolutions (Baque et al., 2018), multi-resolution models (Lino et al., 2021; 2022), and improved pooling strategies (Cao et al., 2023). More recently, transformer-based architectures have been investigated for physics-based learning, leveraging attention mechanisms to capture long-range dependencies and complex interactions across mesh-based representations (Dosovitskiy et al., 2020; Jiang et al., 2023b; Zhou et al., 2026). Tools such as PhysicsNeMo (PhysicsNeMo Contributors, 2023), PyTorch Geometric (Fey & Lenssen, 2019), and Deep Graph Library (Wang et al., 2019) provide convenient foundations for implementing these methods.

Despite these advances, widespread adoption remains hindered by a critical bottleneck: the lack of large-scale, diverse, and standardized datasets for training and benchmarking. Existing datasets often cover narrow physical regimes, rely on ad hoc formats, or are tied to specific libraries, limiting reusability and interoperability. Furthermore, many datasets are tailored to isolated challenges (e.g., time dependence) but fail to accommodate others (e.g., geometric variation). This fragmentation is particularly problematic in the context of recent developments in physics foundation models (Yang et al., 2023; McCabe et al., 2024a;b; Birk et al., 2024; Sun et al., 2025), which require large, flexible, and standardized sources of training data.

Contributions. To address these limitations, we present PLAID, a standardized data layer for physics simulation data that bridges simulation and machine learning workflows. It introduces a unified data model for heterogeneous mesh-based simulations, enabling interoperable datasets and a reproducible benchmark spanning multiple sources of heterogeneity:

- **Problem identification.** We identify heterogeneous simulation data (variable meshes, geometries, and topologies) as a key bottleneck in physics machine learning, limiting reliable evaluation and generalization.
- **Unified data model.** We introduce PLAID (Physics-Learning AI Data model), a generic, flexible, and extensible representation for physics simulations, supporting time-dependent problems, remeshing, topology variability, and mixed-element unstructured meshes.
- **Realistic dataset collection.** We release six datasets in structural mechanics and computational fluid dynamics, reflecting industrial scenarios with complex geometries and multi-source heterogeneity.
- **Reproducible benchmarking.** We provide a standardized evaluation protocol and benchmark results across multiple modeling paradigms (GNNs, operator learning, transformers), enabling fair comparison under heterogeneous conditions.
- **Open evaluation platform.** We extend the benchmark into a community-driven framework through online competitions hosted on Hugging Face, fostering continuous and scalable evaluation.

We provide an accompanying software library to facilitate dataset creation, reading, and high-level interaction, leveraging Hugging Face infrastructure for efficient streaming and sharing.

In Section 2, we review relevant dataset efforts in the literature. Section 3 introduces the PLAID data model and implementation, along with six publicly released datasets in structural mechanics and computational fluid dynamics, presented in Section 4, that showcase rich variability in physics and numerical complexity. In Section 5, we provide performance benchmarks across a range of machine learning methods, hosted on Hugging Face to allow community participation and continual updates. We conclude with perspectives in Section 6.

2. Related Work

Progress in machine learning has been largely driven by the availability of large, diverse, and carefully curated datasets (Achiam et al., 2023; Touvron et al., 2023; AI, 2025). Natural language processing models are trained on web-scale data (Gao et al., 2020; Penedo et al., 2023; Zhao, 2023; Liu et al., 2024), and vision models routinely leverage billions of image–text pairs (Schuhmann et al., 2022; Bai et al., 2023; Chen et al., 2024).

In contrast, datasets for physics learning remain comparatively underdeveloped. Early benchmarks targeted standard physics problems and reference simulations (Takamoto et al., 2022a; Gupta & Brandstetter, 2022b; Hao et al., 2023). More recent datasets have focused on complex, domain-specific settings (Hersbach et al., 2020; Bonnet et al., 2022; Kohl et al., 2023; Janny et al., 2023; Hassan et al., 2023; Chung et al., 2023b; Yu et al., 2023; Toshev et al., 2024; Ashton et al., 2024). The Well, introduced by Ohana et al. (2024), includes a large list of datasets for various physics, but is limited to structured grids (uniformly sampled domains).

Structural mechanics simulations, with non-linear constitutive laws, are of paramount importance for industrial design, and are poorly represented in available datasets. Most available datasets use a data model that limit their evolution and generality. Complex industrial settings include vertices and element tags, heterogeneous data with remeshing, multiple meshes of various dimensions, topologies and mixed element types, compatible with commercial codes routinely used by design engineers. Besides, most datasets come with a library dedicated to the dataset, featuring specific commands and hypothesis, which limit their wide adoption.

Comparison with existing datasets. Our goal is to support fair and reproducible surrogate modeling across heterogeneous simulation settings. Existing datasets are often limited in their ability to capture realistic industrial scenarios, including complex geometries, variable topologies, and heterogeneous data structures. To position PLAID within this landscape, we provide a comparison in Table 1, where "sample heterogeneity" refers to possible presence of multiple meshes per sample, with differing dimensions, topologies, element types and variable topology.

3. PLAID standard

PLAID introduces a unified data model for heterogeneous physics-based datasets, designed to support machine learning workflows and modern data pipelines. It also refers to the accompanying library that implements this model, available at [LINK REMOVED FOR ANONYMITY]. The primary goal of the library is to provide a general and flexible framework for defining physics-based datasets, along with a corresponding learning task. The data model is built on CGNS (Poinot & Rumsey, 2018), leveraging its well-established formalization of a wide range of physical configurations.

PLAID datasets are provided either in a human-readable format or through efficient backend storage such as Hugging Face Datasets (Lhoest et al., 2022) and Zarr (Miles et al., 2026). In the former case, YAML files can be opened with any text editor, while CGNS files containing physical configurations can be visualized using tools such as ParaView, see Figure 1. In the latter, we benefit from advanced data management capabilities, including online streaming and feature-wise access, allowing subsets of variables to be read without loading the entire dataset.

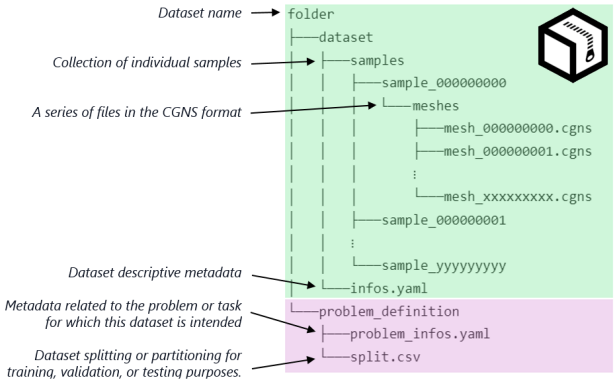


Figure 1. PLAID files structure (CGNS backend).

Additionally, PLAID offers high-level utilities for constructing, handling and parallel reading/writing datasets efficiently. Documentation is available [LINK REMOVED] with usage examples and tutorials showing how one can create a PLAID dataset from its own data. We also mention Muscat, a finite element toolbox available on GitLab (Bordeu et al., 2023; 2025), containing various reader and writers from and to various files formats used in numerical simulation codes for physics, and routines to generate the CGNS data structures used in PLAID. Samples can feature multiple meshes, scalars, fields and time series. To illustrate how PLAID handles such heterogeneous data, we highlight a few representative accessors:

- `sample.get_field_names(name, zone_name, base_name, location, time)`: retrieves a field identified by `name` within the specified `zone_name`, `base_name`, `location` (e.g., Vertex, CellCenter, FaceCenter), and `time` in the CGNS structure. Both fields and meshes may evolve over time, enabling support for remeshing and dynamic field appearance or disappearance.
- `sample.get_field(name)`: provides simplified access by automatically handling default values, avoiding the need to specify `zone_name`, `base_name`, and `location` in unambiguous cases.
- `sample.show_tree(time)`: returns a summary of the CGNS tree at the specified time step.

More examples are provided in Appendix C.

4. PLAID datasets

Among the datasets presented below, some (`Tensile2D`, `2D_MultiScHypEl`, `Rotor37` and `AirfRANS`) have been previously introduced in the literature, while others (`2D_ElPlDynamics`, `2D_profile`, and `VKI-LS59`) are newly released here. All are unified within the PLAID standard.

4.1. Dataset design rationale.

The datasets are designed to span increasing levels of complexity and realism, from controlled settings to industrial-scale simulations involving heterogeneous geometries, variable topology, and nonlinear physics. Table 2 summarizes the motivation behind each dataset in terms of machine learning challenges and industrial relevance.

All datasets involve variable geometries, a key aspect of industrial design, and are generated using simulation codes employed in industry (Z-set, OpenRadioss, elsA, and BROADCAST), together with constitutive laws for solids and turbulence models used in practice. For instance, `Rotor37` corresponds to the surface solution of a full 3D CFD simulation with complex physics. `VKI-LS59` reflects production-level CFD simulations and involves multiple geometrical supports of different dimensionality, while

Table 1. Comparison of dataset collections across key characteristics. * Geometrical variations addressed via density. "Mixed" denotes datasets spanning multiple physics families.

Dataset collection	Steady	Transient	2D+3D	Complex domains	Geometrical variations	Sample heterogeneity	Physics
Mech.-MNIST (Lejeune, 2020)	✓	✓	×	×	×	×	Solids
PDEArena (Gupta & Brandstetter, 2022a)	×	✓	✓	×	×	×	Fluids
BubbleML (Hassan et al., 2023)	×	✓	✓	×	×	×	Boiling
BLASTNet (Chung et al., 2023a)	✓	✓	✓	×	×	×	Fluids
PDEBench (Takamoto et al., 2022b)	✓	✓	✓	×	×	×	Fluids
The Well (Ohana et al., 2024)	×	✓	✓	✓	∼*	×	Mixed
PINNacle (Hao et al., 2024)	✓	✓	✓	✓	×	×	Mixed
PLAID (ours)	✓	✓	✓	✓	✓	✓	Mixed

2D_ElPlDynamics introduces time-dependent behavior with highly nonlinear phenomena such as damage, fracture, and non-local constitutive laws (see C.2 and C.3). This progression in complexity is essential to evaluate machine learning models beyond simplified settings and highlights the challenges posed by heterogeneous data in realistic scenarios.

4.2. Structural mechanics

4.2.1. TENSILE2D [LINK REMOVED]

Tensile2d is a simple dataset of 2D quasistatic non-linear structural mechanics simulations, in small deformations and plane strain regimes, solved with Z-set (Mines ParisTech and ONERA the French aerospace lab, 1981-present) using the finite element method. The material is modeled with a non-linear constitutive law. The dataset computes the deformation of a structure subjected to an imposed negative constant pressure at the top, and zero displacement at the bottom, see Figure 2 (left). Only the steady-state solution is kept.

Input variability in the dataset are the unstructured meshes (variable shape, number of nodes and connectivity), the pressure P at the top boundary condition (scalar) and 5 scalars modeling the non-linear constitutive law: (p1, p2, p3, p4 and p5). Outputs of interest are 4 scalars (max_von_mises, max_q, max_U2_top and max_sig22_top) and 6 fields (U1, U2, q, sig11, sig22 and sig12). Seven nested training sets are provided, as well as a testing set and two out-of-distribution samples.

4.2.2. 2D_MULTISCHYPEL [LINK REMOVED]

2D_MultiScHypEl, standing for 2D multiscale hyperelasticity, is a dataset of 2D quasistatic non-linear structural mechanics simulations under large deformation and plane strain conditions, solved with DOLFINx (Baratta et al., 2023) using the finite element method. The material behavior follows a compressible hyperelastic constitutive law, capturing complex non-linear responses. Each simulation corresponds to the homogenization of a porous representative volume element (RVE), subject to kinematically uniform boundary conditions (KUBC) (Yvonnet, 2019), see Figure 2 (right).

Input variability in the dataset are the unstructured meshes (variable shape, number of nodes, connectivity and topology–the number of circular inclusions) and the 3 scalars modeling the KUBC, namely the components C11, C12, and C22 of the macroscopic right Cauchy-Green deformation tensor. Outputs of interest are 1 scalar (effective_energy) and 7 fields (displacements u1, u2; first Piola-Kirchhoff stress components P11, P12, P22, P21 and the strain energy density field psi). Various training and testing sets are provided (both across all topologies and within each topology class).

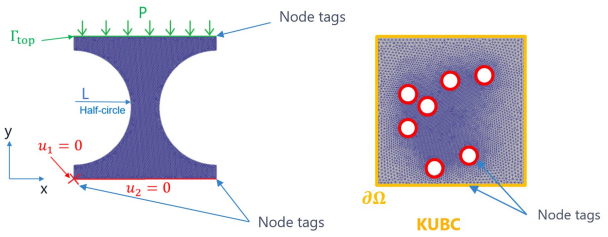


Figure 2. Tensile2d (left) and 2D_MultiScHypEl (right).

4.2.3. 2D_ELPLDYNAMICS [LINK REMOVED]

2D_ElPlDynamics, standing for 2D elasto-plasto dynamics, is a dataset of 2D dynamic non-linear structural mechanics simulations, in large deformations and plane strain regimes, solved with OpenRadioss (Community, 2022) using the finite element method. The material is modeled with a non-linear elastoplastic law, with damage (modeled using element erosion), failure and a non-local method for reducing mesh sensitivity. The dataset computes the transient deformation of a 2D structure, subjected to imposed displacement on the right and zero displacement on the left, see Figure 3 (left).

Table 2. Overview of PLAID datasets, associated machine learning challenges, and industrial relevance.

Dataset	ML Challenge	Industrial relevance
Tensile2d	Unstruct. mesh, variable size fields, nonlinear laws	Metallic materials with elastoviscoplastic behavior
2D_MultiScHypEl	Unstruct. mesh, variable size fields, variable topology	Representative volume elements for bi-materials
2D_ElPlDynamics	Time-dependent, topology variation, nonlin. materials	Structural integrity in extreme conditions
Rotor37	3D, shocks of variable position, nonlinear model	Design of rotors in compressors of rotating machinery
2D_profile	Variable size fields, variable shock positions, nonlin.	Design of wings and propellers
VKI-LS59	Variable shock classes, periodic setting, nonlin.	Design of rotors in turbines of rotating machinery

Input variability in the dataset are the unstructured meshes (variable shape, number of nodes, connectivity and topology). Outputs of interest are 3 fields (U_x and U_y the displacement fields at the nodes, and EROSION_STATUS a boolean field at the elements describing the state – valid or broken – of each element). A training and a testing set are provided.

4.3. Computational fluid dynamics

4.3.1. Rotor37 [LINK REMOVED]

Rotor37 is a dataset of 3D compressible steady-state Reynolds-Averaged Navier-Stokes (RANS) simulations, solved with elsA (Cambier, Laurent et al., 2013) using the finite volume method. Large scale simulations around the rotor37 blade inside a 3D duct have been computed, with inflow, outflow and periodic boundary conditions. An adequate turbulence model and laws of the wall have been chosen. The dataset only keeps the steady-state solution at the boundary of the blade inside the duct, and scalars of interest, see Figure 3 (right).

Input variability in the dataset are the block-structured anisotropic meshes (variable shape, normals at the blade surface are provided) and 2 scalars (the pressure P and the rotation speed Omega of the blade). Outputs of interest are 3 scalars (Massflow, Compression_ratio and Efficiency) and 3 fields (Density, Pressure, Temperature). Eight nested training sets and a testing set are provided.

