Author version

Published as:

Sperry, M., & Hwang, J. T. (2026). Multifidelity surrogate modeling for 3D aerodynamic flow field prediction using graph neural networks. In AIAA AVIATION 2026 Forum (Paper No. AIAA 2026-4802). https://doi.org/10.2514/6.2026-4802

BibTeX for this reference:

https://lsdolab.github.io/lsdo-publications/publication/sperry2026multifidelity/

```
@inproceedings{sperry2026multifidelity,
  author = {Mark Sperry and John T. Hwang},
  title = {Multifidelity Surrogate Modeling for 3D Aerodynamic Flow Field Prediction
    Using Graph Neural Networks},
  booktitle = {AIAA AVIATION 2026 Forum},
  year = {2026},
  doi = {10.2514/6.2026-4802},
  url = {https://doi.org/10.2514/6.2026-4802}
}
```

Multifidelity Surrogate Modeling for 3D Aerodynamic Flow Field Prediction Using Graph Neural Networks

Mark Sperry[1] and John T. Hwang[2]
University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093

A major bottleneck in adjoint-based multidisciplinary design optimization (MDO) problems involving aerodynamics is the large computational cost associated with high-fidelity computational fluid dynamics (CFD) models, which can make the MDO problem take days to solve even with dozens of processors. Surrogate models offer a practical way to reduce this cost, but many existing approaches focus on predicting integrated quantities such as lift and drag rather than full aerodynamic fields. Recent advances in machine learning have made full 3D flow-field prediction increasingly available, but exploration of hybrid models that directly incorporate low-fidelity solvers is limited. In addition, relatively few studies have demonstrated these models within differentiable, gradient-based optimization workflows. In this work, we use graph neural networks to build a multifidelity model to predict the 3D flow field around blended-wing-body (BWB) aircraft configurations. The model, trained on 100 CFD training cases, uses a differentiable panel method solution as a low-fidelity input and predicts the high-fidelity RANS flow field in the fluid domain. We embed the model in a fully differentiable workflow, enabling gradient-based optimization. Compared with a purely data-driven model, the multifidelity model improves volume field prediction accuracy across all reported variables and reduces lift and drag errors to 10.29% and 5.11% on the test dataset. In addition, the resulting surrogate model evaluation takes around 1.2 seconds on an RTX 3090 GPU, making it more than two orders of magnitude faster than the CFD solver. We further demonstrate the approach on a drag-minimization optimization problem with over 50 design variables, where the multifidelity model leads to an optimized design in good agreement with CFD with lift and drag errors below 6%.

I. Introduction

Multidisciplinary design optimization (MDO) is an engineering design methodology that can consider complex interactions between disciplines through numerical optimization. Solving large-scale MDO problems with many design variables can take several hours or even days [1]. In practice, computational fluid dynamics (CFD) is often one of the main computational bottlenecks during optimization.

To mitigate this issue, engineers commonly rely on low-fidelity models with simplified physics or data-driven surrogate models. For aerodynamics, low-fidelity models often come in the form of vortex lattice methods or panel methods which are computationally efficient and predict surface field quantities. Due to their limited physics, they neglect viscous, turbulent, and compressible flow effects. On the other hand, surrogate models trained on high-fidelity CFD data often focus on direct prediction of integrated quantities of interest. However, for MDO, field quantities are often required to capture fully accurate physics, or to perform multidisciplinary coupling such as aerostructural analysis.

Recently, advances in machine learning have led to a growing interest in building surrogate models for full 3D flow prediction as opposed to 2D or 1D quantities. However, existing deep learning approaches for full 3D field prediction remain mostly data-driven, relying on high-fidelity CFD data. Relatively little attention has been given to hybrid modeling approaches that combine cheap low-fidelity information with traditional data-driven techniques. Additionally, their application to gradient-based optimization, essential for multidisciplinary design optimization (MDO), remains limited.

In this paper, we address these gaps by developing a multifidelity model using graph neural networks to predict 3D aerodynamic flow fields around blended-wing-body aircraft configurations. The model uses differentiable panel method predictions as low-fidelity inputs and corrects them towards high-fidelity CFD states. Further, we embed the model in a fully differentiable workflow, enabling gradient-based optimization. We show that the panel method inputs improve

[1]PhD Candidate, Department of Mechanical and Aerospace Engineering, AIAA student member
[2]Associate Professor, Department of Mechanical and Aerospace Engineering, AIAA senior member

accuracy compared to the purely data-driven approach with a predicted lift and drag error of 10.29% and 5.11% on a CFD test set.

In addition, we demonstrate the gradient-based capabilities of the workflow by solving a simple aerodynamic shape optimization problem in which the multifidelity optimized geometry leads to lift and drag force errors of 2.41% and 5.74%, respectively.

II. Background

High-fidelity aerodynamic analysis is often a major computational bottleneck in design optimization, particularly in multidisciplinary design optimization (MDO), where repeated evaluations of aerodynamic quantities are required. To reduce this cost, surrogate models are widely used in place of CFD solvers. Traditional data-driven surrogate models often focus on predicting integrated quantities such as lift and drag coefficients, since these are the primary quantities needed for optimization. However, there is interest in surrogate models that predict spatial field outputs, including pressure and other flow-field quantities for coupled discipline analysis.

Due to recent advances in deep learning and increased attention towards data-driven models, recent work has explored surrogate models based on machine learning for aerodynamic field prediction [2]. Graph-based neural networks are a natural choice for this setting because they can operate directly on unstructured discretizations of the geometry. For example, graph network-based simulators, which represent physical systems as graphs, simulate physical phenomena by evolving node states through message passing [3]. MeshGraphNets demonstrated that message-passing neural networks on unstructured meshes can successfully learn dynamic physical systems [4]. X-MeshGraphNet extends MeshGraphNets by introducing graph-partitioning, halo regions, and eliminating the dependence on meshes [5]. In aerodynamics, Hines and Bekemeyer used a graph neural network to predict large-scale aircraft surface pressure distributions, demonstrating the applicability of graph-based surrogate models for industrial applications [6]. More recently, masked pretraining strategies have also been explored for graph neural networks in CFD, showing improved accuracy and data efficiency across a range of fluids problems [7]. Finally, physics-informed graph convolutional networks use physics-informed PDEs and boundary condition information in addition to data loss terms, showing promising results for aerodynamics applications [8].

Related work has also explored alternatives to graph-based approaches. Fourier neural operators learn in Fourier space to predict PDE solutions, showing success in fluids applications [9]. Geometry-informed neural operators extend this by allowing Fourier-based operators to be evaluated on irregular geometry using graph neural operators [10]. Deep operator networks have also been explored for geometry-dependent compressible aerodynamic flow prediction, including those in hypersonic and supersonic regimes [11]. Implicit neural representations have been used to predict aerodynamic fields and were shown to outperform MeshGraphNets on some aerodynamic prediction tasks [12].

Recent studies have also moved toward full 3D flow-field prediction. Roznowicz et al. demonstrated large-scale graph-based surrogate modeling for 3D external aerodynamic flow fields [13], while Flow3DNet proposed a deep learning framework for predicting structured 3D wing flow fields [14]. DoMINO is a multiscale, point-cloud-based architecture that uses local geometric information to predict flow states at discrete points [15]. There has also been work exploring 3D flow prediction for hypersonic aircraft utilizing attention mechanisms [16]. Finally, Transolver is a transformer-based neural PDE operator that can predict states on general geometries [17]. These studies show that recent machine learning models are able to predict volumetric aerodynamic states on large geometries. However, many of these approaches are primarily data-driven, learning a direct surrogate from inputs to flow states, relying only on CFD data.

A related direction is hybrid modeling, which aims to combine inexpensive, low-fidelity models with high-fidelity data. In multifidelity modeling, a model learns correlations between low- and high-fidelity data to improve predictive accuracy without relying on a large number of high-fidelity samples. Further, discrepancy modeling involves learning a surrogate to correct a lower-fidelity model to the desired higher-fidelity behavior [18]. Prior work has shown that multifidelity surrogate models can improve prediction accuracy compared with models trained only on high-fidelity data, especially when the amount of high-fidelity data is limited.

These hybrid modeling approaches have also been explored for aerodynamic applications. Belbute-Peres et al. combined a differentiable CFD solver with a graph convolutional network to build a hybrid neural model for fluid prediction [19]. In aerodynamics, VortexNet learns a correction to a vortex lattice method solution in order to predict aircraft surface pressures using a graph neural network [20, 21]. Other approaches have combined separate neural networks trained on low- and high-fidelity data to predict aerodynamic coefficients and fields [22, 23]. Multifidelity machine-learning surrogates have also been explored for aerodynamic design optimization. For example, Zhang et al. developed a multifidelity deep neural network framework for aerodynamic shape optimization by combining low-

and high-fidelity CFD data within a surrogate-based optimization loop [24], and later work extended multifidelity neural-network ideas to more generalizable aerodynamic wing-shape optimization settings [25].

