MULTI-FIDELITY AERODYNAMIC SURROGATE MODELING FOR CONCEPTUAL DESIGN USING GRAPH NEURAL NETWORKS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Yiren Shen
June 2026

© 2026 by Yiren Shen. All Rights Reserved.
Re-distributed by Stanford University under license with the author.



This work is licensed under a Creative Commons Attribution-Noncommercial 3.0 United States License.
https://creativecommons.org/licenses/by-nc/3.0/legalcode

This dissertation is online at: https://purl.stanford.edu/cp141hk7861

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Juan Alonso, Primary Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mykel Kochenderfer

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daniel Tartakovsky

Approved for the Stanford University Committee on Graduate Studies.

Kenneth Goodson, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format.

Abstract

New product development is inherently multidisciplinary, constraint-driven, and multi-objective. Aircraft development, due to its highly sophisticated nature, demands a rigorous approach. As a result, a phased product development cycle is used: Starting from product requirements, the product progresses through conceptual design, preliminary design, and detailed design. Early design phases prioritize rapid trade studies and system-level feasibility, while later stages focus on manufacturing implementation. Throughout the design process, the confidence that the final product will meet requirements increases, while commitments of technical and financial resources intensify.

Among these phases, conceptual design, which translates program requirements into concrete aircraft configurations, remains particularly challenging. Despite being the most critical design stage in terms of life-cycle cost impact, the conceptual design phase is characterized by opacity and fluidity. In this phase, prior program experience and domain knowledge are used to navigate the design space, and designers must iterate rapidly to conduct trade studies across multiple candidates and perform optimizations. Under these constraints, low-fidelity analysis tools are often employed, along with empirical formulas. The limited fidelity of such approaches may produce biased or inaccurate predictions that mislead the search direction. This creates a fundamental fidelity-cost gap in the design process: *the desired level of analysis fidelity is unattainable because of the cost required to generate it.*

The conceptual design of supersonic transport (SST) exemplifies this gap. The need for SST to operate across multiple flow regimes, together with the strong coupling among the design variables used to tune vehicle performance, creates a compelling need to introduce high-fidelity analysis into the conceptual design process. Among the many challenges in conceptual design, planform selection for low-speed stability, which requires accurate predictions of lift and moment coefficients and their derivatives, has long been a bottleneck.

To address this problem, this dissertation presents a multi-fidelity surrogate model (`MF-VortexNet`). The model uses graph neural network (GNN) to perform field-level corrections to low-fidelity vortex lattice method (VLM) predictions by leveraging information learned from high-fidelity computational fluid dynamics (CFD) simulations. The proposed framework assimilates pressure field data from high- and low-fidelity sources into a common reference representation based on graphs and constructs a mapping function for low-fidelity correction. This approach enables richer engineering insight through field-level corrections, allows arbitrary aerodynamic coefficients to be evaluated from the augmented field variables, and demonstrates geometric generalizability across a design space.

This dissertation further examines how the predictive accuracy of `MF-VortexNet` scales with training data size through an empirical study. The results quantify how increasing sampling density in the design space improves surrogate accuracy, providing guidance for dataset construction. The scaling analysis also reveals the limitations of GNNs for surrogate modeling, specifically, scaling up the feature dimension yields diminishing returns in model performance.. In addition, the proposed multi-fidelity surrogate framework is integrated into a conceptual design environment (`SUAVE`) and applied to a stability-constrained supersonic transport planform optimization problem. This integration enables high-fidelity-informed design exploration, improves design confidence across the design space without sacrificing computational efficiency, and allows nonlinear aerodynamic effects to be incorporated into early-stage multidisciplinary design optimization (MDO). Along this process, an open-source multi-fidelity aerodynamic dataset (`double-delta-aero`) that contains both high-fidelity and low-fidelity simulations is created to serve as a benchmark problem for the community.

Together, this dissertation demonstrates that using graph-based machine learning to bridge the fidelity-cost gap between solvers offers a practical path toward higher-confidence conceptual design, particularly for aircraft concepts that have little historical knowledge. More broadly, it shows that the combination of machine learning and classical aerodynamic tools can be a promising direction for reducing new product development program risk and advancing future research in multi-fidelity MDO.

Acknowledgments

Being able to pursue my Ph.D. studies at Stanford University has been a true privilege. Without the intellectual freedom, generous support, world-class mentors, and amazing cohort of colleagues I found here, completing this thesis would have been not only more difficult, but also far less enjoyable. I am deeply grateful to all those who have shaped my Ph.D. journey, both intellectually and personally.

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Juan Alonso. His guidance, vision, care, and support have been central to my development as a researcher. The idea behind this thesis first took shape in a routine one-on-one meeting, where we discussed ways to augment the analysis fidelity of conceptual design tools. Over the years, through his encouragement and guidance, that idea matured and eventually became this dissertation. I am also grateful for his patience and support when my Ph.D. journey was disrupted by the global pandemic. During a difficult period, he supported me through my downtime and helped me return to the Ph.D. program. This thesis would not have been possible without his support.

I am sincerely thankful to my reading committee members, Professor Mykel Kochenderfer and Professor Daniel Tartakovsky. Over the years, I was fortunate to take classes with them and receive their advice. My initial exposure to machine learning-based multi-fidelity modeling came from Professor Tartakovsky's work, and since then, I have deeply appreciated every one-on-one meeting we have had. Those conversations greatly shaped the structure and direction of this dissertation. Similarly, I learned a great deal from Professor Kochenderfer's optimization class, and many of the optimization methods used in this thesis can be traced back to his lecture notes. I would also like to thank my oral examination committee members, Professor Tonghun Lee and Professor Madeleine Udell. Toward the final quarter of my Ph.D., I was able to discuss my work and receive thoughtful feedback from them. Their feedback helped me strengthen the final form of this thesis.

Throughout my Ph.D., I received support from several organizations and grants. I would like to thank Robert Bosch LLC for supporting my first-year study on geometric sensitivity, with Daniel Mayer and Nijso Beishuizen as external collaborators. I was later supported by the Air Force Office of Scientific Research through a Multi-University Research Initiative (MURI), and by NASA under the Commercial Supersonics Technology (CST) program, Supersonic Configurations at Low Speeds (SCALOS). A significant portion of this thesis is computational, and I would like to thank the Stanford Research Computing Center and Stanford Data Science for providing computational resources on the Sherlock and Marlowe clusters. I also thank the Google Cloud Research Credit Program for granting access to GPU resources on the Google Cloud Platform.

I would also like to thank the Department of Aeronautics and Astronautics for providing an intellectually stimulating and supportive environment. I am especially grateful to the department staff, including Patrick Ferguson, Joyce Jun, and Jenny Scholes, among many others, for their program support and for offering me valuable teaching opportunities.

This work also benefited from collaborations and conversations with researchers beyond Stanford. I am grateful to those who provided feedback on my work in surrogate modeling and supersonic transport aircraft design. I am especially indebted to my colleagues from the University of Washington, including Professor Eli Livne, Kuang-Ying "Eddie" Ting, Chester Nelson, and Kenneth Wiersema, for their valuable discussions on supersonic aircraft configurations and for generously sharing their knowledge and experience within the SCALOS program. I would also like to thank Charlie Lew and Nadine Auda from Rune Aero Inc. for their feedback and collaboration on machine learning-based surrogate modeling, and Marshall Galbraith from MIT for his coaching and support on the geometric sensitivity project.

To my labmates at the Aerospace Design Laboratory (ADL), I owe many thanks. I was fortunate to overlap with many ADL alumni, from Jayant Mukhopadhaya, who patiently helped me debug scripts for my qualification exam project, to more recent colleague Beverley Yeo, from whom I learned about mixed-precision solvers. I would like to give special acknowledgment to Emilio Botero, Jacob Needels, Harsh Patel, and Zan Xu for their guidance and collaboration. I also want to thank Atharva Aalok, Ujjwal Dalmia, Aurelien Ghiglino, Sangjoon Lee, Simon Li, Brian Munguía, Rina Onishi, Arjun Shah, Oliver Szavuj, Ashok Viswanathan, and Molly "Miao" Zhang for their friendship. A Ph.D. is often described as an individual pursuit, but in reality, mine was shaped by countless conversations,

discussions, meals, walks, trips, and shared experiences. I have benefited greatly from the technical insights, feedback, and encouragement of my peers. Every moment counts, and I thank you all for making this period of my life memorable.

Most importantly, I would like to thank my family. Their love, sacrifice, and belief in me have been the foundation of everything I have done. I am deeply grateful to my parents, Hong and Weiyong, for their support, for giving me the opportunity to receive the best education, and for giving me the confidence to pursue this path. I would like to thank my cousins, Yicheng and Zhicong, for being my great companions in childhood and for always serving as role models.

Finally, I would like to thank my wife, Ran, who has unwaveringly supported and encouraged me throughout my higher education. Although we have often had to travel long distances and have pursued different professional paths, you inspire me at every moment. I am fortunate to have you by my side. Your presence is woven into every part of this work, and I thank you for being there in my life.

I believe personal growth is a continuous and constructive process, and everyone I have encountered has played a role in shaping who I am. To the many people I was not able to acknowledge explicitly, including my previous advisors, colleagues, and friends, thank you for being part of this journey.

Nomenclature

AOA	angle of attack
API	application programming interface
ASO	aerodynamic shape optimization
CAD	computer-aided design
CFD	computational fluid dynamics
CG	center of gravity
CPU	central processing unit
DNN	deep neural network
DOE	design of experiments
DV	design variable
GNN	graph neural network
GPR	Gaussian Process Regression
GPU	graphics processing unit
HF	high-fidelity
HSCT	High Speed Supersonic Transport
LHS	Latin Hypercube Sampling
LF	low-fidelity

MAC mean aerodynamic chord

MDO multidisciplinary design optimization

MF multi-fidelity

ML machine learning

MSE mean-square error

NRMSE normalized root mean square error

PCA principal component analysis

PDE partial differential equation

POD proper orthogonal decomposition

PSO particle swarm optimization

QOI quantities of interest

RANS Reynolds-averaged Navier-Stokes

RMSE root mean square error

ROM reduced order modeling

RSAC reference supersonic airplane concept

SCALOS Supersonic Configurations at Low Speeds

SST supersonic transport

VLM vortex lattice method

Contents

Abstract **iv**

Acknowledgments **vi**

Nomenclature **ix**

1 Introduction **1**

1.1 Aircraft Design Process 2

1.1.1 Conceptual Design 3

1.1.2 Preliminary Design 5

1.1.3 Detailed Design 6

1.2 Supersonic Transport (SST) Development 7

1.2.1 History of SST Development 7

1.2.2 SST Conceptual Design 8

1.2.3 Design Challenges 10

1.3 Fidelity Limitations in Conceptual Design 13

1.4 Multi-Fidelity Methods 16

1.4.1 Model Representation 16

1.4.2 Types of Low-Fidelity Models 17

1.4.3 Model Management Strategies 18

1.4.4 The Need for Multi-Fidelity Modeling in Engineering Design 20

1.4.5 Relevant Multi-Fidelity Methods for SST Design 21

1.5 Contributions 25

1.6 Dissertation Structure 27

2 Multi-Fidelity Aerodynamic Modeling **28**
2.1 High-Fidelity Solver: Reynolds-averaged Navier-Stokes 29
2.2 Low-Fidelity Solver: Vortex Lattice Method 31
2.3 Multi-Fidelity Workflow 33

3 Surrogate Modeling **35**
3.1 Machine Learning Architecture 35
3.1.1 Input Encoders 38
3.1.2 Convolutional Block 39
3.1.3 Output Decoder 41
3.1.4 Skip Connections 41
3.2 Training 42
3.2.1 Multi-Phases Training 42
3.2.2 Loss Function 42
3.2.3 Optimizer and Training Scheme 44
3.2.4 Hyper-parameter Optimization 45
3.3 Dataset 46
3.3.1 Dataset Preparation 47
3.3.2 Node and Edge Features 48
3.4 Results 50
3.4.1 Local Loading Coefficient (ΔC_p) Predictions 50
3.4.2 Aerodynamic Coefficients Predictions 55
3.5 Prediction Explanations 63
3.6 Discussion 68

4 The Multi-Fidelity Aerodynamics Dataset **70**
4.1 Geometry Definition 70
4.2 Design Space 72
4.3 Design of Experiments 72
4.3.1 Sampling Methods 72
4.3.2 Sampling Results 75
4.4 Numerical Simulation 78
4.4.1 CFD Meshing 78
4.4.2 CFD Solver Settings 81

4.4.3 CFD Results and Discussion . . . 82
4.4.4 VLM Settings . . . 83
4.5 Data Availability . . . 83
4.6 Discussion . . . 83

5 Scaling **85**
5.1 Scaling Behavior under Data and Model Variation . . . 85
5.2 Methods . . . 86
5.2.1 Data . . . 86
5.2.2 Model . . . 88
5.2.3 Training . . . 90
5.3 Results . . . 92
5.3.1 Scaling Behavior under Constrained Computation . . . 92
5.3.2 Scaling Behavior under Unconstrained Computation . . . 97
5.4 Discussion . . . 103

6 Application to Conceptual Design Optimization **106**
6.1 Problem Definition . . . 106
6.2 Optimizer . . . 108
6.3 Dataset . . . 111
6.4 Surrogate Model Results . . . 112
6.4.1 Training Geometries . . . 112
6.4.2 Test Geometries . . . 118
6.5 Planform Optimization Results . . . 122
6.5.1 Optimization Results . . . 122
6.5.2 Validation . . . 125
6.6 Discussion . . . 130

7 Conclusion **132**
7.1 Thesis Contribution . . . 132
7.2 Future Work . . . 135

A Dataset "double-delta-aero" Geometries **139**

List of Tables

1.1 Summary of surrogate modeling approaches for aerodynamic design and their suitability with respect to the desired properties listed in Section 1.3. 22

2.1 Summary of multi-fidelity analysis tools. 29

3.1 List of the hyper-parameters studied, along with their value range and definitions. 45
3.2 List of the nodal and edge features used by the `VortexNet`, along with their dimension and intended purposes. 49
3.3 Test set RMSEs between ΔC_p^{pred} and ΔC_p^{HF}, grouped by wing geometries. . 53
3.4 Unseen geometries RMSEs 54

4.1 List of design variables and their corresponding upper and lower bounds. . . 72
4.2 Dataset sizes at six levels. 75
4.3 Mesh statistics across refinement levels. 80

5.1 Model sizes at four levels. 88
5.2 Additional training hyper-parameters after optimization for all models. . . . 89

6.1 Mass properties of system components used for center of gravity computation. 107
6.2 Flight conditions at the supersonic and subsonic design points. 108
6.3 List of design variables and their corresponding upper and lower bounds for the optimization problem. 109
6.4 Comparison of VLM and GNN normalized root mean square error (NRMSE) for C_L and C_M across all free-stream conditions for geometries within the training geometries set. 117
6.5 Averaged NRMSE for the training geometries set, summarized by methods. 117

6.6 Comparison of VLM and GNN NRMSE for C_L and C_M for geometries within the hold-out test set. 121
6.7 Averaged NRMSE for the test geometries set, summarized by methods. . . 121
6.8 Comparison of design variables for VLM-based and GNN-guided optimized designs. 123
6.9 Comparison of performance metrics for VLM-based and GNN-guided optimized designs. 126
6.10 CFD validation of K_n for intermediate optimal configurations obtained across VLM-based and GNN-guided optimization iterations. 130

A.1 Design parameter table for the `double-delta-aero` dataset. 139
A.2 Design parameter table for the hold-out test set in the `double-delta-aero` dataset. 148
A.3 Association between geometries and the corresponding dataset levels. 149

List of Figures

1.1 Phases of aircraft design process. 4
1.2 Recreated version of Fig. 10 from Wilde et al. [148] showing the Concorde wing components. 12

2.1 Block diagram of the surrogate model training and the GNN-guided aerodynamic shape optimization workflow. Each block represents an intermediate stage in the design workflow, with "online" stages colored gold and "offline" stages colored blue, while the tools used for each specific process are annotated alongside the arrows. 34

3.1 Schematic of the proposed `VortexNet` GNN architecture. Colors indicate different network blocks. A black arrow represents direct message passing between blocks, and a blue arrow denotes a skip connection to the receiving block. The figure also presents snapshots of the graph at each computational step, showing nodes, edges, and their associated feature arrays. 38
3.2 The HF surface pressure at the lower (figure (a)) and upper (figure (b)) surfaces obtained from CFD for a wing with 55° leading edge sweep and a root airfoil profile of NACA 2416 at AOA= 16.4°, Ma = 0.47, and Re = 9.78×10^6. 51
3.3 Lattice panels and local loading coefficients ΔC_p are visualized using color maps ranging from $\Delta C_p = 0$ to $\Delta C_p = 1.5$, with (a) VLM ΔC_p^{LF}, also serving as the LF input, (b) `VortexNet` $\Delta C_p^{\mathrm{pred}}$, and (c) the reference HF ΔC_p^{HF} obtained by projecting the CFD surface pressure onto the lattice panels. 52

3.4 Lattice panels and local loading coefficients ΔC_p are visualized using color maps ranging from $\Delta C_p = 0$ to $\Delta C_p = 1.5$ for four new geometries. From top to bottom: 60° NACA 0013, 60° NACA 3416, 70° NACA 0013, and 70° NACA 3416. From left to right: ΔC_p^{LF} obtained from the VLM, `VortexNet` ΔC_p^{pred} predictions, and the projected CFD ΔC_p^{HF}. 56
3.5 Relative errors for C_L and C_M from the VLM against the CFD reference. . 58
3.6 Relative errors for C_L and C_M from `VortexNet` against the CFD reference. The region to the right of the dash-dotted line (20°) denotes extrapolation beyond the training range. 58
3.7 NRMSEs for C_L and C_M (from left to right, column-wise), comparing QOIs computed by VLM, integrated QOIs from the surrogate model (ΔC_p^{pred}), and QOIs computed from ΔC_p^{HF}, all against the CFD reference. The results are grouped by geometries, with NRMSE values represented by color. 60
3.8 Relative errors for C_L and C_M from the VLM against the CFD reference for four unseen geometries. 61
3.9 Relative errors for C_L and C_M from `VortexNet` against the CFD reference for four unseen geometries. The region to the right of the dash-dotted line (20°) denotes extrapolation beyond the training range. 62
3.10 NRMSEs for C_L and C_M (from left to right, column-wise), comparing QOIs computed by the VLM, integrated QOIs from the surrogate model (ΔC_p^{pred}), and QOIs computed from ΔC_p^{HF}, all against the CFD reference. 63
3.11 Association between nodal feature clusters extracted from the first Convolutional Block and their corresponding spatial locations on the wing surface. . 64
3.12 Ablation study results illustrating ablation error ($\Delta C_p^{\text{mod}} - \Delta C_p^{\text{pred}}$) for node features within each cluster (columns), and the projected latent space nodal features plotted against the principal axes, with scatter dot colors indicating the corresponding clusters of the nodes. The rows represent the specific layers where the ablation study is conducted. 66

4.1 Geometry definition and its design variables. 71
4.2 Illustration of vehicle configurations sampled from the design space. Training samples from the Level 6 set are shown in blue, while samples from the holdout test set are shown in red. 76

4.3 Design space coverage and design variable distributions of the dataset. . . . 77
4.4 Surface meshes of a representative geometry used for the mesh sizing study. 79
4.5 Mesh sizing study: convergence of C_D and C_M with respect to mesh size. . 81
4.6 Pressure coefficient (C_P) visualizations and Q-criterion iso-surfaces at $Q = 100$ for different vehicle configurations, illustrating variations in geometry and flow states. 82

5.1 Scaling behavior of test mean-square error (MSE) as a function of the training set size D for models of different sizes, where D denotes the number of distinct geometry-flow snapshots represented as graphs in the dataset. 93
5.2 Test MSE scaling with D for four models arranged from (a) to (d): Mini, Small, Medium, and Large, along with their log-log linear fit for the power-law region. 95
5.3 Data scaling exponents (α_D) versus model sizes (N). 96
5.4 Test loss history versus accumulated compute for all models and dataset sizes. 98
5.5 Train loss history versus accumulated compute for all models and dataset sizes. 100
5.6 Best test MSE as a function of dataset size and model size: (a) data scaling and (b) model scaling. 101
5.7 Best test loss versus normalized compute, where normalized compute is defined as the total training FLOPs divided by the FLOPs per optimization step, equivalent to the number of weight-update steps required to reach the minimum test loss. 102

6.1 Lattice panels and local loading coefficient (ΔC_p) for various configurations and angle of attacks (AOAs). Left to right: VLM, GNN, and reference CFD results. 114
6.2 quantities of interests (QOIs) and their derivatives sweep for selected configurations from the training geometries set. AOA from 9° to 21°. 115
6.3 Test set ΔC_ps. Left to right: VLM, GNN, and reference CFD results. . . . 119
6.4 QOIs and their derivatives sweep for configurations from the test geometries set. AOA from 11° to 19°. 120
6.5 Evolution of design variables during optimization. 124
6.6 Evolution of GNN-guided wing configurations, visualized by the camber plane in the x-y plane with the z-coordinate indicated by color. 125

6.7 Evolution of VLM-guided wing configurations, visualized by the camber plane in the x-y plane with the z-coordinate indicated by color. 125
6.8 Optimal wing design ΔC_p distributions: (a) VLM-based configuration evaluated using VLM, (b) GNN-guided configuration evaluated using the GNN surrogate model, (c) VLM-based configuration evaluated using CFD, and (d) GNN-guided configuration evaluated using CFD. 127
6.9 Comparison of QOIs for (a) C_L and (b) C_M for optimal configurations. . . 128
6.10 Comparison of static margin K_n for optimal configurations. 129

A.1 Base airfoil used, obtained from [69]. 149

Chapter 1

Introduction

Aircraft designers have long navigated a fundamental tradeoff between computational cost and physical accuracy. In conceptual design, engineers explore large design spaces to find configurations that satisfy aerodynamic performance, stability, and constraints under strict time and budget limitations. Low-fidelity tools such as vortex lattice methods enable rapid exploration but neglect critical nonlinear flow phenomena, including viscous effects, separation, and vortex-induced aerodynamics [84]. High-fidelity computational fluid dynamics, in contrast, captures these effects with significantly greater accuracy but at computational costs that prohibits broad exploration [114]. As a result, conceptual design workflows traditionally rely on simplified physics in early phases and defer high-fidelity analyses to later stages [106, 137]. This creates high uncertainty in early decisions, as inaccuracies from low-fidelity predictions can propagate through the design cycle and lead to infeasible solutions.

Multi-fidelity methods offer a promising path by combining information from different analyses to improve predictive accuracy without incurring the full cost of high-fidelity simulations [97]. However, most existing surrogate approaches in aerodynamic design focus on correcting a limited set of predefined quantities of interest, such as lift or drag coefficients, and do not reconstruct the underlying flow field. This restriction limits flexibility, interpretability, and integration with multidisciplinary optimization frameworks. Recently, machine learning approaches are increasingly used for surrogate modeling, particularly due to their capability to reconstruct high-dimensional variables, such as flow fields. Despite the increased interest, systematic frameworks that integrate field-level multi-fidelity learning into conceptual design workflows and studies that characterize their data and computational requirements remain limited. This gap inspires this dissertation, which presents the

development of a graph neural network-based multi-fidelity surrogate capable of predicting quasi-high-fidelity pressure fields from low-fidelity representations.

This chapter discusses the motivation for this research in greater detail, including an overview of the aircraft design process and supersonic transport systems. It examines the challenges in supersonic transport conceptual design, which serves as the primary test case for this dissertation, and reviews the limitations of current multi-fidelity and conceptual design methodologies. Furthermore, it provides a justification for using graph-based learning for field-aware surrogate modeling. The chapter concludes with a detailed statement of the research objectives and contributions and an overview of the dissertation's organization.

1.1 Aircraft Design Process

Aircraft design is inherently multi-disciplinary, strongly-coupled, constraint-driven, and multi-objective. As highlighted by Sadraey, it requires the participation of at least six disciplines, including flight dynamics, aerodynamics, propulsion, aero-structure, management, and engineering design [108]. These disciplines interact among others, such that changes in one domain feedback into others. As a result, analysis and optimization must treat the aircraft as a coupled system. Meanwhile, aircraft, as the end product from design, due to its high-cost and high-risk nature, is under tight constraints from regulations, safety, mission requirements, and environmental impact. These factors combined define aircraft design to be multi-objective: A well-designed aircraft shows balance among performance, cost, safety, and maintainability [108].

These design requirements necessitate a structured framework to coordinate the interacting disciplines and manage risk across the product life cycle [14, 24, 106, 108]. Hence, aircraft design is best understood within a systems engineering framework that views development as a structured life-cycle process. Blanchard and Fabrycky describe this process as beginning with identification of a need and proceeding through requirements definition, functional analysis and allocation, design synthesis, evaluation, validation, operation, and eventual disposal [14]. Within this broader life cycle, the design and development effort is traditionally organized into three progressively refined phases: conceptual design, preliminary design, and detailed design. This phased decomposition reflects increasing information maturity and growing commitment of technical and financial resources. It also mirrors the

hierarchical nature of complex systems: conceptual design addresses the aircraft at the system level, preliminary design resolves subsystem architecture and interactions, and detailed design focuses on component-level implementation. Each phase concludes with a major decision gate that determines whether the program advances to the next stage, thereby embedding engineering judgment within within the larger program management and life-cycle cost framework [14]. Figure 1.1 illustrates this design process, with the major design phases highlighted in boxes as well as the preceding and succeeding program cycle, which includes Aircraft Design Requirements definition and Test Article Fabrication.

1.1.1 Conceptual Design

Within this phased cycle, following the definition of Aircraft Design Requirements, conceptual design is typically considered as the "formal" start of an new or derivative aircraft program [129]. Conceptual design translates mission needs into a feasible aircraft configuration. Mission requirements define the design problem and typically include payload capacity, range and endurance, cruise and maximum speeds, cruise altitude, takeoff and landing distance, stability and control constraints, operational considerations, acquisition cost, and compliance with regulatory standards such as the Federal Aviation Regulations [8, 24]. These requirements collectively define the design space and establish the criteria by which candidate configurations are evaluated.

The central question of conceptual design is whether an affordable aircraft can be built that satisfies the stated requirements [106]. If not, the requirements themselves must be revisited. Thus, conceptual design is inherently iterative: requirements guide configuration development, and emerging configuration insights may prompt refinement of those requirements. Within this phase, designers explore a wide range of configuration concepts and conduct trade studies across aerodynamic performance, weight, propulsion integration, stability characteristics, cost implications, and operational feasibility. The configuration arrangement typically includes wing and tail geometry, fuselage shape, internal layout of crew and payload, engine installation, landing gear placement, and major structural features. Multiple candidate concepts are often developed in parallel and evaluated against mission objectives before converging toward a single balanced configuration.

A key characteristic of the conceptual design phase is its fluidity [106]. As new performance information becomes available from trade studies, geometry and sizing parameters update continuously. Such inner-loop iterative optimization is highlighted in Figure 1.1 by

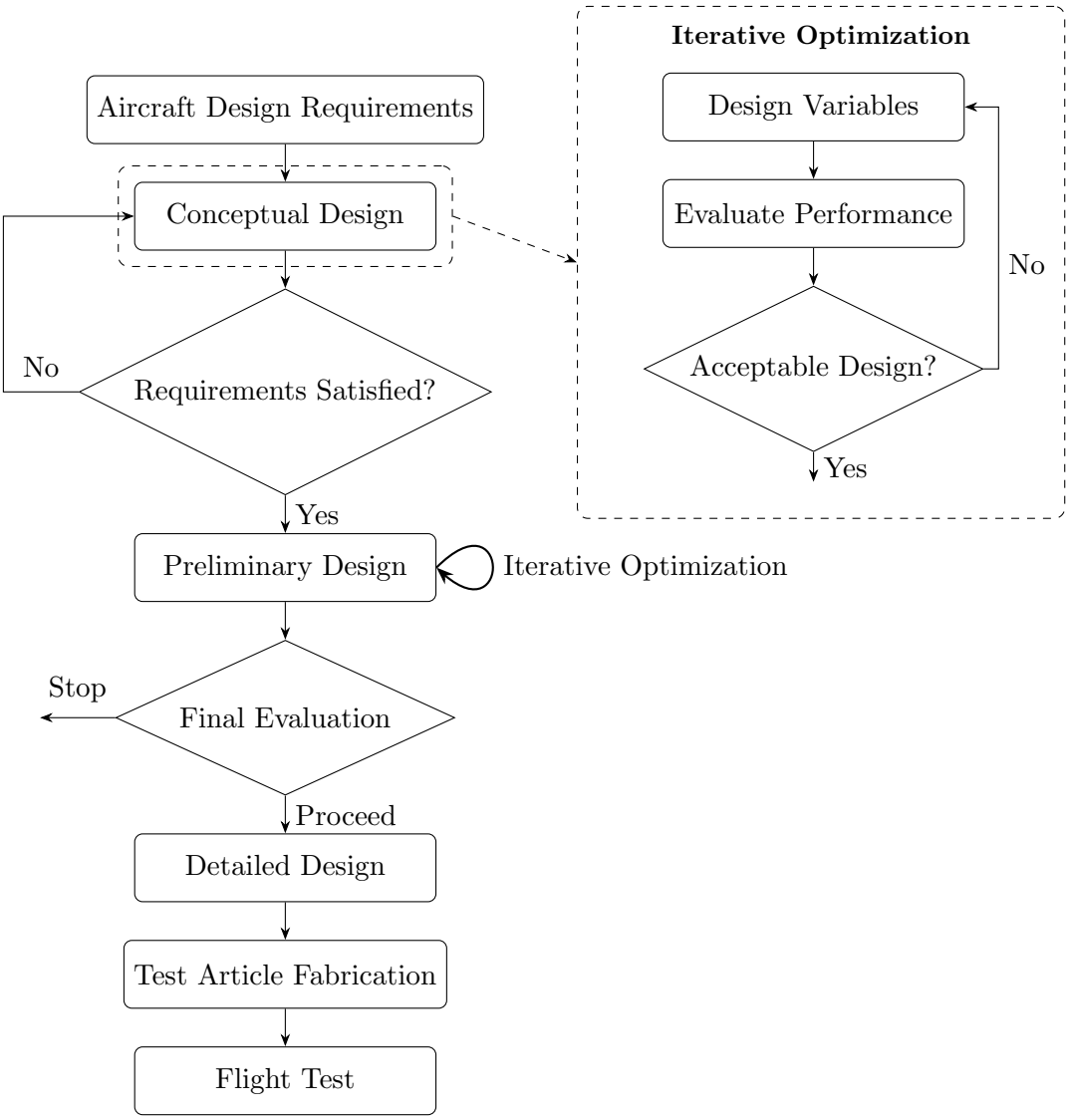


Figure 1.1: Phases of aircraft design process.

the side of the "Conceptual Design" box, marked as "Iterative Optimization." Because configuration changes frequently, the analysis environment must support rapid iteration [106]. high-fidelity (HF) geometrical and physical models, such as computer-aided design (CAD) and CFD-based models, provide higher modeling fidelity but are often too computationally expensive and workflow-intensive to set up. In addition, these tools are often too rigid for rapid design iterations. Consequently, conceptual design widely employs low-fidelity (LF) methods, such as VLM, panel methods, empirical sizing laws, historical benchmarks, and

semi-analytical correlations [24, 106]. These tool chains enable rapid exploration of large design spaces and maintain robustness under frequent geometric modifications. In addition, success in the conceptual design phase relies heavily on the past experience of domain experts; consequently, this phase, as quoted by Sadraey, "is typically led by the most experienced engineers within an organization" [108].

The primary objective of conceptual design is to narrow the design space to one or a few baseline configurations. These selected configurations then serve as the foundation for the preliminary design phase, during which only minor configuration changes should be necessary [106]. This objective gives rise to a second key characteristic of conceptual design: the "locked-in" effect.

As a program progresses into later phases, increasingly detailed product-specific knowledge is generated, resulting in greater confidence that the design satisfies the aircraft design requirements. However, this maturation is accompanied by accumulating development costs, making design changes progressively more expensive and difficult to implement. Conceptual design represents only a small fraction of the overall vehicle development program, accounting for less than 10% of total cost according to Spitz et al., who studied multiple aircraft programs in the 1990s [129]. Despite this relatively small investment, decisions made during this phase are estimated to determine 70–90% of the aircraft's total life-cycle cost [8, 108]. Conceptual architectural choices therefore exert a disproportionate influence over the program outcome.

1.1.2 Preliminary Design

The objective of preliminary design shift from conceptual design phase's broad exploration to detailed subsystem refinement and risk reduction [106]. At this stage, disciplinary experts join the design team to address major subsystems, including structures, landing gear, propulsion integration, and control systems. Aerodynamic layout is refined, and structural components are developed with increased fidelity. Sub-components such as spars, ribs, and control surfaces are designed, sized and evaluated. The vehicle outer mold line (OML) is defined using higher fidelity tool, such as CAD, establishing the aerodynamic geometry that will carry forward into production.