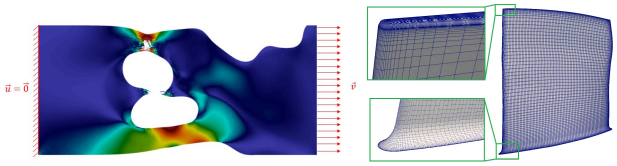


Figure 3. 2D_ElPlDynamics (left) and Rotor37 (right).

4.3.2. 2D_PROFILE [LINK REMOVED]

2D_profile is a dataset of 2D compressible steady-state Reynolds-Averaged Navier-Stokes (RANS) simulations, solved with elsA (Cambier, Laurent et al., 2013) using the finite volume method. The flow is computed around 2D profiles, which present large deformation around shapes resembling airfoils or propeller blades, on a large refined meshes, with inflow, outflow and periodic boundary conditions, at a transonic regime. An adequate turbulence model and laws of the wall have been chosen. The dataset only keeps the steady-state solution on a zone cropped close to the profile, see Figure 4 (left).

Input variability in the dataset are the unstructured anisotropic meshes (variable shape, number of nodes and connectivity). Outputs of interest are 4 fields (Mach, Pressure, Velocity-x and Velocity-y). A training and a testing set are provided.

4.3.3. VKI-LS59 [LINK REMOVED]

VKI-LS59 is a dataset of 2D compressible steady-state Reynolds-Averaged Navier-Stokes (RANS) simulations, solved with BROADCAST (Poulain et al., 2023) using the finite volume method with high-order corrections. The flow is computed around the VKI-LS59 blade, with inflow, outflow and periodic boundary conditions. A Spalart-Allmaras turbulence model has been chosen, see Figure 4 (right).

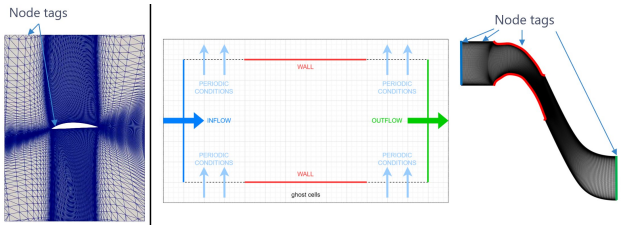


Figure 4. 2D_profile (left) and VKI-LS59 (right).

Input variability in the dataset are the block-structured anisotropic meshes (variable shape, number of nodes and connectivity, the distance field to the blade surface is provided) and 2 scalars (angle_in and mach_out). Outputs of interest are 6 scalars (Q, power, Pr, Tr, eth_is and angle_out) and 7 fields (ro, rou, rov, roe, nut, mach and M_iso – this last being only defined at the surface of the blade). Eight nested training sets are provided, as well as a testing set.

4.3.4. AirfRANS (Bonnet et al., 2022)

AirfRANS is a dataset of external aerodynamics, featuring steady-state Reynolds-Averaged Navier-Stokes (RANS) simulations over airfoils at a subsonic regime, proposed by Bonnet et al. (2022), which we refer to for a detailed description. In addition to the six original datasets, we provide three variants of AirfRANS in PLAID format [LINKS REMOVED].

Input variability in the dataset are the anisotropic meshes (variable shape, number of nodes and connectivity, the distance field to the airfoil surface is provided) and 2 scalars (angle_of_attack and inlet_velocity). Outputs of interest are 2 scalars (C_D and C_L) and 4 fields (nut, Ux, Uy and p). Various training and testing sets are provided.

4.4. Dataset collection

The collection of proposed datasets is publicly available through dedicated communities on Zenodo and Hugging Face [LINKS REMOVED]. It covers a diverse set of challenging scenarios, including heterogeneous meshes, variable topology, structural mechanics and CFD applications, and time-dependent problems. While the collection can be further extended, it already provides a practical basis for benchmarking surrogate models in realistic industrial settings.

Since these datasets are intended as open benchmarks for the community, outputs on the test sets are not released; instead, evaluation tools are provided to compute scores. We provide high-level descriptions of the underlying physical models and assumptions in Tables 3 and 4, while avoiding full disclosure of simulation details that would enable reconstruction of the high-fidelity solvers and compromise the integrity of the benchmark. Some field outputs are illustrated in Table 5.

5. Benchmark

We mention that we do not provide benchmark tools and results for the AirfRANS datasets, since outputs are public on the testing sets, and various benchmarks have already been reported by Casenave et al. (2024) and Bonnet et al. (2022), as well as in a competition at NeurIPS 2024 (Yagoubi et al., 2025).

5.1. Methods

We focus the benchmark on a representative set of widely used methods, covering graph-based, operator-learning, and transformer-based approaches, in order to provide a meaningful comparison across distinct modeling paradigms:

- MeshGraphNets (MGNs) (Pfaff et al., 2021) are graph neural networks that utilize an encode-process-decode architecture, transforming mesh data into graph structures, processing them through message passing, and decoding the results to predict field outputs.
- Mesh Morphing Gaussian Processes (MMGP) (Casenave et al., 2024) rely on mesh morphing, finite element interpolation and dimensionality reduction to pretreat mesh-based data into a low dimensional embedding, and utilizes Gaussian processes to predict output scalars and fields.
- Vi-Transformer (Dosovitskiy et al., 2021a) and Augur [1] rely on mesh partitioning to build tokens related to local mesh information and utilize a transformer to predict scalar and field outputs.
- Fourier Neural Operators (FNOs) (Li et al., 2020) learn mappings between functions by operating in the Fourier domain.
- Modulated Aerodynamic Resolution Invariant Operator (MARIO), introduced by Catalani et al. (2025), builds upon Catalani et al. (2024) and exploits implicit neural representations, which model continuous signals by mapping input coordinates directly to output values, without relying on discrete grids or explicit storage.

For more details on the methods and their respective advantages/drawbacks, refer to Appendix B.

5.2. Evaluation metric

Accuracy of the trained models is evaluated by computing RRMSEs (Relative Root Mean Square Errors). Let $\{\mathbf{f}^i_{\text{ref}}\}_{i=1}^{n_\star}$ and $\{\mathbf{f}^i_{\text{pred}}\}_{i=1}^{n_\star}$ be respectively the reference and prediction of a field output on the testing set. The RRMSE is defined as

$$\text{RRMSE}_f(\mathbf{f}_{\text{ref}}, \mathbf{f}_{\text{pred}}) = \left(\frac{1}{n_\star} \sum_{i=1}^{n_\star} \frac{\frac{1}{N^i} \|\mathbf{f}^i_{\text{ref}} - \mathbf{f}^i_{\text{pred}}\|_2^2}{\|\mathbf{f}^i_{\text{ref}}\|_\infty^2} \right)^{1/2},$$

where N^i is the number of nodes in the mesh of sample i, $n_\star$ is the number of samples in the testing set, and $\|\mathbf{f}^i_{\text{ref}}\|_\infty$ is the maximum component in the vector $\mathbf{f}^i_{\text{ref}}$. Similarly for scalar outputs, the following relative RMSE is computed:

$$\text{RRMSE}_s(\mathbf{s}_{\text{ref}}, \mathbf{s}_{\text{pred}}) = \left(\frac{1}{n_\star} \sum_{i=1}^{n_\star} \frac{|s^i_{\text{ref}} - s^i_{\text{pred}}|^2}{|s^i_{\text{ref}}|^2} \right)^{1/2}.$$

The score of a submission, total_error, is the mean over fields and scalars RRMSEs.

[1] Augur is a commercial solution.

Table 3. Dataset collection description: model and simulation volume.

Dataset	Simulation code	Model	Nb samples	Volume Zenodo	Volume HF
Tensile2d	Z-set	2D quasistatic non-linear structural mechanics, small deformations, non-linear constitutive law	702	290 MB	383 MB
2D_MultiScHypEl	DOLFINx	2D quasistatic non-linear structural mechanics, finite elasticity	1,140	350 MB	419 MB
2D_ElPlDynamics	OpenRadioss	2D dynamic non-linear structural mechanics, non-linear non-local constitutive law	1,018	5.7 GB	8.6 GB
Rotor37	elsA	3D Navier-Stokes (RANS)	1,200	3.3 GB	4.0 GB
2D_profile	elsA	2D Navier-Stokes (RANS)	400	660 MB	814 MB
VKI-LS59	BROADCAST	2D Navier-Stokes (RANS)	839	1.9 GB	2.3 GB
AirfRANS original	OpenFOAM	2D Navier-Stokes (RANS)	1,000	9.3 GB	15.6 GB
AirfRANS clipped				9.7 GB	18.2 GB
AirfRANS remeshed				520 MB	611 MB

Table 4. Dataset collection description: data and splits, a $*$ in the second column means that the number of nodes and connectivity are constant in the dataset – the position of the nodes still varies.

Dataset	Mesh (mean nodes)	Inputs	Outputs	Splits (train/test)
Tensile2d	tri (9,428)	mesh, 6 scalars	4 scalars, 6 fields	500 / 200
2D_MultiScHypEl	tri (5,692)	mesh, 3 scalars	1 scalar, 7 fields	764 / 376
2D_ElPlDynamics	tri (25,429)	mesh	3 fields	1,000 / 18
Rotor37	quad (29,773*)	mesh, 2 scalars	4 scalars, 3 fields	1,000 / 200
2D_profile	tri (37,042)	mesh	4 fields	300 / 100
VKI-LS59	quad (36,421*)	mesh, 2 scalars	6 scalars, 7 fields	671 / 168
AirfRANS original	quad (179,776)	mesh, 2 fields	2 scalars, 4 fields	various splits
AirfRANS clipped	tri (179,779)			
AirfRANS remeshed	tri (7,624)			

We mention that scalar quantities correspond to application-specific performance indicators (e.g., efficiency, stress extrema), ensuring that the evaluation reflects domain-relevant criteria.

5.3. Benchmark results

The global `total_error` for each method and dataset is reported in Table 6, while all individual RRMSE metrics are provided in Table 7 in Annex A. These results define a fixed and fully reproducible reference benchmark, based on standardized datasets fixed data splits, evaluation protocols, and implementations publicly available at submission time.

Beyond the fixed benchmark defined in this work, we introduce online benchmarking applications hosted on Hugging Face as competitions with no end date, see [LINK REMOVED]. These applications extend the benchmark into a long-term, community-driven evaluation framework, while preserving the fixed reference defined in this paper. Each benchmark comes with a visualization interface, detailed descriptions of inputs and outputs, and instructions for accessing the data and constructing prediction files. Anyone can register and submit predictions, which are automatically ranked based on `total_error` as defined in Section 5.2. See Section D for additional details. The platform is already active, with over 70 community submissions at the time of writing, indicating early adoption by the community.

MMGP does not support variable mesh topologies, limiting its applicability to `2D_MultiScHypEl` and `2D_ElPlDynamics`, and often requiring custom preprocessing. However, when morphing is inexpensive, it remains efficient with good accuracy (e.g., `Tensile2d`, `Rotor37`). MARIO achieves strong performance across datasets, with slightly lower results on `2D_MultiScHypEl`. Vi-Transformer and Augur show consistent performance across all datasets. FNO degrades on unstructured or anisotropic meshes due to projections to regular grids, and incurs significant overhead on `Rotor37`.

These results consistently show that methods operating directly on meshes outperform grid-based approaches under heterogeneous settings, highlighting data heterogeneity as a key driver of current limitations.

6. Conclusion and perspectives

PLAID provides a unified data layer for heterogeneous physics simulation data, preserving the complexity of raw simulations while enabling interoperable datasets and re-

Table 5. Dataset collection examples of field outputs illustrations.

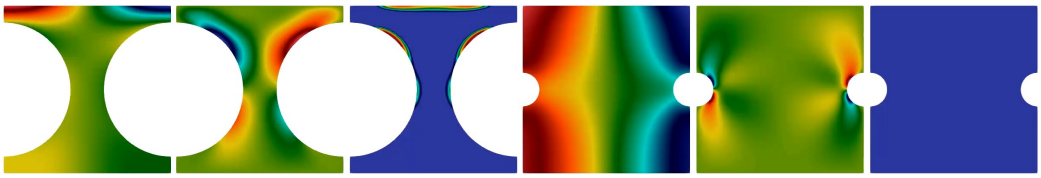
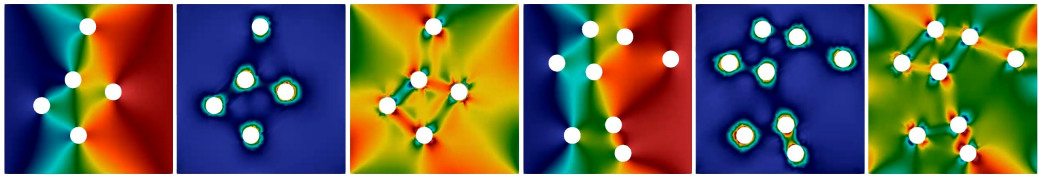
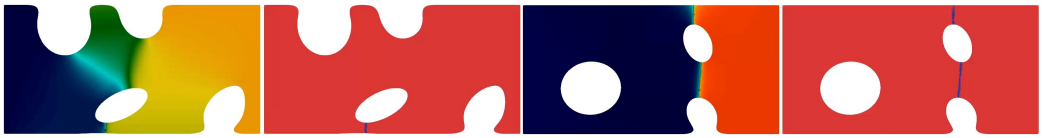
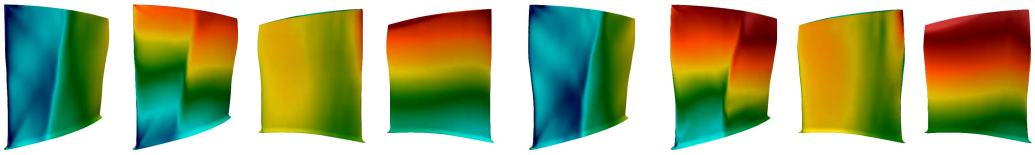
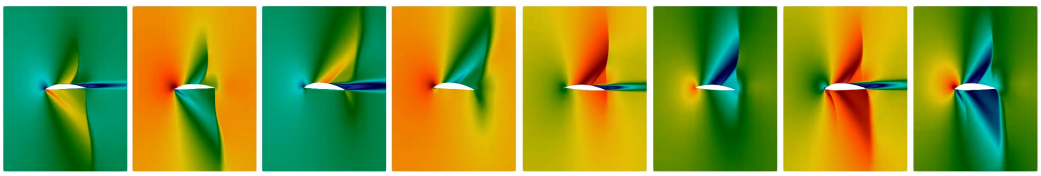
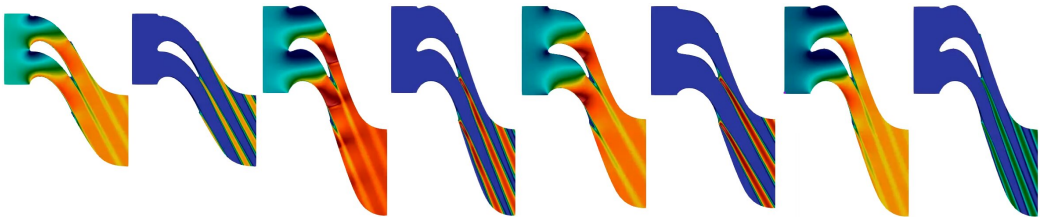
Dataset	Examples of field outputs
Tensile2d	
2D_MultiScHypEl	
2D_ElPlDynamics	
Rotor37	
2D_profile	
VKI-LS59	

Table 6. total_error on PLAID benchmarks, best on each line is **bold**, second best is underlined. Missing entries for MMGP correspond to non-applicable settings.

total_error	MGN	MMGP	Vi-Transf.	Augur	FNO	MARIO
Tensile2d	0.0673	**0.0026**	0.0116	0.0154	0.0123	0.0038
2D_MultiScHypEl	0.0437	-	0.0325	**0.0232**	0.0302	0.0573
2D_ElPlDynamics	0.1202	-	0.0227	0.0346	**0.0215**	0.0319
Rotor37	0.0074	**0.0014**	0.0029	0.0033	0.0313	0.0017
2D_profile	0.0593	0.0365	0.0312	0.0425	0.0972	**0.0307**
VKI-LS59	0.0684	0.0312	0.0193	0.0267	0.0215	**0.0124**

producible evaluation under realistic industrial conditions. By unifying diverse datasets, storage backends, and benchmark protocols, PLAID turns heterogeneous simulations (with variable geometries, meshes, topologies, time dependence, and physical regimes) into reusable machine learning benchmarks. This work highlights that progress in scientific machine learning requires not only improved architectures, but also standardized data representations, reusable data infrastructure, and community-driven evaluation protocols.