Although recent work has demonstrated full-field prediction beyond scalar and surface quantities, fewer studies use hybrid modeling approaches that leverage low-fidelity solvers as input priors or as a correction base for full 3D state prediction. Additionally, work on applying these models for gradient-based design optimization workflows is also limited. These gaps are important in the MDO setting, where both accurate flow-field prediction and efficient derivative-based models are required.

III. Methodology

This section describes the multifidelity graph neural network and the differentiable aerodynamic surrogate for gradient-based optimization. We first define the problem of interest, the geometry, and the CFD dataset used for training. We then present the graph neural network architecture used and the low-fidelity panel-method component for the multifidelity model. Finally, we describe the details of the training procedure and the differentiable surrogate model for gradient-based optimization.

A. Problem Setup and Data

We consider the problem of predicting the 3D flow field around blended-wing-body (BWB) aircraft configurations at a nominal cruise condition. The baseline configuration, as shown in Figure 1, uses a symmetric airfoil in the centerbody, which blends into a transonic airfoil on the outer wing. The geometry, originally created in OpenVSP, is parameterized by a set of planform and shape variables summarized in Table 1 along with the sampling ranges used for data generation. The parameterization is implemented using `lsdo_geo`[1] [26], a differentiable geometry software package developed for multidisciplinary design optimization (MDO). All cases are evaluated at a fixed cruise condition: Mach 0.75, zero angle of attack, and an altitude of 35,000 ft.

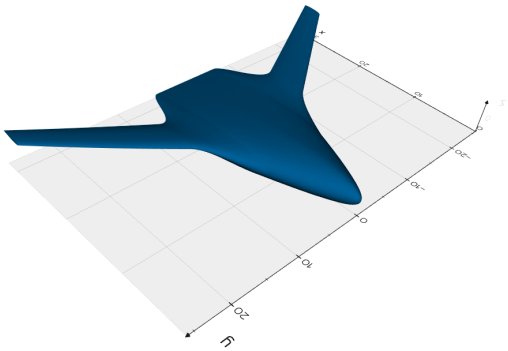

Fig. 1 The blended-wing-body configuration baseline geometry.

The high-fidelity dataset is generated using steady RANS CFD simulations with the open-source `OpenFOAM` solver. Each case is evaluated on a fixed-topology 3D volume mesh with approximately 639k cells representing half of the fluid domain. Boundary conditions include a no-slip wall condition on the aircraft surface, farfield conditions on the outer domain faces, and a symmetry condition along the aircraft centerbody plane. Turbulence was modeled using the Spalart–Allmaras model. A total of 125 CFD cases are generated using Latin hypercube sampling of the geometry parameter space shown in Table 1. All geometry variations are introduced through perturbed mesh volume points obtained by warping the baseline reference mesh in accordance with the specified geometry parameters. Each CFD simulation is run for a conservative 1000 solver time steps and all cases are verified to be below a maximum residual tolerance of 1×10^{-6} for all variables.

[1] https://github.com/LSDOlab/lsdo_geo

Table 1 BWB geometry parameters and sampling ranges from the baseline configuration for Latin hypercube sampling.

Variable	# of parameters	Sampling range
wing sweep displacement	1	±6 m
wing span	1	±7 m
centerbody span	1	±2 m
transition span	1	±2 m
wing chord	2	±1.5 m
centerbody chord	3	±8 m
wing twist	4	±10°
centerbody twist	2	±5°
wing thickness	8×4	±10%
centerbody thickness	8×4	±10%
wing camber	6×4	±10%
centerbody camber	6×4	±10%

Table 2 Multilevel graph hierarchy and message-passing depth during the downsampling and upsampling phases.

Level	# nodes	# edges	avg. edge length	avg. # cells / cluster	# down MP layers	# up MP layers
G^0	706k	2.89M	3.75 meters	1.00	1	1
G^1	305k	1.42M	7.57 meters	2.31	3	3
G^2	119k	0.54M	14.81 meters	5.92	7	—

B. Graph Neural Network Model Architecture

We use a multiscale graph neural network (GNN) as the main 3D flow-field prediction model due to the use of an unstructured mesh. Figure 2 shows the custom U-shaped GNN multiscale architecture used for the model based on the Graph U-Net idea [27].

For this problem, we directly use the CFD mesh to construct the graph representing the fluid domain around the BWB. Specifically, the volume mesh is converted to a graph by building the dual graph of the mesh where volume mesh cell centers are treated as the graph nodes and shared faces define edges between the nodes. Since volume cell centers do not exist on the wall, farfield, and symmetry faces, we build additional graph nodes on their respective face centers to encode boundary condition information into the graph. Edges are connected between these ghost nodes and their corresponding adjacent cell volume nodes. We let this graph be denoted by

$$G^0 = (V^0, E^0), \tag{1}$$

where each node in V^0 corresponds to a mesh cell center or a boundary face center, and edges connect nodes with shared faces in the mesh.

To create the multiscale hierarchy, the coarser graphs G^1 and G^2 are constructed from G^0 by applying several graph clustering passes. In each pass, candidate edges are first sorted by length, and nodes connected by these edges are iteratively merged if the edge lengths are below a fixed threshold. To keep boundary information, edges connected to boundary nodes are excluded from clustering. Each coarsening pass builds a fine-node to coarse-node map which is stored for downsampling and upsampling between graph levels. Coordinates for the coarsened nodes are taken as the average of the coordinates of the fine nodes assigned to that cluster.

Details of the resulting graphs are summarized in Table 2. A single edge in E^2 is approximately 4 times longer than an an edge in E^0, allowing messages to propagate farther in the domain per iteration.

Inputs to the network are passed in as features for each node $i \in V^0$ and edge $(i, j) \in E^\ell$ as summarized in Table 3. The input node feature vector $x_i \in \mathbb{R}^{14}$ is a concatenation of the six state values p, $\mathbf{v}$, T, and $\tilde{\nu}$ together with an

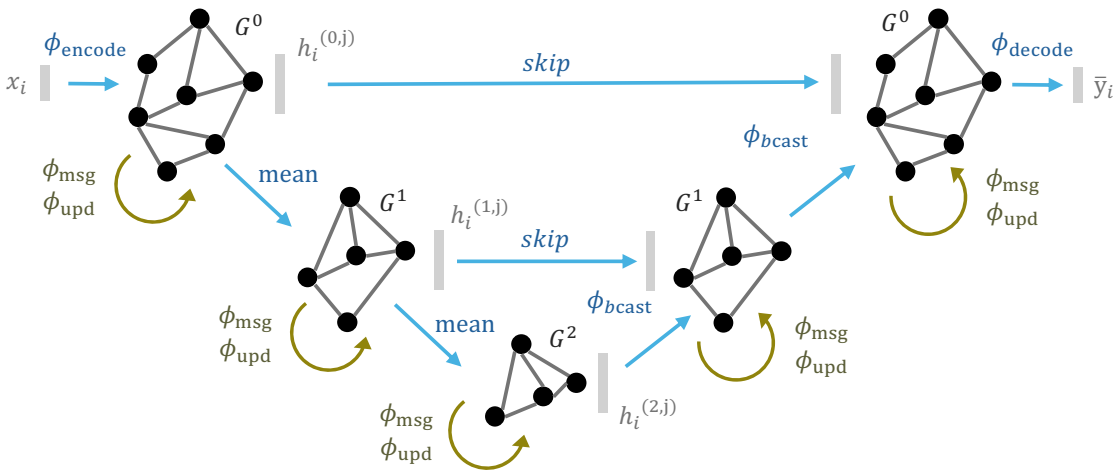

Fig. 2 The general GNN model with three graph levels. The fine graph is constructed directly from the CFD mesh while the coarsened graphs are precomputed by clustering nearby nodes together.

Table 3 Graph neural network input and output features

Category	Feature	Type
Node input	Normalized freestream/approximate states	$\mathbb{R}^6$
Node input	Approximate wall distance	$\mathbb{R}$
Node input	Direction to nearest wall	$\mathbb{R}^3$
Node input	Node type indicator	$\{0,1\}^4$
Edge input	Edge direction	$\mathbb{R}^3$
Edge input	Edge length	$\mathbb{R}$
Edge input	Edge type indicator	$\{0,1\}^4$
Node output	Normalized primitive flow variables	$\mathbb{R}^6$

approximate wall-distance value, an approximate unit vector pointing to the nearest wall, and a one-hot indicator for whether the node is part of the fluid domain, wall, farfield, or symmetry plane. Edge features $e_{ij} \in \mathbb{R}^8$ include the edge's unit direction, length and corresponding one-hot type indicators and are built for all edge levels. Because message-passing is performed in both directions, edge features are constructed for both orientations, where the unit direction vector is reversed.

Node features for V^0 are initially encoded into a 45-dimensional latent vector using an encoder network.

$$h_i^{(0,0)} = \phi_{\text{encode}}(x_i), \tag{2}$$

where $\phi_{\text{encode}} : \mathbb{R}^{14} \to \mathbb{R}^{45}$ is a learned multilayer perceptron (MLP). Latent vectors for the coarsened graphs $h_i^{(1,0)}$ and $h_i^{(2,0)}$ are initialized by downsampling from the higher-resolution graphs to the coarsened graph by averaging over the finer latent vectors.

