FOURIER NEURAL OPERATORS FOR TWO-PHASE, 2D MOLD-FILLING PROBLEMS RELATED TO METAL CASTING

- A PREPRINT -

Edgard M. Minete, Mathis Immertreu, Fabian Teichmann, Sebastian Müller

Friedrich-Alexander-Universität Erlangen-Nürnberg, Chair of Casting Technology, Fürth, 90762, Germany

October 31, 2025

ABSTRACT

A mold filling problem is a flow in which a fluid advances into a cavity and occupies it under given process and geometric constraints. In metal casting, mold filling is a representative filling problem where the hydrodynamics govern defect formation, microstructure, and the final cast part quality. Evaluating candidate designs that improve these outcomes often requires running many expensive transient computational fluid dynamics simulations, which slows the exploration of large configuration and parameter spaces. With this in mind, we borrow a real-world casting example and pose it as a simplified 2D operator learning problem. In our proposed method, a graph based encoder aggregates local neighborhood information on an input unstructured mesh and encodes geometry and boundary data. Then, a Fourier spectral core acts on a regular latent grid and captures global interactions across the domain. Finally, a graph based decoder projects the latent fields to a target mesh. Our model simultaneously forecasts velocities, pressure, and volume fraction over a fixed horizon and generalizes across varying ingate locations and process settings. On held out geometries and inlet conditions, it reproduces large scale advection and the fluid-air interface evolution with localized errors only near steep gradients. Mean relative L_2 errors are about $5\,\%$ across velocities, pressure and volume fraction fields. The solver runs 10^2 – 10^3 faster than traditional computational fluid dynamics and provides rapid predictions for design workflows. We additionally perform ablation studies that show monotonic accuracy loss with stronger spatial subsampling of input vertices, and a gentler deterioration with temporal subsampling. Lastly, investigations on the importance of the training dataset size show that our model presents only a small error growth when the training data are reduced by $50\,\%$. These results establish neural operators as efficient surrogates for 2D mold filling and filling problems in general, and enable fast design in the loop exploration and optimization of gating systems in metal casting.

Keywords Neural operator · Fourier neural operator · Surrogate modeling · Filling problems · Mold filling · Metal casting · Cahn–Hilliard–Navier–Stokes

Nomenclature

Symbol	Unit	Description
A	mm	Inlet size
B_{r_d}	mm	Ball of radius r_d, center x_i
B	mm	Inlet horizontal position
C	deg	Inlet angle
D_Ω	–	Input mesh
D_{wi}	mm	Distance to the initial interface
D	mm	Inlet height
E	mm	Outlet size
F_x	N/m^3	Surface tension force in x
F_y	N/m^3	Surface tension force in y
F	mm	Cavity height
G_b	mm	Generic point (ball B_r)
G	mm	Cavity width
H	–	Total number of prediction steps
I	–	Inlet setup
K	–	GNO local aggregation operator
L_{opt}	–	Training/optimization loss
N	–	Number of mesh vertices
R^ℓ	–	Learnable spectral weights
T	s	Temporal horizon of the NO
U	–	Latent signal
V	%	Inlet velocity magnitude on Γ_{in}

W	–	Point-wise linear map
Δt	s	Time step size in simulations
Γ_{in}	–	Inlet boundary
Γ_{out}	–	Outlet boundary
Γ_{w}	–	No-slip wall boundary
Ω	–	Mold / ingate computational domain
Ψ		Chemical potential
α	–	Volume fraction of fluid
β_n	rad	Angle of wall normal
χ	–	Mobility tuning parameter
ℓ	–	Per–step, per–field relative L_2 error
γ_t	–	Temporal weight
γ		Mobility parameter
$\hat{\alpha}$	–	Predicted volume fraction of fluid
$\hat{\theta}$	–	Neural network kernel parameters
$\hat{p}$	Pa	Predicted pressure
$\hat{q}$	–	Prediction field $\in \{\hat{u}, \hat{v}, \hat{p}, \hat{\alpha}\}$
$\hat{u}$	m/s	Predicted velocity in x-direction
$\hat{v}$	m/s	Predicted velocity in y-direction
κ	–	Learnable neural network kernel
λ		Mixing energy density parameter
$\hat{\mathbf{u}}$	m/s	Predicted velocity vector field
$\mathbf{u}$	m/s	Velocity vector field
$\mathbf{x_i}$	mm	Query vertice
$\mathbf{x}$	mm	Coordinate vector
$\mathcal{F}$	–	Fast Fourier Transform
$\mathcal{G}$	–	Geometry descriptor
$\mathcal{L}^l$	–	Fourier layer
$\mathcal{N}_\vartheta$	–	Neural operator
$\mathcal{S}$	–	Solution map
μ_1	Pa·s	Dynamic viscosity of fluid 1
μ_2	Pa·s	Dynamic viscosity of fluid 2
ϕ_0	–	Initial phase-field variable
ϕ	–	Phase-field variable
ρ_1	kg/m^3	Density of fluid 1
ρ_2	kg/m^3	Density of fluid 2
ρ	kg/m^3	Mixture density
τ	–	Causality parameter
θ_w	deg	Static contact angle at the wall
θ		Process parameters
ε	mm	Interface thickness
ζ	–	Inlet flow direction at Γ_{in}
a	–	Input Bundle on Ω
c	–	Cumulative past loss
f_{man}	–	Feature function
g	m/s^2	Gravitational acceleration
k	–	Prediction steps
p	Pa	Pressure
q	–	Target field $\in \{u, v, p, \alpha\}$
r_d	mm	Ball radius
r	–	Relative L_2 percentage error
s_d	–	Data availability percentage
s_s	–	Spatial subsampling factor
s_t	–	Temporal subsampling factor
t	s	Time
u	m/s	Velocity in x-direction
v	m/s	Velocity in y-direction
x	mm	Spatial coordinate in x
y	mm	Spatial coordinate in y

1 Introduction

The description and analysis of fluid flows and their intrinsic properties constitute a central focus of research due to their significant influence on phenomena across various scientific disciplines and everyday life. Understanding fluid behavior, governed by parameters such as viscosity, flow regime, and forces, is critical for applications that range from biological processes to engineering systems. In many of these fields, the flow of fluid filling cavities has become an area of growing interest. In manufacturing, fluid flows are particularly relevant in metal casting, where the flow of molten metal inside the mold strongly affects metallurgical properties, casting geometry, and the occurrence of defects, which in turn influence the mechanical performance of the final casting. Metal casting is among the oldest manufacturing processes of humanity, with archaeological evidence dating back over $7,000$ years [1]. Over time, numerous casting variants, from simple hand casting in sand to High Pressure Die Casting (HPDC) in complex metal molds, have been developed. Despite their differences, each variant involves molten metal being poured into a cavity, representing a specific filling problem. The design of the gating system, the casting geometry, melt properties, and process parameters together govern the filling dynamics. These factors control turbulence, air and oxide entrapment, heat transfer, and solidification pathways, thereby influencing defect formation, microstructure evolution, and ultimately, the mechanical properties of the casting [2, 3].

1.1 CFD simulation and optimization of mold filling in metal casting

Metal casting has evolved from early gravity-based methods such as lost wax and sand casting to modern pressure-assisted processes. Along this trajectory, HPDC consolidated controlled filling, thermal management, and automation, enabling thin walls, short cycles, and consistent quality for aluminum and magnesium components [4]. Understanding mold filling is essential for assessing casting quality. The gating system governs the hydrodynamics of melt flow and thereby the formation of defects and the evolution of microstructure. Poor layouts increase turbulence, entrain oxides, and promote bifilm formation, providing nucleation sites for porosity [2, 5]. The design of the gating system and the cross section of the ingates influence pressure losses, flow distribution, and solidification rates. Unsuitable choices lead to shrinkage and microstructural inhomogeneity [2, 3]. In thin-walled parts, multiple ingates shorten flow paths

and homogenize filling, supporting grain refinement [6]. Flow kinematics also control the dispersion of particles and additives [7], while even the orientation of the gating system alters head pressure and porosity distribution, affecting strength and ductility [8]. These mechanisms align with observations in aluminum castings, where large or clustered pores reduce ductility and increase property scatter [2, 3, 9–11]. Optimized gating reduces porosity and enhances strength by producing cleaner, denser microstructures [10]. Thus, gating design is a primary control variable for defect incidence, microstructure, and mechanical performance.

Mold design is therefore widely assisted by Computional Fluid Dynamics (CFD) simulation to predict melt flow and solidification. Since the late 20th century, commercial CFD tools [12–14] have enabled virtual exploration of mold filling by solving coupled flow and heat transfer, thereby accelerating design iteration cycles compared to trial-and-error. In practice, simulations have been applied to optimize gating layouts. Kwon and Kwon [15] used iterative Computer-Aided Engineering (CAE) analysis to refine a HPDC gating and overflow system to reduce air entrapment, while Zhao et al. [16] optimized thin-walled AlSi10MnMg HPDC beams with Flow-3D simulations, demonstrating reduced air defects and improved surface quality. In sand casting, Brůna et al. [17] combined simulation with experiments to show how gating design affects melt velocity, turbulence, and bifilm formation.

Recent advances in multiphase CFD extend these capabilities by capturing phenomena such as air entrainment, surface turbulence, and thermal gradients. Interface capturing approaches include the Volume-of-Fluid (VOF) method [18] and phase-field formulations. For instance, Fuwa et al. [19] applied a Cahn-Hilliard model to predict lamination defects in zinc alloy die casting. Advanced models also account for turbulence and Fourier type heat conduction [18]. While such high fidelity simulations improve predictive accuracy, fully coupled three-dimensional, two-phase, non-isothermal models remain computationally demanding [18]. Each candidate design still requires a numerically intensive transient simulation, which hinders the rapid exploration of large design spaces. As a result, gating design often depends on engineering heuristics, and selected solutions may only be locally optimal.

To address these challenges, optimization frameworks have been introduced. Adjoint based gradient optimizers such as the method of moving asymptotes [20] and its globally convergent variant [21], as well as multiobjective evolutionary algorithms such as NSGA-II [22], are commonly applied to engineering design problems. In casting, surrogate assisted approaches are increasingly explored. Shahane et al. [23] demonstrated neural network surrogates trained on finite volume simulations combined with NSGA-II for multi-objective optimization of die casting solidification, while Papanikolaou et al. [24] coupled CFD with NSGA-II in counter gravity casting, illustrating both the promise and computational constraints of such methods.

1.2 Neural solvers for filling problems

In the early 21st century, machine learning approaches began to complement traditional scientific methods [25]. The advent of Convolutional Neural Networks (CNNs), evolved from multilayer perceptron networks [26], represent an early machine learning success. CNNs take advantage of the structured grid, e.g. of images, and the hierarchical nature of the features via convolution and pooling layers, respectively, to fight the curse of dimensionality and simplify the learning problem. Recent research increasingly focuses on embedding biases into neural networks operating on structured and unstructured grid data [27]. Gilmer et al. [28] unified several graph models under the message passing neural network framework, where learned messages along edges and permutation invariant readout capture local interactions that compose into accurate global predictions on molecular benchmarks. Velickovic et al. [29] introduced attention on graphs so that a node can weight its neighbors adaptively, which improved both transductive and inductive node classification and showed that data driven neighborhood selection strengthens representation quality when graph structure is irregular. Complementing these advances, Kipf and Welling [30] proposed graph convolutional networks that approximate spectral filters with localized operations, enabling efficient semi supervised learning on citation networks and establishing a simple baseline that supports deeper architectural variants. On spherical signals from omnidirectional vision and climate data, Cohen et al. [31] designed spherical convolutions computed in the spectral domain to ensure rotation equivariance and reported gains on shape recognition and physics regression. In geometry processing of meshes, Masci et al. [32] constructed intrinsic geodesic patches so the network learns curvature aware features that support segmentation, retrieval, and correspondence on non-Euclidean surfaces. In Computational Fluid Dynamics (CFD), Li et al. [33] developed Partial Differential Equation (PDE) surrogates on irregular shapes by combining graph encodings of geometry with Fourier layers on a latent grid, achieving large computation speedups for three dimensional aerodynamics tasks such as pressure prediction and drag estimation.