Richer system-to-system interactions are studied, including wing–fuselage interference effects. Aerodynamics is evaluated with greater rigor, particularly with respect to vehicle stability and control surface effectiveness. Because higher-fidelity analysis is required to

accurately capture these interactions, advanced computational models dominate evaluation during this phase. CFD is employed to generate refined aerodynamic predictions, provide nonlinear aerodynamic data for handling qualities and stability analysis, and supply aerodynamic loads and aeroelastic inputs for structural assessment [30]. In addition, CFD plays a critical role in aerodynamic shape optimization (ASO), where local optimization is performed for wing [71, 113], tail [113, 115], and engine-component geometries [117]. Finite element method (FEM) is introduced to analyze structural dynamics, including buckling, fatigue, aeroelastic effects, and flutter [24]. Extensive scaled-model wind tunnel testing is also conducted to further reduce uncertainty in vehicle performance and increase confidence that the design satisfies the aircraft design requirements.

Preliminary design may span several months to multiple years depending on program complexity [106], and typically multiple design–assessment iterations are required, as indicated by the "Iterative Optimization" arrow in Figure 1.1. Its primary objective is to reduce technical uncertainty sufficiently to justify transition to full-scale development [24, 106].

1.1.3 Detailed Design

Upon completion of preliminary design, the vehicle is comprehensively evaluated for compliance with the aircraft design requirements, technical feasibility, cost realism, and certification pathway viability. If sufficient confidence is achieved, the configuration is formally frozen and the program transitions to detailed design [24]. In this final development phase, emphasis shifts from analysis-driven refinement to manufacturability and production readiness. Every component is finalized; tooling, material selection, sheet metal forming or composite layup procedures, fastener design, and assembly sequences are defined; and complete production drawings are generated [24]. The vehicle transitions from an analytical construct to a realizable physical system.

Across these phases, the aircraft design process progresses from flexibility and exploration to commitment. Conceptual design prioritizes rapid trade studies and system-level feasibility. Preliminary design emphasizes subsystem integration and higher-fidelity validation. Detailed design undertakes manufacturing implementation. This progression reflects both system engineering philosophy and program management economics: the ability to influence performance and cost is greatest in the early phases, while analytical fidelity and resource commitment increase over time.

1.2 Supersonic Transport (SST) Development

1.2.1 History of SST Development

A supersonic transport (SST) is a civilian supersonic aircraft designed to transport passengers at speeds greater than the speed of sound. Its development, viewed from a chronological perspective, can be divided into three phases. The first wave of development occurred in the 1960s. Three major aircraft development programs aimed at operating at Mach 2 or higher emerged during this period: the Boeing 2707, the Concorde, and the TU-144 programs. These early programs demonstrated the technical feasibility of SST, but were eventually phased out due to a variety of economic, environmental, and political challenges [137]. Interest in civil supersonic aviation declined in the 1970s due to both economic and environmental concerns. At the same time, many countries banned supersonic flight over land because of sonic boom, severely limiting market potential [137]. As discussed by Torenbeek, these programs represented a "great technical achievement but economic disasters."

In the 1990s, market projections for the 1995–2015 period in the North-Mid Pacific and Far East aviation markets forecast a six- to seven-fold increase in passenger miles relative to 1971–1989 [137]. Stimulated by these projections and informed by lessons learned from the first wave of SST programs, a second generation of SST research initiatives emerged. These aircraft typically targeted ranges exceeding 10,000 kilometers to serve trans-Pacific routes, with passenger capacities above 250 seats [19]. Notable programs during this period include the High-Speed Civil Transport (HSCT) program led by NASA [19, 137, 144], and the European Supersonic Commercial Transport (ESCT) initiative led by Aerospatiale, British Aerospace, and Daimler-Benz Aerospace Airbus [19, 37, 137]. These programs were primarily research-oriented and did not progress to prototype development. It was ultimately concluded that the available technology was insufficient to achieve both economic viability and compliance with environmental requirements, particularly regarding sonic boom mitigation [137].

In recent years, there has been renewed interest in the development of commercial supersonic aircraft [9]. This resurgence is driven by both market demand and technological advances that may allow new designs to overcome the limitations faced by earlier SST programs. The primary motivation remains the ability to significantly reduce long-distance travel time: higher speeds reduce block time, differentiate product offering for premium travelers, and allow airliners to complete more flights within a given time period [137].

Modern studies also suggest that SSTs operating at these moderate cruise speeds around Mach 1.6–2.0 may provide a better balance between performance and economic feasibility [137]. Multiple SST programs surged in this phase, including Boom Overture, Aerion AS2, and Spike S-512.

Supporting these renewed commercial efforts, significant research over the last two decades has focused on SST concept development, sonic-boom mitigation, and advanced numerical methods [63, 71, 86, 91, 115, 144]. Among these efforts, the Supersonic Configurations at Low Speeds (SCALOS) project was initiated to address key aerodynamic challenges in next-generation SST design, particularly the low-speed performance of slender supersonic configurations shaped for favorable trade-offs between supersonic efficiency, sonic-boom mitigation, and low-speed handling constraints [69, 90, 135]. Throughout SCALOS, extensive work has produced aerodynamic and aeroelastic MDO tools, flight-control co-design methodologies, wind-tunnel testing techniques, experimental datasets for diverse configurations, noise-prediction simulations, surrogate-modeling approaches, and the development of the reference supersonic airplane concept (RSAC) [39, 81, 90, 113, 114, 116, 121, 124, 134, 136, 147].

1.2.2 SST Conceptual Design

Supersonic aircraft design differs fundamentally from the design of conventional subsonic transports due to the dominant role of compressibility effects, shock waves, and wave drag in determining aerodynamic performance. In supersonic flight, aerodynamic performance is dominated by shock formation and the associated pressure drag generated [106] by both the aircraft volume (body and wing wave drag [137]) and the lift (wave drag due to lift and vortex-induced drag [137]). As a result, supersonic aircraft typically achieve substantially lower aerodynamic efficiency than subsonic transports. While modern subsonic transports commonly operate with cruise lift-to-drag (L/D) ratios on the order of 18–20, supersonic aircraft typically achieve values in the range of approximately 4–7 [80, 106]. In addition to reduced cruise L/D ratios, supersonic aircraft experience significant shifts in aerodynamic center location with increasing Mach number, which introduces important stability and trim challenges [69, 80, 106, 144]. Furthermore, unlike conventional transports that operate primarily in the subsonic regime, SST aircraft must perform across a wide flight envelope that includes low-speed takeoff and landing, transonic acceleration, and sustained supersonic cruise [80, 137]. These aerodynamic characteristics impose stringent constraints on aircraft

configuration design.

The conceptual design scope of SST aircraft thus encompasses multiple tightly coupled configuration elements, including fuselage shaping, wing planform design, fuselage–wing relationship, airfoil selection, propulsion integration, and empennage configuration [80, 148]. These components must be designed together to minimize wave drag, maintain stability across the flight envelope, and ensure favorable aerodynamic interactions between aircraft elements. Within this context, planform development is one of the most critical tasks. Planform development involves definition of several geometric parameters that collectively describes the wing configuration and lifting surface arrangements [80, 148]. These parameters include the relative wing location to fuselage, wing aspect ratio, wing types and planform shape, sweep, twist, dihedral, and chord distribution, and more broadly with airfoil selection, spanwise flow control, and additional flow control surface design [59].

Among these parameters, planform shape, including wing type, leading-edge sweep, and aspect ratio, together with leading-edge camber, plays a particularly central role in determining aerodynamic performance [148]. Supersonic wings are typically highly swept and thin in order to reduce wave drag and control shock formation. As a result, delta, arrow, trapezoidal, and compound planforms are commonly used for SST configurations. The theoretical basis for these planform choices is discussed in detail in Chapter 6 of Torenbeek's supersonic aircraft textbook [137]. The key takeaway is that increasing leading-edge sweep can reduce wave drag and drag due to lift by placing the wing leading edge behind the Mach cone, thereby creating a subsonic leading edge and enabling favorable leading-edge suction effects. This aerodynamic consideration naturally leads to arrow- or delta-like planforms, which are frequently paired with thin airfoils and sharp leading edges in SST design [106].

However, at higher cruise Mach numbers, it is often infeasible to maintain sweep angles everywhere larger than the Mach cone angle, causing portions of the wing, particularly the outboard sections, to experience locally supersonic leading edges [106]. In such situations, the ability to generate and maintain leading-edge suction becomes important, making leading-edge camber a critical design feature [106, 137, 148]. In addition, at low speeds, sharp leading edges can promote the formation of leading-edge vortices, which enhance lift at high angles of attack and improve maximum lift capability [106]. However, the development, maintenance, and effective use of such vortex systems are also strongly planform dependent [148], highlighting the strong sensitivity of SST aerodynamics to planform geometry.

Even within a reduced scope of planform design, focusing only on planform shape and leading-edge camber, aerodynamic performance exhibits highly nonlinear effects and strong design variables (DVs) couplings, making the balance of aerodynamic performance across multiple flight regimes difficult. Historically, SST conceptual design has relied on aerodynamic build-up methods that combine simplified theoretical models with experimental observations and empirical corrections. These approaches estimate aerodynamic forces and moments across the flight envelope by combining analytical tools and aerodynamic databases [80, 138, 148]. For example, during the Concorde program, slender-body theory, linear aerodynamic models, and experimental observations were combined to guide wing geometry selection [148]. Lift was approximated as linear with incidence using analytical lift-curve slopes dependent on aspect ratio, sweep, and Mach number, while drag was decomposed into zero-lift drag and drag due to lift, with empirical corrections from experimental databases.

More recent SST conceptual design studies employ similar aerodynamic build-up approaches using integrated computational environments. For example, Welge et al. [144] used a suite of aerodynamic analysis tools to estimate aerodynamic performance across the flight envelope. Similarly, supersonic drag predictions were obtained using linear aerodynamic theory augmented by database-based corrections, including tools such as TEA80 for wave drag prediction and A389 for supersonic lift-dependent drag estimation, while low-speed aerodynamic characteristics were evaluated using vortex lattice methods supplemented by wind-tunnel database corrections [144]. Similar database-driven approaches have also been used in more recent conceptual design studies on the development of RSAC configuration [69]. These methods illustrate how aerodynamic build-up models have historically guided the development and evaluation of SST wing planform during conceptual design.

1.2.3 Design Challenges

Based on the preceding discussion, the conceptual aerodynamic design of SST configurations presents several fundamental challenges. These challenges arise from three primary sources: conflicting aerodynamic objectives inherent to SST configurations, the gradual locking-in of design decisions as the development process progresses, and the difficulty of accurately modeling nonlinear aerodynamics during conceptual design.

1. Conflicting aerodynamic objectives in SST configuration design. As discussed in Section 1.2.2, the aerodynamic characteristics of SST aircraft are strongly governed by wing planform geometry. However, these geometric parameters simultaneously influence multiple aerodynamic objectives, many of which are inherently conflicting. For example, reducing leading-edge camber (droop) across the wing span can improve supersonic drag performance but may worsen subsonic aerodynamic characteristics and increase the lift-curve slope [148]. Similarly, increasing the wing span may reduce subsonic drag but increases supersonic drag and potentially degrades stability characteristics [148]. Adjustments to sweep distribution may improve low-speed stability, but excessive reductions in outboard sweep can lead to significant penalties in supersonic drag [148]. These trade-offs illustrate that small modifications in planform geometry can significantly alter aerodynamic performance across different flight regimes.

Because these aerodynamic objectives are tightly coupled, optimizing a single performance metric often leads to deterioration in other aspects of aircraft performance. As a result, SST conceptual design requires balancing multiple competing objectives. This characteristic motivates the need to view the configuration design problem holistically, integrating aerodynamic performance, stability, and other disciplinary considerations within a MDO framework.

2. The "locked-in" effect and performance compromise. Another challenge arises from the progressive reduction in design flexibility as aircraft development advances from conceptual design toward preliminary and detailed design, a "locked-in" effect as introduced in Section 1.1.1. Early configuration decisions strongly constrain the design space available in later stages of development. During the Concorde program, for example, once the wing planform had been established, subsequent aerodynamic refinements were limited to increasingly localized regions of the wing geometry: Not compromising the central wing box position, the designers limited their aerodynamic optimization on the wing apex, wing tip, and leading edge regions [148]. These parts are illustrated by shaded shadows in Fig. 1.2, to show the extent of design space reduction. It shall be clear that such localized adjustments are unlikely to fundamentally alter the global aerodynamic limitations introduced by the initial planform selection. Consequently, design decisions made during conceptual design can effectively lock in key aerodynamic characteristics, making it difficult to recover globally optimal configurations after higher-fidelity aerodynamic information (such as CFD

and wind tunnel experiments) becomes available in later design stages.

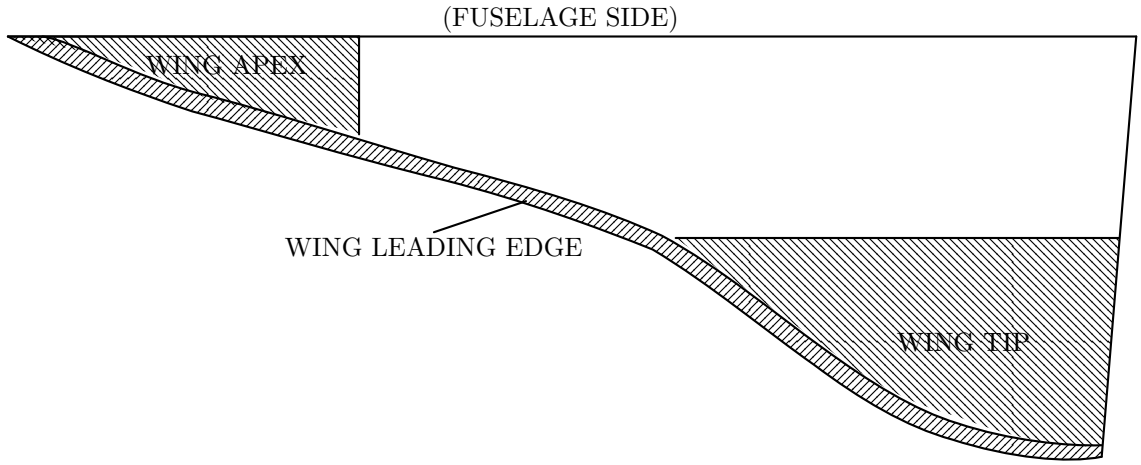


Figure 1.2: Recreated version of Fig. 10 from Wilde et al. [148] showing the Concorde wing components.

3. Modeling limitations and nonlinear aerodynamic effects. A further difficulty arises from the limitations of aerodynamic modeling during conceptual design. SST aircraft frequently operate under conditions where nonlinear aerodynamic phenomena dominates. In particular, low-speed operation of slender delta-wing configurations typically involves high angles of attack, where aerodynamic forces are dominated by complex flow mechanisms such as vortex lift, vortex–boundary layer interactions, and flow separation [135]. These nonlinear flow phenomena strongly influence aircraft stability and control characteristics.

In practice, conceptual design methods often rely on simplified theoretical models or historical aerodynamic databases, which early SST design studies noted can be too idealized and imprecise to fully capture the underlying flow physics [148]. When new configurations fall outside the range of existing aerodynamic databases, designers may be forced to rely on simplified linear models that provide insufficient fidelity. Similar challenges have been observed in modern SST conceptual design programs. For example, Livne et al. [69] report that potential-flow solvers used during the SCALOS program frequently produced inconsistent aerodynamic predictions, requiring systematic comparisons against theoretical solutions and Reynolds-averaged Navier-Stokes (RANS) CFD simulations to correct aerodynamic build-up models. These limitations highlight the difficulty of accurately evaluating

aerodynamic performance and stability characteristics of SST configurations using conventional conceptual design tools.

1.3 Fidelity Limitations in Conceptual Design

As discussed in Section 1.2.2, modern aircraft development increasingly relies on tightly integrated MDO workflows [78, 141]. Conceptual design frameworks such as `SUAVE` [74, 145] integrate multiple disciplinary analyses at multiple fidelity levels. In particular, aerodynamics are commonly computed using the VLM [17, 22, 49, 84, 94]. While detailed formulation is introduced later in Section 2.2, overall the VLM models lifting surfaces using potential-flow theory, enabling rapid estimation of pressure distributions and aerodynamic coefficients while maintaining very low computational cost. Tools like VLM play an essential role in enabling broad exploration of the design space by offering a cost-effective methods.

Despite its utility, the simplifying assumptions underlying the VLM limit its applicability in aerodynamic regimes dominated by nonlinear flow physics. In particular, the potential-flow formulation neglects viscous effects, vortex-boundary-layer interactions, and flow separation, which are critical for accurately predicting aerodynamic behavior at high AOAs. These limitations become especially problematic in the conceptual design of SST. For example, accurate prediction of aerodynamic coefficients and pressure distributions is necessary to enforce low-speed stability and control (S&C) constraints during takeoff-and-landing conditions [90, 113]. However, under high AOA flow conditions encountered in these regimes, LF predictions can deviate significantly from the true aerodynamics. Consequently, incorporating S&C constraints into SST planform development has long been a challenge, requiring the development of new conceptual design methodologies.

Several approaches have been proposed to improve the predictive capability of VLM-based models. One common strategy is to incorporate theoretical or empirical corrections, such as the Polhamus correction [102] for vortex lift modeling and aerodynamic database [13] for engineering level aerodynamics. While such corrections can partially improve aerodynamic predictions, they introduce additional challenges related to generality and applicability. In particular, correction parameters must often be tuned for specific wing geometries and flow conditions in order to produce accurate results [45, 82]. When exploring a large conceptual design space, including configurations with novel planforms or unconventional

lifting-body geometries, validating these correction parameters across different configurations becomes difficult.

Higher-fidelity methods, such as CFD, can model complex flow physics that are difficult to capture with VLM. For example, detached-eddy simulation (DES) has demonstrated the ability to capture vortical flow structures and predict vortex breakdown locations in delta-wing flows [85]. RANS is also widely used in aerodynamic analysis and design [25, 146]. The fidelity and computational cost of RANS simulations lie between DES and VLM, and the appropriateness of using RANS for low-speed SST S&C modeling is supported recent studies by Seraj and Martins [113, 114, 115]. Specifically, Seraj and Martins validated the use of a RANS-based CFD solver to quantify lift (C_L), drag (C_D), and pitching-moment (C_M) coefficients for slender delta-wing configurations [114]. They then formulated an MDO problem in `MACH-Aero` [76] to perform free-form deformation (FFD)–based shape optimization of the wing under both takeoff-and-landing and supersonic (cruise) conditions, subject to longitudinal stability constraints [113]. Separately, Seraj and Martins embedded trim sizing into a drag-minimization optimization of the main wing and control surfaces, highlighting the importance of incorporating stability constraints and aerodynamic interactions between aircraft components [115].

However, directly embedding CFD simulations within conceptual design optimization loops remains prohibitive: the design space, when compounded with the mission state space, leads to an unbearable cost explained in detail in later Section 1.4.4. Furthermore, as noted by Seraj and Martins [115], large shape deformations during aerodynamic optimization can introduce mesh distortions that lead to numerical instability or solver divergence, further limiting the practical design space that can be explored using CFD alone. Finally, the OML of the aircraft is often not fully defined during conceptual design, making repeated high-fidelity analyses even more challenging.

This fundamental trade-off between model fidelity and computational cost creates a persistent fidelity gap in conceptual aircraft design: LF models enable rapid exploration of large design spaces but lack the physical accuracy required for reliable prediction of nonlinear aerodynamics; HF simulations, on the other hand, provide accurate predictions of aerodynamics but are too expensive to be used extensively in conceptual design studies. As a result, designers often lack the ability to perform high-confidence aerodynamic assessments during the early stages of the design process and loss the opportunity to further optimize the vehicle beyond its local minima constrained by the initial concept as discussed

in Section 1.2.3.

To bridge this fidelity gap, *multi-fidelity methods* are often employed [39, 88]. These multi-fidelity models leverage information from both low- and high-fidelity data sources to improve LF solver's information quality while maintaining low evaluation cost. By learning the discrepancy between low- and high-fidelity simulations during an offline training phase, these models can provide rapid predictions with accuracy approaching that of high-fidelity simulations during the online optimization phase. Such class of method is explained in detail in the later Section 1.4. Given the previous discussion on the fidelity gap in SST conceptual design, the author proposes the following capabilities for an effective multi-fidelity model:

1. Generalization across the design space. Conceptual design involves exploration of a broad design space containing diverse aircraft configurations and operating conditions. A useful multi-fidelity model must therefore provide interpolation across this design space while leveraging sparse high-fidelity samples. In particular, the model should improve the predictive capability of the underlying low-fidelity analysis beyond the limited configurations directly observed in the high-fidelity training data.

2. Computational efficiency for online deployment. Due to the many-query (defined in Section 1.4) nature of planform development, the multi-fidelity model must remain computationally inexpensive during online deployment. The evaluation cost of the surrogate should be comparable to or less than that of the underlying low-fidelity solver, ensuring that the analysis tool chain can support the rapid turnaround required in conceptual design.

3. Prediction of field quantities. Many surrogate modeling approaches focus on predicting a predefined set of aerodynamic QOIs. While effective for certain applications, this strategy can limit flexibility in multidisciplinary design optimization. Predicting high-dimensional aerodynamic fields such as surface pressure distributions provides a more general representation of the flow physics. Once the aerodynamic field is available, arbitrary QOIs can be computed *a posteriori*; in addition, field variables enables coupling between aerodynamic analysis and other disciplinary models within the MDO framework.

These considerations highlight the need for multi-fidelity modeling strategies capable of transferring knowledge from sparse high-fidelity simulations to low-cost conceptual design environments while preserving both computational efficiency and physical interpretability. The following section reviews the major classes of multi-fidelity methods developed to address this challenge.

1.4 Multi-Fidelity Methods

1.4.1 Model Representation

A model is a representation of a process used to explain and predict natural phenomena. In engineering applications, these models describe the input–output relationship, such that the model can be denoted as a function $f : \mathcal{Z} \to \mathcal{Y}$ that maps an input $\mathbf{z} \in \mathcal{Z}$ to an output $\mathbf{y} \in \mathcal{Y}$ [97]. The input domain $\mathcal{Z} \subseteq \mathbb{R}^d$ has dimension d and can represent boundary and initial conditions for CFD models, or DVs in an aerodynamic database. The output domain $\mathcal{Y} \subseteq \mathbb{R}^{d'}$ has dimension d' and can represent surface or volume field quantities for CFD models, or the QOIs in an aerodynamic database. These model outputs $\mathcal{Y}$ are often used in *outer-loop* applications such as optimization and uncertainty quantification, where $\mathcal{Y}$ is used to drive the convergence of DVs for optimal design or to estimate statistics of the QOIs [97].

It must be understood that all models include modeling assumptions and are therefore only approximations of the true physical phenomena. Assuming the true physical process from $\mathcal{Z} \to \mathcal{Y}$ is captured by function $\xi(\cdot)$, then the relationship among models f, variable inputs $\mathbf{z}_i$, and observed outputs $\mathbf{y}_i$ can be summarized as:

$$y_i = \xi(\mathbf{z}_i) + e_i = \rho f(\mathbf{z}_i, \theta) + \delta(\mathbf{z}_i) + e_i, \tag{1.1}$$

where e_i is the aleatoric observation error of the ith observation, ρ is an unknown regression parameter, and $\delta(\cdot)$ is a model inadequacy function that is independent of the function output $f(\cdot)$ [52]. The modeling error $\delta(\cdot)$, typically referred to as *epistemic uncertainty*, originates from simplified model assumptions, missing physics, or a lack of knowledge of $\xi(\cdot)$ [126].

In many applications, such as modeling for SST aerodynamics, multiple models are available that estimate the same output quantity with varying levels of approximation quality [97, 123]. It is convenient to define a *high-fidelity model* $f_{\text{HF}} : \mathcal{Z} \to \mathcal{Y}$ as a model that estimates outputs with the accuracy necessary for the assessment tasks [97], thereby having a relatively smaller epistemic error $\delta_{\text{HF}}(\mathbf{x}_i)$. Similarly, we define a *low-fidelity model* f_{LF} as a model that estimates the same outputs with lower accuracy than the HF model, thereby having a potentially larger $\delta_{\text{LF}}(\mathbf{x}_i)$.

Under such a formulation, multi-fidelity methods leverage LF models $f_{\text{LF}}^{(i)}$, where i denotes the class of LF models queried, to achieve modeling and computational speedups, while resorting to HF models $f_{\text{HF}}^{(i)}$ to ensure accuracy and convergence guarantees [97]. Two key components of multi-fidelity methods are thus (1) the selection of LF models that provide *useful approximations* of the HF model, and (2) a *model management strategy* that combines information from multiple sources and establishes accuracy and convergence guarantees.

One can reorganize Equation 1.1 to obtain a more quantitative definition of the "useful approximation" described above. If the aleatoric error e_i is ignored, the difference between HF and LF models, expressed in the Kennedy-O'Hagan formulation [52], can be written in residual form as

$$r(\mathbf{z}) = y_{\text{HF}} - \alpha y_{\text{LF}} = \underbrace{(\rho_{\text{HF}} f_{\text{HF}}(\mathbf{z}) - \alpha \rho_{\text{LF}} f_{\text{LF}}(\mathbf{z}))}_{\text{trend mismatch}} + \underbrace{(\delta_{\text{HF}}(\mathbf{z}) - \alpha \delta_{\text{LF}}(\mathbf{z}))}_{\text{discrepancy mismatch}}, \tag{1.2}$$

where α is a scaling coefficient and $r(\mathbf{z})$ is the residual between HF and scaled LF models under input $\mathbf{z}$.

If α is selected as the least-squares optimal coefficient, $\alpha = \text{Cov}(y_{\text{HF}}, y_{\text{LF}})/\sigma(y_{\text{LF}})$, where $\sigma(\cdot)$ denotes the variance and $\text{Cov}(\cdot)$ denotes the covariance, the variance of the residual becomes $\sigma(r) = \sigma(y_{\text{HF}})(1 - \rho_{\text{HF,LF}}^2)$. The term $\rho_{\text{HF,LF}}$ is the Pearson correlation coefficient between y_{HF} and y_{LF}. Thus, a "useful approximation" is an LF model that captures the dominant variation of the HF response after optimal scaling, such that $\sigma(r)/\sigma(y_{\text{HF}}) = 1 - \rho_{\text{HF,LF}}^2 \ll 1$. In other words, the LF model must be strongly correlated with the HF model.

1.4.2 Types of Low-Fidelity Models

We now turn to the second component of multi-fidelity methods: model management strategy. Peherstorfer et al. [97] and Fernández-Godino [32] classify LF models by their construction into three categories: *simplified models*, *projection-based models*, and *data-fit models*.

1. Simplified Low-Fidelity Models Simplified models are obtained by introducing physical or numerical simplifications to the governing equations of a high-fidelity model using domain expertise. The goal is to retain the dominant physics while reducing computational cost. Common strategies include adopting lower-order physical models, using

coarser spatial or temporal discretizations, neglecting nonlinear terms, or relaxing solver convergence tolerances. In SST aerodynamics evaluation, for example, a common fidelity hierarchy progresses from large eddy simulation (LES), RANS, Euler equations, to VLM models. If on take LES as the HF model, the remainder can be viewed as LF models.

2. Projection-Based Low-Fidelity Models Projection-based models arise from model order reduction techniques in which the governing equations of the HF system are projected onto a low-dimensional subspace that captures the dominant dynamics of the system. The low-dimensional subspace is typically constructed from simulation snapshots, that represent the system's behavior under different conditions. Common methods for generating reduced bases include proper orthogonal decomposition (POD), reduced basis methods, Krylov subspace techniques, dynamic mode decomposition, and balanced truncation [97].

3. Data-Fit Low-Fidelity Models Data-fit models approximate the input–output relationship of the HF model directly from sampled data, treating the HF simulation as a black box. Instead of simplifying the governing equations, these models construct surrogate representations using interpolation or regression techniques. Common data-fit models include polynomial chaos expansions (PCE), radial basis functions (RBF), sparse-grid interpolation, Gaussian process regression, and support vector machines. These models are especially useful when the internal structure of the HF model is inaccessible. Although surrogate models can be viewed as LF representations of HF systems, they also provide a data-assimilation mechanism for the model management strategies discussed next.

1.4.3 Model Management Strategies

The models across fidelities in the previous section can then be combined through a multi-fidelity model management strategy: a process definition regarding how information from different fidelities is utilized during the solution process. In the literature [97, 32], three primary model management strategies are commonly identified: *adaptation*, *fusion*, and *filtering*.

1. Adaptation The low-fidelity model is calibrated to better approximate the high-fidelity model. Typical techniques include additive or multiplicative corrections based on discrepancies between model outputs, Gaussian process-based calibration, and adaptive

reduced-order models whose bases or operators are updated as new high-fidelity data become available [97, 111]. Adaptation can also be implemented through techniques such as space mapping, which iteratively aligns the parameter spaces of low- and high-fidelity models [32]. Mathematically, the iterative calibration can be expressed as:

$$\hat{y}_{\mathrm{HF}} = \rho(\mathbf{z}) y_{\mathrm{LF}}(\mathbf{z}) + \delta(\mathbf{z}), \tag{1.3}$$

where $\hat{y}_{\mathrm{HF}}$ is the approximated HF evaluations, $\mathbf{z}$ are parameter inputs at certain fidelity, $\rho(\mathbf{z})$ represents the multiplicative correction, and $\delta(\mathbf{z})$ represents the additive correction.

Similarly, the space mapping correction can be expressed as:

$$\hat{y}_{\mathrm{HF}} = \mathcal{F}(y_{\mathrm{LF}}, \mathbf{z}_{\mathrm{LF}}) \approx y_{\mathrm{HF}}, \tag{1.4}$$

where $\mathcal{F}(\cdot)$ is a mapping function learned from data, that embodies the model management strategy.

2. Fusion Rather than modifying the low-fidelity model itself, fusion approaches integrate information from both low- and high-fidelity evaluations in a unified framework. Examples include control variate methods for uncertainty propagation and multi-fidelity surrogate models such as co-kriging that integrate data from multiple sources into a single predictive model [77, 89, 97]. More recently, multi-fidelity digital twins have extended this fusion concept by leveraging multi-fidelity (MF) inputs to construct dynamically updated virtual representations of physical systems [103].

3. Filtering In this approach, low-fidelity models act as inexpensive screening tools that identify candidate samples or designs before invoking the expensive high-fidelity model [97]. For example, multi-stage Markov Chain Monte Carlo methods first evaluate candidate samples using a low-fidelity likelihood before computing the high-fidelity likelihood, while multifidelity stochastic collocation and importance sampling methods use low-fidelity models to determine where high-fidelity simulations should be conducted [53, 54].

1.4.4 The Need for Multi-Fidelity Modeling in Engineering Design

The discussion on multi-fidelity methods so far has focused on modeling error, sources of fidelities, and model management strategies. However, the motivation for adopting multi-fidelity methods has not been thoroughly discussed: why conduct LF evaluations when HF information is obtainable? To answer this question, the design-assessment-optimization cycle must be considered within the broader aircraft design process described in Section 1.1. Throughout the design process, iterative optimization (Figure 1.1) involves multiple outer-loop evaluations, meaning that the corresponding assessment model must be executed repeatedly. Such repetitive evaluation is commonly referred to as *many-query* applications [32, 97].

HF models, while providing accurate assessments, may incur substantial cost. It is important to note that this cost does not consist solely of the computational expense of model evaluation, but also includes the cost associated with setting up the evaluation: assembling $\mathbf{z}$ in Equation 1.1, as well as analyzing the model output y_i. For example, when SST aerodynamics is evaluated using a RANS solver, designers typically first discretize the vehicle OML through mesh generation, which incurs a pre-processing cost c_{pre}. Subsequently, the RANS simulation is performed at a computational cost c_{query}. Finally, fields of interest or quantities of interest are defined, evaluated, and passed to the outer-loop optimizer for iterative optimization, leading to post-processing cost c_{post}. The total cost of conducting one HF assessment is therefore the sum of these cost components. In many-query applications, these costs accumulate rapidly and can easily exceed the allocated budget.

In contrast, LF methods, compared with HF methods, typically reduce not only c_{query} but also c_{pre} and c_{post}. For example, many LF fluid solvers are meshless, thereby avoiding the pre-processing costs associated with HF geometry OML definition and mesh generation. The total cost of conducting one LF evaluation is therefore much more manageable than that of an HF assessment. Multi-fidelity methods therefore aim to reduce the reliance on HF evaluations in many-query applications. By strategically querying LF models, the number of required HF evaluations can be significantly reduced, provided that the LF data offer useful approximations of the HF information and that an effective model management strategy is used to combine information from multiple fidelity sources.

1.4.5 Relevant Multi-Fidelity Methods for SST Design

As discussed in Section 1.4.2 and Section 1.4.3, numerical models are built to associate predictions across the fidelity gap. Adaptation is one of the popular model management strategies, and depending on the model types, existing surrogate modeling approaches for aerodynamic design can be broadly categorized into several classes based on their modeling paradigm, as summarized in Table 1.1.

While all method classes can be formulated in a multi-fidelity manner, it is critical to differentiate them based on the desired use case for SST conceptual development using the metrics highlighted in Section 1.3. Specifically, we would like the surrogate model to be (1) generalizable across the design space, (2) computationally efficient for online deployment, and (3) capable of field prediction.

Most of these models are orders of magnitude more computationally efficient relative to HF simulations, which makes this metric less informative for comparison. However, these methods differ in their ability to generalize across the design space and to predict high-dimensional flow-field quantities. The purpose of this section is to survey the landscape of multi-fidelity surrogate modeling methods applicable to SST conceptual design and to rationalize the choice of GNN as the final modeling approach.

