Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A fully GPU-based workflow for building physics emulators of hypersonic flows

Fabian Paischer

paischer@ml.jku.at

JKU Linz / EMMI AI

Dylan Rubini

EMMI AI

Deniz Bezgin

Technical University of Munich https://orcid.org/0000-0001-5299-3102

Aaron Buhendwa

Technical University of Munich

David Hauser

EMMI AI

Florian Sestak

JKU Linz / EMMI AI

Johannes Brandstetter

JKU Linz / EMMI AI

Sebastian Kaltenbach

EMMI AI

Nikolaus Adams

Technical University of Munich https://orcid.org/0000-0001-5048-8639

Article

Keywords:

Posted Date: July 31st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10150662/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: There is **NO** Competing Interest. There are no competing interests.

A fully GPU-based workflow for building physics emulators of hypersonic flows

Fabian Paischer[*2,3] **Dylan Rubini**[*3] **Deniz A. Bezgin**[1] **Aaron B. Buhendwa**[1]
David Hauser[3] **Florian Sestak**[2,3] **Johannes Brandstetter**[2,3] **Sebastian Kaltenbach**[3]
Nikolaus A. Adams[1]

[1] Chair of Aerodynamics and Fluid Mechanics, TU Munich, Germany
[2] ELLIS Unit, Institute for Machine Learning, JKU Linz
[3] EMMI AI, Linz

Abstract

The ability to resolve complex physical phenomena with high fidelity and at low computational cost is central to addressing key challenges in modern engineering. A prime example lies in hypersonic flows, where the precise prediction of the flowfield topology, in particular with respect to shock wave location and intensity, is critical. Yet hypersonic flows continue to be a stumbling block for traditional reduced-order models and neural emulators that struggle to capture steep gradients in flow states with physical consistency in applications of industrial relevance. To that end, we introduce a fully GPU based workflow that integrates accelerated data generation with the training of neural emulators augmented by uncertainty quantification and physics-aware refinement. Our workflow is enabled by a differentiable high-fidelity solver (JAX-Fluids) which we employ for rapid data generation and residual-based improvement of the neural emulator to enhance physical consistency. Residual-based refinement sidesteps the need for costly data generation enabling training on cases where only mesh and input parameters are available while substantially reducing residuals and improving physical consistency. Together, differentiable simulation and residual-based refinement yield physics emulators that remain reliable beyond their training distribution, a key requirement for deploying surrogates in real-world engineering design loops.

1 Introduction

Fluid dynamics is fundamental to many processes in nature and technology, and its numerical simulation routinely ranks among the top-level compute-resource allocations at Tier 0 computing facilities [SKA+14]. Despite its complex and inherently multi-scale character, flow fields exhibit coherent behavior characterized by universal dependencies. A long-standing challenge for predictive simulation, particularly in high-speed transport and propulsion, is the simultaneous presence of multi-scale flow structures such as coherent vortices, eddies, and shocks. Shockwaves are among the most consequential of these phenomena, governing processes from the evolution of galaxies [MH80] to the feasibility of high-speed propulsion [Urz18]. They are characterized by discontinuities in the macroscopic flow state, such as extreme gradients in pressure, density, temperature, and momentum, which arise when the local flow speed $\|\mathbf{u}(\mathbf{x}, t)\|$ exceeds the local speed of sound $a(\mathbf{x}, t)$, i.e., when the local Mach number $Ma(\mathbf{x}, t) = \|\mathbf{u}\|/a \geq 1$.

Hypersonic flowfield predictions are particularly challenging for both numerical simulation and data-driven modeling. They are characterized by Mach numbers beyond approximately five and

[*]Equal contribution, corresponding author: Fabian Paischer (paischer@ml.jku.at)

Preprint.

exhibit strong shock interactions, high-enthalpy effects, and stringent conservation requirements. Such flow problems have historically served as driving applications for high-performance computing, motivating both methodological advances and community benchmarks [WRLB+25, RHH+13]. The computational cost of numerical simulations has made data-driven alternatives to classical computational fluid dynamics (CFD) a foremost research interest [BNK20, BEF19, KKL+21]. These range from full surrogate substitution of the numerical solver via Physics Emulators (PEs) to machine-learned (ML) acceleration of existing CFD methodologies.

In this work, we present a fully GPU-based workflow for hypersonic flows enabled by the differentiable finite-volume solver JAX-Fluids [BBA23, BBA25]. The proposed workflow encompasses (i) GPU-accelerated data generation, (ii) pre-training of neural emulators, and (iii) target-free, residual-based refinement. Data generation is based on a Cartesian multi-block mesh that enables efficient parallelization on GPU and seamless integration with various ML architectures and training paradigms. For pre-training, we investigate two complementary architectures, namely irregular-grid based AB-UPT [ABK+25] and regular-grid based vision transformer [ViT, DBK+21]. Furthermore, we investigate the trade-off between deterministic and probabilistic training paradigms. We conduct scaling studies with respect to both model size and dataset size for all architectures and training paradigms. For target-free residual-based refinement the pre-trained PE generates a candidate solution which is evaluated against the residual of the underlying partial differential equation (PDE) computed by the differentiable solver. The resulting gradient signal is backpropagated into the emulator weights without requiring any target flow fields, mirroring exactly the numerical discretization used during data generation.

Our experiments reveal several key findings for architecture and training paradigm selection and refinement for PEs in the hypersonic regime. Among the two architectures, AB-UPT achieves the highest accuracy in data-abundant settings, while the ViT outperforms in data-scarce regimes due to the strong inductive bias provided by its regular-grid structure. Flow matching trades point-wise accuracy for generative modeling capability, but provides off-the-shelf uncertainty and acts as implicit data augmentation, yielding a smaller gap between in-distribution and out-of-distribution performance than either deterministic architecture. For physics-aware refinement, we find that backpropagation through the PDE residual leads to substantial reductions in conservation residuals with little to no changes in field-level accuracy, suggesting that the pre-trained models already capture the dominant flow structure and the refinement primarily corrects local physical consistency. Notably, the target-free setup, which conditions on the computational mesh and input parameters without requiring reference flow fields exhibits the largest improvement in residuals.

Overall our contributions are as follows.

- We present a fully GPU-based workflow for hypersonic flow emulation that integrates data generation, surrogate pre-training, and physics-aware refinement within a single differentiable pipeline built on the JAX-Fluids solver.
- We enable regular-grid ML architectures to be agnostic to grid topology parameters (e.g., block count and ordering of the block-structured meshes) by combining absolute and relative positional encodings based on coordinates in physical space.
- We evaluate two complementary neural architectures (AB-UPT and ViT) and two training paradigms (deterministic vs probabilistic) and conduct scaling studies with respect to both model size and dataset size, identifying distinct data-efficiency and accuracy trade-offs across regimes.
- We introduce a target-free refinement stage that improves physical consistency by backpropagating PDE residuals into the pre-trained neural PE weights without requiring reference flow fields, and demonstrate its advantage over field-value fine-tuning.

2 Related Work

Machine learning has been integrated into CFD workflows in several complementary ways. One line of work augments classical numerical schemes with learned components. Neural networks (NNs) have served as troubled-cell indicators that locate where limiting is needed in high-order discretizations [RH18], as local and parsimonious modifications within physics-constrained implicit Large-Eddy Simulation (LES) that lead to modifications of classical shock-capturing schemes [BBSA25], and as learned correction operators that recover fine-grid accuracy from coarse-grid solvers [KSA+21].

A related line uses reinforcement learning to discover effective closures [NDLK21, FKL+25]. A third direction bypasses the solver entirely and trains neural networks as end-to-end PEs, e.g., deep convolutional models that map airfoil geometry directly to Reynolds-averaged Navier–Stokes (RANS) fields [TWPH20]. A common enabler across all of these settings is the availability of a state-of-the-art CFD solver with algorithmic differentiation capability for end-to-end pipelines [BBA23, BBA25].

A more general formulation seeks to learn the parameter-to-solution operator of a PDE as a mapping between function spaces. Neural operators provide a scalable and resolution-invariant framework for learning mappings and offer orders-of-magnitude speedups over traditional numerical solvers [LKA+20, LJP+21, AKL+24]. Additionally, for unstructured discretizations, mesh-based graph networks exploit the adjacency structure of the simulation grid to learn local update rules [PFSGB20]. Recently, purely transformer-based [VSP+17] formulations [AFS+24, ABK+25] have shown to successfully scale to industry relevant complexity and effectively capture long range dependencies. Despite their flexibility, neural PEs still face well-documented practical limitations, for example heavy data requirements, sensitivity to training-distribution shifts, lack of rollout robustness, and the absence of guaranteed physical consistency in complex regimes [VB22]. Most of the current work therefore has focused on optimizing model errors, for instance RANS equations [GD26]. Questions of scalability and applicability beyond such regimes, including hypersonic flight, remain open.

Although small-scale fluctuations in velocity, pressure, and density can in principle be resolved on sufficiently fine grids, shocks remain genuine discontinuities and require dedicated nonlinear schemes [Tor09]. Shocks are weak solutions of the underlying flow equations and obey precise jump relations between pre- and post-shock states, namely the Rankine–Hugoniot conditions [LeV92]. These properties make reliable, high-resolution prediction of shocks and shock interactions challenging even for classical schemes and notoriously difficult for purely data-driven surrogates. Only with the explicit inclusion of inductive biases for physical consistency have physics-informed neural networks [KKL+21] delivered predictions with correct shock locations, satisfied jump conditions, and maintained positivity of the flow state [MJK20, JMAK22]. The optimization of a discrete loss provided by a numerical discretization of the governing equations on a chosen mesh has more recently proven sufficiently effective to address inverse inference problems in three-dimensional steady-state transonic and supersonic flows [BBK+25, PCD+25], although in that setting some flow-field data must still be available.

For generative tasks, denoising diffusion models have begun to be explored in fluid mechanics. They produce sample-diverse predictions and naturally provide a posterior from which uncertainty can be estimated [HJA20]. Diffusion models enable fast forecasting of distributional quantities of interest in high-dimensional dynamical systems [GKK24, MLR+24]. In incompressible turbulence, diffusion models have generated physically plausible three-dimensional flow states from scratch [LLHPG24] and delivered calibrated uncertainty for airfoil flows over a range of Reynolds numbers and angles of attack [LT24]. In the compressible regime, denoising diffusion models have been examined for moderately supersonic flow [AA25]. However, a fully GPU-based workflow that combines parameterized data generation, neural emulator training across complementary architectures, uncertainty quantification, and physics-aware refinement for complex hypersonic flows has not yet been demonstrated.

3 The Neural Physics Emulator Pipeline

Drawing inspiration from the success of Large Language Models (LLMs) in natural language processing [AAA+23, TAB+23], we propose a workflow for building PEs that ranges from data generation to pre-training and fine-tuning. While LLMs excel at modeling linguistic structures, a PE is specifically designed to learn the complex functional relationships of a physical system. Despite their differences, both LLMs and PEs can be based on the same underlying attention mechanism [VSP+17].

We deliberately choose the term *Physics Emulator* to distinguish this work from generic black-box regression models. The term *Physics* signals that the systems of interest are physical and not just arbitrary input-output mappings. The term *Emulator* is adopted from the statistical literature on computer experiments, where it denotes a fast, probabilistic surrogate trained on the outputs of a computationally expensive simulator [SWMW89, KO01]. In that tradition, an emulator is not merely a curve fit, it is a high-fidelity statistical proxy designed to reproduce the full input-output behavior of

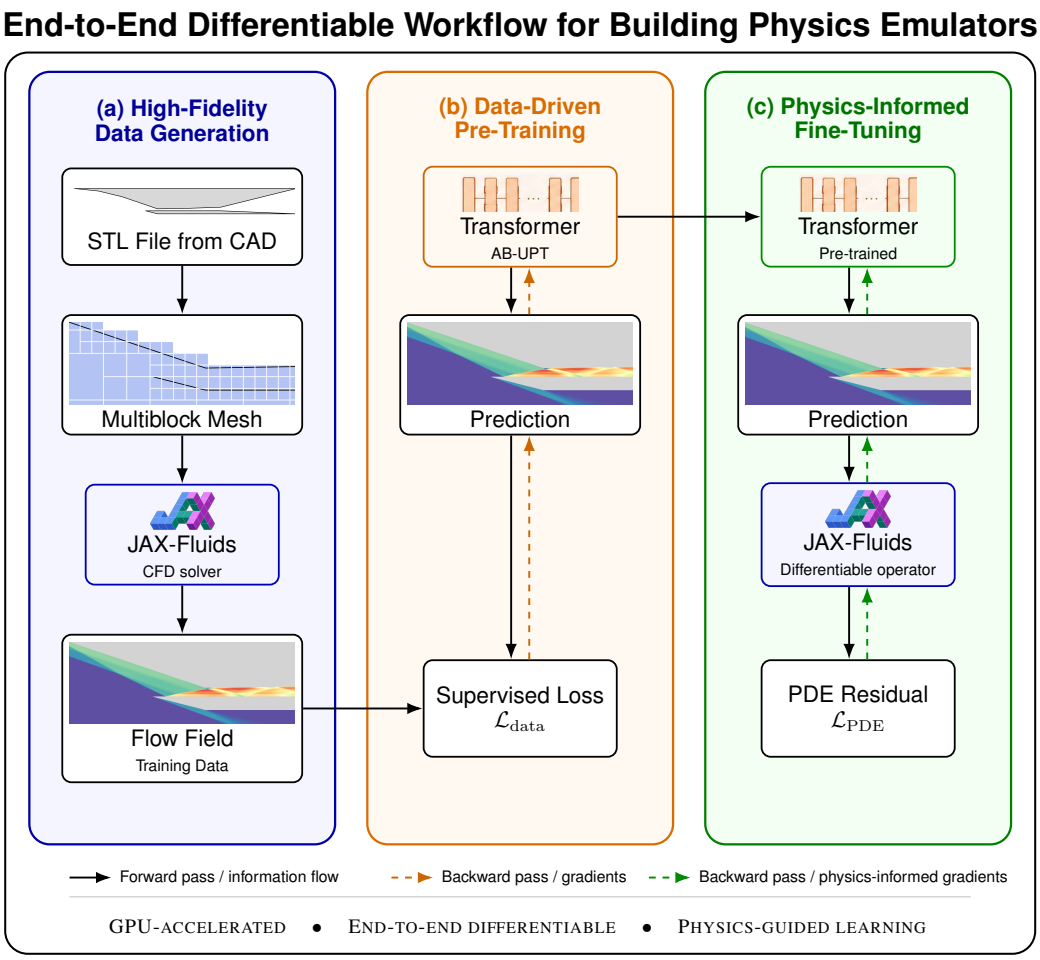


Figure 1: A fully GPU-accelerated and end-to-end differentiable workflow for constructing physics emulators of complex flow phenomena. The workflow consists of three stages. (a) High-Fidelity Data Generation: Starting from an STL representation of the geometry, a multi-block mesh is automatically generated. JAX-Fluids then performs high-fidelity CFD simulations until a steady-state solution is obtained, yielding the training dataset. (b) Data-Driven Pre-Training: Physics emulators are pre-trained in a supervised manner using the high-fidelity dataset generated in stage (a). (c) Physics-Informed Fine-Tuning: After pre-training, the emulators are refined in a target-free manner by minimizing the residuals of the governing equations. The residuals are computed by evaluating the differentiable JAX-Fluids solver on the model predictions, enabling end-to-end gradient-based optimization through the model and the CFD solver itself.

the underlying code. Consequently, a Physics Emulator is such a surrogate purpose-built for physical simulators, functioning as a drop-in replacement for solvers such as CFD, mapping input parameters to complete physical fields in milliseconds rather than hours.

To effectively serve as a neural surrogate for hypersonic applications, we identify the following desirable characteristics for a PE, the first two of which we consider essential:

- **Differentiability**: The model must support end-to-end automatic differentiation for seamless integration into gradient-based design optimization and for physics-based fine-tuning using differentiable CFD.

- **Physical consistency**: Predictions must respect the conservation laws of mass, momentum, and energy as much as possible.

- **Uncertainty capabilities:** When predictive uncertainty is relevant for the downstream task, the PE should be able to provide a posterior distribution that can be sampled to yield uncertainty estimates, rather than only point predictions.