Message-passing is then carried out on each graph G^0, G^1, and G^2. At graph level ℓ and message-passing iteration t, a single directional message along an edge $(i, j) \in E^\ell$ is computed as

$$m_{ij}^{(\ell,t)} = \phi_{\text{msg}}\left(\left[h_i^{(\ell,t)},\ h_j^{(\ell,t)},\ e_{ij}\right]\right), \tag{3}$$

where $\phi_{\text{msg}} : \mathbb{R}^{98} \to \mathbb{R}^{45}$ is a learned MLP. Because message-passing is performed in both edge directions, the

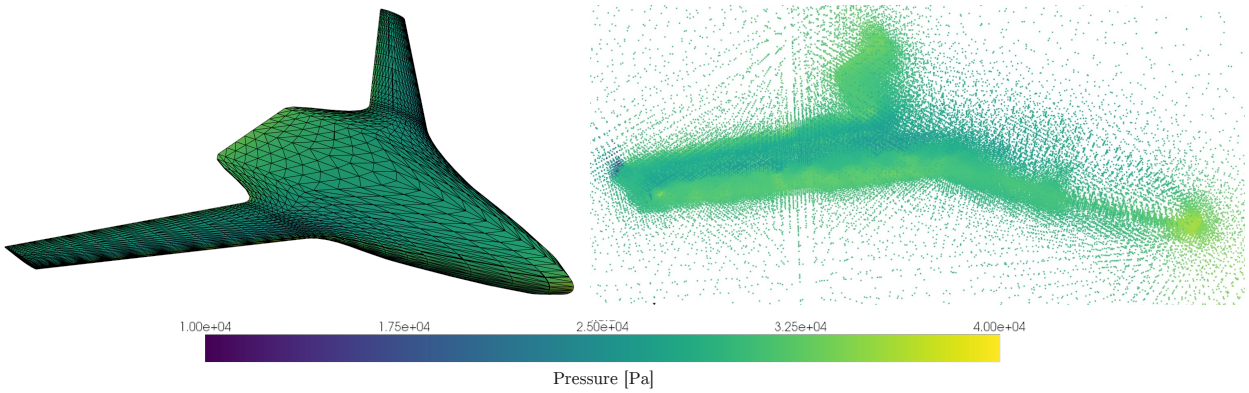

Fig. 3 Direct panel method pressure predictions at panel centers (left) and the interpolation to the 3D volume graph nodes (right).

aggregated message at a node i is

$$\bar{m}_i^{(\ell,t)} = \frac{1}{\deg(i)} \sum_{j \in \mathcal{N}(i)} m_{ji}^{(\ell,t)}, \tag{4}$$

where $\mathcal{N}(i)$ and $\deg(i)$ are the neighboring nodes and the number of neighbors for node i. Then, the node latent vectors at node i are updated through a residual network:

$$h_i^{(\ell,t+1)} = h_i^{(\ell,t)} + \phi_{\text{upd}}\Big(\big[h_i^{(\ell,t)},\ \bar{m}_i^{(\ell,t)},\ g^{(\ell,t)}\big]\Big), \tag{5}$$

where $\phi_{\text{upd}} : \mathbb{R}^{45} \to \mathbb{R}^{45}$ is another MLP and $g^{(\ell,t)}$ is an average of all latent vectors of the preceding message-passing iteration to capture global effects.

During the unpooling stage, coarse node latent vectors are upsampled back to fine latent vectors through a learned network based on the coarse node latent vector and saved skip connections:

$$\tilde{h}_i^{(\ell)} = h_i^{(\ell,\text{skip})} + \phi_{\text{bcast}}\Big(\big[h_i^{(\ell,\text{skip})},\ h_{C(i,\ell+1)}^{(\ell+1)}\big]\Big), \tag{6}$$

where $\phi_{\text{bcast}} : \mathbb{R}^{90} \to \mathbb{R}^{45}$ is an MLP and $C(i, \ell+1)$ is the corresponding clustered node in $V^{\ell+1}$ associated with node $i \in V^{\ell}$. Finally, the outputs are decoded using a final decoder network on G^0:

$$\bar{y}_i = \phi_{\text{decode}}(h_i^{(0,\text{out})}) \in \mathbb{R}^6, \tag{7}$$

where $\bar{y}_i$ corresponds to the six state outputs p, $\mathbf{v}$, T, and $\tilde{\nu}$. We note that the outputs are decoded for all nodes, but predictions at boundary nodes are ignored as they do not have values associated with them on the original CFD mesh. Further, MLPs ϕ_{msg}, ϕ_{upd}, and ϕ_{bcast} share parameters across graph levels, reducing parameter count and encouraging similar behavior across scales.

For this implementation, all learned mappings in the network are implemented as MLPs with two hidden layers and SiLU activation functions. ϕ_{encode}, ϕ_{bcast}, and ϕ_{decode} use hidden widths of 48 while ϕ_{upd} and ϕ_{msg} use hidden widths of 85.

Here, we differentiate the two model variants built from this same architecture. The first is a purely data-driven RANS surrogate model, which uses normalized freestream values as the flow inputs at each node. The second is a panel-method-guided multifidelity GNN, which uses normalized approximate panel method flow predictions as the input state at each node. The two models will be referred to as RANS-GNN and PG-RANS-GNN for the remainder of this paper. In the PG-RANS-GNN case, the network also predicts a correction to the panel-predicted state from the network outputs:

$$y_i^{\text{P}} = \hat{y}_i^{\text{P}} + \bar{y}_i, \tag{8}$$

where $\hat{y}^{\text{P}} \in \mathbb{R}^6$ denotes approximate normalized panel method states at node i. The RANS-GNN model predicts the normalized flow state directly:

$$y_i^{\text{D}} = \bar{y}_i. \tag{9}$$

Aside from the aforementioned differences, the two models are identical.

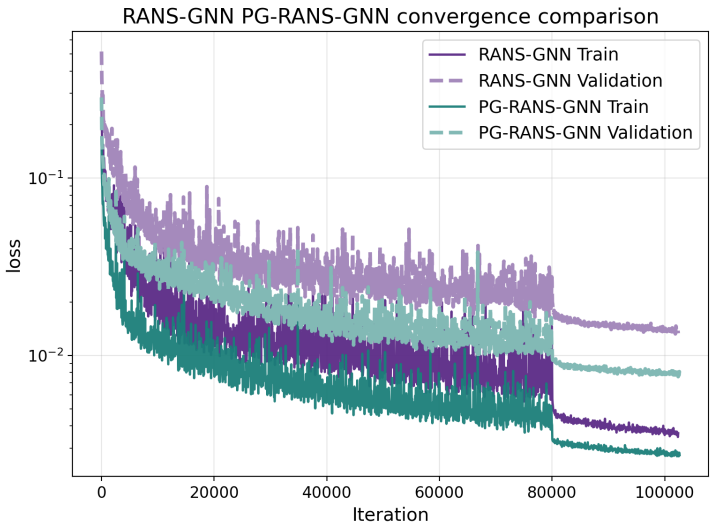

Fig. 4 Training and validation convergence histories for the data-driven RANS-GNN and multifidelity PG-RANS-GNN models. Both models were trained using the same optimizer and learning-rate schedule. PG-RANS-GNN reaches a lower final training and validation loss.

C. Low-fidelity Panel Method Inputs

The purpose of the multifidelity model is to provide the neural network with a cheap, physics-based approximation of the flow field before it predicts the full high-fidelity solution, allowing the network to focus on the more complex physics without having to learn the lower-frequency behavior. To provide the model with approximate flow states, we use a low-fidelity panel method prediction as the input to the PG-RANS-GNN model. As the panel method assumes inviscid and incompressible flow, it does not capture viscous, turbulent, or compressibility effects and is thus used for the low-fidelity model.

The panel method is computed using `VortexAD` [28], a fast, GPU-compatible, differentiable panel method solver developed for MDO. A panel mesh consisting of 5,264 triangles projected onto the geometry is used to evaluate surface pressures and velocities. Additional induced velocities and pressures at 1,000 randomly specified graph node coordinates are computed from the source and doublet distributions to serve as additional approximations of the volume field. Using these known quantities in addition to the known farfield freestream values, we interpolate values at each node of the graph to serve as node feature inputs for the PG-RANS-GNN multifidelity model using an inexpensive inverse-distance weighting algorithm. The algorithm uses the four nearest known points for each node which are precomputed based on the baseline mesh.

An example of the interpolated pressure distribution for the baseline geometry is shown in Figure 3. To maintain a consistent set of six flow inputs, we use freestream values for the temperature and the turbulence variable at each graph node because they are not computed by the panel method.