Neural solvers are a set machine learning approaches employing neural network to solve differential equations. Neural solvers approaches can be broadly categorized into Physics-informed neural networks (PINNs), neural operators, and hybrid techniques combining traditional numerical methods with neural networks [34]. In PINNs [35], the solution field is represented by a neural network trained to penalize PDE residuals at interior collocation points while simultaneously enforcing Initial Conditions (IC)/Boundary Conditions (BC) at initial and boundary collocation points, respectively. PINNs have been successfully applied across fluids, structures, and dynamical systems [35–38], among others. However,

because the loss of the PINNs typically targets a specific PDE instance, geometry, and parameter set, models must be retrained or substantially fine-tuned when either changes, limiting scalability across the many distinct PDE instances encountered in a design optimization problem.

Neural operators learn mappings from function spaces to solution fields, approximating the solution operator itself rather than an individual PDE instance [34, 39]. Because of their broad applicability across geometric settings and parameter regimes, neural operators have attracted growing attention. For instance, Lu et al. [40] introduced DeepONets, a class of neural operators that combine a branch network for input functions and a trunk network for output coordinates and proved a universal approximation theorem for operators, establishing a practical architecture for supervised operator regression. Building on this foundation, Lin et al. [41] used operator learning to predict multiscale bubble growth where a network captured interface evolution across strong scale separation, while Oommen et al. [42] coupled neural operators with autoencoders to represent two-phase microstructure evolution and linked latent dynamics to physical order parameters. Li et al. [43] proposed Fourier Neural Operator (FNO), a class of neural operators that lifts fields to Fourier space, learns global integral kernels on selected modes, and projects back to the spatial domain. This design delivers resolution invariance and zero shot super resolution together with strong accuracy on Burgers, Darcy, and Navier Stokes systems and large speedups over spectral solvers. Follow up theory established universal approximation and error bounds that clarify conditions under which Fourier layers recover solution operators and thereby strengthened the mathematical foundation of this approach [44]. Qin et al. [45] further analyzed the spectral behavior of Fourier operators, identified a parametrization bias toward dominant frequencies, and proposed spectral boosting modules that recover non dominant content and reduce error on a range of PDE benchmarks. Beyond regular grids, Li et al. [46] introduced a geometry aware variant that learns a deformation to a uniform latent mesh so that Fourier layers act on irregular discretizations while preserving discretization convergence and accelerating inference across diverse geometries. Wen et al. [47] designed an augmented operator for multiphase flow that injects U-Net style multi-resolution mixing into the spectral core. Taken together, all these developments ground Fourier and other types of neural solvers for filling problems by enabling operator learning surrogates on unstructured meshes that accelerate design exploration while maintaining accuracy across geometries.

1.3 Aim and contribution

Metal casting represents an important manufacturing method that can be characterized as a filling problem. Specifically, the hydrodynamics of mold filling decisively govern defect formation, microstructure evolution, and thus final cast part quality. CFD simulations are the industry standard approach for analyzing mold filling and therefore for guiding mold and gating system designs. Yet, these simulations are typically executed in iterative workflows, where each candidate layout demands a numerically intensive transient simulation. While running a single simulation is generally viable, the outcome design configuration is likely suboptimal. At the same time, the exploration of large design spaces that are often encountered in engineering problems is slow. Consequently, research about surrogate models that can approximate filling problems are becoming a relevant research avenue.

In the recent years, artificial intelligence has begun to reshape scientific computing by complementing first principles modeling with data driven surrogates. In scientific machine learning, operator learning frameworks such as DeepONet [40] and the FNO [43] approximate PDE solution operators directly in function space, enabling surrogate evaluations that are orders of magnitude faster than conventional numerical solvers while preserving accuracy on established benchmarks. Yet, filling problems in the manufacturing industry remain comparatively underexplored due to the difficulty of representing strongly coupled, geometry-dependent spatio-temporal dynamics under varying IC/BC, and the large, structured design space encountered in design optimization. To address this gap, this work develops an operator learning surrogate for a filling problem. Specifically, we study a simplified yet representative two-phase mold filling problem and train a neural solver to approximate the coupled Cahn–Hilliard–Navier–Stokes (CHNS) solution operator on unstructured meshes, with generalization across diverse gate geometries and placements. Our main contributions are:

- This work formulates 2D mold filling, a representative case of the broader family of filling problems, as an operator learning problem and propose a Fourier-Graph neural solver that maps problem definitions such as geometry, initial and boundary conditions, and process parameters to transient fields of velocity, pressure, and volume fraction.
- The resulting neural solver models generalize across parametric gating design spaces, amortizing learning over many PDE instances and delivering orders-of-magnitude speedups relative to CFD while retaining predictive accuracy for fluid-air interface and flow fields.
- This study analyses the model susceptibility to spatial and temporal subsampling and long-horizon error growth, and ablate architectural and training choices such as Fourier modes, latent resolution, and temporal rollout to identify stability and accuracy drivers.
- We show that aggregating point-wise PDE knowledge from multiple designs yields robust surrogates that can be conditioned on new gate configurations to produce fast, reliable flow predictions.

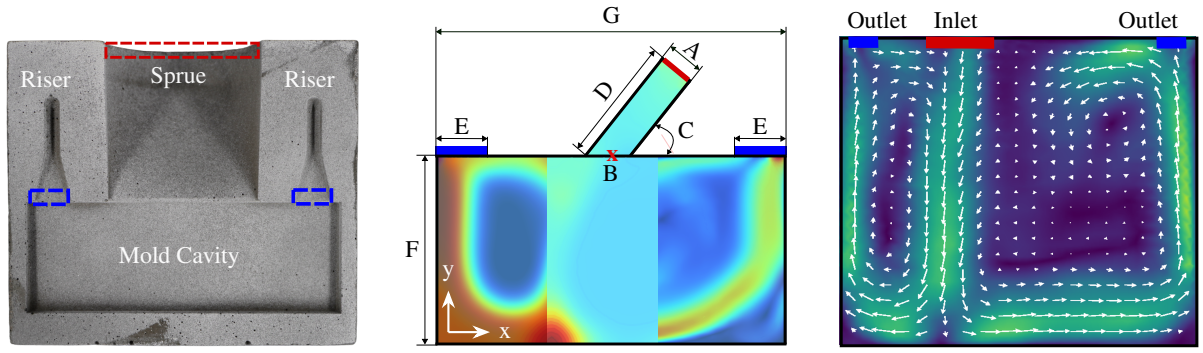
Paper organization: Sections 2.1 and 2.2 formalize the problem setup and governing equations, respectively. Section 2.3 present our CFD dataset generation strategy and Section 2.4 our proposed Fourier-Graph neural operator learning setup. Section 2.5 summarizes the ablation studies performed. Section 3.1 broadly investigates the general solver capabilities. Sections 3.2, 3.3, and 3.4 evaluate the performance of the proposed neural solver under spatial and temporal subsampling, as well as data efficiency, respectively. Finally, Section 4 concludes and outlines future research opportunities.

2 Methodology

The following methodology is adopted to learn a parametrized two-phase mold filling simulation using neural solvers. It begins by introducing the CHNS problem and a parametric mold cavity model, which, together with IC and BC data, define the mold filling design space. Then, we formalize the governing equations in strong and numerical form. Next, the generation of supervised datasets via CFD on the parametric cavity model is detailed and the neural solver theory and training recipe are presented. Finally, the metrics employed in the evaluation protocol and the ablation studies on spatial and temporal subsampling, and the availability of data are outlined.

2.1 Problem formulation and assumptions

Isothermal two-phase mold filling is modeled by the incompressible Navier-Stokes [48, 49] equations coupled with a diffuse-interface Cahn-Hilliard [50] formulation. The unknowns are the velocity field $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^2$, pressure $p(\mathbf{x}, t) \in \mathbb{R}$, and the volume fraction $\alpha(\mathbf{x}, t) \in [0, 1]$[1] for some time $t \in T$ with $t \geq 0$. Thermal and solidification effects are neglected to isolate the fluid-dynamic contribution to filling patterns.



(a) Experimental concrete mold **(b)** Mold cavity model **(c)** Exemplaric field prediction

Figure 1: Casting model. Figure 1a shows an experimental concrete mold with sprue, cavity, and two risers. Figure 1b depicts the parametric cavity model with filling colors representing numerical simulation outcomes. Figure 1c illustrates an exemplarily velocity field prediction on a rectangular domain affinely normalized to $[0, 1]^2$.

Let $\Omega \subset \mathbb{R}^2$ denote the mold model composed of the ingate pipe and the rectangular cavity. Its boundary $\partial\Omega$ is partitioned into a single inlet Γ_{in} at the top wall, two outlets Γ_{out}, and no-slip walls Γ_{w}. Figure 1 outlines this setup: Figure 1a shows an experimental concrete mold, taken from the study published by Link et al. [51], with sprue and two risers that inspired our model. Figure 1b highlights the parametric cavity model with inlet size A, horizontal position B, angle C, vertical offset D, outlet size E, height F, and width G. Finally, Figure 1c illustrates a predicted velocity field on the affinely normalized domain $[0, 1]^2$.

At $t = 0\,\text{s}$, a steady inflow with a diffuse interface consistent with the CHNS formulation is imposed on Γ_{in}. The inflow has a Dirichlet velocity of magnitude V and direction ζ. On the walls Γ_{w}, we impose no-slip condition. The outlets Γ_{out} are designed to suppress backflow and to compensate hydrostatic pressure. Given a geometry descriptor $\mathcal{G}$ defining Ω, process parameters θ, and inlet data (V, ζ), the task is twofold: (i) numerically solve the forward CHNS problem to obtain $(\mathbf{u}, p, \alpha)$ fields on $\Omega \times (0, T]$ and (ii) learn an operator $\mathcal{N}_\vartheta$ that approximates the solution map $\mathcal{S}$

$$\mathcal{S} : (\mathcal{G}, \theta, V, \zeta) \;\mapsto\; (\mathbf{u}, p, \alpha), \tag{1}$$

from supervised pairs generated by the numerical solver, yielding fast surrogate predictions $(\hat{\mathbf{u}}, \hat{p}, \hat{\alpha})$

$$(\hat{\mathbf{u}}, \hat{p}, \hat{\alpha}) = \mathcal{N}_\vartheta(\mathcal{G}, \theta, V, \alpha). \tag{2}$$

[1] Bold symbols denote vectors (e.g., $\mathbf{u}$), plain symbols denote scalars (e.g., p,ϕ)

2.2 Governing equations

On $\Omega \times (0, T]$, the governing incompressible CHNS [52] system for the velocity field $\mathbf{u} = (u, v)$, pressure p, and volume fraction α reads

$$\nabla \cdot \mathbf{u} = 0 \tag{3}$$

for the continuity equation and

$$\rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) = -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + F_x, \tag{4}$$

$$\rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) = -\frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + F_y + \rho g, \tag{5}$$

for the momentum conservation, with g acting in the $-y$ direction, (F_x, F_y) being the surface tension force components, and ρ the fluid density. The volume fractions of the two fluids are modeled with a phase field model based on the Cahn-Hilliard equations

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} = \frac{\gamma\lambda}{\varepsilon^2}\left(\frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2}\right) \tag{6}$$

$$\Psi = -\varepsilon^2\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right) + (\phi^2 - 1)\phi \tag{7}$$

with phase-field variable ϕ, chemical potential Ψ, mixing energy density λ, interface thickness ε, and mobility parameter $\gamma = \chi\varepsilon^2$ for mobility tuning parameter χ. This leads to the volume fractions of both fluids via

$$\alpha_2 = \begin{cases} 0, & \phi < -1, \\ \dfrac{1+\phi}{2}, & -1 \le \phi \le 1, \\ 1, & \phi > 1, \end{cases} \qquad \alpha_1 = 1 - \alpha_2. \tag{8, 9}$$

The surface tension contributions to the Navier–Stokes equations in each direction are

$$F_x = \frac{\lambda}{\varepsilon^2}\,\Psi\,\frac{\partial \phi}{\partial x}, \qquad F_y = \frac{\lambda}{\varepsilon^2}\,\Psi\,\frac{\partial \phi}{\partial y}. \tag{10, 11}$$

and the density and dynamic viscosity are interpolated via

$$\rho = \rho_1\alpha_1 + \rho_2\alpha_2, \qquad \mu = \mu_1\alpha_1 + \mu_2\alpha_2 \tag{12,13}$$

for the density ρ_1 and dynamic viscosity μ_1 of fluid one and the density ρ_2 and dynamic viscosity μ_2 of fluid two. On the inlet Γ_{in}, Figure 1b, the boundary conditions read:

$$u = V\cos(C), \qquad v = V\sin(C). \tag{14, 15}$$

On wetted walls Γ_{w}, the no-slip $u = v = 0$ conditions read

$$\frac{\gamma\lambda}{\varepsilon^2}\left(\cos(\beta_n)\frac{\partial \Psi}{\partial x} + \sin(\beta_n)\frac{\partial \Psi}{\partial y}\right) = 0, \tag{16}$$

$$\varepsilon^2\left(\cos(\beta_n)\frac{\partial \phi}{\partial x} + \sin(\beta_n)\frac{\partial \phi}{\partial y}\right) = \varepsilon^2\cos(\theta_w)\left(\left(\frac{\partial \phi}{\partial x}\right)^2 + \left(\frac{\partial \phi}{\partial y}\right)^2\right), \tag{17}$$

for the respective angle β_n of the normal vector of the wall, static contact angle at the wall θ_w, and chemical potential Ψ. At the outlets Γ_{out} the backflow is suppressed and on the whole domain the hydrostatic pressure is compensated. The initial phase field ϕ_0 for the part of the domain filled with the first fluid is set to

$$\phi_0 = -\tanh\left(\frac{D_{wi}}{\sqrt{2}\,\varepsilon}\right), \tag{18}$$

and for the part of the domain filled with the second fluid to

$$\phi_0 = \tanh\left(\frac{D_{wi}}{\sqrt{2}\,\varepsilon}\right), \tag{19}$$

for the distance to the initial interface D_{wi}. Unless otherwise stated, any variable dependent initialization or constant not explicitly specified adheres to the default COMSOL Multiphysics [53] settings used in our simulations.