Impact Statement

This work aims to advance machine learning for scientific and engineering simulation by improving dataset standardization, reproducible benchmarking, and interoperability across simulation and learning workflows. Its potential positive impacts include more transparent evaluation of surrogate models and broader reuse of physics simulation data. Possible risks include misuse of surrogate models outside their validated regimes, overreliance on benchmark scores, and unintended disclosure of sensitive industrial simulation data when releasing datasets. These risks are mitigated here by focusing on benchmark protocols, documented data representations, and controlled release of test outputs.

Software and Data

All code, datasets, and benchmark entry scripts are finalized. For anonymity, access links are removed in this submission and will be provided upon acceptance.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Adams, B. M., Bohnhoff, W. J., Dalbey, K. R., Ebeida, M. S., Eddy, J. P., Eldred, M. S., Hooper, R. W., Hough, P. D., Hu, K. T., Jakeman, J. D., Khalil, M., Maupin, K. A., Monschke, J. A., Prudencio, E. E., Ridgway, E. M., Robbe, P., Rushdi, A. A., Seidl, D. T., Stephens, J. A., Swiler, L. P., and Winokur, J. G. Dakota 6.21.0 documentation. Technical Report SAND2024-15492O, Sandia National Laboratories, Albuquerque, NM, November 2024. Available online from http://snl-dakota.github.io.

AI, M. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation, 2025. https://ai.meta.com/blog/llama-4-multimodal-intelligence.

Allen, K., Lopez-Guevara, T., Stachenfeld, K., Sanchez-Gonzalez, A., Battaglia, P., Hamrick, J., and Pfaff, T. Physical design using differentiable learned simulators. *arXiv preprint arXiv:2202.00728*, 2022.

Amsallem, D., Zahr, M., Choi, Y., and Farhat, C. Design optimization using hyper-reduced-order models. *Structural and Multidisciplinary Optimization*, 51(4):919–940, 2015.

Ashton, N., Mockett, C., Fuchs, M., Fliessbach, L., Hetmann, H., Knacke, T., Schonwald, N., Skaperdas, V., Fotiadis, G., Walle, A., Hupertz, B., and Maddix, D. Drivaerml: High-fidelity computational fluid dynamics dataset for road-car external aerodynamics, 2024.

Bai, H., Mou, S., Likhomanenko, T., Cinbis, R. G., Tuzel, O., Huang, P., Shan, J., Shi, J., and Cao, M. Vision datasets: A benchmark for vision-based industrial inspection, 2023.

Baque, P., Remelli, E., Fleuret, F., and Fua, P. Geodesic convolutional shape optimization. In *International Conference on Machine Learning*, pp. 472–481. PMLR, 2018.

Baratta, I. A., Dean, J. P., Dokken, J. S., Habera, M., Hale, J. S., Richardson, C. N., Rognes, M. E., Scroggs, M. W., Sime, N., and Wells, G. N. DOLFINx: the next generation FEniCS problem solving environment. preprint, 2023.

Barral, N., Taddei, T., and Tifouti, I. Registration-based model reduction of parameterized PDEs with spatio-parameter adaptivity. *Journal of Computational Physics*, 499:112727, 2024a. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2023.112727.

Barral, N., Taddei, T., and Tifouti, I. Registration-based model reduction of parameterized PDEs with spatio-parameter adaptivity. *Journal of Computational Physics*, 499:112727, 2024b.

Baudin, M., Dutfoy, A., Iooss, B., and Popelin, A.-L. *OpenTURNS: An Industrial Software for Uncertainty Quantification in Simulation*, pp. 1–38. Springer International Publishing, Cham, 2016. ISBN 978-3-319-11259-6. doi: 10.1007/978-3-319-11259-6_64-1.

Birk, J., Hallin, A., and Kasieczka, G. Omnijet-α: the first cross-task foundation model for particle physics. *Machine Learning: Science and Technology*, 5(3):035031, aug 2024. doi: 10.1088/2632-2153/ad66ad.

Bonnet, F., Mazari, J., Cinnella, P., and Gallinari, P. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35: 23463–23478, 2022.

Bordeu, F., Casenave, F., and Cortial, J. Basictools: a numerical simulation toolbox. *Journal of Open Source Software*, 8(86):5142, 2023. doi: 10.21105/joss.05142.

Bordeu, F. et al. MUSCAT: Mesh Utilities and Solver for Computational Analysis Toolkit. https://gitlab.com/drti/muscat, 2025.

Breiman, L. Random forests. *Machine learning*, 45:5–32, 2001.

Brotzge, J. A., Berchoff, D., Carlis, D. L., Carr, F. H., Carr, R. H., Gerth, J. J., Gross, B. D., Hamill, T. M., Haupt, S. E., Jacobs, N., et al. Challenges and opportunities in numerical weather prediction. *Bulletin of the American Meteorological Society*, 104(3):E698–E705, 2023.

Cambier, Laurent, Heib, Sébastien, and Plot, Sylvie. The onera elsa cfd software: input from research and feedback from industry. *Mechanics & Industry*, 14(3):159–174, 2013. doi: 10.1051/meca/2013056.

Cao, Y., Chai, M., Li, M., and Jiang, C. Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network. 2023.

Casenave, F., Akkari, N., Bordeu, F., Rey, C., and Ryckelynck, D. A nonintrusive distributed reduced-order modeling framework for nonlinear structural mechanics—application to elastoviscoplastic computations. *International Journal for Numerical Methods in Engineering*, 121(1):32–53, 2020.

Casenave, F., Staber, B., and Roynard, X. MMGP: a Mesh Morphing Gaussian Process-based machine learning method for regression of physical problems under nonparametrized geometrical variability. *Advances in Neural Information Processing Systems*, 36, 2024.

Catalani, G., Agarwal, S., Bertrand, X., Tost, F., Bauerheim, M., and Morlier, J. Neural fields for rapid aircraft aerodynamics simulations. *Scientific Reports*, 14(1):25496, 2024.

Catalani, G., Fesquet, J., Bertrand, X., Tost, F., Bauerheim, M., and Morlier, J. Towards scalable surrogate models based on neural fields for large scale aerodynamic simulations, 2025.

Chen, T.-S., Siarohin, A., Menapace, W., Deyneka, E., wei Chao, H., Jeon, B. E., Fang, Y., Lee, H.-Y., Ren, J., Yang, M.-H., and Tulyakov, S. Panda-70m: Captioning 70m videos with multiple cross-modality teachers, 2024.

Chung, W. T., Akoush, B., Sharma, P., Tamkin, A., Jung, K. S., Chen, J., Guo, J., Brouzet, D., Talei, M., Savard, B., et al. Turbulence in focus: Benchmarking scaling behavior of 3d volumetric super-resolution with blastnet 2.0 data. *Advances in Neural Information Processing Systems*, 36:77430–77484, 2023a.

Chung, W. T., Akoush, B., Sharma, P., Tamkin, A., Jung, K. S., Chen, J. H., Guo, J., Brouzet, D., Talei, M., Savard, B., Poludnenko, A. Y., and Ihme, M. Turbulence in focus: Benchmarking scaling behavior of 3D volumetric super-resolution with BLASTNet 2.0 data. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023b.

Community, O. Openradioss: Open-source finite element solver for dynamic event analysis, 2022. https://openradioss.org/.

Da Veiga, S., Bénard, C., Gonon, T., et al. Lagun. https://gitlab.com/drti/lagun, 2025.

Dai, X., Chalkidis, I., Darkner, S., and Elliott, D. Revisiting transformer-based models for long document classification, 2022.

Daniel, T., Casenave, F., Akkari, N., and Ryckelynck, D. Model order reduction assisted by deep neural networks (rom-net). *Advanced Modeling and Simulation in Engineering Sciences*, 7:1–27, 2020.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021a.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021b.

Durairaj, A. K. and Chinnalagu, A. Transformer based contextual model for sentiment analysis of customer reviews: A fine-tuned bert. *International Journal of Advanced Computer Science and Applications*, 12(11), 2021. doi: 10.14569/IJACSA.2021.0121153.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Fortunato, M., Pfaff, T., Wirnsberger, P., Pritzel, A., and Battaglia, P. Multiscale meshgraphnets. *arXiv preprint arXiv:2210.00612*, 2022.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Geelen, R., Balzano, L., and Willcox, K. Learning latent representations in high-dimensional state spaces using polynomial manifold constructions. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 4960–4965. IEEE, 2023.

Gilmer, J., Schoenholz, S., Riley, P., Vinyals, O., and Dahl, G. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Gupta, J. K. and Brandstetter, J. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022a.

Gupta, J. K. and Brandstetter, J. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022b.

Hao, Z., Yao, J., Su, C., Su, H., Wang, Z., Lu, F., Xia, Z., Zhang, Y., Liu, S., Lu, L., et al. PINNacle: A Comprehensive Benchmark of Physics-Informed Neural Networks for Solving PDEs. *arXiv preprint arXiv:2306.08827*, 2023.

Hao, Z., Yao, J., Su, C., Su, H., Wang, Z., Lu, F., Xia, Z., Zhang, Y., Liu, S., Lu, L., et al. Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes. *Advances in Neural Information Processing Systems*, 37:76721–76774, 2024.

Harsch, L. and Riedelbauch, S. Direct prediction of steady-state flow fields in meshed domain with graph networks. *arXiv preprint arXiv:2105.02575*, 2021.

Hassan, S. M. S., Feeney, A., Dhruv, A., Kim, J., Suh, Y., Ryu, J., Won, Y., and Chandramowlishwaran, A. BubbleML: A multi-physics dataset and benchmarks for machine learning. In *Advances in Neural Information Processing Systems*, 2023.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730): 1999–2049, 2020.

Jamal, S. and Wimmer, H. An improved transformer-based model for detecting phishing, spam, and ham: A large language model approach, 2023.

Janny, S., Benetteau, A., Nadri, M., Digne, J., Thome, N., and Wolf, C. Eagle: Large-scale learning of turbulent fluid dynamics with mesh transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Jiang, J., Li, G., Jiang, Y., Zhang, L., and Deng, X. Transcfd: A transformer-based decoder for flow field prediction. *Engineering Applications of Artificial Intelligence*, 123: 106340, 2023a. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2023.106340.

Jiang, J., Li, G., Jiang, Y., Zhang, L., and Deng, X. Transcfd: A transformer-based decoder for flow field prediction. *Engineering Applications of Artificial Intelligence*, 123: 106340, 2023b.

Kabalan, A., Casenave, F., Bordeu, F., Ehrlacher, V., and Ern, A. Morphing techniques for model order reduction with non parametric geometrical variabilities. In *16ème Colloque National en Calcul de Structures*, 2024.

Kabalan, A., Casenave, F., Bordeu, F., and Ehrlacher, V. O-MMGP: Optimal Mesh Morphing Gaussian Process regression for solving PDEs with non-parametric geometric variations. *arXiv preprint arXiv:2502.11632*, 2025a.

Kabalan, A., Casenave, F., Bordeu, F., Ehrlacher, V., and Ern, A. Elasticity-based morphing technique and application to reduced-order modeling. *Applied Mathematical Modelling*, 141:115929, 2025b. ISSN 0307-904X. doi: https://doi.org/10.1016/j.apm.2025.115929.

Karypis, G. and Kumar, V. Metis – unstructured graph partitioning and sparse matrix ordering system, version 2.0. 01 1995.

Kim, Y., Choi, Y., Widemann, D., and Zohdi, T. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110841.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks, 2017.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kohl, G., Chen, L., and Thuerey, N. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation. *arXiv*, 2023. doi: 10.48550/arXiv.2309.01745.

Lee, K. and Carlberg, K. T. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404: 108973, 2020.

Lejeune, E. Mechanical mnist: A benchmark dataset for mechanical metamodels. *Extreme Mechanics Letters*, 36: 100659, 2020.

Lhoest, Q., del Moral, A. V., von Platen, P., Wolf, T., Šaško, M., Jernite, Y., Thakur, A., Tunstall, L., Patil, S., Drame, M., Chaumond, J., Plu, J., Davison, J., Brandeis, S., Sanh, V., Scao, T. L., Xu, K. C., Patry, N., Liu, S., McMillan-Major, A., Schmid, P., Gugger, S., Raw, N., Lesage, S., Lozhkov, A., Carrigan, M., Matussière, T., von Werra, L., Debut, L., Bekman, S., and Delangue, C. huggingface/datasets: 2.8.0, December 2022.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Lino, M., Cantwell, C., Bharath, A., and Fotiadis, S. Simulating continuum mechanics with multi-scale graph neural networks. *arXiv preprint arXiv:2106.04900*, 2021.

Lino, M., Fotiadis, S., Bharath, A., and Cantwell, C. Multi-scale rotation-equivariant graph neural networks for unsteady eulerian fluid dynamics. *Physics of Fluids*, 34(8), 2022.

Liu, Y., Cao, J., Liu, C., Ding, K., and Jin, L. Datasets for large language models: A comprehensive survey, 2024.

Marelli, S. and Sudret, B. *UQLab: A Framework for Uncertainty Quantification in Matlab*, pp. 2554–2563. 2015. doi: 10.1061/9780784413609.257.

McCabe, M., Blancard, B. R.-S., Parker, L. H., Ohana, R., Cranmer, M., Bietti, A., Eickenberg, M., Golkar, S., Krawezik, G., Lanusse, F., Pettee, M., Tesileanu, T., Cho, K., and Ho, S. Multiple physics pretraining for physical surrogate models, 2024a.

McCabe, M., Régaldo-Saint Blancard, B., Parker, L., Ohana, R., Cranmer, M., Bietti, A., Eickenberg, M., Golkar, S., Krawezik, G., Lanusse, F., Pettee, M., Tesileanu, T., Cho, K., and Ho, S. Multiple physics pretraining for spatiotemporal surrogate models. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 119301–119335. Curran Associates, Inc., 2024b.