D. Training

The network is trained in a supervised manner to predict the six primitive flow variables on the graph for each training geometry. All target variables are normalized using the mean and standard deviation from the training set. For a single training geometry, the loss is defined as the sum of the mean-squared error between the predicted and normalized states over all graph nodes across the six states:

$$\mathcal{L} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} \sum_{k=1}^{6} w_i \left(y_{i,k} - \hat{y}_{i,k}^{CFD} \right)^2 \tag{10}$$

where N is the total number of graph nodes in the fluid domain, k indexes the six predicted flow variables, $y_{i,k}$ is the predicted normalized value of state k at node i, and $\hat{y}_{i,k}^{CFD}$ is the corresponding normalized CFD target value. Surface-adjacent graph nodes have $w_i = 4$ and the rest of the interior nodes have $w_i = 1$ to put slightly more emphasis

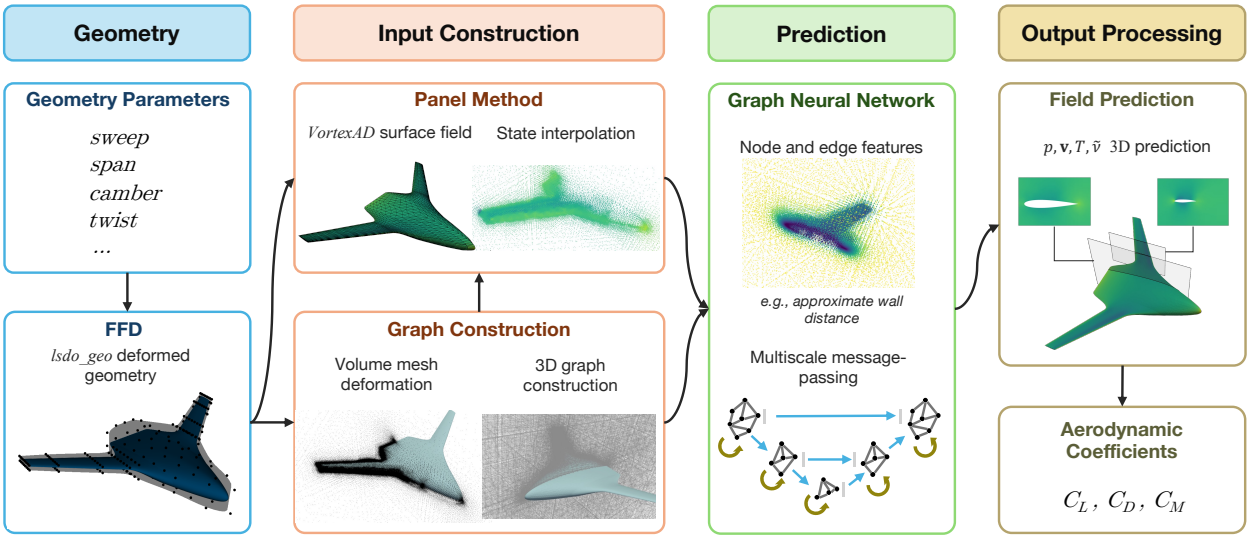

Fig. 5 An overview of the full differentiable workflow for the PG-RANS-GNN model, enabling gradient-based optimization. Derivatives between major components are automatically chained together using a combination of CSDL and JAX. The RANS-GNN model is identical but skips the panel method input step.

on near-surface quantities. Although scalar forces are available, they are not included in the loss function and the model is only trained on state values so as not to overemphasize integrated quantities.

The convergence histories for both models are shown in Figure 4. Both the RANS-GNN and PG-RANS-GNN models are trained using the same optimizer and training schedule for a fair performance comparison between the two models. Training is performed using the AdamW optimizer and the learning rate was lowered from an initial 10^{-3} to 10^{-4} after plateauing of the validation loss. Due to GPU memory limitations, batch size is limited to a single geometry configuration per iteration. Of the 125 original CFD cases, 100 cases were randomly selected for training, 23 for testing and 2 for validation. Training was terminated when the validation loss stopped decreasing for a fixed number of iterations. The best model was chosen by selecting a checkpoint with the lowest validation loss. For this study, we also use a fixed set of hyperparameters.

On an NVIDIA RTX 3090 GPU, a single forward evaluation of both graph-neural networks takes roughly 0.4 seconds while a backward pass requires approximately 1.2 seconds. Total training time for a model takes around 30 hours. The entire model is implemented using the JAX library [29].

E. Differentiable Prediction Model

Finally, we describe the end-to-end differentiable workflow that enables gradient-based optimization with the multifidelity GNN model. A key feature that enables differentiability is the use of a fixed graph topology that eliminates the need to use discrete methods that introduce discontinuities. This process is based on some CFD-based optimization workflows where a baseline mesh is warped in accordance with surface geometry changes.

An overview of the pipeline is summarized in Figure 5. First, geometry variations are introduced through a free-form deformation parameterization. The corresponding CFD graph node coordinates from V^0 are then updated using inverse-distance weighting based on the updated surface. Because the graph connectivity is fixed, the precomputed node clustering and pooling/unpooling maps remain unchanged across geometry changes. The newly warped graph points are then used to compute the approximate wall distance and direction vector per node as well as edge lengths and directions per edge to set the geometric node and edge features. Because cluster node mappings, closest surface nodes, and inverse-distance weighting nearest points are precomputed based on the baseline geometry mesh without any dependence on discrete logic during optimization, feature construction is fully differentiable.

As described in Section III.C, the panel method also evaluates the induced velocities and pressures given the deformed geometry; inverse-distance weighting is then used to interpolate quantities to the volume graph nodes.

Finally, PG-RANS-GNN, detailed in section III.B, is used to compute field quantities from which aerodynamic values can be extracted. The data-driven RANS-GNN model is also evaluated using the same aforementioned workflow,

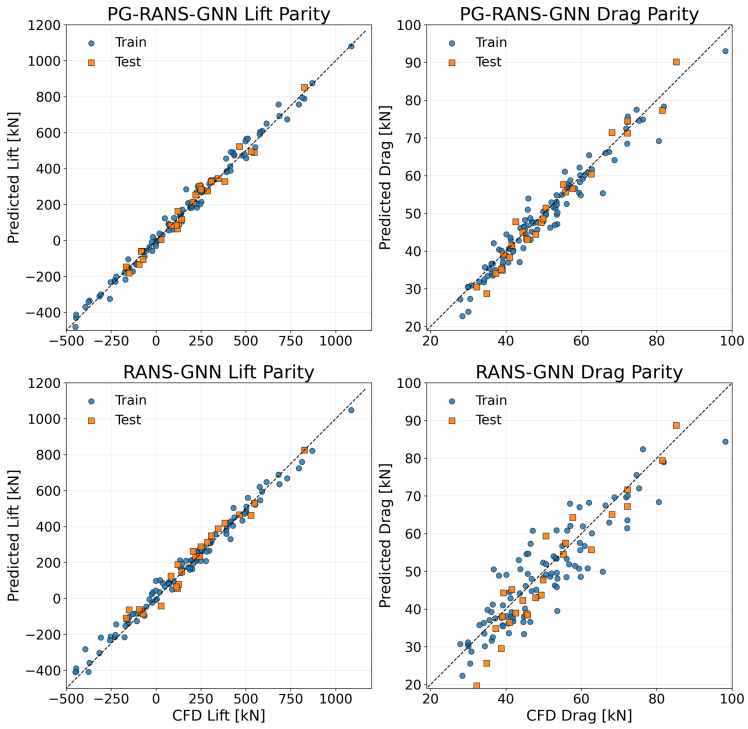

Fig. 6 Comparison between predicted and true aerodynamic force values for the RANS-GNN and PG-RANS-GNN models.

but flow field node input features are set as constants from the freestream condition.

Optional aerodynamic forces can be computed using predicted field quantities and the surface boundary mesh information. The pressure contribution is computed by integrating the predicted pressure over the wall faces using surface-adjacent pressure predictions. In addition, viscous contributions are computed using a wall function approximation along with the tangential component of velocity and the remaining states. The resulting surface forces are integrated and decomposed into the lift and drag directions.

Derivatives are provided for the geometry parameterization and the panel method by using `lsdo_geo` and `VortexAD`, respectively, as they are both implemented using `CSDL` [2], a graph-based modeling language for MDO that provides automatic differentiation capabilities. For the mesh-warping, IDWARP provides analytical vector-Jacobian products of perturbed volume points with respect to a surface mesh [30]. The graph neural network itself, input feature construction, and force calculation have derivatives provided through its JAX implementation. Finally, derivatives between major components are chained together using a combination of `CSDL` and JAX. Custom operations and vector-Jacobian product rules are utilized to connect components.

IV. Results

In this section, we evaluate the proposed surrogate models in several ways. First, we compare the predictive accuracy and computational cost of the data-driven graph neural network (RANS-GNN) and the panel-method multifidelity graph neural network (PG-RANS-GNN) on a test dataset. Second, we perform ablation studies to evaluate the impact of various components of the architecture. Finally, we perform optimization on a simple demonstration problem to demonstrate the ability to perform gradient-based optimization using this approach.