To construct a Physics Emulator that satisfies these requirements, we propose a fully GPU-resident pipeline comprising three phases, see Figure 1:

1. **Data Generation:** First, data generation requires defining a robust design space. This involves parameterizing the geometries, material properties and boundary conditions of the target engineering system. By establishing a comprehensive parametric envelope, we ensure the resulting model will be exposed to a diverse and representative set of physical scenarios. Based on the parameterized setup, high-fidelity data must be generated, which requires a scalable and accurate solver.
2. **Model Training:** Given a dataset of high-fidelity simulations, we pre-train a neural PE. This step includes the selection of a suitable model architecture and training process. For instance, in case predictive uncertainty should be quantified a probabilistic framework is required.
3. **Model Fine-tuning:** The base model can be fine-tuned based on specific quantities of interest (e.g., conservation laws). These quantities can be chosen after pre-training and can be done in a target-free manner.

The following sections discuss all three phases in detail:

3.1 Data Generation: JAX-Fluids

Within the scope of this work, high-fidelity flow-field data are generated using JAX-Fluids [BBA23, BBA25], a high-order, fully differentiable finite-volume solver for compressible single- and two-phase flows. JAX-Fluids combines high-order shock-capturing discretizations, GPU-acceleration, automated Cartesian multi-block meshing, and end-to-end automatic differentiation within a single JAX-based framework, making it particularly well suited for training of PEs. In particular, these properties enable using the solver not only as an offline data generator, but also as a differentiable physics engine during model training and downstream optimization.

Hypersonic flows are characterized by complex flow phenomena like shock-shock interactions, shock-interface interactions, wave dynamics, viscous-inviscid interactions, flow separation, and multi-species effects. Accurately resolving these phenomena requires numerical methods that are both robust in the presence of discontinuities and sufficiently accurate in smooth regions of the flow. In this work, the data are generated in the inviscid limit governed by the compressible Euler equations which corresponds to typical application scenarios of high Reynolds numbers. At hypersonic Mach numbers considered here, the intake flowfield and the integral performance metrics are dominated by the shock structure which the Euler equations admit as weak solutions satisfying Rankine–Hugoniot jump conditions. For hypersonic intakes operating at flight Reynolds numbers, viscous regions are confined to relatively thin boundary layers and inviscid analysis is the standard scope for preliminary scramjet-inlet design [HP94]. The PE and refinement methodology described in subsequent section is not specific to the Euler equations and extends to the full Navier–Stokes system by adding viscous effects.

JAX-Fluids follows a high-order Godunov-type finite-volume formulation. Shock waves are captured using nonlinear solution-adaptive reconstruction together with approximate Riemann solvers. For the cases considered in this work, steady-state solutions are obtained by explicit time advancement until the residuals of the governing equations fall below a prescribed tolerance. The design of JAX-Fluids is motivated by the requirements of ML workflows for computational physics. In particular, three aspects are central to the present work: Cartesian multi-block meshes, GPU acceleration, and automatic differentiability.

Cartesian multi-block mesh. A central requirement for large-scale training-data generation is a mesh-generation procedure that is robust, automated, and computationally efficient. In general, CFD mesh strategies can be classified as follows: structured versus unstructured meshes, and body-fitted versus immersed boundary methods. Unstructured body-fitted meshes offer high geometric flexibility and are therefore well suited for complex configurations, but high-order methods are difficult to

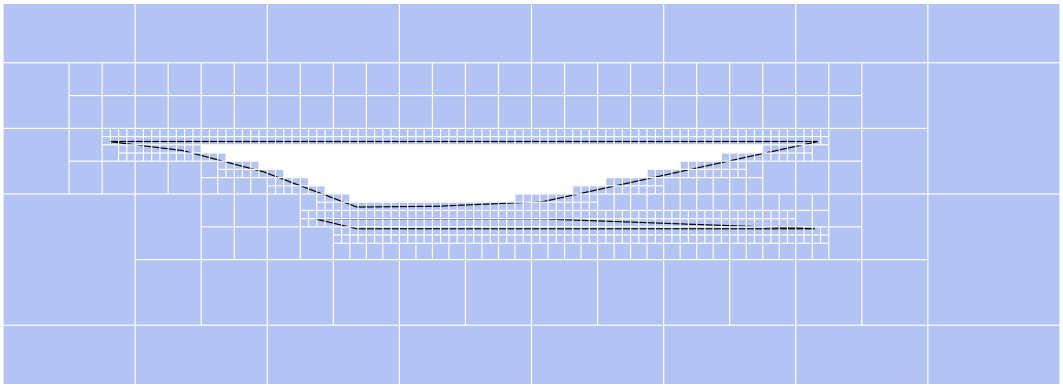


Figure 2: Schematic of the multi-block grid for the generic scramjet demonstrator configuration.

implement efficiently due to irregular cell connectivity and thus indirect memory access. Structured body-fitted meshes, such as curvilinear grids, are more favorable for high-order schemes because they retain regular cell connectivity, but complex geometries with fine geometric features are difficult to represent.

In this work, we therefore employ a structured Cartesian multi-block mesh combined with a conservative cut-cell immersed-boundary method. The geometry is represented implicitly by a level-set function, and cells intersected by the fluid-solid interface are treated as cut cells. This approach combines the geometric flexibility of immersed boundary methods with the numerical efficiency of structured Cartesian grids. In particular, it enables automated meshing of fine geometric features while retaining the regular data layout and stencil structure required for efficient high-order finite-volume discretizations. The multi-block formulation enables local refinement through quadtree- or octree-type subdivision, allowing high resolution near solid boundaries and interfaces while keeping a coarser resolution elsewhere. Figure 2 shows the multi-block grid of a representative geometry considered in this work.

The Cartesian multi-block grid is particularly advantageous for GPU-accelerated simulation. The solution variables within each block are stored as dense multidimensional arrays, leading to regular memory access, efficient vectorization, and reduced indirect addressing compared with unstructured grids. As a result, the dominant numerical kernels are well matched to modern accelerators, where performance depends strongly on data locality, coalesced memory access, and high arithmetic throughput.

GPU-acceleration. JAX-Fluids is implemented in JAX and compiled through XLA for execution on modern accelerators. Within each block, the solver operations are expressed as batched array operations, enabling large numbers of cells to be processed in parallel. The multi-block decomposition also provides a natural parallelization strategy: a set of individual blocks can be assigned to different XLA devices, while communication is limited to halo exchanges. In JAX-Fluids, this is enabled through JAX primitives such as `jax.shard_map` and `jax.lax.ppermute`. This GPU-resident design is essential for generating large datasets over broad parameter spaces, including geometry and Mach number. It is also important when JAX-Fluids is used inside the training loop to evaluate physics-based losses.

Differentiable solver. JAX-Fluids is a fully differentiable solver that allows calculation of gradients of objective functions by automatic differentiation. These gradients are consistent with the discretized PDEs, i.e., they are consistent with the governing equation and the chosen numerical discretization. Flow field predictions of a trained PE can be passed through JAX-Fluids to fine-tune it for achieving high-fidelity, physically-aware predictions in complex flows without inconsistencies with the data-generating numerics.

Table 1: Qualitative comparison of neural physics emulator instantiations. ✓ = fully satisfied, ✗ = not satisfied. Different methods and training paradigms exhibit different advantages.

Characteristic	**AB-UPT** [ABK+25]	**ViT** [DBK+21]	**Flow Matching** [LCBH+22]
Point cloud	✓	✗	✗
Block-stacked regular grid	✗	✓	✓
Native UQ support	✗	✗	✓
Single-pass inference	✓	✓	✗

3.2 Neural Architectures for Physics Emulators

We investigate two neural architectures and two different modeling paradigms, each offering distinct trade-offs for predicting hypersonic flowfields on octree-based Euclidean mesh data. The octree structure decomposes the domain into axis-aligned blocks of uniform resolution, i.e., the count and ordering of blocks vary between cases. This meshing strategy is increasingly adopted by modern GPU-based solvers because it maps naturally to parallel hardware [JES26, CSS+25], making scalable emulation of such grids a broadly relevant objective. To enable regular-grid architectures to be invariant to block count and ordering, we encode each patch's physical coordinates using complementary absolute (sinusoidal) and relative (rotary) positional encodings. Combined, they give a single global attention a complete picture of patch location and pairwise displacement across blocks. In contrast, architectures based on point-wise representations can simply ingest the raw point cloud. This enables a range of neural architectures, and we consider the following instantiations of the PE: (i) a field--based approach designed for irregular grids [ABK+25, AB-UPT], (ii) a regular-grid-based vision transformer [DBK+21, ViT], (iii) a generative flow matching model operating on the regular-grid representation [LCBH+22].

Table 1 summarizes the qualitative trade-offs among the three approaches. AB-UPT treats the mesh as an irregular point cloud and yields smooth predictions but does not have built-in uncertainty estimates. The ViT, by contrast, ingests the regular-grid octree representation and benefits from the highly optimized attention kernels available for uniform tensor data. The flow matching model is also based on a regular-grid, but replaces the deterministic prediction head with a stochastic denoising process, trading single-pass efficiency for predictive uncertainty estimates.

Each model takes as input the simulation grid, either as a point cloud (AB-UPT) or in the block-stacked regular-grid representation (ViT, Flow Matching), together with a conditioning vector comprising the 15 geometry parameters of the scramjet configuration and the free-stream Mach number M_∞. The neural PE then predicts the corresponding flow field as a multi-channel output with C=8 channels, where each channel represents a different physical quantity, namely pressure p, density ρ, velocity $\mathbf{u}$, temperature T, enthalpy h, total pressure P_t, kinetic energy k, and Mach number Ma. The model is trained by minimizing the mean squared error over all grid points,

$$\mathcal{L}_{\text{data}} = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|^2 , \tag{1}$$

where $\mathbf{y}_i \in \mathbb{R}^C$ is the ground-truth flow state at grid point i, $\hat{\mathbf{y}}_i$ is the corresponding prediction, and N is the number of grid points. Predicting primitive (i.e., density, velocity, and pressure) and derived quantities (i.e., temperature, enthalpy, total pressure, kinetic energy, and Mach number) jointly avoids the error accumulation that arises when derived fields are reconstructed from predicted primitives (see Section I). The trade-off is that the predicted derived quantities are not guaranteed to be consistent with those recomputed from the predicted primitives. However, supervising only the primitives would leave the derived fields without a direct training signal and folding their computation into the loss introduces potentially ill-conditioned gradients through the nonlinear derivations as well as additional weighting hyperparameters across heterogeneous scales.

3.3 Physics-aware Model Refinement

The third phase of our workflow refines the pre-trained base model through physics-aware optimization to facilitate physical consistency. This stage exploits the end-to-end differentiability of the JAX-Fluids solver to embed the governing flow equations directly within the ML optimization loop. Whereas the initial training phase relies on a standard supervised loss the refinement phase minimizes the point-wise discrete residuals of the compressible Euler equations

$$\mathcal{L}_{\text{PDE}} = \sum_{(i,j)\in\Omega} \sum_{k=1}^{4} w_k \left(R_{i,j}^k\right)^2 \Delta x \Delta y, \tag{2}$$

where $R_{i,j}^k$ denotes the point-wise residual for the k-th conserved quantity in cell (i, j). w_k is the corresponding loss weight. Here, the conserved quantities comprise mass, x-momentum, y-momentum, and total energy. For details we refer to Section F.4.

Using JAX-Fluids, we evaluate the residual-based loss $\mathcal{L}_{\text{PDE}}$ on the simulation mesh with the same numerical discretizations employed during data generation, ensuring consistency between training and inference. A key advantage of this approach is that it improves generalization and can be performed in a target-free manner. It requires only a differentiable numerical solver and simulation meshes for the configurations of interest, i.e., no additional ground-truth solutions are needed. The configuration space can therefore be cheaply expanded beyond that of the training data since the loss is computed directly from the flow equations. This makes our model refinement a comparatively inexpensive fine-tuning procedure that avoids the cost of running new high-fidelity simulations.

4 Experiments

This section details our experimental findings. We begin with an overview of the two datasets utilized for model training. We then present the training details and results of the baseline model, contrasting the various methodological approaches introduced previously. Within this analysis, we focus on scalability and provide results for scaling along the data and model axis. Finally, we present the results of the model refinement phase.

4.1 Datasets

The experimental evaluation utilizes two distinct datasets of scramjet configurations in hypersonic conditions, $D1$ and $D2$, both generated using JAX-Fluids as detailed in Section 3.1. Each scramjet case is parameterized by its design-parameter vector $\mathbf{p}_i \in \mathbb{R}^d$ (see Figure 3), where $\mathbf{p}_i$ concatenates geometry and inflow parameters of case i. $D1$ represents a high-fidelity dataset, whereas $D2$ is target-free, i.e., it only comes with mesh and input parameters without ground-truth field data. For the pre-training stage of our models, we rely on $D1$, whereas for fine-tuning we also leverage $D2$. We list both datasets and their corresponding properties in Table 2.

Table 2: Overview of the two datasets used for model training and evaluation. D_1 prioritizes numerical accuracy through minimized residuals, while D_2 comprises only meshes for new scramject parameterizations.

Property	**D1**	**D2**
Parameters	✓	✓
Mesh	✓	✓
Field-data	✓	✗
Generation time	Steady-state simulation	Only meshing
Number of samples	7081	7845
Primary use	Accuracy	Refinement

To evaluate the different neural PEs we construct different dataset splits. Stacking the design-parameter vectors of all cases considered yields the parameter matrix $\mathbf{P} \in \mathbb{R}^{N\times d}$. First, we fit an Isolation Forest [LTZ08] with $T = 100$ trees on the parameter matrix $\mathbf{P}$ and obtain an anomaly score $s_i = \text{decision_function}(\mathbf{p}_i)$ for each case in the dataset, where s_i corresponds to points that are

isolated by shorter random partitioning paths and are therefore marked anomalous in design space. We select 10% of cases with the lowest scores and assemble them into an out-of-distribution (OOD) set. The remaining in-distribution pool is then randomly partitioned into train/val/test according to a 80/10/10 split. We verify the construction post-hoc by visualizing the anomaly scores of the Isolation Forest per split (see Figure 12 in Section E), confirming that OOD samples lie on the periphery of the parameter manifold while train/val/test overlap in the interior.

4.2 Evaluation Protocol

We evaluate each PE against the reference JAX-Fluids simulations on three engineering key performance indicators (KPIs) that quantify aspects of scramjet performance, supplemented by a qualitative comparison of the density field. They are derived from the full predicted flow, so that they expose the downstream consequences of the predictions, such as total-pressure recovery up to the inlet to combustor, cumulative total-pressure loss across the scramjet from the inlet plane to the nozzle exit, and peak thermal load on the wetted geometry.

Total-pressure ratio π_d. Following standard inlet-performance practice, we define

$$\pi_d = \frac{\langle p_t|_{x_{\text{stn2}}}\rangle}{\langle p_t|_{x_{\text{inlet}}}\rangle} \tag{3}$$

i.e., the ratio of mass-flux-weighted total pressure between the inlet plane and station 2, taken at the axial mid-point of the scramjet and restricted to the fluid side of the embedded geometry by the signed-distance level set. An ideal isentropic compression delivers $\pi_d = 1$, whereas losses (e.g., due to shocks) between capture surface and entrance of the combustor reduce π_d.

Total-pressure loss ratio Λ. Whereas π_d characterizes the inlet up to station 2, the total-pressure loss ratio

$$\Lambda = \frac{\langle p_t|_{x_{\text{inlet}}}\rangle - \langle p_t|_{x_{\text{exit}}}\rangle}{\langle p_t|_{x_{\text{inlet}}}\rangle} \tag{4}$$

captures the cumulative loss across the entire duct, from the inlet plane to the nozzle exit. The two pressure-based KPIs are deliberately complementary.

Peak surface temperature. The third KPI is a structural-design figure of merit, namely the peak temperature on the scramjet wetted surface. From the predicted temperature fields we extract the peak surface temperature and report the 95th percentile. Peak wall temperature in a hypersonic intake drives the choice of thermal-protection material and the cooling-system budget for the vehicle.

