Learning 3D Hypersonic Flow with Physics-Enhanced Neural Fields: A Case Study on the Orion Reentry Capsule

Anonymous authors
Paper under double-blind review

Abstract

We develop a 3D aerothermodynamic simulator for the Orion reentry capsule at hypersonic speeds, a timely case study given its role in upcoming lunar missions. The large computational meshes required for these scenarios make traditional computational fluid dynamics impractical for full-mission performance prediction and control. In this work, we propose physics-enhanced 3D neural fields for predicting steady hypersonic flow around aerodynamic bodies. The model maps spatial coordinates and angle of attack to pressure, temperature, and velocity components. We enhance the base model with Fourier positional feature mappings, which allow it to capture the sharp discontinuities typical of hypersonic flows, and further constrain the solution by imposing no-slip and isothermal wall conditions. We compare our proposed approach to other surrogate alternatives, such as graph neural networks, and demonstrate its superior performance in capturing the steep gradients ubiquitous in this regime. Our formulation yields a continuous and computationally efficient aerothermodynamic surrogate that supports rapid exploration of operating conditions based on angle of attack variation under realistic flight profiles. While we focus on Orion, the proposed framework provides a general methodology for data-driven simulation in 3D hypersonic aerothermodynamics.

1 Introduction

Reynolds-Averaged Navier-Stokes (RANS), Large Eddy Simulation (LES), and Direct Numerical Simulations (DNS) are all Computational Fluid Dynamics (CFD) approaches (Roache, 1972; Anderson, 1995; Tu et al., 2018) that approximate the Navier-Stokes equations (Navier, 1822; Stokes, 1845) governing fluid flows. Fidelity depends on purpose, with DNS for detailed research physics and RANS for most industrial engineering, since DNS becomes extremely expensive beyond simple geometries and engineers must trade accuracy for computational feasibility (Pope, 2000). Artificial neural networks offer fast simulation and rapid prototyping. Trained on CFD data, they can produce solutions orders of magnitude faster than traditional CFD. Because numerical simulations already have error, neural networks add another layer of uncertainty as approximations.

This trade-off can be acceptable for early exploration of geometry and flow design configurations in aircraft and spacecraft development, especially considering that our neural simulator can predict fields that take hundreds of hours to converge using traditional CFD (Section 3.5) in the order of seconds (Section 6.4). Once promising candidates are identified, higher-resolution CFD simulations can refine the results, followed by wind-tunnel testing. In this way, the design process could follow a natural progression: rapid neural exploration, targeted CFD simulation, and eventual experimental testing.

In this work, we develop a 3D aerothermodynamic simulator using the Orion reentry capsule as a test case. We focus on this configuration due to its high degree of symmetry, which makes it well suited for an initial study, as well as its relevance to upcoming lunar missions and real-world applications. Our contributions are as follows:

- We generate an ML-ready 3D CFD dataset for the Orion reentry capsule, sweeping the angle of attack (AoA) and utilizing advanced mesh refinement techniques to accurately model the hypersonic bow shock.
- We propose the use of physics-enhanced 3D neural fields to predict steady hypersonic flow around the geometry.
- We perform a thorough empirical study comparing neural fields and graph neural networks (GNNs) in the context of hypersonic flow modeling. This helps shed light on strengths and weaknesses of different learning methods.
- We conduct an in-depth aerothermodynamic analysis after training our models, evaluating performance not only using machine learning metrics but also from an aerodynamicist's perspective.

2 Related work

In aerodynamics, reduced-order models (ROMs) are simplified mathematical representations designed to capture the essential behavior of complex aerodynamic systems while significantly reducing computational cost (Ripepi et al., 2018). Within the literature, both unsteady laminar and turbulent flow regimes have been studied using methods that extract coherent structures and flow patterns to inform dynamical models with predictive and control capabilities (Durmaz et al., 2013; Stabile et al., 2017; Guzman-Inigo et al., 2019; Nidhan et al., 2020; Schmidt et al., 2018; Fukami et al., 2021; Giannopoulos & Aider, 2020). Linear models have shown good performance even in nonlinear and highly nonlinear, large-scale systems; however, they typically require real-time, partial measurements of the full system state to account for nonlinear effects (Guzman-Inigo et al., 2019; Mikhaylov et al., 2021; Loiseau et al., 2018; Savarino & Papadakis, 2022). In contrast, nonlinear models rely on more sophisticated architectures but often suffer from tuning difficulties and reduced interpretability (Nair & Goza, 2020; Kim et al., 2021; Rozov & Breitsamter, 2021).

In parallel, deep learning approaches have been increasingly applied to a wide range of aerospace engineering problems, including turbulence modeling, aerodynamic shape optimization, wall-flux-based wall models, and rocket liquid engine design to name a few (Ling et al., 2016; Borde et al., 2021a;b; Sayyari et al., 2022; Fang et al., 2019; Kaandorp, 2018; Li et al., 2022; Xu et al., 2021; Li & Zhang, 2021; Liu et al., 2023; Lozano-Duran & Bae, 2020; Waxenegger-Wilfing et al., 2021). Deep learning methods are capable of approximating complex input-output relationships and capturing highly nonlinear dynamics directly from data. Previous aerothermodynamic simulators include those developed by Borde et al. (2023) for 2D flows around rocket geometries.

3 CFD data generation

In this section we describe the CFD simulation data generation process, which is later used to train our artificial neural network models.

3.1 The Orion reentry capsule

The Orion reentry capsule was originally conceived by Lockheed Martin in the early 2000s for NASA's Constellation program, an attempt to return to the Moon before the year 2020, which was subsequently canceled. Orion was later redesigned for several other programs and is now expected to be used in NASA's Artemis program. The first unit, named CM-002, was launched on November 16, 2022, on Artemis I, an uncrewed Moon-orbiting mission (see Figure 1). This project marked NASA's return to

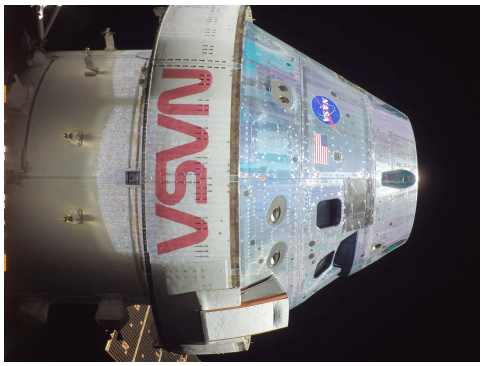


Figure 1: Orion, Artemis I, on flight day 13 (Nov. 28, 2022). Credit: NASA.

lunar exploration since the Apollo program five decades prior. In fact, in Greek mythology, Artemis is Apollo's twin sister, and NASA chose the name to explicitly echo the Apollo program while signaling a new era of lunar exploration. In particular, Orion provides protection from solar radiation, as well as advanced and reliable technologies for communication and life support, and, importantly for this work, enables high-speed entry into Earth's atmosphere (NASA, 2023), which we simulate.

3.2 Geometry

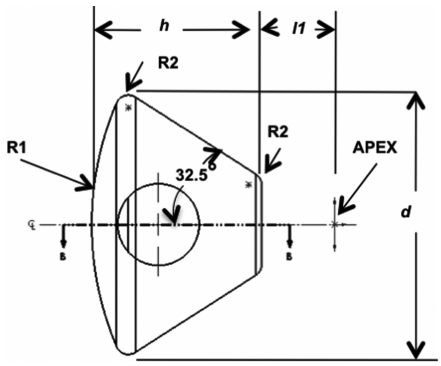


Figure 2: Orion CM Outer Mold Line. Figure from Brown et al. (2010).