2.3 Dataset generation

To probe the capabilities of neural solvers, we generate two synthetic datasets by varying the model parameters of the filling problem presented in Figure 1b and numerically solving the CHNS equations in COMSOL. These two datasets, DS_1 and DS_2, were conceptualized to represent increasing levels of learning complexity for neural solvers. In DS_1, the inlet horizontal position is fixed and only the inlet velocity magnitude and direction vary, representing a learning problem with localized momentum injection. In DS_2, the inlet horizontal position also varies, producing more complex flow regimes.

The parametric design space of the mold filling problem is summarized in the Table 1. The geometric parameter ranges and step counts follow Figure 1 and Table 1: $A \in [10, 25]\,\mathrm{mm}$, $B \in [10, 90]\,\mathrm{mm}$ (fixed in DS_1, variable in DS_2), $C \in [10, 90]^\circ$, $D = 30\,\mathrm{mm}$, $E = 10\,\mathrm{mm}$, $F = 50\,\mathrm{mm}$, $G = 100\,\mathrm{mm}$, and $V \in [0.1, 0.9]\,\%$.

In all simulation cases, a steady inflow on the inlet Γ_{in} with Dirichlet velocity of magnitude V and direction ζ is imposed at $t = 0\,\mathrm{s}$, the walls Γ_{w} are considered no-slip, and the outlets Γ_{out} suppress backflow and compensate hydrostatic effects. As initial condition, we consider the inlet pipe filled with water at 373 K, whereas the rectangular mold is filled with air at 373 K. As an outcome, each simulation produces time-resolved velocity $\hat{\mathbf{u}} = (\hat{u}, \hat{v})$, pressure $\hat{p}$, and volume fraction $\hat{\alpha}$ fields on an unstructured mesh, yielding sequences suitable for mesh-aware operator learning. The number of mesh elements stays below 3000 elements and depends on the BC and geometric characteristics. The total temporal integration spans $t \in [0, 5]\,\mathrm{s}$ with a fixed time step of $\Delta t = 0.01\,\mathrm{s}$. The resulting datasets comprise 939 simulations for the simplified DS_1 setting and 6320 for the complete DS_2 setting. Each simulation runs for about 2 min on a High Performance Computer (HPC) cluster while using two *Intel Xeon Gold 6326 Ice Lake CPUs* with 2×16 cores at 2.9 GHz.

Table 1: Mold filling design space. Parametric ranges employed for the generation of the DS_1 and DS_2 mold filling datasets. DS_2 extends DS_1 by additionally varying the inlet horizontal position, thereby allowing momentum injection at different locations of the upper rectangle wall.

Parameter	Unit	Value range	Steps DS_1	Steps DS_2
Inlet size (A)	mm	10 – 25	10	10
Inlet horizontal position (B)	mm	10 – 90	fixed	10
Inlet angle (C)	°	10 – 90	10	10
Inlet vertical offset from top wall (D)	mm	30	fixed	fixed
Inlet velocity (V)	%	0 – 100[2]	10	10

2.4 Proposed method

The main goal of the current study is to understand to what extent Fourier neural solvers can approximate parametric transient filling problems with varying IC and BC conditions. Therefore, we borrow and simplify a casting problem from an existing study, see Section 2.1, and seek to map parametric mold geometries, IC and BC conditions as well as process parameters to time-resolved velocity $(\hat{u}, \hat{v})$, pressure $\hat{p}$, and volume fraction $\hat{\alpha}$ fields. Towards this goal, we introduce a Fourier-Graph neural solver that is based on and extends the Geometry-Informed Neural Operator (GINO) [33]. In GINO, a Graph Neural Operator (GNO) encoder first aggregates local features from an unstructured input mesh and maps them to a regular latent grid. On this grid, a FNO stack captures long-range couplings while modulating Fourier features according to the inlet velocity via Adaptative Instance Normalization (AdaIN) [54]. Finally, a GNO decoder projects the learned latent predictions back onto an unstructured mesh that is not necessarily the same as the input. In summary, the GINO design combines the ability of GNO in handling unstructured meshes and learning local features with the efficiency of FNO for modeling global interactions. However, GINO presents the drawback of only modeling single field, steady-state problems, whereas the regarded casting problem is a transient, multi-field problem.

To address these limitations, we introduce a Fourier-Graph method that extends GINO in four different ways, see Figure 2. First, we modify the output of the GNO encoder to broadcast the 2D latent grid to 3D, thereby augmenting the latent space to handle transient features. Second, given the parametric nature of our mold filling problem, we set AdaIN to modulate the learned Fourier features based on the inlet setup I composed of inlet velocity V, size A, horizontal position B, and angle C. Third, we condition the input unstructured meshes on the initial field conditions and inlet mask to assist model learning. Finally, we employ an adapted causal rollout loss, inspired by Wang et al. [55], which reweighs temporal errors to prioritize earlier time steps during training, mirroring the forward-marching behavior of standard time-stepping solvers. With these extensions, our proposed Fourier-Graph neural operator is capable of predicting 3D multi-field transient mold filling simulations. In the following, Section 2.4.1 presents the Fourier neural solver theory,

[2]Velocity fractions that enforce Reynolds number < 2300 at the ingate pipe.

Section 2.4.2 formalizes our Fourier-Graph learning objective and optimization, and Section 2.4.3 the metrics we used for training and evaluation.

2.4.1 Fourier Neural Solver

The proposed Fourier-Graph approach is composed of a encoder–spectral–decoder architecture, as depicted in Figure 2. The GNO encoder follows the idea of Graph Neural Networks (GNNs) [56], where the neighborhood information of a vertice is aggregated. Let $\mathbf{x_i}$ denote a query node in the unstructured input mesh D_Ω connected to all mesh points within a ball $G_b \in B_{r_d}(\mathbf{x}) \subset D_\Omega$ with radius $r_d \in \mathbb{R}_+$ centered at $\mathbf{x_i}$. This construction realizes the integral operator

$$K(f_{man})(\mathbf{x}) \;=\; \int_{B_{r_d}(\mathbf{x})} \kappa_{\hat{\theta}}(\mathbf{x}, b)\, f_{man}(\mathbf{x})\, db\;, \quad \mathbf{x_i} \in D_\Omega. \tag{20}$$

for an input function f_{man} describing the geometry, inlet mask, and initial conditions, and a learnable neural network kernel κ with parameters $\hat{\theta}$. The GNO encoder produces mesh-aware features on a regular 2D latent grid, which are broadcasted to form a 3D latent representation that serves as input to the spectral core.

On the 3D latent grid, each Fourier layer $\mathcal{L}^l$ of the spectral core transforms the latent signal into the frequency space, applies a linear mapping to the retained low-frequency modes, and brings it back to the spatial domain via an inverse Fourier transform. In parallel, the 3D latent grid is processed through a pointwise linear transformation. Finally, the two data streams are summed and passed through a nonlinear activation. In compact form, for a layer ℓ, we therefore have

$$U^{\ell+1} = \sigma\big(W^\ell U^\ell + \mathcal{F}^{-1}\big(R^\ell \odot \mathcal{F}(U^\ell)\big)\big)\,, \tag{21}$$

where U denotes the latent signal (e.g. at the first layer $U = K$), $\mathcal{F}$ is the Fast Fourier Transform (FFT) on the 3D latent grid, R^ℓ are learnable spectral weights supported on a truncated set of modes, and W is a point-wise linear map. This truncated spectral parametrization is modulated through AdaIN based on the inlet setup I composed of inlet velocity V, size A, horizontal position B, and angle C, and captures the dominant global couplings of the incompressible flow and interface transport, while the linear maps preserve local information. At the end, the stack of FNO layers produces a fixed-horizon sequence on a regular latent grid, which is used by the GNO decoder to predict velocities $(\hat{u}, \hat{v})$, pressure $\hat{p}$, and volume fraction $\hat{\alpha}$ at each time step $t \in T$. Note that the output mesh does not necessarily have the same (x, y) coordinates as the input mesh.

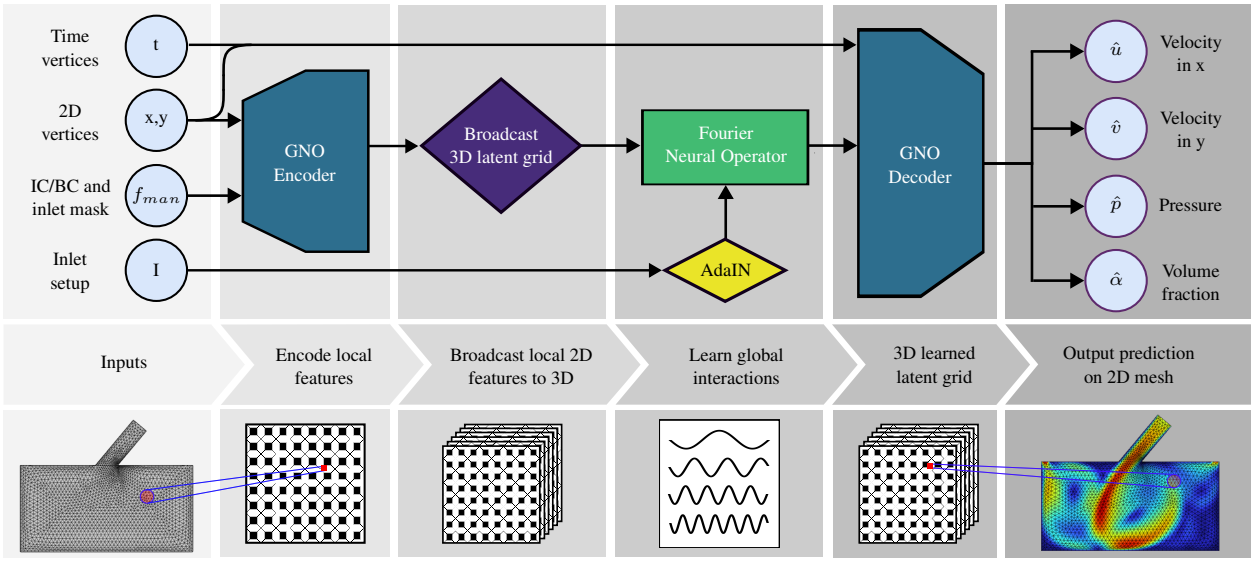


Figure 2: Fourier–Graph neural solver. The model receives as input the unstructured mold geometry mesh, the inlet mask, and the initial fields at $t_0 = 0$. A geometry-aware GNO encoder lifts the inputs to a regular 2D latent grid that is broadcasted to 3D. The Fourier spectral core FNO captures long-range couplings while modulating Fourier features through AdaIN based on the inlet setup I composed of inlet velocity V, size A, horizontal position B, and angle C. Finally, for each time step $t \in T$, a GNO decoder maps the 3D latent grid to $(\hat{\mathbf{u}}, \hat{p}, \hat{\alpha})$ fields at arbitrary (x, y) spatial locations within the mold domain.