Miles, A., jakirkham, Bennett, D., Stansby, D., Orfanos, D. P., Hamman, J., Bussonnier, M., Moore, J., Jones, M., Augspurger, T., Cherian, D., Rzepka, N., Spitz, H., Verma, S., Bourbeau, J., Fulton, A., Abernathey, R., Lee, G., Kristensen, M. R. B., Patel, Z., Gold, I., Chopra, S., Hunt-Isaak, I., Rocklin, M., Zimmerman, N., AGHANGU, A. D., Chai, C. P., and de Andrade, E. S. zarr-developers/zarr-python: v3.1.6, March 2026. URL https://doi.org/10.5281/zenodo.19121107.

Mines ParisTech and ONERA the French aerospace lab. Zset: nonlinear material & structure analysis suite. http://www.zset-software.com, 1981-present.

Ohana, R., McCabe, M., Meyer, L., Morel, R., Agocs, F., Beneitez, M., Berger, M., Burkhart, B., Dalziel, S., Fielding, D., et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37: 44989–45037, 2024.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023.

Perez, R. C., Da Veiga, S., Garnier, J., and Staber, B. Gaussian process regression with Sliced Wasserstein Weisfeiler-Lehman graph kernels. In *International Conference on Artificial Intelligence and Statistics*, pp. 1297–1305. PMLR, 2024.

Perez, R. C., da Veiga, S., Garnier, J., and Staber, B. Learning signals defined on graphs with optimal transport and Gaussian process regression, 2025.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

PhysicsNeMo Contributors. Nvidia physicsnemo: An open-source framework for physics-based deep learning in science and engineering. https://github.com/NVIDIA/physicsnemo, February 2023.

Poinot, M. and Rumsey, C. L. Seven keys for practical understanding and use of cgns. In *2018 AIAA Aerospace Sciences Meeting*, pp. 1503, 2018.

Poulain, A., Content, C., Sipp, D., Rigas, G., and Garnier, E. Broadcast: A high-order compressible cfd toolbox for stability and sensitivity using algorithmic differentiation. *Computer Physics Communications*, 283:108557, 2023.

Press, W. H. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

Samanidou, E., Zschischang, E., Stauffer, D., and Lux, T. Agent-based models of financial markets. *Reports on Progress in Physics*, 70(3):409, 2007.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35: 25278–25294, 2022.

Sun, J., Liu, Y., Zhang, Z., and Schaeffer, H. Towards a foundation model for partial differential equations: Multioperator learning and extrapolation. *Phys. Rev. E*, 111: 035304, Mar 2025. doi: 10.1103/PhysRevE.111.035304.

Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., and Niepert, M. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35: 1596–1611, 2022a.

Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., and Niepert, M. Pdebench: An extensive benchmark for scientific machine learning. *Advances in neural information processing systems*, 35: 1596–1611, 2022b.

Toshev, A., Galletti, G., Fritz, F., Adami, S., and Adams, N. Lagrangebench: A lagrangian fluid mechanics benchmarking suite. *Advances in Neural Information Processing Systems*, 36, 2024.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. LLaMA: Open and efficient foundation language models, 2023.

Viceconti, M. and Hunter, P. The virtual physiological human: ten years after. *Annual review of biomedical engineering*, 18(1):103–123, 2016.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Williams, C. and Rasmussen, C. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Wu, X., Jiang, L., Wang, P.-S., Liu, Z., Liu, X., Qiao, Y., Ouyang, W., He, T., and Zhao, H. Point transformer v3: Simpler, faster, stronger, 2024.

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., and Luo, P. Segformer: Simple and efficient design for semantic segmentation with transformers, 2021.

Yagoubi, M., Danan, D., Leyli-Abadi, M., Mazari, A., Brunet, J.-P., Kabalan, A., Casenave, F., Ma, Y., Catalani, G., Fesquet, J., et al. Neurips 2024 ML4CFD competition: Results and retrospective analysis. *arXiv preprint arXiv:2506.08516*, 2025.

Yang, L., Liu, S., Meng, T., and Osher, S. J. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39), September 2023. ISSN 1091-6490. doi: 10.1073/pnas.2310142120.

Yu, S., Hannah, W., Peng, L., Lin, J., Bhouri, M. A., Gupta, R., Lütjens, B., Will, J. C., Behrens, G., Busecke, J., et al. Climsim: A large multi-scale dataset for hybrid physics-ml climate emulation. *Advances in neural information processing systems*, 36:22070–22084, 2023.

Yvonnet, J. *Computational homogenization of heterogeneous materials with finite elements*, volume 258. Springer, 2019.

Zhao, H., Jiang, L., Jia, J., Torr, P., and Koltun, V. Point transformer, 2021.

Zhao, J. LLMDataHub: Awesome Datasets for LLM Training, 2023. https://github.com/Zjh-819/LLMDataHub.

Zhou, H., Wu, H., Shangguan, H., Ma, Y., Weng, H., Wang, J., and Long, M. Transolver-3: Scaling up transformer solvers to industrial-scale geometries. *arXiv preprint arXiv:2602.04940*, 2026.

Zienkiewicz, O. C. and Morice, P. *The finite element method in engineering science*, volume 1977. McGraw-hill London, 1971.

Zienkiewicz, O. C. and Taylor, R. L. *The finite element method for solid and structural mechanics*. Elsevier, 2005.

Zintgraf, L. M., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S. Fast context adaptation via meta-learning, 2019.

A. Detailed benchmark results

Detailed individual metrics for all methods and all datasets are provided in Table 7.

B. Details on the ML models used in the benchmark

We briefly present the main competing methods used in the benchmark, together with practical details about their implementation. Readers are encouraged to refer to the original publications for a complete description of each method.

All methods are trained on the same dataset splits and evaluated using the standardized PLAID benchmark protocol described in Section 5. Full implementation details, including complete hyperparameter configurations, training scripts, and data preprocessing pipelines, are provided in the accompanying code repository [LINK REMOVED], ensuring full reproducibility of all experiments (except for the commercial solution from Augur).

We briefly highlight how each method handles data heterogeneity, which is a central challenge addressed in this work.

B.1. MGN

MeshGraphNet (MGN) operates directly on mesh-based representations, enabling it to naturally handle geometric variability and unstructured discretizations.

B.1.1. Method

Pfaff et al. (2021) introduced MGN, a framework designed for learning mesh-based simulations using graph neural networks. The model is capable of being trained to simulate dynamic solutions by passing messages over a meshed domain, predicting acceleration at each mesh node at a given time step. This prediction allows for the calculation of the output field at the next time step through forward integration. Specifically, MGN is trained using one-step supervision and can be applied iteratively to generate long trajectories during inference. The architecture of MeshGraphNet is composed of encoding, processing, and decoding steps. In this work, MGN has been adapted to predict steady-state fields.

We utilize the following features as input (see Figure 5 for the workflow diagram):

- the distance of each node to the boundary,
- the type of node,
- the coordinates of the node.

B.1.2. Experiments

In this section, we provide a summary of the experiments conducted on various datasets.

For all datasets, we trained two separate models: one focused on field predictions and the other on scalar predictions. For scalar outputs, a readout layer inspired by Kipf & Welling (2017) is added to the model. Except for the `2D_profile` and `2D_ElPlDynamics` datasets, we only required a single model since it does not include scalar prediction tasks.

The LeakyReLU is chosen as the activation function, and all models are trained for 1000 epochs, except the `2D_ElPlDynamics` dataset with 100 epochs.

The input node features consist of those introduced in the previous section, combined with input scalars if they exist. Given two node coordinates x_i and x_j, the calculation for edge features is based on $\exp(-||x_i - x_j||_2^2/(2h^2))$, where h epresents the median value of the edge lengths within the mesh.

The rest of architecture details and training information are outlined in Table 8 and Table 9.

B.2. MMGP

The Mesh Morphing Gaussian Process (MMGP) method relies on mesh morphing to align geometries, which limits its applicability in the presence of topological variations.

Table 7. RRMSE and total_error on PLAID benchmarks, best on each line is **bold**, second best is underlined. Missing entries for MMGP correspond to non-applicable settings.

Field, *scalar* output	MGN	MMGP	Vi-Transf.	Augur	FNO	MARIO
`Tensile2d`						
U1	0.0788	**0.0015**	0.0086	0.0093	0.0174	0.0023
U2	0.1237	**0.0009**	0.0091	0.0135	0.0110	0.0030
sig11	0.1726	**0.0031**	0.0184	0.0187	0.0250	0.0040
sig22	0.0560	**0.0013**	0.0102	0.0099	0.0057	0.0017
sig12	0.0570	**0.0021**	0.0146	0.0121	0.0135	0.0023
max_von_mises	0.0185	**0.0050**	0.0090	0.0219	0.0085	0.0088
max_U2_top	0.0292	**0.0053**	0.0203	0.0344	0.0152	0.0063
max_sig22_top	0.0030	**0.0017**	0.0021	0.0030	0.0021	0.0023
total_error	0.0673	**0.0026**	0.0116	0.0154	0.0123	0.0038
`2D_MultiScHypEl`						
u1	0.0400	-	0.0173	0.0140	**0.0115**	0.0336
u2	0.0444	-	0.0172	0.0164	**0.0117**	0.0377
P11	0.0383	-	0.0337	**0.0185**	0.0353	0.0536
P12	0.0670	-	0.0581	**0.0316**	0.0513	0.1067
P22	0.0383	-	0.0343	**0.0189**	0.0359	0.0539
P21	0.0663	-	0.0571	**0.0311**	0.0510	0.1053
psi	0.0443	-	0.0312	**0.0239**	0.0329	0.0456
effective_energy	**0.0111**	-	0.0113	0.0312	0.0120	0.0220
total_error	0.0437	-	0.0325	**0.0232**	0.0302	0.0573
`2D_ElPlDynamics`						
U_x	0.0195	-	0.0186	0.0264	**0.0031**	0.0059
U_y	0.2208	-	**0.0269**	0.0427	0.0399	0.0580
total_error	0.1202	-	0.0227	0.0346	**0.0215**	0.0319
`Rotor37`						
Density	0.0114	0.0031	0.0063	0.0055	0.0840	**0.0026**
Pressure	0.0114	0.0030	0.0062	0.0053	0.0836	**0.0026**
Temperature	0.0024	0.0008	0.0019	0.0012	0.0086	**0.0005**
Massflow	0.0061	**0.0005**	0.0010	0.0028	0.0046	0.0019
Compression_ratio	0.0060	**0.0005**	0.0011	0.0028	0.0042	0.0016
Efficiency	0.0071	**0.0005**	0.0007	0.0019	0.0031	0.0016
total_error	0.0074	**0.0014**	0.0029	0.0033	0.0313	0.0017
`2D_profile`						
Mach	0.0604	0.0439	0.0359	0.0469	0.0988	**0.0355**
Pressure	0.0466	0.0208	**0.0170**	0.0248	0.0785	0.0196
Velocity-x	0.0735	0.0471	**0.0408**	0.0538	0.1148	0.0460
Velocity-y	0.0566	0.0342	0.0309	0.0445	0.0967	**0.0215**
total_error	0.0593	0.0365	0.0312	0.0425	0.0972	**0.0307**
`VKI-LS59`						
nut	0.1656	0.0822	0.0498	0.0483	0.0846	**0.0259**
mach	0.0451	0.0309	0.0232	0.1896	0.0180	**0.0112**
Q	0.0716	**0.0023**	0.0052	0.0074	0.0047	0.0052
power	0.0403	**0.0057**	0.0083	0.0070	0.0062	0.0077
Pr	0.0068	0.0026	0.0024	0.0040	0.0019	**0.0018**
Tr	0.0001	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
eth_is	0.1912	0.1224	0.0621	0.1228	0.0539	**0.0453**
angle_out	0.0263	0.0033	0.0031	0.0048	0.0027	**0.0023**
total_error	0.0684	0.0312	0.0193	0.0267	0.0215	**0.0124**

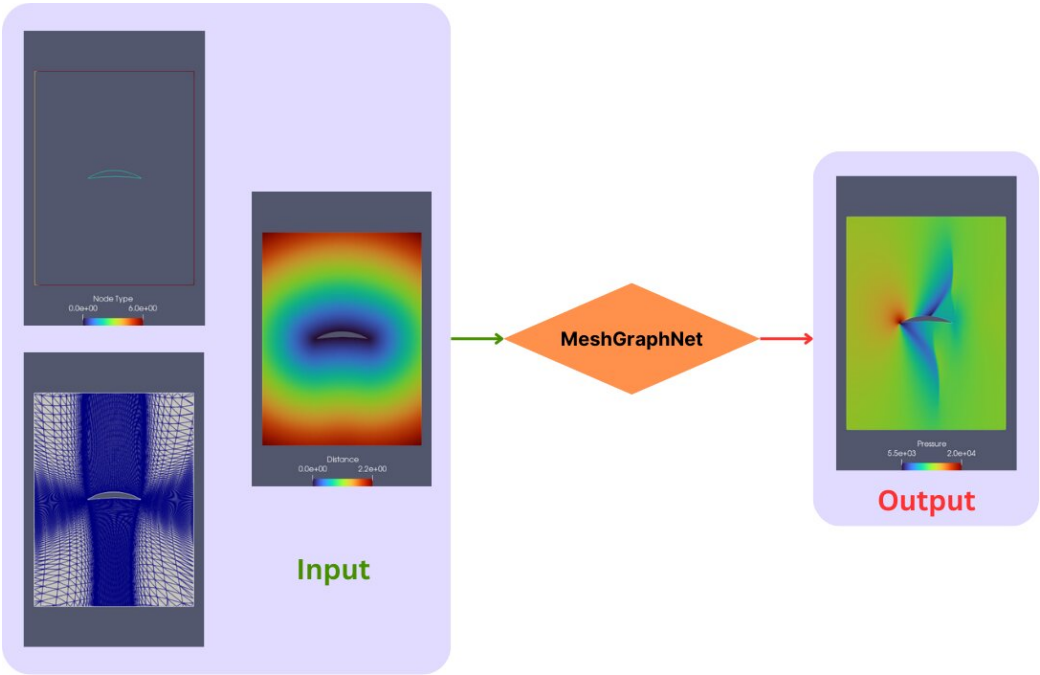


Figure 5. Illustration of MGN workflow to predict steady-state pressure field of a sample from the `2D_profile` dataset.

Table 8. Field MGN: Architecture details and training statistics across datasets.

Dataset	Message Passing Steps	Latent Size	Nbe epochs	Batch size	Training Time	Hardware
Tensile2d	10	16	1000	1	3h46min	1 × A100
2D_MultiScHypEl	10	32	1000	1	5h54min	1 × A100
Rotor37	10	64	1000	1	19h24min	1 × A100
2D_profile	10	128	1000	1	17h27min	1 × A100
VKI-LS59	10	64	1000	1	16h32min	1 × A100
2D_ElPlDynamics	15	64	100	8	15h43min	8 × A100

Table 9. Scalar MGN: Architecture details and training statistics across datasets. The `2D_profile` and `2D_ElPlDynamics` benchmarks do not include scalar outputs.

Dataset	Message Passing Steps	Latent Size	Nbe epochs	Batch size	Training Time	Hardware
Tensile2d	10	32	1000	1	4h6min	1 × A100
2D_MultiScHypEl	10	16	1000	1	6h	1 × A100
Rotor37	10	16	1000	1	10h	1 × A100
VKI-LS59	10	16	1000	1	9h13min	1 × A100

B.2.1. METHOD

We refer the reader to (Casenave et al., 2024) for a complete presentation of the method. MMGP combines four main ingredients: (i) mesh morphing, (ii) finite element interpolation, (iii) dimensionality reduction, and (iv) Gaussian process regression. Together, these enable learning mappings between geometries and solution fields for PDEs, even when the input geometry is provided as non-parametrized meshes.