While the geometry shown in Figure 1 incorporates numerous surface features such as windows, access hatches, and thruster ports, these details are neglected in the present work. The adopted geometry consists of a smooth, continuous outer mold line (OML) without local surface irregularities or protuberances. This idealized representation is appropriate for preliminary CFD simulations, where the primary objective is to capture the dominant aerodynamic behavior of the vehicle rather than the secondary effects associated with small-scale geometric features. More concretely, we utilize the OML geometry from the ballistic range tests reported in Brown et al. (2010): see Figure 2. The geometry is uniformly scaled to full scale using a maximum diameter of $d = 5\,\mathrm{m}$. The remaining geometric parameters are obtained by applying the same scaling to the reference dimensions.

3.3 Flow solver and models

The CFD solver STAR-CCM+ is used to perform steady 3D laminar simulations of the geometry. A third-order MUSCL central differencing (MUSCL-CD) scheme is employed for the spatial discretisation of convective fluxes (Van Leer, 1979), and the inviscid fluxes are computed with the AUSM+ flux-vector splitting scheme (Liou, 1996). The simulations are performed at a Mach number $M = 5$ and an altitude of 50 km. The static pressure and temperature are obtained from the International Standard Atmosphere (ISA) model, while the dynamic viscosity is computed using Sutherland's law. The freestream conditions are $T_0 = 1.30 \times 10^3\,\mathrm{K}$, $p_0 = 6.14 \times 10^6\,\mathrm{Pa}$, $M = 5.00$, and $Re_d = 9.68 \times 10^7$ (based on $d = 5\,\mathrm{m}$). Data is obtained for the geometry at these conditions and several AoAs in the range $0° \leq \alpha \leq 45°$. Simulations were not extended beyond $\alpha = 45°$, as the capsule is not expected to encounter such high AoAs during reentry under the flow conditions considered. Computations at higher angles were therefore deemed unnecessary.

Two main modeling simplifications are adopted to reduce the number of variables, as the primary objective of this work is to assess the feasibility of the neural field framework in the context of 3D hypersonics rather than to achieve high-fidelity flow predictions: (1) air is modeled as a calorically perfect, ideal gas, and (2) the flow is assumed to be fully laminar. The first simplification breaks down at the high temperatures generated behind the bow shock in this speed regime, where real gas effects and temperature-dependent thermodynamic properties become important. The second simplification is generally invalid due to the high Reynolds number Re_d of these simulations, for which fully turbulent flow conditions would be expected. Both real gas and turbulence models can be incorporated in future work to increase the fidelity of the simulations. Despite this being a simplification, it must be highlighted that these are reasonable assumptions in academic research, and do not have a direct impact on the downstream methodology discussed in this paper in terms of deep learning.

3.4 Simulation setup

The computational domain is a cylinder of diameter $10d$ and length $25d$. Freestream boundary conditions are applied to each boundary of the domain and the wall is defined as no-slip isothermal ($T_w = 300\,\mathrm{K}$). The mesh is generated within STAR-CCM+ and consists of prism layer cells near viscous boundaries surrounded by polyhedral mesh elsewhere. A mesh convergence study with respect to aerodynamic coefficients of lift, drag,

pitch moment as well as with respect to the wall heat flux was first performed, leading to a final converged mesh of 22.8 million cells (130.7 million nodes) for the case at $\alpha = 0°$. Cases at high AoAs generally result in a lower number of cells due to the more slender bow-shock and smaller wake volume.

The final mesh is shown in Figure 3 for the case at $\alpha = 0°$: a centerplane cross-section of the mesh, superimposed on the computed Mach number field. The wake flow captured in the snapshot continues to oscillate despite residual convergence, as a steady simulation cannot fully converge an inherently unsteady wake (see Appendix A.1).

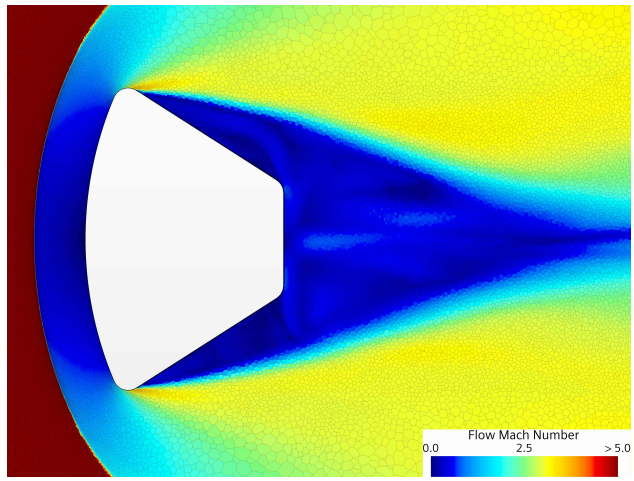


Figure 3: Converged 3D mesh of the Orion CM geometry at $M = 5$, $\alpha = 0°$. Centreplane cross-section overlaid with Mach number contour.

3.5 CFD simulation time

Each simulation converged after approximately 60k iterations, equating to about 130 hours of wall time on 128 CPU cores, i.e., approximately 16.6k CPU-hours per simulation.

4 Model: 3D neural fields

Next, we describe our neural network design.

4.1 Baseline configuration

The basic model configuration takes as input 3D coordinates in space and AoA, and predicts static pressure, temperature, and the three components of velocity:

$$(p, T, v_x, v_y, v_z) = f_1(x, y, z, \alpha). \tag{1}$$

After training, the neural network can rapidly predict the flow field for any AoA, and it can also, in principle, provide arbitrarily high spatial resolution.

4.2 Fourier positional feature mappings

Naively fitting a multi-layer perceptron (MLP), as in Equation 1, leads to poor performance because MLPs struggle to capture high-frequency variations in the input given their spectral bias. These variations are particularly prevalent in high-speed aerodynamics due to the presence of shock waves (abrupt, nearly discontinuous changes in the properties of a fluid), which the neural network perceives as steep gradients in the training data. To address this issue, we leverage Fourier positional feature mappings (Tancik et al., 2020)

which project the 3D coordinates, $\mathbf{x} = (x, y, z)$ to sinusoidal embeddings using a Gaussian projection matrix: $\Gamma(x, y, z) = \Gamma(\mathbf{x}) = [\cos(2\pi\mathbf{B}\mathbf{x}), \sin(2\pi\mathbf{B}\mathbf{x})]^T \in \mathbb{R}^{2m}$, where $\mathbf{B} \in \mathbb{R}^{m\times 3}$ is a random projection matrix sampled from $\mathcal{N}(0, \sigma^2)$. Both m and σ are hyperparameters which we adjust experimentally. Hence, the model with Fourier positional mappings can be described as:

$$(p, T, v_x, v_y, v_z) = f_2(\Gamma(x, y, z), \alpha). \tag{2}$$

Note that the Fourier feature mappings are applied only to the spatial coordinates. This design choice reflects the fact that, within the considered regime, the flow field varies "smoothly" with respect to AoA. Therefore, the inherent spectral bias of MLPs toward low-frequency functions is desirable, as it promotes smooth interpolation across angles and discourages spurious high-frequency variations along this global parameter. Equation 1 and Equation 2 constitute our "purely" data-driven baselines. Next we describe how we incorporate physical constraints into the model.