1. Scalar surrogate models One common surrogate modeling approach is to learn corrections to scalar-valued aerodynamic force and moment coefficients (QOIs) through Bayesian frameworks, such as multi-fidelity Gaussian Process Regression (GPR), to synthesize multiple data sources [6, 52, 60, 89]. Among these works, Mukhopadhaya et al. provide an excellent example of applying GPR to generate multi-fidelity probabilistic aerodynamic databases for the simulation of commercial aircraft, synthesizing data from VLM, RANS CFD, and wind tunnel experiments [89].

Similarly, polynomial chaos expansion (PCE) has been used for robust design optimization, where aerodynamic responses are modeled as scalar QOIs to construct statistical measures such as mean and variance within the ASO loop [34, 39, 95, 152]. More recently, artificial neural networks (ANNs) have been used to construct such data-driven models [1, 4, 55]. While ANNs provide greater modeling flexibility due to their highly customizable architectures, these types of surrogate models still focus on QOI prediction and may face difficulties in rigorous uncertainty propagation compared to GPR- or PCE-based methods.

Method Class	Representative Work	Generalization	Field Prediction
Scalar Surrogate Models (GP, PCE, ANN, ...)	[1, 4, 34, 39, 6, 52, 55, 60, 74, 89, 95, 152]	Low–High (formulation-dependent)	No
Reduced-Order Models (POD, POD-Kriging, nonlinear ROMs)	[10, 21, 26, 27, 41, 58, 87, 98, 110, 132]	Low–Moderate	Yes
Image-based Neural Networks (CNN, PINNs)	[92, 123, 127, 130, 142]	Low–Moderate (mesh constrained)	Yes
Operator Learning / Implicit Representations	[3, 20, 64, 65, 70, 105, 153]	High (function-space generalization)	Yes
Geometry-Aware Neural Models (PointNet, GNN)	[12, 23, 33, 46, 68, 99, 118, 119, 143, 150]	High (geometry invariant)	Yes

Table 1.1: Summary of surrogate modeling approaches for aerodynamic design and their suitability with respect to the desired properties listed in Section 1.3.

While these approaches are effective for reducing the cost of many-query applications, they are limited to predicting predefined QOIs. This limitation arises primarily from the "curse of dimensionality," whereby the computational cost scales quadratically with the problem dimension, and the model struggles to capture correlations among samples due to increasing domain sparsity. As a result, these methods cannot capture spatially resolved flow features or support the evaluation of arbitrary QOIs *a posteriori*, which limits their applicability to SST design problems involving complex and nonlinear flow phenomena.

2. Field surrogate models based on reduced-order modeling reduced order modeling (ROM) is frequently used for surrogate modeling of field data [10]. By decomposing the flow field into a set of spatial modes, proper orthogonal decomposition (POD) enables a parameterized representation of field variables based on the linear combination of a truncated set of basis vectors. Using regression to predict modal amplitudes as a function of input parameters provides a non-intrusive approach for field surrogate modeling [21, 132].

Classical POD, as a reduced-order modeling method, can be used to interpolate across a design space if a mapping between modal amplitudes and DVs is defined. POD-kriging is one such approach, where GPR is used to estimate the maximum-likelihood modal coefficients at given geometric DVs [87]. Recent approaches have also sought to integrate POD-based surrogate modeling into multi-fidelity frameworks, including through the use of manifold alignment to combine data from different sources [58, 98]. However, the linear projection assumption of POD can limit accuracy when modeling complex flow features. To address this limitation, nonlinear dimensionality reduction methods, such as manifold learning (e.g., ISOMAP) and autoencoder-based reduced-order models, have been explored in both single- and multi-fidelity settings [26, 27, 41, 110].

Yet, while these approaches improve modeling efficiency, their generalization across substantially different geometries remains challenging. Existing studies largely focus on FFD-based OML perturbations, whereas conceptual design requires exploration of much larger design spaces involving major planform and configuration-level geometric variations.

3. Image-based Neural Networks Deep neural networks provide an alternative approach for modeling complex nonlinear flow fields. In particular, convolutional neural network (CNN) and related architectures have been applied to learn mappings between input parameters and flow-field quantities [92, 123, 127, 130]. Physics-informed neural networks (PINNs) further incorporate governing equations into the training process to improve physical consistency [142]. These approaches can represent high-dimensional field quantities, but their rasterized operators typically require consistent spatial topology across training samples. For large geometric variations, maintaining this correspondence becomes difficult and often requires domain transformations to a common reference shape or implicit geometry encodings such as distance fields. These requirements limit their generalization across the broad design spaces encountered in MDO.

4. Operator learning and implicit representations More recently, operator-learning frameworks and implicit neural representations have been proposed to learn mappings between function spaces $\mathcal{G} : \mathcal{A} \to \mathcal{U}$, where $\mathcal{A}$ is the function space of partial differential equation (PDE) parameters and $\mathcal{U}$ is the function space of PDE solutions. Methods such as DeepONet, Fourier Neural Operators (FNO), and related architectures aim to approximate solution operators in a mesh-independent manner [64, 70, 105]. Geometry-informed neural

operators and other extensions have been applied to aerodynamic prediction and parametric flow modeling [3, 65, 153]. Implicit neural representations have also been used for flow-field reconstruction [20].

These approaches offer strong generalization across discretizations; in fact, when properly trained, such operator models are discretization-agnostic, as the learned operator resides in a continuous function space. However, the representation of complex geometries and their integration with multi-fidelity data sources remain active areas of research. In addition, many of these operator-learning models, especially transformer-based architectures, are computationally expensive to train. Their high training cost and data requirements make domain-specific surrogate construction challenging.

5. Geometry-aware neural models To address the limitations associated with mesh-dependent representations from image-based neural networks, recent work has shifted toward geometry-aware models that operate directly on unstructured data. Point-cloud-based models, such as `PointNet`, encode geometric and physical information as nodal features and can directly predict surface pressure distributions [119]. Graph neural networks (GNNs) extend this idea by modeling interactions between nodes through edges to generate graph-level field predictions (e.g. pressure, velocity), enabling the representation of complex geometries and physical constraints [12, 23, 62, 68, 99, 118, 150]. Due to their graph-based structure, these models naturally accommodate geometric variation and nonuniform spatial sampling, making them well suited for aerodynamic design problems with large geometry changes. GNNs also show strong potential for multi-fidelity modeling because they can learn mappings between field quantities defined on different discretizations. [33, 46, 63, 143].

Taken together, these approaches highlight the fundamental trade-offs in existing surrogate modeling methods for aerodynamic design. Scalar surrogate models offer strong efficiency but are limited to predefined QOIs and cannot capture spatially resolved flow physics. Reduced-order modeling techniques enable field prediction but struggle with nonlinear flow features and large geometric variations. Image-based neural networks are constrained by field rasterization requirements, limiting their applicability across diverse geometries. Operator learning and implicit representations provide a promising framework for discretization-agnostic modeling; however, their training cost can be prohibitive for domain-specific applications.

In contrast, geometry-aware neural models, particularly GNNs, naturally operate on unstructured representations that encode both geometric and physical information. This enables strong generalization across varying geometries while supporting direct prediction of high-dimensional flow fields. Based on these considerations, GNN-based models are well suited to address the combined requirements of field prediction, geometric generalization, inference efficiency, and multi-fidelity integration in SST conceptual design.

1.5 Contributions

Building on the discussion of the aircraft design process (Section 1.1 and Section 1.2), the challenges and need for higher-fidelity conceptual design methodologies (Section 1.3), and the identification of multi-fidelity methods as a means to introduce higher confidence in conceptual design (Section 1.4), it is now possible to address the challenge of conceptual design under S&C constraints for SSTs. Specifically, by developing a multi-fidelity, GNN-based field surrogate modeling framework that integrates LF VLM analysis with HF CFD data, this dissertation seeks to bridge the fidelity gap in conceptual aerodynamic design and proposes a new paradigm for data assimilation across multiple solvers. The main contributions are summarized as follows:

Contribution 1: Multi-fidelity field-prediction surrogate for aerodynamic correction. This work proposes a graph neural network (GNN)-based mapping function, `MF-VortexNet` (hereafter referred to as `VortexNet`), to enhance the accuracy of VLM in conceptual aircraft design. The model learns a mapping from LF lattice-wise pressure loadings to projected high-fidelity RANS CFD surface pressure distributions, effectively correcting separation and nonlinear vortex effects that are not captured by VLM. By representing panel-wise pressure, geometry, and lattice connectivity as a graph, and by leveraging message-passing operations, the surrogate uses low-fidelity aerodynamic analysis as a field-level physics prior to capture local flow–geometry interactions and reconstruct quasi-CFD-level pressure fields. The resulting model demonstrates strong geometric generalizability and significantly improved predictive accuracy of QOIs (C_L and C_M) as well as their derivatives ($C_{L,\alpha}$ and $C_{M,\alpha}$) relative to standalone VLM, while maintaining negligible additional computational cost during inference. The methodology and part of the results associated with this contribution were previously presented in Shen et al. [121, 124, 125].

Contribution 2: Open-source multi-fidelity aerodynamic dataset for field-based learning. To support the development and evaluation of multi-fidelity surrogate models, this work introduces an open-source aerodynamic dataset for a parametrized double-delta wing configuration, named `double-delta-aero`. The dataset contains 272 unique geometries with multiple AOAs ranging from 11° to 19° at low speed (Ma = 0.3), capturing complex vortex-dominated flow physics. A nested Saltelli sampling scheme is employed to generate a structured design space with controlled geometric variability, enabling future sensitivity analysis and uncertainty quantification. The dataset includes aerodynamic coefficients, surface and volume flow solutions, and full CFD setup files in open formats, providing a reusable benchmark for the community. The construction and release of this dataset were previously documented in Shen and Alonso [122].

Contribution 3: Empirical scaling laws for multi-fidelity GNN surrogate accuracy. This study further investigates how the predictive accuracy of trained `VortexNet` models scale with training data, model size, and computation budget. Using nested subsets of the `double-delta-aero` dataset, ranging from 8 to 256 geometries, an empirical scaling study of `VortexNet` is conducted across multiple sampling densities and model sizes. The results demonstrate that prediction error improves primarily through data scaling, with an identifiable power-law regime over part of the dataset range, while direct model-size scaling provides weaker gains. This contribution addresses a key gap in the literature, where scaling laws for physics-informed, multi-fidelity GNN models remain largely unexplored, and offers guidance for data sizing, model selection, and compute allocation in aerodynamic surrogate modeling and MDO applications. Part of the scaling results are previous presented in Shen and Alonso [122].

Contribution 4: Integration of multi-fidelity surrogate into conceptual design and MDO. The `VortexNet` surrogate model and the proposed multi-fidelity workflow are integrated into the `SUAVE` conceptual design environment and applied to a SST planform ASO test case. In particular, the framework is applied to a stability-constrained double-delta planform shape optimization problem, enabling high-fidelity-informed design exploration within a conceptual design setting. The improved fidelity enables S&C-constrained

design, addressing a key bottleneck in traditional LF analysis. This integration demonstrates that `VortexNet` can enhance design fidelity without sacrificing computational efficiency, improve design confidence across the entire design space, and enable the incorporation of nonlinearity-governed aerodynamic effects into early-stage design, thereby addressing a fundamental limitation in early-stage MDO. The optimization workflow and stability-constrained design demonstration build on the previously published work in Shen and Alonso [120], while the results are updated using the "`double-delta-aero`" dataset introduced in Contribution 2.

1.6 Dissertation Structure

This dissertation is organized as follows: Chapter 2 provides an overview of the multi-fidelity framework, including the digital design and analysis environment, the roles of individual software components, and the data pipeline. Additionally, the fundamentals of the two solvers at LF and HF, namely VLM and CFD, respectively, are introduced.

Chapter 3 presents the multi-fidelity field surrogate model, `VortexNet`, in detail. This chapter describes the GNN architecture, data pipeline, and training procedure. A performance assessment based on a delta wing test case is provided, including field prediction error, model generalizability across the design space, and an analysis of the prediction mechanism through latent space ablation.

Chapter 4 introduces the double-delta aerodynamic dataset, an open-source benchmark featuring more realistic SST planforms at low speeds and moderate AOA. This dataset is used in subsequent chapters to support model scaling analysis and aerodynamic shape optimization.

In Chapter 5, the data scaling behavior of `VortexNet` is investigated, leading to recommendations on dataset size and empirical scaling laws. Chapter 6 integrates insights from the previous chapters and incorporates `VortexNet` into the conceptual design environment `SUAVE`. The fidelity gains achieved by the proposed surrogate modeling approach are demonstrated through an SST planform optimization case, targeting cruise drag minimization under low-speed stability constraints.

Finally, the dissertation concludes with a discussion of key findings and directions for future work in Chapter 7.

Chapter 2

Multi-Fidelity Aerodynamic Modeling

As discussed in Section 1.4, aerodynamics can be evaluated using multiple models and data sources across different fidelity levels. In general, multi-fidelity aerodynamic data may be curated from computational simulations, wind-tunnel testing, or flight experiments. For conceptual design applications, however, computational models are especially useful because they can be repeatedly evaluated across large design spaces. Two classical physical models commonly used for this purpose are VLM and CFD. This chapter describes the formulations of these computational models, as well as the data pipeline used to conduct these analyses.

Using the variable definition in Equation 1.1, we define the field of interest (FOI) as the pressure field obtained on a reference domain. Such a reference domain can be arbitrary: physical reference planes, such as the pressure distribution on the wing OML surface or on the lattice panels used in VLM, can serve as valid reference domains. Alternatively, abstraction and projection can be used to define reference domains independent of the vehicle geometry. In this study, we adopt a physically defined reference domain. Specifically, we aim to estimate the pressure distribution (C_p) over the wing OML from CFD for HF aerodynamics, and the distributed loading coefficient (ΔC_p), defined as the pressure difference across lattice panels due to wing-induced velocity, from VLM for LF aerodynamics.

Two software packages are used. The LF aerodynamics are computed using a VLM solver in the `SUAVE` [72, 74] software package. For the HF data, we use the surface pressure coefficient computed with the `SU2` CFD solver [30, 96]. The computational costs of LF and

HF simulations differ drastically, as highlighted in Table 2.1. In this chapter, the physical and numerical models are presented in Section 2.1 and 2.2 for CFD and VLM, respectively.

Model	Fidelity	Cost	Software
VLM	Low	$\mathcal{O}(10$ CPU-sec$)$	`SUAVE`
RANS-CFD	High	$\mathcal{O}(3.6 \times 10^5$ CPU-sec$)$	`SU2`

Table 2.1: Summary of multi-fidelity analysis tools.

Beyond the solvers themselves, an analysis pipeline is required to construct geometry and orchestrate different analysis components, such as meshing and the execution of LF and HF solvers. For this purpose, `SUAVE` is used to drive the multi-fidelity analyses, and Section 2.3 describes the analysis pipeline in detail.

2.1 High-Fidelity Solver: Reynolds-averaged Navier-Stokes

To simulate compressible flows, `SU2` solves the Navier-Stokes equations expressed in differential form as

$$\mathcal{R}(U) = \frac{\partial U}{\partial t} + \nabla \cdot \bar{F}^c(U) - \nabla \cdot \bar{F}^v(U, \nabla U) = 0, \tag{2.1}$$

where $\mathcal{R}(\cdot)$ denotes the residual, $\bar{F}^c$ is the convective flux and $\bar{F}^v$ is the viscous flux, and ∇ is the spatial differential operator $\nabla = [\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}]$.
The vector of conserved variables, U, can be expressed as

$$U = [\rho, \rho\bar{v}, \rho E]^\mathsf{T}, \tag{2.2}$$

where ρ is the fluid density, $\bar{v} = [u, v, w]^\mathsf{T}$ is the flow speed in a Cartesian system of reference, and E is the total energy per unit mass. The convective and viscous fluxes are given by

$$\bar{F}^c = \begin{bmatrix} \rho\bar{v} \\ \rho\bar{v} \otimes \bar{v} + \bar{\bar{I}}p \\ \rho E\bar{v} + p\bar{v} \end{bmatrix} \tag{2.3}$$

and

$$\bar{F}^v = \begin{bmatrix} 0 \\ \bar{\bar{\tau}} \\ \bar{\bar{\tau}} \cdot \bar{v} + \kappa \nabla T \end{bmatrix} \tag{2.4}$$

respectively, where p is the static pressure, $\bar{\bar{\tau}}$ is the viscous stress tensor, T is the temperature, κ is the thermal conductivity, and μ is the viscosity. The viscous stress tensor can be expressed in vector notation as

$$\bar{\bar{\tau}} = \mu \left(\nabla \bar{v} + \nabla \bar{v}^T \right) - \mu \frac{2}{3} \bar{\bar{I}} \left(\nabla \cdot \bar{v} \right). \tag{2.5}$$

For a perfect gas with gas constant R and constant specific heat ratio γ, the ideal gas equation of state can be used to determine the pressure and temperature as $p = (\gamma - 1)\rho \left[E - 0.5(\bar{v} \cdot \bar{v}) \right]$ and $T = p/(\rho R)$, respectively. Laminar viscosity, μ, is calculated using Sutherland's law and thermal conductivity can be computed as $\kappa = \mu c_p / Pr$, where c_p is the specific heat capacity at constant pressure and Pr is the Prandtl number, or it can be set to a constant value. Steady, turbulent flows can be simulated in `SU2` by solving the RANS equations. In accordance with the Boussinesq hypothesis, the effective viscosity can be computed as the sum of the laminar viscosity (μ_d) and the turbulent viscosity (μ_t) as

$$\mu = \mu_d + \mu_t, \tag{2.6}$$

where the prediction of μ_t is dependent on the particular choice of turbulence model used.

In this work, the one-equation Spalart-Allmaras turbulence model with rotation correction is used to solve for the value of the turbulent viscosity [128, 25]. The turbulence model is selected to achieve a balance between computational cost and adequate vortex effects quantification accuracy, as discussed in a previous study by Seraj and Martins [114]. All simulations employ the Jameson-Schmidt-Turkel scheme for convective discretization [47] and implicit time integration with an adaptive Courant–Friedrichs–Lewy number to converge the steady-state solution. The vehicle boundary condition is specified by a no-slip, adiabatic wall, and a characteristic-based free-stream is used for the far-field boundary. A Cauchy convergence criterion is used, such that the solver stops when the change in C_D for the wing over the previous 100 iterations becomes less than 1×10^{-4}.

2.2 Low-Fidelity Solver: Vortex Lattice Method

The VLM solver used in `SUAVE` is based on `VORLAX` [83]. It models compressibility-corrected, inviscid, attached flows for both subsonic and supersonic conditions by using a panel method that represents the base geometry with infinitesimally thin, cambered or uncambered panels [15]. Each panel consists of a horseshoe vortex with trailing vortices extending to infinity and a bound vortex at the quarter-chord location. The control point $[x_c, y_c, z_c]$ is located at the three-quarter-chord point of each panel. The `VORLAX` implementation of VLM differs from typical textbook VLM [11] as multiple application-specific correction terms are introduced. To assist the discussion in later sections, we include a short summary of VLM. One should refer to Miranda's original manuscript for the exact algorithm used in `VORLAX` [83].

The fluid dynamics solved by VLM are based on potential flow theory. Under the assumptions of steady, irrotational ($\nabla \times \mathbf{v} = 0$), incompressible, and inviscid flow, the Navier–Stokes equations (Eq. 2.1) can be reduced to Laplace's equation, $\nabla^2\phi = 0$, where ϕ is the velocity potential and velocity field $\mathbf{v} = \nabla\phi$. In aerodynamic applications, this equation is solved as a boundary-value problem, in which distributed vortex filaments with strength Γ_n are placed throughout the physical domain, forming a lattice grid. The filament strengths are then solved such that the local velocity at the control points satisfies the non-penetration boundary condition; specifically, the local velocity must be tangent to the camber surface at these control points. To do so, the local velocity must be computed from $\{\Gamma_n\}$. According to the Biot–Savart law, the induced velocity from a vortex filament can be described as:

$$\mathbf{v} = \frac{\Gamma_n}{4\pi} \int_c \frac{d\mathbf{l} \times \mathbf{r}}{r^3}, \tag{2.7}$$

where $d\mathbf{l}$ is an infinitesimal segment of the vortex filament, $\mathbf{r}$ is a vector pointing from the vortex filament to a point in space, and c is the length of the vortex filament. For a horseshoe vortex with known endpoints A and B in space, the induced velocity vector $\mathbf{v}$ at any point C in space can be expressed as:

$$\mathbf{v}_{AB} = \frac{\Gamma_n}{4\pi} \left(\frac{\mathbf{r}_0}{|\mathbf{r}_1 \times \mathbf{r}_2|} \left(\frac{\mathbf{r}_0\mathbf{r}_1}{|\mathbf{r}_0\mathbf{r}_1|} - \frac{\mathbf{r}_0\mathbf{r}_2}{|\mathbf{r}_0\mathbf{r}_2|} \right) \right), \tag{2.8}$$

where $\mathbf{r}_0$ is the vector from point A to B, $\mathbf{r}_1$ is the vector from point A to C, and $\mathbf{r}_2$ is the vector from point B to C. In Equation 2.8, all vectors can be computed if the control point

and vortex filament locations are known, thereby establishing a function between Γ_n and $\mathbf{v}$. In a system with multiple vortex filaments, the resultant induced velocity at any point in space is the linear superposition of all vortex filament contributions. For a system with N horseshoe vortices (panels), at the control point m, the induced velocity is then

$$\mathbf{v}_m = \sum_{n=1}^{N} C_{m,n}\Gamma_n, \tag{2.9}$$

where $C_{m,n}$ is the influence matrix computed from Equation 2.8.

Applying the tangency boundary condition $\mathbf{n} \cdot \mathbf{v} = 0$ on both the body surface and the wake uniquely defines the system. These boundary conditions, together with the spatial correlations, can be conveniently summarized into a right-hand side vector (RHS), resulting in the final linear system for $\mathbf{\Gamma}$, the vector all vortex strengths at all filaments, as follows:

$$AIC\mathbf{\Gamma} = RHS, \tag{2.10}$$

where AIC is the aerodynamic influence coefficient matrix representing the induced velocity contributions from each vortex filament to each panel's control points, and RHS represents the free-stream flow contributions to the boundary conditions. The linear system can thus be solved for individual lattice vortex strengths.

Once the vortex filament strengths are solved, the pressure coefficient distribution is computed by

$$C_p = -\frac{2}{q_\infty^2}(U_\infty u + V_\infty v), \tag{2.11}$$

where U_∞ and V_∞ are the components of the free-stream velocity vector of magnitude q_∞, and u and v are the induced velocity components. C_p is the field pressure coefficient ($C_p = (P - P_\infty)/q_\infty$). In our study, an infinitesimally thin panel is used such that the pressure difference between the lower and upper wing surfaces, ΔC_p, is the C_p computed from the previous equation.

Finally, ΔC_p is converted to forces and moments as follows: For each panel, the local loading coefficient is computed by multiplying ΔC_p by the panel area factor. The loading is then decomposed into normal and axial force components, where the axial component is further corrected using a leading-edge thrust correction [29]. If the Polhamus suction correction is active [102], this leading-edge thrust is rotated by 90° to be normal to the leading-edge panel chord. The forces are then rotated from the local chord coordinate system

to the body axes using the chord incidence angle and dihedral. The dimensional forces and moments for the vehicle are obtained by integrating the panel strip areas from the leading edge to the trailing edge and applying angle-of-attack and sideslip angle transformations.

2.3 Multi-Fidelity Workflow

The `SUAVE` conceptual design environment is used to generate vehicles from geometry parameters, drive low-fidelity simulations, and create surface-mesh tessellations using its built-in `OpenVSP` [93] interface. Depending on the subsequent applications, the model representations of a given vehicle configuration may have different fidelities.

At the lowest fidelity, an instance of the `Vehicle` class in `SUAVE` is created, taking in the corresponding design variable vector $\mathbf{x} \in \mathbf{R}^n$, where n is the dimension of the design space, and a prescribed process that constructs a vehicle from $\mathbf{x}$. Details of the computer program structure, including the `Vehicle` class definition, can be found in Lukaczyk et al. [72]. When $\mathbf{x}$ is changed, different instances are created to represent different vehicle configurations.

When different analyses are conducted on the vehicle instance, different types of instance representation are needed. When the VLM (see Section 2.2) is utilized, a lattice grid of the vortex distribution is created based on the `Vehicle` instance by incorporating the encoded geometric parameters. Additional configuration settings, such as the number of panels or the distribution functions used for the lattice discretization, may change the vehicle-geometry representation for a given instance. The lattice grid and vortex distribution are used by the VLM solver to perform aerodynamic analyses.

Likewise, for CFD simulations (Section 2.1), a representation of the OML of the vehicle is required by the solver. Under such analyses, we use `SUAVE`'s `OpenVSP` interface to translate the `SUAVE` instance to an OML boundary representation. Subsequently, `OpenVSP`'s `ComputeCFDMesh` function is used to generate a tessellated surface mesh. This mesh is then imported into `Pointwise`, together with a far-field boundary defined with a diameter of 40 mean aerodynamic chords (MACs), to define the fluid domain. The fluid domain is then further discretized to generate the volume mesh for CFD analyses.

The overall workflow for `VortexNet` surrogate model training and GNN-guided aerodynamic shape optimization is shown in Figure 2.1. Two phases exist: in the offline phase, shown as blue-colored blocks in Figure 2.1, the surrogate model described in Chapter 3 is trained using a dataset of vehicle configurations ($\{\mathbf{x}_s\}$) sampled from a predefined design

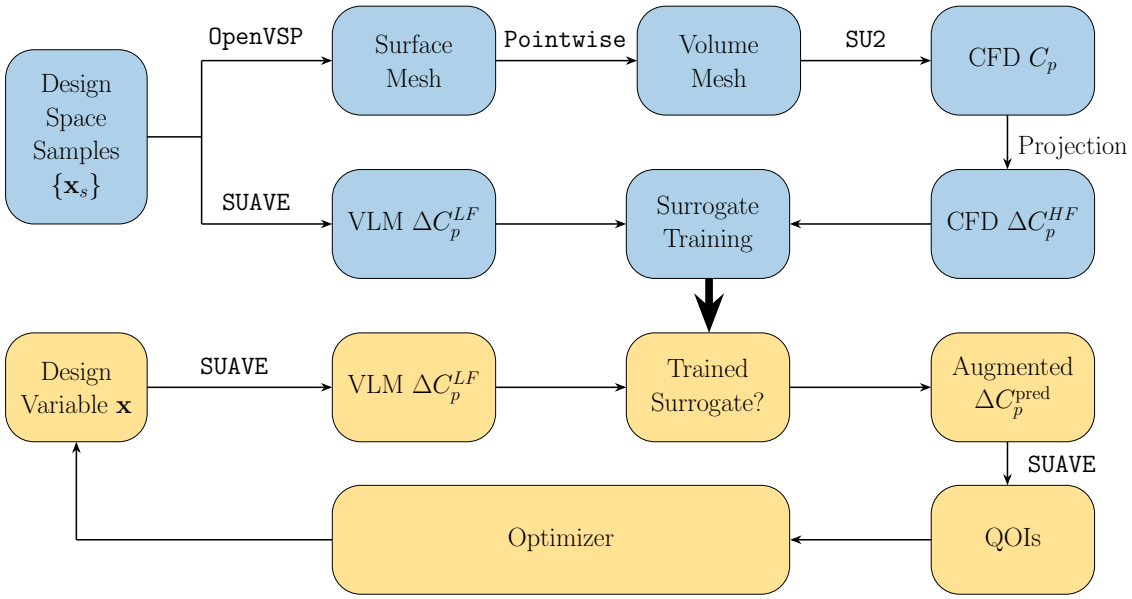


Figure 2.1: Block diagram of the surrogate model training and the GNN-guided aerodynamic shape optimization workflow. Each block represents an intermediate stage in the design workflow, with "online" stages colored gold and "offline" stages colored blue, while the tools used for each specific process are annotated alongside the arrows.

space. Following vehicle definition, two fidelity levels branch out to conduct the LF and HF simulations. The LF simulations use the VLM solver in `SUAVE`, while the HF simulations employ the `OpenVSP` API in `SUAVE` to build surface meshes, followed by volume mesh construction using `Pointwise`, and CFD computation using `SU2`. The final paired set of LF and HF field data is then used to train the `VortexNet` surrogate model.

Once trained, the surrogate model is ready to be deployed for online assessment, shown as gold-colored blocks in Figure 2.1. The thick black arrow connecting "Surrogate Training" and "Trained Surrogate" in Figure 2.1 highlights this transition. For online assessment, the design variable $\mathbf{x}$ is fed into `SUAVE`'s vehicle definition. Subsequently, an LF VLM simulation is conducted, which serves as the input to the surrogate model. When surrogate model augmentation is used, the surrogate model then enhances the VLM-predicted ΔC_p^{LF} to a quasi-HF field prediction ΔC_p^{pred}. This augmented ΔC_p is then fed back into `SUAVE` for QOIs evaluation. When the surrogate model is not used, the VLM-predicted ΔC_p^{LF} is used for QOIs evaluation.

Chapter 3

Surrogate Modeling

Leveraging the *adaptation* MF model management strategy as discussed in Section 1.4.3, a mapping function $\mathcal{F}$ between the LF field variables q^{LF} into their HF counter part q^{HF} is used as the MF surrogate model. In general, such a mapping can be expressed as

$$q^{HF} \sim \mathcal{F}\left(q^{LF}, \omega_{\mathcal{L}}\right), \tag{3.1}$$

where $\omega_{\mathcal{L}}$ represents the weights and biases of the network under a specific loss function $\mathcal{L}$. In general, q may denote an arbitrary physical field, including surface quantities (e.g., pressure or skin-friction) or volumetric variables (e.g., velocity). In this study, we focus specifically on the pressure field over a reference plane, as the planform pressure distribution is directly related to the S&C dynamics, a LF modeling challenge as discussed in Section 1.3. Such a mapping can be defined if the LF field variables q^{LF} exhibit an injective correspondence with the HF field variables q^{HF}, a hypothesis that has not been tested in this study. To construct such a surrogate model, a GNN-based U-net-like (explained later in Section 3.1) ML autoencoder is used. The network architecture, the training pipeline, and model performance on a test dataset are presented in detail in the following sections.

3.1 Machine Learning Architecture

A graph consists of a set of nodes and edges $G = (V, E)$, where $V = \{1, ..., n\}$ is a set of nodes and E is a set of edges between nodes $i, j \in V$. A GNN is thus a model that performs machine learning on graphs to incorporate information about the structure of the graph to

understand the relationship among nodes and edges [40]. This capability is classified into three prediction tasks: node, edge, and graph predictions. The node prediction capability is of particular interest to the authors, as this type of prediction uses graph information to generate predictions on nodal values through message passing rounds. In each message passing round, each node updates its nodal embedding using information aggregated from its neighbors' embeddings. This type of prediction integrates well with mesh-based simulations, such as VLM or RANS solvers used in this study, as the simulation results' data structure can be easily represented by a graph G [46, 149, 154].

Recently, multiple studies have leveraged GNN architectures to predict physical fields, including across varying fidelities. Pfaff et al. proposed `MeshGraphNets`, which use message-passing mechanisms to learn mesh-based simulations on irregular and time-evolving computational grids with noticeable efficiency and accuracy for both static and dynamic systems [99]. This feature is desirable for the surrogate modeling task in this paper, as the surrogate model must not be sensitive to spatial discretization changes, such as mesh morphing introduced by geometry variation in the MDO cycle. Shao et al. proposed an auto-encoder GNN surrogate called `PIGNN-CFD` for rapid prediction of urban wind fields based on irregular unstructured mesh data from CFD simulations [118]. In their work, they added the Reynolds stress tensor residual to the loss function, thereby implicitly building physical constraints into the surrogate model. For multi-scale and multi-fidelity applications, U-Net-type architectures are also commonly used [33]. These architectures follow an "U-shaped" encoder–decoder structure: the encoder progressively extracts increasingly abstract features, while the decoder reconstructs the output at the target resolution. Skip connections between corresponding encoder and decoder levels allow the model to retain local spatial information while also capturing global context. Deshpande et al., Yang et al., and Wei and Nikolaos demonstrated the effectiveness of U-net type GNNs in physical simulations, particularly in the domains of finite element method (FEM)-based structural simulations and smooth particle hydrodynamics (SPH)-based fluid simulations [28, 143, 150]. Their work highlighted the superior performance of graph U-Nets in multi-scale representation learning, accommodating unstructured mesh data, modeling long-range interactions, and being mesh topology-agnostic. Finally, Liu et al. [68] used a graph attention network-based [140] surrogate for fluid simulations and found that the attention mechanism helps to describe physical relationships among graph nodes [68].

The multi-fidelity VortexNet model (`MF-VortexNet`), hereafter `VortexNet`, proposed in

this paper, has an architecture inspired by the aforementioned works. Specifically, it utilizes a multi-head message aggregation and passing layer using Graph Attention Network v2 (`GATv2`) [16] and incorporates U-net-type skip connections between encoders and decoders [34]. The use of multi-head attention enables the network to attend to richer feature subspaces by concurrently building multiple regression models between a given node and its neighbors, thereby enhancing its feature-capturing capability. Additionally, as the network may require multiple layers of message aggregation and passing, U-net type skip connections are employed to short-circuit deep encoders, improving the network's efficiency in capturing low-frequency features. Overall, the network is designed to handle graph sizes typically used in VLM simulations, which contain hundreds to thousands of nodes. It aims to capture nonlinear flow patterns in RANS simulations and augment VLM results, providing a tool to increase design fidelity during the conceptual design phase.