An overview of the workflow is illustrated in Figure 6, which should be read from left to right. On the left are sample-specific input geometries; on the right are the corresponding solution fields defined on these geometries.

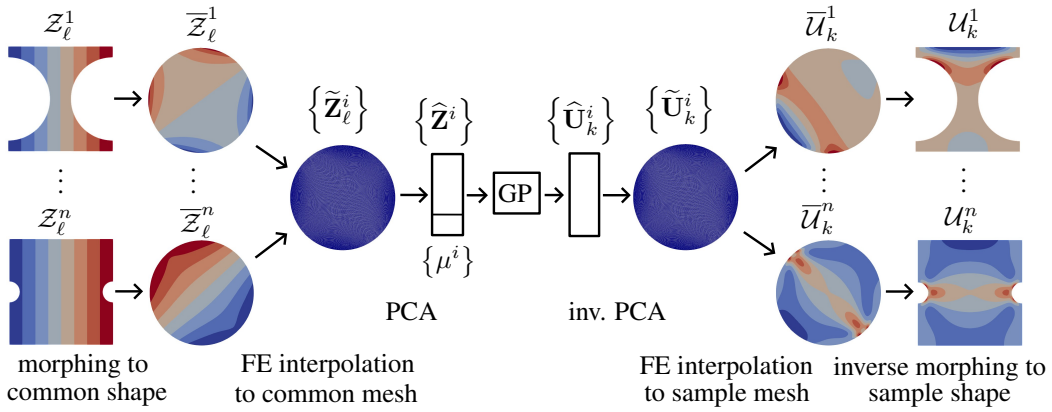


Figure 6. Illustration of the MMGP inference workflow for the prediction of an output field of interest (Casenave et al., 2024).

Since input meshes are not parametrized, they must first be embedded into a learnable space. MMGP does this by interpreting mesh vertex coordinates as continuous fields (e.g., the x-coordinate field shown in the left column of Figure 6, exhibiting vertical iso-lines). Each mesh is then deterministically morphed onto a reference geometry—the unit disk in this 2D example, but it can be one of the training samples shape. Next, each sample morphed coordinate fields are projected onto a common mesh of the reference geometry via finite element interpolation. This ensures all samples share a consistent discretization, making them compatible with standard dimensionality reduction techniques like PCA. The result is a compact, fixed-size representation of the geometry. When scalar inputs are present, they can be concatenated to the reduced vector.

A similar procedure is applied to the output fields: morphing onto the reference geometry, projection onto the common mesh, and PCA compression yield low-dimensional field representations aligned with the geometric embeddings.

These deterministic preprocessing steps transform the original complex problem—mapping between high-dimensional and irregularly discretized fields—into a standard regression task between low-dimensional vectors. This enables the use of classical regression models; we adopt Gaussian process regression due to its robustness, accuracy, and built-in uncertainty quantification.

MMGP offers several practical advantages: it handles large meshes efficiently, produces interpretable models, and delivers high accuracy in our experiments, with uncertainty estimates. In industrial design applications, where data can lie on low-dimensional manifolds, small models like MMGP can be especially effective—provided that the reparametrization (or embedding) is constructed appropriately, here with the morphing.

The main limitations of MMGP are tied to the morphing step, which currently requires problem-specific setup, and the fact that morphing and interpolation must still be performed at inference time. These challenges are addressed in recent works (Kabalan et al., 2025b; 2024), which introduce automatic alignment and online-efficient morphing strategies. Kabalan et al. (2025a) proposed improvements, where optimization techniques are used to generate morphings that maximize PCA compression.

All mesh and field operations are implemented using the Muscat library (Bordeu et al., 2025; 2023). An upcoming release will include a GPU-accelerated finite element interpolation routine, significantly improving inference latency.

Additional improvement of MMGP are possible, by replacing the linear decoder of the PCA by a non-linear one that accounts for high-order interactions among the selected POD modes and includes a rotation of the POD basis and a polynomial correction, as proposed by Geelen et al. (2023).

Physics-based models compatible with the morphing, finite element interpolation and dimensionality reduction of MMGP have been proposed. The physics equation can be efficiently assembled and solved on the low-dimension space spanned by the PCA modes obtained after morphing, instead of using data-driven low-dimensional models. In (Barral et al., 2024a), a hyper-reduced least-square Petrov-Galerkin scheme is used to reduced the Navier-Stokes equations, with morphing. While much more complicated to utilize, we expect such methods to greatly improve the accuracy, with a moderate additional computation cost.

B.2.2. EXPERIMENTS

Hyperparameters and training statistics for the MMGP experiments are listed in Table 10. We first mention that MMGP has not been applied to the `2D_ElPlDynamics` and `2D_MultiScHypEl` datasets, since the method is yet to be extended to variable topology settings.

We notice that `Rotor37` and `VKI-LS59` do not require morphing, since the samples' meshes have the same number of nodes. In `Tensile2d` and `2D_profile`, systematic morphing strategies to align the shapes are sufficient, with respectively Tutte barycentric embedding (Casenave et al., 2024) (Annex B) and elasticity-based automatic morphing (Kabalan et al., 2025b).

Since the `VKI-LS59` dataset exhibits discontinuities due to the presence of shock waves, a non-linear decoder (Geelen et al., 2023) was employed to reconstruct the fields of interest. For the compression of the `mach` fields, 5 POD modes and a polynomial order of 3 were used, while 40 POD modes were retained for the compression of the `nut` fields. Since polynomial decoders are prone to overfitting, the number of modes and the polynomial order were selected through a k-fold cross-validation procedure on the training set.

Since the solution fields of `2D_profile` and `VKI-LS59` feature complex structures (e.g. shocks of variable position), we expect the involved optimal morphing strategy of Kabalan et al. (2025a) to significantly improve the results of MMGP on these cases.

Table 10. Hyperparameters and training statistics for the MMGP experiments (on an AMD EPYC 9534 CPU). Training times include all preprocessing (morphing, finite element interpolation and dimensionality reduction), in addition to the training of the Gaussian processes. *For `VKI-LS59`, X-Y stands for the number of modes and polynomial order of the decoder for the `mach` and `nut` fields respectively. **Not including morphing time (which takes approximately 10min on 300 cores).

Dataset	Morphing	PCA modes (shape)	PCA modes (field)	GP kernel	Training time	Hardware
Tensile2d	Tutte	8	8	Matérn 5/2	13min02s	128 cores
Rotor37	None	32	64	Matérn 5/2	6min13s	128 cores
2D_profile	Elasticity	16	32	RBF	18min32s**	12 cores
VKI-LS59	None	13	5-3/40-1*	Matérn 5/2	4min13s	64 cores

B.3. Vi-Transformer and Augur

These methods rely on mesh partitioning and tokenization, enabling flexible handling of irregular geometries while preserving local structural information.

B.3.1. METHOD

Transformers for long context range regression. The natural way of dealing with mesh-based regression problems is to use GNN models which rely on message-passing. Although these are great at capturing information locally, they struggle to retrieve it at long distances. Indeed, the smallest number of GNN layers needed to have a receptive field that covers the whole graph is half the diameter of the graph. This becomes computationally impractical in the context of large simulation meshes. This behavior is analogous to Convolutional Neural Networks (CNNs) in Computer Vision (CV) where long-range dependencies are only captured at the deeper levels of the network. One way of alleviating this is to consider transformer architectures, which compute similarities between all the input tokens simultaneously thanks to the attention mechanism, thus removing the need to have infeasibly deep networks.

Transformers on very large data. One of the main challenges of transformers in this case is to handle the large size of the meshes (in the order of tens of thousands of points per mesh, and up to millions with practical industrial problems).

Currently, the computational bottleneck of transformers is a widely considered subject: given N tokens of dimension D, the critical issue of self attention is that one needs $N^2 \times \mathcal{D}$ operations where $\mathcal{D} \approx D$ is the size of the embedding of each token, and N^2 is the cost to compute the Gram matrix of the N tokens (this computation cost is also a memory one as storing the matrix requires also $N^2\mathcal{D}$ numbers).

Many papers have focused on the possibility to linearize the cost of self-attention, for example:

- Kitaev et al. (2020) introduces Reformer which considers the formulation of the attention mechanism : $\mathrm{softmax}\left(\frac{QK}{\sqrt{D}}\right)$ with the key and query matrices (respectively K and Q), capitalizing on the fact that for a given query Q_i, only the keys which provide high dot products with Q_i will have a significant impact on the value of $\mathrm{softmax}\left(\frac{Q_i K^T}{\sqrt{D}}\right)$. Therefore, Reformer makes use of locality-sensitive hashing for only computing the $Q_i K_j^T$ products with the p keys that are closest to a query, where $p \in \mathbb{N}$ is a chosen hyperparameter, efficiently linearizing the self attention.

- Wang et al. (2020) introduces Linformer. Coarsely, Linformer relies on the Nyström approximation to approximate the Gram matrix of self attention. Precisely, while the Nyström approximation replaces an $n \times n$ symmetric matrix A by UU^T where U is only $n \times k$ containing the eigenvectors of largest eigenvalue, Linformer offers to learn E, F such that $\mathrm{softmax}\left(\frac{QK}{\sqrt{D}}\right) \approx EF^T$. This also offers a linear approximation of the self attention computation.

This has also been tackled in CV tasks (Dosovitskiy et al., 2021b), where self-attention is not applied on pixels directly but on pixel-patches that aggregate pixel neighborhoods into tokens, thus drastically reducing the self-attention's input sequence length.

Transformers for large scale point-wise regression. The most used transformer architectures are in one of two categories. The auto-regressive sequence-to-sequence transformers, mostly used in Natural Language Processing (NLP) for text generation, and the sequence-to-class ones which are used both in NLP, as in sentiment analysis (Durairaj & Chinnalagu, 2021), spam detection (Jamal & Wimmer, 2023), long document classification (Dai et al., 2022), and CV with image classification (Dosovitskiy et al., 2021b).

Both are quite different from the point-wise regression objective of the PLAID benchmarks. Indeed, the first method generates new token sequences of arbitrary lengths, while the second only makes use of transformer encoders with neural network heads to obtain a probability distribution on a set of classes.

Some work has been conducted in order to tackle regression problems with transformers:

- Segformer (Xie et al., 2021) addresses this in the case of image segmentation; it uses a multiscale U-type transformer to sequentially downscale the input image, and uses a multiscale MLP head to decode these downscaled states into the output segmentation mask.

- Point Transformer (Zhao et al., 2021) also uses a U-style encoder-decoder architecture, this time on 3D point-cloud data for both segmentation and classification.

- TransCFD (Jiang et al., 2023a) tackles airfoil surrogate CFD modeling by using a decoder-only architecture from a latent embedding of the input geometry. It relies on structured regular grids (images) of the inputs, and not arbitrary mesh discretizations.

- Point Transformer V3 (Wu et al., 2024) groups points together and computes attention scores within these groups. Local and long-distance information are captured through different serializations of the input mesh.

Both Segformer and TransCFD make use of the regular nature of their data to precisely decode (and/or encode) the output (and/or input) fields. Point Transformers, on the other hand, handle unstructured point-cloud data. Although these methods fit the nature of the PLAID benchmark, we propose lighter methods that stick more closely to the classical transformer model.

Vi-Transformer for mesh field regression. The chosen approach relies on a transformer encoder architecture and is analogous to Vision Transformers (Vi-Transformer). Rather than considering each node of the mesh as a token by its own, the encoder takes as input tokenized point-cloud patches. Local information is kept within the patches while long-range information is retrieved through the transformer's mapping, which compares all token pairs together. The general architecture of the Vi-Transformer is depicted in Figure 7. Although this model primarily focuses on steady-state simulations, it can be trained in an auto-regressive way to learn unsteady simulation increments; and reconstructing, on inference, the full evolution by auto-regressively estimating the considered fields.

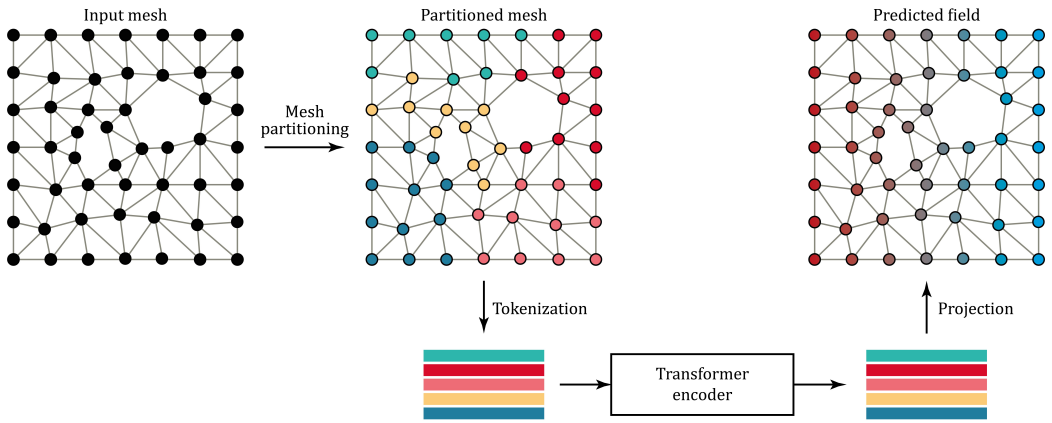


Figure 7. Vi-Transformer architecture. Input meshes are partitioned using the Metis domain decomposition algorithm (Karypis & Kumar, 1995). Each such sub-domain is then tokenized before passing through the transformer encoder. In the end, each token is decoded into its domain's corresponding fields. Input scalars are embedded during the tokenization procedure while output scalars are estimated as uniform fields.

Augur Transformer model. Augur has developed Transformer models specifically designed for numerical simulations. These models share fundamental architectural similarities with Vision Transformers (ViT), where the computational mesh is decomposed into patches. Each patch is embedded into a latent space, resulting in the input tokens for the Transformer architecture. This approach enables information exchange between local patches across long spatial distances, similar to how ViTs process image data.

The key innovation in Augur's approach lies in the decoding mechanism, addressing a critical question: how to properly reconstruct the output field from the processed sequence of tokens? In traditional ViT architectures, direct reconstruction from individually processed tokens can result in discontinuities at patch interfaces due to insufficient global context integration. Augur models overcome this limitation by incorporating a global information vector that aggregates data from all tokens. The decoder then uses a combination of point-specific information, processed local features, and global context to produce a more robust and consistent output field. Furthermore, unlike ViTs, Augur models do not treat scalar predictions as constant fields but instead derive them directly from the global information vector, enhancing prediction accuracy. The general architecture of the Augur model is depicted in Figure 8.

B.3.2. Experiments

Both the Vi-Transformer and Augur models rely on a relatively small number of hyperparameters. These include the patch size (i.e., the number of nodes per patch), the latent dimension onto which the aggregated patches are projected, and the Transformer encoder hyperparameters, such as the number of heads, the number of transformer encoder layers and the dimension of the feedforward layer. Table 11 details the hyperparameters for the Vi-Transformer, while Table 12 outlines those for the Augur model.

B.4. FNO

Fourier Neural Operators (FNO) operates on regular grids and therefore requires projection from unstructured meshes, which may introduce approximation errors in highly heterogeneous settings.