4.3 No-slip boundary condition

The no-slip boundary condition in fluid dynamics states that the fluid velocity at the wall is equal to the velocity of the wall itself. For a stationary wall, this means the fluid immediately next to the surface has zero velocity. Just above this layer of "stuck" fluid, the velocity gradually increases to match the free-stream flow. This gradual change in velocity forms a boundary layer: a thin region near the wall where viscous effects dominate. The boundary layer is critical in aerothermodynamics because it affects drag, surface heating, and the overall behavior of the flow. To model this effect in the neural network, we define κ as the perpendicular distance from the wall with $\kappa = 0$ on the surface (see Appendix A.2). We then scale the predicted velocity components to enforce the no-slip condition:

$$(p, T, v_x, v_y, v_z)_{\mathrm{BLv}} = \big(p, T, v_x(1 - e^{-\beta_1\kappa}), v_y(1 - e^{-\beta_2\kappa}), v_z(1 - e^{-\beta_3\kappa})\big), \tag{3}$$

where (p, T, v_x, v_y, v_z) are the outputs of the Fourier-based network f_2 and $\beta_1, \beta_2, \beta_3$ are learnable scalings. Importantly, κ is not an input to the network; it is applied only to scale the velocity outputs after prediction. Pressure and temperature remain unchanged, so the rest of the model behaves identically to f_2. This approach biases the solution toward physically plausible boundary layer profiles while keeping the underlying neural network unchanged. Note that this trick does not simulate real viscous physics: the shape of the boundary layer is imposed by the function $1 - e^{-\beta\kappa}$, not by solving the Navier-Stokes equations. Also, in practice we are just modulating the network output near the wall rather than forcing a fixed exponential profile: the multiplicative factor ensures that the velocity goes to zero at the wall, but the neural network can still produce non-monotonic or complex variations in the boundary layer away from the wall. Thus, it is more flexible than a simple prescribed boundary layer.

4.4 Isothermal wall boundary condition

Specifying that a wall is isothermal means that the wall is maintained at a fixed temperature throughout the simulation, in our case this value is $T_w = 300$ K. This is often used to model realistic cases, such as actively cooled spacecraft surfaces, where engineers control the wall temperature to prevent overheating during high-speed flight. Importantly, an isothermal wall does not impose a particular temperature gradient at the surface; rather, the gradient develops naturally based on the flow conditions and heat transfer between the fluid and the wall. This boundary condition influences the behavior of the boundary layer, heat flux to the surface, and overall aerothermal characteristics, making it a useful approximation when the wall temperature can be actively regulated. Similar to the no-slip boundary condition for velocity, enforcing an isothermal wall produces a thermal boundary layer. To encourage the network to respect this condition, we scale the predicted temperature based on the perpendicular distance from the wall:

$$T_{\mathrm{BLT}} = T_w + \big(T - T_w\big)\big(1 - e^{-\beta_4\kappa}\big), \quad \text{with } T = f_2^T(\Gamma(x, y, z), \alpha), \tag{4}$$

where T is the temperature predicted by the Fourier-based network f_2 and β_4 is a learnable scaling. At the wall ($\kappa = 0$), this ensures $T_{\mathrm{BLT}} = T_w$, while away from the wall the network retains full flexibility to produce complex temperature profiles.

5 Other surrogate alternatives

In this section, we motivate the use of neural fields as compared to other surrogate models. We perform an empirical exploration in Section 6.2.

5.1 Neural operators

It is useful to distinguish the learning problem addressed in this work from the one typically considered in the neural operator literature (Kovachki et al., 2021). In our particular case, the target quantity is a parametric flow field $\mathbf{q}_\alpha(\mathbf{x}) = \big(p(\mathbf{x},\alpha),\, T(\mathbf{x},\alpha),\, v_x(\mathbf{x},\alpha),\, v_y(\mathbf{x},\alpha),\, v_z(\mathbf{x},\alpha)\big), \qquad \mathbf{x} = (x,y,z) \in \mathbb{R}^3$. Our model directly approximates this field through a coordinate-based representation $\hat{\mathbf{q}}_\alpha(\mathbf{x}) = f_\theta(\mathbf{x},\alpha)$, or, with Fourier features as in Equation 2, $\hat{\mathbf{q}}_\alpha(\mathbf{x}) = f_\theta\big(\Gamma(\mathbf{x}),\alpha\big)$. This is a *neural field*: the spatial coordinate $\mathbf{x}$ is part of the input, and the network returns the predicted state at that location.

By contrast, a neural operator aims to learn a mapping between *functions*. In abstract form, one seeks an operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$, where $\mathcal{A}$ is a space of input functions or conditions and $\mathcal{U}$ is a space of output solution fields. In the context of this work, one may idealize such a mapping as $\mathcal{G}(\alpha) = \mathbf{q}_\alpha(\cdot)$, that is, the input is the operating condition α, while the output is the *entire* flow field $\mathbf{q}_\alpha(\cdot)$ over space. Evaluating the predicted solution at a point $\mathbf{x}$ then corresponds to

$$\hat{\mathbf{q}}_\alpha(\mathbf{x}) = \big[\mathcal{G}_\theta(\alpha)\big](\mathbf{x}). \tag{5}$$

The key distinction is that for neural fields the coordinate $\mathbf{x}$ is an input to the model, whereas in Equation 5 the model outputs a full spatial field and $\mathbf{x}$ appears only when that field is evaluated. In simple terms, in a neural field, we ask the network for one point at a time. In a neural operator, the network gives the whole field at once.

This difference has important implications for our data regime. A neural operator is trained across multiple realizations of the underlying parametric problem, $\{\alpha^{(i)},\, \mathbf{q}_{\alpha^{(i)}}(\cdot)\}_{i=1}^{N}$, and therefore benefits primarily from having a large number N of distinct CFD solutions. In contrast, the neural-field formulation exploits the fact that each single CFD realization provides a very large number of supervised spatial samples, $\{(\mathbf{x}_j, \alpha^{(i)}),\, \mathbf{q}_{\alpha^{(i)}}(\mathbf{x}_j)\}_{j=1}^{M_i}$, with M_i equal to the number of mesh points in simulation i. In our case, N is small because each hypersonic CFD run is extremely expensive (130 hours each), while each M_i is very large due to the high-resolution three-dimensional mesh. This makes the coordinate-based neural-field formulation particularly well suited to our setting, since it effectively transforms a "small data" problem into a "big data" problem and is hence strategic for this specific engineering bottleneck.

Many widely used neural operator architectures (Li et al., 2021), are most convenient on structured discretizations, whereas our CFD data is defined on large unstructured meshes with locally refined resolution near shocks and walls (Section 3). Coordinate-based neural fields avoid this restriction, since they only require point locations and corresponding target states. For this reason, they provide a simple and natural way to learn continuous surrogates on irregular CFD discretizations.

5.2 Graph neural networks

Following this discussion, a natural alternative is GNNs, which are able to operate on irregular grids. The first challenge in this case, is that standard commercial software used for large-scale CFD simulations with advanced mesh refinement do not typically allow one to export the final mesh. As a consequence, we cannot do message passing based on the simulation grid, and we must construct k-NN graphs using spatial proximity instead. A message passing GNN layer l (ignoring edge and graph level features for simplicity here) is computed as

$$\mathbf{h}_i^{(l+1)} = \phi\Big(\mathbf{h}_i^{(l)}, \bigoplus_{j\in\mathcal{N}(v_i)} \psi(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)})\Big). \tag{6}$$

Note that here we use i and j to index (latent) features referring to the node v_i (do not mistake with velocity) at which we perform a computation and its neighboring node, respectively. $\mathcal{N}(v_i)$ is the one-hop neighborhood. Also, ψ is a message passing function, $\bigoplus$ is an aggregation function, which must be permutation-invariant, and lastly, ϕ is a readout or update function. Depending on the GNN ψ and ϕ are implemented differently. Importantly, the update equation is local and it is only dependent on the one-hop neighborhood of the node at each layer. For clarity, Equation 6 can be further decoupled into three update rules (Borde & Bronstein, 2025):

$$\mathbf{m}_{ij}^{(l)} \leftarrow \psi(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}); \quad \mathbf{a}_i^{(l)} \leftarrow \bigoplus_{j\in\mathcal{N}(v_i)} \mathbf{m}_{ij}^{(l)}; \quad \mathbf{h}_i^{(l+1)} \leftarrow \phi(\mathbf{h}_i^{(l)}, \mathbf{a}_i^{(l)}) \qquad \text{(Message, Aggregate, Update)}$$

This localized bias is a good inductive bias for many classical graph representation learning tasks such as transductive learning over citation networks (Yang et al., 2016), but can also be a potential weakness, as we will show in Section 6.2. This is because of the homophily bias intrinsic to famous GNNs (Kipf & Welling, 2016), which states that nearby neighbors have similar targets labels. Effectively, they act as low-pass filters, which is counterproductive when trying to learn sharp gradients and discontinuities as those present in hypersonic flows.