Figure 3.1 provides a high-level schematic of the proposed `VortexNet` network. In this figure, the computational graph before and after each major block is shown in black boxes. The graph consists of 4 nodes arranged in a Delta wing layout, and 5 black solid lines connecting them represent edges in the graph. The authors want to emphasize that the edges and nodes shown here are just for illustration purposes and do not represent the actual graph topology of the input data. The purple vectors (represented by purple boxes) beside each node are the corresponding node feature arrays, and the yellow vectors (represented by yellow boxes) beside each edge are the edge feature arrays. Specific details regarding the node and edge features are presented later in Table 3.2. As the graph passes through the network, the node feature array and edge feature array may change size, which is indicated by the lengthening or shortening of the boxes representing the arrays. Hence, each individual graph in the boxes represents a snapshot of the graph's state. In this chart, a black arrow indicates a direct connection among stages, and a blue arrow represents a U-net type skip connection that will be explained in the subsequent sections.

The input graph first goes through an Input Encoder, which maps the input features to a latent dimension. The encoded graph then goes through a series of Convolutional Blocks. These blocks use multi-head attention mechanisms, and hence the feature array size may be multiplied by the number of heads. In this network, the first half of the Convolutional Blocks serves as the encoder, and the second half of the Convolutional Blocks serves as the decoder. The decoder blocks receive information from both their previous blocks and the graph information passed on by skip connections. Finally, an Output Decoder, which

utilizes an architecture similar to the Convolutional Block but with optional fully connected layers to adjust the output dimension, is used to produce nodal predictions for a field value of interest, such as pressure or velocity fields. The following sections describe the key components of the `VortexNet`.

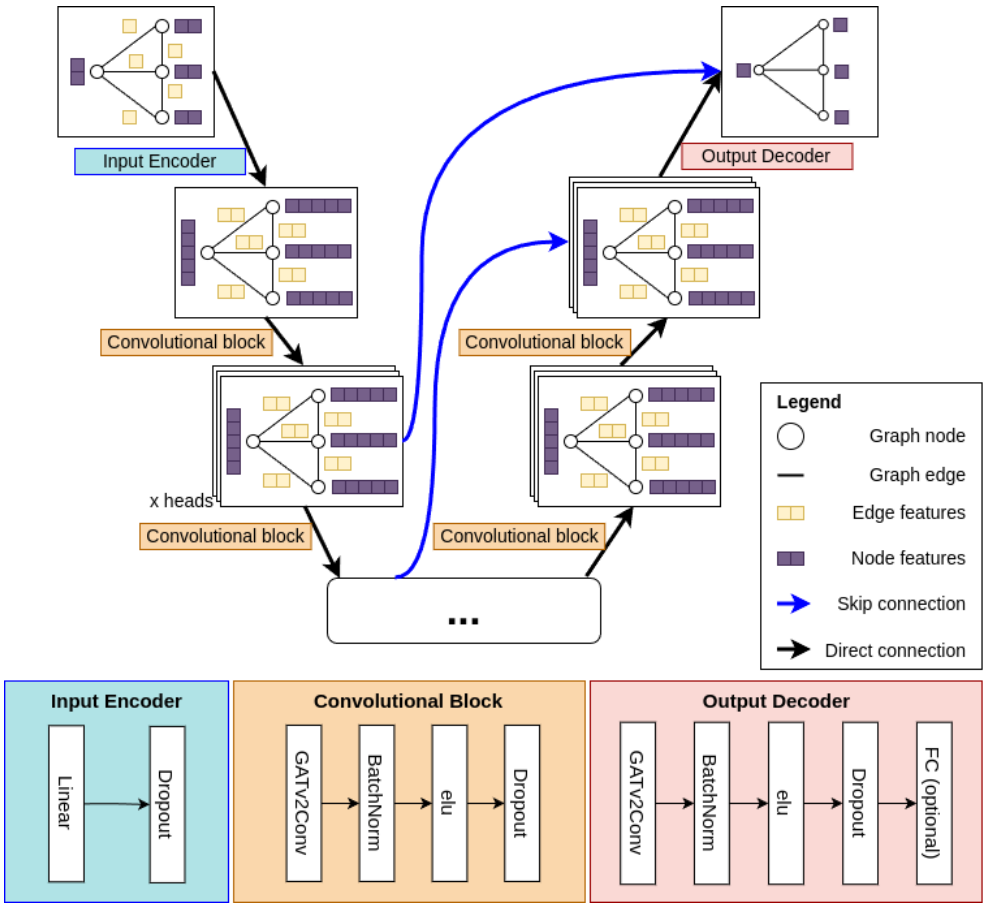


Figure 3.1: Schematic of the proposed `VortexNet` GNN architecture. Colors indicate different network blocks. A black arrow represents direct message passing between blocks, and a blue arrow denotes a skip connection to the receiving block. The figure also presents snapshots of the graph at each computational step, showing nodes, edges, and their associated feature arrays.

3.1.1 Input Encoders

The Input Encoder comprises two components: a node encoder and an edge encoder. The objective of including this input encoder is to uplift and enrich the input node features and

edge features. In the current work, an 11-dimensional node features and a 1-dimensional edge features are used. Hence, a feature encoding to expand the input feature array to a higher-dimensional space is beneficial for the latter stage of the Convolutional Blocks, where latent relationships will be identified. A linear transformation is used for both the node encoder and the edge encoder, such that

$$\mathbf{h}_i^{'} = \mathbf{W}_n \mathbf{h}_i + \mathbf{b}_n, \tag{3.2}$$

where $\mathbf{h}_i^{'}$ is the encoded feature, $\mathbf{h}_i$ is the input feature, $\mathbf{W}_n$ are the learnable weights and $\mathbf{b}_n$ are the learnable biases. The node encoder encodes node feature to a dimension of $[\mathbb{F} \times \mathbb{F}]$, where $\mathbb{F}$ is defined by the hyper-parameter `hidden_feature`, and the edge encoder encodes edge feature to a dimension of $\mathbb{F}$. After the linear layer, a dropout layer is used for the node feature array.

3.1.2 Convolutional Block

In each Convolutional Block, the primary mechanism for updating node features based on graph structure is by using `GATv2Conv` layers [16]. Specifically, the `GATv2Conv` computes a weighted average of the transformed featured of the neighborhood nodes for each nodes, expressed as,

$$\mathbf{h}_i^{'} = \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \mathbf{h}_j \right), \tag{3.3}$$

where $\mathbf{h}_i^{'}$ is the new feature representation at node i, $\mathcal{N}_i$ is the set of neighboring nodes connected to node i by direct edges, α_{ij} and $\mathbf{W}$ are the learned weights, $\mathbf{h}_j$ is the input feature array at node j, and σ is an activation function. The attention weights α_{ij} indicate the importance of node j's features to node i and allow every node to attend to every other node to which it is connected in the graph [16]. This mechanism enables the Convolution Block to capture how information propagates across the graph and, when the node and edge features are appropriately designed, to operate in a manner that is independent of the underlying mesh discretization. The actual attention function is defined as:

$$\alpha_{ij} = \mathrm{softmax}_j \left(e(\mathbf{h}_i, \mathbf{h}_j) \right), \tag{3.4}$$

and

$$e(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{a}^T \text{LeakyReLU}(W_n \cdot [\mathbf{h}_i || \mathbf{h}_j] + W_e \cdot E_{i.j}), \tag{3.5}$$

where $e(\mathbf{h}_i, \mathbf{h}_j)$ is the feature importance score of neighbor node j to node i. Matrices $\mathbf{a}$, W_n and W_e are learned weights, $E_{i.j}$ are the corresponding edge features, and $(||)$ denotes matrix concatenation [16].

The `PyTorch GATv2Conv` layer employs multiple attention heads that allow the model to attend to information from different representation subspaces. In the multi-head attention scenario, multiple independent attention mechanisms execute the transformation of Equation 3.3, and then the resulting feature array $\mathbf{h}_i^{'}$ is concatenated.

Utilizing this `GATv2Conv` layer, the Convolution Block consists of one `GATv2Conv` layer that maps the input graph to an output graph with a node feature array in $\mathbb{R}^{\mathbb{F} \times \texttt{heads}}$ after concatenation. Additionally, `heads` is a hyper-parameter specifying the number of heads. The input-output dimension is consistent across all layers except for the first Convolution Block after the Input Encoder, where the input feature dimension is $\mathbb{R}^{\mathbb{F} \times \mathbb{F}}$. After the `GATv2Conv` layer, the feature passes through batch normalization, an exponential linear unit (elu) activation function, and a dropout regularization layer. The dropout layer includes a tunable hyper-parameter named `dropout_rate`.

In the `VortexNet` architecture, the Convolutional Block is repeated $\texttt{hops} - 1$ times, where `hops` is a tunable hyperparameter specifying the network depth. It is worth noting that in many previous graph U-net-like architectures, pooling layers are used to successively reduce the computational graph size in deeper encoder layers to minimize non–influential information overhead [28, 46, 150, 143]. However, such pooling layers are not used in the current network because the implications of using these methods under the current MF learning setting remain uncertain to the authors at this time: although sequential graph sparsification across layers improves encoding efficiency, the final field prediction must ultimately be generated at the original training target resolution. Consequently, unpooling operations are required to reconstruct a dense graph from the coarsened latent representation [46]. These reconstruction procedures may introduce systematic bias into the predicted fields and therefore require careful architectural design and model validation. In contrast, architectures without pooling sacrifice computational efficiency but avoid the potential reconstruction-induced bias. This trade-off led the authors to omit nodal pooling between layers. A discussion on the potential future direction of including pooling layers in the `VortexNet` architecture is presented in the Results section (Section 3.4).

3.1.3 Output Decoder

The Output Decoder's major role is to map latent space features to the desired output dimension. This block contains layers identical to those used in a Convolution Block, except that the `GATv2Conv` layer now maps the feature array from $\mathbb{R}^{\mathbb{F}\times\texttt{heads}}$ to $\mathbb{R}^1$. An optional fully connected (FC) layer is appended at the end of this block to enable graph size adjustment for the final output. In the context of this work, since the input graph and training labels have identical graph sizes, this fully connected layer is not used.

Although edge features can be extracted after the Output Decoder block, they are discarded. Consequently, the Output Decoder provides scalar value predictions at each node, which represent the field value predictions that mimic the HF training labels.

3.1.4 Skip Connections

The skip connection is marked by the blue curvy arrow in Figure 3.1. For the ith Convolutional Block, the output is connected to the $(\texttt{hops} - i)$th block's output. The first encoder block is skip-connected to the output decoder, the second to the final decoder block, and so on in reverse order. Dimension adjustment is required for the first skip connection only through a linear layer; all other encoder–decoder pairs have matching feature dimensions and therefore require no adjustment. A skip connection weight hyper-parameter (`Alpha`) is used to control the amount of skip connection information that the subsequent decoder utilizes, such that the feature array $\mathbf{h}_i$ seen by the subsequent decoder is

$$\mathbf{h}_i = \mathrm{A}\mathbf{h}_{i,\text{skip connection}} + (1 - \mathrm{A})\mathbf{h}_{i,\text{previous block}}, \tag{3.6}$$

and this skip connection weight (A) is applied uniformly across all skip connections.

Alternative model architectures were explored, including models without multi-head attention or skip connections. Based on trial and error and engineering judgment, the architecture presented here was found to provide sufficient flexibility for the domain-specific multi-fidelity surrogate modeling task considered in this work. The corresponding architectural choices can be adjusted through the hyperparameters presented in Table 3.1.

3.2 Training

The prediction accuracy of `VortexNet` relies heavily on the training scheme. Throughout experiments, it has been observed that a training scheme with a combination of learning rate scheduling, regularization techniques, and physical loss terms helps `VortexNet` to achieve higher prediction accuracy. The following sections discuss the training scheme used in this work.

3.2.1 Multi-Phases Training

The entire dataset, as described in Section 3.3, is randomly partitioned into a $30\% - 70\%$ test-training split. For network training, only the training set is exposed to the `VortexNet`, while the test set is reserved for performance characterization. The training set is further randomly partitioned into four batches. In each fold of the k-phase training, one batch of the training set serves as the cross-validation set while the remaining subsets serve as the training set. This technique exposes the model to diverse data distributions and prevents overfitting to specific cross-validation subsets [5].

In standard k-fold cross-validation training, the models trained in each fold are independent and the goal is to evaluate the robustness of the architectural design. If there is a need to select a model, a voting algorithm is typically employed. In the current study, however, we made minor adjustments to the model voting mechanism. Instead of training mutually independent models in each fold, our training script recycles the best model weights identified so far from all previous folds' training. Specifically, a register actively stores the best-performing model, as indicated by the cross-validation loss, and in each fold training resumes from the current best model.

In using this training scheme, the cross-validation set in the later folds may overlap with the training data of previous folds, potentially inflating the model performance in later folds. However, this strategy was chosen for two reasons. First, the size of the total training set is limited compared to the number of trainable weights in the `VortexNet`. Using standard k-fold training prohibits making full use of the training set. Second, the way physical loss terms are used in training necessitates such model recycling, as discussed in the next section.

3.2.2 Loss Function

The loss function used for training is a compound loss defined as:

$$\mathcal{L} = \text{SmoothL1Loss}\left(q^{HF}, q^{\text{pred}}\right) + \Lambda\,\text{PhysicalLoss}\left(q^{\text{pred}}\right), \tag{3.7}$$

where "SmoothL1Loss," and "PhysicalLoss" are the loss components; q^{HF} is the high-fidelity data provided as training labels; $q^{\text{pred}} = \mathcal{F}\left(q^{\text{LF}}, w_{\mathcal{L}}\right)$ is the `VortexNet` predicted field output, and Λ is the weight of the physical loss which is controlled by two hyper-parameters as described in Equation 3.9. The "SmoothL1Loss" is well-documented in the `PyTorch` manual [104].

The "PhysicalLoss" term is introduced to improve the model's prediction capability. In designing surrogate models, two typical approaches are commonly employed: physics-driven and data-driven. In the physics-driven approach, the governing equations can either be embedded in the aggregation function or incorporated into the loss term. For surrogate models aimed at HF flow field prediction, the residuals of the Navier-Stokes equations at selected points are often used to drive the network convergence [31]. While accurate, such networks are computationally expensive to train. Alternatively, in the data-driven approach, the physical loss is omitted. However, this approach often suffers from high variance in prediction errors.

Considering that the proposed surrogate model is intended for conceptual design purposes, where trend-wise accuracy and delta values of QOIs are more critical than the absolute prediction accuracy, we aim to employ a lightweight physical constraint. This constraint helps calibrate the surrogate model predictions for trend-wise quality without adding significant complexity to the network architecture. In this study, we propose a physical loss term based on LF VLM simulation to improve the network's trend-wise prediction quality. Variations in nonlinear flow features, such as boundary layer effects, could be considered to be equivalent to the errors committed through violations of the "PhysicalLoss" term.

To compute the physical loss, recall that VLM solves the linear equation shown in Equation 2.10. When a surrogate model is used to predict $q^{\text{pred}} = \Delta C_p^{\text{pred}}$, this field can be passed backward through the stored lattice geometry information to compute the predicted vortex strength distribution, Γ^{pred}. The physical loss term can thus be expressed as a 2-norm of the residual:

$$\text{PhysicalLoss}(q^{\text{pred}}) = ||AIC \cdot \Gamma^{\text{pred}}(q^{\text{pred}}) - RHS||. \tag{3.8}$$

In the above physical loss computation, no real-time computation of the AIC and RHS

matrices is required, as this information can be stored during the dataset assembly phase, thereby creating minimal overhead for network training.

When the physical loss weight Λ is large, the surrogate-predicted ΔC_p^{pred} will closely resemble the VLM results. However, this undermines the intent of using the surrogate model to map LF field data to HF field data. To address this, an exponential decay of the physical loss weight Λ is applied during the k-fold training. Initially, the model is trained to closely match the VLM solver, so that a good trend-wise baseline is established. While the model keeps the learned mapping, the physical constraint is gradually relaxed. Towards the final folds, the surrogate model's training convergence is primarily driven by data to capture the non-linear flow patterns present in HF simulations. The following equation is used to decay the physical loss weight across folds:

$$\Lambda_i = \Lambda^{\max}\left(\Lambda^{\min} + \left(1-\Lambda^{\min}\right)\exp\left(-\ln\left(\frac{1}{\Lambda^{\min}}\right)\frac{i-1}{k-1}\right)\right), \quad \text{for } i \in \{1,\ldots,k\} \tag{3.9}$$

where $\Lambda^{\max}$ is a hyper-parameter (`max_phys_loss`) representing the initial physical loss weight, $\Lambda^{\min}$ is a hyper-parameter (`min_phys_loss`) that specifies the target final magnitude of the physical loss weight, and k is the number of folds in k-fold training.

3.2.3 Optimizer and Training Scheme

Optimization is performed using the ADAM optimizer with an initial learning rate specified by the hyperparameter `learning_rate` and a decay rate defined by `decay_rate` [56]. A dynamic learning rate scheduler is implemented to halve the learning rate when the validation loss ceases to improve across training epochs. In addition, an early stopping criterion is applied: if model performance does not improve for 150 consecutive epochs, training for the current fold is terminated and the best-performing model up to that point is retained to mitigate overfitting. The validation loss extends the objective defined in Equation 3.7 by incorporating a sign-violation regularization term. This term is defined as the average number of nodal prediction sign violations relative to the ground truth. Including this indicator in the validation metric supports more reliable early stopping by discouraging physically inconsistent predictions. In particular, it reduces overfitting that may otherwise introduce high-variance errors in the afterbody wing panels, where ΔC_p magnitudes are typically small but contribute significantly to aerodynamic moment due to their relatively large moment arms. To further improve prediction robustness, 3% random noise is added

to all node features during training. Across all experiments and hyperparameter configurations, a batch size of 32 is used. Training is conducted on an NVIDIA A100 graphics processing unit (GPU).

3.2.4 Hyper-parameter Optimization

The surrogate model possesses nine hyper-parameters that need to be optimized. The hyper-parameter optimization study is conducted using `Optuna`, an open-source hyper-parameter optimization framework that automates hyper-parameter search using advanced sampling schemes and efficient pruning algorithms [2]. The hyper-parameters, along with their bounds listed in Table 3.1, define the search space, and `Optuna` automatically identifies the optimal hyper-parameter settings that minimize the model test loss over the test set. The model test loss is defined as the Mean Squared Error (MSE) between the predicted ΔC_p^{pred} and the corresponding HF ΔC_p^{HF} across the test set.

Hyper-parameter Name	Value Range	Definitions
`hops`	int$[2, 16]$	total layers of Convolutional Block
`hidden_feature`	int$[3, 64]$	hidden layers feature width
`heads`	int$[1, 8]$	number of heads for multi-heads attention
`dropout_rate`	$[0.05, 0.4]$	dropout regularization weight
`Alpha`	$[0.0, 1.0]$	skip connection bandwidth
`max_phys_loss`	$[0, 0.5]$	maximum physical loss weight in Equation 3.9
`min_phys_loss`	$[1, 1000] \times 10^{-5}$	minimum physical loss weight in Equation 3.9
`learning_rate`	$[1, 1000] \times 10^{-4}$	initial learning rate to the ADAM optimizer
`decay_rate`	$[1, 10000] \times 10^{-6}$	decay rate to the ADAM optimizer

Table 3.1: List of the hyper-parameters studied, along with their value range and definitions.

In total, 80 hyper-parameter configurations are generated by `Optuna` in this study. The training time varies across configurations, but on average, each configuration takes 1.75 hours to train. The average test loss across all hyper-parameter configurations is 4.13×10^{-2}. Across trained models with test losses below 3×10^{-2}, the aerodynamic coefficients computed

from the predicted fields are similar. However, the predicted pressure fields exhibit minor stylistic differences, characterized by localized, small-area checkerboard-like fluctuations. These patterns are low in amplitude and do not correspond to large-magnitude oscillations. The final model is selected for its robustness, exhibiting consistent field prediction accuracy across the design space.

The final selected model adopts nine Convolution Blocks with an additional output decoder, resulting in a total of ten message-passing rounds. The hidden feature dimension is 63 with four attention heads. A dropout rate of 8.5×10^{-2} is applied. The skip-connection bandwidth is 9.7×10^{-3}, indicating that most information propagates through the deeper layers. The initial physics loss is 0.42 and decays to 5.8×10^{-5} throughout training. The learning rate is 1.3×10^{-4}, with a decay rate of 1.6×10^{-6}.

3.3 Dataset

The dataset for surrogate model training and testing consists of 15 3-D Delta wing geometries at two levels of fidelity. The HF surface pressure field data is generated using the `SU2` CFD solver as described in Section 2.1. The flow domain is discretized using `Pointwise`'s automatic unstructured volume mesh capability [101]. The outer mold line of each underlying geometry is accurately captured by enforcing a maximum surface mesh-to-geometry deviation of less than 1×10^{-5} [m]. Across all geometries, identical mesh property definitions are used to ensure consistency in global mesh properties and boundary conditions, including the T-Rex boundary layer mesh growth profile, leveraging `Pointwise`'s scripting capability. Due to variations in surface body geometry, the total number of elements in the fluid domain ranges from 4.89 to 19.47 million. The resultant mesh achieves $y^+ < 1$ for most parts of each wing, except near the wing tips at the trailing edge, satisfying the turbulence model's y^+ requirements.

The LF data is generated using the VLM implementation in `SUAVE`. For all geometries, a 30×30 lattice discretization is applied in the chord-wise and span-wise directions. The lattice is evenly distributed in the chord-wise direction and follows a cosine distribution in the span-wise direction.

The 15 Delta wing geometries are parameterized using two design variables: leading edge sweep angle and root airfoil shape. The leading edge sweep angles used in this study are 55°, 65°, and 75°. Across all geometries, the root chord length is kept constant at 0.65

[m], resulting in variations in span and reference area of the Delta wing. The root airfoil shapes are taken from NACA 4-digit airfoils, and in this study, NACA 0010, NACA 0016, NACA 0024, NACA 2416, and NACA 4416 are used. All 15 combinations of the root airfoil shape and leading edge sweep angle are generated using `SUAVE`.

Each geometry is then simulated under 40 different free-stream conditions, varying AOA, Mach number (Ma), and Reynolds number (Re). The AOA is sampled from 0° to 20°, the Ma from 0.35 to 0.5, and the Re from 6.5×10^6 to 1×10^7. A Latin Hypercube Sampling (LHS) technique is used to generate samples across the three free-stream variables. It is worth noting that the LF VLM model does not utilize the Reynolds number in its simulation. In total, 600 test samples are generated across the geometry design space and the free-stream variable space.

Two additional holdout datasets are prepared using a similar workflow to assess the model's generalizability to previously unseen geometries and free-stream conditions. These datasets are introduced in Section 3.4.1 and Section 3.4.2, respectively, and are not exposed to the surrogate model during training. All CFD runs are conducted on the Sherlock Cluster at the Stanford Research Computing Center (SRCC), utilizing a 64-core 2.4 [GHz] partition.

3.3.1 Dataset Preparation

The HF and LF datasets differ significantly in their dimensions. While the LF data resides in $\mathbb{R}^{30\times30}$, the HF data dimension is defined by the number of surface mesh nodes, which varies based on the geometry. Although GNN-based architectures can accommodate mapping between these different graph structures, training such surrogate models can be computationally expensive and is not the focus of this work.

In this study, the goal is to demonstrate the capability of utilizing such architectures to build surrogate models that enhance conceptual vehicle design at a relatively low computational cost. To achieve this, the authors decided to standardize the HF and LF graphs by projecting data from both sources onto a reference state. One approach is to use the lattice panels defined in the VLM as the reference plane and project the HF surface pressure field onto this surface. As VLM computes the local loading coefficient $\Delta C_p^{\text{LF}} = C_{p,\text{lower}} - C_{p,\text{upper}}$, it was decided that the standardization step for the HF data should project the surface pressure onto the lattice panels to generate the HF local loading coefficient, ΔC_p^{HF}. This process is described by the following equations:

$$\Delta C_p^{\mathrm{HF}} = C_{p,\mathrm{lower}}^{\mathrm{HF}} - C_{p,\mathrm{upper}}^{\mathrm{HF}}, \tag{3.10}$$

where C_p^{HF} is a vector in $\mathbb{R}^{30\times 30}$. The pressure coefficient for each side can be calculated by:

$$C_{p,\mathrm{side}}^{\mathrm{HF}} = \frac{1}{q_{\mathrm{ref}}} \left[\frac{\sum_{i\in\mathcal{I}} \delta_{\mathrm{side}}(i) P_i A_i}{\sum_{i\in\mathcal{I}} \delta_{\mathrm{side}}(i) A_i} \right]_{\mathcal{I}=1}^{N}, \tag{3.11}$$

where P_i is the pressure, A_i is the cell area at HF node i, $\mathcal{I}$ is the set of all surface nodes whose (x_i, y_i) coordinates lie within the boundary of a lattice panel, indexed from 1 to N, under vertical projection, and q_{ref} is the reference pressure computed from free-stream conditions. $\delta_{\mathrm{side}}(i)$ is an upper or lower surface selection mask that identifies whether a node belongs to the lower or upper surface based on its z_i coordinate (applicable for the root airfoils used in the current study). For example, the $\delta_{\mathrm{lower}}(i)$ mask can be defined as:

$$\delta_{\mathrm{lower}}(i) = \begin{cases} 1 & \text{if } z_i < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

3.3.2 Node and Edge Features

As discussed in Section 3.3, the VLM simulation is Reynolds number agnostic. To enrich the aerodynamic features used in the surrogate model, free-stream conditions, including free-stream AOA, Ma, and Re, are incorporated alongside the local loading coefficient ΔC_p^{LF}. Because the proposed surrogate model is intended to augment aerodynamic prediction for design space exploration in conceptual design, it is essential to include geometric properties that influence aerodynamics in the surrogate model inputs. These geometric properties include wing thickness, surface curvatures, normalized lattice panel control point coordinates, and surface slopes. All geometric features are defined and evaluated at the nodal level. The control point coordinates are normalized using a characteristic length. In this study, the root chord length of 0.654 [m] is selected because it remains invariant across geometries within the design space. Furthermore, all geometries are translated so that the wing apex coincides with the origin. This treatment facilitates scale-invariant surrogate modeling and improves model generalizability.

At the edge level, the Euclidean distance computed from normalized coordinates between nodes is used to represent nodal connectivity patterns and spatial relationships

among nodes. For each node, the Euclidean distances to its four nearest neighboring nodes are included in the edge feature. The combination of nodal and edge features allows the `VortexNet` surrogate model to capture the spatial and aerodynamical information for aerodynamic characterization.

The nodal and edge features used by the `VortexNet` surrogate model, along with their intended purposes, are summarized in Table 3.2.

Feature Name	Dimension	Purpose
ΔC_p^{LF}	$[N_{\mathrm{nodes}} \times 1]$	VLM local loading coefficient
AOA $[rad]$	$[N_{\mathrm{nodes}} \times 1]$	free-stream AOA
Ma_∞	$[N_{\mathrm{nodes}} \times 1]$	free-stream Mach number
Re_∞	$[N_{\mathrm{nodes}} \times 1]$	free-stream Reynolds number
Thickness	$[N_{\mathrm{nodes}} \times 1]$	wing thickness at lattice panel control points
Upper Surface Curvature	$[N_{\mathrm{nodes}} \times 1]$	wing upper surface curvature at lattice panel control points
Lower Surface Curvature	$[N_{\mathrm{nodes}} \times 1]$	wing lower surface curvature at lattice panel control points
Upper surface slope	$[N_{\mathrm{nodes}} \times 1]$	wing upper surface chord-wise slope at lattice panel control points
Lower surface slope	$[N_{\mathrm{nodes}} \times 1]$	wing lower surface chord-wise slope at lattice panel control points
x_c	$[N_{\mathrm{nodes}} \times 1]$	normalized lattice panel control points x-coordinates
y_c	$[N_{\mathrm{nodes}} \times 1]$	normalized lattice panel control points y-coordinates
Euclidean distance	$[4N_{\mathrm{nodes}} \times 1]$	normalized Euclidean distances from one control point to its four nearest neighbors'

Table 3.2: List of the nodal and edge features used by the `VortexNet`, along with their dimension and intended purposes.

Once these node and edge features are assembled, two types of feature normalization are available: a z-score standardization and a tanh standardization. For the z-score standardization, normalization is performed for all nodal features except for the normalized coordinates, as they are scaled by the characteristic length. For the tanh standardization, the coordinate are normalized by the MAC, while all other features are normalized using tanh transformation. All results presented in this chapter uses z-score standardization, while results presented in Chapter 5 and Chapter 6 utilize tanh standardization. The two

standardization methods does not introduce significant performance difference once properly trained. The difference in data pipeline between chapters are due to the chronological order of experimental development, as well as the later one offers easier integration to a MDO pipeline.

3.4 Results

In this work, we focus on predicting local loading coefficients (ΔC_p). Once ΔC_p is predicted, arbitrary QOIs can then be calculated using the existing multi-fidelity workflow as discussed in Section 2.3. To highlight the effectiveness of the proposed multi-fidelity workflow, two prediction accuracy tests are performed. First, we evaluate the surrogate model `VortexNet`'s ΔC_p prediction quality. We examine the RMSE of panel-wise ΔC_p^{pred} prediction against the projected CFD reference value ΔC_p^{HF} for geometries within the surrogate model's training set across various free-stream conditions. To further demonstrate `VortexNet`'s generalizability in maintaining comparable prediction quality across geometries within the dataset's design space, unseen geometries are introduced and the surrogate model's accuracy is evaluated. These results are presented in Section 3.4.1. Second, to quantify the aerodynamic QOI prediction accuracy gain of adopting the multi-fidelity framework, lift (C_L) and pitching moment (C_M) coefficients are evaluated using `VortexNet`-predicted ΔC_p^{pred} and compared against the CFD reference and VLM benchmark. This comparison is conducted for geometries that are used for `VortexNet` training, and for the unseen geometries as mentioned in Section 3.4.1. The results are presented in Section 3.4.2. Overall, the proposed multi-fidelity workflow showed robustness with respect to underlying geometry changes and a significant fidelity gain over the VLM benchmark. To rationalize and understand the prediction mechanism and assist further development of `VortexNet`-like GNN-based surrogate models, a latent space ablation study is presented in Section 3.5 to discuss how the graph features are utilized across the latent layers in `VortexNet`.

3.4.1 Local Loading Coefficient (ΔC_p) Predictions

To evaluate the model's performance, we consider two distinct aspects. First, to quantify its prediction accuracy, we examine the root-mean-square error (RMSE) of the field predictions ΔC_p^{pred} against the projected CFD references ΔC_p^{HF}, as outlined in Section 3.3. Second, to assess the model's generalizability and its potential for assisting in design space exploration,

we evaluate its ΔC_p^{pred} RMSE against ΔC_p^{HF} using new geometries that were held out from the surrogate model training. The following sections discuss these two assessments.

Predictions for Seen Geometries during Training

The RMSEs for the training set, test set, and the entire dataset are 0.1064 with an R^2 value of 0.9519, 0.1065 with an R^2 value of 0.9500, and 0.1064 with an R^2 value of 0.9513, respectively. No significant difference between the test and training sets is observed, and most of the variance in the dataset is captured by the trained model, indicating that the resultant model is well-trained.

To visualize the ΔC_p from the VLM, `VortexNet` predictions, and CFD references, a geometry configuration at high AOA was randomly selected, featuring a leading edge sweep angle of 55° and a root airfoil of NACA2416. The free-stream conditions are an AOA of 16.4°, Ma of 0.47, and Re of 9.78×10^6. The corresponding CFD pressure on the upper and lower wing surfaces are presented in Figure 3.2. As shown in the figure, the surface pressure exhibits a "vortex lift" pattern: on the upper surface toward the wing outboard near the trailing edge, suction effects induced by the separated vortex are evident.

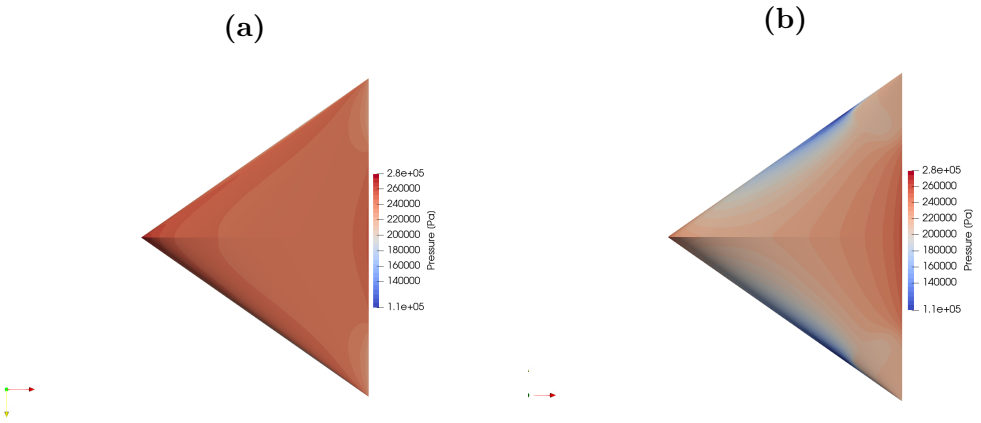


Figure 3.2: The HF surface pressure at the lower (figure (a)) and upper (figure (b)) surfaces obtained from CFD for a wing with 55° leading edge sweep and a root airfoil profile of NACA 2416 at AOA= 16.4°, Ma = 0.47, and Re = 9.78×10^6.