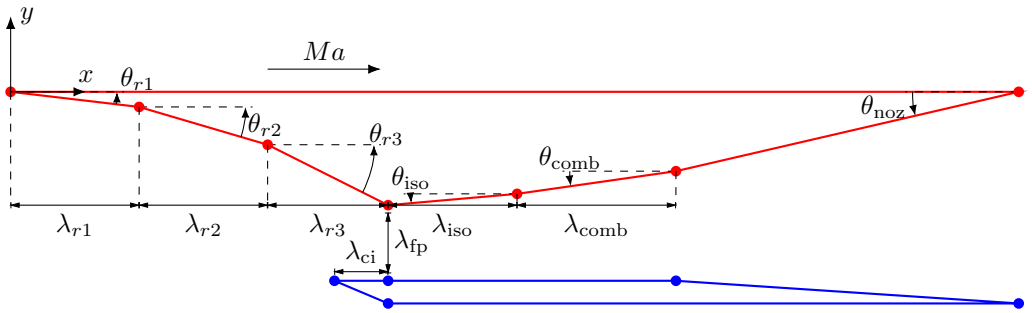


Figure 3: **Schematic of the parametrized scramjet geometry illustrating the design-parameters $\mathbf{p}$.** Here, $\lambda_{r1}, \lambda_{r2}, \lambda_{r3}$ are the intake ramp length fractions. λ_{iso} and λ_{comb} denote the isolator and combustor length fractions. $\theta_{r1}, \theta_{r2}, \theta_{r3}$ are the intake ramp angles, and $\theta_{\text{iso}}, \theta_{\text{comb}}, \theta_{\text{noz}}$ the isolator, combustor, and nozzle angles, all measured relative to the x-axis. λ_{fp} is the flow path height fraction and λ_{ci} is the cowl intake length fraction. The length fractions are defined with respect to a fixed reference length and the cowl has a fixed height. In addition to the geometry parameters, the design parameters $\mathbf{p}$ also contains the inflow Mach number Ma.

Table 3: **Performance of different physics emulators on hypersonic flowfields.** Relative L2 errors (%) for pressure p, density ρ, velocity $\boldsymbol{u}$, enthalpy h, total pressure p_t, kinetic energy k, temperature T, and Mach number Ma for AB-UPT, ViT, and Flow Matching (FM) across three random seeds.

Split	Model	p	ρ	$\boldsymbol{u}$	h	p_t	k	T	Ma
Val	AB-UPT	$2.16_{\pm 0.015}$	$1.77_{\pm 0.013}$	$0.38_{\pm 0.000}$	$1.42_{\pm 0.001}$	$2.33_{\pm 0.028}$	$1.76_{\pm 0.019}$	$1.42_{\pm 0.001}$	$0.81_{\pm 0.006}$
	ViT	$2.86_{\pm 0.017}$	$2.24_{\pm 0.015}$	$0.48_{\pm 0.005}$	$1.65_{\pm 0.021}$	$2.62_{\pm 0.043}$	$2.15_{\pm 0.018}$	$1.65_{\pm 0.021}$	$0.95_{\pm 0.015}$
	FM	$8.64_{\pm 1.211}$	$6.15_{\pm 0.787}$	$1.13_{\pm 0.099}$	$3.13_{\pm 0.202}$	$3.65_{\pm 0.144}$	$5.03_{\pm 0.536}$	$3.13_{\pm 0.202}$	$1.55_{\pm 0.053}$
Test	AB-UPT	$2.14_{\pm 0.018}$	$1.76_{\pm 0.014}$	$0.38_{\pm 0.001}$	$1.43_{\pm 0.009}$	$2.33_{\pm 0.008}$	$1.75_{\pm 0.022}$	$1.43_{\pm 0.009}$	$0.82_{\pm 0.005}$
	ViT	$2.84_{\pm 0.038}$	$2.24_{\pm 0.023}$	$0.48_{\pm 0.005}$	$1.68_{\pm 0.014}$	$2.63_{\pm 0.021}$	$2.16_{\pm 0.016}$	$1.68_{\pm 0.014}$	$0.96_{\pm 0.006}$
	FM	$8.51_{\pm 1.263}$	$6.06_{\pm 0.831}$	$1.12_{\pm 0.113}$	$3.12_{\pm 0.222}$	$3.68_{\pm 0.111}$	$4.97_{\pm 0.559}$	$3.12_{\pm 0.222}$	$1.56_{\pm 0.051}$
OOD	AB-UPT	$3.13_{\pm 0.038}$	$2.41_{\pm 0.042}$	$0.51_{\pm 0.010}$	$1.78_{\pm 0.028}$	$2.77_{\pm 0.058}$	$2.23_{\pm 0.046}$	$1.78_{\pm 0.028}$	$0.99_{\pm 0.008}$
	ViT	$4.01_{\pm 0.054}$	$2.99_{\pm 0.043}$	$0.62_{\pm 0.008}$	$2.03_{\pm 0.026}$	$2.95_{\pm 0.022}$	$2.69_{\pm 0.029}$	$2.03_{\pm 0.026}$	$1.13_{\pm 0.011}$
	FM	$9.89_{\pm 1.375}$	$6.97_{\pm 0.892}$	$1.28_{\pm 0.112}$	$3.55_{\pm 0.225}$	$4.11_{\pm 0.123}$	$5.63_{\pm 0.565}$	$3.55_{\pm 0.225}$	$1.75_{\pm 0.043}$

4.3 Pre-Training Results

We present results for the three different instantiations of PEs and their performance in the following sections. Table 3 summarizes the relative L_2 errors of all fields across the validation, test, and OOD splits. All of the results in this section are exclusively based on training on $\mathcal{L}_{\text{data}}$ and cover a two different architectures and modeling paradigms within a typical toolbox for providing PE solutions for engineering problems.

4.3.1 AB-UPT

As illustrated in Figure 4, the model maintains high fidelity across the relevant metrics. Parity plots for π_d, Λ, and peak surface temperature (Figure 4a-c) show tight clustering along the identity line. Furthermore, the density field predictions (Figure 4e) closely mirror the ground-truth solver (Figure 4d). The relative error map (Figure 4f) reveals that the highest discrepancies are localized at shockwaves and the exhaust nozzle with an average relative L_2 error around 3%. Finally, we report performance metrics in Table 4 showing that training and inference speed for AB-UPT is slower than other methods. Specifically, training sequences are longer due to the selection of anchor points and inference speed is much slower as AB-UPT decodes field values for each grid position during a single forward pass. This also results in elevated GFLOPs and memory footprint.

Additionally, we analyze data and model scaling as shown in Figure 5. For data scaling, we train a model with $\sim 25M$ parameters on subsampled training splits. The different splits comprise $\{50, 150, 450, 1200, 2800, 4816\}$ samples using farthest point sampling to maintain coverage of the design parameter space. For training details and hyperparameter searches we refer the reader to Section G. For model scaling experiments, we scale the model from $\sim 9M$ parameters to $\sim 100M$ parameters. AB-UPT exhibits a steady trend of improvement for data scaling on validation, test, and OOD splits and achieves the lowest loss on all datasets compared to competitors. The trend for model scaling differs slightly. There is a noticeable improvement when scaling the model from $\sim 9M$ to $\sim 25M$ parameters. Beyond $25M$ parameters the test error flattens and further increases in depth yield marginal improvements. We interpret this as a data-bottlenecked regime in which the model's representational capacity already exceeds what the training distribution can support.

4.3.2 ViT

We present the results for our trained ViT model in Figure 6. The KPIs are comparable to the ones achieved by AB-UPT. Further, the error distribution shows higher error around shockwaves, expansion fans, reflected shocks, and in the exhaust nozzle. Especially at the exhaust nozzle the error pattern is different to the one obtained by AB-UPT. Specifically, the error pattern appears to be an artifact of patching. A potential remedy would be reducing the patch size or adding a smoothing operation (e.g., by convolution) in the decoder. We experimented with the latter and found no significant performance gains. Furthermore, decreasing the patch size leads to longer token sequences which results in substantial additional cost when performing self-attention as sequence length scales quadratically with patch size. The performance metrics in Table 4 clearly show that ViT is more efficient in terms of training and inference compared to AB-UPT while also reducing memory footprint and GFLOPs.

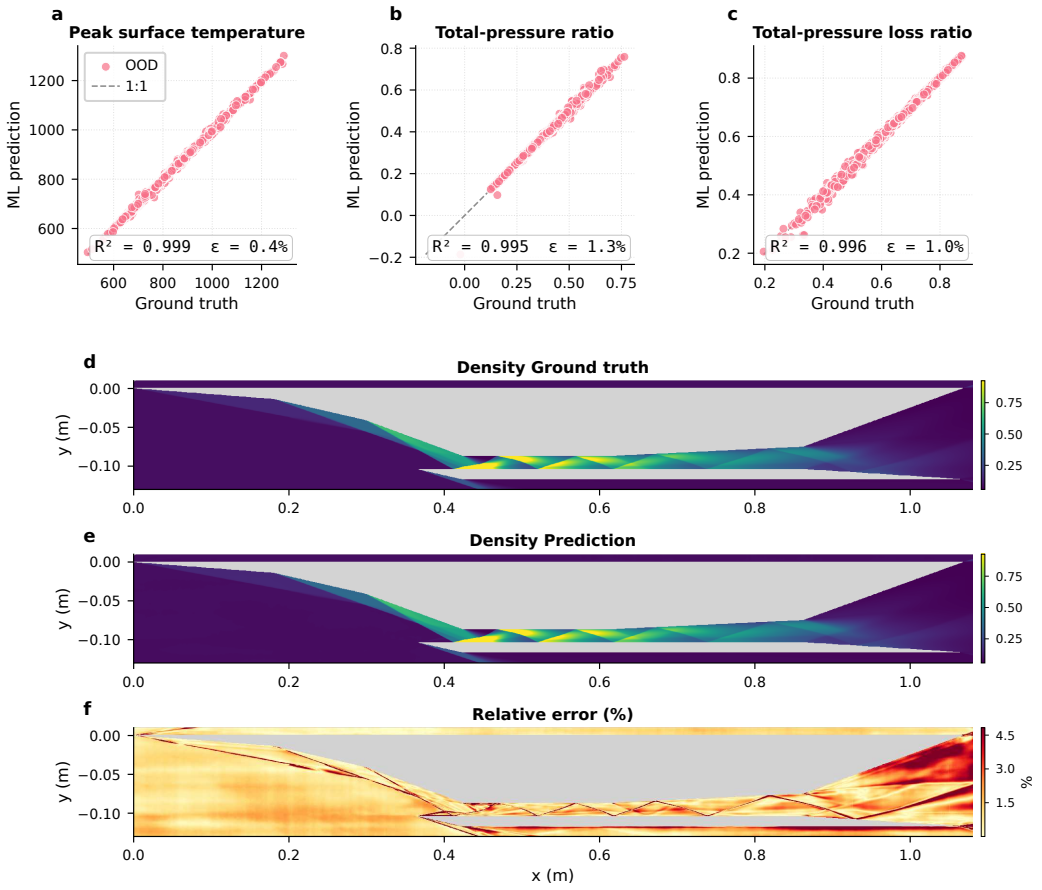


Figure 4: **Prediction error of AB-UPT on hypersonic scramjet.** Parity plots of predicted vs ground-truth on the OOD test data for **a**, peak surface temperature, **b**, total pressure ratio, and **c**, total pressure loss. We also report the ground-truth **d** and predicted **e** density field along with the relative error in percentage points **f** for a random case in the OOD test data.

The data-scaling results in Figure 5 uncover an interesting finding. At data-scarce regimes (50 samples) ViT attains lower average L_2 error across all fields compared to AB-UPT, however in larger data regimes AB-UPT recovers and outperforms ViT. In the data-scarce regime, ViT's regular-grid structure provides a strong spatial inductive bias that reduces the number of learnable degrees of freedom. AB-UPT must simultaneously infer both the spatial structure and the physical field mapping, a task for which 50 training samples provide weak information. In terms of model scale we observe similar behavior as for AB-UPT, again with a consistent offset in error.

4.3.3 Flow Matching with ViT backbone

We present the results for flow matching for an ensemble of 10 members in Figure 7. While we observe similar correlation coefficients for the different KPIs, the field error is significantly elevated for flow matching. The predictive uncertainty clearly highlights that the highest uncertainty is located around shockwaves. Compared to ViT predictions patching artifacts are visually less prevalent as we report the posterior mean over 10 ensemble members, effectively acting as a smoothing operator. Considering performance metrics in Table 4, flow matching generally is efficient during training, but requires more memory and inference time due to ensembling compared to ViT.

The data-scale results in Figure 5 show that flow matching generally yields worse performance than AB-UPT or ViT. However, we make several interesting observations. First, the average gap between in-distribution and OOD evaluation splits is smaller for flow matching ($9.63\% \pm 3.69\%$) than for AB-UPT ($24.23\% \pm 5.72\%$) or ViT ($22.10\% \pm 7.11\%$). Second, elevated test errors indicate that

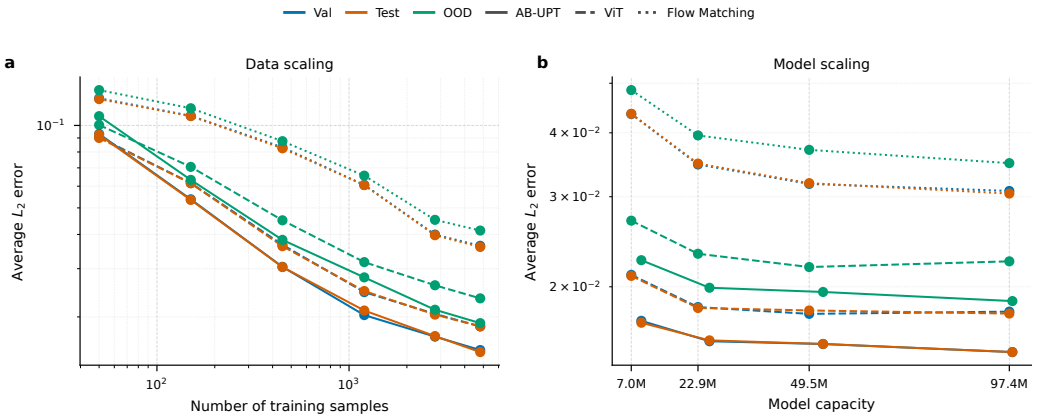


Figure 5: **Scaling laws for the different modeling paradigms.** Data **(a)** and model scaling laws **(b)** for AB-UPT, ViT, and Flow Matching on validation, test, and OOD extrapolation sets.

Table 4: **Performance metrics for the different physics emulators.** We report number of parameters, latency, FLOPs, and peak memory consumption over 10 inference passes. AB-UPT processes the entire point cloud grid as queries and flow matching ensembles over 10 samples.

Model	Params (M)	Train (ms)	Inference (ms)	FLOPs (G)	Peak mem (MB)
AB-UPT	25.67	29.64 ± 0.13	2089.87 ± 22.95	506834.2	13739
ViT	22.92	10.06 ± 0.02	10.07 ± 0.05	757.9	282
Flow Matching	23.34	8.15 ± 0.02	237.54 ± 6.08	7602.1	375

there appears to be a fundamental trade-off between accuracy and robustness for flow matching. This might be traced back to the deterministic nature of the problem setup. As flow matching is a probabilistic method it adds uncertainty estimation and additional robustness, at the cost of accuracy. We observe similar model scaling behavior as for ViT, albeit with a consistent offset in error.

Uncertainty Quantification In practice it is vital to obtain estimates on the predictive uncertainty of the emulator to inform decision making. Regions that exhibit high predictive uncertainty indicate that model predictions may not be trusted. We analyze the predictive uncertainty obtained by the different neural PEs. In particular, we visualize the correlation between the per-sample RMSE for cases in the OOD evaluation set and check for correlation with the per-sample mean of the ensemble variance. To obtain an ensemble for the deterministic ViT and AB-UPT models, we train three model instances with different random seeds. For flow matching we construct ensemble members of 10 initial noise samples and average them over three seeds. We report our findings on the predicted total pressure field for flow matching (Figure 8a), ViT (Figure 8b), and AB-UPT (Figure 8c). Flow matching achieves the highest Pearson correlation (0.80) and coefficient of determination ($R^2 = 0.64$). This pattern is consistent across other predicted fields such as enthalpy, kinetic energy, Mach number, and temperature as we show in Figure 13.