6 Quantitative results and aerothermodynamic analysis

In the following section we empirically evaluate our methodology. We start by studying the importance of Fourier mappings in the context of hypersonic flow prediction, in particular focusing on our 3D generalization of previous 2D studies (Borde et al., 2023). Next, we perform a one-to-one comparison with other surrogate alternatives from Section 5, which we find to be suboptimal for hypersonics. Lastly, we discuss the importance of enforcing boundary conditions, and analyze the final simulation time as compared to traditional CFD.

6.1 On the importance of Fourier mappings for learning hypersonic flow features

The neural field architecture is trained using a 10-layer MLP with a hidden dimension of 1024. We utilize the Adam optimizer and MSE loss for 20 epochs with a learning rate of 10^{-4} and a mini-batch size of 256. We start our experimental procedure by calibrating the model hyperparameters. All learnable scaling coefficients are initialized to $\beta_1 = \beta_2 = \beta_3 = \beta_4 = 5$, but are free to vary independently during optimization. The AoA training split consists of $\{0°, 5°, 10°, 20°, 25°, 35°, 40°, 45°\}$ while validation is performed on the unseen angles $\{15°, 30°\}$.

In particular, we compare the baseline configuration presented in Equation 1 to that in Equation 2. In Table 1, we observe that adding Fourier mappings improves performance as compared to passing the coordinates directly to the MLP, as expected. We perform a grid search over $\sigma \in \{5, 15, 30, 45, 60, 75, 85, 90, 100\}$. The parameter σ directly controls the frequency spectrum of functions the network can represent and learn efficiently. For low values of $\sigma \in [5, 15]$, the network behaves similarly to a standard MLP with a strong spectral bias toward low frequencies, and hence promotes learning smooth, slowly varying functions. This leads to underfitting near shocks and smeared discontinuities. On the other hand, for high values of $\sigma > 75$, very high-frequency components dominate, and the network can represent extremely sharp features, but optimization becomes harder and there is a risk of fitting noise or mesh artifacts. Indeed, we find the sweet spot to be at intermediate values, which provide a balanced spectrum and allow the network to effectively learn both smooth free-stream flow features and sharp gradients. Concretely, $\sigma = 45$ empirically provides a good trade-off in terms of expressivity and generalization. We use this configuration for all experiments in the next subsection.

Table 1: Final validation MSE for different encoding configurations.

Encoding	None (Base model)	σ=5	σ=15	σ=30	σ=45	σ=60	σ=75	σ=85	σ=90	σ=100
Val. MSE	0.01539	0.00856	0.00726	0.00635	**0.00484**	0.00485	0.00526	0.00528	0.00536	0.00557

6.2 Empirical comparison with other surrogate alternatives

Following the discussion in Section 5, we focus on comparing neural fields to GNN-based surrogates. We consider two types of GNNs: GCN (Kipf & Welling, 2017) and GAT (Veličković et al., 2018), each with 10 layers and augmented with Fourier positional feature mappings with $\sigma = 45$ (based on our best configuration in Table 1). This layer count is chosen to roughly match the number of parameters of the MLP-based neural field (approximately 20 million trainable weights and biases).

We construct a k-NN graph using the 3D spatial coordinates of the nodes. To investigate the impact of graph connectivity, we perform an ablation study by varying the parameter k, which determines the number of neighbors per node. Note that the effective receptive field (depth) of the GNN corresponds to its number of layers. As we can see in Table 2, as we increase k the validation loss increases. $k = 2$ (the sparsest case we explore) gives the best performance, although this is still worse than neural fields using the same frequency encodings which achieves a validation loss of 0.00484.

Table 2: Validation loss for different k values and $\sigma = 45$.

k	2	4	6	8	10
GCN	0.00710	0.00714	0.00782	0.00947	0.00923
GAT	0.00663	0.00670	0.00678	0.00775	0.00815

In Table 3, we further average results over 3 random seeds for $k = 2$, and test removing the Fourier mappings. We can see that Fourier mappings help GNN performance too. The GAT model performs slightly better than the GCN, but still does not beat the neural field. This empirically verifies our discussion in Section 5.2 regarding the unsuitability of GNNs for hypersonic flows. This is further supported by studying the dissipative behavior of GNNs at the hypersonic bow shock and the boundary layer in Section 7.1.

Table 3: Validation loss (mean $\pm$ std over 3 seeds) for GCN and GAT.

Model	No Fourier	Fourier ($\sigma = 45$)
GCN	0.06771 $\pm$ 0.00093	0.00706 $\pm$ 0.00053
GAT	0.06645 $\pm$ 0.00311	0.00682 $\pm$ 0.00068

Lastly, it is important to note that we cannot further test message-passing propagation over the original mesh connectivity structure, since most commercial CFD software does not allow exporting it. This is a further logistical constraint for GNNs. Nevertheless, by visually inspecting Figure 3, we can observe that most nodes have 4 or more neighbors and are also connected to nearby nodes in terms of spatial coordinates. Ergo, we would not expect a GNN that uses the original mesh adjacency matrix to outperform the neural fields; instead, its performance is likely to be similar to the results in Table 2.

6.3 Adding physical boundary conditions

To assess the impact of the boundary-condition enforcement, we compare four configurations: no boundary conditions, no-slip velocity (BC_v), isothermal wall (BC_T), and both combined. As shown in Table 4, enforcing boundary conditions substantially improves performance in unseen AoAs, with the largest gains achieved when both velocity and thermal wall conditions are applied. We use the best-performing model for all subsequent analyses.

Table 4: Effect of boundary condition enforcement on val AoAs (mean ± std across runs, 3 seeds). $\sigma = 45$.

Boundary conditions	None	BC_v	BC_T	BC_v + BC_T
Final Val. MSE	0.00483 ± 0.00039	0.00444 ± 0.00035	0.00292 ± 0.00181	**0.00228 ± 0.00002**

6.4 Neural simulation time

Each full-field prediction (pressure, temperature, and all velocity components) takes less than 5 seconds on a single node with 8 H100 GPUs, parallelized with a batch size of 2,097,152 per GPU, or approximately 37 seconds when run serially on a single H100. Prediction time can be reduced further through increased parallelization (e.g., to under 3 seconds using two nodes) or using optimized CUDA kernels which we do not explore here. This contrasts with 130 hours of wall time using traditional CFD (Section 3). That is, a 93,600x speedup when using an H100 node, and a 156,000x when using two.

7 Aerothermodynamic analysis

Next, we evaluate the model's predictions from an aerodynamic perspective. We follow the same progression as in Section 6.

7.1 Challenges capturing sharp discontinuities in hypersonic flows

In Figure 4 we compare the error near the capsule (with respect to the CFD simulation) for the baseline neural field vs that produced by the neural field with Fourier $\sigma = 45$. More concretely, the mismatch between the ground truth and the model approximation is plotted using the normalized percentage error $\mathcal{E}_i = 100 \cdot \frac{|q_i - \tilde{q}_i|}{\max\left(|\tilde{q}_i| + |q_i|,\ \tau\right)}$ [%] where q_i is the true state and $\tilde{q}_i$ approximated state and τ is a small positive threshold used in the denominator so the percent error remains bounded when the states are near zero.