The aforementioned C_p is then projected onto lattice panels. Figure 3.3(c) shows the projected ΔC_p^{HF}, which serves as the "ground-truth" reference. The projected pressure field preserves flow features from the CFD, including faint traces of vortex lift at the wing

outboards. In Figure 3.3(a), the ΔC_p^{LF} computed from the VLM is shown. This pressure field follows potential flow theory, and no vortex lift or flow separation is observed. The VLM pressure field serves as input to `VortexNet` as part of the nodal features.

Figure 3.3(b) shows the ΔC_p^{pred} predicted by `VortexNet`. Comparing these pressure fields, it is evident that the predicted pressure field exhibits strong visual similarity to the projected CFD pressure field, both in the predicted ΔC_p values and in the overall pressure distribution. However, artifacts of reconstruction error are noticeable near the wing apex at the central body. Despite these discrepancies, the authors believe that the surrogate model's predictions provide higher fidelity than those of VLM, as it captures nonlinear flow features.

Although not shown, ΔC_p^{pred} predictions for other geometries and free-stream conditions follow a similar pattern, with `VortexNet`'s predictions appearing visually closer to the projected CFD fields (ΔC_p^{HF}) than to those of the VLM (ΔC_p^{LF}).

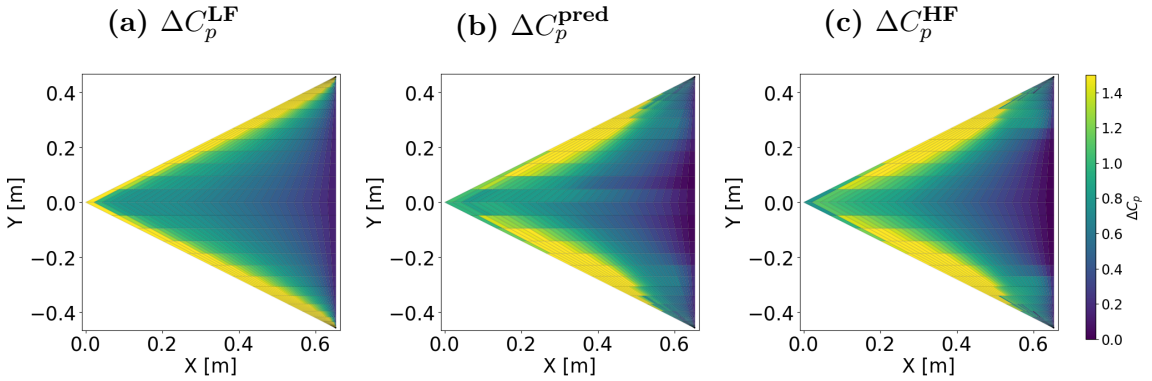


Figure 3.3: Lattice panels and local loading coefficients ΔC_p are visualized using color maps ranging from $\Delta C_p = 0$ to $\Delta C_p = 1.5$, with (a) VLM ΔC_p^{LF}, also serving as the LF input, (b) `VortexNet` ΔC_p^{pred}, and (c) the reference HF ΔC_p^{HF} obtained by projecting the CFD surface pressure onto the lattice panels.

To provide a more quantitative analysis of the ΔC_p prediction error, we compared the RMSE across all geometries in the test set. The corresponding values are presented in Table 3.3. The RMSE ranges from 0.0566 to 0.1579 in the observed dataset with a standard deviation of 0.0366 across all geometries. Using the leave-one-out one sample t-test, under the null hypothesis that the prediction RMSEs are consistent for all geometries, it is observed that one geometry with a leading-edge sweep angle of 75° and root airfoil shape of and NACA 0010 is an outlier. For the remaining geometries, the trained model

maintains a relatively consistent prediction accuracy. Overall, `VortexNet` has maintained generalizable field prediction capability across the design space. However, the performance discrepancy among geometries indicates that validating model generalizability is necessary before deploying surrogate models to assess whether prediction accuracy across the design space is sufficient for downstream applications.

Sweep Angle [°]	**Root Airfoil (NACA 4-digits)**				
	0010	**0016**	**0024**	**2416**	**4416**
55	0.0566	0.0964	0.1330	0.0872	0.0875
65	0.0942	0.1159	0.0577	0.1046	0.1016
75	0.1579	0.0945	0.0900	0.1506	0.1286

Table 3.3: Test set RMSEs between ΔC_p^{pred} and ΔC_p^{HF}, grouped by wing geometries.

Predictions for Unseen Geometries

One potential use case of the proposed `VortexNet` is to assist in design space exploration during conceptual design. In such applications, a design space is first defined by specifying a set of geometric parameters and their respective bounds. A coarse sampling of geometries, using both HF and LF methods, is then conducted to train the surrogate model. Subsequent design space exploration can rely on the surrogate model's augmentation of LF methods to provide quasi-high-fidelity aerodynamic evaluations. Consequently, it is essential to evaluate the proposed surrogate model's generalizability to geometries not included in the training dataset.

To study generalizability, we tested geometries and feature arrays not encountered by the network during training and evaluation. Four design points within the geometric parameter ranges of the training set's design space, as tabulated in Table 3.3, were selected. Specifically, we chose two leading-edge sweep angles (60° and 70°) and two root airfoils (NACA 0013 and NACA 3416) that induce thickness and camber variations. A combination of these design variables yields four new wing geometries: 60° NACA 0013, 60° NACA 3416, 70° NACA 0013, and 70° NACA 3416. Each geometry is then simulated under 11 different far-field conditions, with AOA, Mach number, and Reynolds number sampled using LHS. The sampling bounds for AOA, Ma, and Re are $[0°, 20°]$, $[0.35, 0.5]$, and $[6.5, 10] \times 10^6$, respectively, matching those used to construct the training set as discussed in Section 3.3.

The resulting RMSEs are listed in Table 3.4. The maximum RMSE observed is 0.3293 for the 70° Delta wing with the NACA 3416 airfoil, and the minimum RMSE is 0.1441 for the 60° Delta wing with the NACA 0013 airfoil. On average, the RMSE is 0.2481, representing a 132.9% increase from the RMSE reported in the previous tests on seen geometries. A reduction in prediction accuracy for unseen geometries is expected, as a similar trend is reported in the previous study [124].

To better understand the potential origin of this degradation, we examine how the RMSE varies with the type of geometric variation from the training set. Within this limited new-geometry dataset, the errors appear to correlate more strongly with camber variation. Configurations introducing a new camber shape exhibit larger RMSE. A hypothesis is that camber variation modifies the global pressure loading pattern and redistributes ΔC_p over a larger portion of the surface, thereby altering both magnitude and spatial gradients of the field. As the current training dataset exposes `VortexNet` to a limited representation of camber profiles, the model's performance may be more sensitive to camber variation. Expanding the design space coverage to include more camber shapes in the `VortexNet` training set may help reduce prediction errors.

Sweep Angle [°]	**Root Airfoil (NACA 4-digits)**	
	0013	**3416**
60	0.1441	0.2695
70	0.2106	0.3293

Table 3.4: Unseen geometries RMSEs

To further diagnose the source of error, a qualitative study was conducted by visually examining the resulting ΔC_p^{pred} fields with reference to VLM and CFD ΔC_p results. For comparison across geometries, we present the ΔC_p for all geometries at an AOA of 18°, Ma of 0.4 and Re of 7×10^6. The results are presented in Figure 3.4.

When comparing `VortexNet`'s prediction accuracy to the results shown in Figure 3.3, we observe that for previously unseen geometries, the surrogate model introduces more artifacts in its predictions. For the 70° NACA 0013 configuration, the surrogate model fails to accurately capture the highest ΔC_p values. Small-wavelength artifacts of high- and low-pressure variation can be seen for 70° geometries near the leading edges toward the wing tips. These checkerboard-like pressure fluctuation patterns are most likely to be the dominant contributing factor for increased RMSE for these test geometries, suggesting that

the dominant error mode is not a global bias in pressure magnitude but rather a high-frequency spatial inconsistency. Further examination in Section 3.4.2, which evaluates the accuracy of integrated QOIs, further supports this interpretation.

Despite these artifacts, when considered alongside other fidelity methods, `VortexNet` continues to demonstrate intermediate capability between VLM and CFD in capturing major flow features. The authors believe that by refining the surrogate model architecture, as discussed in Section 3.5, to promote spatial smoothness and multi-scale consistency and improving the training data sampling scheme, the model's generalizability and prediction accuracy can be further enhanced.

3.4.2 Aerodynamic Coefficients Predictions

We consider the CFD-predicted surface pressure coefficients and aerodynamic coefficients as the "ground-truth" reference. The error is thus the difference between QOIs computed with and without using the surrogate model. In the current study, two sources of error exist. The first is an epistemic discretization error introduced during the surrogate model training database preparation. Instead of using CFD surface pressures (C_p), the surrogate model is trained on and predicts projected CFD surface pressures ΔC_p^{HF}. This projection process introduces discretization errors that cannot be reduced during model training and will lead to differences in computed QOIs. The second error is the surrogate model's modeling error due to its imprecise representation of ΔC_p^{HF}. This error can be reduced through model training and hyper-parameter optimization. In this study, the surrogate's modeling error is minimized by using optimal model hyper-parameters, while we have not extensively explored error reduction techniques for the epistemic discretization error.

We assess the workflow's performance from two perspectives. First, to evaluate the fidelity gain of adopting `VortexNet` in the multi-fidelity workflow, we study the QOIs performance of the workflow on geometries seen during training but at unseen free-stream conditions. We compare the QOIs accuracy obtained from the VLM, the `VortexNet` surrogate model, and projected CFD against the CFD ground truth. These results are presented in Section 3.4.2. Second, we examine the QOIs prediction characteristics of the proposed workflow on geometries not seen during surrogate model training, to demonstrate its performance in interpolating within the design space. This generalization assessment is presented in Section 3.4.2.

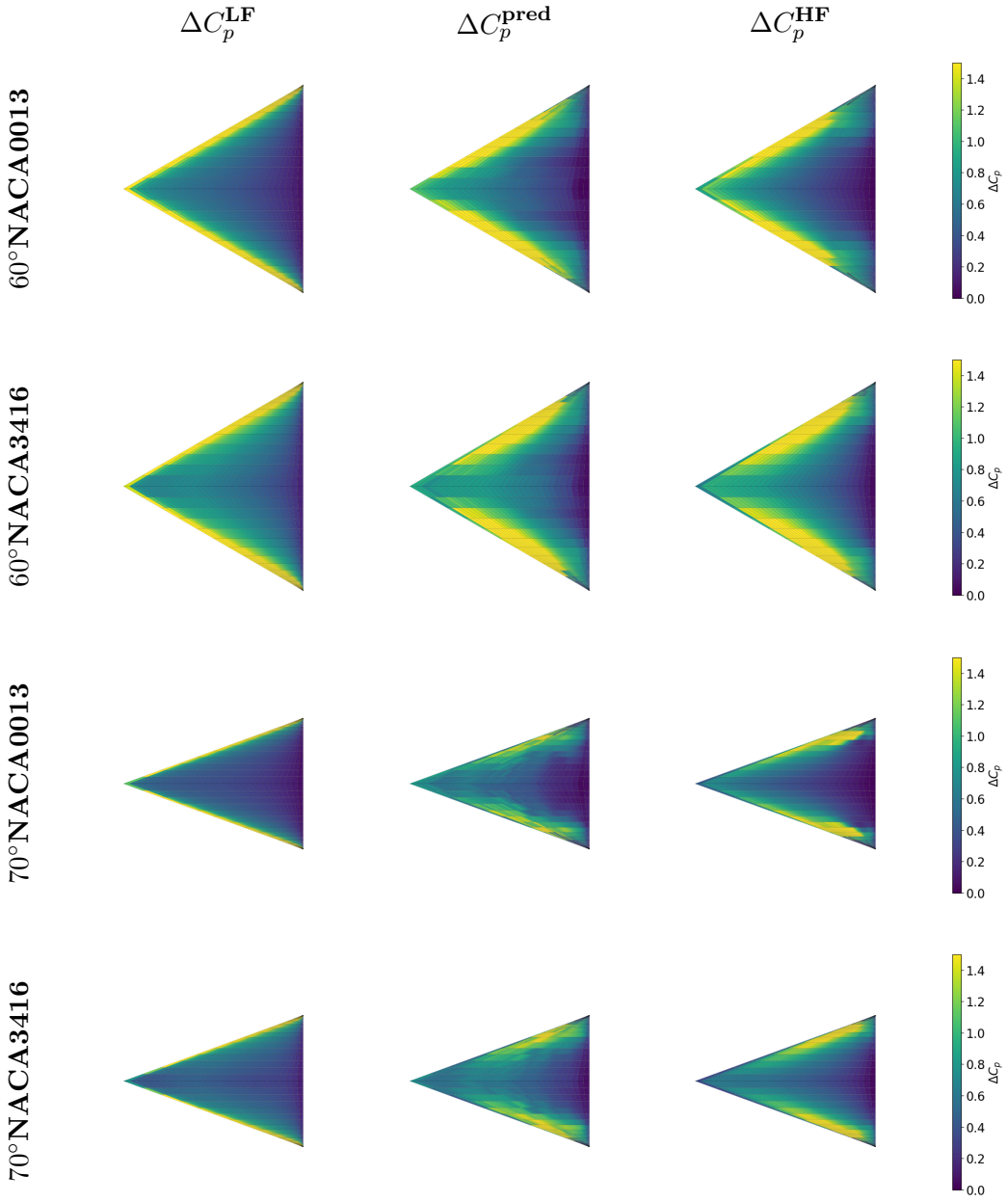


Figure 3.4: Lattice panels and local loading coefficients ΔC_p are visualized using color maps ranging from $\Delta C_p = 0$ to $\Delta C_p = 1.5$ for four new geometries. From top to bottom: 60° NACA 0013, 60° NACA 3416, 70° NACA 0013, and 70° NACA 3416. From left to right: ΔC_p^{LF} obtained from the VLM, `VortexNet` ΔC_p^{pred} predictions, and the projected CFD ΔC_p^{HF}.

Aerodynamic Coefficients Predictions for Seen Geometries

Our first effort to quantify the proposed workflow's QOIs prediction accuracy compares QOIs integrated from the surrogate model predictions ΔC_p^{pred} with those integrated from the projected CFD surface pressures ΔC_p^{HF}, which represent `VortexNet`'s modeling error, and QOIs obtained directly from CFD, which contain both the epistemic discretization error and the surrogate model's modeling error. These QOIs are compared against the results computed directly by VLM, which serves as the LF benchmark. We examine the C_L and C_M polars over an AOA range from 5° to 25° in 2° increments. The interval from 20° to 25° lies outside the training set range (0°–20°; see Section 3.3) and is intentionally included to assess model robustness under mild extrapolation. Although the drag coefficient can be computed using VLM, the absence of viscous effect modeling renders cross-comparisons between VLM and CFD misleading. Therefore, this study focuses exclusively on lift and moment coefficient evaluations. The free-stream Mach number is held constant at 0.4, and the Reynolds number at 7.0×10^6.

Figure 3.5 shows the relative error for C_L and C_M with AOA obtained from VLM for each individual Delta wing geometry against the reference QOIs obtained from CFD. The relative error is defined as the percentile difference between the QOIs from the lower-fidelity source and the higher-fidelity source. For each figure, the relative error is styled by the underlying geometry, such that data series with circle, square, and diamond markers indicate the sweep angle of 55°, 65°, and 75°, respectively. The color of the data series, from blue, purple, green, orange, to red, indicates root airfoil shapes of NACA 0010, NACA 0016, NACA 0024, NACA 2416, and NACA 4416, respectively. The solid black line corresponds to the average relative error across all geometries. Across AOAs, the average error increases, with the mean average error being 20.9% and 24.1% for C_L and C_M. The error presented in Figure 3.5 serves as the benchmark for LF error. Figure 3.6 presents the QOIs relative errors obtained from `VortexNet`, with an identical plotting framework to Figure 3.5.

When comparing the QOIs predictions errors from `VortexNet` (Figure 3.6) against those from the VLM (Figure 3.5), the surrogate model demonstrates strong agreement with the CFD references and provides significantly better predictions than those from the VLM. The mean errors are significantly lower than those reported by the VLM, being -1.3% and 0.3% for C_L and C_M, respectively. Across AOAs, the average error obtained from `VortexNet` shows smaller variation and remains relatively close to zero: the average C_L error ranges from -3.1% to 2.8%, and the C_M error ranges from -1.6% to 5.8%. In

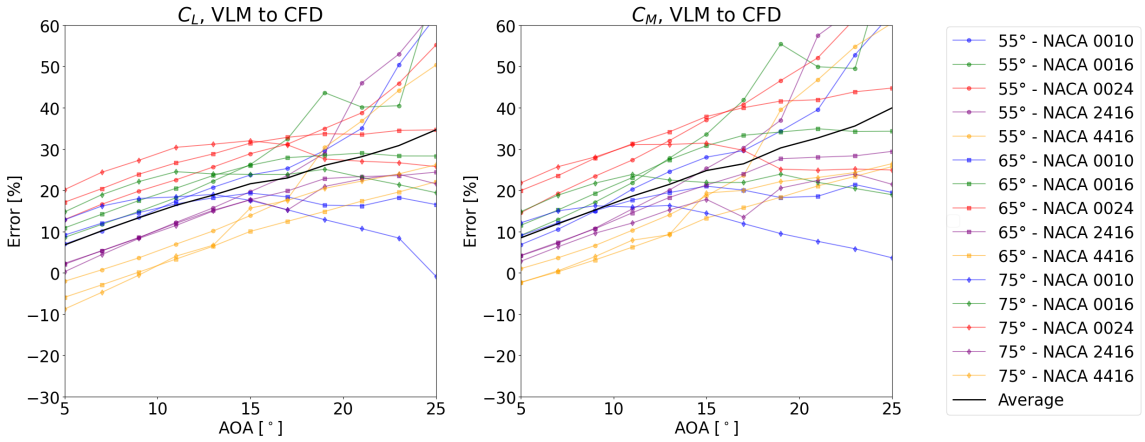


Figure 3.5: Relative errors for C_L and C_M from the VLM against the CFD reference.

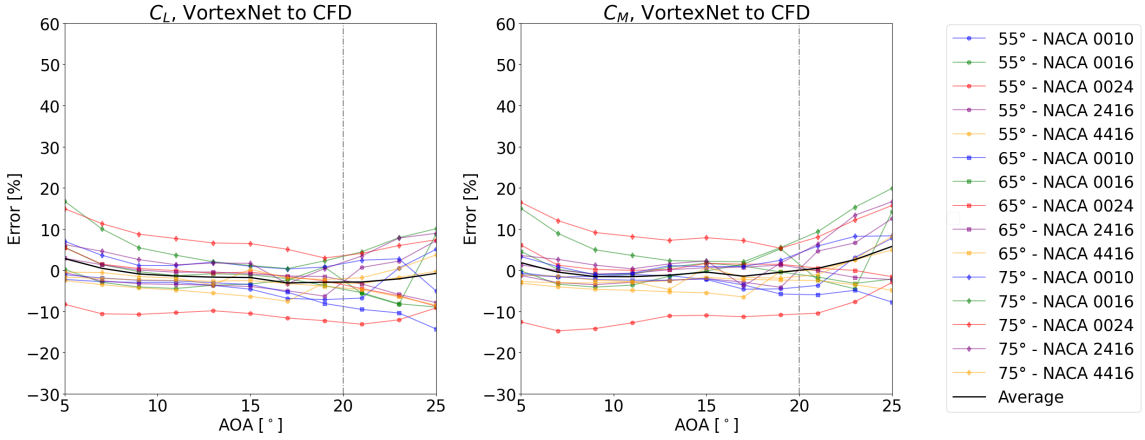


Figure 3.6: Relative errors for C_L and C_M from `VortexNet` against the CFD reference. The region to the right of the dash-dotted line (20°) denotes extrapolation beyond the training range.

contrast, the QOIs obtained from VLM show strong dependence on AOA, with the largest average errors observed at 25°, reaching 34.7% and 40.0% for C_L and C_M, respectively. In addition, for each individual geometry, the errors obtained from `VortexNet` exhibit a much weaker dependency on AOA compared to the VLM results, and the inter-geometry error spread is also reduced. Such a characteristic is desired for stability constrained conceptual design, as these error bounds can be incorporated into the design framework. Within the extrapolated region (to the right of the dash-dotted line at 20° in Figure 3.6), the average errors for both QOIs increase. However, the deviation remains largely monotonic, and

both the error magnitude and the spread across geometries are smaller than those obtained from the VLM (Figure 3.5). These improvements, evident in both the trend-wise capturing and the alignment of absolute values for the QOIs, significantly enhance the fidelity of aerodynamic analyses for these platforms.

As the relative error describes the prediction discrepancy for individual samples, using it for this comparison across all free-stream conditions will result in a cumbersome error representation. Instead, we rely on the normalized root-mean-square error (NRMSE) to describe the error, defined as:

$$\text{NRMSE}(f^{\text{method}}) = \frac{1}{\overline{f^{\text{HF}}}} \sqrt{\frac{1}{N} \sum_{i=1}^{N} (f_i^{\text{method}} - f_i^{\text{HF}})^2}, \tag{3.13}$$

where f_i^{method} is a QOI evaluated by a specific method, f_i^{HF} is the reference HF QOI, $\overline{f^{\text{HF}}}$ is the mean of the reference QOI values, and N is the total number of samples in the set. The NRMSE thus represents the deviation of the predicted QOI from the reference value, normalized by the average reference value, allowing for comparison across QOIs.

To highlight the fidelity gain of adopting `VortexNet` in the multi-fidelity workflow, we computed the NRMSEs for QOIs computed using the projected CFD, the `VortexNet` surrogate model predictions, and the VLM, all compared against the CFD reference values. The NRMSEs are grouped by geometries and workflow methods so that, for each wing geometry under a given method, the NRMSE represents an average deviation of predictions across all AOAs, including the small extrapolation region, against CFD references. The results are shown in Figure 3.7, with C_L and C_M listed from left to right. The VLM-only errors are displayed in the top row, the `VortexNet` surrogate-augmented errors in the middle row, and the projected CFD errors are shown in the bottom row.

On average, the epistemic discretization error induced by calculating QOIs using projected surface pressures is 3.8%. An examination of the ΔC_p^{HF} field reveals that the majority of this discretization error originates near the wing apex at the central body. This region exhibits large errors due to the use of span-wise cosine spacing, which results in larger panel sizes and excessive averaging of pressure fields with high gradients. The average NRMSEs for C_L and C_M are 4.6% and 3.0%, respectively.

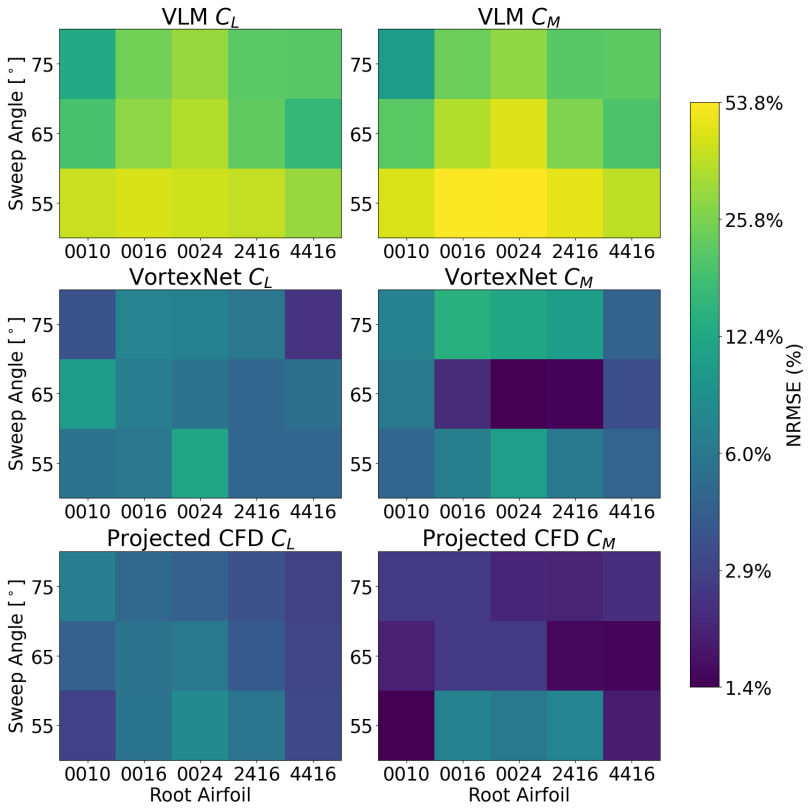


Figure 3.7: NRMSEs for C_L and C_M (from left to right, column-wise), comparing QOIs computed by VLM, integrated QOIs from the surrogate model (ΔC_p^{pred}), and QOIs computed from ΔC_p^{HF}, all against the CFD reference. The results are grouped by geometries, with NRMSE values represented by color.

The QOIs computed using the surrogate model have an average NRMSE of 6.1%. Considering that the minimum QOI prediction error using the surrogate model is the epistemic discretization error discussed earlier (since the projected CFD ΔC_p^{HF} is considered as "ground truth" for the surrogate model during training), the fact that this test returns an error close to the epistemic discretization error indicates that the surrogate's modeling error is low. The corresponding NRMSEs for C_L and C_M are 6.0% and 6.3%, respectively. In comparison, the VLM method has an average NRMSE of 30.2%. Specifically, the NRMSEs for C_L and C_M are 28.1% and 32.3%, respectively. Although the database in this test contains extrapolation samples for the surrogate model, the average NRMSE of the QOIs from the proposed mixed-fidelity workflow is still smaller than that of VLM by almost fivefold.

Aerodynamic Coefficients Predictions for Unseen Geometries

Our final test case assesses the accuracy of the proposed workflow in predicting QOIs for unseen geometries. By comparing the QOI accuracy against the results presented in the previous section, the model's geometric generalizability can be assessed. We select the four new Delta wing geometries presented in Section 3.4.1. Specifically, two sweep angles, 60° and 70°, and two root airfoils, NACA 0013 and NACA 3416, are used. The free-stream conditions match those reported in Section 3.4.2.

Similar to the relative error comparison in Section 3.4.2, the relative error for the VLM QOIs and `VortexNet` QOIs against the CFD reference is plotted in Figure 3.8 and Figure 3.9. For each figure, the relative error is styled by the underlying geometry, such that data series with circle and square markers indicate the sweep angle being 60° and 70°, respectively. The color of the data series, from blue to red, indicates the root airfoil shape, ranging from NACA 0013 to NACA 3416.

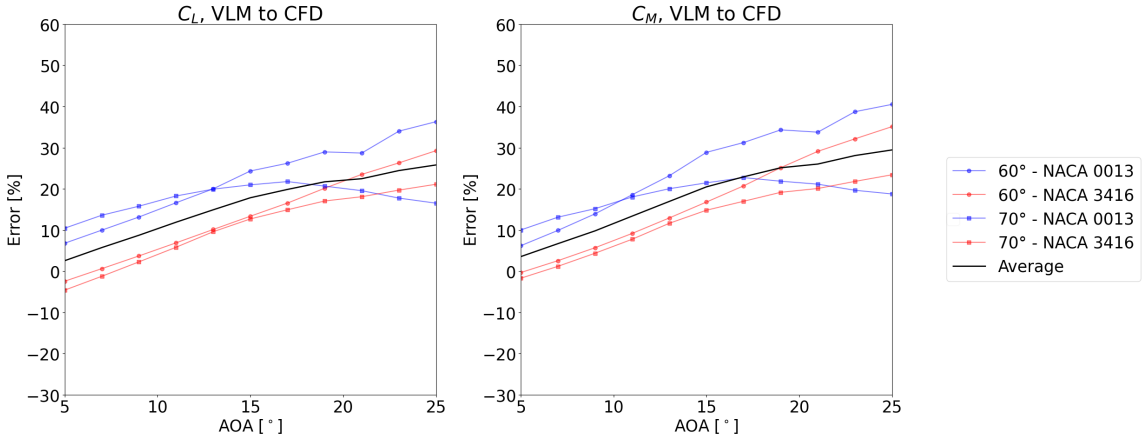


Figure 3.8: Relative errors for C_L and C_M from the VLM against the CFD reference for four unseen geometries.

For all unseen geometries, we observe prediction quality and trend-wise characteristics similar to those in Section 3.4.2. The average relative errors from the VLM are 16.0% for C_L and 18.4% for C_M, while `VortexNet` reduces these to -2.0% and -0.7%, respectively. Beyond improving mean accuracy, `VortexNet` also decreases the dispersion of QOIs across geometries and reduces the dependency of QOI error on AOA. Overall, fidelity improvement over the VLM results is observed.

Finally, we compare the NRMSEs for QOIs obtained from the VLM, the `VortexNet`

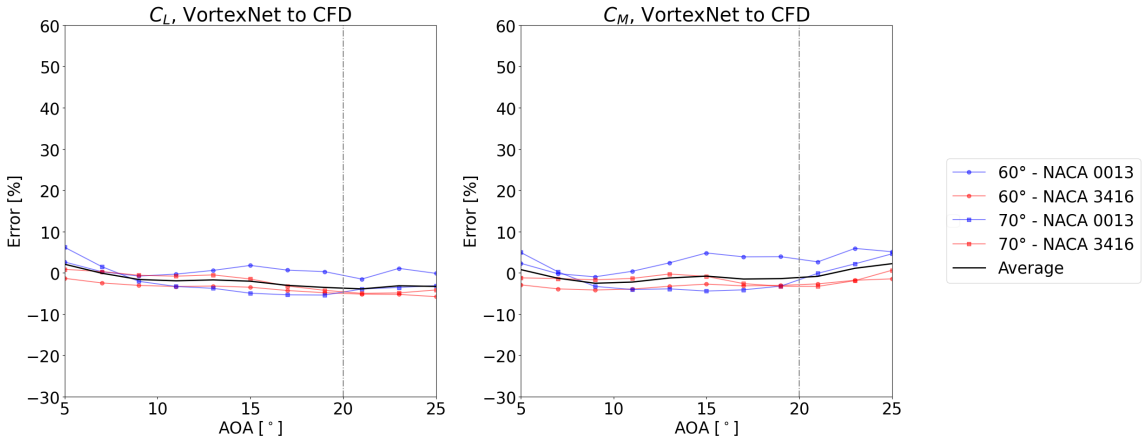


Figure 3.9: Relative errors for C_L and C_M from `VortexNet` against the CFD reference for four unseen geometries. The region to the right of the dash-dotted line (20°) denotes extrapolation beyond the training range.

surrogate model, and projected CFD against the CFD references. The results are presented in Figure 3.10. On average, the epistemic discretization error induced by calculating QOIs using projected surface pressures is 2.9%, which does not deviate much from the error quantification in Section 3.4.2. The average NRMSEs for C_L and C_M are 4.1% and 1.7%, respectively. Although this error is lower than the corresponding average error in Section 3.4.2, the reduction is most likely due to the fact that the current test involves a database that contains fewer geometric variations.

When `VortexNet` is used for QOI evaluation, we observe an average NRMSE of 3.4%. The NRMSEs for C_L and C_M are 3.5%, and 3.4%, respectively. The difference between the surrogate model-predicted NRMSEs and the projected CFD NRMSEs is comparable to the results of the previous test in Section 3.4.2. We conclude that the workflow does not present significant deviations in QOI prediction accuracy when applied to unseen geometries. For comparison, the average NRMSE for QOIs predicted by the VLM is 24.0%, with NRMSEs for C_L and C_M being 22.4% and 25.7%, respectively. When comparing the average NRMSE obtained from `VortexNet` with that computed by the VLM, a sevenfold reduction in error is observed.

Across all assessments, the proposed surrogate model outperforms the VLM-based approach in QOI prediction accuracy. Additionally, a significant performance degradation is not observed for this dataset after introducing new geometries. This observation supports the proposed workflow's generalizability potential in accommodating geometry variation

within the design space. However, a more detailed sensitivity study with a richer design space inclusion is needed in the future to demonstrate the proposed workflow's effectiveness for more complex vehicle conceptual design.