A. Model Performance

Table 4 summarizes the average test-set errors for predicted volume state variables and derived lift and drag coefficients. The table also shows approximate training and evaluation costs for each model. The field relative L_2 errors

[2]https://github.com/LSDOlab/CSDL_alpha

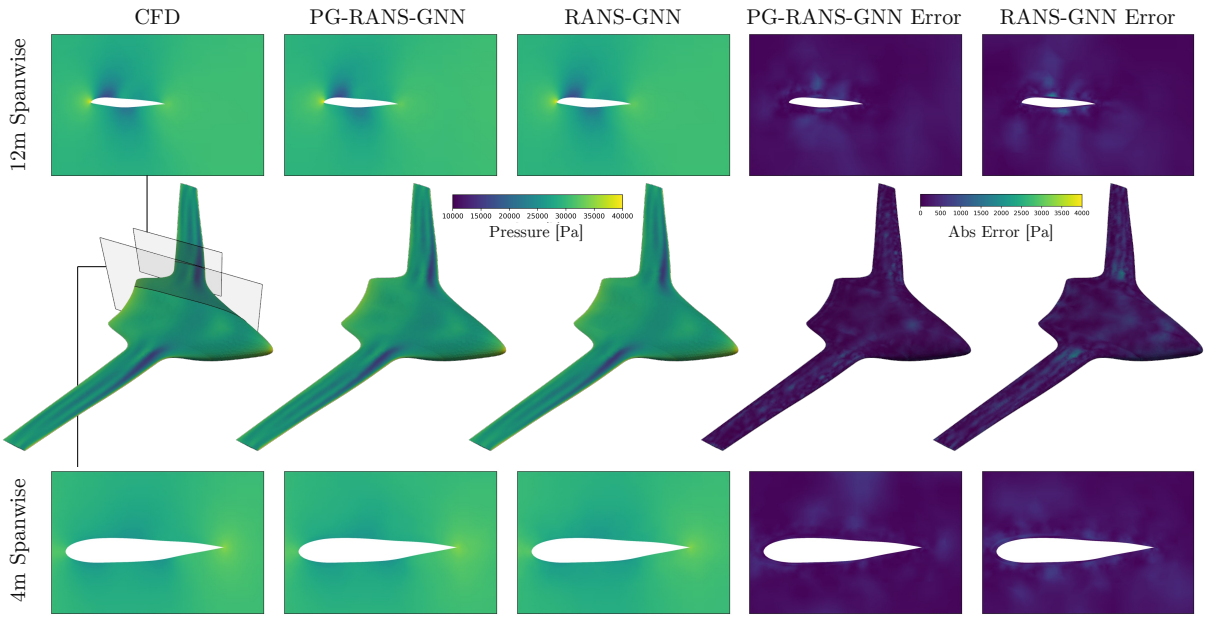

Fig. 7 3D pressure field comparison between the panel-method multifidelity graph neural network PG-RANS-GNN, the data-driven graph neural network RANS-GNN, and CFD pressures on a sample test geometry.

are unweighted, evaluated in physical units, and compared against the OpenFOAM CFD results. Reported values are averaged across the 23 test geometries. Similarly, the force errors compare reconstructed forces from the predicted field quantities to the raw OpenFOAM force output and are also averaged across the test set. To verify that the reported force errors are from prediction error rather than postprocessing error, the same force construction function was applied to the raw CFD fields and recovered both lift and drag forces to within 0.1% of the direct OpenFOAM force outputs. The lift relative error denominator has a minimum of 250 kN to prevent low-lift edge cases from significantly affecting the average relative error. A comparison between all predicted and known force values for both models on the training and test sets is shown in Figure 6.

The timing comparisons are estimates as they depend heavily on hardware and are intended only for general comparison. For CFD, the evaluation time was estimated using 15 MPI ranks on a standard desktop with 20 i7-2700 CPUs. The PG-RANS-GNN evaluation time includes the construction of the node features from the panel method evaluation along with the additional evaluation of 1,000 induced velocities in the flow field. Finally, the mesh-warping time is not included in the OpenFOAM or GNN evaluation times as it heavily depends on the number of MPI ranks used. For this case, performing mesh deformation on 700k points adds several seconds of run time on 15 i7-2700 processors.

As expected, the OpenFOAM CFD is the most expensive, while the panel method is much cheaper but only provides low-fidelity aerodynamic estimates. The two learned neural networks require offline training and data generation, but provide RANS predictions that are around two orders of magnitude faster than CFD when considering the mesh deformation step.

Although the upfront training cost of 30 hours is significant, it can be amortized in MDO problems which require a large amount of model evaluations such as in the case with optimization under uncertainty. In such settings, thousands of function evaluations may be required, where the total CFD evaluation cost can outweigh the one-time cost of training.

Among the two learned models, PG-RANS-GNN shows an error reduction between 14% to 50% compared to RANS-GNN across all reported quantities including the scalar forces. Although the panel method by itself produces large force errors, its use as an input still improves the final neural-network prediction. The RANS-GNN evaluation cost is much lower than that of the multifidelity model as it does not require the panel method solver for its evaluation. We also note that the CFD mesh and the neural network graph share an identical fixed topology throughout the workflow, with geometry changes only introduced through mesh warping. This design choice is advantageous for optimization workflows such as this one as it simplifies the learning process, but it also means that the model is not currently validated for remeshed geometries. As a result, its generalization beyond this specific mesh topology is still uncertain.

Figures 7 and 8 compare the predicted pressure and temperature fields from the RANS-GNN and PG-RANS-GNN

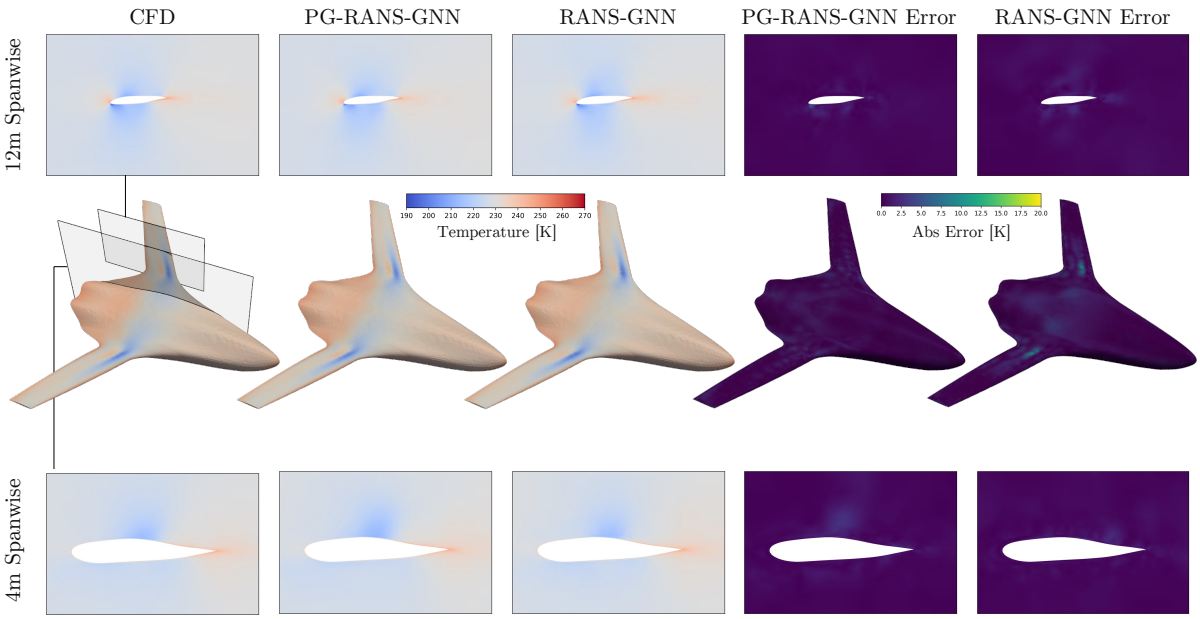

Fig. 8 3D temperature field comparison between the panel-method multifidelity graph neural network PG-RANS-GNN, the data-driven graph neural network RANS-GNN, and CFD temperatures on a sample test geometry.

Table 4 Performance and accuracy comparison across the high-fidelity CFD solver, low-fidelity panel method, and neural networks on 23 test geometries. Reported evaluation times are for inference with a single geometry and exclude the mesh-warping for the OpenFOAM, RANS-GNN, and PG-RANS-GNN models.

	Computational Cost			Volume Field Rel. L_2 Error (%)				Force Error (%)	
Model	Hardware	Train. (hr)	Eval. (s)	p	$\mathbf{v}$	T	$\tilde{\nu}$	Lift	Drag
OpenFOAM	i7-2700 (x15)	–	~1000	–	–	–	–	–	–
Panel Method	RTX 3090	–	~**0.18**	N/A	N/A	N/A	N/A	16.78	86.23
RANS-GNN	RTX 3090	~30	~0.34	1.24	4.03	0.46	21.37	12.89	10.56
PG-RANS-GNN	RTX 3090	~30	~1.23	**0.98**	**3.48**	**0.38**	**17.81**	**10.29**	**5.11**

models against the CFD solution for two random test geometry samples. Both models recover the main flow features, indicating that the graph neural networks are able to learn the overall 3D flow trends. The PG-RANS-GNN pressure prediction shows slightly lower absolute error compared to the RANS-GNN pressure for this sample. For the temperature case, a similar pattern can be observed where the RANS-GNN model has slightly higher errors near the transition region.