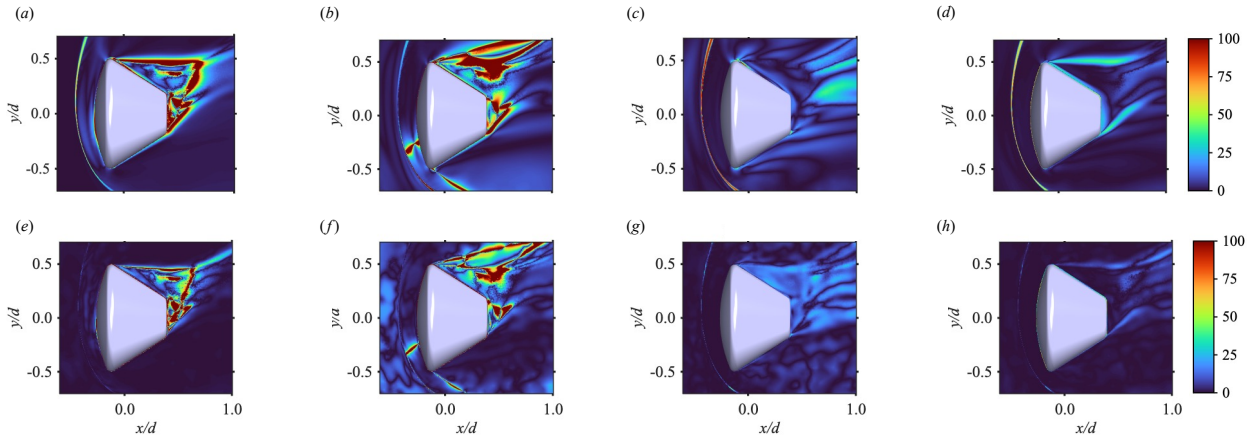


Figure 4: Flow field prediction error with respect to CFD ground truth on the $z/d = 0$ plane at $\alpha = 15°$ (validation AoA), comparing neural field models with (e, f, g, h) and without Fourier features (a, b, c, d), as in Table 1. In terms of columns, we have the horizontal v_x/U_∞ $(a,\ e)$, vertical velocity v_y/U_∞ $(b,\ f)$, pressure p/p_∞ $(c,\ g)$, and temperature T/T_∞ $(d,\ h)$. Contours show the normalized percentage error.

As shown, without Fourier positional feature mappings (top row), errors are localized around the shock and other features such as the free-shear layer, the Prandtl-Meyer expansions, and the recirculating flow behind the capsule (features critical to hypersonic flows) whereas with $\sigma = 45$ (bottom row), the error distribution becomes more uniform and practically disappears at the bow shock.

Next, we show that GNNs struggle to capture hypersonic discontinuities. In particular, we compare the neural field model against GAT predictions with $k = 2$ and $k = 10$; all of them use Fourier positional feature

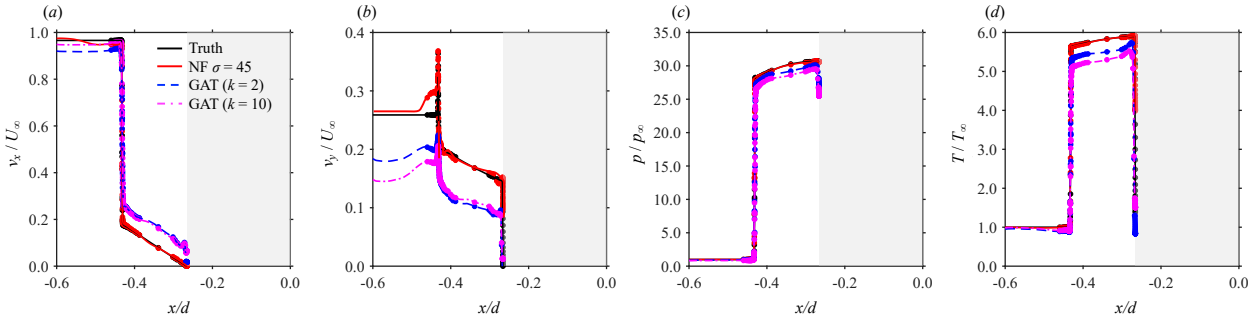


Figure 5: Shock gradient predictions along the $y/d = 0$ symmetry line and plane $z/d = 0$ for $\alpha = 15°$. (a) v_x/U_∞, (b) v_y/U_∞, (c) p/p_∞, (d) T/T_∞. The shaded gray area marks the capsule body.

mappings with $\sigma = 45$ (Table 2). In Figure 5 we extract the main bow shock and boundary layer gradients by seeding the volumetric flow data along the symmetry line $y/d = 0$ on the symmetry plane $z/d = 0$, as conveniently done in aerodynamic 3D flows around symmetric bodies. In the present case, the free-stream velocity vector has projections only in x and y (since the sideslip angle is zero), therefore the flow has strong in-plane gradients that can be extracted from $z/d = 0$.

Additionally, to extract gradients in the v_z velocity component, which otherwise remains virtually zero at $z/d = 0$ due to symmetry, the $y/d = 0$ line is sampled on the $z/d = 0.2$ plane, as illustrated in Figure 6. The GNNs (both with $k = 2$ and $k = 10$) clearly struggle to capture sudden jumps in velocity, pressure, and temperature near the shock. Notably, a more pronounced smoothing of the steep gradients in the flow states can be observed with a larger k, as expected due to the dissipative nature driven by neighborhood aggregation in message-passing. This aligns with the higher validation error in Table 2.

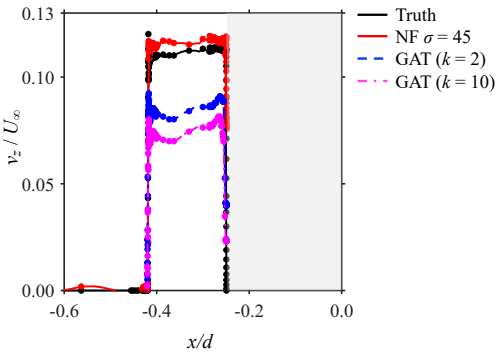


Figure 6: The normalized velocity v_z/U_∞ is extracted on the $z/d = 0.2$ plane and along the $y/d = 0$ symmetry line ($\alpha = 15°$).

7.2 Flow field prediction

After determining that the neural field with Fourier positional feature mappings performs best (both from a machine learning metrics perspective and from an aerothermodynamic analysis perspective) we further explore the predictions of this configuration. In particular, we focus on the best model in Table 4, which uses all boundary conditions. The horizontal and vertical velocities v_x and v_y, static pressure p and static temperature T spatial fields are displayed in Figure 7 for both the CFD data serving as the ground truth and the model prediction at $\alpha = 15°$. The data is extracted from a plane at $z/d = 0$ and interpolated on a structured x-y grid with uniform spacing $\Delta x/d = 0.004$ and $\Delta y/d = 0.003$.