4.4 Physics-aware Model Refinement

After pre-training, we perform physics-aware refinement aimed at improving physical consistency as well as predictive capabilities of the PE. Each refinement run starts from a pre-trained checkpoint and minimizes a weighted sum of up to three terms,

$$\mathcal{L} = \mathcal{L}_{\text{data}} + w_{\text{div}}\,\mathcal{L}_{\text{div}} + \lambda\,\mathcal{L}_{\text{PDE}}, \tag{5}$$

namely a supervised data-reconstruction loss $\mathcal{L}_{\text{data}}$, a divergence term from the base model $\mathcal{L}_{\text{div}}$, and a physics loss $\mathcal{L}_{\text{PDE}}$. $\mathcal{L}_{\text{div}}$ and $\mathcal{L}_{\text{PDE}}$ are target-free, i.e., they require only the mesh and input parameters, with no ground-truth field data. The physics loss exploits the differentiability of the JAX-Fluids solver, evaluating the point-wise residual of the discretized equations via the solver's residual operator (see Appendix F.4). Since neither target-free term consumes labeled data, refinement

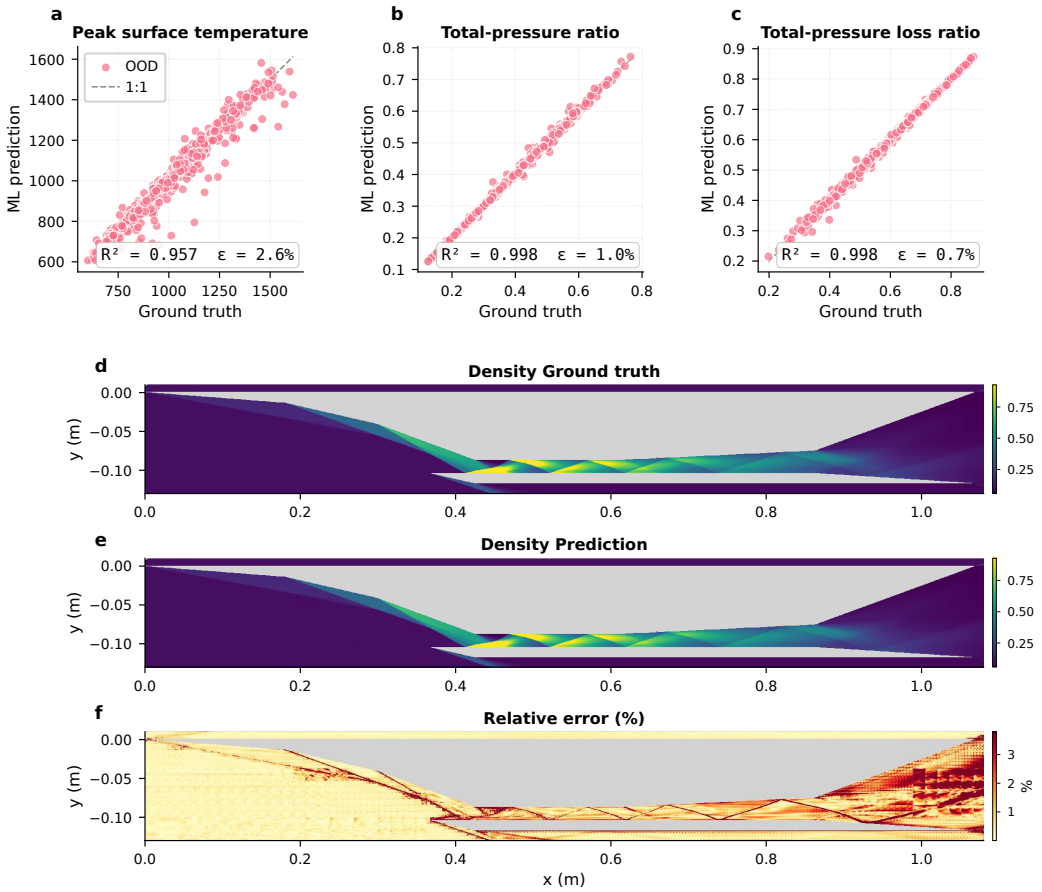


Figure 6: **Prediction error of ViT on hypersonic scramjet.** Parity plots of predicted vs ground-truth on the OOD test data for **a**, peak surface temperature, **b**, total pressure ratio, and **c**, total pressure loss. We also report the ground-truth **d** and predicted **e** density field along with the relative error in percentage points **f** for a random case in the OOD test data.

requires no additional simulations beyond those already used for pre-training. We provide further implementation and training details in Section F.4.

We present results for our proposed model refinement strategy showing that (i) model refinement improves performance after pre-training, (ii) the addition of the physics loss improves generalization, and (iii) refinement can be entirely target-free on simulation meshes without the need for solved flow-field data. For this line of experiments we take the ViT model from Section 4.3.2 as a base model and conduct different fine-tuning experiments. The residual operator acts naturally on the ViT's regular grid, whereas recovering a full grid from AB-UPT is expensive (see Table 4). Furthermore, ViT is the cheapest method in terms of FLOPs and inference speed compared to AB-UPT and flow matching,

To demonstrate the effectiveness of fine-tuning based on differentiating through the residual operator of JAX-Fluids, we perform the following experiments. First, as a baseline we simply fine-tune the Base model for the same amount of steps as we use for other fine-tuning strategies (Base + Data). Second, we add a loss term on the residual operator (Base + Data + Residual). Finally, to show that no ground-truth data are required for the fine-tuning procedure, we remove the data loss term and train only on the residual loss, plus the divergence loss for regularization. We include $\mathcal{L}_{\text{div}}$ as only training on $\mathcal{L}_{\text{PDE}}$ collapses to degenerate fields with near-zero residuals; the divergence term anchors the prediction to the pre-trained prior, supplying the missing constraint to select physically meaningful solutions.

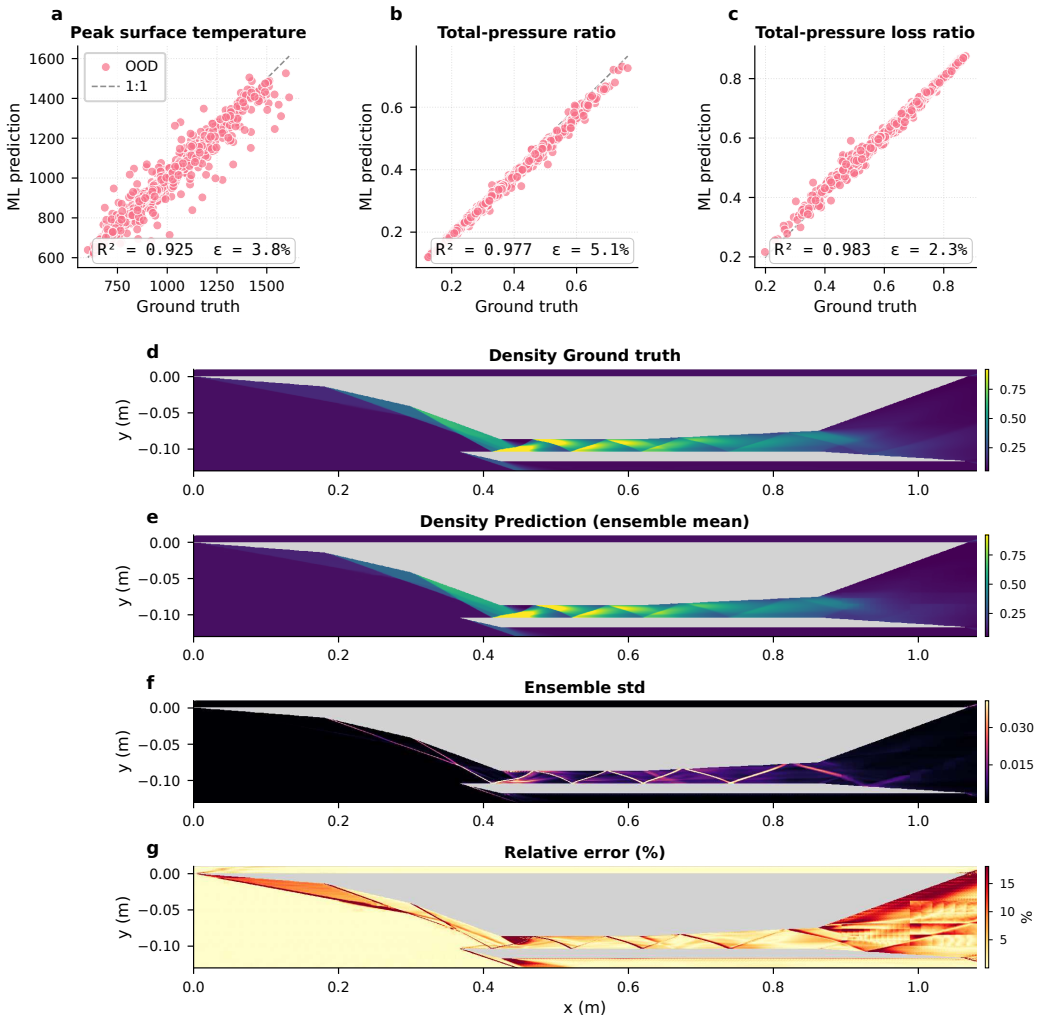
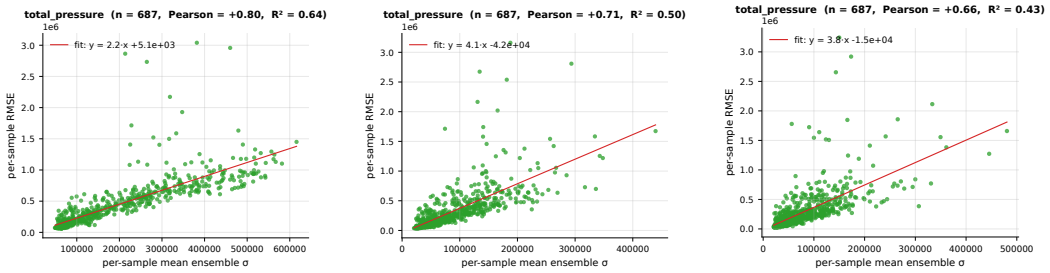


Figure 7: **Prediction error of Flow Matching on hypersonic scramjet.** Parity plots of predicted vs ground-truth on the OOD test data for **a**, peak surface temperature, **b**, total pressure ratio, and **c**, total pressure loss. We also report the ground-truth **d** and predicted **e** density field along with the ensemble standard deviation **f** and relative error in percentage points **g** for a random case in the OOD test data.

We report the results on the different conservation terms in $\mathcal{L}_{\text{PDE}}$ in Table 5. Our main findings are three fold, namely (i) the addition of $\mathcal{L}_{\text{PDE}}$ significantly reduces residuals and in turn improves physical consistency, (ii) target-free refinement consistently leads to lowest residuals, and (iii) additional gains can be obtained by extending the coverage of the parameter space by augmenting **D1** with **D2**. In Figure 10 we show the distribution of residuals for the different conservation terms to highlight the significant improvement for target-free refinement. In addition we show in Table 9 in Section I that the model after refinement is more thermodynamically consistent in the sense that derived quantities align better with the ground-truth steady state. Moreover, in Figure 9 we visualize the improvement on the different residual terms compared to the base model which highlights that our physics-aware model refinement significantly reduces residuals especially in regions around the shockwaves, indicated by the highlighted blue region.

Finally, we investigate the effect of refinement on the predicted flowfields. Table 6 reports the relative improvement in field-level L2 error for the different refinement variants. Generally we observe minor fluctuations ($< 1\%$) on the derived quantities (h, p_t, k, T, Ma). The difference is more pronounced for the primitive fields p, ρ, and $\mathbf{u}$, where we observe a slight degradation of field errors up to $\sim 6\%$ for target-free refinement. However, this degradation is less pronounced on the OOD set



a. Flow matching **b.** ViT **c.** AB-UPT

Figure 8: **Correlation between predictive uncertainty and error of the total-pressure field across the three emulators.** Each panel compares the predicted distribution of P_t against the JAX-Fluids reference on the out-of-distribution test split for the flow-matching ViT (**a**), the deterministic ViT (**b**) and the AB-UPT model (**c**). We show p_t as it is the direct integrand of the total-pressure ratio π_d used as performance KPI. Correlation of p_t propagates one-to-one into π_d at every probe location. The flow-matching surrogate is the most strongly calibrated on p_t.

Table 5: Mean per-sample RMS of the conservation-law residuals for mass r_ρ, x-momentum $r_{\rho u}$, y-momentum $r_{\rho v}$, and energy r_E, each normalized by its characteristic flux scale, for different models fine-tuned on different loss terms. Lower is more physically consistent. Averaged over three random seeds. $*$ denotes extension of training data with additional target-free cases while matching the number of update steps.

Split	Model	R_ρ	$R_{\rho u}$	$R_{\rho v}$	R_E
Val	Base	$0.2753_{\pm 0.0002}$	$0.2590_{\pm 0.0001}$	$0.7055_{\pm 0.0023}$	$0.3097_{\pm 0.0003}$
	+ Data	$0.2721_{\pm 0.0001}$	$0.2557_{\pm 0.0001}$	$0.7001_{\pm 0.0000}$	$0.3060_{\pm 0.0001}$
	+ Data + Residual	$0.1271_{\pm 0.0004}$	$0.1131_{\pm 0.0005}$	$0.5154_{\pm 0.0004}$	$0.1363_{\pm 0.0006}$
	+ Divergence + Residual	$0.0825_{\pm 0.0000}$	$0.0734_{\pm 0.0001}$	$\mathbf{0.3714}_{\pm 0.0000}$	$0.0856_{\pm 0.0000}$
	+ Divergence + Residual*	$\mathbf{0.0806}_{\pm 0.0001}$	$\mathbf{0.0715}_{\pm 0.0000}$	$0.3732_{\pm 0.0000}$	$\mathbf{0.0839}_{\pm 0.0000}$
Test	Base	$0.2729_{\pm 0.0001}$	$0.2573_{\pm 0.0003}$	$0.6961_{\pm 0.0024}$	$0.3070_{\pm 0.0010}$
	+ Data	$0.2693_{\pm 0.0001}$	$0.2537_{\pm 0.0001}$	$0.6910_{\pm 0.0000}$	$0.3027_{\pm 0.0001}$
	+ Data + Residual	$0.1251_{\pm 0.0004}$	$0.1116_{\pm 0.0005}$	$0.5093_{\pm 0.0004}$	$0.1338_{\pm 0.0006}$
	+ Divergence + Residual	$0.0810_{\pm 0.0000}$	$0.0723_{\pm 0.0000}$	$\mathbf{0.3667}_{\pm 0.0000}$	$0.0840_{\pm 0.0000}$
	+ Divergence + Residual*	$\mathbf{0.0792}_{\pm 0.0001}$	$\mathbf{0.0705}_{\pm 0.0000}$	$0.3689_{\pm 0.0000}$	$\mathbf{0.0824}_{\pm 0.0000}$
OOD	Base	$0.3126_{\pm 0.0033}$	$0.2886_{\pm 0.0026}$	$0.8219_{\pm 0.0068}$	$0.3517_{\pm 0.0026}$
	+ Data	$0.3120_{\pm 0.0000}$	$0.2873_{\pm 0.0000}$	$0.8244_{\pm 0.0005}$	$0.3506_{\pm 0.0000}$
	+ Data + Residual	$0.1601_{\pm 0.0005}$	$0.1404_{\pm 0.0005}$	$0.5940_{\pm 0.0005}$	$0.1716_{\pm 0.0006}$
	+ Divergence + Residual	$0.1061_{\pm 0.0000}$	$0.0932_{\pm 0.0001}$	$\mathbf{0.4234}_{\pm 0.0001}$	$0.1105_{\pm 0.0000}$
	+ Divergence + Residual*	$\mathbf{0.1042}_{\pm 0.0001}$	$\mathbf{0.0915}_{\pm 0.0000}$	$0.4249_{\pm 0.0000}$	$\mathbf{0.1090}_{\pm 0.0000}$

($\sim 1\%$) and comparable to refinement using $\mathcal{L}_{\text{data}}$. Our interpretation is that residual-based refinement substantially improves local conservation properties but yields only minor changes in aggregate field accuracy, suggesting that the quality of the pre-trained prior remains the dominant factor in prediction accuracy of the field values.

5 Discussion

No absolute hierarchy in model choice. We observe different trade-offs between the architectures and training paradigms we investigated. In the data-scarce regime the ViT's regular-grid inductive bias makes it more sample-efficient than AB-UPT's point-cloud representation. Flow matching, while trailing the deterministic models on accuracy, exhibits a substantially smaller in-distribution-to-OOD performance gap and the strongest correlation between predictive uncertainty and per-sample error on derived quantities. Rather than seeking a single best architecture, practitioners should select the model

Table 6: Relative change (%) over the Base model in the relative L2 error of static pressure p, density ρ, velocity $\boldsymbol{u}$, enthalpy h, total pressure p_t, kinetic energy k, temperature T, and Mach number Ma, for the fine-tuned variants (positive = lower error than Base). Computed from the mean errors over three random seeds. $*$ denotes extension of training data with additional target-free cases while matching the number of update steps.