Finally, we note that the results shown are for one fixed set of train/test splits and network initializations, meaning different initializations can lead to better or worse performance for both models.

B. Ablation Studies

In this section, we investigate the contributions of the multiscale graph hierarchy and the panel-method input. All ablation experiments use identical training and test sets. The reported test losses were not used to select or tune the final model presented in the preceding section.

We first study the effect of the graph hierarchy by comparing single-level, two-level, and three-level variants of the model. This was done by first removing G^1 and G^2 from the original model, leading to the single-level model. Then, only the coarsest graph G^2 was removed to obtain the two-level model variant. Each of the three models is trained

Table 5 Ablation study of the graph hierarchy for the data-driven RANS-GNN and multifidelity PG-RANS-GNN models. The full three-level variant yields the lowest test loss for both models, indicating the benefits of the multiscale architecture. All models are trained independently using the same data split and training procedure.

Variant	Active Graph Levels	RANS-GNN Test Loss	PG-RANS-GNN Test Loss
Single-level	G^0	0.0442	0.0201
Two-level	G^0, G^1	0.0210	0.0142
Three-level (proposed)	G^0, G^1, G^2	**0.0142**	**0.0099**

Table 6 Ablation study comparing the RANS-GNN and PG-RANS-GNN models for different training-set sizes. PG-RANS-GNN achieves lower test loss at all training-set sizes, and its relative advantage increases as the number of training samples decreases, indicating the benefits of the multifidelity approach when the amount of data is limited.

# Training Samples	RANS-GNN Test Loss	PG-RANS-GNN Test Loss	Improvement (%)
25	0.0265	0.0153	42.3
50	0.0194	0.0118	39.2
100	**0.0142**	**0.0099**	30.2

independently using the same data split and training procedure. Because the networks ϕ_{msg}, ϕ_{upd}, and ϕ_{bcast} are shared across graph levels, the model parameterization remains fixed for all three models with the exception of ϕ_{bcast} being inactive for the single-level case.

Table 5 provides a performance comparison between the three model variants for both RANS-GNN and PG-RANS-GNN models. The results show that adding coarser graph levels consistently reduces test loss for both the RANS-GNN and PG-RANS-GNN models. This indicates that incorporating multiscale graph information improves predictive accuracy even with a mostly fixed parameter count. Importantly, this implies that we can improve model performance without relying on expensive message-passing operations on the full fine graph G^0. Finally, PG-RANS-GNN outperforms RANS-GNN across all graph hierarchies, suggesting that the panel method input provides value in addition to the multiscale information.

Next, we investigate how the panel method input changes model performance depending on the amount of available training data. In addition to the full model trained on all 100 samples, we train both RANS-GNN and PG-RANS-GNN using a random selection of 25 and 50 samples from the original training set.

A performance comparison is summarized in Table 6. As expected, the results show a decrease in predictive performance as the number of data samples shrinks. However, the PG-RANS-GNN model consistently outperforms the RANS-GNN model for all training set sizes and its relative improvement increases as the amount of data decreases. This trend suggests that using the multifidelity approach is especially valuable when access to data is limited.

C. Optimization Demonstration

To demonstrate the gradient-based optimization capabilities of our model, a simple drag-minimization aerodynamic optimization problem was considered:

$$\begin{aligned} \min_{\mathbf{x}} \quad & C_D \\ \text{subject to} \quad & C_L = 0.2, \\ & -10\% \leq \text{camber}_{\text{wing}} \leq 10\%, \\ & -10\% \leq \text{camber}_{\text{body}} \leq 10\%, \\ & -6^\circ \leq \text{twist}_{\text{centerbody}} \leq 6^\circ, \\ & -8^\circ \leq \text{twist}_{\text{wing}} \leq 8^\circ, \end{aligned} \tag{11}$$

The optimization problem consists of 54 design variables in total. Camber is parameterized using 48 design

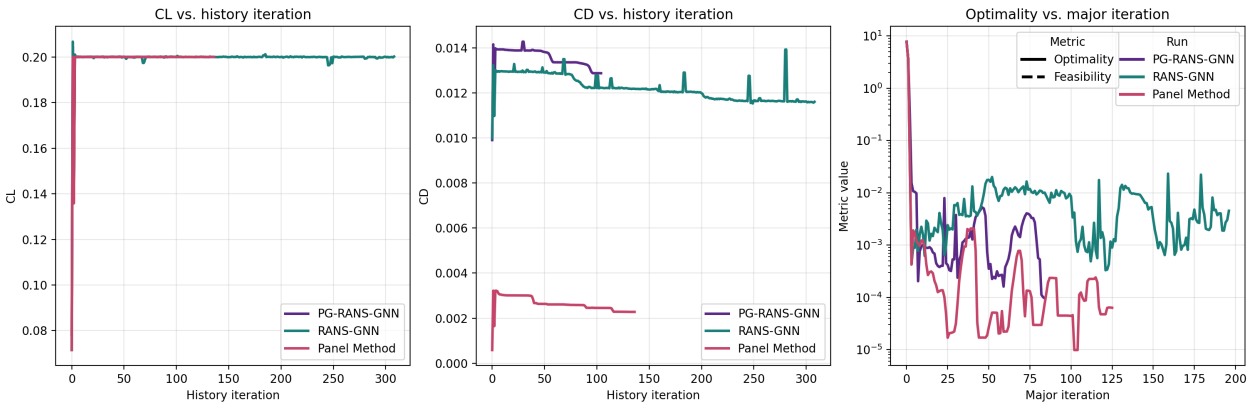

Fig. 9 Optimization convergence history of the demonstration optimization problem for the PG-RANS-GNN, RANS-GNN, and reference panel method models.

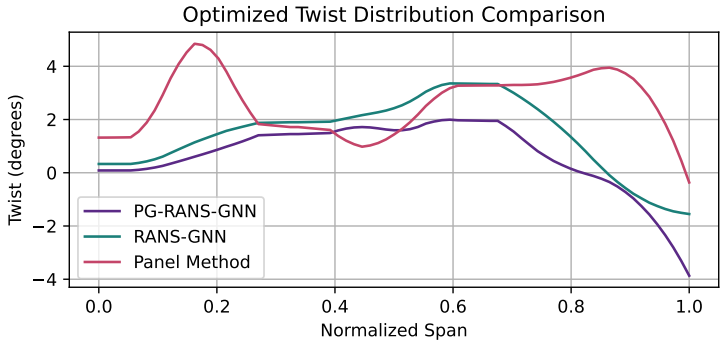

Fig. 10 Optimized twist distribution along the span of the aircraft for the PG-RANS-GNN, RANS-GNN, and the reference panel method model.

variables, while twist is parameterized using 6 variables distributed across the span. All remaining geometry parameters, including the planform variables, are held fixed at their baseline values. Because the planform is fixed throughout the optimization, a constant reference area is used when computing the lift and drag coefficients, C_L and C_D. As the baseline geometry does not satisfy the constraints, the purpose of this optimization is to find a design that satisfies the predetermined lift constraint and reduces drag as much as possible. Identical optimization problems were solved using the PG-RANS-GNN, RANS-GNN, and the panel method for reference.

The optimization is performed at a fixed cruise operating condition of Mach 0.75, zero angle of attack and an altitude of 35,000 ft, as described in Section III.A. Thus, the optimized aerodynamic quantities are evaluated within the same operating conditions used for the neural network training.

As described in Section III.E, the implementation combines JAX and CSDL within a fully differentiable workflow. Derivatives are propagated through each component of the pipeline, enabling the use of gradient-based optimization.

The optimization was carried out using the interior-point method from the MODOPT optimization library [31] with a feasibility and optimality tolerance of 10^{-4} for both RANS-GNN and PG-RANS-GNN. All geometric design variables are initialized at the baseline values, with identical initial camber perturbations across models. The convergence history is shown in Figure 9, where it can be observed that the PG-RANS-GNN, RANS-GNN, and reference panel method models all found feasible designs satisfying the lift constraint while reducing drag. This confirms that the surrogate models can be successfully used in a gradient-based workflow. All three cases exhibited similar convergence behavior, where the lift constraint was satisfied quickly while drag decreased over the course of the optimization. The optimized twist distribution for the three models are shown in Figure 10.