2.4.2 Learning objective and optimization

Let the input bundle $a = (\text{geometry mask}, IC/BC, A, \ldots, V)$ be defined on Ω. We learn a discretization–invariant neural operator as a supervised, fixed–horizon operator regressor over the four (u, v, p, α) simulation field targets. For prediction steps $k = 1, \ldots, H$, target fields $q \in (u, v, p, \alpha)$, and prediction fields $\hat{q} = (\hat{u}, \hat{v}, \hat{p}, \hat{\alpha})$, the per–step, per–field relative L_2 error over all N vertices is given by

$$\ell_q^{(k)} = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{q}_i^{(k)} - q_i^{(k)}\right)^2}{\frac{1}{N}\sum_{i=1}^{N}\left(q_i^{(k)}\right)^2} \tag{22}$$

To stabilize multi–step forecasts during training we apply a causal rollout weighting [55] that increases the weight on a time step only after the residuals at earlier steps are reduced. In order to do that, we form the cumulative past loss for each field $c_q^{(k)} = \sum_{j<k} \ell_q^{(j)}$ and assign a single temporal weight $\gamma_t{}^{(k)}$ to step k by combining all the distinct field contributions

$$\gamma_t^{(k)} = \min_q \exp\big(-\tau\, c_q^{(k)}\big) = \exp\Big(-\tau \max_q \sum_{j<k} \ell_q^{(j)}\Big), \qquad \tau > 0. \tag{23}$$

where τ denotes the causality parameter that controls the steepness of the weights. Notice that we adopt a conservative temporal weight approach by selecting the minimum weight among all fields instead of a per-field minimum. Combining Equations (22) and (23) finally leads to our optimization loss L_{opt}

$$L_{\text{opt}} = \frac{1}{4H} \sum_{q\in\{u,v,p,\alpha\}} \sum_{k=1}^{H} \gamma^{(k)}\, \ell_q^{(k)} \tag{24}$$

which was employed in all experiments we performed.

During training, all models were optimized with the same training recipe. Specifically, we employed the AdamW [57] optimizer, a reduce-on-plateau learning rate schedule, and early stopping on a held-out validation split. Furthermore, we considered single-simulation mini-batches on DS_1 and DS_2 with a fixed, non-autoregressive forecast horizon of $H = 10$ steps. For training, a single *NVIDIA A100 GPU* with 40 GB VRAM was employed, and training convergence was typically achieved within 48 – $72\,\text{h}$ per model. Under the available computational budget, the best performing configuration, used as the default unless stated otherwise, employs a Fourier-Graph operator with $48 \times 48 \times 48$ modes in the Fourier core, a graph radius of $24\,\text{mm}$ for message passing, a latent-grid resolution of 80, an initial learning rate of 1×10^{-4}, and weight decay of 10^{-5}.

2.4.3 Evaluation metrics

During evaluation, we report the time-averaged relative L_2 percentage error *per field* and a *mean* across all fields. The relative L_2 percentage error r at step k for a target field q is given by

$$r_q^{(k)} = \frac{\|\hat{q}^{(k)} - q^{(k)}\|_2}{\|q^{(k)}\|_2} \tag{25}$$

The time-averaged relative error r_q for field q is

$$r_q = \frac{1}{H}\sum_{k=1}^{H} r_q^{(k)} \tag{26}$$

When the mean is reported, we consider the mean time relative average error across all fields:

$$r = \tfrac{1}{4} \sum_{q\in\{u,v,p,\alpha\}} r_q \tag{27}$$

2.5 Ablation studies

Neural operator models learn PDE solution operators that are mesh-agnostic, generalizing across distinct discretization strategies and resolutions [43, 58]. However, this characteristic has not been evaluated extensively for filling problems with extensive parametric design spaces. To characterize how the discretization and amount of data impact the neural solver performance, we therefore conducted controlled ablations over spatial and temporal subsampling, as well as training set size. Table 2 summarizes all the ablation studies we performed.

First, we investigate the impact of different spatial subsampling factors $s_s \in \{1, \dots, 5\}$ on DS_1 and DS_2 by uniformly sampling vertices during training. In this scenario, our motivation is to evaluate how coarsening the input training meshes impact the modeling of high frequency field data by reporting the model performance at full mesh resolutions. Second, we evaluate the model robustness against temporal subsampling factors $s_t \in \{1, \dots, 6\}$ on DS_1 and DS_2 by varying the time step size Δt between steps $k = 1, \dots, H$ used for training. In this case, we seek to evaluate whether changing the simulation time step impacts the model learning ability or not, i.e. evaluate the model ability to learn more abrupt field changes between each transient step prediction. Finally, we investigate the role of data availability s_d on DS_2 by training with $\{100, 90, 80, 70, 60, 50\}\,\%$ of the available data corpus. In this scenario, we progressively shrink the size of the training dataset by sampling from it and report the mean and standard deviation. Our motivation is to estimate the amount of data necessary to achieve certain performance levels with neural solvers. Together, these ablation studies map the Fourier-Graph vulnerability to temporal and spatial samplings and training data volume.

Table 2: Summary of experiments. We investigate the spatial subsampling of mesh vertices and the temporal subsampling of trajectories on both datasets (DS_1, DS_2). Data-efficiency is evaluated on DS_2 by training with 100 – $50\,\%$ of the available data.

No	Experimental variable	Unit	Variable range						Dataset
1	Spatial subsampling factor s_s	-	1	2	3	4	5	-	DS_1, DS_2
2	Temporal subsampling factor s_t	-	1	2	3	4	5	6	DS_1, DS_2
3	Amount of training data s_d	%	100	90	80	70	60	50	DS_2

3 Results and Discussion

In this section, we examine the performance and robustness of the proposed Fourier-Graph neural solver across different scenarios, see Table 2 and Section 3.1. Unless stated otherwise, all models were trained with the loss and training procedures highlighted in Section 2.4.2, and evaluated with the metrics outlined in Section 2.4.3. For clarity of presentation, the predicted fields are resampled and visualized on a uniform 1000×1000 grid.

3.1 General Exploration

We first evaluate the ability of the neural solver to predict transient flow fields in a representative mold filling scenario. Figure 3 presents the predicted velocity field $\hat{\mathbf{u}} = (\hat{u}, \hat{v})$ for a selected DS_2 sample, alongside the target CFD velocity field $\mathbf{u}$, and the associated pointwise Euclidean error $\|\hat{\mathbf{u}} - \mathbf{u}\|_2$. The velocity fields were produced with a model trained with temporal subsampling factor $s_t = 6$ that forecasts $0.6\,\mathrm{s}$ into the future. To assess the temporal consistency of the model, we display snapshots at prediction steps $k \in \{0, 1, 2, 3, 6, 9\}$ with each snapshot corresponding to a prediction time $t = k \times 0.06\,\mathrm{s}$. Qualitatively, the figure shows that the proposed Fourier-Graph neural solver reproduces the target flow patterns with fidelity, capturing the motion and spreading of the fluid jet within the cavity and the overall advection dynamics. More generally, the model generalizes across different ingate locations and process conditions present in the dataset, indicating that it has learned an abstract representation of the flow physics of different mold geometries rather than overfitting to a single sample instance. This ability to handle varying geometries and inputs is a hallmark of operator learning approaches [46]. Combined with a graph-based encoder, this yields mesh-invariant predictions consistent with recent advances in Fourier operators on arbitrary domains [33].

The largest discrepancies between prediction and simulation appear in regions of sharp gradients and complex small-scale structures, specially near the fluid-air interface, in shear layers, and in recirculation zones. In these localized areas, the neural solver tends to smooth out fine details, as seen by attenuated velocity peaks and blurred fluid-air interfaces in the error maps. We hypothesize two main contributing factors to that: First, the spectral truncation in the Fourier layers acts as a low-pass filter, limiting the highest Fourier modes, and therefore diminishing the model capacity to represent sharp, high-frequency features. This spectral bias of FNO-based models has been widely discussed in the literature and is considered a fundamental trade-off for efficiency in Fourier-based models [59, 60]. Second, because we train by minimizing a mean squared error loss, the learned solution in regions with inherent ambiguity, such as turbulent eddies or diffuse interfaces, approximates an average of possible outcomes. Such averaging leads to over-smoothed predictions that underestimate extreme values or rapid fluctuations. These effects are clearly visible in the velocity field, where the predicted flow $\hat{\mathbf{u}}$ lacks some of the sharpest velocity spikes present in the target velocity field $\mathbf{u}$, and very thin splashes or filamentary structures are slightly smeared out. We note however that this behavior is not unique to our model, and that even enhanced FNO architectures require special measures, e.g. attention mechanisms or learned deformations, to resolve boundary layers and other steep features [46, 60]. Despite these localized errors, the global flow organization is preserved across all reported time steps, with the predicted velocity vectors (quiver arrows in Figure 3) maintaining the correct directionality and relative magnitude, therefore indicating that the proposed neural solver captures the bulk transport of momentum through the mold filling domain.

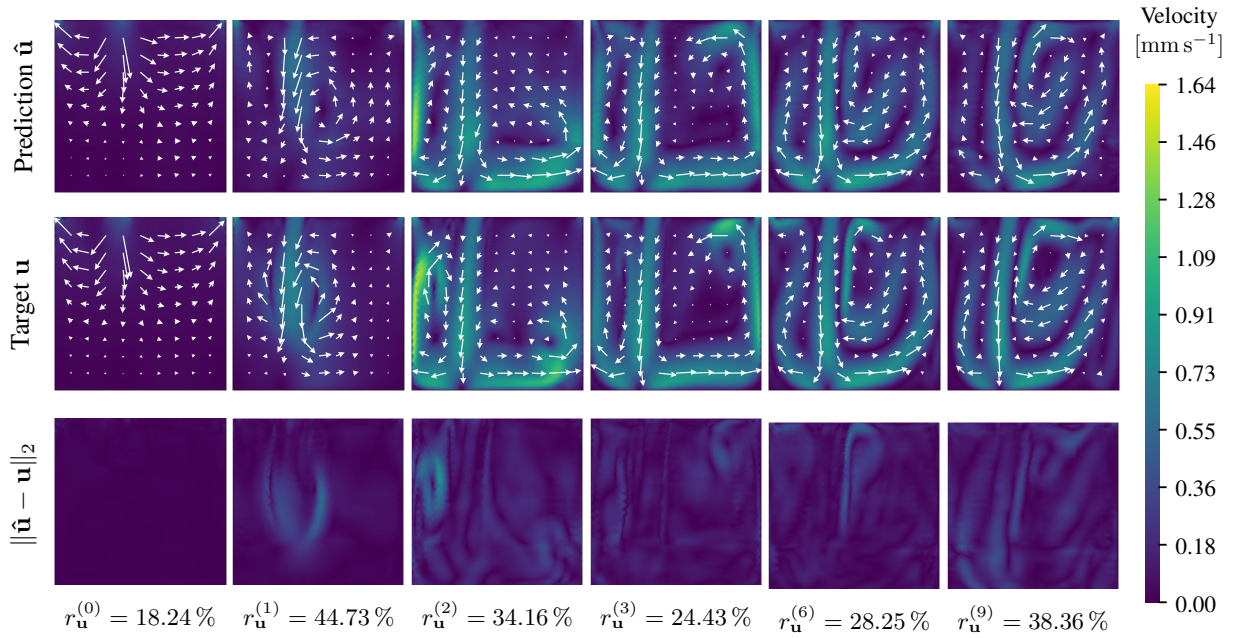
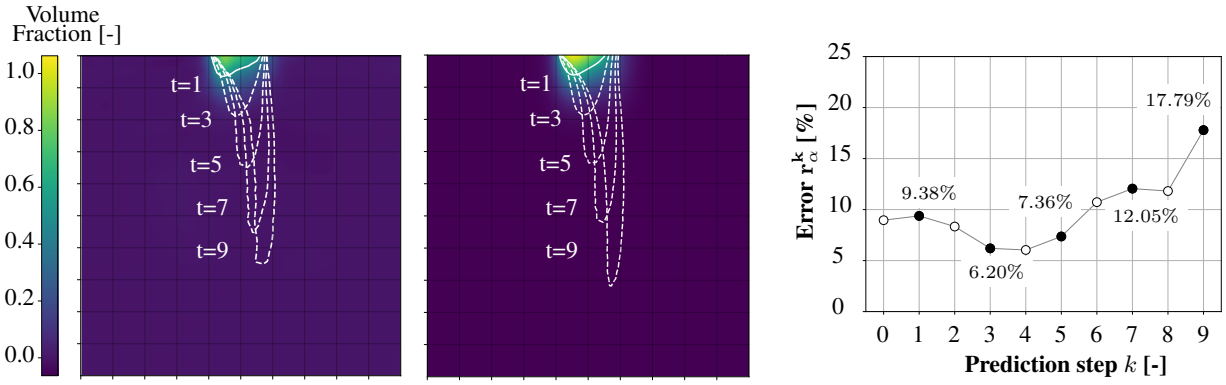


Figure 3: Velocity field prediction. Velocity field prediction $\hat{\mathbf{u}}$ (row 1), target CFD velocity field $\mathbf{u}$ (row 2), pointwise Euclidean error $\|\hat{\mathbf{u}} - \mathbf{u}\|_2$ (row 3), and relative L_2 error $r_{\mathbf{u}}^{(k)}$(row 4) at simulation steps $k \in \{0, 1, 2, 3, 6, 9\}$ with $k \times 0.06\,\mathrm{s}$ of a mold filling simulation. Quiver overlays indicate flow direction (array orientation) and velocity magnitude (arrow length).