FNO belong to the class of operator learning architectures, i.e., they aim at approximating mappings between function

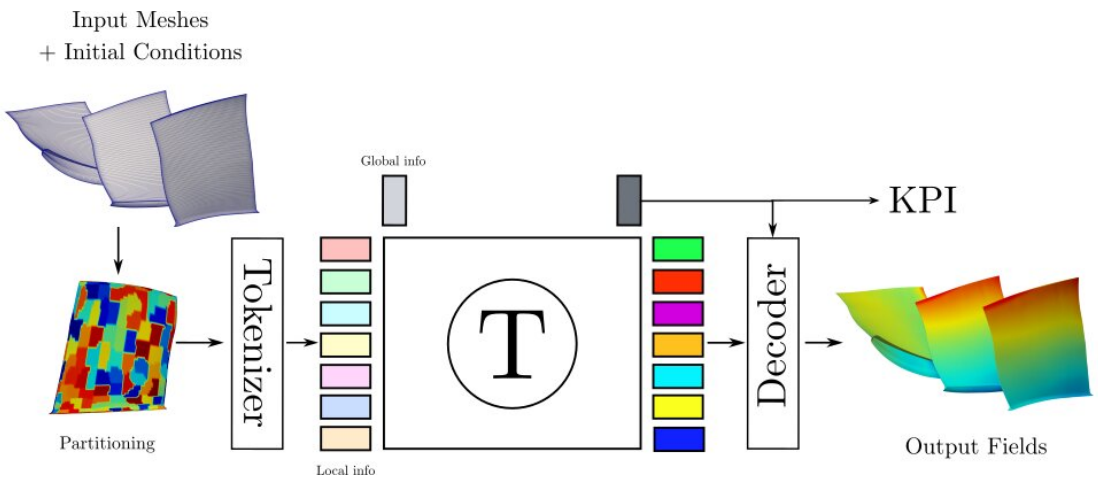


Figure 8. Augur Transformer architecture: Input meshes are partitioned using the Metis domain decomposition algorithm. Each subdomain is then tokenized before being passed through the Transformer. An additional global tensor is added to the Transformer to gather global information. Output fields are reconstructed using a decoder that leverages both local and global information. Output scalars (KPIs) are predicted directly from the global tensor.

Table 11. (Vi-Transformer) Hyperparameters and training statistics for the Vi-Transformer experiments. Training times include all preprocessing (domain decomposition, tokenization), in addition to the training of the model itself. The number of attention heads is kept at 16 for all experiments.

`Dataset`	Patch size	Latent dimension	Feedforward dimension	Nb encoder layers	Training time	Hardware
`Tensile2d`	10	1024	512	5	2h43	1 × A30
`2D_MultiScHypEl`	10	1024	512	5	2h07	1 × A30
`Rotor37`	30	1024	512	5	3h59	1 × A30
`2D_profile`	25	1024	512	5	6h31	1 × A30
`VKI-LS59`	20	1024	512	5	6h39	1 × A30
`2D_ElPlDynamics`	100	256	128	2	48min	1 × A30

Table 12. (Augur) Hyperparameters and training statistics for the Augur experiments. Training times include all preprocessing (domain decomposition, tokenization), in addition to the training of the model itself.

`Dataset`	Patch size	Latent dimension	Feedforward dimension	Nb encoder layers	Training time	Hardware
`Tensile2d`	16	512	2048	8	1h11	1 × RTX 2080Ti
`2D_MultiScHypEl`	4	128	512	8	7h48	1 × RTX 2080Ti
`Rotor37`	32	256	1024	8	2h30	1 × RTX 2080Ti
`2D_profile`	128	256	512	4	1h51	1 × RTX 2080Ti
`VKI-LS59`	64	512	2048	4	2h15	1 × RTX 2080Ti
`2D_ElPlDynamics`	256	256	4	4	1h30	1 × RTX 2080Ti

spaces. By leveraging the Fast Fourier Transform (FFT) within their layers, FNOs represent functions on regular grids, effectively working in a finite-dimensional discretized space. This architecture enables learning transformations in the frequency domain, which has been shown to be particularly effective for modeling complex physical systems compared to standard convolutional neural networks.

B.4.1. METHOD

We use the classical FNO architecture introduced by Li et al. (2020). Let $u : \mathbb{R}^2 \to \mathbb{R}^k$ denote the input field. Each FNO layer is defined as the sum of a local linear transformation and a non-local integral operator parameterized in the Fourier domain:

$$\mathcal{J}[u](x) = \sigma\left(Wu(x) + \mathcal{F}^{-1}\left(W^*\mathcal{F}[u]\right)(x)\right), \tag{1}$$

where $W \in \mathbb{R}^{k\times k}$ and W^* are learnable parameters, $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, and σ is a pointwise non-linear activation function. Stacking multiple such layers yields a global operator that captures long-range dependencies through spectral convolution.

A key limitation of FNO models is that they operate on regular grids due to the use of FFT. As a result, applying FNOs to data defined on unstructured meshes requires a preprocessing step that projects the fields onto a regular grid, followed by a postprocessing step mapping predictions back to the original mesh. In our experiments, these projection operations are performed using Muscat (Bordeu et al., 2025; 2023).

B.4.2. EXPERIMENTS

B.4.3. 2D_ELPLDYNAMICS

FNO can be used to model transient dynamics in physics-based systems. Among the datasets introduced in Section 4, `2D_ElPlDynamics` is the only one featuring time-dependent behavior with evolving topology.

Training procedure. The training is performed in an autoregressive manner: given the input fields at time t, the model predicts the fields at time $t + dt$, similarly to an explicit time integration scheme. Once trained, full trajectories are obtained by recursively applying the model starting from the initial condition.

The input features consist of the displacement fields provided by the dataset (`U_x`, `U_y`), augmented with spatial coordinate fields (x, y), which are commonly used in FNO-based models to encode positional information.

As FNO operates on regular grids, the fields defined on unstructured meshes are first projected onto a regular grid before being processed by the network. After prediction, the output fields are projected back onto the original mesh to enable comparison with reference solutions. We summarize the input/output quantities in Table 13.

The training was parallelized on 40 GPUs (A100) and lasted 6 hours. Inference and testing can be performed on a single GPU to compute the metrics presented in Table 7.

B.4.4. OTHER DATASETS

For the remaining datasets, which correspond to stationary problems, we train FNO models to directly map input quantities to output fields without temporal recursion.

The input features depend on each dataset and typically include scalar parameters (e.g., boundary conditions or material coefficients) together with spatial coordinate fields. When field inputs are available, they are projected onto a regular grid and concatenated as additional channels.

Similarly to the transient case, all data defined on unstructured meshes are projected onto regular grids prior to training. Predictions produced by the FNO are then mapped back onto the original meshes for evaluation using the PLAID benchmark metrics.

Model and training parametrization. We summarize all the models parametrization in Table 14.

Table 13. Features throughout the learning process. The simulation at t (column 1) is projected to a regular grid (column 2), which serves as input to the FNO model. The model predicts the fields at $t + dt$ (column 3), which are then mapped back to the original mesh.

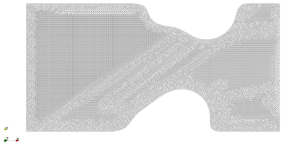
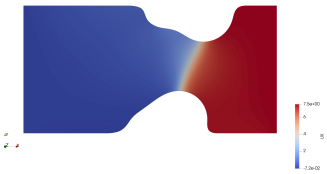
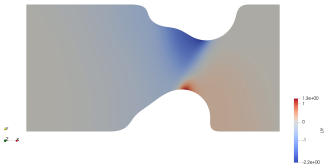
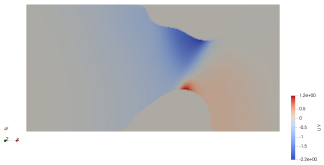
Attribute	Simulation at t	Input FNO	Output FNO
Mesh		-	-
U_x			
U_y			

Table 14. FNO hyperparameters and training statistics.

Dataset	Grid shape	Layer count	Latent channels	Decoder layers	Decoder layers size	Fourier modes	Training time	Hardware
Tensile2d	201^2	4	64	4	64	16^2	2h46	$1 \times$ A30
2D_MultiScHypEl	201^2	4	64	4	64	16^2	4h12	$1 \times$ A30
Rotor37	64^3	4	32	4	32	32^2	7h19	$1 \times$ A30
2D_profile	601×801	4	32	2	32	32^2	2h28	$1 \times$ A30
VKI-LS59	301×121	4	64	4	64	16^2	2h36	$1 \times$ A30
2D_ElPlDynamics	301×151	4	64	1	64	20^2	6h	$40 \times$ A100

B.5. MARIO

Modulated Aerodynamic Resolution-Invariant Operator (MARIO) leverages implicit neural representations, allowing continuous modeling of fields without relying on fixed discretizations, which improves robustness to heterogeneous geometries.

MARIO is a deep learning model designed to approximate the solution operator of a partial differential equation (PDE) (Catalani et al., 2025), involving geometric variability. It leverages Conditional Neural Fields (or Implicit Neural Representations) to learn the mapping between spatial coordinates from a mesh, geometric information (e.g., via the signed distance function, SDF), inflow conditions, and the resulting physical field. Unlike mesh-based methods, INRs represent continuous fields through neural network parameterizations, enabling resolution-independent predictions and flexible evaluation. MARIO extends this approach to handle multiple geometries and operating conditions through a conditioning mechanism.

B.5.1. Method

Modulated INR architecture. MARIO implements a conditional neural field approach where a single neural network architecture can represent multiple distinct signals through a conditioning mechanism. The conditioning variable $z = [\mu_{\text{geom}}, \mu]$ encodes both geometric parameterization μ_{geom} and operating conditions μ (e.g., angle of attack, Mach number, Reynolds number).

The main network is a multilayer perceptron (MLP) where the layer outputs are modulated by sample-specific vectors:

$$f_{\theta,\phi}(x) = W_L(\eta_{L-1} \circ \eta_{L-2} \circ \cdots \circ \eta_1 \circ \gamma(x)) + b_L \tag{2}$$

$$\eta_l(\cdot) = \text{ReLU}(W_l(\cdot) + b_l + \phi_l(z)) \tag{3}$$

where $\phi_l(z) = [h_\psi(z)]_l \in \mathbb{R}^{d_l}$ are layer-specific modulation vectors obtained from the hypernetwork h_ψ that processes the conditioning variable z. The main network parameters θ are shared for all samples and consist of the weights and biases matrices W_l, b_l. In MARIO, an explicit shape encoding μ_{geom} is used as input of the architecture to properly model geometric variability. In many real-world applications, a geometric parameterization is not available or insufficient to capture complex shapes. Therefore, a learning mechanism to obtain compact geometric representations from the SDF fields is adopted. These encoding process leverages a separate Neural Field encoder, that maps input coordinates to output SDF values, while fitting latent shape representations.

Geometry encoding mechanism. For each geometry's signed distance function (SDF), a meta-learning optimization procedure based on CAVIA (Zintgraf et al., 2019) adapts a shared neural network $f_{\theta_{in},\phi_{in}}$ to represent different shapes. Given the shared network parameters θ_{in} and hypernetwork parameters ψ, the latent representation $\mu_{geom} = z_{in}^{(K)}$ for geometry i is obtained by solving:

$$z_{in}^{(0)} = 0 \tag{4}$$

$$z_{in}^{(k+1)} = z_{in}^{(k)} - \alpha \nabla_{z_{in}^{(k)}} \mathcal{L}_{in}(f_{\theta_{in},\phi_{in}}(x), sdf_i), \quad \text{for } 0 \leq k \leq K-1 \tag{5}$$

where $\phi_{in} = h_\psi(z_{in}^{(k)})$, α is the inner loop learning rate, and K is the number of optimization steps (typically set to 3). The loss $\mathcal{L}_{in}$ measures the reconstruction error between the true SDF field and its prediction over a sampling grid defined on the input domain.

This optimization process, illustrated in Figure 9, yields a compact latent code $\mu_{\text{geom}} = z_{in}^{(K)}$ that captures the essential geometric features.

Fourier feature encoding. To address the spectral bias inherent in neural networks, MARIO employs Fourier feature encoding for the input coordinates:

$$\gamma(x) = [\cos(2\pi \mathbf{B}x), \sin(2\pi \mathbf{B}x)] \tag{6}$$

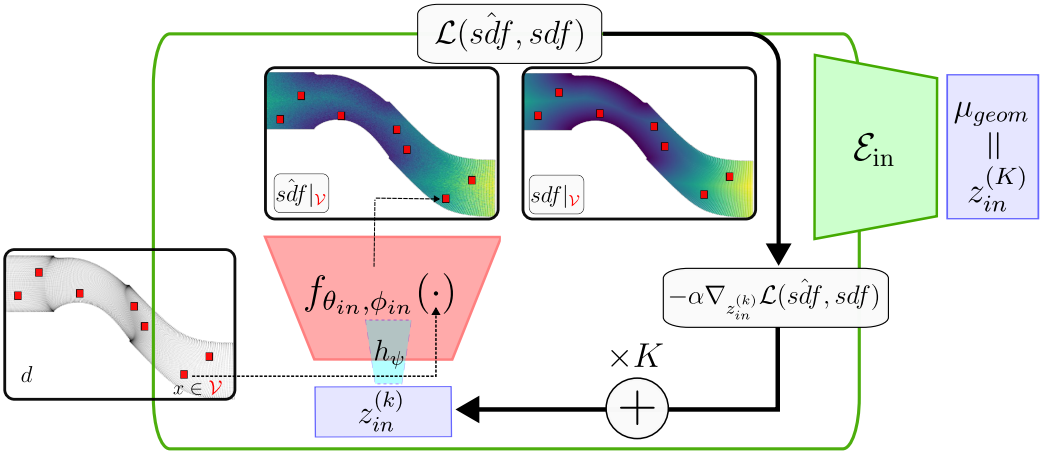


Figure 9. MARIO geometry encoding process.

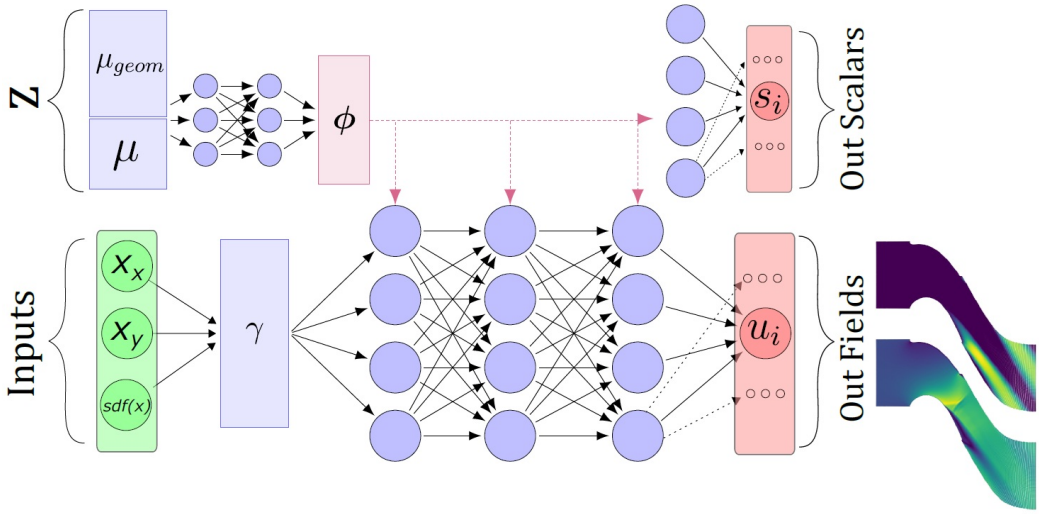


Figure 10. MARIO architecture.

where $\mathbf{B} \in \mathbb{R}^{m \times d}$ contains frequency vectors sampled from a Gaussian distribution $\mathcal{N}(0, \sigma)$. This encoding enables the network to better capture high-frequency details in the output fields.