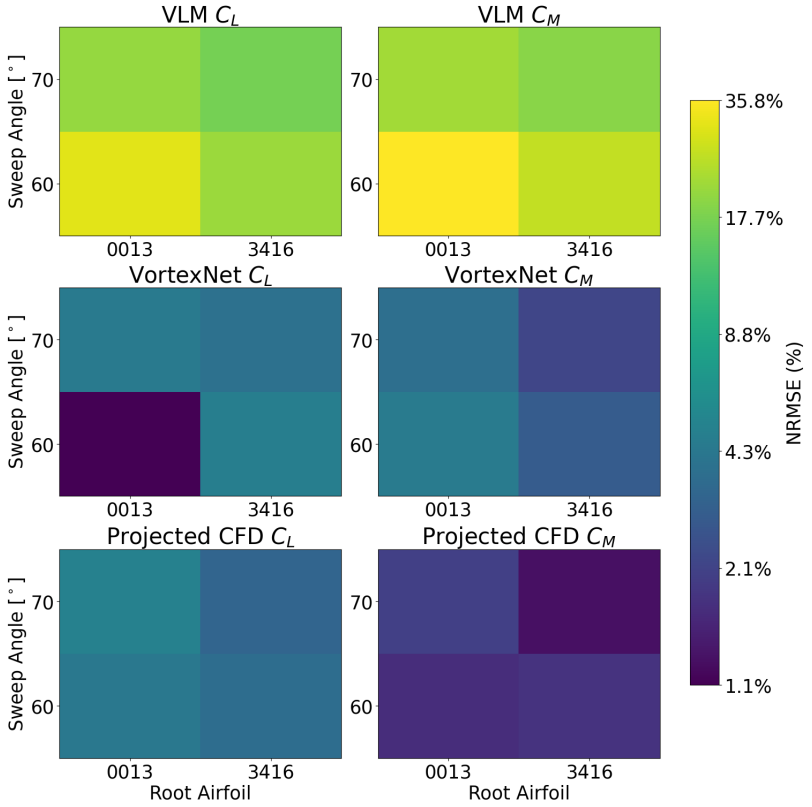


Figure 3.10: NRMSEs for C_L and C_M (from left to right, column-wise), comparing QOIs computed by the VLM, integrated QOIs from the surrogate model (ΔC_p^{pred}), and QOIs computed from ΔC_p^{HF}, all against the CFD reference.

3.5 Prediction Explanations

While the proposed `VortexNet` demonstrates strong prediction accuracy and generalizability, the underlying mechanism by which the GNN leverages and aggregates input graph features remains unclear. The complexity of GNNs makes extracting human-intelligible explanations from trained networks challenging [151]. Nevertheless, the authors have conducted a preliminary analysis of the `VortexNet`'s latent space. This section aims to understand how input graph features are utilized across latent layers and provide guidance for

refining future `VortexNet`-like GNN architectures.

Among methods for latent space analysis, we performed an ablation study on latent space features to investigate the resultant ΔC_p predictions. For the data sample, we selected a representative test case featuring a 55° NACA 4416 wing at an AOA of 10.9° and a Mach number of 0.49, as this combination exhibits richer nonlinear fluid dynamics and is therefore well suited for latent space analysis. The study begins with the nodal feature latent space obtained from the output of the first Convolutional Block shown in Figure 3.1. At this stage, the nodal feature array was recorded, and clustering was performed on the feature array. Since the node features exist in a high-dimensional space ($\mathbb{R}^{\mathbb{F}\times\texttt{heads}}$), the t-SNE dimensionality reduction technique was applied to project the feature space to three dimensions [139]. K-means clustering was then used to classify nodes into four clusters [75], and their corresponding nodal indices were recorded. From the clustering results, we observe that different clusters correspond to distinct regions of the wing. Nodes belonging to "Cluster 1" and "Cluster 2" correspond to the right and left sections of the wing afterbody. Nodes within "Cluster 3" correspond to the central forebody, and nodes in "Cluster 4" correspond to the leading-edge region of the wing. This clusters-to-wing-region association is presented in Figure 3.11.

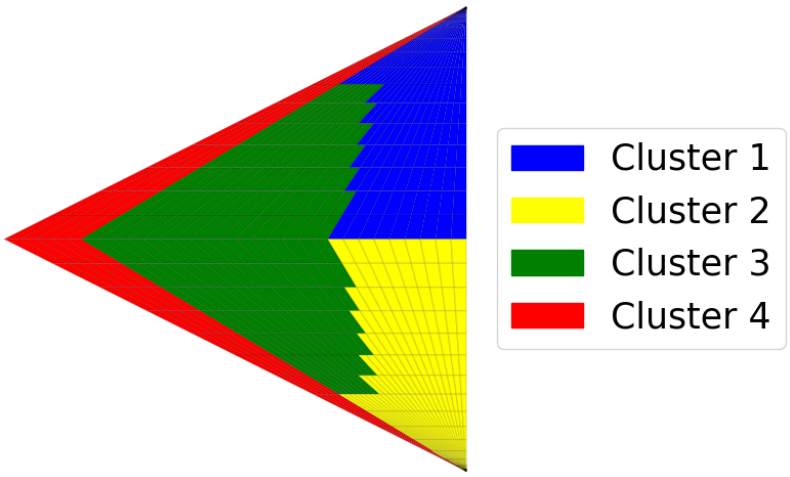


Figure 3.11: Association between nodal feature clusters extracted from the first Convolutional Block and their corresponding spatial locations on the wing surface.

Using the recorded node-to-cluster assignments, the ablation study proceeds by recursively isolating the contribution of each cluster. At a given layer, nodal feature vectors associated with all other clusters are set to zero, thereby retaining only the features of the cluster under investigation. This procedure is repeated across all Convolutional Block layers up to the final prediction layer.

For example, when analyzing the contribution of Cluster 1 after the fourth Convolutional Block (Layer 5), we first extract the nodal feature tensor from the output of Layer 5. The feature vectors corresponding to nodes in all other clusters are then manually zeroed out. The modified global nodal feature tensor is subsequently propagated through the remaining Convolutional Blocks to produce a final modified prediction, denoted as ΔC_p^{mod}. This procedure enables the isolation of the contribution of specific wing-region groups to the overall ΔC_p prediction. By examining the cluster-specific predictions across multiple layers, the progressive effects of multi-hop graph convolution can be analyzed.

The results of the ablation study are presented in Figure 3.12. The figure consists of two parts. In the left panel, the ablation errors corresponding to each of the four clusters identified in Figure 3.11 are shown in separate columns. The ablation error is defined as the difference between ΔC_p^{mod}, obtained by isolating a specific cluster at a designated layer, and the original `VortexNet` prediction ΔC_p^{pred} computed using the unmodified nodal feature tensor. This ablation error quantifies the influence of a given cluster, ablated at a specific layer, on the final model output.

While the ablation error across all layers is studied, we present selected representative layers rather than all intermediate layers, which would introduce excessive figures without additional insight. Specifically, the ΔC_p^{mod} results for ablations performed at Layer 1 (after the first Convolutional Block), Layer 5 (the intermediate layer, analogous to a bottleneck in an encoder–decoder architecture), and Layer 9 (the final feature prior to the Output Decoder) are plotted. These layers represent key stages in the feature transformation process.

In the right panel, the relationships among nodal features across clusters are visualized. Principal Component Analysis (PCA) is used to project the corresponding nodal features across graphs onto a two-dimensional space visualized by the first two principal components. The scatter points corresponding to each cluster are colored consistently with the color scheme used in Figure 3.11.

From the ablation error results, it can be observed that learned field features become increasingly localized across network depth. At shallow layers, removing nodal features

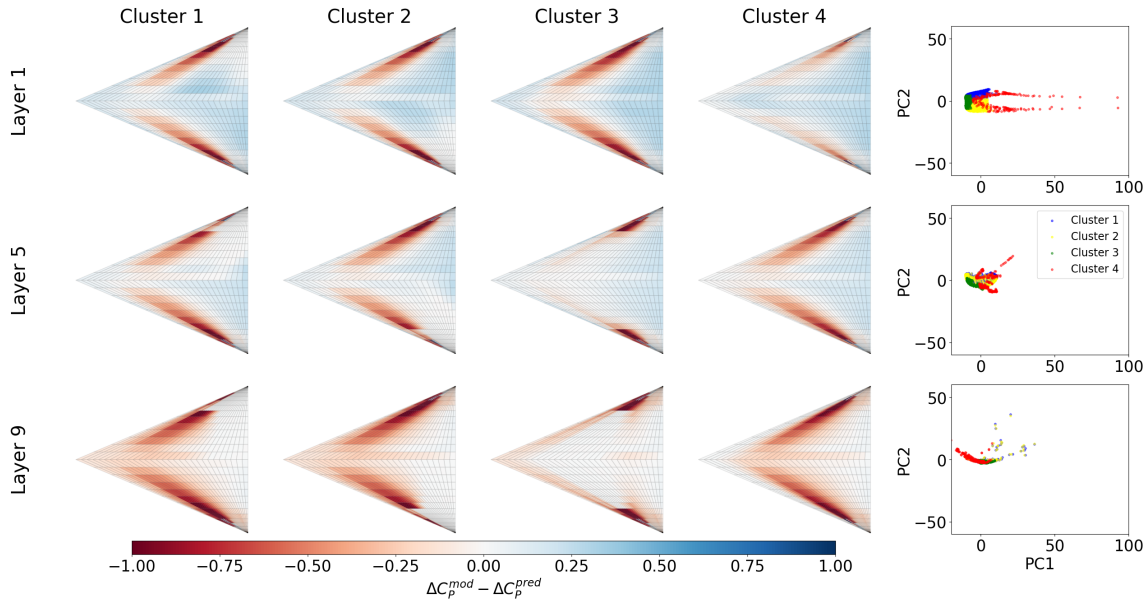


Figure 3.12: Ablation study results illustrating ablation error ($\Delta C_p^{\text{mod}} - \Delta C_p^{\text{pred}}$) for node features within each cluster (columns), and the projected latent space nodal features plotted against the principal axes, with scatter dot colors indicating the corresponding clusters of the nodes. The rows represent the specific layers where the ablation study is conducted.

from a specific region introduces global effects that extend beyond the corresponding spatial cluster. This behavior is evident in the right and left wing afterbody regions when examining ΔC_p^{mod} at Layer 1 for Cluster 1 and Cluster 2. The removal of surrounding nodal features produces substantial errors, shown as significant negative pressure deviations in regions near the leading edge that extend inside the original clusters where nodal features are kept. A similar phenomenon is observed for the ablation error associated with Cluster 3. Interestingly, the ablation error obtained when preserving only Cluster 4 (the leading-edge region) is smaller than the errors corresponding to the other clusters. This suggests that, at shallow layers, the leading-edge region contains more globally informative features that `VortexNet` leverages for pressure field reconstruction. This conjecture is further supported by the distribution of ΔC_p^{LF}, where the highest pressure magnitudes and sharpest gradients are concentrated near the leading edge.

At the intermediate layer (Layer 5), removing specific wing regions no longer induces substantial global error variations. The ablation errors associated with the left wing afterbody from Cluster 1's ΔC_p^{mod}, the right wing afterbody from Cluster 2's, and the central

afterbody from Cluster 4's show comparable magnitudes and do not differ significantly. This suggests that a coherent global pressure field representation has already been established by this stage of the network, and truncating features from individual regions primarily produces localized effects. Such localization is further reflected in the sharp spatial boundaries of the ablation errors for each cluster. For Cluster 1 and Cluster 2, we no longer observe the cross-region error propagation seen in earlier layers; eliminating nodes outside a region does not strongly affect predictions within that region. This trend is particularly apparent for Cluster 3, where the ablation error exhibits a clear cutoff along two outboard sections, resembling the geometric extent of the cluster itself, as shown in Figure 3.11. This localization effect becomes even more pronounced in deeper layers. At Layer 9, truncating features from a specific region affects only that eliminated region and produces negligible influence on the preserved regions. The preserved regions exhibit near-zero ablation error, indicating that ΔC_p^{mod} within those regions is effectively identical to the original prediction ΔC_p^{pred} obtained without truncation.

This progressive localization of ablation error across layers is consistent with the message-passing mechanism of GNNs, in which nodal features are iteratively aggregated from neighboring nodes at each hop. For a preserved region to achieve accurate predictions, it must incorporate information from adjacent regions, especially from the leading edge region in the present study. Once this information propagates into the preserved region, it is further aggregated locally to refine the field prediction. Because `VortexNet` is built upon a GNN architecture, information propagation across the graph is governed by the number of message-passing hops, implemented as stacked Convolutional Block layers. For larger graphs with finer panel discretizations, deeper networks would therefore be required to ensure that information from physically informative regions can traverse sufficiently across the graph and influence distant nodes.

However, deeper networks are more challenging to train and prone to high-variance error. To address this limitation, improved edge definitions that connect nodes beyond direct neighbors could be implemented. Alternatively, using a fully connected graph may help mitigate this issue, though the implications of adopting such a graph remain unclear as the addition of trainable parameters may decrease the prediction accuracy under limited HF samples. Additionally, since pooling layers are not utilized in this network, we believe that incorporating a properly designed pooling and unpooling mechanism, such as the one proposed by Immordino et al. [46], could assist information propagation by reducing the

computational graph size toward deeper layers. This highlights the need for future research on optimizing graph definitions and model architectures in `VortexNet`-like surrogate modeling techniques.

Finally, we examine the feature representations shown in the right panel of Figure 3.12. In the shallower layers, the distribution of nodal feature clusters loosely follows the positional distribution of the corresponding nodes in physical space. For example, the relative separation between the left and right afterbody sections can still be identified along the principal axes. As depth increases, this grouping by physical proximity diminishes. By Layer 5, the majority of nodes collapse into a single dominant cluster, with only a subset of leading-edge nodes remaining separated. This behavior suggests that, at this stage, features associated with flow physics become more influential than purely geometric location. Because the left and right outboard sections experience symmetric free-stream conditions, their physical separation becomes a less dominant organizing factor in clustering. Additionally, from Layer 1 to Layer 5, the overall aspect ratio and spatial spread of the nodal feature distribution decrease, as evidenced by the reduced extent of the combined cluster manifold. This contraction of the latent space suggests a potential mechanism underlying `VortexNet`'s generalizability: by nonlinearly mapping the input geometry–flow state into a more compact latent representation, the network facilitates smoother interpolation across nodes. Such compactness may enable consistent field prediction across varying geometries and flow conditions. At the last layer, the clusters re-expand while preserving the left–right symmetry between Cluster 1 and Cluster 2, further supporting the conjectured mechanism between geometric structure and flow-conditioned feature organization in the learned latent space.

3.6 Discussion

This chapter presents `VortexNet`, a GNN-based multi-fidelity surrogate model for aircraft conceptual design. The model uses low-fidelity VLM field predictions as input and learns to generate quasi-high-fidelity pressure field predictions informed by CFD data. Unlike traditional surrogate models that predict only scalar quantities of interest, `VortexNet` predicts field data, allowing arbitrary aerodynamic quantities to be evaluated within a conceptual design workflow at a cost comparable to low-fidelity analysis.

The chapter also describes the ML model architecture, data curation, and training

strategy of `VortexNet`. The model combines U-net-style skip connections for global feature propagation with GAT convolutional blocks for capturing local flow features. Data assembly methods, implicit physical constraint enforcement, and tailored training schemes are designed, experimented, and introduced to support data-efficient field-based multi-fidelity learning across a representative conceptual design space for delta wings.

The model is evaluated through qualitative field reconstruction, quantitative error metrics, integrated aerodynamic coefficient prediction, and generalization tests on unseen geometries. Results show that `VortexNet` substantially improves pressure field and aerodynamic coefficient predictions relative to the low-fidelity solver, including a five- to seven-fold reduction in NRMSE for lift and moment coefficients. Although prediction quality decreases for unseen geometries, the model demonstrates promising geometric generalization capability for design space exploration. A latent space ablation study further shows that the model learns by jointly encoding geometric and flow features into a compact, interpolatable latent representation.

Chapter 4

The Multi-Fidelity Aerodynamics Dataset

As discussed in Section 1.5, in the open domain, multi-fidelity aerodynamic datasets for surrogate model training and benchmarking are limited. Building on the demonstrated effectiveness of `VortexNet`, as discussed in Chapter 3, a more SST-relevant planform dataset can support downstream applications, such as model scaling analysis and aerodynamic shape optimization, while also providing an additional benchmark problem for the community.

This section details the double-delta multi-fidelity aerodynamic dataset. We first define the parametric double-delta wing geometry and its six-dimensional design space. Next, we describe the design of experiments (DOE), which utilizes a nested Saltelli sampling scheme to ensure scalable sampling. Finally, we present our numerical simulation framework and results, including the automated meshing pipeline and solver configurations for generating paired VLM and CFD data.

4.1 Geometry Definition

Six design variables are used to define the double-delta wing shape. Figure 4.1 shows the half-wing about the central-body symmetry plane. The geometry has six design parameters, including B, the wing span (in inches); BW2, the fractional location of the inboard-to-outboard break-chord relative to the semi-span; SW1 (Λ_{in}), the inboard leading-edge sweep angle; SW2 (Λ_{out}), the outboard leading-edge sweep angle; SR2 (Λ_{te}), the outboard trailing-edge sweep angle; and DROOP (δ), the camber-deflection angle applied to the baseline

airfoil. The baseline airfoil is constructed from the root-airfoil geometry of the RW23 wing presented by Wiersema et al. [147] and is applied uniformly across the span.

To allow a richer representation of camber variation within the design space, δ acts by deflecting the baseline airfoil's leading edge camber about the 30% chord from leading edge. The airfoil morphing process begins with cubic-spline fitting of the baseline airfoil's upper and lower surfaces from their discrete coordinates. The camber and thickness distributions are then extracted on a shared chordwise grid. The droop morphing is subsequently applied to the baseline camber, introducing C^1 and C^2 parametric discontinuities [38] at the pivot point and producing two separate Bézier curves for the leading and trailing cambers, respectively. To ensure smoothness, the Bézier control-point parameters for the leading camber line are solved using a sequential least-squares programming (SLSQP) solver in `SciPy` to minimize the C^1 and C^2 continuity residuals between the leading- and trailing-camber curves at the pivot point. The modified camber is then combined with the original thickness distribution to reconstruct the upper and lower surfaces, with an additional round of smoothing applied to both surfaces' curves.

The spanwise twist distribution is defined as follows: the inboard section, from the root to the break-chord location, maintains a constant twist of 1.5° leading edge up, while the outboard section applies a linear washout from 1.5° to −1.5° toward the tip.

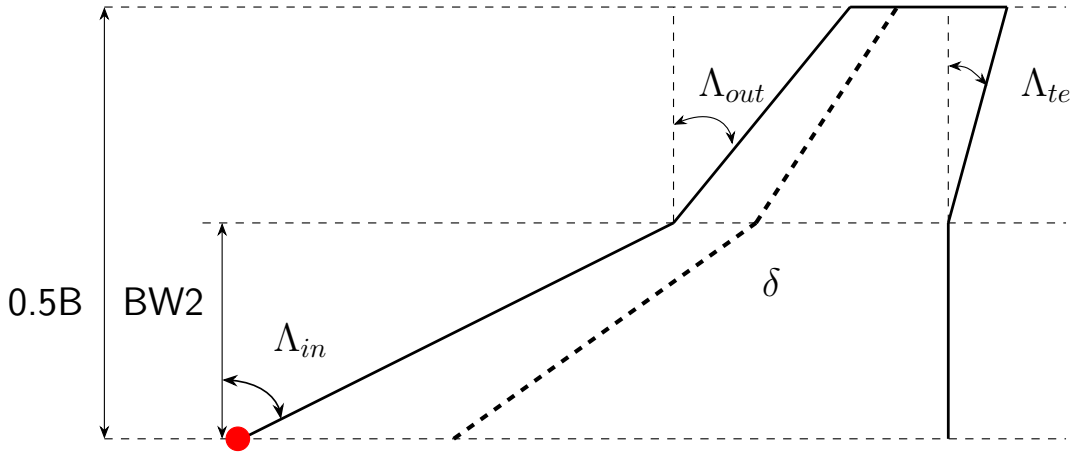


Figure 4.1: Geometry definition and its design variables.

4.2 Design Space

The design space $\mathbf{x}$ is defined as $\mathbf{x} = [\delta, \Lambda_{in}, \Lambda_{out}, \Lambda_{te}, BW2, B]^T, x_i \in [x_i^{min}, x_i^{max}]$, where x_i^{min} and x_i^{max} are the bounds. The selection of the design variable bounds is based on first referencing the design space of recent commercial supersonics technology program, including Supersonic Configurations at Low Speeds (SCALOS) [69], and High-Speed Civil Transport (HCST) [36]. From these studies, the nominal inboard and outboard leading edge sweep (Λ_{in} and Λ_{out}) is 68° and 45° respectively. The aspect ratio of the wing platform is taken to be 2.5. From these nominal values, we consider the design space included in SCALOS program [69], and adjust the design variable bounds so that the created design space does not produce non-physical designs. Specifically, design variable combinations that lead to negative tip chord under our DOE. This constraint reduces the design space represented in this dataset, resulting in a Λ_{in} upper bound smaller than those explored in the SCALOS program. However, the constraint ensures that the resulting samples are balanced, as no non-physical designs are removed from the samples.This design space is presented in Table 4.1.

Design Variable	Lower	Upper	Nominal
δ (°)	-8	7	0
Λ_{in} (°)	47	67	65
Λ_{out} (°)	35	60	45
Λ_{te} (°)	0	25	10
BW2	0.32	0.47	0.4
B (inches)	1100	1300	1200

Table 4.1: List of design variables and their corresponding upper and lower bounds.

4.3 Design of Experiments

4.3.1 Sampling Methods

The DOE involves the selection of combinations of design variables to generate geometry samples. We have three major goals for the DOE. Firstly, we aim to achieve efficient and homogeneous design-space filling, such that the sampling scheme does not introduce sampling bias that leads to clustered design samples within the design space, which could bias the performance of subsequently trained surrogate models. Secondly, the sampling

scheme should produce an extendable dataset, such that when the dataset needs to be expanded, new sampling points can be added without requiring existing experiments to be re-run. This property is particularly important when the experiment is costly, such as the CFD simulations in this study. Finally, we want the sampling scheme to support future downstream assessments of the surrogate models trained from the dataset, including applications involving sensitivity analyses or uncertainty quantification.

Popular sampling methods are considered, including factorial sampling, random sampling, Latin Hypercube Sampling (LHS), and quasi-random sequences. The quasi-random sequence sampling methods, particularly Sobol sequence, meet the previously-mentioned DOE goals better. Sobol sequence constructs a sequence of spatial quadrature points that enable fast convergence of n-dimensional integral, that is, to find a sequence x_i with length P such that,

$$\frac{1}{P}\sum_{i=0}^{P-1} f(x_i), \tag{4.1}$$

approximates the integration

$$\int_{[0,1]^n} f(x)dx. \tag{4.2}$$

Solving for the Sobol sequence directly is costly, and modern software packages use fast Sobol sequence algorithms in which subsequent Sobol numbers are generated by XOR-ing previous Sobol number with a predefined table of direction numbers [48]. For the current DOE, we adopt `SALib` [42], which contains a Sobol sequence generation routine by Joe and Kuo [48]. For a required P samples in n dimensional space, the Sobol sequence generator generates a base matrix S of size $(P + \text{skipvalue}) \times 2n$. The skipvalue is applied to discard the first several rows of the Sobol sequence matrix, as repetitions may occur [18], thereby improving the uniformity of samples generated. For a predefined skipvalue and problem dimension, the generated Sobol sequence is deterministic. Throughout the experiments, we set the skipvalue to 16.

From the base matrix, we follow Saltelli's sampling scheme [109] using `SALib`'s `sample.saltelli` function with function call of `saltelli.sample(problem,skipvalue=16,calc_second_order=False)`. The process firstly constructs two sampling matrices A and B, such that

$$A[i,:] \leftarrow S[i, 0:n] \quad \text{and} \quad B[i,:] \leftarrow S[i, n:2n]. \tag{4.3}$$

Then assembling n cross-sampling matrices $AB^{(k)}$ by cross swapping columns of B to a

copy of the A matrix. Mathematically, this is equivalent to,

$$AB^{(k)} = (A[:,1], ..., A[:,k-1], B[:,k], A[:,k+1], ..., A[:,n]) \text{ for } k \in [1,n]. \tag{4.4}$$

The final design samples matrix $\tilde{X}$ in unit hypercube is constructed by concatenating all A, B, and AB matrices vertically, such that,

$$\tilde{X} = \begin{bmatrix} A \\ AB^{(1)} \\ \vdots \\ AB^{(n)} \\ B \end{bmatrix} \in \mathbb{R}^{P(n+2)\times n}. \tag{4.5}$$

The final design samples X can be obtained from $\tilde{X}$ by column-wise scaling using the upper-and-lower bounds of the corresponding design variables.

The adopted sampling scheme meets our DOE goals. Firstly, the Saltelli sampling scheme is deterministic and reproducible. By generating samples from a Sobol sequence, the method achieves uniform space-filling properties that minimize clustering and sampling bias, leading to balanced datasets ideal for machine-learning model training. Secondly, because the Sobol sequence is nested, doubling the dataset size yields a refined design-space coverage without modifying the existing design points, ensuring that dataset scaling does not require rerunning existing experiments. Moreover, the Saltelli framework naturally supports variance-based sensitivity analysis and uncertainty quantification, thereby enriching the downstream applications of the dataset.

The Saltelli sequence samples X are used exclusively for model training. Because Saltelli sampling enforces a homogeneous distribution, randomly partitioning X to form a test set would disrupt this structure and introduce bias into the remaining training samples. To avoid this issue, we generate a separate holdout test set independent of X with 16 geometries. The holdout test set is generated using LHS sampling via `SALib`'s `sample.latin` function [42]. The resulting unit-hypercube samples are subsequently scaled using the same procedure employed in the Saltelli sampling process described previously.

4.3.2 Sampling Results

As a result of the sampling scheme described, one can generate geometry configurations at different levels of density in a nested structure. We took six values of P, in multiples of 2, to generate six levels of datasets, shown in Table 4.2. Each geometry in the dataset is evaluated at nine AOAs, ranging from 11° to 19° in increments of 1°.

Level	**Base Sequence Length (P)**	**Geometry Configurations**	**Number of AOAs**	**Dataset Size (D)**
1	1	8	9	72
2	2	16	9	144
3	4	32	9	288
4	8	64	9	576
5	16	128	9	1152
6	32	256	9	2304

Table 4.2: Dataset sizes at six levels.

Together, these configurations constitute of the training geometry set of the dataset. Due to the nested structure of the sampling scheme, the set of geometries in the Level 1 dataset is a strict subset of those in the Level 2 dataset, and this nested structure is preserved across all higher dataset levels. The largest dataset is designated as the Level 6 set. All designs in the Level 6 set are shown in Figure 4.2(a) in top view, and the corresponding side view at the root in Figure 4.2 (b). In both figures, the blue edges denote the configurations included in the training geometry set, and the red edges indicate the configurations belonging to the holdout test set.

From Figure 4.2(a), it is evident that the training set spans a wide design space, ranging from low-aspect-ratio trapezoidal wings (with the largest tip-chord lengths) to delta wings (with the smallest tip-chord lengths). The sampling density is relatively uniform across the design space. Similarly, the holdout test set also exhibits a consistent sampling density, as indicated by the even distribution of red edges across the design space. Similarly, uniform design space coverage for δ can be observed in Figure 4.2(b).

To further illustrate the sampling density, Figure 4.3 presents the pair plots of the design variables for samples in the Level 6 set and holdout set. The off-diagonal panels show the scatter plots of paired design variables, while the diagonal panels show the histograms of the sampling density with respect to each design variable. Samples belonging to the training geometry set are plotted in blue, while samples corresponding to the holdout test set are

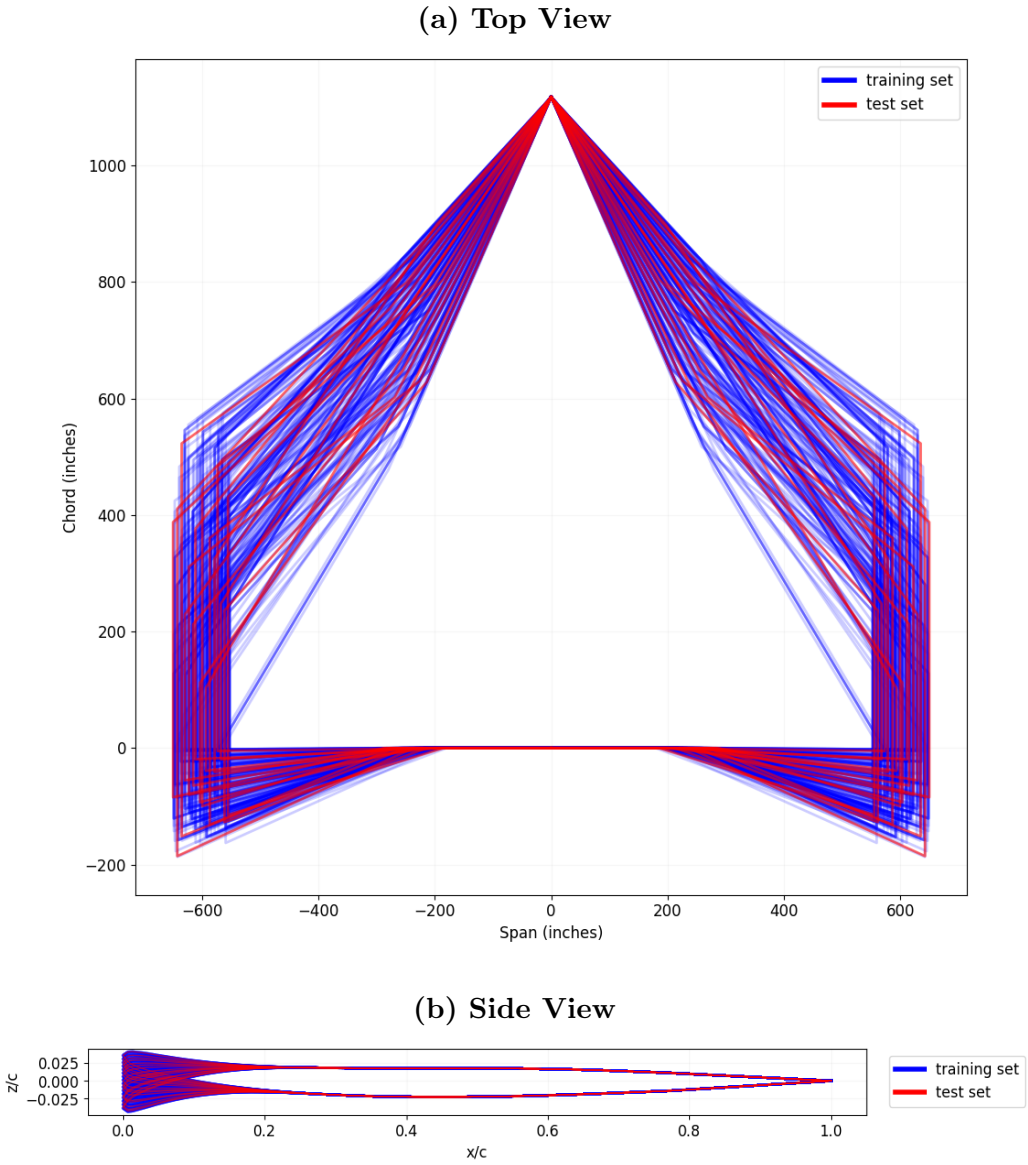


Figure 4.2: Illustration of vehicle configurations sampled from the design space. Training samples from the Level 6 set are shown in blue, while samples from the holdout test set are shown in red.

shown in red. As indicated by the paired plots, no specific clustering is observed across any design variable pairs, and within each design variable, a uniform sampling density is achieved across the range of design-variable bounds. When examining the distribution of the holdout test set, the test samples also span the design space evenly. Because the sampling scheme for the training set is nested, the sampling density distribution across training set levels is consistent, and hence the sampling density pair plots for the sparser sampling levels are not shown.

Configuration-specific design variables are listed in Appendix A. The Level 6 training set geometries are listed in Table A.1, and the LHS hold-out set geometries are listed in Table A.2.

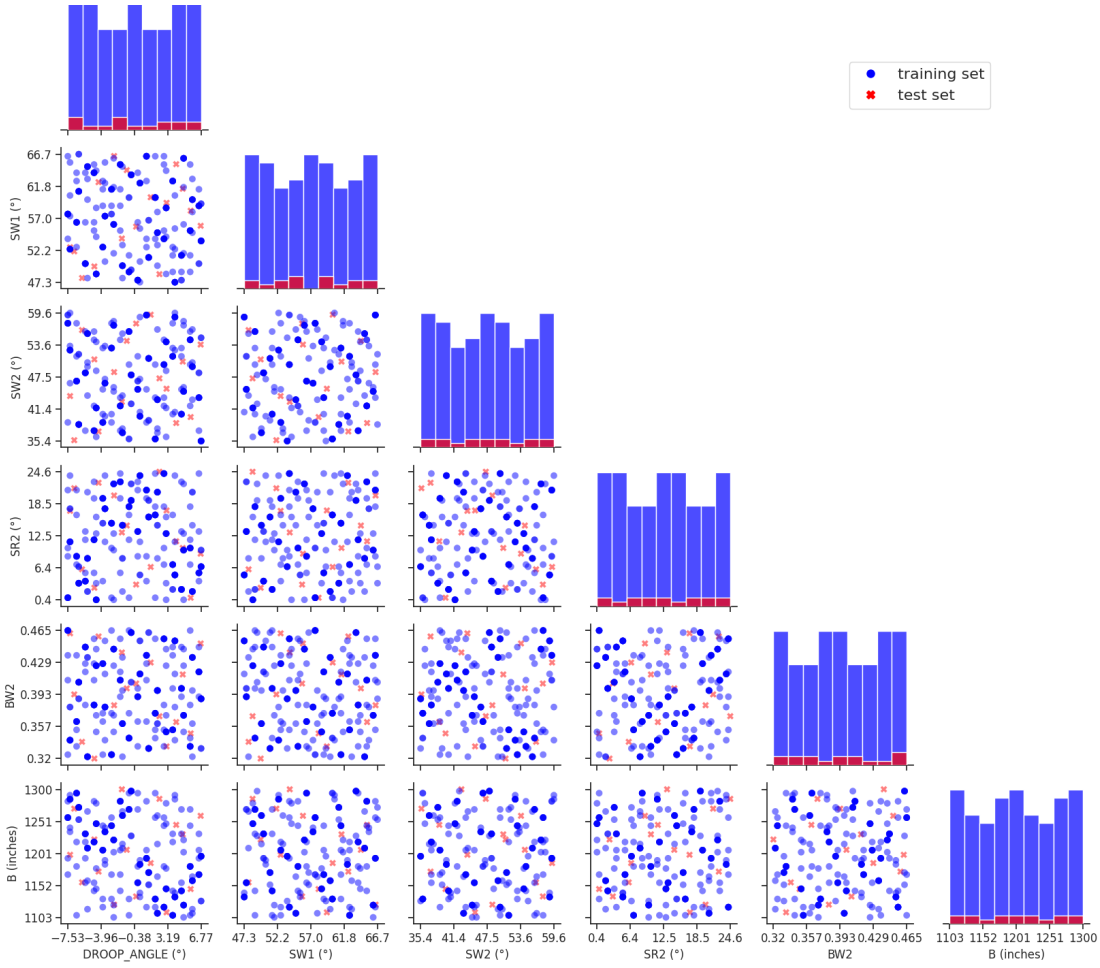


Figure 4.3: Design space coverage and design variable distributions of the dataset.