Final pressure field cross sections for the PG-RANS-GNN and RANS-GNN optimized cases can be seen in Figure 11. The final optimized surface pressure distributions for both models compared against the corresponding OpenFOAM

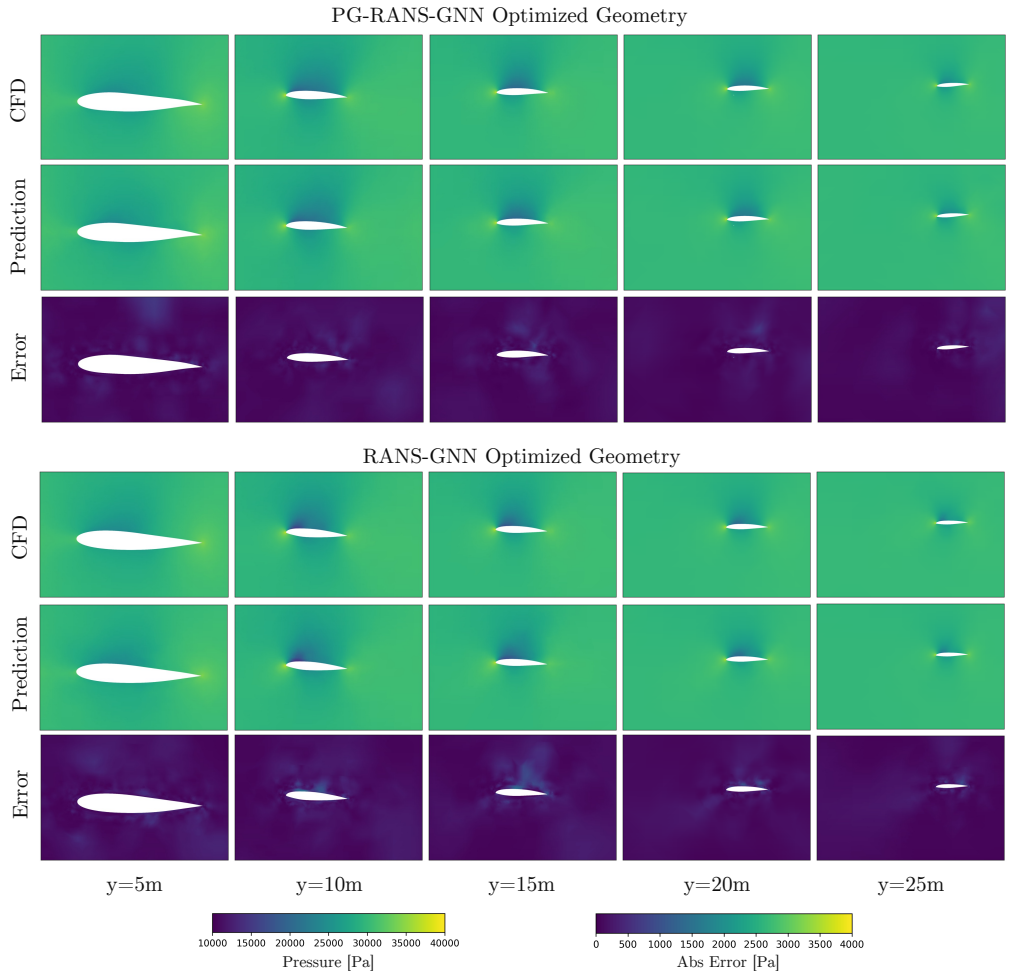

Fig. 11 Pressure field cross sections of the optimized PG-RANS-GNN (top) and RANS-GNN (bottom) geometries.

CFD solutions are shown in Figure 12. Both models recover the main pressure field behavior of the optimized designs although errors remain in some areas.

To evaluate whether the optimized designs remain accurate under CFD, all final optimized designs were rerun in OpenFOAM. The results are tabulated in Table 7. The PG-RANS-GNN design gives the best agreement in lift, slightly underpredicting the target by 2.4%. The RANS-GNN optimized geometry gives a less accurate lift coefficient, underpredicting lift by 13.89%. Both optimized drag coefficients are similar, with both designs producing $C_D \approx 0.012$ which is slightly different from their predicted values. This demonstrates the accuracy benefits of the multifidelity model, finding a similar drag value to the RANS-GNN model with much better lift satisfaction.

Both models outperform the panel method solution; with the latter leading to a design that overpredicts lift and significantly underpredicts drag. This indicates that the panel method alone is not sufficient for this optimization due to its lower-fidelity physics assumptions.

Overall, these trends are consistent with the test dataset errors summarized in Section IV.A, where the panel method coupled with the graph neural network led to higher model accuracy than either approach alone.

V. Conclusion

In this paper, we developed a multifidelity model using message-passing graph neural networks for predicting 3D aerodynamic flow fields around blended-wing-body aircraft configurations in a workflow suitable for gradient-based optimization. The proposed approach combines a low-fidelity panel-method solver with a graph neural network that predicts the corresponding high-fidelity RANS flow field on the CFD mesh. By embedding the surrogate model within

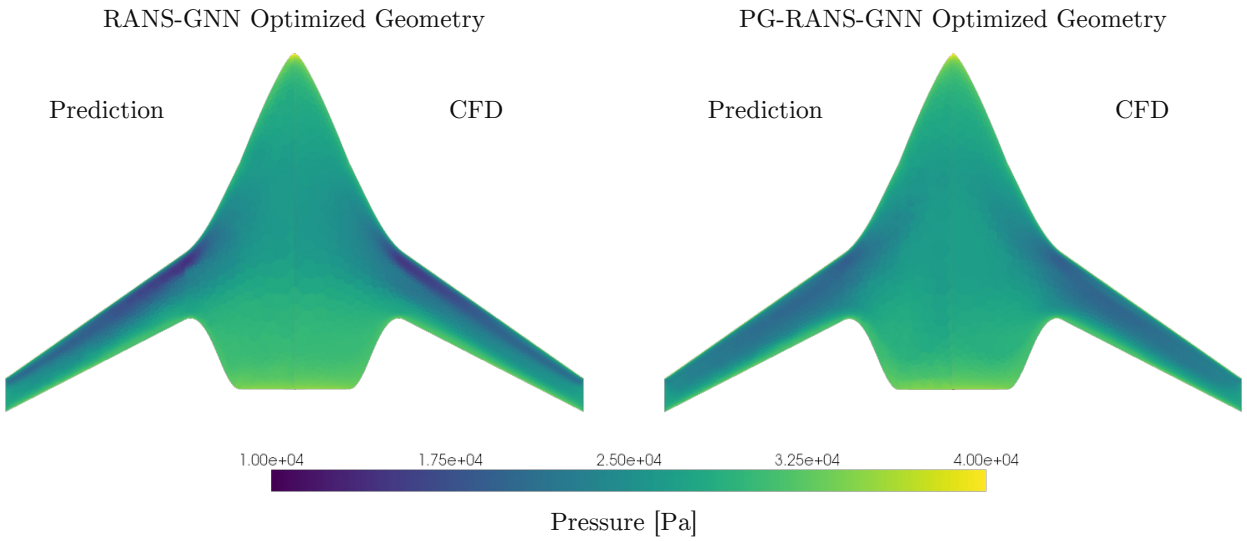

Fig. 12 The optimized pressure distributions for the RANS-GNN model (left) and PG-RANS-GNN model (right) compared against the corresponding OpenFOAM CFD solutions.

Table 7 Comparison of optimized design accuracy for RANS-GNN, PG-RANS-GNN, and the reference panel method models with corrected aircraft force coefficients.

Model	C_L^{pred}	C_L^{OF}	C_L Error (%)	C_D^{pred}	C_D^{OF}	C_D Error (%)
Baseline	–	0.072	–	–	0.0096	–
Panel Method (Optimized)	0.200	0.258	22.49	0.0023	0.0184	87.66
RANS-GNN (Optimized)	0.200	0.176	13.89	0.0116	0.0121	3.75
PG-RANS-GNN (Optimized)	0.200	0.195	2.41	0.0129	0.0122	5.74

a fully differentiable workflow, the method enables efficient field prediction along with derivatives for gradient-based MDO. We trained this multifidelity model along with a reference data-driven model using 100 CFD solution cases.

The results show that incorporating the panel method into the model improves predictive accuracy compared to a data-driven model. On the test set, the multifidelity model achieved relative L_2 volume errors of less than 4% for pressure, temperature, and velocity while achieving lift and drag errors of 10.29% and 5.11% compared to the CFD reference. Incorporating the panel method into the surrogate model reduced errors across all reported volume-state variables and improved reconstructed lift and drag accuracy, despite the low-fidelity panel method itself being significantly less accurate than CFD. In addition, both surrogate models are significantly faster than the CFD solver itself. Ablation studies revealed that the benefits of the panel method input increased as the amount of data decreased. In the simplified demonstration problem, both the data-driven and multifidelity models converged successfully, confirming the differentiability of the workflow. However, the multifidelity model produced an optimized design whose aerodynamic performance agree more closely with CFD results, particularly for lift. Overall, these results suggest that multifidelity surrogate models can offer a balance between speed, accuracy, and differentiability for future aircraft MDO workflows.

Several important directions remain for future work. First, a more extensive ablation study is needed to determine how the architecture affects the influence of the low-fidelity inputs on the predictions. This includes a more systematic study of the contributions from the panel method inputs, input features, as well as standard hyperparameter selection. Further, as the CFD data was generated in-house with limited resources, using a better-validated and larger dataset would strengthen confidence in our model. Increasing the size of the training set to account for different operating conditions will also be necessary for more advanced MDO problems. Finally, using this approach to accelerate coupled optimization problems such as aerostructural cases is an important direction for future work.

Acknowledgments

The authors would like to acknowledge the Multidisciplinary Science and Technology Center of the Aerospace Warfare Directorate, Air Force Research Laboratory for funding and support for this effort through the Collaborative Center for Design and Research of Interdisciplinary Systems program.