Scalar output prediction. In addition to predicting coordinate-dependent fields, MARIO can also predict global scalar quantities for each sample. Since these scalar outputs are global properties of the solution (e.g., power coefficients, efficiency metrics), they depend only on the sample-specific information encoded in the modulation vectors. The scalar prediction is therefore implemented as:

$$s = W_s \cdot \phi_{agg} + b_s \tag{7}$$

where ϕ_{agg} represents an aggregation of the modulation vectors produced by the hypernetwork. This single-layer transformation efficiently leverages the already learned sample representation without requiring additional feature extraction.

The architecture of MARIO is illustration in Figure 10.

Training procedure. MARIO is trained using a weighted loss function that balances field prediction accuracy and scalar output accuracy:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{field}} + (1 - \alpha) \cdot \mathcal{L}_{\text{scalar}} \tag{8}$$

where $\alpha \in [0, 1]$ is a weighting parameter. The field loss $\mathcal{L}_{\text{field}}$ is computed as the mean squared error between predicted and target fields across spatial locations, while the scalar loss $\mathcal{L}_{\text{scalar}}$ is the mean squared error of the global quantities.

Key advantages. MARIO presents three major benefits: (i) it is resolution-invariant and can be evaluated at arbitrary spatial locations; (ii) it overcomes spectral bias through multiscale Fourier encodings; and (iii) it adapts to geometry-specific variations via bias modulation using the auxiliary network h_ψ.

B.5.2. EXPERIMENT

The MARIO model is applied to the six datasets, with the parametrization and training procedures provided in Table 15.

Table 15. MARIO hyperparameters and training statistics.

Dataset	Geom. Hyper. depth	Geom. Hyper. width	Geom. latent dim	Hypernet. depth	Hypernet. width	INR depth	INR width	freq. nb.	Training time	Hardware
Tensile2d	1	128	16	3	256	6	256	64	3h40	1 × A30
2D_MultiScHypEl	1	128	100	3	256	6	256	64	6h52	1 × A30
Rotor37	1	128	32	3	256	6	256	64	4h45	1 × A100
2D_profile	1	128	8	3	256	6	256	64	2h43	1 × A100
VKI-LS59	1	128	16	3	256	6	256	64	8h45	1 × A100
2D_ElPlDynamics	1	128	16	3	256	6	256	64	26min	1 × A30

C. Additional details on PLAID

We illustrate further the capabilities of PLAID by providing some additional commands to retrieve information from our datasets directly from Hugging Face.

C.1. Tensile2d

Tensile2d is a simple dataset, for which standard and simple PLAID commands are sufficient to retrieve the data:

```python
from datasets import load_dataset
from plaid.containers.sample import Sample
import pickle

# Load the dataset
hf_dataset = load_dataset("PLAID-datasets/Tensile2d", split="all_samples")

# Get split ids
ids_train = hf_dataset.description["split"]["train_500"]

# Get inputs/outputs names
in_scalars_names = hf_dataset.description["in_scalars_names"]
out_fields_names = hf_dataset.description["out_fields_names"]

# Get samples
sample = Sample.model_validate(pickle.loads(hf_dataset[ids_train[0]]["sample"]))

# Examples of data retrievals
nodes = sample.get_nodes()
elements = sample.get_elements()
nodal_tags = sample.get_nodal_tags()

for sn in ["P", "p1", "p2", "p3", "p4", "p5"]:
    scalar = sample.get_scalar(sn)
```

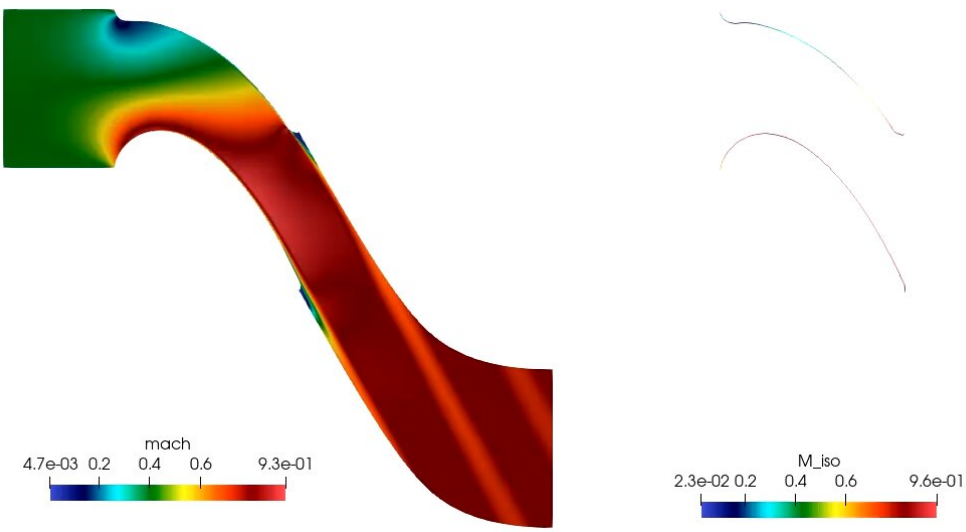


Figure 11. Illustration of the first sample in the train split of `VKI-LS59`: (left) fluid domain, (right) blade surface domain.

```
# outputs
for fn in ["U1", "U2", "q", "sig11", "sig22", "sig12"]:
    field = sample.get_field(fn)

for sn in ["max_von_mises", "max_q", "max_U2_top", "max_sig22_top"]:
    scalar = sample.get_scalar(sn)
```

The geometrical support in PLAID samples can be easily converted to Muscat meshes:

```
from Muscat.Bridges import CGNSBridge
CGNS_tree = sample.get_tree()
mesh = CGNSBridge.CGNSToMesh(CGNS_tree)
```

C.2. `VKI-LS59`

`VKI-LS59` also contains stationary configurations, meaning only one time step per sample, but features a complex geometrical setting, with a 2D fluid domain and a 1D blade surface domain, see Figure 11.

The fluid domain contains 2D elements in a 2D ambient space, hence is contained in the CGNS base called "Base_2_2". For the blade surface domain, we have 1D elements in a 2D ambient space: the CGNS base is then "Base_1_2". The corresponding data are retrieved as follows:

```
from datasets import load_dataset
from plaid.containers.sample import Sample
import pickle

# Load the first sample of the train split
hf_dataset = load_dataset("PLAID-datasets/VKI-LS59", split="all_samples")
ids_train = hf_dataset.description["split"]["train"]
sample = Sample.model_validate(pickle.loads(hf_dataset[ids_train[0]]["sample"]))

# Examples of data retrievals
for fn in ["sdf", "ro", "rou", "rov", "roe", "nut", "mach"]:
```

```
    field = sample.get_field(fn, base_name="Base_2_2")
M_iso = sample.get_field("M_iso", base_name="Base_1_2")
for sn in sample.get_scalar_names():
    scalar = sample.get_scalar(sn)

nodes_fluid = sample.get_nodes(base_name="Base_2_2")
nodes_blade_surface = sample.get_nodes(base_name="Base_1_2")
elements_fluid = sample.get_elements(base_name="Base_2_2")
elements_blade_surface = sample.get_elements(base_name="Base_1_2")
nodal_tag_fluid = sample.get_nodal_tags(base_name="Base_2_2")
```

The meshes for the fluid domain and blade surface domain can also be converted to Muscat meshes:

```
from Muscat.Bridges import CGNSBridge
CGNS_tree = sample.get_tree()
mesh_fluid = CGNSBridge.CGNSToMesh(CGNS_tree, baseNames=["Base_2_2"])
mesh_blade = CGNSBridge.CGNSToMesh(CGNS_tree, baseNames=["Base_1_2"])
```

C.3. `2D_ElPlDynamics`

`2D_ElPlDynamics` contains additional complexity: time-dependent data and a field located at the center of the elements. When retrieving data, the default location of the fields is at the vertices. For other type of fields, location mush be specified. The corresponding commands are provided below:

```
from datasets import load_dataset
from plaid.containers.sample import Sample
import pickle

# Load the first sample of the train split
hf_dataset = load_dataset("PLAID-datasets/2D_ElastoPlastoDynamics", split="all_samples")
ids_train = hf_dataset.description["split"]["train"]
sample = Sample.model_validate(pickle.loads(hf_dataset[ids_train[0]]["sample"]))

# Examples of data retrievals
time_steps = sample.get_all_time_values()

for time in time_steps:
    for fn in ["U_x","U_y"]:
        field = sample.get_field(fn, time = time)
    field = sample.get_field("EROSION_STATUS", location="CellCenter", time = time)

CGNS_tree = sample.get_tree(time = 0.1)
```

D. Benchmarking online applications

Anyone wishing to participate in our benchmarks, hosted at [LINK REMOVED], should create a Hugging Face account. However, no account is required to browse the website or view the leaderboards. To participate, users simply need to train their model independently and submit predictions on the testing set. We do not require participants to upload their models. Two separate leaderboards are maintained, each based on a hidden subset of the test set, in order to discourage tentatives to overfit on the testing set.

We illustrate the benchmarking application using the `VKI-LS59` dataset as an example.

Figure 12 shows the benchmark homepage. A navigation menu is available on the left-hand side, allowing users to browse the site and log in. This page also provides examples of the dataset output fields and includes a visualization tool, where users can select a training sample ID and an output field to display.

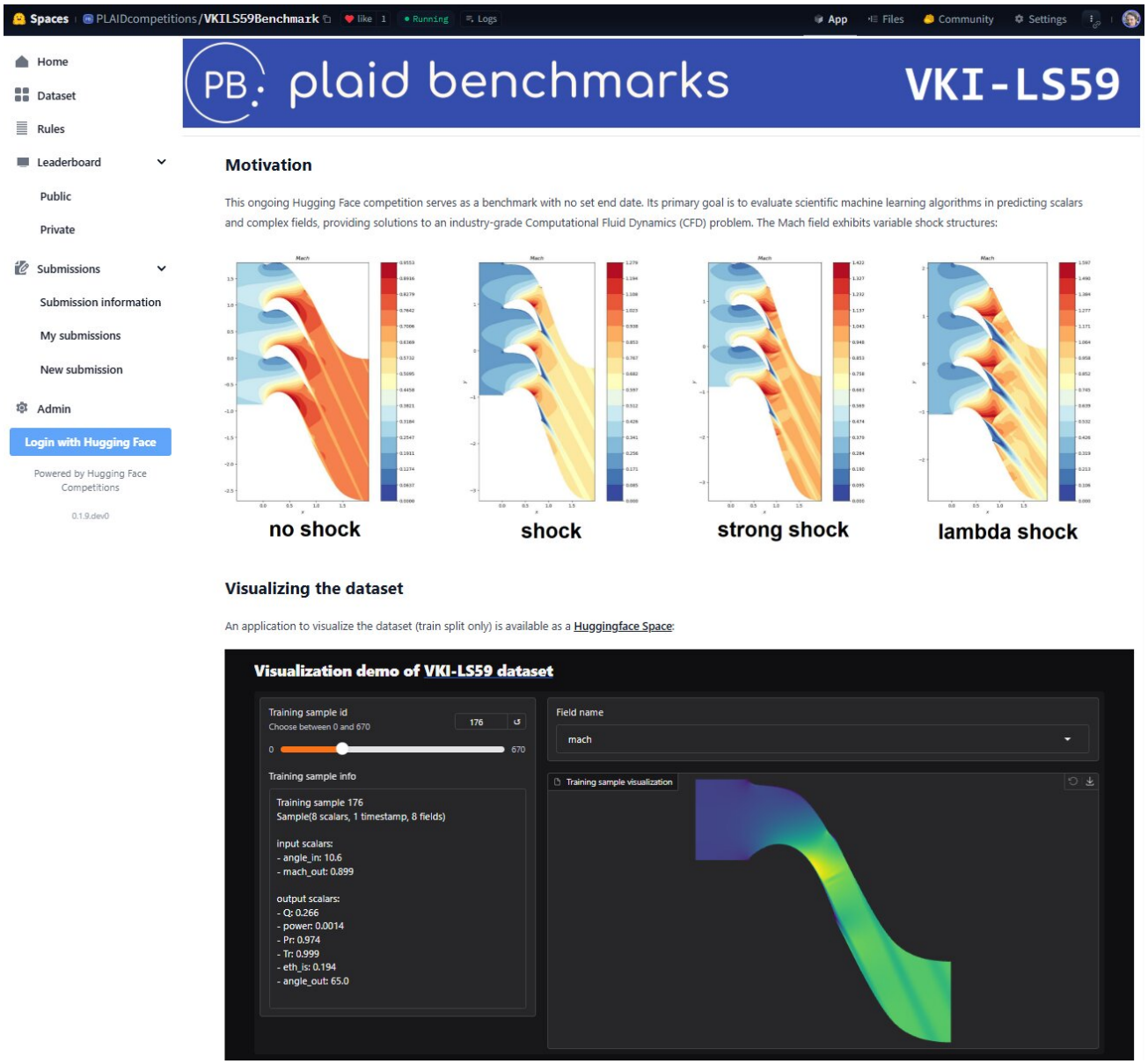


Figure 12. "Home" page of the benchmarking application on the `VKI-LS59` dataset.

Figure 13 provides detailed instructions on how to retrieve the dataset, including a description of the inputs and outputs used in the benchmark. Example commands are also provided to retrieve the samples and the required associated data.

The set of rules applying to the benchmark is presented in Figure 14.

Figure 15 provides detailed instructions on how to generate and submit the prediction file. The scoring function used for evaluation is also described.

Figure 16 illustrates the user's submissions page and the submission interface.

Figure 17 shows the public leaderboard as it appeared at the time of submission of this work.

Spaces | PLAIDcompetitions/VKILS59Benchmark | like 1 | Running | Logs | App | Files | Community | Settings

- Home
- Dataset
- Rules
- Leaderboard
  - Public
  - Private
- Submissions
  - Submission information
  - My submissions
  - New submission
- Admin

Login with Hugging Face

Powered by Hugging Face Competitions

0.1.9.dev0

PB: plaid benchmarks VKI-LS59

Downloading the dataset

The dataset used for the competition is based on the VKI-LS59 blade and is provided in Plaid format:

- on Zenodo
- as a Hugginface dataset

Plaid, which stands for "Physics Learning AI Datamodel," enables the handling of heterogeneous physics datasets. A dataset and its associated learning problems are self-contained within a Plaid file. Details on accessing the information are provided below.

Using the dataset

To use the dataset, you need to install `plaid>=0.1`, which is available as a Conda package or from the sources.

In the competition, we use a subset of the dataset with the following specifications:

- **Splits**: "train" and "test"
- **Inputs**:
  - 2 scalars: `angle_in` and `mach_out`
  - Geometrical support (the mesh). Redondant information is available in the form of the signed distance function field `sdf`
- **Outputs**:
  - 6 scalars: `Q`, `power`, `Pr`, `Tr`, `eth_is` and `angle_out`
  - 2 fields: `mach` and `nut`

Note that the dataset does not contain outputs for the "test" split.