The per step relative L_2 errors for the velocity field show the temporal accuracy trajectory of the model. In this particular sample, the $r_{\mathbf{u}}^{k}$ error rise sharply from roughly $18.24\,\%$ at step $k = 0$ to about $45\,\%$ at $k = 1$, then fall to about $24\,\%$ at $k = 3$, before increasing again toward about $38\,\%$ by step $k = 9$. This pattern aligns well with the velocity field predictions $\hat{\mathbf{u}}$ of the Figure 3. Namely, the early loss peak follows the initial jet formation and acceleration, when sharp gradients appear. Furthermore, the mid horizon predictions reflects a period when the bulk advection is well organized and easier to predict, while the later rise coincides with the increasing number of regions with interface and in recirculation zones, which increases local discrepancies. The non-monotonic trajectory of loss values confirms that the model corrects after the first transient yet gradually loses fine scale detail over a longer horizon due to the flow complexity, which is consistent with the smoothing effect previously mentioned.

Predicting the fluid-air interface through the volume fraction field α is critical for mold filling applications. With this in mind, Figure 4 illustrates the accuracy of the proposed Fourier-Graph model in predicting the fluid-air interface for a model with $H = 0.1\,\mathrm{s}$ total prediction horizon. Although our simulation data were generated with a diffuse interface method (Cahn-Hilliard phase-field), we extract an approximate interface location by plotting the $\alpha = 0.5$ iso-contour of the volume fraction field. Figure 4a shows the predicted volume fraction field $\hat{\alpha}$ at several time steps, while Figure 4b shows the corresponding target volume fraction field α. In order to allow for direct comparison, both prediction and target plots present an overlay of the fluid-air interface. At early times, e.g. at $t = 0.01\,\mathrm{s}$ ($k = 1$), the predicted and true interfaces coincide almost exactly, indicating that the neural solver initially advects the fluid-air interface front at the correct speed. Over longer rollouts, a gradual divergence becomes evident. At $t = 0.09\,\mathrm{s}$ ($k = 9$), the interface front slight shifts between the predicted and actual interface, particularly in regions of high curvature. Nonetheless, the overall shape and extent of the filling front remain well captured. We quantify the interface prediction error in Figure 4c, which plots the relative L_2 error $r_{\alpha}^{(k)}$ of the volume fraction field α as a function of the prediction step k. The error grows roughly monotonically with time horizon, reaching about $18\,\%$ relative discrepancy at the final step. Notably, the increase is gradual and there is no explosive error accumulation, suggesting the rollout remains stable over the tested $0.1\,\mathrm{s}$ horizon. Together, the velocity field predictions and fluid-air interface results indicate that the Fourier-Graph network aligns well with the parabolic nature of momentum transport and the hyperbolic advection of the phase front. In contrast, the pressure field, governed by an elliptic constraint, poses a greater challenge, which we analyze next.

Figure 5 highlights a sequence of pressure field predictions $\hat{p}$ from our proposed Fourier-Graph neural solver. We show snapshots of the predicted pressure field $\hat{p}$ alongside the target pressure field p, as well as the pointwise Euclidean $\|\hat{p} - p\|_2$ and relative L_2 percentage errors for steps $k \in (2, 4, 6, 7, 8, 9)$ with horizon $0.06\,\mathrm{s} \times k$. By overlaying pressure iso-contours, we observe that the neural solver can reproduce the broad pressure distribution in the cavity. However, the



(a) Prediction **(b)** Simulation Target **(c)** Relative L_2 $r_\alpha^{(k)}$ Error

Figure 4: Fluid-air interface tracking with iso-contour $\alpha = 0.5$ as a proxy. Figs. 4a and 4b show the predicted $\hat{\alpha}$ and simulation target volume fraction α fields, respectively. The background colormap and the continuous fluid-air interface overlay curves corresponds to step $k = 0$, while dashed curves denote the fluid-air interfaces at steps $k \in \{3, 5, 7, 9\}$ with $0.1\,\mathrm{s} \times k$ increments. Figure 4c shows the relative L_2 error $r_\alpha^{(k)}$ at each step k, with filled circle markers corresponding to the errors between fluid-air interfaces depicted in Figs. 4a and 4b.

error maps in Figure 5 reveals the presence of concentrated deficiencies at locations of steep pressure gradients. For instance, near the advancing fluid front and more prominently along dynamic impact zones where the flow impinges on the walls of the mold cavity. The largest pointwise pressure errors occur right after the liquid impingement, where the true solution exhibits sharp spikes that the model cannot fully capture. Quantitatively, the pointwise Euclidean error in pressure is in general higher than that of velocity or volume fraction at comparable times, confirming that the pressure field is the most challenging variable to be modeled. This finding is consistent with the reasoning that pressure solutions involve global elliptic constraints, and that small discrepancies can manifest if the global coupling of the model is not reasonably accurate. In the proposed Fourier-Graph architecture, the graph-based decoder does introduce some non-local message passing, which helps mitigate this issue by propagating information across the domain, but a residual gap still remains. Similar observations have been made in other operator-learning studies [47], where U-nets backbones helped to improve the predictions Fourier-based neural solvers.

The last row of in Figure 5 shows the relative L_2 error $r_p^{(k)}$ for different steps k. At step $k = 2$, the model shows a high error of $73\,\%$, which is followed by a rapid decay to values ranging $16 - 19\,\%$ at later steps. The initial high value matches the first impingement phase when steep pressure gradients appear and the model wrongly approximates fast transients. The subsequent plateau indicates that pressure errors stabilize rather than escalate over the tested 10-step horizon, which agrees with the absence of spurious oscillations in the maps. Together, these results reinforce that pressure is most vulnerable during the first transient steps and, once the flow organizes, the operator remains stable even though fine gradients remain less accurate.

Overall, the general exploration results establish that our Fourier-Graph neural solver can mimic the key behavior of a two-phase filling simulation by correctly advecting the fluid-air interface and reproducing the velocity and pressure fields. We also observed a systematic tendency to smooth sharp features, yet the results still justify their use as surrogates for design optimization, where consistency along the temporal prediction trajectory is critical. Moreover, the neural solver achieves these results at a fraction of the computational cost of transient CFD. In comparable settings, neural operators such as FNO have been reported to deliver $10^2 - 10^3$ speedups over traditional numerical solvers [43, 61], enabling rapid, design-in-the-loop evaluations that were previously impractical.

3.2 Spatial Subsampling

We next investigate the trade-off between simulation resolution and model accuracy by subsampling the training data spatially. Spatial subsampling here means we thin out the mesh vertices used during training, effectively training the network 38on a coarser version of the field data. This experiment reveals how sensitive the neural solver is to missing fine-scale spatial information, and by extension, how it might perform if deployed on lower-resolution simulations for efficiency. We train separate model instances on progressively coarser samplings of the mesh: factors $s_s = 1$ (no subsampling, using all mesh vertices), 2 (using 1 out of 2 vertices), 3, 4, and 5 (using only 1 in 5 vertices, i.e. $20\,\%$ of vertices). After training, we evaluate each model on the full-resolution meshes of the DS_1 and DS_2 datasets to understand how well the proposed neural solvers recover the high-resolution fields. Moreover, we also consider a DS_2^{60}

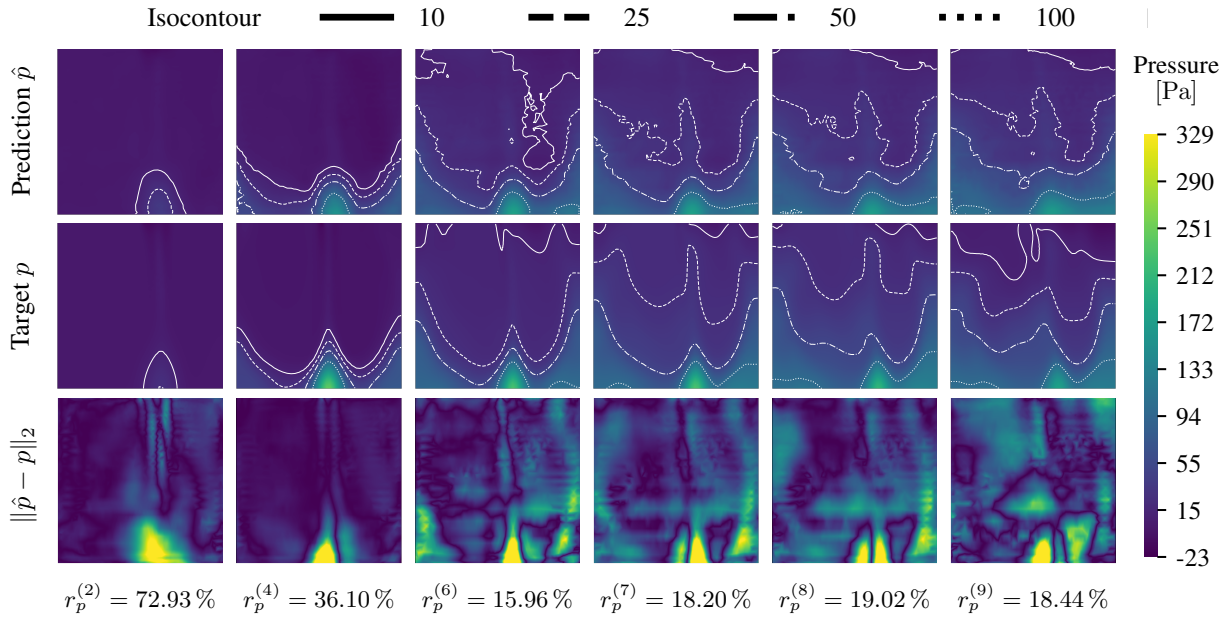


Figure 5: Pressure field prediction. Pressure field prediction $\hat{p}$ (row 1), target CFD pressure field p (row 2), pointwise Euclidean error $\|\hat{p} - p\|_2$ (row 3), and relative L_2 error $r_\alpha^{(k)}$ at simulation steps $k \in \{2, 4, 6, 7, 8, 9\}$ with $k \times 0.06\,\mathrm{s}$ of a mold filling simulation. Quiver overlays indicate flow direction (array orientation) and velocity magnitude (arrow length). Isocontour lines show distinct pressure levels.

dataset whose training data are temporally subsampled with factor $s_t = 6$ to examine whether a longer predicted time window interacts with spatial resolution changes.