Split	Model	p	ρ	$\boldsymbol{u}$	h	p_t	k	T	Ma
Val	+ Data	+0.75	+0.52	+0.49	−0.49	−0.94	+0.20	−0.49	−0.83
	+ Data + Residual	+1.95	+0.96	−0.07	−0.17	−1.04	+0.65	−0.17	−0.59
	+ Divergence + Residual	−3.68	−5.78	−5.95	−0.59	−1.89	−0.53	−0.59	−0.89
	+ Divergence + Residual*	−3.54	−5.77	−5.98	−0.58	−2.13	−0.45	−0.58	−0.85
Test	+ Data	+0.92	+0.85	+0.93	−0.02	+0.66	+0.82	−0.02	+0.27
	+ Data + Residual	+2.13	+1.29	+0.35	+0.30	+0.53	+1.26	+0.30	+0.53
	+ Divergence + Residual	−3.90	−5.55	−5.61	−0.47	−0.75	−0.15	−0.47	+0.03
	+ Divergence + Residual*	−3.87	−5.60	−5.64	−0.28	−0.74	−0.03	−0.28	+0.24
OOD	+ Data	+0.29	−0.01	+0.41	−0.75	−0.44	+0.02	−0.75	−0.60
	+ Data + Residual	+2.27	+1.57	+1.31	−0.14	−0.19	+0.77	−0.14	−0.00
	+ Divergence + Residual	+0.20	−1.57	−1.53	−0.33	−0.73	+0.10	−0.33	+0.15
	+ Divergence + Residual*	+0.36	−1.42	−1.49	−0.20	−1.23	+0.27	−0.20	+0.19

class that matches the data budget, the downstream decision (point estimate versus uncertainty-aware), and the inference-time computational constraint.

Physics-aware refinement improves physical consistency. Model refinement based on the physics loss consistently improves residuals, while field errors only exhibit slight variations. Since residuals are computed via derivatives and non-linear transformations of the state variables, even small modifications to the predicted fields can produce disproportionately larger changes in the residuals. Most importantly, target-free refinement, only requires meshes and design parameters and never observes a ground-truth field. This recovers the physical consistency benefits of PINN-style residual minimization [KKL+21] while avoiding their well-known optimization difficulties [WYP22], as the physics-aware refinement starts from a trained prior.

Implications for design optimization. Because the PE is fully differentiable and the geometric and inflow conditioning is explicit, the trained emulator can be inserted directly into gradient-based design loops as in [BBSA25, PCD+25], returning sensitivities of integrated KPIs with respect to the 15 geometric parameters and the freestream Mach number in milliseconds. The error numbers on the OOD set are comparable to typical engineering tolerances in early-stage design, suggesting that the PE could already be deployed for design-space exploration with full-solver verification reserved for selected candidates. However, we note that we have not explored our PE in such a setting yet.

Practical recommendations. Based on our findings, we propose a practical guide for developing physics emulators in hypersonic regimes. (1) **Establish model requirements.** Determine whether the downstream task demands point estimates or uncertainty quantification, what inference latency is acceptable, and how much training data are available. In data-scarce settings, architectures with strong inductive biases such as the regular-grid ViT are preferable. When data are abundant and accuracy is paramount, more expressive pointwise architectures such as AB-UPT should be favored. If predictive uncertainty is needed, probabilistic formulations such as flow matching offer built-in distributional estimates. (2) **Establish scaling behavior.** Train at progressively larger dataset sizes and monitor the validation error. As long as the scaling curve has not saturated and the compute budget permits, generating additional data is the most impactful investment, as global prediction quality is determined by the pre-training stage. (3) **Apply target-free refinement.** Apply residual-based refinement to improve local physical consistency, ideally on top of a well-converged base model. This step requires only meshes and design parameters, no reference flow fields, making it applicable to new regions of the design space at minimal cost. In our experiments, refinement primarily improves local conservation residuals with limited effect on aggregate field accuracy, suggesting it complements rather than replaces pre-training data.

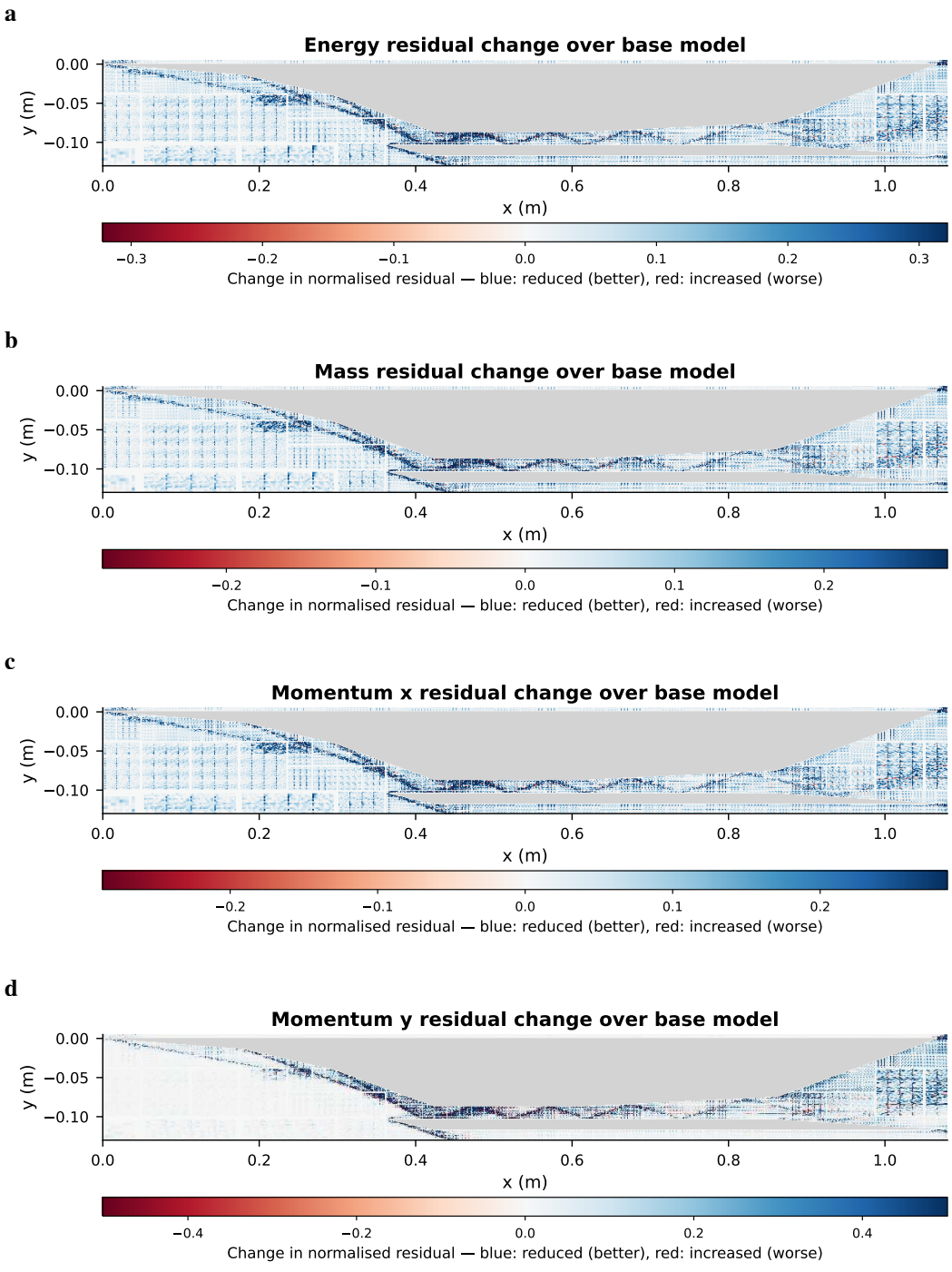


Figure 9: **Improvement of residuals after target-free refinement.** We report the change in normalized residuals where blue indicates improvement over the base model and red indicates worse residuals for **a**, energy conservation, **b**, mass conservation, **c**, x-component of momentum conservation, and **d**, y-component of momentum conservation.

6 Limitations

Scope of the present study and outlook. All scramjet configurations in this work are two-dimensional. While two-dimensional configurations omit certain physical effects, they already

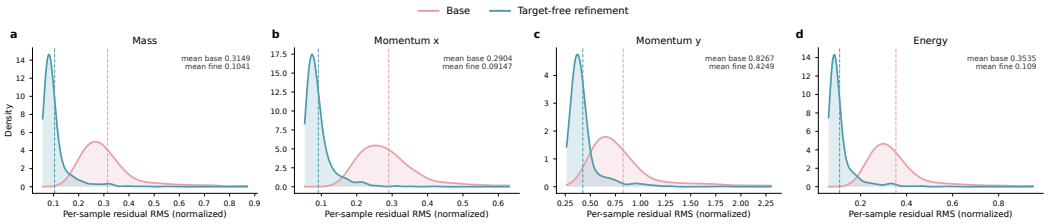


Figure 10: **Distribution of residuals on the OOD test set.** We report residuals for the conservation terms mass **(a)**, momentum in x and y direction **(b–c)**, and energy **(d)**. Base refers to the base model prior to refinement. Target-free refinement significantly reduces residuals and hence improves physical consistency.

contain sharp discontinuities and multi-scale flow structures that remain highly challenging for neural emulators due to spectral bias [RBA+19, XZL+19]. Extension to 3D, including turbulence, shock-boundary-layer interaction, and finite-rate chemistry, is a natural next step which we have not explored here. The pre-training dataset comprises roughly $7{,}000$ simulations, which our model-scaling analysis reveals to be the binding constraint beyond $\sim$25M parameters. A promising direction to overcome this bottleneck is pre-training on cheaper, intentionally under-converged data and recovering solution quality through the target-free refinement stage, further reducing the cost of data generation.

The differentiability requirement can be a barrier. Our workflow requires the solver to be differentiable and the residual to be evaluated with the same discretization used to produce the training data. JAX-Fluids meets both requirements, but the wider CFD ecosystem remains dominated by CPU-based, non-differentiable codes. The gap is closing through native GPU rewrites in JAX and PyTorch, and through emerging tooling that lowers the porting cost and we expect target-free refinement to become applicable across an increasing fraction of the engineering simulation stack in the future. In the interim, compatible differentiable approximations of legacy solvers may offer a useful bridge.

7 Conclusions

We introduce a fully GPU-based workflow for neural physics emulators specifically designed for the stringent requirements of hypersonic flow simulation. Our workflow ranges from data generation via JAX-Fluids to pre-training and physics-aware refinement leveraging the differentiable code.

To make regular-grid architectures applicable to the adaptive block-structured octree meshes produced by modern GPU-native solvers, we combine absolute and relative positional encodings in physical space, rendering the model agnostic to block count and ordering. This enables a direct comparison of regular-grid and pointwise paradigms on the same data, where we uncover various trade-offs. In addition to spatial representation, we compare deterministic and probabilistic training paradigms and conduct scaling studies with respect to model and data size. Probabilistic modeling trades uncertainty estimates for accuracy and generally exhibits a narrower gap between in-distribution and OOD performance. Furthermore, regular-grid methods outperform pointwise methods in data-scarce regimes. While pointwise methods yield highest accuracy in data-abundant regimes, they are slowest during inference followed by probabilistic and regular-grid methods. Therefore, the appropriate choice depends on the data budget, downstream requirements, and inference-time constraints.

Our physics-aware refinement depends only on the mesh and design parameters, hence can be carried out in a fully target-free manner without reference flowfields. This is significant for two reasons. First, new regions of the design space can be reached through meshing alone, at a small fraction of the cost compared to data generation. Second, the physics loss encourages the surrogate to produce solutions that better satisfy the discretized conservation laws than the base model, which is particularly valuable in extrapolatory regimes where additional training data are unavailable.

Our results demonstrate that a differentiable solver can serve as a refinement engine for neural PE. As differentiable GPU-native solvers mature and pre-training datasets grow, this paradigm could extend the practical reach of neural PE to increasingly complex flow regimes. Our workflow is a step toward

a broader role for solvers in CFD: not only as primary simulation tools, but as differentiable engines for training and refining physics emulators.

Acknowledgement

NAA acknowledges funding through ERC Advanced Grant Project No. 101094463. DAB, ABB, and NAA gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer JUWELS [Jue21] at Jülich Supercomputing Centre (JSC).

References

[AA25] R. Abaidi and N.A. Adams. Exploring denoising diffusion models for compressible fluid field prediction. *Computers & Fluids*, 298:106665, 2025.

[AAA+23] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[ABK+25] Benedikt Alkin, Maurits Bleeker, Richard Kurle, Tobias Kronlachner, Reinhard Sonnleitner, Matthias Dorfer, and Johannes Brandstetter. Ab-upt: Scaling neural cfd surrogates for high-fidelity automotive aerodynamics simulations via anchored-branched universal physics transformers. *arXiv preprint arXiv:2502.09692*, 2025.

[AFS+24] Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. *Advances in Neural Information Processing Systems*, 37:25152–25194, 2024.

[AKL+24] Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320 – 328, 2024.

[BBA23] Deniz A. Bezgin, Aaron B. Buhendwa, and Nikolaus A. Adams. Jax-fluids: A fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows. *Computer Physics Communications*, 282:108527, 2023.

[BBA25] Deniz A. Bezgin, Aaron B. Buhendwa, and Nikolaus A. Adams. Jax-fluids 2.0: Towards hpc for differentiable cfd of compressible two-phase flows. *Computer Physics Communications*, 308:109433, 2025.

[BBK+25] Aaron B. Buhendwa, Deniz A. Bezgin, Petr Karnakov, Nikolaus A. Adams, and Petros Koumoutsakos. Data-driven shape inference in three-dimensional steady-state supersonic flows: Optimizing a discrete loss with jax-fluids. *Phys. Rev. Fluids*, 10:084902, Aug 2025.

[BBSA25] Deniz A. Bezgin, Aaron B. Buhendwa, Steffen J. Schmidt, and Nikolaus A. Adams. Ml-iles: End-to-end optimization of data-driven high-order godunov-type finite-volume schemes for compressible homogeneous isotropic turbulence. *Journal of Computational Physics*, 522:113560, 2025.

[BCCD08] Rafael Borges, Monique Carmona, Bruno Costa, and Wai Sun Don. An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. *Journal of Computational Physics*, 227(6):3191–3211, 2008.

[BEF19] Michael P Brenner, Jeff D Eldredge, and Jonathan B Freund. Perspective on machine learning for advancing fluid mechanics. *Physical Review Fluids*, 4(10):100501, 2019.

[BNK20] Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52(Volume 52, 2020):477–508, 2020.

[CLH+23] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.

[CSS+25] Anthony Carreon, Jagmohan Singh, Shivank Sharma, Shuzhi Zhang, and Venkat Raman. A gpu-based compressible combustion solver for applications exhibiting disparate space and time scales, 2025.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[EUD18] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

[FKL+25] Paul Fischer, Sebastian Kaltenbach, Sergey Litvinov, Sauro Succi, and Petros Koumoutsakos. Optimal lattice boltzmann closures through multi-agent reinforcement learning. *arXiv preprint arXiv:2504.14422*, 2025.

[GD26] Niloy Gupta and Karthik Duraisamy. Computational and physical considerations for the development of machine learning augmented turbulence models. *International Journal of Heat and Fluid Flow*, 117:110089, 2026.

[GKK24] Han Gao, Sebastian Kaltenbach, and Petros Koumoutsakos. Generative learning for forecasting the dynamics of high-dimensional complex systems. *Nature Communications*, 15(1):8904, 2024.

[HG16] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[HKAH06] X Y Hu, B C Khoo, N A Adams, and F L Huang. A conservative interface method for compressible flows. *Journal of Computational Physics*, 219:553–578, 2006.

[HP94] William H Heiser and David T Pratt. *Hypersonic airbreathing propulsion*. Aiaa, 1994.

[JES26] Khodr Jaber, Ebenezer Essel, and Pierre Sullivan. Gpu-native embedding of complex geometries in adaptive octree grids applied to the lattice boltzmann method. *Computer Physics Communications*, 324:110155, 04 2026.

[JMAK22] Ameya D. Jagtap, Zhiping Mao, Nikolaus Adams, and George Em Karniadakis. Physics-informed neural networks for inverse problems in supersonic flows. *Journal of Computational Physics*, 466:111402, 2022.