The $(a,\ b)$ panels in Figure 7 show that the model has learned the main flow features. The bow shock is resolved exceptionally well and also the dominant wake orientation and separation region are captured. Most of the error (panel c) is within the recirculatory flow zone, however the fine details of the flow therein are not physical given the unsteady nature of the flow and the type of simulations employed (see Appendix A.1). Any inaccuracies from the model are therefore not concerning. Similar observations can be drawn from the vertical velocity v_y (panels $d,\ e,\ f$) except for the lower side of the bow shock in $x/d \in [0.0, 1.0]$, $y/d \in [-1.0, -0.5]$ where the model prediction displays some noise. This is likely due to the lower density mesh in this region of the flow. Note that the lower bow shock region of the flow is of limited engineering interest as it does not influence the pressure or heat flux on the vehicle, justifying the lower mesh density. For the sake of clarity, these inaccuracies were found also in the non-interpolated data so they are not a mere visualization issue but a real model behavior. The pressure and temperature states (panels $g,\ h,\ i$ and $j,\ k,$

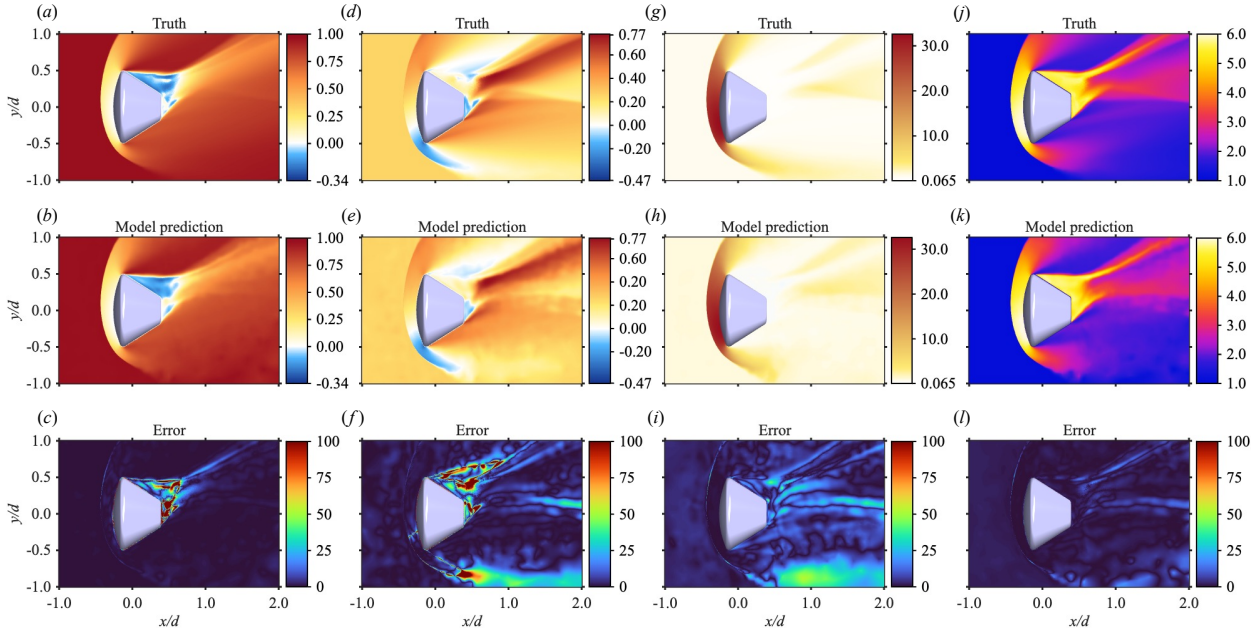


Figure 7: Comparison of the horizontal v_x/U_∞ (a, b), vertical velocity v_y/U_∞ (d, e), pressure p/p_∞ (g, h), and temperature T/T_∞ (j, k) fields between the ground truth and model predictions at $\alpha = 15°$. The normalized percentage error is shown in (c, f, i, l). Contours extracted on the plane $z/d = 0$.

l) are reconstructed very accurately; the pressure being affected in the lower bow shock area seemingly by a similar reason to the vertical velocity.

The overall $\mathcal{L}_2$-norm based error is computed from the original CFD grid points as $\mathcal{L}_2 = 100 \cdot \frac{\|q_i - \tilde{q}_i\|_2}{\|q_i\|_2 + \|\tilde{q}_i\|_2}$ [%]. The $\mathcal{L}_2$ errors are 4.92% for v_x, 8.20% for v_y, 11.97% for v_z, 3.04% for pressure p, and 6.24% for temperature T. All states except the v_z velocity (not shown for brevity in Figure 7) are approximated with less than 10% error, the pressure state p displaying only $\approx 3\%$ error. The v_z velocity is the least accurate perhaps due to its comparably smaller magnitude.

7.3 Pressure distribution prediction on the surface of the Orion reentry capsule

In high-speed bluff body flows, the major contributor to the total aerodynamic drag exerted on the vehicle originates from pressure forces (Anderson, 2003).

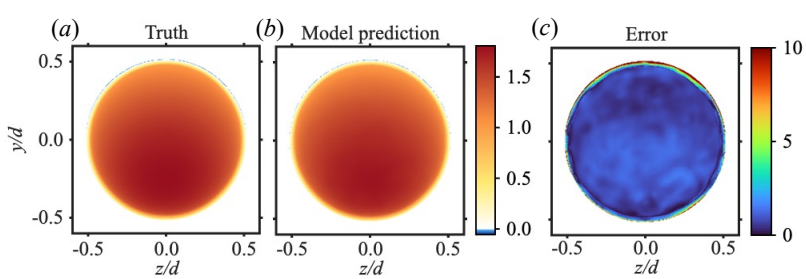


Figure 8: Comparison of the pressure coefficient C_P on the frontal surface of the capsule for (a) the ground truth and (b) the model prediction at $\alpha = 15°$. The normalized percentage error is presented in (c).

Hence, we examine the predictive accuracy of the model on the pressure coefficient distribution. Figure 8 presents a comparison of the pressure coefficient C_P, computed as

$$C_P = \frac{\frac{p}{p_\infty} - 1}{0.5\gamma M_\infty^2}. \tag{7}$$

The figure reveals that the model predicts the distribution over the capsule forebody up to a high degree of accuracy, with maximum error within 10% occurring on the shoulder (Figure 9) where the flow accelerates through an expansion fan. Strong pressure gradients are therefore present in this region of flow and prediction is clearly more challenging compared to locations with more gentle surface curvature. Nonetheless, the recorded error is acceptable.

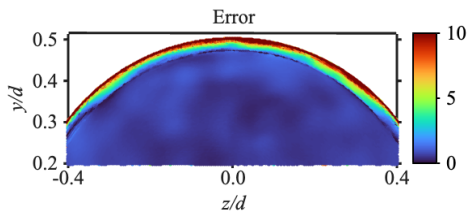


Figure 9: Zoomed-in view of the error on the C_P prediction over the capsule shoulder region at $\alpha = 15°$.

These results indicate that the pressure coefficient derived from the predicted pressure field is successfully reconstructed for an AoA not seen by the model during the training phase. The same set of plots presented in this section is repeated in Appendix B for $\alpha = 30°$.

8 Conclusion

We have shown that neural fields can serve as effective hypersonic simulators, with the potential to reduce simulation time from hundreds of hours to seconds. Although CFD can also be parallelized, the iterative nature of its solvers often limits strong scaling, so adding nodes typically does not yield near-linear reductions in wall time; in contrast, batched neural inference does, which is another key strength of our method. Our results also corroborate that Fourier features can effectively learn sharp discontinuities characteristic of hypersonic flows and that enforcing physics-informed boundary conditions substantially improves model performance. Finally, we have characterized the limitations of alternative surrogate models, concluding that neural fields provide superior logistical flexibility by bypassing the high data requirements of neural operators and the mesh-connectivity constraints of GNNs, while simultaneously capturing the complex discontinuities inherent to the hypersonic regime.

Our overarching long-term objective is to streamline the processes of aerodynamic design, mission analysis, and control, enabling faster and more thorough exploration of potential solutions during the initial phases of space mission development. Achieving this vision requires general models capable of effectively simulating the flow field around any type of geometry, which is, of course, a data-hungry endeavor.

References

J. D. Anderson. *Computational Fluid Dynamics: The Basics with Applications*. McGraw–Hill, 1995.

J.D. Anderson. *Modern Compressible Flow: With Historical Perspective*. Aeronautical and Aerospace Engineering Series. McGraw-Hill Education, 2003.

H. S. O. Borde, D. Sondak, and P. Protopapas. Convolutional neural network models and interpretability for the anisotropic reynolds stress tensor in turbulent one-dimensional flows. *Journal of Turbulence*, 23:1 – 28, 2021a.