Table 3 reports the relative L_2 percentage error of the model on each field (velocity components u, v, pressure p, and volume fraction α) as a function of the spatial sampling factor s_s. The overall trend is clear and monotonic, with even a modest reduction in spatial resolution during training significantly impacting accuracy and performance degradation growing steadily with larger s_s. For instance, on DS_2 the mean relative L_2 percentage error rises from about $4.5\,\%$ at full resolution ($s_s = 1$) to $6.6\,\%$ at $s_s = 5$. The simplified DS_1 shows a similar behavior (from ~ 3.97 to $7.46\,\%$ mean error) when going from no subsampling to $s_s = 5$. The DS_2^{60} case with fewer temporal frames but a longer horizon likewise sees its mean error increase from 6.28 to $7.53\,\%$ at $s_s = 5$. These numbers confirm the intuitive result that training the neural operator on coarser spatial data biases it towards low-frequency content and impairs its ability to reconstruct fine details. Among the individual field variables, pressure is consistently the most sensitive to spatial coarsening. For example, on DS_1 the pressure error roughly doubles (4.58 to $9.07\,\%$) when s_s increases from 1 to 5. By contrast, the volume fraction α is less affected by subsampling (error increasing from 3.40 to $6.66\,\%$ on DS_1), reflecting that the diffuse interface can still be roughly located even with fewer points. Lastly, the velocity components show only intermediate sensitivity to spatial subsampling. Overall, these field-wise trends mirror the qualitative behavior seen in Section 3.1, with features that require high spatial resolution such as pressure spikes and thin fluid-air interfaces being the first to suffer when the training data are under-resolved.

Table 3: Spatial subsampling performance for datasets DS_1, DS_2, and DS_2^{60}. Entries report the per-field and mean relative L_2 error (%) evaluated at full resolution meshes for models trained with uniform spatial sampling factors $s_s \in \{1, \ldots, 5\}$. DS_1 and DS_2 use no temporal subsampling. DS_2^{60} uses a temporal subsampling factor of $s_t = 6$.

Spatial subsampling factor s_s	**Relative L_2 Error [%]**														
	$\boldsymbol{u}$			$\boldsymbol{v}$			$\boldsymbol{p}$			$\boldsymbol{\alpha}$			**Mean**		
	$\mathbf{DS}_1$	$\mathbf{DS}_2$	$\mathbf{DS}_2^{60}$	$\mathbf{DS}_1$	$\mathbf{DS}_2$	$\mathbf{DS}_2^{60}$	$\mathbf{DS}_1$	$\mathbf{DS}_2$	$\mathbf{DS}_2^{60}$	$\mathbf{DS}_1$	$\mathbf{DS}_2$	$\mathbf{DS}_2^{60}$	$\mathbf{DS}_1$	$\mathbf{DS}_2$	$\mathbf{DS}_2^{60}$
1	3.96	4.49	6.48	3.92	4.52	6.74	4.58	5.15	6.28	3.40	3.90	5.60	**3.97**	**4.52**	**6.28**
2	5.96	5.62	7.40	6.11	5.75	7.84	7.10	6.80	7.92	5.89	5.37	7.15	**6.27**	**5.89**	**7.58**
3	6.56	6.37	7.43	6.54	6.54	7.83	8.45	7.51	7.77	6.30	6.19	7.09	**6.96**	**6.61**	**7.53**
4	6.51	6.13	7.41	6.67	6.23	7.78	8.00	8.19	7.80	6.41	5.97	7.09	**6.90**	**6.63**	**7.52**
5	7.08	6.26	7.47	7.02	6.42	7.86	9.07	7.52	7.67	6.66	6.10	7.12	**7.46**	**6.58**	**7.53**

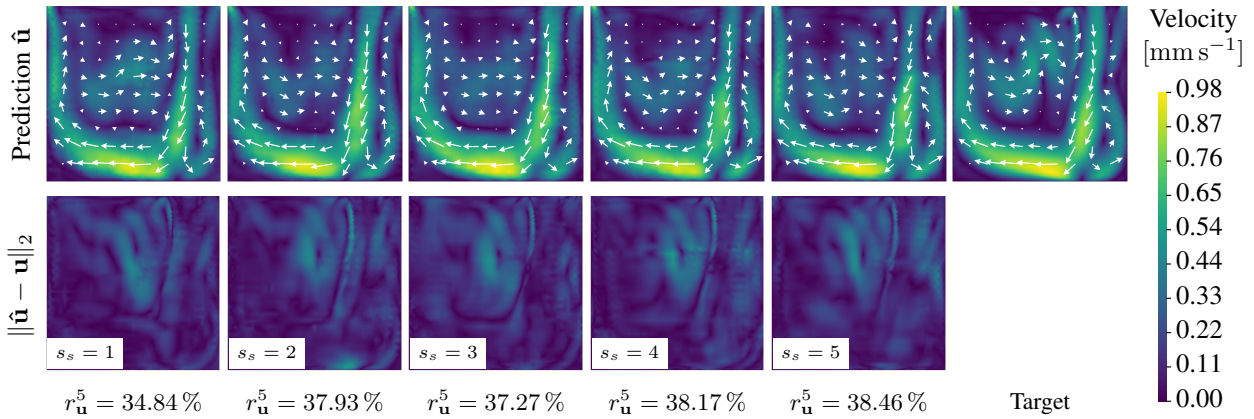
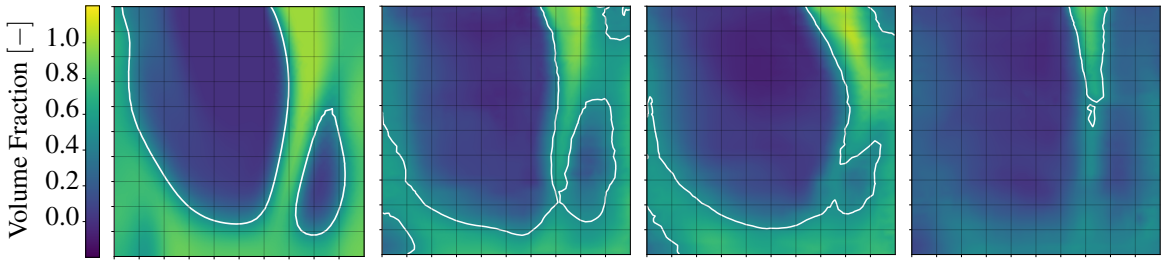


Figure 6: Impact of spatial subsampling on the velocity field prediction. Spatial subsampling predictions at factors $s_s \in 1, 2, 3, 4, 5$ (first five columns) versus the simulation target (last column). The bottom row shows the pointwise Euclidean error $\|\hat{u} - u\|_2$ for each s_s.

Figure 6 and Figure 7 visually illustrate the impact of spatial subsampling on the prediction of the model. In Figure 6, we compare the predicted velocity field against the target CFD velocity field at $t = 0.42\,\mathrm{s}$ for models trained with spatial resolution factor $s_s = 1$ (full resolution), 3, and 5. The top row shows the velocity field and the bottom row shows the corresponding pointwise Euclidean error $\|\hat{u} - u\|_2$. We see that the large-scale flow structures, such as the general direction of the fluid jet and the circulation pattern in the cavity, remain largely intact even for the heavily subsampled model ($s_s = 5$). This indicates that the network still captures the bulk momentum transport correctly. However, as s_s increases, fine details begin to blur or disappear. The error maps confirm a systematic growth of localized errors with higher subsampling, with errors concentrated along the interface and in regions of steep velocity gradients. This visual evidence supports the notion that coarse training effectively acts as a low-pass filter, removing access to high-wavenumber information. As a result, the neural solver cannot reconstruct information absent from training, yielding larger errors precisely where high-frequency details matter most.

The velocity pointwise Euclidean error $r^5_{\mathbf{u}}$ at step $k = 5$ increases with stronger spatial subsampling. At full resolution, the error is about $35\,\%$ and rises to roughly $38.5\,\%$ at the highest subsampling factor. This trend mirrors the mean errors in Table 3 and confirms that coarser training emphasizes low frequency content at the cost of local fidelity.

In Figure 7, we focus on the fluid-air interface predictions for $s_s \in 1, 3, 5$. Plotted are the volume fraction fields at $t = 0.42\,\mathrm{s}$ with the $\alpha = 0.5$ iso-contour delineating the fluid-air interface. At full resolution ($s_s = 1$, Figure 7b), the predicted interface aligns closely with the true interface, Figure 7a. At $s_s = 3$, the interface begins to thicken and drift slightly, and the neural solver starts to miss thin portions of the fluid-air interface. By $s_s = 5$, the deterioration is clear, with the predicted fluid-air interface being completely degenerate and the model unable to maintain a sharp front prediction. Nevertheless, even in the $s_s = 5$ case, the large-scale shape of the filling region is correct. All these observations reinforce that aliasing from spatial subsampling preferentially removes the high-frequency content needed for sharp gradients, e.g. pressure spikes and thin interfaces, while leaving the coarse flow pattern relatively untouched.



(a) Simulation target ($s_s = 1$) **(b)** $s_s = 1$ **(c)** $s_s = 3$ **(d)** $s_s = 5$

Figure 7: Effect of spatial subsampling s_s on the fluid-air interface fidelity. Predicted volume fraction fields $\hat{\alpha}$ with interface overlays for sampling factors $s_s \in \{1, 3, 5\}$ are compared with the simulation target in Figure 7a. As s_s increases, the fluid-air interface (white contour, $\alpha = 50\,\%$) prediction accuracy degrades with thickening and drifting, while the large-scale flow pattern remains similar. All predictions are shown at $t = 0.42\,\mathrm{s}$.

Interestingly, when comparing DS_2 and DS_2^{60} in Table 3, we find that extending the temporal prediction horizon does not dramatically worsen the spatial subsampling effect. The DS_2^{60} errors are uniformly higher than DS_2 at each s_s, but the increment in error from $s_s = 1$ to 5 is similar for DS_2 and DS_2^{60}. This suggests that within the examined range, spatial resolution is the limiting factor, and training on a coarser mesh degrades performance, regardless of whether we are predicting $0.1\,\mathrm{s}$ or $0.6\,\mathrm{s}$ ahead. In practice, this means our neural solver can tolerate reasonably long rollout horizons without catastrophic error growth, as long as the spatial information content is sufficient. It also implies that spatial resolution of training data is critical for capturing fine physics and one cannot simply compensate a coarse mesh with a shorter time horizon or vice versa. For future applications, this points to the value of adaptive meshing or smart sampling, where dedicating more spatial degrees of freedom in regions of interest (e.g. near the interface or ingates) has the potential to improve model accuracy without the cost of training with fine meshes. Indeed, other authors have proposed hybrid strategies that alternate high-fidelity simulation with a neural operator to leap in time [42], and architectures inspired by multi-grid refinement to better handle multi-scale features [62]. Such approaches could potentially alleviate the aliasing issues observed in this work.

3.3 Temporal Subsampling

We now examine the impact of temporal resolution on the accuracy of the neural solver. Similarly to the spatial subsampling strategy, we train models on datasets where intermediate time frames are omitted. A temporal subsampling factor s_t means the network sees only every s_t-th simulation frame during training while still forecasting a $k = 10$ step horizon. We experimented with $s_t = 1$ (no temporal thinning, i.e. $\Delta t = 0.01\,\mathrm{s}$ per step), up to $s_t = 6$ (using a $\Delta t = 0.06\,\mathrm{s}$ step, which for a 10-step rollout gives the $0.6\,\mathrm{s}$ horizon of DS_2^{60}). This tests the tolerance of the model to missing temporal information, where larger s_t forces it to bridge bigger gaps between input frames, which may challenge its ability to capture fast transient phenomena like pressure spikes or small splashes. On the other hand, training with a coarser time step can act as a form of regularization and reduce the total number of frames needed, which is advantageous for computational efficiency.

Table 4 summarizes the relative L_2 errors for velocity (u, v), pressure (p), and volume fraction (α) fields on the test set, for models trained with various s_t, using the datasets DS_1 and DS_2. We see a small, approximately linear degradation in accuracy as the temporal subsampling factor increases. On DS_1, the mean error grows from $3.97\,\%$ at $s_t = 1$ to $6.07\,\%$ at $s_t = 6$. On DS_2, the mean error goes from 4.52 to $6.41\,\%$ over the same range. This is a less expressive decline compared to the spatial subsampling case. In fact, the network seems relatively robust up to about $s_t = 3$. Beyond that, errors increase but remain within a narrow band for $s_t = \{4, 5, 6\}$. One interesting feature is a noticeable peak in pressure error at $s_t = 2$. A similar bump is seen for DS_2. This suggests a non-monotonic behavior where skipping every other frame ($s_t = 2$) misaligns some dynamics and causes a worst-case phase error, whereas skipping more frames leads the model to effectively learn on a consistently coarser timeline, to which it can adapt. In other words, there may be a resonance or aliasing effect at $s_t = 2$ where the temporal frequency of certain oscillations, e.g. pressure waves or vortex shedding, is poorly sampled, causing a more pronounced error. Once s_t is larger, those high-frequency modes are entirely filtered out, and the model learns the slower effective dynamics, resulting in slightly better error than the $s_t = 2$ case. This phenomenon underscores that the relationship between temporal resolution and accuracy is not strictly linear and that it depends on the interplay between the characteristic timescales of the simulation and the sampling interval.