[Jue21] Juelich Supercomputing Centre. JUWELS Cluster and Booster: Exascale Pathfinder with Modular Supercomputing Architecture at Juelich Supercomputing Centre. *Journal of large-scale research facilities*, 7(A183), 2021.

[KKL+21] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[KO01] Marc C. Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

[KSA+21] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[LCBH+22] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[LeV92] R.J. LeVeque. *Numerical Methods for Conservation Laws*. Lectures in Mathematics ETH Zürich, Department of Mathematics Research Institute of Mathematics. Springer Basel AG, 1992.

[LH25] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

[LJP+21] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[LKA+20] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[LLHPG24] Marten Lienen, David Lüdke, Jan Hansen-Palmus, and Stephan Günnemann. From zero to turbulence: Generative modeling for 3d flow simulation. In *International Conference on Learning Representations*, volume 2024, pages 5203–5220, 2024.

[LT24] Qiang Liu and Nils Thuerey. Uncertainty-aware surrogate models for airfoil flow simulations with denoising diffusion probabilistic models. *AIAA Journal*, 62(8):2912–2933, 2024.

[LTZ08] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[MH80] Christopher P. McKee and David J. Hollenbach. Interstellar shock waves. *Annual Review of Astronomy and Astrophysics*, 18(Volume 18, 1980):219–262, 1980.

[MJK20] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

[MLR+24] Roberto Molinaro, Samuel Lanthaler, Bogdan Raonić, Tobias Rohner, Victor Armegioiu, Stephan Simonis, Dana Grund, Yannick Ramic, Zhong Yi Wan, Fei Sha, et al. Generative ai for fast and accurate statistical computation of fluids. *arXiv preprint arXiv:2409.18359*, 2024.

[NDLK21] Guido Novati, Hugues Lascombes De Laroussilhe, and Petros Koumoutsakos. Automating turbulence modelling by multi-agent reinforcement learning. *Nature Machine Intelligence*, 3(1):87–96, 2021.

[PCD+25] Fabian Paischer, Leo Cotteleer, Yann Dreze, Richard Kurle, Dylan Rubini, Maurits Bleeker, Tobias Kronlachner, and Johannes Brandstetter. Going with the speed of sound: Pushing neural surrogates into highly-turbulent transonic regimes. *arXiv preprint arXiv:2511.21474*, 2025.

[PFSGB20] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

[PX23] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[RBA+19] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron C. Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, pages 5301–5310. PMLR, 2019.

[RH18] Deep Ray and Jan S Hesthaven. An artificial neural network as a troubled-cell indicator. *Journal of computational physics*, 367:166–191, 2018.

[RHH+13] Diego Rossinelli, Babak Hejazialhosseini, Panagiotis Hadjidoukas, Costas Bekas, Alessandro Curioni, Adam Bertsch, Scott Futral, Steffen J. Schmidt, Nikolaus A. Adams, and Petros Koumoutsakos. 11 pflop/s simulations of cloud cavitation collapse. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, New York, NY, USA, 2013. Association for Computing Machinery.

[SAL+24] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[Sha20] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[SKA+14] Jeffrey P Slotnick, Abdollah Khodadoust, Juan Alonso, David Darmofal, William Gropp, Elizabeth Lurie, and Dimitri J Mavriplis. Cfd vision 2030 study: a path to revolutionary computational aerosciences. Technical report, 2014.

[SWMW89] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

[TAB+23] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[Tor09] Eleuterio F. Toro. Riemann solvers and numerical methods for fluid dynamics [texte imprimé] : a practical introduction / eleuterio f. toro, 2009.

[TWPH20] Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA journal*, 58(1):25–36, 2020.

[Urz18] Javier Urzay. Supersonic combustion in air-breathing propulsion systems for hypersonic flight. *Annual Review of Fluid Mechanics*, 50(Volume 50, 2018):593–627, 2018.

[VB22] Ricardo Vinuesa and Steven L. Brunton. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358 – 366, 2022.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[WRLB+25] Benjamin Wilfong, Anand Radhakrishnan, Henry Le Berre, Daniel Vickers, Tanush Prathi, Nikolaos Tselepidis, Benedikt Dorschner, Reuben Budiardja, Brian Cornille, Stephen Abbott, Florian Schäfer, and Spencer Bryngelson. Simulating many-engine spacecraft: Exceeding 1 quadrillion degrees of freedom via information geometric regularization. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '25, page 14–24, New York, NY, USA, 2025. Association for Computing Machinery.

[WYP22] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

[XZL+19] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *CoRR*, abs/1901.06523, 2019.

[ZS19] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

A Author Contribution

DR implemented models, the training pipeline and preprocessing routines for the scramjet cases, with FS contributing to the model implementation and training pipeline. FP performed scaling law, uncertainty and self-consistency analyses, generated figures and contributed to the design of the experiments. DAB and ABB generated the simulation data with JAX-Fluids and created data-specific figures. DH ran experiments and created visualizations on residual-based fine-tuning. JB, SK, and NAA supervised the research and contributed to the conceptualization and design of the experiments. All authors reviewed and discussed the results and contributed to writing and revising the manuscript.

B Data Availability

The data used in this work is available at https://huggingface.co/datasets/paischer101/hypersonic-scramjets.

C Competing Interest

The authors declare no competing interests.

D Physical and Numerical Model

The fluid state at position $\mathbf{x} = [x, y]^T$ and time t is described either by the vector of primitive variables $\mathbf{W} = [\rho, u, v, p]^T$ or by the vector of conservative variables $\mathbf{U} = [\rho, \rho u, \rho v, E]^T$. Here, ρ denotes the density, u and v are the velocity components in the x- and y-directions, respectively, and p is the pressure. The velocity vector is denoted by $\mathbf{u} = [u, v]^T$. The total energy per unit volume is given by

$$E = \rho e + \frac{1}{2}\rho \mathbf{u} \cdot \mathbf{u}, \tag{6}$$

where e is the specific internal energy.

The present study considers inviscid compressible flows governed by the two-dimensional Euler equations,

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{U})}{\partial y} = 0, \tag{7}$$

where $\mathbf{F}$ and $\mathbf{G}$ are the convective flux vectors in x- and y-directions, respectively,

$$\mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{bmatrix}, \qquad \mathbf{G}(\mathbf{U}) = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}. \tag{8}$$

The system is closed using the ideal-gas equation of state,

$$p = (\gamma - 1)\rho e, \qquad c = \sqrt{\gamma \frac{p}{\rho}}, \tag{9}$$

where γ is the ratio of specific heats and c is the speed of sound. In this work, we use $\gamma = 1.4$. Equivalently, the pressure can be recovered from the conservative variables as

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho(u^2 + v^2)\right). \tag{10}$$

The compressible Euler equations are discretized on a Cartesian grid using a high-order Godunov-type finite-volume formulation [Tor09]. The computational domain is divided into rectangular control volumes with uniform cell sizes Δx and Δy. Cell centers are indexed by (i, j), and $\mathbf{U}_{i,j}$ denotes the cell averaged conservative state. Numerical fluxes through the vertical and horizontal cell faces are denoted by $\mathbf{F}_{i\pm\frac{1}{2},j}$ and $\mathbf{G}_{i,j\pm\frac{1}{2}}$, respectively.

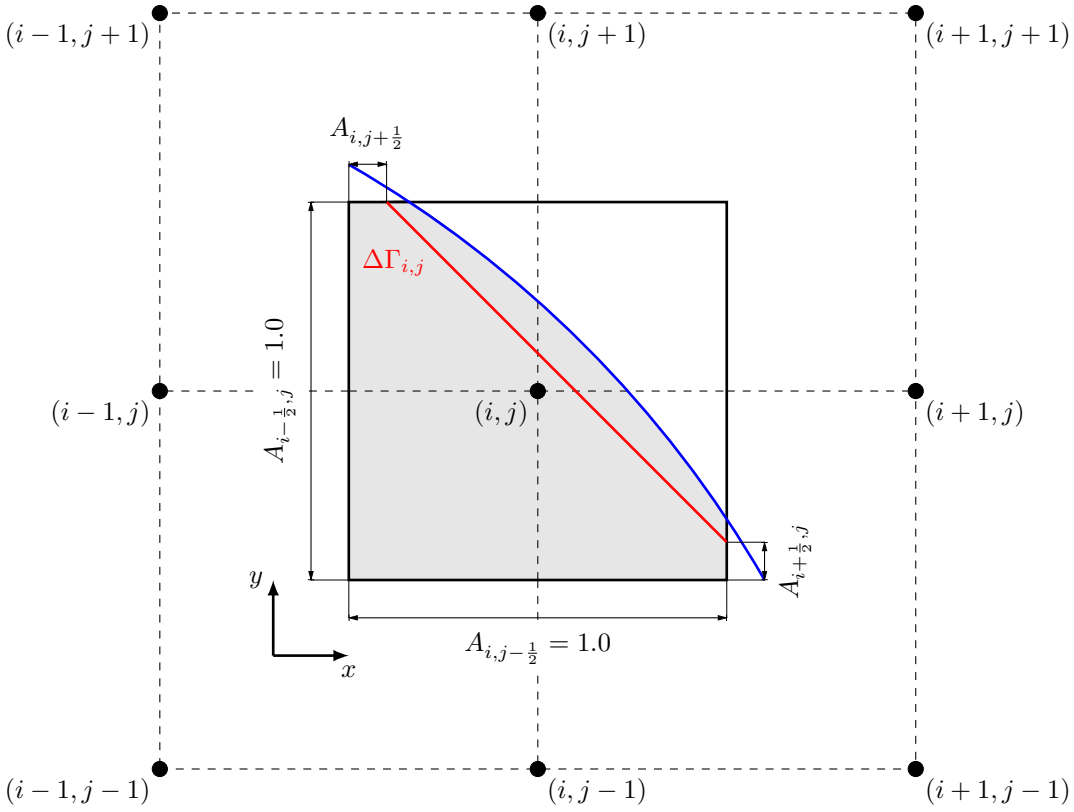


Figure 11: Schematic of a cut cell.

The numerical fluxes are computed using a two-step high-order Godunov procedure. First, left and right states are reconstructed at each cell face from the neighboring cell averages using a shock-capturing reconstruction scheme. In this work, we employ the fifth-order WENO-Z reconstruction [BCCD08]. Second, the reconstructed states are used as input to an approximate Riemann solver. Here, we use the HLLC Riemann solver [Tor09] to compute the numerical fluxes across the cell faces.

Immersed solid boundaries are represented using a conservative sharp-interface cut-cell method [HKAH06]. The solid-fluid interface is described implicitly by a level-set function $\phi(\mathbf{x})$, where ϕ satisfies the signed-distance property $\|\nabla\phi\| = 1$. The interface location is given by the zero level set

$$\Gamma = \{\mathbf{x} \mid \phi(\mathbf{x}) = 0\}. \tag{11}$$

Cells intersected by the interface are referred to as *cut cells*, while cells that are entirely filled with fluid are referred to as *full cells*. A schematic illustration of a cut cell, including the fluid volume fraction, face apertures, and interface segment, is shown in Figure 11.

For each cell, we define the fluid volume fraction $\alpha_{i,j} \in [0, 1]$, which denotes the fraction of the cell area occupied by fluid. We also define the face apertures $A_{i\pm\frac{1}{2},j}$ and $A_{i,j\pm\frac{1}{2}}$, which denote the fractions of the corresponding cell faces open to the fluid phase. For full fluid cells, all apertures and the volume fraction are equal to one. The standard finite-volume discretization is therefore recovered as a special case of the cut-cell formulation.

The semi-discrete conservative update for both full cells and cut cells is written as

$$\begin{aligned}\frac{d}{dt}\left(\alpha_{i,j}\mathbf{U}_{i,j}\right) =& \frac{1}{\Delta x}\left(A_{i-\frac{1}{2},j}\mathbf{F}_{i-\frac{1}{2},j} - A_{i+\frac{1}{2},j}\mathbf{F}_{i+\frac{1}{2},j}\right) \\ &+ \frac{1}{\Delta y}\left(A_{i,j-\frac{1}{2}}\mathbf{G}_{i,j-\frac{1}{2}} - A_{i,j+\frac{1}{2}}\mathbf{G}_{i,j+\frac{1}{2}}\right) + \frac{1}{\Delta x\Delta y}\mathbf{X}_{i,j}.\end{aligned} \tag{12}$$

Here, the cell face fluxes are weighted by the corresponding apertures, and $\mathbf{X}_{i,j}$ denotes the interface flux contribution inside cell (i, j). For full cells, no solid-fluid interface is present and therefore $\mathbf{X}_{i,j} = \mathbf{0}$. Together with $\alpha_{i,j} = 1$ and unit apertures, Eq. (12) reduces to the standard semi-discrete finite-volume update on a Cartesian grid.

The interface flux represents the force exerted by the immersed boundary on the fluid. In two dimensions, it is given by

$$\mathbf{X}_{i,j} = \begin{bmatrix} 0 \\ p_\Gamma \Delta\Gamma_x \\ p_\Gamma \Delta\Gamma_y \\ p_\Gamma \Delta\mathbf{\Gamma}_{i,j} \cdot \mathbf{v}_\Gamma \end{bmatrix}, \qquad \Delta\mathbf{\Gamma}_{i,j} = \begin{bmatrix} \Delta\Gamma_x \\ \Delta\Gamma_y \end{bmatrix} = \begin{bmatrix} \left(A_{i+\frac{1}{2},j} - A_{i-\frac{1}{2},j}\right) \Delta y \\ \left(A_{i,j+\frac{1}{2}} - A_{i,j-\frac{1}{2}}\right) \Delta x \end{bmatrix}. \tag{13}$$

Here, p_Γ is the pressure at the interface, $\mathbf{v}_\Gamma$ is the interface velocity, and $\Delta\mathbf{\Gamma}_{i,j}$ is the oriented interface length vector associated with the cut cell. The components of this vector correspond to the projections of the interface segment onto the coordinate directions and are obtained from the differences of opposite face apertures. The interface pressure is approximated by the cell-center pressure of the corresponding cut cell. Since only static solid bodies are considered in this work, the interface velocity is zero, $\mathbf{v}_\Gamma = \mathbf{0}$, and the energy contribution of the interface flux vanishes. The interface flux therefore contributes only to the momentum equations through the pressure force exerted by the solid boundary.

The volume fraction $\alpha_{i,j}$, the face apertures $A_{i\pm\frac{1}{2},j}$ and $A_{i,j\pm\frac{1}{2}}$, and the oriented interface length vector $\Delta\mathbf{\Gamma}_{i,j}$ are computed from the level-set representation of the geometry using a marching-squares reconstruction. This yields a conservative cut-cell discretization that preserves the structured Cartesian layout while allowing immersed solid boundaries to be represented sharply.

E Data Generation

To split the dataset into in-distribution and out-of-distribution subsets without imposing an arbitrary cutoff on any single design variable, we use an unsupervised Isolation Forest [LTZ08] over the d-dimensional vector of varying scramjet design parameters. The forest comprises 100 isolation trees fit with a fixed random seed. The Isolation Forest builds 100 random binary trees that recursively split the design space, and assigns each sample a score s_i equal to the average number of splits needed to isolate it. Samples in sparse regions are isolated in few splits and therefore receive low scores, which we use to flag out-of-distribution candidates. We designated as out-of-distribution the k samples with the lowest scores, where k is fixed by the requested OOD fraction. Figure 12 shows the distribution of the anomaly scores over all cases and the different datasets ($D1$, $D2$) to illustrate that the selected OOD split generally corresponds to the tail of the distribution, so edge cases of the generated data. While there is minimal overlap in the anomaly scores between $D2$ and the selected OOD samples we stress that these do not correspond to exact same parameter settings. In fact, there is no exact match between any of the cases in the overlapping anomaly scores in terms of geometry and inflow parameters so the parameter space they cover differ.