Retrieving Samples

Depending on whether you use the Zenodo or Hugging Face dataset, there are slight differences in how to retrieve the samples:

- **Hugging Face dataset**:

```
from datasets import load_dataset
from plaid.containers.sample import Sample
import pickle

hf_dataset = load_dataset("PLAID-datasets/VKI-LS59", split="all_samples")

ids_train = hf_dataset.description["split"]['train']
ids_test  = hf_dataset.description["split"]['test']

sample_train_0 = Sample.model_validate(pickle.loads(hf_dataset[ids_train[0]]["sample"]))
sample_test_0 = Sample.model_validate(pickle.loads(hf_dataset[ids_test[0]]["sample"]))
```

More information can be found in the Hugging Face support documentation for Plaid

- **Zenodo**:

```
from plaid.containers.dataset import Dataset
from plaid.problem_definition import ProblemDefinition

dataset = Dataset()
dataset._load_from_dir_('/path/to/plaid/dataset', verbose = True)

problem = ProblemDefinition()
problem._load_from_dir_('/path/to/plaid/problem_definition')

ids_train = problem.get_split('train')
ids_test  = problem.get_split('test')

sample_train_0 = dataset[ids_train[0]]
sample_test_0 = dataset[ids_test[0]]
```

Retrieving Data from Samples

Once samples are obtained, the following commands to retrieve the data are common to both cases:

```
# inputs
nodes = sample.get_nodes(base_name="Base_2_2")
elements = sample.get_elements(base_name="Base_2_2")
nodal_tags = sample.get_nodal_tags(base_name="Base_2_2")
sdf = sample.get_field("sdf", base_name="Base_2_2")
angle_in = sample.get_scalar("angle_in")
mach_out = sample.get_scalar("mach_out")

# outputs
mach = sample.get_field("mach", base_name="Base_2_2")
nut = sample.get_field("nut", base_name="Base_2_2")

for sn in ["Q", "power", "Pr", "Tr", "eth_is", "angle_out"]:
    outscalar = sample.get_scalar(sn)
```

- **Fields**: 1D arrays
- **Nodes**: 2D NumPy arrays
- **Elements**: Dictionaries with element type names as keys and 2D connectivity arrays as values
- **Nodal Tags**: Dictionaries with tag names as keys and 1D arrays of corresponding node indices as values

Figure 13. "Dataset" page of the benchmarking application on the `VKI-LS59` dataset.

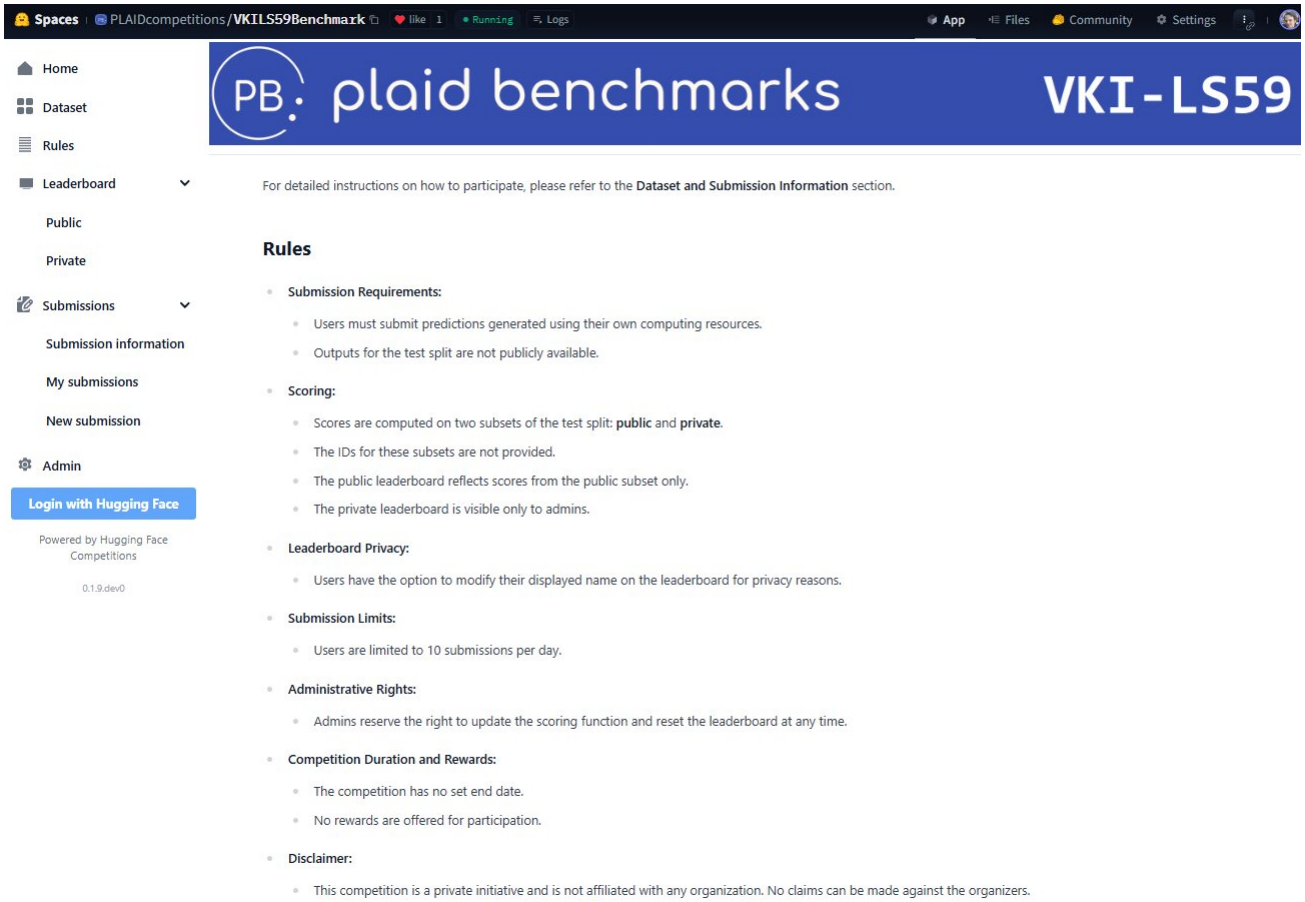


Figure 14. "Rules" page of the benchmarking application on the `VKI-LS59` dataset.

Figure 15. "Submission information" page of the benchmarking online application on the `VKI-LS59` dataset.

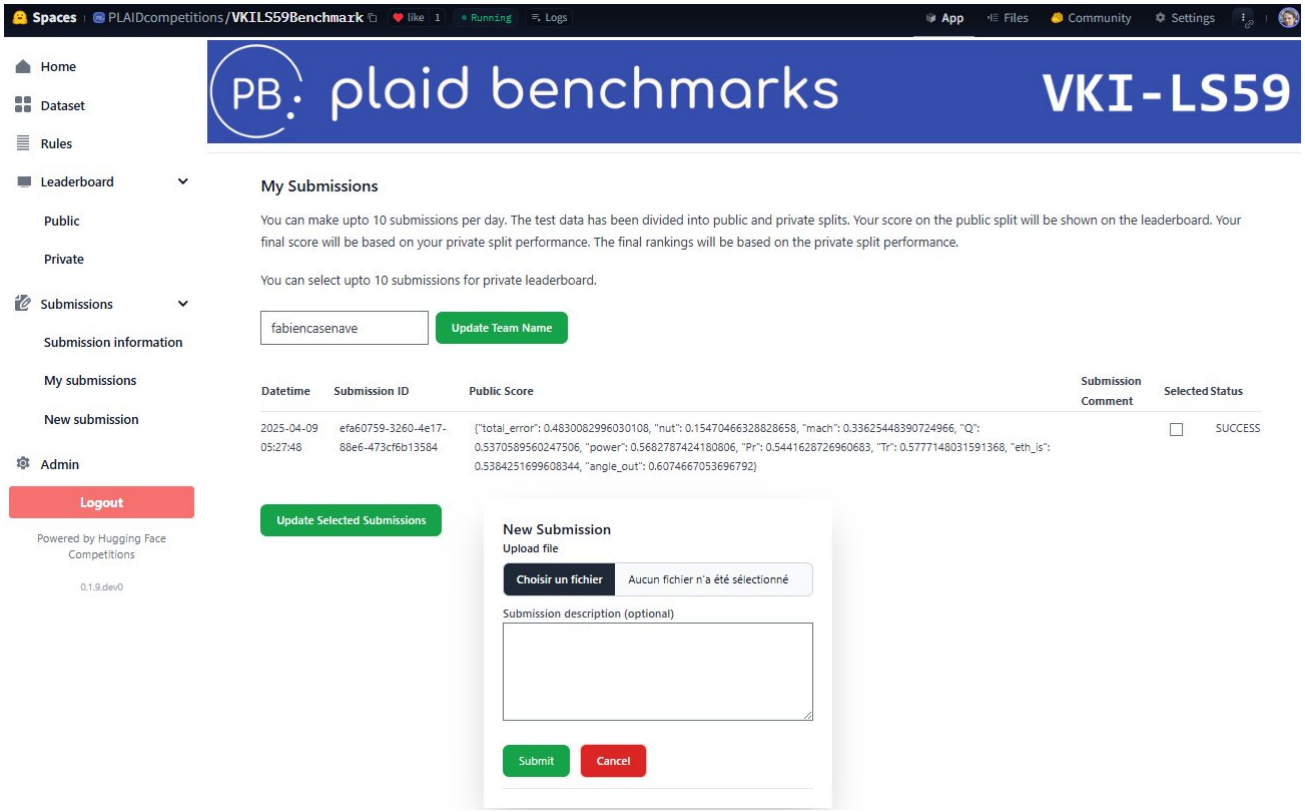
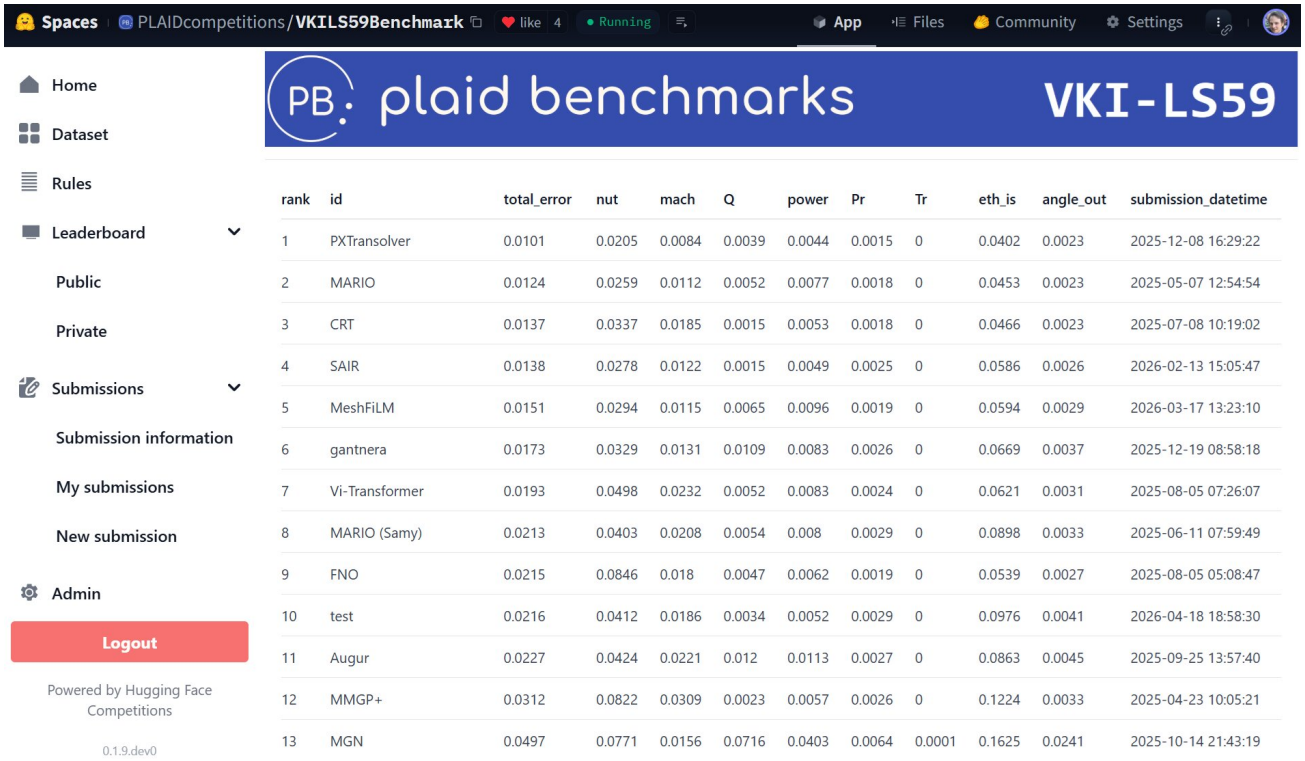


Figure 16. "My submissions" page of the benchmarking application on the `VKI-LS59` dataset.



rank	id	total_error	nut	mach	Q	power	Pr	Tr	eth_is	angle_out	submission_datetime
1	PXTransolver	0.0101	0.0205	0.0084	0.0039	0.0044	0.0015	0	0.0402	0.0023	2025-12-08 16:29:22
2	MARIO	0.0124	0.0259	0.0112	0.0052	0.0077	0.0018	0	0.0453	0.0023	2025-05-07 12:54:54
3	CRT	0.0137	0.0337	0.0185	0.0015	0.0053	0.0018	0	0.0466	0.0023	2025-07-08 10:19:02
4	SAIR	0.0138	0.0278	0.0122	0.0015	0.0049	0.0025	0	0.0586	0.0026	2026-02-13 15:05:47
5	MeshFiLM	0.0151	0.0294	0.0115	0.0065	0.0096	0.0019	0	0.0594	0.0029	2026-03-17 13:23:10
6	gantnera	0.0173	0.0329	0.0131	0.0109	0.0083	0.0026	0	0.0669	0.0037	2025-12-19 08:58:18
7	Vi-Transformer	0.0193	0.0498	0.0232	0.0052	0.0083	0.0024	0	0.0621	0.0031	2025-08-05 07:26:07
8	MARIO (Samy)	0.0213	0.0403	0.0208	0.0054	0.008	0.0029	0	0.0898	0.0033	2025-06-11 07:59:49
9	FNO	0.0215	0.0846	0.018	0.0047	0.0062	0.0019	0	0.0539	0.0027	2025-08-05 05:08:47
10	test	0.0216	0.0412	0.0186	0.0034	0.0052	0.0029	0	0.0976	0.0041	2026-04-18 18:58:30
11	Augur	0.0227	0.0424	0.0221	0.012	0.0113	0.0027	0	0.0863	0.0045	2025-09-25 13:57:40
12	MMGP+	0.0312	0.0822	0.0309	0.0023	0.0057	0.0026	0	0.1224	0.0033	2025-04-23 10:05:21
13	MGN	0.0497	0.0771	0.0156	0.0716	0.0403	0.0064	0.0001	0.1625	0.0241	2025-10-14 21:43:19

Figure 17. "Public leaderboard" page of the benchmarking application for the `VKI-LS59` dataset, as of April 19, 2026. Additional leaderboard entries have been submitted by external contributors.