Table 4: Temporal subsampling performance for datasets DS_1 and DS_2. Relative L_2 errors $(\%)$ are reported for each temporal sampling factor and predicted variable. A temporal subsampling factor of $s_t = 1$ means that we are considering the full temporal resolution available in the dataset, whereas a factor of $s_t = 6$ means considering only one every sixth frame.

Temporal subsampling factor s_t	**Relative L_2 Error [%]**									
	u		v		p		α		**Mean**	
	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2	DS_1	DS_2
1	3.96	4.49	3.92	4.52	4.58	5.15	3.40	3.90	**3.97**	**4.52**
2	5.56	5.79	5.45	5.81	8.72	8.22	4.97	5.32	**6.18**	**6.29**
3	5.91	6.07	5.94	6.14	6.87	6.94	5.18	5.59	**5.98**	**6.19**
4	6.15	6.32	6.15	6.49	6.54	6.74	5.21	5.72	**6.01**	**6.32**
5	6.18	6.47	6.31	6.63	6.16	6.54	5.42	5.76	**6.02**	**6.35**
6	6.29	6.60	6.58	6.88	6.07	6.46	5.34	5.71	**6.07**	**6.41**

Overall, the temporal subsampling experiments demonstrate that our neural solver maintains a reasonable accuracy even when trained on data that is temporally sparse. The velocity and interface variables are especially resilient with errors increasing only gradually with s_t. This indicates that the network can internally infer a continuous trajectory through time despite only seeing widely spaced simulation time steps. The pressure field, as expected, is more sensitive to temporal coarsening and displays errors rising more noticeably since rapid pressure transients may be missed at large Δt. Yet, even for pressure, the errors plateau for $s_t \geq 3$, suggesting the model does not completely fail but rather converges

to solving a slightly smoothed version of the physics. From a practical standpoint, this result is promising and implies one can trade some temporal resolution for speed or data volume reduction, without catastrophic loss of accuracy. Indeed, training with $s_t > 1$ effectively means needing fewer time steps from expensive CFD simulations, which can save considerable computational resources. In applications like casting where certain integrated outcomes, e.g. final fill time, overall flow pattern, are more important, a slightly time-thinned training regime could be an acceptable compromise. We caution, however, that if one needs to capture very fast transients, high temporal resolution remains important. In summary, within the $0.6\,\mathrm{s}$ prediction horizon studied, our neural solver exhibits robustness to missing frames.

3.4 Training Data

Finally, we analyze the data efficiency of the proposed Fourier-Graph neural solver by varying the size of the training dataset. Gathering extensive CFD simulation data can be costly, so it is important to know how performance degrades if we train with less data, and whether there are diminishing returns to adding more training samples. Starting from the full DS_2 dataset, we create reduced training sets containing 90, 80, 70, 60 and $50\,\%$ of the total number of samples. Each subset is chosen by uniform random sampling without replacement of the DS_2 simulations, and we train five separate model instances for each subset size using different random draws to account for any sample selection bias. The network architecture and training hyperparameters remain the same as in Section 2.4.2 for all these experiments. We evaluate each model on the same held-out test set and report the mean and standard deviation of the relative L_2 mean percentage errors over the five runs. Table 5 presents the results of this study.

Table 5: Data efficiency on DS_2: Relative L_2 error $(\%)$ on the held-out test set as a function of training-set size. Entries are mean $\pm$ standard deviation over five random seeds. Percentages indicate the fraction of DS_2 used for training (uniform subsampling without replacement). All runs use identical model architecture and training recipe.

Percentage from total data [%]	**Relative L_2 Error [%]**				
	u	v	p	α	**Mean**
100	4.49	4.52	5.15	3.90	$\mathbf{4.52}$
90	6.37 ± 0.06	6.61 ± 0.07	6.66 ± 0.06	5.57 ± 0.14	$\mathbf{6.30 \pm 0.04}$
80	6.44 ± 0.05	6.66 ± 0.05	6.59 ± 0.17	5.67 ± 0.08	$\mathbf{6.34 \pm 0.05}$
70	6.50 ± 0.03	6.75 ± 0.06	6.75 ± 0.11	5.70 ± 0.04	$\mathbf{6.43 \pm 0.03}$
60	6.51 ± 0.03	6.77 ± 0.06	6.75 ± 0.16	5.77 ± 0.06	$\mathbf{6.45 \pm 0.05}$
5	6.67 ± 0.08	6.93 ± 0.10	6.67 ± 0.06	5.96 ± 0.12	$\mathbf{6.56 \pm 0.05}$

When considering $100\,\%$ of the DS_2 dataset, our proposed model achieves a mean error of $4.52\,\%$ on the test set. Reduction to $90\,\%$ of the data leads to a mean error of about $6.30 \pm 0.04\,\%$. At $80\,\%$ data, $6.34 \pm 0.05\,\%$. This trend continues until $50\,\%$, where the model accuracy achieves $6.56 \pm 0.05\,\%$. Several observations can be made about these outcomes. First, the overall performance degrades continuously and smoothly as the training set is reduced below $90\,\%$. The mean error increases roughly linear with decreasing data, and even with only half the data the error is still within $\approx 22\,\%$ of the full-data case. This smooth trend suggests that our model has not yet hit a data saturation point in the regime tested, and that it can still learn somewhat from each additional sample, but returns diminish as more data is added. In fact, the improvements beyond about $70 - 80\,\%$ of the data are very minor (the difference between $80\,\%$ and $100\,\%$ is only 0.18 percentage points in mean error). This implies diminishing returns, and that beyond a certain dataset size, additional simulations contribute little new information, and the generalization of the model error flattens out. A related observation is the low variance across the five random subsets at each level, where the standard deviation of error is on the order of $0.05\,\%$ or less in most cases, meaning the performance of the model is quite reproducible and not overly dependent on exactly which samples are picked. Together, these points indicate that the training process is stable and that the data samples in DS_2 are not highly redundant but also not vastly unique. In practical terms, this suggests that careful curation of the training set may be more valuable than throwing vast amounts of training data. For example, ensuring that the most physically distinct scenarios, e.g. different ingate positions, flow rates, etc., are included might achieve near-optimal accuracy without needing every possible simulation. Our findings support the view that targeted data selection or augmentation could outperform brute-force data generation in efficiency. Similar conclusions have been reached by other researchers. Wen et al. [47] report that their U-FNO-based surrogate for CO_2 injection needed only one-third the data of a standard CNN to reach the same accuracy, and other work on multi-fidelity training has shown one can combine a few high-resolution simulations with many low-resolution ones to train effective models at reduced cost [63]. In our case, using $50 - 70\,\%$ of the dataset with smart sampling might be sufficient to achieve within ≈ 0.1 of the minimum attainable error. Beyond about $75\,\%$ of the data, the effort spent generating more simulations might be better invested in diversifying conditions or improving the model rather than increasing the amount of training data.

In summary, the data-reduction study underscores that the proposed Fourier-Graph neural solver is data-efficient. Not only it does not require an extremely large simulation dataset to learn the underlying physics, but it also generalizes well even when trained on a subset of the full dataset. This is encouraging for real-world deployment, since obtaining simulation or experimental data is often a limiting factor. The small error increment under data removal suggests that the operator-learning approach is effectively extracting the key patterns from the flow physics. Once these are learned, additional examples yield diminishing returns. This aligns with the view that neural operators learn function-to-function mappings that interpolate smoothly between training scenarios. We emphasize that this does not imply that additional data are unnecessary. If parts of the parameter space are unsampled, the model will fail there. Rather, our results suggest that once the training distribution adequately covers the space, returns diminish. Future work could explore active learning strategies, where the model itself identifies which new simulations would most reduce its uncertainty, thereby optimizing data collection.

4 Conclusion

This work presents a Fourier-Graph neural solver for modeling 2D two-phase mold filling. The proposed model maps geometry, IC/BC data, and inlet settings to transient (u, v, p, α) fields on unstructured meshes. Results show that the surrogate model is able to reproduce the large scale advection and the advance of the fluid-air interface across unseen gating locations and inlet parameters while maintaining a stable rollout over the studied horizons. The aggregate error remains within a few percent across velocity, pressure, and volume fraction, and qualitative comparisons indicate that errors concentrate near steep gradients and high-curvature portions of the interface. Overall, velocity and volume fraction predictions are robust, whereas pressure is more sensitive.

The ablations quantify how training resolution and data volume shape performance. Spatial subsampling of the training mesh causes a monotonic increase in error, with performance degradation concentrating around sharp, gradient-rich fronts. Temporal subsampling yields gentler deterioration and the model remains tolerant to missing frames within the studied horizon. Finally, data reduction produces a smooth and moderate error increase with diminishing gains beyond roughly $75\,\%$ of the dataset. These findings offer practical guidance for dataset design, with spatial resolution being prioritized when accuracy is more important. Regarding dataset size and diversity, our investigations suggest that additional data should be curated to expand diversity rather than at quantity once the core regimes are covered. Taken together, these results support neural operators as fast surrogates for design-in-the-loop studies of gating systems.

The present study also reveals clear limitations, with the pressure field remaining the most challenging quantity for modeling. In general, the proposed neural solver delivers under resolved flows around localized gradients and along the fluid-air interface. In future works, we plan to improve pressure fidelity and physics consistency across steps. Possible research avenues include complementing the optimization loss with relative L_2 that punishes interface contour errors, divergence statistics, and global mass errors. We also plan to test longer horizons and spatial and temporal sampling strategies to study stability at extended times. Another promising research direction is the exploration of multi fidelity data schedules that combine a small set of high resolution cases with many coarse cases to improve sample efficiency and flow accuracy. Finally, expansions to 3D and thermal-solidification coupling are natural and necessary modelings to realistically surrogate casting physics.

Acknowledgement

The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). The hardware is funded by the German Research Foundation (DFG).

Author contributions statement

Edgard M. Minete: Software; Conceptualization; Methodology; Validation; Formal analysis; Investigation; Data Curation; Writing – Original Draft; Visualization. **Mathis Immertreu**: Software; Validation; Investigation. **Fabian Teichmann**: Conceptualization; Methodology; Writing – Original Draft; Writing – Review & Editing; Supervision; Project administration; Funding acquisition. **Sebastian Müller**: Methodology; Resources; Writing – Original Draft; Writing – Review & Editing; Supervision; Funding acquisition.

References

[1] Miljana Radivojević, Thilo Rehren, Ernst Pernicka, Dušan Šljivar, Michael Brauns, and Dušan Borić. On the origins of extractive metallurgy: new evidence from Europe. *Journal of Archaeological Science*, 37(11):2775–2787,

2010. ISSN 0305-4403. doi:https://doi.org/10.1016/j.jas.2010.06.012. URL https://www.sciencedirect.com/science/article/pii/S0305440310001986.

[2] Alireza Nourian-Avval and Ali Fatemi. Characterization and analysis of porosities in high pressure die cast aluminum by using metallography, X-ray radiography, and micro-computed tomography. *Materials*, 13(14):3068, jul 2020. doi:10.3390/ma13143068.

[3] Jan Majernik, Stefan Gaspar, Jan Kmec, Monika Karkova, and Jozef Mascenik. Possibility of utilization of gate geometry to modify the mechanical and structural properties of castings on the Al-Si basis. *Materials*, 13(16), 2020. ISSN 1996-1944. doi:10.3390/ma13163539. URL https://www.mdpi.com/1996-1944/13/16/3539.

[4] Dirk Lehmhus. Advances in metal casting technology: A review of state of the art, challenges and trends—Part I: Changing markets, changing products. *Metals*, 12(11), 2022. ISSN 2075-4701. doi:10.3390/met12111959. URL https://www.mdpi.com/2075-4701/12/11/1959.