F Implementation Details

F.1 AB-UPT

We instantiate AB-UPT as a volume-only configuration, which discards the surface branch and operates solely on a 3D volumetric anchor set, since the scramjet task requires field predictions inside the flow domain rather than on a surface. From every simulation we draw $N_{\text{vol}} = 16{,}384$ volume anchor points uniformly at random from the mesh. Volumetric coordinates are linearly rescaled into $[0, 1000]^2$ before being fed to the network, and we predict all eight target fields, namely static pressure, density, two-component velocity, specific enthalpy, total pressure, kinetic energy, temperature and Mach number. Each field is standardized per-channel using statistics computed once over the training split. Each anchor position $\mathbf{x}_i \in \mathbb{R}^2$ is first lifted to a token of width $d = 384$ via a continuous sin–cos Fourier embedding followed by a two-layer GELU MLP [HG16]. We additionally apply rotary positional encodings [SAL+24, RoPE] inside attention. The trunk is a stack of ten self-attention blocks followed by four additional volume-decoder blocks. Therefore, during training anchors are used as both queries and keys/values. We select $h = 3$ heads, an MLP expansion factor

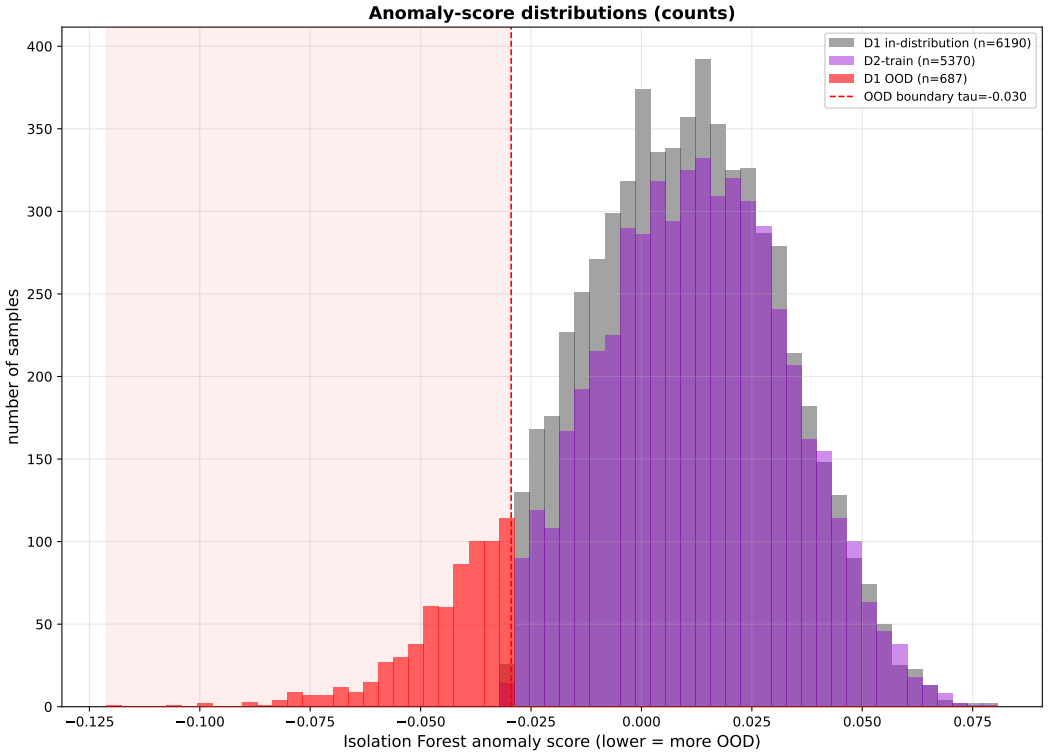


Figure 12: **Isolation-Forest diagnostic for the out-of-distribution split.** We report the distribution of anomaly scores obtained via Isolation Forest fitted on the geometry and inflow parameters. Samples at the lower end of the distribution are selected to be OOD. Importantly there is no overlap in the parameter vectors between $D2$ and the OOD test set, so no information is leaked to the OOD set. The remaining sets are randomly selected from the remaining samples.

of 4, and truncated-normal initialization ($\sigma = 0.02$). A final linear head decodes each anchor token to the eight-channel field prediction. The model is conditioned on a 16-dimensional design vector (15 geometry parameters and the inflow Mach number) that is injected into every transformer block via the AdaLN-style scale/shift pathway [PX23] with a per-condition dimension of 16. Training minimizes an unweighted sum of channel-wise MSE losses on the standardized targets, optimized with Lion [CLH+23] at a peak learning rate of 1×10^{-5}, weight decay 0.05, gradient-norm clipping at 1.0, and a linear warm-up over 5% of training followed by a cosine decay to 1×10^{-6}. We train for 250 epochs in float16 mixed precision with an effective batch size of one simulation per step, where each step performs attention over $16{,}384$ anchor tokens. We select the final checkpoint by the best average relative L_2 loss on the validation set.

For scaling experiments we vary the depth of the model ranging from $\approx 10M$ parameters to $\approx 100M$ parameters. Specifically, we vary the depth in terms of transformer blocks in $\{1, 10, 25, 50\}$ as it allows higher learning rates for larger models compared to scaling in width. For each of these variants we keep the same model hidden dimension as $d = 384$.

F.2 ViT

We compare against a grid-native Vision Transformer [DBK+21, ViT] tailored to the block-structured, obtained by adaptively-refined meshes produced by the JAX-Fluids solver. Each simulation is exposed to the network as a stack of N_B Cartesian blocks of shape $H \times W$ (in our case $H = W = 64$), with per-block isotropic spatial scale and a Boolean fluid mask flagging cells outside the wetted domain. Coordinates are min–max rescaled to $[0, 1000]^2$ and per-block scales are min–max rescaled to $[0.2, 1000]$ before being passed to the network. All eight target fields are channel-wise z-score normalized using the same precomputed statistics as for the AB-UPT model. Patchification is performed with a non-overlapping patch size of $p = 16$ via 2D average pooling on the coordinate

grid. Blocks with zero fluid fraction are pruned and the remaining blocks are right-padded across the batch so that token sequences are concatenated over all blocks of a sample. Each patch centroid is then encoded with a continuous sin–cos Fourier embedding followed by a SiLU MLP [EUD18] into a token of width $d = 384$, and RoPE is applied inside every attention head, derived from the same average-pooled centers.

The backbone is a stack of $L = 8$ pre-norm transformer blocks with $h = 3$ heads, RMSNorm ($\varepsilon = 10^{-6}$) [ZS19], a SwiGLU FFN [Sha20] with expansion factor 4, and AdaLN-Zero conditioning on the 16-dimensional design vector. The design vector is first embedded by a shared SiLU MLP and then drives a per-block linear head that produces six modulation parameters (shift, scale, gate for MSA and MLP) per channel. Weights are initialized with Xavier-uniform on all linear layers and the AdaLN modulation projection and the final linear layer are zero-initialized, while the MSA/MLP gate biases are reset to 1.0 so blocks are fully active at step zero and the network does not collapse to the identity (a failure mode we observed under the standard AdaLN-Zero initialization in this regression setting). The output head projects each patch token to $p^2 \cdot C_{\text{out}}$ channels which are unpatchified back to the original cell grid and sliced into the eight field-specific predictions.

The training objective is identical to AB-UPT, an unweighted sum of eight per-field MSE losses on the standardized targets and optimized with Lion at peak learning rate 1×10^{-4}, weight decay 0.05, gradient-norm clipping at 1.0, and a linear warm-up over 5% of training followed by cosine decay to 1×10^{-6}. We train for 250 epochs in float16 mixed precision with an effective batch size of one simulation per step and select the final checkpoint on the validation loss.

F.3 Flow Matching

The third method replaces the deterministic regression objective with a flow-matching generative model on the same multi-block grid representation. We adopt the linear stochastic-interpolant formulation [LCBH+22, LH25]. For a clean sample $x_0 \in \mathbb{R}^{B \times N_B \times H \times W \times C}$ we collect all eight target fields (velocity, density, pressure, enthalpy, total pressure, kinetic energy, temperature, Mach number) and Gaussian noise $\varepsilon \sim \mathcal{N}(0, I)$. Then we sample a timestep $t \sim \mathcal{U}[0, 1]$ uniformly per sample and form the straight-line interpolant $z_t = t, x_0 + (1 - t)\varepsilon$. Given a conditioning vector c, the network is parameterized to predict the clean state $x_0^{\text{pred}} = f_\theta(z_t, t, c)$ rather than the velocity or the noise, and we train under the analytic velocity-matching loss $\mathcal{L} = \frac{1}{|\Omega|} \sum_\Omega \left|(x_0 - z_t)/(1 - t) - (x_0^{\text{pred}} - z_t)/(1 - t)\right|^2$. Training is done only on fluid cells Ω defined by the AMR fluid mask and with $(1 - t)$ clamped from below by $t_\varepsilon = 5 \times 10^{-2}$ for numerical safety near $t = 1$. The corruption is i.i.d. standard Gaussian across cells, blocks and channels.

The denoiser shares the same backbone as the deterministic ViT baseline ($d = 384$, 8 transformer blocks, 3 heads, MLP ratio 4, RMSNorm + SwiGLU + AdaLN-Zero, 2D RoPE on average-pooled patch centres) and differs only in the input and conditioning pipelines. The noisy field z_t is patchified by a two-stage bottleneck patch embedding with stride $p = 16$ projecting the eight target channels to a PCA-like bottleneck of width 64, followed by a 1×1 convolution back to width $d = 384$. The scalar timestep is mapped through a sinusoidal embedding of width 256 and a SiLU MLP into a vector of width d. The 16-dimensional design vector (15 geometry parameters concatenated with the inflow Mach number) is independently embedded by a SiLU MLP of width d, and the two are summed to form the AdaLN-Zero modulation token fed into every transformer block. The final layer predicts $p^2 \cdot C_{\text{out}}$ patch channels which are unpatchified back to the original cell grid.

We train for 250 epochs in bfloat16 mixed precision with an effective batch size of one simulation per step, using Lion at peak learning rate 3×10^{-6}, weight decay 0.05, gradient-norm clipping at 1.0, a 5% linear warm-up and a cosine decay to 1×10^{-6}. At inference we draw $x_T \sim \mathcal{N}(0, I)$ and integrate the learned probability flow ODE $\mathrm{d}x/\mathrm{d}t = (x_0^{\text{pred}}(x_t, t) - x_t)/(1 - t)$ from $t = 0$ to $t = 1$ with explicit Euler for $N - 1$ steps and a closed-form last step $x \leftarrow \alpha x + (1 - \alpha)x_0^{\text{pred}}$ with $\alpha = (1 - t_{\text{next}})/(1 - t_{\text{cur}})$ that lands the trajectory exactly on the model's terminal x_0 estimate. In practice we always sweep over a variety of integration steps and always report the one resulting in the best average relative L_2 error across all field predictions on the validation set.

F.4 Physics-aware Refinement

This appendix details how the differentiable JAX-Fluids solver [BBA23, BBA25] is used for physics-aware refinement of neural emulators. We note that the solver, its discretization, the multi-block mesh, and the level-set-based immersed boundary method are described in Section 3.1.

Residual loss. A steady-state solution of the compressible Euler equations, denoted by $\mathbf{U}^*$, satisfies

$$\mathbf{R}\left(\mathbf{U}^*\right) = \frac{\partial \mathbf{F}(\mathbf{U}^*)}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{U}^*)}{\partial y} = 0$$

where the convective flux vectors $\mathbf{F}$ and $\mathbf{G}$ are defined in Section D (see Equation (8)). Thus, at steady state, the divergence of the convective fluxes vanishes. For a flow field $\mathbf{U}$ that does not satisfy the steady-state Euler equations exactly,

$$\mathbf{R}\left(\mathbf{U}\right) \neq 0.$$

Point-wise residual calculation. In general, predictions of the physics emulator do not exactly satisfy the governing equations. We exploit this property to define a physics-based loss for fine-tuning. Given a surrogate prediction, we evaluate the discrete PDE residual using JAX-Fluids and minimize it by backpropagating through both JAX-Fluids and the physics emulator.

The discrete residual vector in cell (i, j) is given by

$$\mathbf{R}_{i,j} \;=\; \frac{\mathbf{F}_{i+\frac{1}{2},j} - \mathbf{F}_{i-\frac{1}{2},j}}{\Delta x} \;+\; \frac{\mathbf{G}_{i,j+\frac{1}{2}} - \mathbf{G}_{i,j-\frac{1}{2}}}{\Delta y}, \tag{14}$$

where the cell-face fluxes $\mathbf{F}_{i\pm\frac{1}{2},j}$ and $\mathbf{G}_{i,j\pm\frac{1}{2}}$ are computed with the same fifth-order WENO-Z reconstruction and HLLC Riemann solver used during data generation, ensuring consistency of the residual operator with the discretization used for generating training data. We note that cut cells contribute through their volume fraction and face apertures as outlined in Section D.

The point-wise residuals are aggregated into a scalar physics loss $\mathcal{L}_{\mathrm{PDE}}$ by squaring each residual component, applying a per-equation weight w_k, and summing over all fluid cells,

$$\mathcal{L}_{\mathrm{PDE}} = \sum_{(i,j)\in\Omega} \sum_{k=1}^{4} w_k \left(R_{i,j}^k\right)^2 \Delta x \Delta y, \tag{15}$$

where $R_{i,j}^k$ denotes the k-th component of $\mathbf{R}_{i,j}$, corresponding to mass, x-momentum, y-momentum, and total energy. We found that the y-momentum had higher normalized residuals than the other conserved quantities and use $w_k = (1,\ 1,\ 0.1,\ 1)$, which down-weights its balance to ensure an equal representation of all components.

Differentiable coupling. Since the reference data were generated with JAX-Fluids, the refinement residual must be evaluated with the same solver: only then is the discrete residual consistent with the numerical discretization underlying the training data, so that minimizing it drives the prediction toward the solver solution rather than toward the fixed point of some other discretization. The surrogate, however, trains in PyTorch, so its predictions and the resulting gradients must be passed between PyTorch and the JAX-based solver. We bridge the two with zero-copy DLPack sharing wrapped in a custom autograd function, giving exact gradients consistent with the discretization. Floating-point inputs are upcast to double precision for the solver and the gradients are downcast back to the surrogate's working precision.

Refinement objective. Every refinement run starts from the same pre-trained ViT checkpoint and optimises a weighted sum of up to three terms,

$$\mathcal{L} = \mathcal{L}_{\mathrm{data}} + w_{\mathrm{div}}\, \mathcal{L}_{\mathrm{div}} + \lambda\, \mathcal{L}_{\mathrm{PDE}},$$

namely a supervised data-reconstruction loss $\mathcal{L}_{\mathrm{data}}$ against the reference fields, a *divergence* loss $\mathcal{L}_{\mathrm{div}}$ that penalises the mean-squared deviation of the prediction from a frozen copy of the pre-trained model, and the physics loss $\mathcal{L}_{\mathrm{PDE}}$. $\mathcal{L}_{\mathrm{PDE}}$ and $\mathcal{L}_{\mathrm{div}}$ are target-free and refinement can therefore be applied to new samples without running a full simulation for each, only the mesh is needed.

A key advantage of the physics and divergence terms is that neither requires ground-truth fields. Both are evaluated from the model's own prediction and the mesh, so we can refine the surrogate on additional samples for which a simulation was never run and only a mesh and its design parameters are available. For these samples the supervised data term cannot be formed and is dropped.

The physics loss alone is not sufficient and leads to degenerate solutions. The divergence term prevents this by penalising deviation from the frozen pre-trained model, acting as a target-free replacement for the reconstruction loss, while the physics loss drives the field toward conservation consistency. This extends the surrogate to new regions of the design space at the cost of meshing rather than a full steady-state simulation.

Fine-tuning uses Lion (learning rate 10^{-6}, weight decay 0.05, gradient-norm clipping 1.0) with a linear-warmup cosine-decay schedule in mixed (fp16) precision. The variants we evaluate, which differ only in the active loss terms and the training data, are summarised in Table 7.

Table 7: Refinement variants. All start from the same pre-trained ViT and share the optimiser and physics-loss settings above, differing only in the active loss terms and the training data (D_1 simulation data, D_2 simulation-free; see Table 2). A dash denotes an inactive term. $*$ denotes fine-tuning on cases without existing field data, hence no data loss can be computed.