4.4 Numerical Simulation

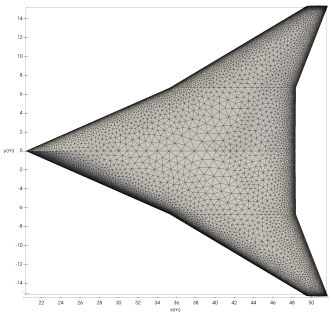
4.4.1 CFD Meshing

Vehicle geometries are generated from the design variable samples using `SUAVE`'s `write_vsp_mesh` interface to produce the corresponding `OpenVSP` geometry model, as discussed in Section 2.3. The CFD mesh generation involves two stages: surface meshes are first produced using `OpenVSP`'s `ComputeCFDMesh` function, and volume meshes are then created in `Pointwise` using its `T-Rex` extrusion function. The `T-Rex` procedure extrudes the surface mesh into the freestream to form an anisotropic boundary layer mesh with controlled growth and resolution suitable for RANS simulations.

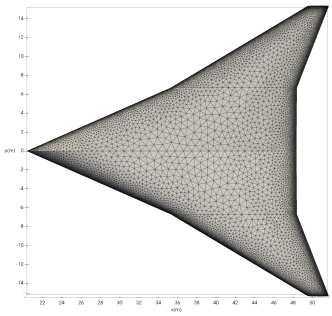
Several parameters control the mesh properties of `OpenVSP`'s `ComputeCFDMesh` surface meshes. These parameters include source definitions, the maximum mesh-edge length (enumerator `CFD_MAX_EDGE_LEN`), the maximum edge-growth ratio (`CFD_GROWTH_RATIO`), and 3D rigorous smoothing and size-field limiting using nearest-neighbor matching of adjacent mesh elements (`CFD_LIMIT_GROWTH_FLAG`). For the source definition, we add a trailing-edge source (`WLineSource`) with an influence radius of 0.1 MAC and a mesh-edge-length target of 0.003 MAC, in addition to the default sources at the leading edge and wing tips. The added trailing-edge refinement improved the volume-mesh quality near the trailing edge, and the refinement size is determined based on engineering judgment. The maximum edge-growth ratio is set to 1.2, and 3D rigorous smoothing is enabled. We control the surface-mesh density by adjusting the maximum mesh-edge length. Six levels of mesh refinement are defined, with maximum edge lengths of $0.1, 0.05, 0.0375, 0.025, 0.02$, and 0.01 MAC. We select a representative geometry from the dataset to determine a mesh configuration that is applied consistently across the full dataset. These surface meshes are shown in Figure 4.4. As expected, increasing the mesh-refinement level increases the mesh density. The corresponding surface cell element counts are 1.01×10^5, 1.05×10^5, 1.08×10^5, 1.18×10^5, 1.25×10^5, and 1.82×10^5 from Level 1 to Level 6 respectively.

The volume mesh is created from the exported surface mesh using a modified implementation of the `VSP2CFD` code [100]. The primary parameters that control the volume-mesh properties are the maximum number of anisotropic `T-Rex` boundary-layer layers (`TRexMaximumLayers`), the first layer thickness (`TRexCondition-setSpacing`), and the boundary decay rate of the unstructured mesh around wall surfaces (`BoundaryDecay`). For all mesh levels, the boundary decay rate is set to 0.85. The first layer thickness values for

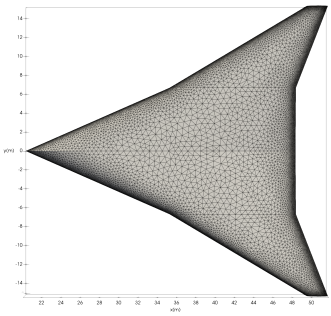
(a) Level 1



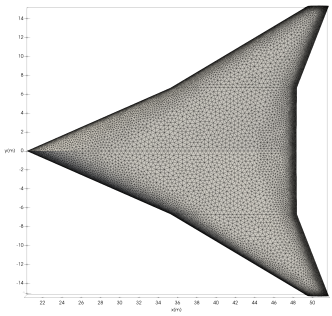
(b) Level 2



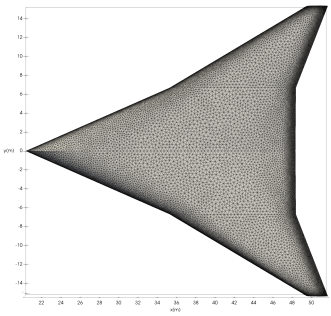
(c) Level 3



(d) Level 4



(e) Level 5



(f) Level 6

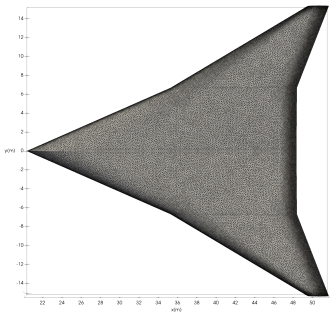


Figure 4.4: Surface meshes of a representative geometry used for the mesh sizing study.

the corresponding surface meshes from Level 1 to Level 6 are 6×10^{-5}, 2×10^{-5}, 9×10^{-6}, 7×10^{-6}, 6×10^{-6}, and 6×10^{-6} (m), respectively. The maximum number of anisotropic layers is set to 25, 35, 40, 50, 45, and 50 layers from Level 1 to Level 6. As a result, the total volume cell counts are 5.32, 6.13, 6.57, 6.96, 7.54, and 10.72 million, respectively. The volume-mesh sizing study focuses primarily on identifying a proper boundary-layer mesh setup that allows the SA-R turbulence model [25] to capture the boundary-layer flow correctly, thereby yielding accurate aerodynamic and flow predictions. Table 4.3 summarizes the mesh configurations used in this study.

Level	**Surface Elements**	**Volume Elements**	y^+
1	1.01×10^5	5.32×10^6	6.93
2	1.05×10^5	6.13×10^6	3.50
3	1.08×10^5	6.57×10^6	1.98
4	1.18×10^5	6.96×10^6	1.66
5	1.25×10^5	7.54×10^6	1.35
6	1.82×10^5	1.07×10^7	1.32

Table 4.3: Mesh statistics across refinement levels.

A mesh convergence study is conducted by running CFD simulations with identical freestream conditions and solver configurations across all mesh levels. We use the representative geometry at Mach 0.3 AOA 13° flow and monitor the convergence of C_D and C_M, as well as the resultant average $\bar{y^+}$. We define the area-averaged y^+ as:

$$\bar{y^+} = \frac{\int_S y^+ dA}{\int_S dA}, \tag{4.6}$$

and the resulting values for each mesh level are reported in Table 4.3. For the SA-R turbulence model without wall functions, we target $\bar{y^+} \approx 1$, which is achieved by selecting mesh level 4 and beyond. The convergence behavior for C_D and C_M is shown in Figure 4.5. It is observed that the Level 4 mesh with 6.96×10^6 volume cells achieves a difference within 1% of the finest grid used in this study while maintaining a manageable computation cost for large dataset generation. Therefore, this mesh configuration is selected and used for all subsequent configurations' meshing.

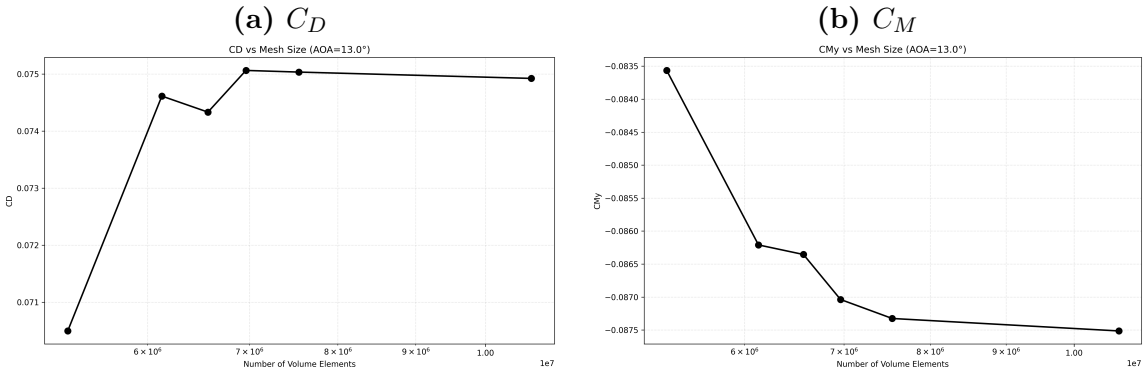


Figure 4.5: Mesh sizing study: convergence of C_D and C_M with respect to mesh size.

4.4.2 CFD Solver Settings

The fluid domain consists of a freestream far-field with a diameter of 40 MAC being 630 (m), centered at the wing apex, and the wing surface. The far-field boundary is assigned with a far-field boundary condition, while the wing surface is assigned with adiabatic wall (no-slip) condition. Freestream condition is specified as standard air, with gas constant of 287.06, at 285 (K), and static pressure of 7.18×10^4 (Pa). The air has a constant heat capacity ratio of 1.4. Sutherland viscosity model is used, with a reference viscosity of $1.715 \times 10^{-5}(\mathrm{Ns/m^2})$ [131], and a constant laminar Prandtl number of 0.72 and turbulent Prandtl number of 0.9. The reference Reynolds number is based on average the MAC of 16 (m), yielding $\mathrm{Re_{ref}} = 8.04 \times 10^7$. The aerodynamic coefficients are computed using a specific moment origin on the body and reference area defined according to the sizing rules in `SUAVE`. These configuration specific reference values are included in the dataset repository.

Spatial discretization employs the Jameson-Schmidt-Turkel (JST) scheme [47] for the flow equations with Green-Gauss gradient reconstruction, using a JST sensor tuned with coefficients (0.5, 0.02). The turbulence equation is discretized using a scalar upwind scheme without MUSCL reconstruction. Temporal advancement uses implicit Euler for both flow and turbulence, with an adaptive Courant-Friedrichs-Lewy (CFL) number. The linear systems are solved by FGMRES with ILU preconditioning and a residual tolerance of 10^{-7}. The convergence criterion is set to be 10^{-6} for the continuity equation, with an additional convergence criterion for for the drag coefficient C_D requiring changes below 10^{-5} over 100 Cauchy iterations. These convergence criteria are applied consistently throughout all runs in the dataset.

4.4.3 CFD Results and Discussion

CFD runs are conducted using the Stanford Research Computing Center (SRCC) facility with a 32-core 2.4 GHz partition. The total computational cost of the dataset is 310,835 CPU-hours for 2,448 snapshots. For illustration of the variety of flow conditions represented in the dataset, Figure 4.6 shows three representative configurations (call signs as subtitles) at AOA of 15°. These geometries are selected from the holdout test set, spanning the platform types ranging from trapezoid wings, cropped Delta wings, to double-delta wings. The wing surfaces are colored by C_P, and Q-criterion iso-surfaces of 100 are added to illustrate the vortex sheets. The variation of geometries leads to drastically different surface pressure distributions and primary vortex structures, enriching the geometric and flow features for surrogate modeling of flow prediction of the double-delta wing family.

(a) 050101-7LK **(b) 09190107DL** **(c) 171801-1NW**

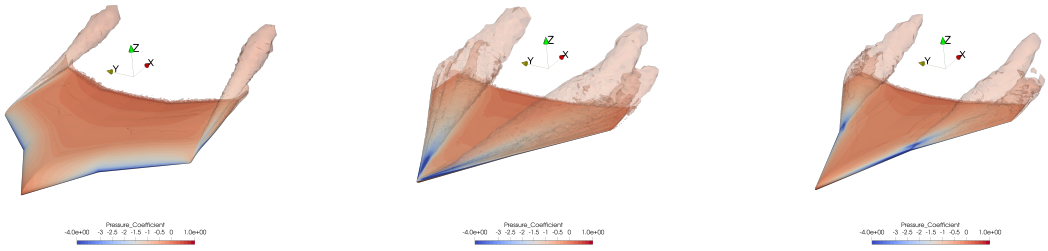


Figure 4.6: Pressure coefficient (C_P) visualizations and Q-criterion iso-surfaces at $Q = 100$ for different vehicle configurations, illustrating variations in geometry and flow states.

Among the 2,448 CFD simulations, 57 runs failed to converge to the specified residual target using the numerical scheme described above. These runs correspond primarily to highly cambered geometries at higher AOAs above 16°. The convergence failure arises from the unsteady nature of critically separated flow, for which the selected physics models and numerical scheme cannot recover a steady solution at the residual target. We believe it is more valuable to provide solutions with consistently converged results, even at a lower fidelity, rather than removing these snapshots. As a result, we lowered the order of the convective scheme and adopted AUSM [66] for these runs. The configuration files contained in the dataset annotate which convective scheme is used for each run. Future extensions of the dataset may consider using higher-fidelity methods, such as detached eddy simulation, for these runs.

4.4.4 VLM Settings

The VLM solver from `SUAVE`, as discussed in Section 2.2, is used to conduct LF simulations. A chordwise-by-spanwise lattice of 30×32 is employed to characterize the wing planform. Across all runs, the Polhamus suction model is activated, with a correction strength corresponding to a 90° rotation of the leading-edge suction vector to the leading-edge camber normal direction. With the current formulation, as shown by Shen and Alonso, VLM results for relevant configurations are relatively accurate up to an AOA of 10° [121].

The LF dataset includes the lattice geometric definition, the local loading coefficient ΔC_p, surface geometry features (including slopes, curvature, and panel thickness), local flow angles, as well as local velocities and circulations.

4.5 Data Availability

The dataset is hosted on our `Huggingface` repository at https://huggingface.co/datasets/yirens/double-delta-aero under Creative Commons Attribution 2.0 Generic (CC-BY-2.0). The repository includes all CFD results, including the `SU2` configuration files, surface and volume solutions in `vtk` format, convergence history files, and integrated aerodynamic loading summary files. The VLM solutions are provided in `json` format. All files are organized and indexed by vehicle call sign for ease of access.

4.6 Discussion

This chapter presents the construction of an open-source multi-fidelity aerodynamic dataset for double-delta wing configurations relevant to SST conceptual design. A six-dimensional parametric geometry definition is introduced, including span, break-chord location, inboard and outboard leading-edge sweep angles, trailing-edge sweep angle, and leading-edge droop. Together, these variables define a broad but realistic design space that spans trapezoidal, cropped-delta, and double-delta wing planforms.

The configurations are sampled from this design space using a purposefully selected DOE strategy. A nested Saltelli sampling scheme based on Sobol sequences is adopted to provide uniform, reproducible, and scalable design-space coverage. This structure allows the dataset to be expanded across multiple sampling levels without rerunning existing simulations, while also supporting future sensitivity analysis and uncertainty quantification. A separate

Latin Hypercube Sampling (LHS) holdout set is generated to evaluate model generalization without disrupting the sampling quality of the training set.

Finally, the chapter details the numerical simulation framework used to produce paired low- and high-fidelity aerodynamic data. HF RANS CFD simulations are generated using an automated `SUAVE`–`OpenVSP`–`Pointwise`–`SU2` workflow, with mesh refinement studies used to select a practical mesh generation protocol for large-scale dataset generation. LF VLM simulations are computed using `SUAVE`. The resulting dataset provides paired CFD and VLM field data across a wide range of geometries and angles of attack for wings at low speed (Ma = 0.3), forming a benchmark resource for multi-fidelity surrogate modeling, model scaling studies, and low-speed SST planform shape optimization.

Chapter 5

Scaling

The previous chapters introduced the `VortexNet` surrogate model (Chapter 3) and the `double-delta-aero` multi-fidelity aerodynamic dataset (Chapter 4). While the model demonstrates strong predictive capability, its practical deployment in conceptual aircraft design also depends on understanding how its performance changes with training scale. In particular, it is important to quantify how much paired low- and high-fidelity data is required, how model capacity affects prediction accuracy, and how computational budget influences the achievable test error.

This chapter investigates `VortexNet`'s empirical scaling behavior, a term defined later in Section 5.1. We examine how the prediction error varies with dataset size, model size, and training computation using subsets of the `double-delta-aero` dataset. The resulting scaling relationships provide guidance for understanding the training cost of `VortexNet`-like surrogate models, selecting efficient dataset sizes and model capacities for domain-specific applications, and providing empirical evidence for the model's training dynamics.

5.1 Scaling Behavior under Data and Model Variation

Neural scaling laws quantitatively describe how model performance improves with increasing training scale. The training scale comprises three major components: data scaling (how performance changes with increasing training dataset size), model scaling (how performance changes with increasing model size), and computation scaling (how model performance changes with increasing computational budget). Foundational work was carried out by Hestness et al. Rosenfeld et al., Kaplan et al., and Hoffmann et al., focusing on training

large language models [43, 44, 50, 107]. Together, the studies show that model performance under a given computational budget C, when measured by an appropriate loss metric, follows a power-law relationship with training dataset size and model size:

$$\epsilon = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{D_c}{D}\right)^{\alpha_D} + \epsilon_0, \tag{5.1}$$

where ϵ is the test loss, N is the model size, D is the dataset size, and ϵ_0 is the irreducible error. Coefficients N_c, D_c, α, α_D, and ϵ_0 are fitting parameters. Such formulation is applicable to GNNs, as indicated by previous study [67].

We define ϵ as the MSE evaluated on the holdout test set. Recall that `VortexNet` learns a mapping $\mathcal{F}$ that takes a low-fidelity pressure–stream-condition–geometry state, represented as a graph G, and produces a predicted pressure distribution over the lattice, C_p^{pred}, that approximates the ground-truth pressure distribution C_p^{gt} obtained from high-fidelity simulations. The test loss is then defined as

$$\epsilon = \text{MSE}(C_p^{\text{gt}}, C_p^{\text{pred}}). \tag{5.2}$$

The remainder of this section outlines the procedure used to identify the scaling-law coefficients based on the test error ϵ. Two experimental configurations are examined: the first quantifies data size scaling at a fixed number of maximum weight update steps, while the second quantifies both data and model size scaling with unlimited updates. While these results presents `VortexNet`'s data requirements, they should not be extrapolated to all GNN-based field surrogate models without discretion.

5.2 Methods

The empirical scaling experiment aims to estimate the exponent coefficients in Equation 5.1, enabling us to quantify how prediction accuracy scales with training data size and to determine the data requirements necessary to achieve a target accuracy. The experiment comprises three components: the training data, the model, and the training configurations.

5.2.1 Data

We define the training data scaling under investigation as the increase in the number of snapshots resulting from higher sampling density over the a defined geometric design space.

As summarized in Table 4.2, using Saltelli sampling, the geometric configurations in the training set (described in Chapter 4) are subdivided into six levels. From Level 1 to Level 6, the training set includes a total of 8, 16, 32, 64, 128, and 256 geometric configurations, respectively. For each level, flow snapshots at five AOAs are selected at 11°, 13°, 15°, 17°, and 19°. Together, our training data scaling involves D values of $40, 80, 160, 320, 640$, and 1280 graphs, respectively. Each graph provides a multi-fidelity representation of a geometric configuration and its associated flow state.

The decision to include only five AOAs per geometry, rather than all available AOAs in the dataset, is motivated by the need to balance dataset size and model capacity under tight computational budget. Because the primary objective of the data scaling study is to determine the geometric sampling density required to train `VortexNet`, we prioritize scaling the number of geometries rather than the number of flow snapshots per geometry. As discussed by Hoffmann et al., dataset size, model size, and computational budget should be scaled proportionally [44]. Subsampling the available AOAs therefore enables training with smaller models at lower computational cost, while preserving the qualitative empirical scaling trends.

An additional consideration concerns how to define the dataset size D for a collection of graphs. Prior studies across different domains have adopted different definitions, including the volume of digital information measured in terabytes [61], the number of individual graphs [133], and the total number of edges [67]. Consequently, there is currently no standardized definition of D in graph-based setting, and its interpretation remains domain-specific. In this work, we define D as the total number of graphs in the dataset. This definition is aligned with the downstream design application, where designers seek to understand how densely the design space needs to be sampled and how many high-fidelity simulations are required to train a surrogate model with sufficient accuracy. Such a definition directly links to the number of simulation runs required and is thus more convenient for budget planning.

Besides the training set, we constructed the test set using the sixteen holdout test geometries, as described in Chapter 4.3. To maintain the consistency with the training set in flow snapshot selection, we include include snapshots at AOAs of 11°, 13°, 15°, 17°, and 19°. In total, the test set comprises 80 graphs. The test set is relatively small. However, considering the even geometric distribution shown in Figure 4.3, we believe the test set contains adequate variability representative of the training set and is suitable for use in test

loss evaluation.

5.2.2 Model

Prior work has shown that performance scaling depends critically on the relationship between dataset size and model capacity. In the under-parameterized or over-parameterized regimes, corresponding to a disproportionately small or large ratios of N/D, the test error no longer exhibits a power-law decay with respect to D, a phenomenon referred to as model scaling collapse [44, 67]. To avoid inconclusive results derived from a single fixed-capacity model, we repeat the scaling experiment using four models with model sizes ranging from 0.10 million to 2.41 million trainable parameters.

The `VortexNet` architecture has three hyper-parameters that are most relevant to model size [124], including the width of the nodal encoding for hidden layers (`hidden_feature`), the number of heads for multi-head attention (`heads`), and the total depth of the message passing blocks (`hops`). The `hops` parameter is related to the extent of nodal information propagation, and for the given 30×32 lattice, we determined that a `hops` of 10 is adequate for the information to propagate across all nodes in a graph. This hyper-parameter is therefore held constant across models. To change the model size, we alter `heads` and `hidden_feature`. Specifically, increasing `heads` and `hidden_feature` increases the model size while widening the model. The resulting models created, along with their numbers of trainable parameter size (N), are presented in Table 5.1.

Level	`hidden_feature`	`heads`	**Model Size (N)**
Mini	13	5	0.103×10^6
Small	27	5	0.538×10^6
Medium	37	5	1.144×10^6
Large	45	6	2.411×10^6

Table 5.1: Model sizes at four levels.

Once the model size is defined, additional hyper-parameters related to regularization and training must be specified [124]. Two strategies are used for the constrained compute and the unconstrained compute test cases.

Constrained Computation

For the constrained computation test case, the maximum number of weight update iterates is fixed. The total computational cost for train a model, regardless of model sizes, is manageable. This creates an opportunity to optimize the training configurations (e.g. learning rate and decay rate) for each model and promote cross-model comparability by adopting a consistent hyper-parameter selection procedure that yields "equally optimal" configurations for each model.

For each model size, we employ `Optuna` [2] to optimize key training and regularization hyper-parameters, including the learning rate and weight decay of the ADAM optimizer, as well as the dropout ratio and skip-connection bandwidth (`Alpha`). Hyper-parameter optimization is performed using a subset of the Level 6 training set, constructed by randomly sampling 2048 snapshots, thereby ensuring that no information from the holdout test set is leaked during model selection. This dataset is further randomly split into an 80%-20% training-validation partition. For each model, the training configuration that achieves the lowest validation MSE is adopted as the training hyper-parameters.

To ensure fair comparison across model capacities, an identical computational budget, as discussed in the following section, is allocated to all models. We conduct a maximum of 35 training runs during hyper-parameter optimization for each model, with the first ten trials used as "warm start" evaluations for `Optuna`. All other `Optuna` optimization settings are left at their default values.

The MSE values of the optimized models are reported in Table 5.2. Despite differences in model sizes and training configurations, comparable test MSE values are achieved. The remaining hyper-parameters are reported to ensure reproducibility. These hyper-parameters are held fixed for each model across all data scaling runs.

Level	**Test MSE**	**Learning Rate**	**Decay**	**Dropout**	`Alpha`
Mini	5.97×10^{-2}	1.14×10^{-3}	2.81×10^{-6}	5.57×10^{-2}	4.50×10^{-1}
Small	5.66×10^{-2}	6.59×10^{-3}	1.54×10^{-6}	1.22×10^{-1}	2.38×10^{-1}
Medium	5.21×10^{-2}	1.60×10^{-3}	1.98×10^{-6}	5.19×10^{-2}	4.10×10^{-1}
Large	5.88×10^{-2}	6.02×10^{-4}	3.28×10^{-6}	1.82×10^{-1}	2.77×10^{-1}

Table 5.2: Additional training hyper-parameters after optimization for all models.

Unconstrained Computation

In the unconstrained computation test case, the maximum number of weight updates is unknown. Furthermore, larger datasets require more mini-batches per epoch. Employing the hyper-parameter optimization approach described in the previous section would result in a computational cost exceeding the author's available resources. It is determined that, based on the hyperparameters presented in Table 5.2, a representative training configuration can be selected and used across all models. As a result, a learning rate of 2.00×10^{-4}, a decay rate of 2.40×10^{-6}, a dropout rate of 1.03×10^{-1}, and an alpha of 3.46×10^{-1} are used, representing an average training configuration that ensures model convergence at the largest dataset size.

5.2.3 Training

The training setup follows that presented in Chapter 3. For each model training at a particular model size N and dataset size D, we repeat the training six times under randomized initial seeds for initial model weights and dataset partitioning. Trial-level MSEs are aggregated to report per-level mean, standard deviation, minimum, and maximum values to improve the robustness of the scaling exponent estimation. Least-squares fitting using `SciPy`'s `curve_fit` function is used to fit the aggregated mean MSEs to the scaling law coefficients in Equation 5.1. All model trainings are executed on a 40GB NVIDIA A100 GPU. The following two training protocols are used for the constrained computation case and the unconstrained computation case respectively.

Constrained Computation

To fit the neural scaling law coefficients in Equation 5.1, we conduct a controlled data-scaling experiment for each model listed in Table 5.1. The training-dataset size D is systematically varied, as discussed in Chapter 5.2.1, while keeping the training procedure fixed. This procedure is designed to isolate the effect of data scaling from variations in model capacity or computational cost.

When performing the D scaling using datasets at six levels, as discussed in Chapter 5.2.1, and across four model sizes N, careful design of the training scheme is needed such that, as a result of the experiment, the fitted α_D represents the training efficiency gain resulting from increased geometric sampling density, rather than being a consequence of increased

computation. As an experimental design choice, we fix the maximum computational budget C^* available for all training runs across all D and N. C^* is defined as the maximum number of weights update steps available for each training fold, and is set to 2000. Four folds are used to train the model; in each fold, a 75%–25% training-to-validation data partitioning is used. In addition to fixing C^*, an early stopping patience of 50 epochs is used across training runs to prevent model overfitting. The early stopping mechanism terminates the training of the current fold when no improvement in validation loss is observed over the preceding 50 epochs.

Setting a fixed maximum number of weight-update iterations introduces an additional challenge for dataset-size scaling. In standard mini-batch gradient descent, each batch corresponds to a single weight-update step; for a training dataset of size D_t and batch size B, a total of D_t/B weight updates are performed per epoch. Since B is constrained by hardware memory and is fixed at 128, the total number of weight updates scales with D_t, introducing an inconsistency when isolating the effect of dataset size on model performance. Moreover, prior work suggests that the optimal computational budget C does not necessarily scale linearly with D [44]. Consequently, during the experiment we found the model hyper-parameters, as discussed in Chapter 5.2.2, are sensitive to C, such that designing a C allocation scheme across model sizes and data sizes for calibrating the performance becomes difficult. An imprudent C scheduling may result in biased power-law observations.

To address this issue and isolate error scaling with respect to D alone, we fix the number of weight-update steps across dataset sizes by adopting a stochastic subsampling scheme. Specifically, at each epoch, we randomly draw B samples from the training set and perform a full-batch gradient update. Because of the stochastic nature of this procedure, and given a sufficiently large number of epochs, the model is exposed to the entire training dataset in expectation, while the total computational budget C remains constant across different dataset sizes. As a result, for each fixed model size, the observed scaling behavior can be attributed to changes in sample distribution introduced by D scaling. Since the same number of flow conditions are selected at each level of the geometry set, this scaling behavior directly reflects gains in prediction accuracy arising from increased sampling density in the design space. The resultant ϵ versus D scaling results are conceptually similar to the iso-FLOP experiments reported by Hoffmann et al. [44].

Unconstrained Computation

In the unconstrained-computation setting, the standard mini-batch gradient descent optimizer is used such that random samples are not drawn from the dataset, and each epoch consists of multiple mini-batches when the dataset size exceeds the batch size. To investigate the relationship between model size, dataset size, and computation, a logger is deployed to test the intermediate models' performance on the hold-out test set during the training. The logger is activated as a frequency for every 10 training epoches. The additional granularity from the unconstrained computation test setup also allows the identification of model overfitting. The remaining training configurations are consistent with those in the constrained-computation case.

5.3 Results

5.3.1 Scaling Behavior under Constrained Computation

The mean test MSE, evaluated on the holdout test set and averaged across all trails at each D, is plotted in Figure 5.1. Different data series represent scaling results from different models, from large to small. Overall, the test MSE decreases as D increases, although the rate of convergence varies across models. From the learning curves, three distinct learning regions can be identified. We highlight these regions by light red, white, and light green bands along the D axis. Such zoning is consistent with the learning curve behavior described by Hestness et al. and Rosenfeld et al. [43, 107].

When the dataset is small (light red region), the learning curves exhibit behavior characteristic of the "small data region"[43], in which model performance is limited by insufficient training data provided. Consequently, the model performs a best-guess prediction. For models investigated in this study, this region corresponds to $D \leq 80$. As D increases beyond the "small data region", a "power-law region" [43] emerges, such that increasing D in this region helps the model to improve its prediction quality for generalization tasks, such as predictions in the holdout test set. Under the current model and training configuration, this region spans $80 \leq D \leq 320$, corresponding to geometry sets of 16 to 64 configurations.

For larger data sizes, at $D \geq 320$, an "asymptotic error region" is observed. We intentionally distinguish the terminology of this region from the "irreducible error region" described by Hestness et al. [43], as the error floor observed in Figure 5.1 should not be

interpreted as an inherent limitation of the dataset or as insufficient model expressivity. Instead, it may arises from training under a fixed computational budget C^*, as discussed in Chapter 5.2.3. To achieve fixed C^* across different values of Ds, subsampling is used and thus large dataset are underutilized in each weight update step. Later section (Chapter 5.3.2) examines the data scaling trend in this region in further detail.

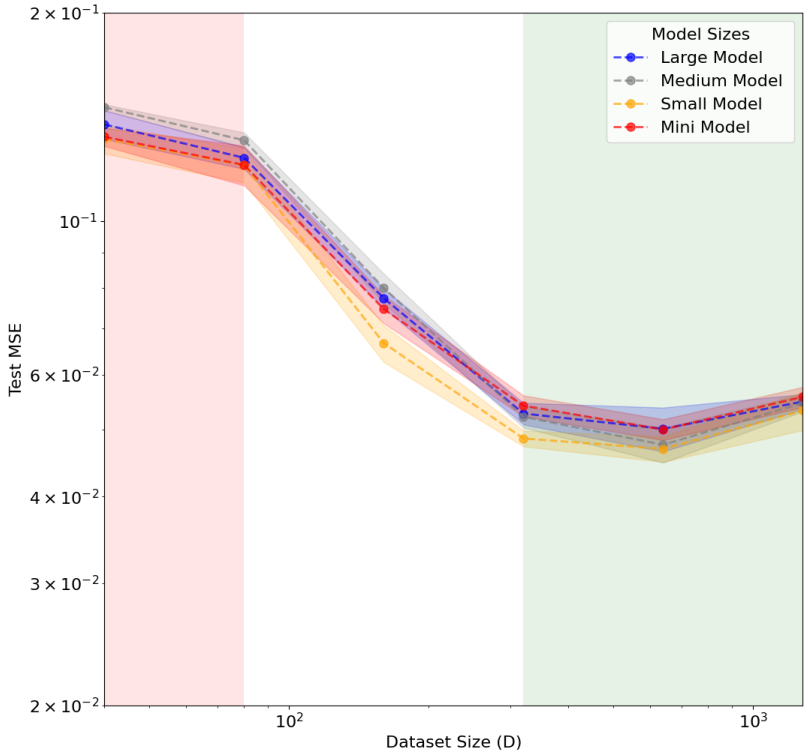


Figure 5.1: Scaling behavior of test MSE as a function of the training set size D for models of different sizes, where D denotes the number of distinct geometry-flow snapshots represented as graphs in the dataset.