[5] Marek Brůna, Marek Galcik, Richard Pastircak, and Elena Kantorikova. Effect of gating system design on the quality of aluminum alloy castings. *Metals*, 14(3), 2024. ISSN 2075-4701. doi:10.3390/met14030312. URL https://www.mdpi.com/2075-4701/14/3/312.

[6] Mohamed Ramadan. Influence of gating design on microstructure and fluidity of thin sections AA320.0 cast hypo-eutectic Al-Si alloy. *AIP Conference Proceedings*, 1966(1):020020, 05 2018. ISSN 0094-243X. doi:10.1063/1.5038699. URL https://doi.org/10.1063/1.5038699.

[7] Dayalan Gunasegaram, M Givord, R.G. O'Donnell, and B R Finnin. Improvements engineered in UTS and elongation of aluminum alloy high pressure die castings through the alteration of runner geometry and plunger velocity. *Materials Science & Engineering A: Structural Materials: Properties, Microstructure and Processing*, 559:276–286, jan 2013. doi:10.1016/j.msea.2012.08.098. URL https://doi.org/10.1016/j.msea.2012.08.098.

[8] Muhammad Huzaifa Raza, Ahmad Wasim, Muhammad Sajid, and Salman Hussain. Investigating the effects of gating design on mechanical properties of aluminum alloy in sand casting process. *Journal of King Saud University - Engineering Sciences*, 33(3):201–212, 2021. ISSN 1018-3639. doi:https://doi.org/10.1016/j.jksues.2020.03.004. URL https://www.sciencedirect.com/science/article/pii/S1018363919305914.

[9] Ewan Lordan, Jaime Lazaro-Nebreda, Yijie Zhang, Kun Dou, Paul Blake, and Zhongyun Fan. On the relationship between internal porosity and the tensile ductility of aluminium alloy die-castings. *Materials Science and Engineering: A*, 778:139107, 2020. ISSN 0921-5093. doi:https://doi.org/10.1016/j.msea.2020.139107. URL https://www.sciencedirect.com/science/article/pii/S0921509320301957.

[10] Ho jung Kang, Ho sung Jang, Seong-Hyo Oh, Pil hwan Yoon, Gyu heun Lee, Sun mi Shin, Jin young Park, and Yoon-Suk Choi. Effects of gate system design on pore defects and mechanical properties of pore-free die-cast Al-Si-Cu alloy. *Materials Today Communications*, 31:103673, 2022. ISSN 2352-4928. doi:https://doi.org/10.1016/j.mtcomm.2022.103673. URL https://www.sciencedirect.com/science/article/pii/S2352492822005360.

[11] H Mayer, M Papakyriacou, B Zettl, and S.E Stanzl-Tschegg. Influence of porosity on the fatigue limit of die cast magnesium and aluminium alloys. *International Journal of Fatigue*, 25(3):245–256, 2003. ISSN 0142-1123. doi:https://doi.org/10.1016/S0142-1123(02)00054-3. URL https://www.sciencedirect.com/science/article/pii/S0142112302000543.

[12] MagmaSoft. MagmaSoft – embedded software development services. https://www.magmasoft.com, 1988. Accessed: March 24, 2025.

[13] ESI Group. ProCast. https://www.esi-group.com/products/procast, 1990. Accessed: March 24, 2025.

[14] Flow Science. FLOW-3D CAST. https://www.flow3d.com/products/flow-3d-cast/, 2011. Accessed: March 24, 2025.

[15] Hyuk-Jae Kwon and Hong-Kyu Kwon. Computer aided engineering (CAE) simulation for the design optimization of gate system on high pressure die casting (HPDC) process. *Robotics and Computer-Integrated Manufacturing*, 55:147–153, February 2019. ISSN 0736-5845. doi:10.1016/j.rcim.2018.01.003. URL http://dx.doi.org/10.1016/j.rcim.2018.01.003.

[16] Xu Zhao, Ping Wang, Tao Li, Bo-yu Zhang, Peng Wang, Guan-zhou Wang, and Shi-qi Lu. Gating system optimization of high pressure die casting thin-wall AlSi10MnMg longitudinal loadbearing beam based on numerical simulation. *China Foundry*, 15(6):436–442, November 2018. ISSN 2365-9459. doi:10.1007/s41230-018-8052-z. URL http://dx.doi.org/10.1007/s41230-018-8052-z.

[17] Marek Brůna, Iveta Vasková, and Marek Galčík. Numerical simulation and experimental validation of melt flow in the naturally pressurized gating system. *Processes*, 9(11):1931, October 2021. ISSN 2227-9717. doi:10.3390/pr9111931. URL http://dx.doi.org/10.3390/pr9111931.

[18] Sebastian Kohlstädt, Michael Vynnycky, Stephan Goeke, and Andreas Gebauer-Teichmann. On determining the critical velocity in the shot sleeve of a high-pressure die casting machine using open source CFD. *Fluids*, 6 (11):386, October 2021. ISSN 2311-5521. doi:10.3390/fluids6110386. URL http://dx.doi.org/10.3390/fluids6110386.

[19] Daiki Fuwa, Takuya Sakuragi, Mai Mizubayashi, Masayuki Kobayashi, and Tetsuya Katsumi. Prediction of laminations in zinc alloy die-casting by gas-liquid two-phase flow simulation. *Materials Transactions*, 60(5): 793–801, May 2019. ISSN 1347-5320. doi:10.2320/matertrans.m2018395. URL http://dx.doi.org/10.2320/matertrans.M2018395.

[20] Krister Svanberg. The method of moving asymptotes—a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373, 1987. doi:10.1002/nme.1620240207.

[21] Krister Svanberg. A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization*, 12(2):555–573, 2002. doi:10.1137/S1052623499362822.

[22] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002. doi:10.1109/4235.996017.

[23] Shantanu Shahane, Narayana Aluru, Placid Ferreira, Shiv G. Kapoor, and Surya Pratap Vanka. Optimization of solidification in die casting using numerical simulations and machine learning. *Journal of Manufacturing Processes*, 51:130–141, March 2020. ISSN 1526-6125. doi:10.1016/j.jmapro.2020.01.016. URL http://dx.doi.org/10.1016/j.jmapro.2020.01.016.

[24] Michail Papanikolaou, Emanuele Pagone, Konstantinos Georgarakis, Keith Rogers, Mark Jolly, and Konstantinos Salonitis. Design optimisation of the feeding system of a novel counter-gravity casting process. *Metals*, 8(10):817, October 2018. ISSN 2075-4701. doi:10.3390/met8100817. URL http://dx.doi.org/10.3390/met8100817.

[25] Microsoft Research. Plenary: The fifth paradigm of scientific discovery. https://www.microsoft.com/en-us/research/video/plenary-the-fifth-paradigm-of-scientific-discovery/, 2022. Video presented at the Research Summit 2022 on October 17, 2022. Accessed: March 24, 2025.

[26] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[27] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *CoRR*, abs/1611.08097, 2016. URL http://arxiv.org/abs/1611.08097.

[28] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[29] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[30] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[31] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. *arXiv preprint arXiv:1801.10130*, 2018.

[32] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015.

[33] Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3D PDEs, 2023. URL https://arxiv.org/abs/2309.00583.

[34] Salah A Faroughi, Nikhil Pawar, Celio Fernandes, Maziar Raissi, Subasish Das, Nima K Kalantari, and Seyed Kourosh Mahjour. Physics-guided, physics-informed, and physics-encoded neural networks in scientific computing. *arXiv preprint arXiv:2211.07377*, 2022.

[35] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[36] Mohammadamin Mahmoudabadbozchelou and Safa Jamali. Rheology-informed neural networks (RhINNs) for forward and inverse metamodelling of complex fluids. *Scientific reports*, 11(1):12015, 2021.

[37] Dimitrios Katsikis, Aliki D Muradova, and Georgios E Stavroulakis. A gentle introduction to physics-informed neural networks, with applications in static rod and beam problems. *Journal of Advances in Applied & Computational Mathematics*, 9:103–128, 2022.

[38] Jochen Stiasny, George S Misyris, and Spyros Chatzivasileiadis. Physics-informed neural networks for non-linear system identification for power system dynamics. In *2021 IEEE Madrid PowerTech*, pages 1–6. IEEE, 2021.

[39] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[40] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[41] Chensen Lin, Zhen Li, Lu Lu, Shengze Cai, Martin Maxey, and George Em Karniadakis. Operator learning for predicting multiscale bubble growth dynamics. *The Journal of Chemical Physics*, 154(10), 2021.

[42] Vivek Oommen, Khemraj Shukla, Somdatta Goswami, Rémi Dingreville, and George Em Karniadakis. Learning two-phase microstructure evolution using neural operators and autoencoder architectures. *npj Computational Materials*, 8(1):190, 2022.

[43] 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.

[44] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 1(3):1–27, 2024.

[45] Shaoxiang Qin, Fuyuan Lyu, Wenhui Peng, Dingyang Geng, Ju Wang, Naiping Gao, Xue Liu, and Liangzhu Leon Wang. Toward a better understanding of Fourier neural operators: Analysis and improvement from a spectral perspective. *arXiv preprint arXiv:2404.07200*, 2024.

[46] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.

[47] Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-FNO — An enhanced Fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

[48] Claude-Louis Marie Henri Navier. Mémoire sur les lois du mouvement des fluides. *Mémoires de l'Académie Royale des Sciences de l'Institut de France*, 6:389–440, 1823. URL https://fr.wikisource.org/wiki/M%C3%A9moire_sur_les_lois_du_mouvement_des_fluides/Texte_entier. Read 18 March 1822; volume issued 1827.

[49] George Gabriel 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. URL https://pages.mtu.edu/~fmorriso/cm310/StokesLaw1845.pdf.

[50] John W. Cahn and John E. Hilliard. Free energy of a nonuniform system. I. Interfacial free energy. *The Journal of Chemical Physics*, 28(2):258–267, 1958. doi:10.1063/1.1744102. URL https://pub.

[51] Janna Link, Fabian Teichmann, Alexander Wetzel, Sebastian Müller, and Bernhard Middendorf. An initial study of ultra high performance concrete as reusable mold material for aluminum casting. *Materials*, 18(1):153, January 2025. ISSN 1996-1944. doi:10.3390/ma18010153. URL http://dx.doi.org/10.3390/ma18010153.

[52] Morton E. Gurtin, Debra Polignone, and Jorge Viñals. Two-phase binary fluids and immiscible fluids described by an order parameter. *Mathematical Models and Methods in Applied Sciences*, 6(6):815–831, 1996. doi:10.1142/S0218202596000341. URL https://doi.org/10.1142/S0218202596000341.

[53] COMSOL. COMSOL Multiphysics® Documentation, 2024. https://www.comsol.com.

[54] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017. URL https://arxiv.org/abs/1703.06868.

[55] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116813, 2024.

[56] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005. doi:10.1109/IJCNN.2005.1555942.

[57] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. arXiv:1711.05101.

[58] Nikola B. Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *CoRR*, abs/2108.08481, 2021. URL https://arxiv.org/abs/2108.08481.

[59] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *CoRR*, abs/2010.08895, 2020. URL https://arxiv.org/abs/2010.08895.

[60] Lele Li, Weihao Zhang, Ya Li, Chiju Jiang, and Yufan Wang. An attention-enhanced Fourier neural operator model for predicting flow fields in turbomachinery cascades. *Physics of Fluids*, 37(3):036121, 03 2025. ISSN 1070-6631. doi:10.1063/5.0254681. URL https://doi.org/10.1063/5.0254681.

[61] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks, 2021. URL https://arxiv.org/abs/2010.03409.

[62] Quang Tuyen Le and Chinchun Ooi. Surrogate modeling of fluid dynamics with a multigrid inspired neural network architecture. *Machine Learning with Applications*, 6:100176, 2021. ISSN 2666-8270. doi:https://doi.org/10.1016/j.mlwa.2021.100176. URL https://www.sciencedirect.com/science/article/pii/S2666827021000888.

[63] Jia-Wei Cui, Wen-Yue Sun, Hoonyoung Jeong, Jun-Rong Liu, and Wen-Xin Zhou. Efficient deep-learning-based surrogate model for reservoir production optimization using transfer learning and multi-fidelity data. *Petroleum Science*, 22(4):1736–1756, 2025. ISSN 1995-8226. doi:https://doi.org/10.1016/j.petsci.2025.02.014. URL https://www.sciencedirect.com/science/article/pii/S1995822625000482.