Variant	Reconstruction	Divergence	Physics λ	Data	Epochs
Data only (baseline)	1	–	–	D_1	50
Data + physics	1	–	0.2	D_1	50
Divergence + physics	–	0.2	0.2	D_1	50
Divergence + physics*	–	0.2	0.2	D_1+D_2	25

G Scaling Experiments

Nested OOD splits with farthest-point sampling. For data-scaling studies we additionally require training subsets of sizes $n_1 < n_2 < \cdots < n_K$ that (i) share a single fixed (val, test, OOD) evaluation set and (ii) are strictly nested: $\mathcal{T}n_1 \subset \mathcal{T}n_2 \subset \cdots \subset \mathcal{T}n_K$. After fixing the evaluation splits as above, we apply greedy farthest-point sampling (FPS) on the training pool. Parameters are min–max normalized to $[0,1]^d$ so that all dimensions contribute equally to Euclidean distance. FPS is initialized at the point closest to the pool centroid (to avoid boundary bias) and at every step appends $\arg\max_i \min j \in \mathcal{S} \|\mathbf{p}_i - \mathbf{p}_j\|_2$ to the selected set $\mathcal{S}$. Because each iteration grows $\mathcal{S}$ by exactly one point, every prefix of length n_k is a valid k-center cover of the training pool, yielding the required nesting property. The training subset of size n_k used in our scaling experiments is the first n_k points of this FPS ordering. Validation, test, and OOD sets are identical across all n_k, isolating the effect of training-set size from variation in the evaluation distribution. Every split is verified to (i) contain no duplicate indices within any subset, (ii) have empty pairwise intersection across subsets, and (iii) cover the full set of discovered run indices.

Hyperparameter searches. The scaling curves we report compare best-tuned configurations at every coordinate of the (model size, dataset size) grid rather than evaluating a single fixed recipe, to avoid systematically favoring any one capacity. For both the deterministic AB-UPT and ViT surrogates we ran a dedicated learning-rate sweep at every (model size, dataset size) cell of the scaling grid, bracketing the value reported in the implementation-details section by one order of magnitude on either side, and selected the run minimizing the in-distribution validation loss for inclusion in the reported curves. For the flow-matching ViT we additionally ran a grid over integration steps $N_{\text{int}} \in \{1, 2, 3, 5, 7, 9, 10, 20, 50, 70\}$, and report the one that performs best on the validation set.

H Predictive Uncertainty

We provide additional results for the correlation between predictive uncertainty and error across the three different methods on the remaining field predictions. In Figure 13 we show correlation on the temperature, Mach number, kinetic energy, and enthalpy fields. Flow matching consistently attains the highest correlation and coefficient of determination compared to AB-UPT and ViT. On the

remaining density and velocity fields (see Figure 14), the picture slightly changes. Specifically, on those quantities AB-UPT exhibits the highest correlation between predictive error and uncertainty. We believe the reason for this is that density and velocity contain sharp localized structures such as shocks and wall-aligned viscuous gradients, whereas the remaining fields are nonlinear algebraic combinations of those primitives and therefore smoother. AB-UPT is less affected by these artifacts because it learns a position-wise mapping from coordinates to field values and can therefore disentangle predictions. ViT on the other hand uses patching with an initial average pooling layer such that the average coordinate per field is mapped to field values. This design choice is deliberate to save computational complexity as smaller patch sizes yield longer token sequences that amplify complexity of the quadratic self-attention. Therefore, we believe that these results are artifacts from design choices and do not reflect a direct disadvantage of regular-grid-based methods.

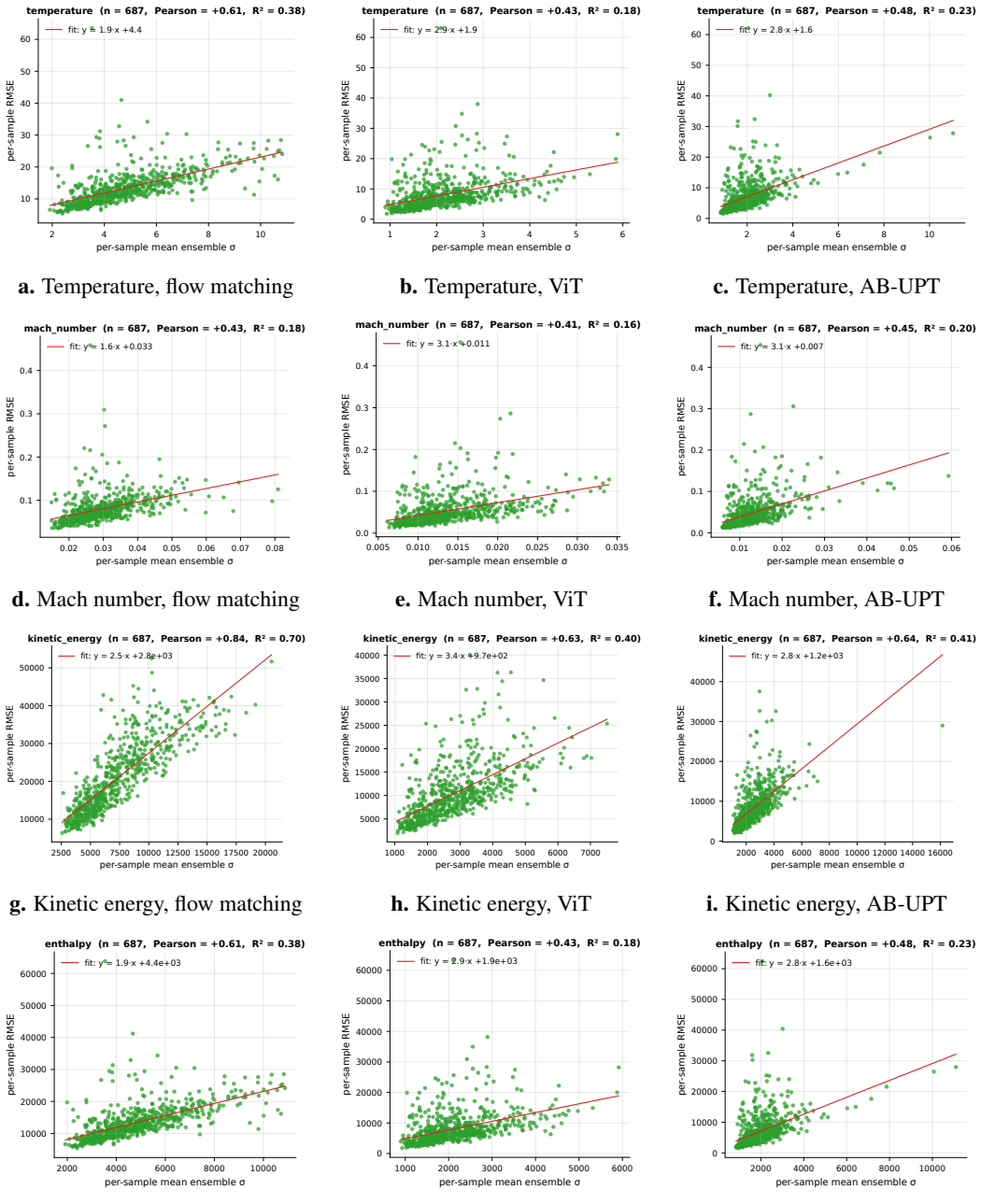
I Thermodynamic Self-consistency of Derived Quantities

We train the physics emulators to predict a redundant set of flow primitives aside from pressure p, density ρ, and velocity $\mathbf{u}$, which are thermodynamically independent. The remaining fields follow in closed form from these primitives through their respective relations. In particular, the static temperature $T = p/(\rho R)$, the specific static enthalpy $h = \frac{\gamma}{\gamma-1} p/\rho$, the Mach number $Ma = |\mathbf{v}|/\sqrt{\gamma p/\rho}$, the total pressure $p_t = p(1 + \frac{\gamma-1}{2} Ma^2)^{\gamma/(\gamma-1)}$ and the kinetic-energy density $\mathrm{KE} = \frac{1}{2}\rho|\mathbf{v}|^2$, with $\gamma = 1.4$ and $R = 287.05\ \mathrm{J\,kg^{-1}\,K^{-1}}$.

We evaluate all emulators on the different splits. For every case the model's full inference path yields the predicted primitives on all fluid cells, with the flow-matching estimate taken as the ensemble mean over 10 independent stochastic samples. For each derived quantity we report two errors against the ground truth, both as the relative L_2 norm taken over the fluid cells, namely the error of model's *predicted* field, and the one of the *derived* field, obtained by applying the relation above to the model's own predicted p, ρ, and $\mathbf{u}$. Each entry of Table 8 lists predicted, /,derived (%). The agreement between both measures the thermodynamic self-consistency of an emulators predictions, while their difference quantifies whether emitting a redundant channel directly is preferable to deriving it from primitives.

Three patterns emerge. First, the temperature and enthalpy columns are identical, because h and T differ only by the constant factor $c_p = \gamma R/(\gamma - 1)$. As relative L_2 error is scale invariant the errors for both are consequently equal. Second, direct prediction is consistently more accurate, oftentimes substantially, as it sidesteps error accumulation through a nonlinear transform. The gap is mild for temperature, enthalpy and Mach number and negligible for the kinetic energy, indicating that the emulators' base predictions are very nearly thermodynamically self-consistent. However, the gap is dramatic for the total pressure. Since P_t depends on Ma through the steep isentropic power law, at the supersonic Mach numbers small velocity and Mach errors are amplified super-exponentially, inflating the derived total-pressure error to $\sim 10^{4}\%$ for the ViT and $\sim 10^{10}\%$ for flow matching, while the corresponding direct heads remain well behaved at 2.5–3.9%. Finally, the ranking across architectures is preserved on every split and all three degrade only modestly on the OOD split without changing order. Together these results show that the different emulators are thermodynamically coherent and argue for directly predicting derived quantities whose reconstruction from the base primitives is nonlinear, most critically the total pressure, but also the Mach number.

Finally, we also report the difference between predicting and deriving the different quantities after physics-aware model refinement in Table 9. Here we compare ViT before and after physics-aware model refinement. Physics-aware refinement results in marginal improvements if the different field quantities are predicted directly. However there is a significant reduction in error after refinement if the different quantities are derived. This difference is particularly pronounced in derived fields that exhibit a nonlinear relation to primitive, like total pressure and Mach number. For the former we sometimes even observe an improvement of around an order of magnitude. This finding indicates that physics-aware model refinement facilitates thermodynamic consistency of the PE.



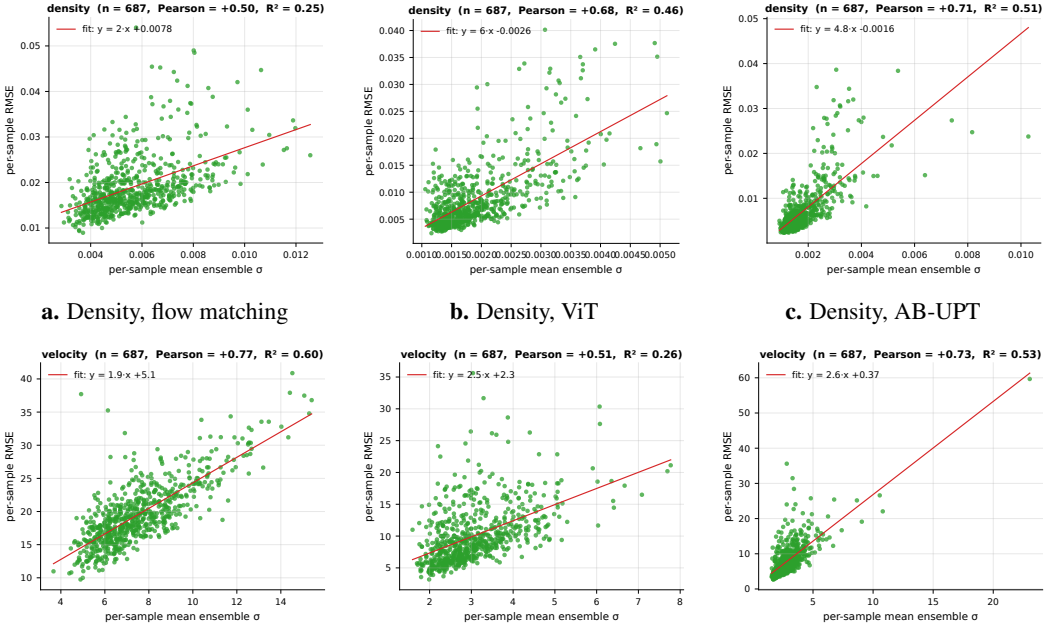
a. Temperature, flow matching **b.** Temperature, ViT **c.** Temperature, AB-UPT

d. Mach number, flow matching **e.** Mach number, ViT **f.** Mach number, AB-UPT

g. Kinetic energy, flow matching **h.** Kinetic energy, ViT **i.** Kinetic energy, AB-UPT

j. Enthalpy, flow matching **k.** Enthalpy, ViT **l.** Enthalpy, AB-UPT

Figure 13: **Correlation of predictive uncertainty and error of emulator predictions for derived fields.** Rows correspond to temperature (**a**–**c**), Mach number (**d**–**f**), kinetic energy (**g**–**i**) and enthalpy (**j**–**l**). Columns correspond, left to right, to flow matching, ViT and AB-UPT. All five fields are nonlinear algebraic combinations of the primitive variables and exhibit a correlation ranking that is consistent across the three surrogates with flow matching exhibiting best correlation between predictive error and uncertainty.



a. Density, flow matching

b. Density, ViT

c. Density, AB-UPT

d. Streamwise Velocity, flow matching

e. Streamwise Velocity, ViT

f. Streamwise Velocity, AB-UPT

Figure 14: **Correlation of predictive uncertainty and error for density and velocity predictions for the three emulators.** Rows correspond to density (**a**–**c**), and streamwise velocity (**d**–**f**), columns correspond, left to right, to flow-matching, ViT and AB-UPT. AB-UPT exhibits the best correlation and coefficient of determination between per-sample RMSE and per-sample average standard deviation as its inductive bias allows capturing position-wise shock and boundary layer artifacts.

Table 8: **Prediction vs deriving different physical quantities after pre-training.** Relative L2 errors (%) for derived quantities enthalpy h, total pressure p_t, kinetic energy k, temperature T, and Mach number Ma for different models for predicting / deriving the different quantities. Predicting derived quantities directly generally results in smaller error.

Split	**Model**	h	P_t	k	T	Ma
Val	AB-UPT	1.56/1.74	2.57/9.64	1.99/1.99	1.56/1.74	0.90/1.14
	ViT	1.67/6.01	$2.66/6.52 \times 10^4$	2.17/2.17	1.67/6.01	0.97/2.54
	Flow matching	2.87/4.08	$3.44/3.67 \times 10^{10}$	4.56/4.58	2.87/4.08	1.42/6.50
Test	AB-UPT	1.53/1.72	2.49/14.20	1.96/1.95	1.53/1.72	0.88/1.12
	ViT	1.69/3.61	$2.61/5.83 \times 10^4$	2.18/2.19	1.69/3.61	0.96/2.48
	Flow matching	2.85/4.09	$3.45/6.2 \times 10^{10}$	4.50/4.51	2.85/4.09	1.42/7.19
OOD	AB-UPT	1.90/2.20	2.93/15.49	2.48/2.48	1.90/2.20	1.06/1.44
	ViT	2.06/12.96	$2.98/7.88 \times 10^4$	2.73/2.73	2.06/12.96	1.14/2.86
	Flow matching	3.25/4.80	$3.86/9.01 \times 10^{10}$	5.08/5.08	3.25/4.80	1.60/9.38

Table 9: **Prediction vs deriving different physical quantities after physics-aware refinement.** Relative L2 errors (%) for derived quantities enthalpy h, total pressure p_t, kinetic energy k, temperature T, and Mach number Ma for ViT before and after model refinement for predicting / deriving the different quantities.

	Quantity	**Val**	**Test**	**OOD**
Pre-training	Enthalpy h	$1.67/6.01$	$1.69/3.61$	$2.06/12.96$
	Total Pressure p_t	$2.66/6.52 \times 10^4$	$2.61/5.83 \times 10^4$	$2.98/7.88 \times 10^4$
	Kinetic Energy k	$2.17/2.17$	$2.18/2.19$	$2.73/2.73$
	Temperature T	$1.67/6.01$	$1.69/3.61$	$2.06/12.96$
	Mach Number Ma	$0.97/2.54$	$0.96/2.48$	$1.14/2.86$
Refinement	Enthalpy h	$1.66/3.66$	$1.68/9.04$	$2.03/4.64$
	Total Pressure p_t	$2.68/1.1 \times 10^4$	$2.64/8.9 \times 10^3$	$2.99/1.2 \times 10^4$
	Kinetic Energy k	$2.16/2.36$	$2.16/2.36$	$2.69/2.82$
	Temperature T	$1.66/3.66$	$1.68/9.04$	$2.03/4.64$
	Mach Number Ma	$0.96/1.55$	$0.96/1.55$	$1.12/1.73$