H. S. O. Borde, D. Sondak, and P. Protopapas. Multi-task learning based convolutional models with curriculum learning for the anisotropic reynolds stress tensor in turbulent duct flow. *ArXiv*, abs/2111.00328, 2021b.

H. S. O. Borde, P. Innocenzi, and F. Savarino. Aerothermodynamic simulators for rocket design using neural fields. In *57th 3AF International Conference on Applied Aerodynamics (AERO 2023)*, Bordeaux, France, March 29–31 2023. URL `https://arxiv.org/abs/2303.10283`.

Haitz Sáez de Ocáriz Borde and Michael Bronstein. Mathematical foundations of geometric deep learning, 2025. URL `https://arxiv.org/abs/2508.02723`.

J. D. Brown, D. W. Bogdanoff, L. A. Yates, and G. T. Chapman. Transonic aerodynamics of a lifting orion crew capsule from ballistic range data. *Journal of Spacecraft and Rockets*, 47(1):36–47, 2010. doi: 10.2514/1.37878.

O. Durmaz, H. D. Karaca, G. D. Ozen, C. Kasnakoglu, and D. F. Kurtulus. Dynamical modelling of the flow over a flapping wing using proper orthogonal decomposition and system identification techniques. *Mathematical and Computer Modelling of Dynamical Systems*, 19(2):133–158, 2013.

R. Fang, D. Sondak, P. Protopapas, and S. Succi. Neural network models for the anisotropic reynolds stress tensor in turbulent channel flow. *Journal of Turbulence*, 21:525 – 543, 2019.

K. Fukami, K. Fukagata, and K. Taira. Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *Journal of Fluid Mechanics*, 909, 2021.

R. Giannopoulos and J. Aider. Data-driven order reduction and velocity field reconstruction using neural networks: The case of a turbulent boundary layer. *Physics of Fluids*, 32, 2020.

J. Guzman-Inigo, M. A. Sodar, and G. Papadakis. Data-based, reduced-order, dynamic estimator for reconstruction of nonlinear flows exhibiting limit-cycle oscillations. *Physical Review Fluids*, 4(11), 2019.

M. L. A. Kaandorp. Machine learning for data-driven rans turbulence modelling. 2018.

H. Kim, J. Kim, S. Won, and C. Lee. Unsupervised deep learning for super-resolution reconstruction of turbulence. *Journal of Fluid Mechanics*, 910, 2021. doi: 10.1017/jfm.2020.1028.

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2016. URL `https://api.semanticscholar.org/CorpusID:3144218`.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

Nikola B. Kovachki, Zong-Yi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *ArXiv*, abs/2108.08481, 2021. URL `https://api.semanticscholar.org/CorpusID:237213378`.

J. Li and M. Zhang. On deep-learning-based geometric filtering in aerodynamic shape optimization. *Aerospace Science and Technology*, 112, 2021.

J. Li, X. Du, and J. Martins. Machine learning in aerodynamic shape optimization. *ArXiv*, abs/2202.07141, 2022.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=c8P9NQVtmnO`.

J. Ling, A. Kurzawski, and J. A. Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155 – 166, 2016.

M. Liou. A sequel to ausm: Ausm+. *Journal of Computational Physics*, 129:364 – 382, 1996.

J. Liu, R. Chen, J. Lou, Y. Hu, and Y. You. Deep-learning-based aerodynamic shape optimization of rotor airfoils to suppress dynamic stall. *Aerospace Science and Technology*, 2023.

J. C. Loiseau, B. R. Noack, and S. L. Brunton. Sparse reduced-order modelling: Sensor-based dynamics to full-state estimation. *Journal of Fluid Mechanics*, 844:459–490, 2018.

A. Lozano-Duran and H. J. Bae. Self-critical machine-learning wall-modeled les for external aerodynamics. 2020.

K. Mikhaylov, S. Rigopoulos, and G. Papadakis. Reconstruction of large scale flow structures in a stirred tank from limited point measurements. *AIChE Journal*, 2021.

N. J. Nair and A. Goza. Leveraging reduced-order models for state estimation using deep learning. *Journal of Fluid Mechanics*, 897, 2020. doi: 10.1017/jfm.2020.409.

NASA. Orion spacecraft. `https://www.nasa.gov/reference/orion-spacecraft/`, 2023. Accessed: 2025-12-28.

C. Navier. Mémoire sur les lois du mouvement des fluides. *Mémoires de l'Académie Royale des Sciences de l'Institut de France*, pp. 389–440, 1822.

S. Nidhan, K. Chongsiripinyo, O. T. Schmidt, and S. Sarkar. Spectral proper orthogonal decomposition analysis of the turbulent wake of a disk at re = 50 000. *Physical Review Fluids*, 5(12), 2020.

S. B. Pope. *Turbulent Flows*. Cambridge University Press, Cambridge, 2000.

M. Ripepi, M. J. Verveld, N. Karcher, T. Franz, M. Abu-Zurayk, S. Görtz, and T. Kier. Reduced-order models for aerodynamic applications, loads and mdo. *CEAS Aeronautical Journal*, 9:171–193, 2018.

P. J. Roache. *Computational Fluid Dynamics*. Hermosa Publishers, 1972.

V. Rozov and C. Breitsamter. Data-driven prediction of unsteady pressure distributions based on deep learning. *Journal of Fluids and Structures*, 104, 2021.

F. Savarino and G. Papadakis. Reconstruction of irregular flow dynamics around two square cylinders from sparse measurements using a data-driven algorithm. *arXiv*, abs/2209.03290, 2022.

M. J. Sayyari, J. Hwang, and K. C. Kim. Unsupervised deep learning of spatial organizations of coherent structures in a turbulent channel flow. *Physics of Fluids*, 2022.

O. T. Schmidt, A. Towne, G. Rigas, T. Colonius, and G. A. Bres. Spectral analysis of jet turbulence. *Journal of Fluid Mechanics*, 855:953–982, 2018.

G. Stabile, S. Hijazi, A. Mola, S. Lorenzi, and G. Rozza. Pod-galerkin reduced order methods for cfd using finite volume discretisation: Vortex shedding around a circular cylinder. *Communications in Applied and Industrial Mathematics*, 8(1):210–236, 2017.

G. G. Stokes. On the theories of the internal friction of fluids in motion, and of the equilibrium and motion of elastic solids. *Transactions of the Cambridge Philosophical Society*, 8:287–305, 1845.

M. Tancik, P. P. Srinivasa, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J.T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *ArXiv*, abs/2006.10739, 2020.

J. Tu, G. H. Yeoh, and C. Liu. *Computational Fluid Dynamics: A Practical Approach*. Butterworth–Heinemann, 2 edition, 2018.

B. Van Leer. Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of Computational Physics*, 32:101, 136, 1979. doi: 10.1016/0021-9991(79)90145-1.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

G. Waxenegger-Wilfing, K. Dresia, J. C. Deeken, and M. Oschwald. Machine learning methods for the design and operation of liquid rocket engines - research activities at the dlr institute of space propulsion. *ArXiv*, abs/2102.07109, 2021.

M. Xu, S. Song, X. Sun, W. Chen, and W. Zhang. Machine learning for adjoint vector in aerodynamic shape optimization. *Acta Mechanica Sinica*, 2021.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, 2016. URL `https://api.semanticscholar.org/CorpusID:7008752`.