The authors would also like to thank Edward Lowell for his assistance with CFD, as well as Jason Kao and Dean Bryson for their regular feedback.

AI was used to make minor edits to the text to improve readability, grammar, and language in accordance with AIAA's ethical standards.

Distribution A: Approved for public release; distribution is unlimited. AFRL-2026-2231.

References

[1] Ruh, M. L., Fletcher, A., Sarojini, D., Sperry, M., Yan, J., Scotzniovsky, L., van Schie, S. P., Warner, M., Orndorff, N. C., Xiang, R., Joshy, A. J., Zhao, H., Krokowski, J., Gill, H., Lee, S., Cheng, Z., Cao, Z., Mi, C., Silva, C., Wolfe, L., Chen, J.-S., and Hwang, J. T., *Large-scale multidisciplinary design optimization of a NASA air taxi concept using a comprehensive physics-based system model*, 2024. https://doi.org/10.2514/6.2024-0771, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-0771.

[2] Bonnet, F., Mazari, A. J., Cinnella, P., and Gallinari, P., "AirfRANS: High Fidelity Computational Fluid Dynamics Dataset for Approximating Reynolds-Averaged Navier-Stokes Solutions," , 2023. URL https://arxiv.org/abs/2212.07564.

[3] Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P., "Learning to simulate complex physics with graph networks," *International conference on machine learning*, PMLR, 2020, pp. 8459–8468.

[4] Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W., "Learning mesh-based simulation with graph networks," *arXiv preprint arXiv:2010.03409*, 2020.

[5] Nabian, M. A., Liu, C., Ranade, R., and Choudhry, S., "X-MeshGraphNet: Scalable Multi-Scale Graph Neural Networks for Physics Simulation," , 2024. URL https://arxiv.org/abs/2411.17164.

[6] Hines, D., and Bekemeyer, P., "Graph neural networks for the prediction of aircraft surface pressure distributions," *Aerospace Science and Technology*, Vol. 137, 2023, p. 108268. https://doi.org/https://doi.org/10.1016/j.ast.2023.108268, URL https://www.sciencedirect.com/science/article/pii/S1270963823001657.

[7] Garnier, P., Lannelongue, V., Viquerat, J., and Hachem, E., "MeshMask: Physics-Based Simulations with Masked Graph Neural Networks," , 2025. URL https://arxiv.org/abs/2501.08738.

[8] Peng, J.-Z., Hua, Y., Aubry, N., Chen, Z.-H., Mei, M., and Wu, W.-T., "Data and physics-driven modeling for fluid flow with a physics-informed graph convolutional neural network," *Ocean Engineering*, Vol. 301, 2024, p. 117551. https://doi.org/https://doi.org/10.1016/j.oceaneng.2024.117551, URL https://www.sciencedirect.com/science/article/pii/S0029801824008886.

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

[10] Li, Z., Kovachki, N. B., Choy, C., Li, B., Kossaifi, J., Otta, S. P., Nabian, M. A., Stadler, M., Hundt, C., Azizzadenesheli, K., and Anandkumar, A., "Geometry-Informed Neural Operator for Large-Scale 3D PDEs," , 2023. URL https://arxiv.org/abs/2309.00583.

[11] Peyvan, A., Kumar, V., and Karniadakis, G. E., "Fusion-DeepONet: A data-efficient neural operator for geometry-dependent hypersonic and supersonic flows," *Journal of Computational Physics*, Vol. 544, 2026, p. 114432. https://doi.org/https://doi.org/10.1016/j.jcp.2025.114432, URL https://www.sciencedirect.com/science/article/pii/S0021999125007144.

[12] Catalani, G., Agarwal, S., Bertrand, X., Tost, F., Bauerheim, M., and Morlier, J., "Neural fields for rapid aircraft aerodynamics simulations," *Scientific Reports*, Vol. 14, No. 1, 2024, p. 25496.

[13] Roznowicz, D., Stabile, G., Demo, N., Fransos, D., and Rozza, G., "Large-scale graph-machine-learning surrogate models for 3D-flowfield prediction in external aerodynamics," *Advanced Modeling and Simulation in Engineering Sciences*, Vol. 11, No. 1, 2024, p. 6.

[14] Zuo, K., Ye, Z., Yuan, X., and Zhang, W., "Flow3DNet: A deep learning framework for efficient simulation of three-dimensional wing flow fields," *Aerospace Science and Technology*, Vol. 159, 2025, p. 109991.

[15] Ranade, R., Nabian, M. A., Tangsali, K., Kamenev, A., Hennigh, O., Cherukuri, R., and Choudhry, S., "DoMINO: A Decomposable Multi-scale Iterative Neural Operator for Modeling Large Scale Engineering Simulations," , 2025. URL https://arxiv.org/abs/2501.13350.

[16] Shen, Y., Huang, W., and guo Wang, Z., "Achieving generalized three-dimensional flow field prediction for high-speed flight vehicles using an attention-inspired architecture," *Computers & Fluids*, Vol. 299, 2025, p. 106726. https://doi.org/https://doi.org/10.1016/j.compfluid.2025.106726, URL https://www.sciencedirect.com/science/article/pii/S0045793025001860.

[17] Wu, H., Luo, H., Wang, H., Wang, J., and Long, M., "Transolver: A Fast Transformer Solver for PDEs on General Geometries," , 2024. URL https://arxiv.org/abs/2402.02366.

[18] Kennedy, M. C., and O'Hagan, A., "Predicting the output from a complex computer code when fast approximations are available," *Biometrika*, Vol. 87, 2000, pp. 1–13. URL https://api.semanticscholar.org/CorpusID:119357596.

[19] Belbute-Peres, F. D. A., Economon, T., and Kolter, Z., "Combining differentiable PDE solvers and graph neural networks for fluid flow prediction," *international conference on machine learning*, PMLR, 2020, pp. 2402–2411.

[20] Shen, Y., Needels, J., and Alonso, J., "VortexNet: A Graph Neural Network-Based Multi-Fidelity Surrogate Model for Field Predictions," , 12 2024. https://doi.org/10.2514/6.2025-0494.

[21] Shen, Y., and Alonso, J., "Graph Neural Network-Guided Aerodynamic Shape Optimization for Conceptual Design of Supersonic Transport Wings," 2025. https://doi.org/10.2514/6.2025-3228.

[22] Taghizadeh, M., Nabian, M. A., and Alemazkoor, N., "Multifidelity graph neural networks for efficient and accurate mesh-based partial differential equations surrogate modeling," *Computer-Aided Civil and Infrastructure Engineering*, Vol. 40, No. 7, 2025, pp. 841–858. https://doi.org/https://doi.org/10.1111/mice.13312, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.13312.

[23] Li, J., Li, Y., Liu, T., Zhang, D., and Xie, Y., "Multi-fidelity graph neural network for flow field data fusion of turbomachinery," *Energy*, Vol. 285, 2023, p. 129405. https://doi.org/https://doi.org/10.1016/j.energy.2023.129405, URL https://www.sciencedirect.com/science/article/pii/S0360544223027998.

[24] Zhang, X., Xie, F., Ji, T., Zhu, Z., and Zheng, Y., "Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization," *Computer Methods in Applied Mechanics and Engineering*, Vol. 373, 2021, p. 113485. https://doi.org/https://doi.org/10.1016/j.cma.2020.113485, URL https://www.sciencedirect.com/science/article/pii/S0045782520306708.

[25] Yang, A., Li, J., and Liem, R. P., "Generalizable Multifidelity Aerodynamic Wing Shape Design Optimization," *Journal of Aircraft*, Vol. 0, No. 0, 0, pp. 1–17. https://doi.org/10.2514/1.C038587, URL https://doi.org/10.2514/1.C038587.

[26] Fletcher, A. H., and Hwang, J. T., "Implicit Nonlinear Geometry Parameterization for Multidisciplinary Design Optimization," *Proceedings of the ASME 2026 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference IDETC/CIE2026*, 2026. Accepted.

[27] Gao, H., and Ji, S., "Graph U-Nets," , 2019. URL https://arxiv.org/abs/1905.05178.

[28] Scotzniovsky, L., and Hwang, J. T., "A fast, memory-efficient panel method for large-scale multidisciplinary design optimization under uncertainty using graph-based modeling," *AIAA AVIATION FORUM AND ASCEND 2025*, 2025, p. 3021.

[29] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Katariya, Y., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q., "JAX: composable transformations of Python+NumPy programs," , 2018. URL http://github.com/jax-ml/jax.

[30] Secco, N. R., Kenway, G. K., He, P., Mader, C., and Martins, J. R., "Efficient mesh generation and deformation for aerodynamic shape optimization," *AIAA Journal*, Vol. 59, No. 4, 2021, pp. 1151–1168.

[31] Joshy, A. J., and Hwang, J. T., "modOpt: A modular development environment and library for optimization algorithms," *Advances in Engineering Software*, Vol. 213, 2026, p. 104084. https://doi.org/10.1016/j.advengsoft.2025.104084.