A Additional CFD data generation details

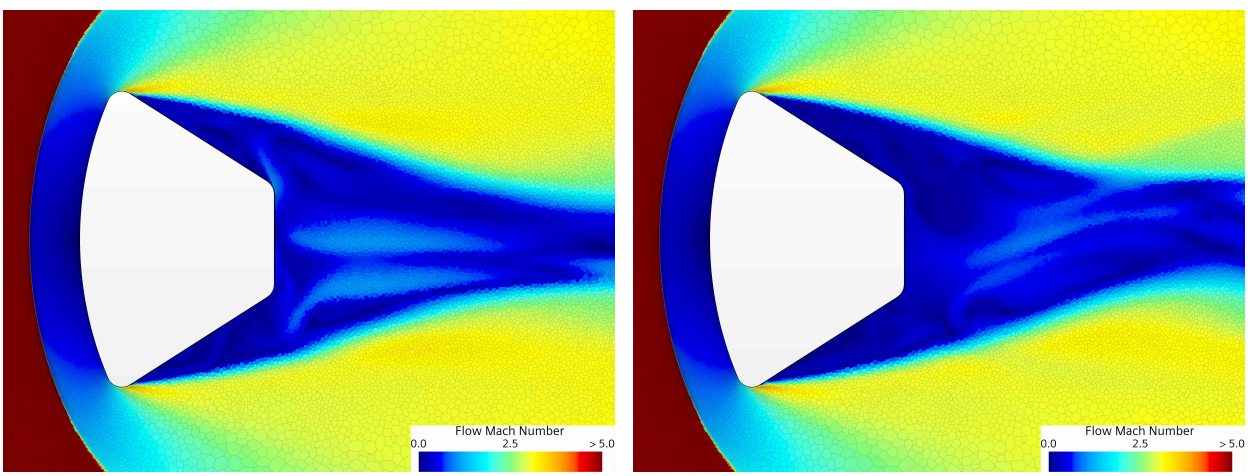
In this appendix we provide additional information regarding the CFD simulations and the calculation of other variables used for training the models.

A.1 Wake oscillations

A steady CFD simulation cannot converge an unsteady wake. Consequently, even when all other flow quantities are converged, the residuals associated with the wake oscillate between low values, and the flow field in the wake region changes from one iteration to another (see Figure 10).

Generally, this issue is addressed by computing a time-averaged solution of the entire flow field (in steady simulations, an *iteration-averaged* solution), in which all flow variables are averaged over a number of time steps (or iterations) corresponding to several convective time scales. This approach was not adopted in the present simulations but will be considered in future work. Instead, the neural network was trained using a single snapshot of the flow field at each AoA.

While this choice does not affect the shocks or boundary layers on the front of the model, which are essentially steady, it does influence the wake, which remains inherently unsteady and varies between iterations. As a consequence, larger prediction errors from the neural network are expected in this region of the flow.



(a) (b)

Figure 10: Snapshot Mach number contour over the centerplane cross-section at $M = 5$, $\alpha = 0°$. The Mach number values in the wake are different between the two iterations because of the unsteadiness in this region of the flow.

A.2 Wall Distance Computation

Let $\mathcal{S} \subset \mathbb{R}^3$ denote the set of points on the capsule surface. For any point $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$, the wall distance κ is defined as:

$$\kappa(\mathbf{x}) = \min_{\mathbf{s} \in \mathcal{S}} \|\mathbf{x} - \mathbf{s}\|_2. \tag{8}$$

To efficiently evaluate κ, a k-d tree is constructed from the wall point cloud using SciPy's `cKDTree` class. For each computational point x, the wall distance is obtained by querying the tree for the nearest surface point. Points identified as wall locations are explicitly assigned the value $\kappa = 0$, while for all other points κ equals the Euclidean distance to the nearest wall point returned by the k-d tree. The computation is performed in chunks to allow processing of the full dataset without exceeding memory limits, and the resulting values of the κ field are stored for subsequent analysis.

B Additional aerothermodynamic analysis results and visualizations

In this appendix, we present extended visualizations that complement the discussion in Section 6. In Figures 11 and 12 we show the prediction at $\alpha = 30°$ and report $\mathcal{L}_2$ errors of 5.09% for v_x, 6.55% for v_y, 11.50% for v_z, 1.58% for pressure p, and 6.19% for temperature T. Similarly to the $\alpha = 15°$ incidence case, the shock and wake structures are well approximated by the model as well as the pressure distribution on the frontal surface of the vehicle.

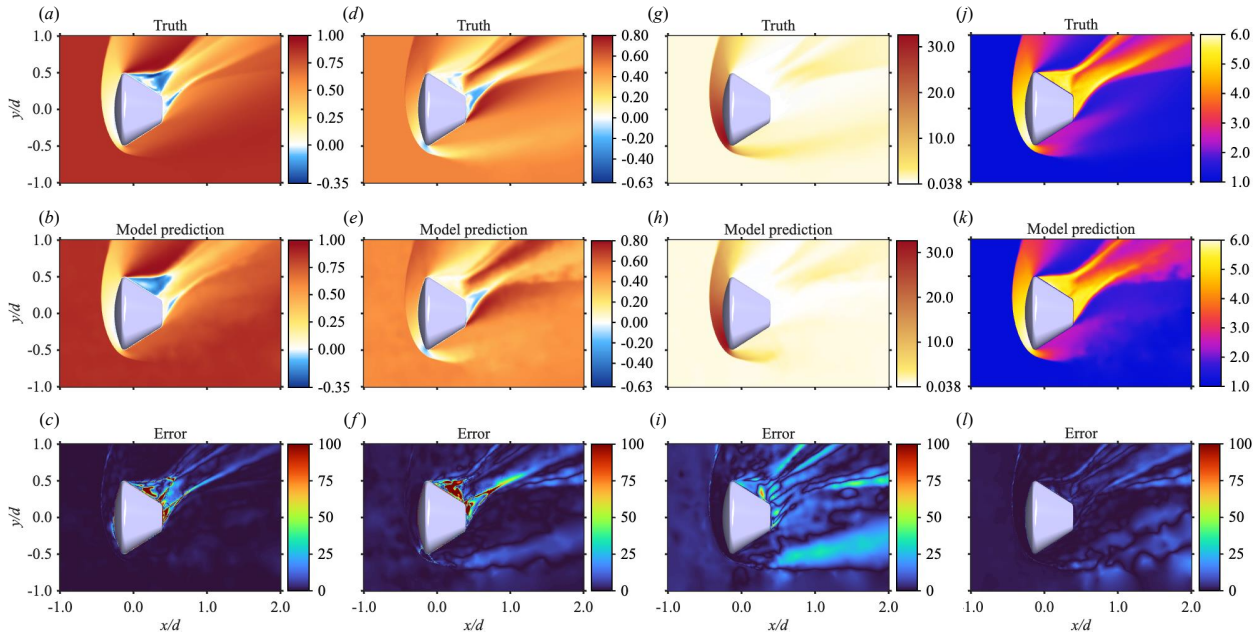


Figure 11: Comparison of ground truth and (best) model predictions at $\alpha = 30°$ for: horizontal velocity v_x/U_∞ (a, b), vertical velocity v_y/U_∞ (d, e), pressure p/p_∞ (g, h), and temperature T/T_∞ (j, k). The normalized percentage error is shown in (c, f, i, l). All contours are extracted on the plane $z/d = 0$.

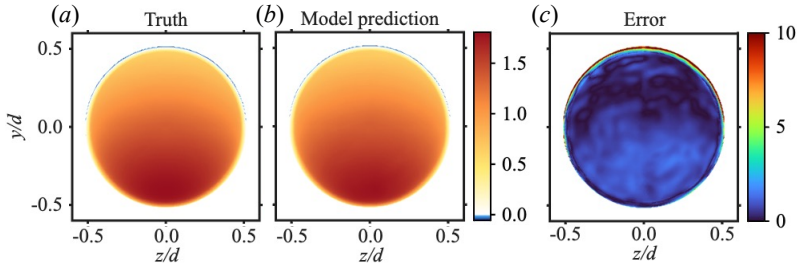


Figure 12: Comparison of the pressure coefficient C_P on the frontal surface of the capsule for (a) the ground truth and (b) the model prediction at $\alpha = 30°$. The normalized percentage error is presented in (c).