Among these regions, the power-law region provides information on data-size scaling. Figure 5.2 presents the test MSE as a function of D, together with the mean value, variability indicated by boxes spanning plus and minus one standard deviation, and the minimum and maximum values shown by whiskers in a box-plot style. The mean values are identical to those presented in Figure 5.1. It is observed that the larger models tends to have smaller coefficient of variation in the power-law region, indicating a more consistent performance across repeated trials.

Within the power-law region, Equation 5.1 can be further reduced to,

$$\epsilon' = \left(\frac{D_c}{D}\right)^{\alpha_D}, \tag{5.3}$$

by absorbing the constant contributions from model size and irreducible error into the test loss, we focus on quantifying the scaling exponent α_D. To do so, we use `SciPy`'s `curve_fit` function to perform a non-linear least squares fit of the data. The mean MSE is treated as the dependent variable, and D is treated as the independent variable, with the corresponding standard deviations used to represent the relative uncertainty of the MSE measurements. The resulting power-law fits are shown as dashed lines in Figure 5.2(a)–(d).

From the power-law fitting, we estimate the scaling exponent α_D to be 0.5484, 0.6220, 0.6654, and 0.6128 for the Mini, Small, Medium, and Large models, respectively. The corresponding coefficient D_c are 1.1360, 1.4090, 1.7928, and 1.4227, respectively. The corresponding R-squared values are 0.9772, 0.9652, 0.9985, and 0.9965 respectively, indicating a high-quality fits for α_D estimation.

These fitted power laws can then be used to answer the question of how much data is required to achieve a target prediction accuracy under the proposed training procedure and computational budget. Assuming that the minimum MSEs obtained from hyper-parameter optimization (Table 5.2) are reasonable estimates of models' best performance under substantially larger dataset and are served as the loss target, we compute the optimal dataset size D^* at the computational budget C^* by inverting Equation 5.3. The resulting values of D^* are 261, 245, 317, and 261 for the Mini, Small, Medium, and Large models, respectively. These values correspond to geometry sets of 52, 49, 63, and 52 geometries, all of which are slightly smaller than the Level 4 dataset listed in Table 4.2.

The size of the geometry set can be further mapped to the corresponding sampling density in the unit hypercube. To this end, we first fit a power law relating the average nearest-neighbor distance h, defined as the average Euclidean distance from each design sample to its nearest neighbor, to the number of geometry samples G contained in a dataset, using the data in Table 4.2. This fit yields $h = 0.62G^{-0.15}$, from which the estimated D^* values correspond to h in the range of 0.333 to 0.346. The corresponding one-dimensional optimal sampling density can then be estimated for space-filling sampling algorithms by accounting for dimensional scaling. Based on this analysis, selecting approximately eight sampling points per dimension is sufficient to achieve optimal surrogate model performance

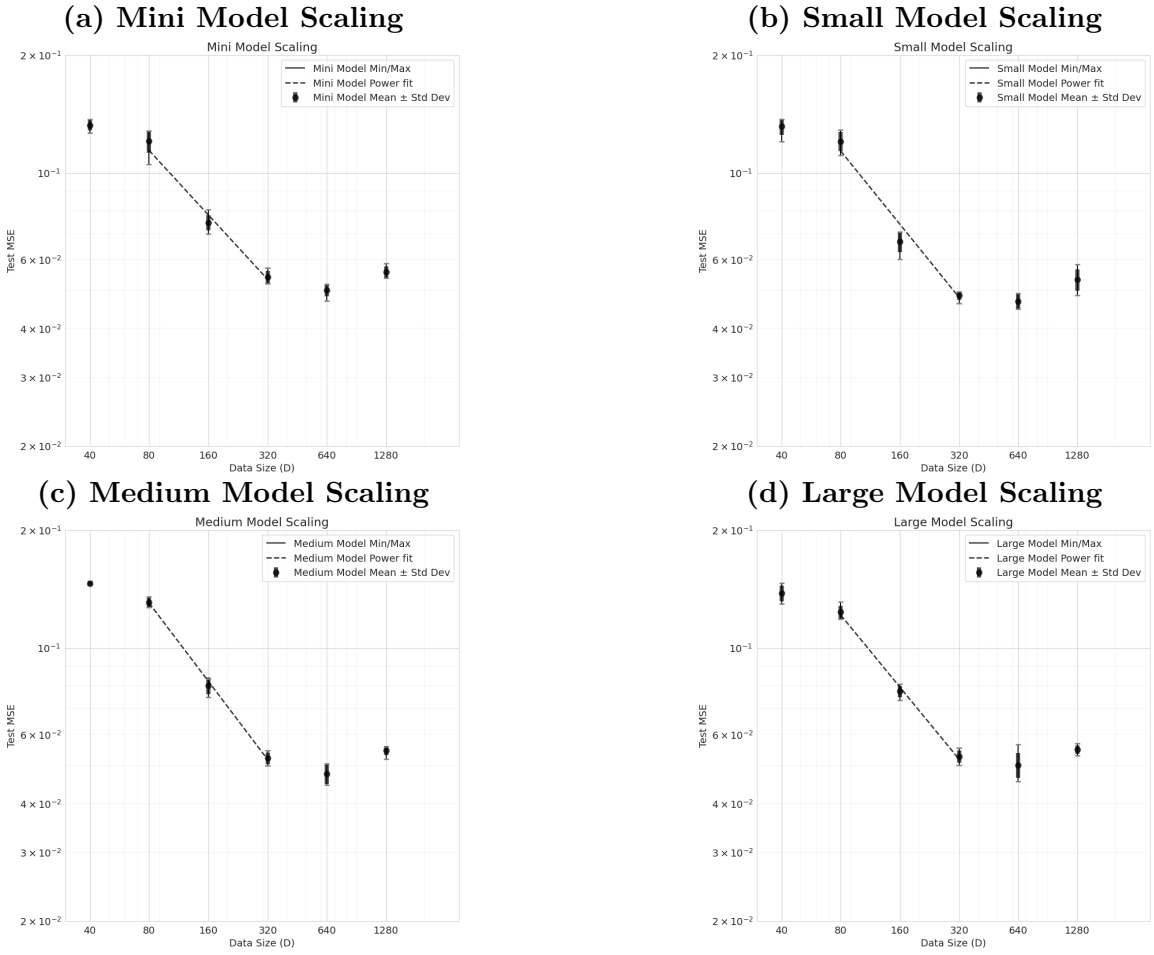


Figure 5.2: Test MSE scaling with D for four models arranged from (a) to (d): Mini, Small, Medium, and Large, along with their log-log linear fit for the power-law region.

under the allocated computational budget C^*.

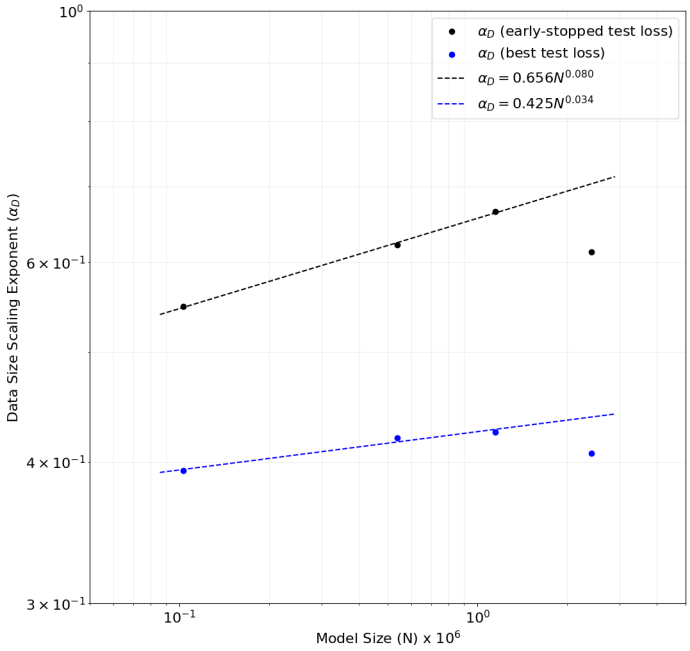


Figure 5.3: Data scaling exponents (α_D) versus model sizes (N).

Finally, we assess how the fitted scaling exponent varies with model size. Figure 5.3's black data series shows the scaling relationship. We observe an approximately linear correlation between α_D and N on a log-log scale when the model size is not excessively large. The Large model, however, deviates from this trend. The exact cause of this deviation requires further investigation, but one plausible explanation is that the model capacity exceeds what can be effectively optimized under the available computational budget. As shown in Table 5.2, the Large model employs a learning rate that is an order of magnitude smaller than those of the smaller models and achieves a higher test MSE than the Medium model, which contradicts typical expectations from model-size scaling [50, 44]. Another plausible explanation is that the large model is overly expressive, such that it learns noise from the training data when inadequate regularization is applied. Finally, the early-stopped test error compared in this section, which assumes that the early-stopping criterion is sufficiently sensitive and robust to prevent model overfitting to the training set, may not be a perfect indicator of the true learning fit. If there is a lag between the onset of model overfitting and the point at which the early-stopping monitor terminates training, the saved model weights

may not be optimal. Such lag exists, as discussed later in Section 5.3.2, whose results are plotted in blue in Figure 5.3 and will be elaborated later.

5.3.2 Scaling Behavior under Unconstrained Computation

Although the results presented in the previous section establish a data-size scaling trend for the early-stopped test error, they also raise several unresolved questions regarding the validity of the fitted scaling law and the observed deviation of the largest model from the model-scaling trend. Specifically, this section seeks to address three questions: (a) what mechanism causes the Large model to exhibit worse data-scaling efficiency than the smaller models; (b) whether the imposed compute limit biases the power-law fit, and whether, in the absence of the subsampling required under the stochastic batch gradient descent used in the previous study, the models continue to improve at large dataset sizes ($D \geq 640$) when trained using standard mini-batch gradient descent; and (c) how the required computation scales with both model size and dataset size. The results presented in this section are intended to answer these remaining questions.

To answer these questions, an additional scaling experiment is conducted using the procedure discussed in Chapter 5.2.3. Specifically, four models are trained across six dataset sizes, and each configuration is repeated six times with randomized seeds. A more granular scaling study is performed by logging model performance on the holdout test set throughout training. Specifically, the test error is evaluated every 10 training epochs.

This test-loss history is presented in Figure 5.4. In this figure, each model's test-loss history is distinguished by color, and individual runs are plotted as semi-transparent solid lines. In addition to the test-loss history of each run, we extract the efficiency frontier of the test loss, defined as the minimum test loss achieved at a given compute cost by retaining only the best test loss at each compute level for each model. These efficiency frontiers are plotted as thick solid lines.

Several observations can be made. First, in general, larger models shift the efficiency frontier to the right, indicating that they require more compute overall. Second, larger dataset sizes require higher compute, which is apparent from the progression of test-loss clusters for each model. For example, the correspondence between test loss and dataset size for the Large model is annotated in the figure, and the corresponding compute clusters range from around 2×10^{14} FLOPs for $D = 40$ to 10^{16} FLOPs for $D = 1280$. Increasing dataset size is generally beneficial for test loss, and this trend is largely maintained up to

$D = 640$, especially for the larger models, where the minimum achieved test loss decreases as model size increases. Third, when examining the efficiency frontier across dataset sizes from 80 to 640, the marginal benefit of additional data appears to diminish as the slope of the efficiency frontier starts to flatten.

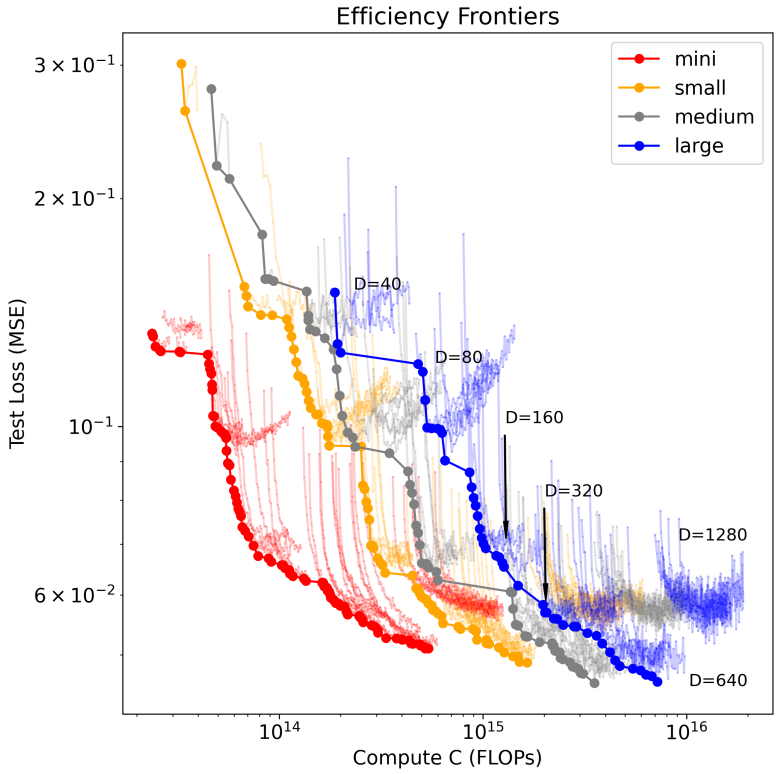


Figure 5.4: Test loss history versus accumulated compute for all models and dataset sizes.

In addition to the three observations above, the test-loss curves exhibit several interesting trends in both the small-data regime ($D < 80$) and the largest-data regime ($D = 1280$). Specifically, in the small-data regime, many runs show increasing test loss with increasing compute, indicating that the model may be overfitting. In the largest-data regime, similar test-loss plateaus appear across the Small, Medium, and Large models, accompanied by relatively high variation. To determine whether the models are underfitting or overfitting, Figure 5.5 is presented to show the corresponding training-loss histories for the runs shown in Figure 5.4. One immediate observation is that the model is indeed in an overfitting regime for dataset sizes smaller than 80, as the training loss continues to decrease across

compute levels while the test loss in Figure 5.4 increases. For dataset sizes between 160 and 640, a more or less consistent scaling of minimum training loss with compute is observed. At the largest dataset size ($D = 1280$), although the training loss still decreases monotonically, a slight increase in the minimum training loss is observed compared with that obtained at $D = 640$.

The slightly worse performance at the largest dataset size, as shown in Figure 5.1, however, does not appear to be caused by overfitting, nor by the compute limits imposed by stochastic mini-batch gradient descent. In this experiment, random subsampling was not used to satisfy batch-size constraints during training. Instead, the observed behavior is more consistent with either an optimization bottleneck or limitations in the dataset itself than with overfitting. This conclusion is supported by the fact that the training loss at $D = 1280$ is also slightly higher than that at $D = 640$ for the Medium and Large models. The training losses of the Mini and Small models, however, still show slight improvement over those at $D = 640$. If the larger dataset were simply causing overfitting, one would expect lower training loss but worse holdout performance.

The observed relationship between test loss and training loss, together with the observation that the smaller models, which have more limited expressiveness, continue to scale to larger dataset sizes when evaluated by training loss, suggests that the failure mode at the largest dataset size is most likely due to diminishing information gain. In the present study, larger datasets are generated by increasing the sampling density within the same bounded design space, rather than by expanding the design space itself. In addition, the freestream condition is limited to a single Mach number, and the same AOA sampling is applied to all geometries. As a result, additional samples at large D do not necessarily provide proportionally more new information. Once the dominant variations of the aerodynamic response surface are sufficiently resolved, the marginal benefit of further densification diminishes. At very high sampling densities, the signal-to-noise ratio of newly added samples may also decrease. Although the additional data is not out-of-distribution, they may contribute limited new information while still carrying numerical uncertainty associated with the data pipeline, such as uncertainty in the simulations and in the ΔC_p projection. Increasingly dense sampling may therefore cause the incremental samples to reflect such noise rather than meaningful new reference graph-to-surface-loading correlations. Under such conditions, further increases in dataset size may yield only marginal improvement in generalization and may also make optimization more difficult. The observed degradation in

training loss for the larger models, which are more expressive and potentially more sensitive to such noise, is consistent with this hypothesis.

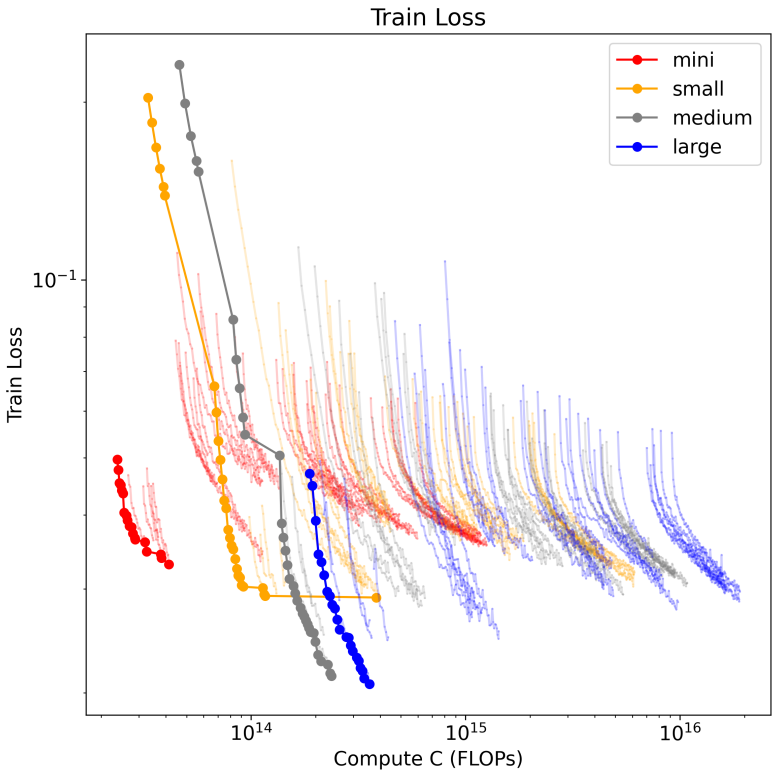


Figure 5.5: Train loss history versus accumulated compute for all models and dataset sizes.

Given these observations, it is believed that the data-scaling exponent reported in Figure 5.3 may be overestimated, as the test loss at $D = 80$ is likely inflated due to overfitting. Therefore, an additional scaling exponent is fitted using the best test loss observed throughout training. The resulting data-scaling and model-scaling relationships are presented in Figure 5.6, where subfigures (a) and (b) show the scaling of the best test loss with dataset size and model size, respectively.

From Figure 5.6, the first observation is that the best test loss does not scale significantly with model size, as indicated by the diminishing coefficients N_c and α_N. In contrast, clear scaling is observed with dataset size, for which we estimate

$$L(D) \sim \left(\frac{0.274}{D}\right)^{0.418}, \tag{5.4}$$

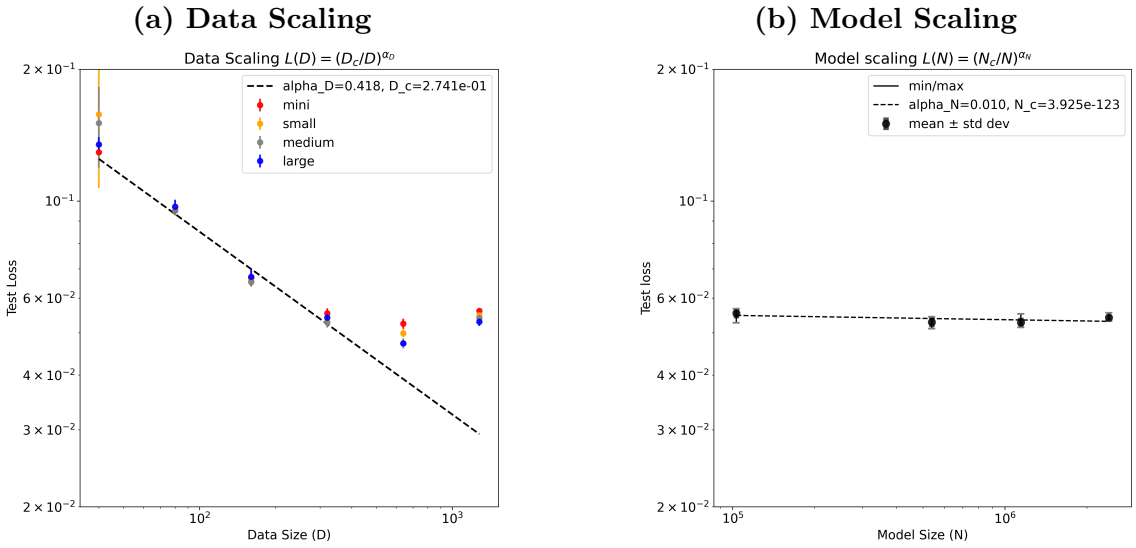


Figure 5.6: Best test MSE as a function of dataset size and model size: (a) data scaling and (b) model scaling.

when fitted to observations from all models. Jointly, when considering Equation 5.1, the scaling results support to drop the (N_c/N) for `VortexNet`'s scaling. Individually, for the Mini, Small, Medium, and Large models, we estimate α_D to be 0.393, 0.420, 0.425, and 0.407, respectively. These exponents are plotted in Figure 5.3 as the blue data series. Similar to the early-stopped test loss reported in the previous section, the data-scaling exponent increases for moderately sized models, while the Large model does not follow the trend line.

Finally, we aims to quantify the optimal computation for achieving the best test loss. To do so, we took average of the best test loss and the corresponding computation FLOPs at the best test loss for each model and data size. The result is then normalized by the average FLOPs-per-weight-update step C^*, which is estimated to be $C^* = 1.78BEN$, where B is the batch size, $E = 3840$ is the edges in the graph used in the dataset, and N is the corresponding model size. This FLOPs-per-step estimate is obtained by conducting a linear fit for all trial runs, and the fit quality is validated by an R2 score of 1.0. Figure 5.7 shows the scaling of the best test loss with normalized compute. Each data point therefore represents the best test loss achieved by a given model–dataset configuration, plotted against the corresponding number of optimization steps needed to attain that minimum.

Overall, the figure shows that the scaling relationship between compute and best test

loss is similar across all models, although the larger models are slightly more efficient than the smaller ones, potentially explaining the increase in α_D shown in Figure 5.3. We fit a power law of the form

$$L(\tilde{C}) = \left(\frac{C_c}{\tilde{C}}\right)^{\alpha_C},$$

which yields $\alpha_C = 0.314$ and $C_c = 0.2567$. Together with a desired test-loss target above the error floor, this compute scaling law can be used to estimate the total computational cost required to train the `VortexNet` surrogate model. By substituting the normalization factor, the computational cost required to achieve a target loss can be estimated as

$$C \sim 0.4569\, B\, E\, N\, L^{-1/0.314}. \tag{5.5}$$

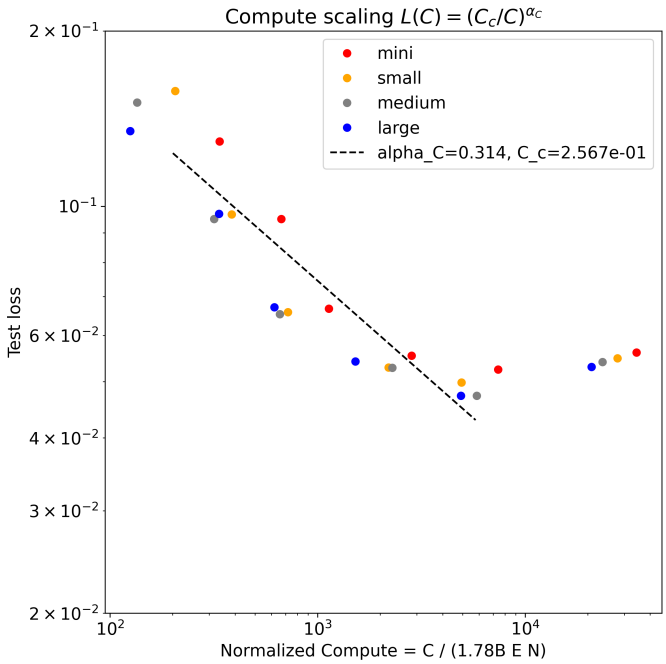


Figure 5.7: Best test loss versus normalized compute, where normalized compute is defined as the total training FLOPs divided by the FLOPs per optimization step, equivalent to the number of weight-update steps required to reach the minimum test loss.

5.4 Discussion

The scaling results in this chapter show that, for `VortexNet`, generalization improves more consistently with dataset size than with model size. Across both the fixed-budget and unconstrained-compute studies, test loss decreases as D increases up to approximately $D = 640$, with the clearest power-law-like behavior occurring in the intermediate range of $80 \leq D \leq 320$. The minor test loss degradation at $D = 1280$ suggests that scaling by denser sampling alone yields diminishing returns. The discrepancy in best test loss among models indicates the degradation is most likely due to information gain, such that the added samples, due to no new flow conditions are added, likely contribute less new information for the model to learn. On the contrary, the additional noise introduced by modeling and data pipeline uncertainties may pose a learning problem that challenging for larger model to distinguish variance from mean, when regularization parameters are not adjusted accordingly. This trend suggests that further study shall be conducted on the value of information included in the dataset.

Model-size scaling shows a different pattern. The best test loss does not exhibit a strong monotonic dependence on model size, indicating that increasing parameter count alone does not substantially reduce the achievable error. Across all models, the best test loss is achieved at a data size of 640 snapshots, and two of the largest models achieve comparable best errors. This suggests that model expressiveness is not the primary limiting factor for the best test performance. Instead, because the dataset was constructed by densifying the sampling across the design space while keeping the free-stream condition fixed, it may not contain sufficient feature variation for the larger models to exploit. More generally, achieving improved test performance requires balancing model variance and bias against the information content of the dataset. It is therefore unlikely to be achieved simply by increasing model size without also increasing the feature richness embedded in the training data.

However, the fitted data-scaling exponents suggest that moderately larger models are somewhat better at utilizing additional data, since both the best-test-loss and early-stopped analyses show increasing α_D before the Large model deviates from the trend. This implies that larger models may improve data efficiency without necessarily improving the final error floor by much. Such phenomena maybe induced by flooring of the dataset, or due to the flooring introduced by regularization and physical loss term that aims to improve model's

performance and reduce high-variance error in small-data regime. The Large model's deviation further suggests that beyond a certain size, additional capacity is no longer effectively exploited under the present optimization, data pipeline and model setting.

This weak sensitivity of test loss to model size is significant because it suggests that directly increasing latent feature dimension has diminishing returns for this class of GNNs. Unlike transformers, GNNs do not appear to scale smoothly through width alone. Depth is also not freely scalable, since it corresponds to additional message-passing rounds and may introduce oversmoothing or other optimization difficulties. As a result, improving GNN scaling for aerodynamic surrogate modeling will likely require more carefully designed feature engineering, such as skip connections, hierarchical pooling and unpooling, or multiscale message-passing, rather than simply enlarging the latent space. Overall, these results suggest that future gains will depend more on improving data diversity and GNN design than on brute-force parameter scaling.

Finally, if we combine the power-law scaling fit in Equation 5.4 and Equation 5.5, a joint requirement for computation scale that leads to optimal model performance can be obtained

$$C(D, N) \sim 2.5632 B\, E\, N\, D^{1.331}. \tag{5.6}$$

This scaling suggests, within the power law regime, the total computation scale is more dependent to the dataset size rather than the model size. The further implication is that for `VortexNet` type of architecture, the pipeline bottleneck is still being able to generate physics prior through CFD simulations. This observation is consistent with Ashton's analysis on data generation and training cost for AI foundation models [7].

Taken together, these results suggest that a training set consisting of approximately 36 to 64 geometries, corresponding to about 180 to 360 total snapshots, is sufficient to achieve near-optimal performance for `VortexNet` under the present workflow, and that a model size on the order of one million parameters is adequate for this task. This data requirement is lower than that reported for other machine-learning surrogate models for field prediction [105, 3, 20, 153], which often require hundreds of geometries and thousands of flow snapshots to achieve comparable geometric generalizability and field accuracy. We attribute this relative data efficiency to `VortexNet`'s small-data-oriented design, including its compact architecture, physics-informed training objectives, and scheduled training strategy.

Overall, the scaling study shows that `VortexNet` is limited less by parameter count

than by the information content of the training data. For the present aerodynamic field-prediction task, about 30 geometries and 300 flow snapshots are sufficient to approach near-optimal performance, while further densifying the same design space under fixed flow conditions yields diminishing returns. This indicates that future improvements are more likely to come from richer flow-feature diversity, and improved graph architectures rather than simply widening the GNN. Since the practical cost of training `VortexNet` is ultimately tied to generating useful HF data, efficient and information-aware dataset design remains central to making MF domain-specific surrogate modeling practical for conceptual design.

Chapter 6

Application to Conceptual Design Optimization

With confidence in `VortexNet`'s generalizability across the design space, as discussed in Chapter 3, and with an understanding of its training-data requirements, as discussed in Chapter 5, it is now possible to integrate these components into a conceptual design optimization workflow. Leveraging the SST planform dataset generated in Section 4.4, this chapter uses a simple conceptual design test problem, namely planform shape optimization, to demonstrate the proposed workflow.

Specifically, a two-point, stability-constrained optimization case is used as a preliminary demonstration of how surrogate-augmented fidelity affects the conceptual design outcome. The purpose is not to provide a fully matured aircraft optimization study, but rather to show that improved aerodynamic fidelity can change what is achievable in conceptual design. By providing more accurate low-speed lift and moment predictions at conceptual-design cost, the surrogate-augmented workflow demonstrates the potential to enable stability-constrained SST planform design earlier in the design process, rather than relying on historical workflows in which such constraints are often addressed through later-stage HF analysis and design correction.

6.1 Problem Definition

We select the vehicle planform defined in Section 4.1 as the study case. The double-delta planform exhibits rich nonlinear aerodynamic behavior and poses significant challenges in

Component Name	Weights [lbs]	Centroid Location-x from Datum [m]
Wing	10 lbs/ft^2	@0.4MAC
Horizontal Tail	5 lbs/ft^2	@0.3MAC
Vertical Tail	5 lbs/ft^2	@0.3MAC
Propulsion System	44540	43.00
Fuselage	29120	32.65
Systems and Equipments	43152	34.36
Fuel and Payload	65099	34.07

Table 6.1: Mass properties of system components used for center of gravity computation.

formulating a stability-constrained planform optimization problem within a conceptual design setting, primarily due to the limited fidelity of LF tools.

Throughout aerodynamic shape optimization, design variables $\mathbf{x} = [\delta, \Lambda_{in}, \Lambda_{out}, \Lambda_{te}, BW2, B]$, as explained in Section 4.1, are allowed to change and thus alter the planform shape. As a result, the center of gravity (CG) of a vehicle needs to be recalculated. To compute CGs, we hold the fuselage, tail, engine, systems and vertical stabilizer locations and weights fixed, so any CG shift arises solely from changes to the wing geometry. To model the effects of the change of wing planform to overall CG, we use the RSAC weight breakdown and compute moment balances about the nose datum, as shown in Table 6.1.

In Table 6.1, the specific weights for aerodynamic surfaces are taken from Welge et al. and Wiersema et al., with centroid locations as defined in `SUAVE`. The remaining system weights and centroids follow the RSAC component-weight breakdown [69]. For fuel and payload, we assume at approach and landing stage, the fuel mass onboard is 35.5% of the total capacity and is stored primarily in forward-body tanks. Payload is taken at 80% capacity, evenly distributed across the fuselage. The centroid location estimate is based on the fuel tanks location and evenly-distributed payload loading condition. These fuel and payload distribution assumptions shall not be understood as the optimal or standard configuration recommended by the RSAC studies, but to provide a reasonable test case for the planform optimization.

The optimization problem investigates how subsonic stability constraints affect the conceptual design of a wing planform optimized for minimum supersonic drag coefficient, $C_{D,\text{sup}}$. We adopt a two-point optimization case, following the example of Seraj and Martins, in which the wing is evaluated under both a supersonic cruise condition and a subsonic take-off–landing condition [113]. The flight conditions, along with the target lift coefficients (C_L^*)

and static margins (K_n^*), are listed in Table 6.2. The static margin is defined as:

$$K_n = -\frac{C_{M,\alpha}}{C_{L,\alpha}}, \tag{6.1}$$

where $C_{M,\alpha}$ and $C_{L,\alpha}$ are the moment and lift coefficient slope and can be approximated by finite difference by perturbing the AOA by a small amount. In the current study, a perturbation of a tenth of the nominal degree is used for this finite difference approximation.

Since we optimize only the wing planform, imposing a trim condition on the wing results in a highly constrained problem with poor convergence. Instead, we bound C_M so that the wing's pitching moment remains small enough to be compensated by other trim surfaces, with minimal impact on C_L. From Seraj, at subsonic speed at Ma = 0.3, a ΔC_M with magnitude of 0.06 can be achieved from the horizontal tail with minimal impact on C_L with 5% or less the nominal value [112]. Thus, it is determined that $|C_M| \leq 0.06$ is a reasonable moment constraint to be imposed to the optimization problem.

	Mach Number	**Altitude [m]**	**Reynolds Number**	C_L^*	C_M^*	K_n^*
Supersonic	1.8	16764	1.01×10^8	0.2	$\lvert C_M^* \rvert \leq 0.06$	-
Subsonic	0.3	500	8.04×10^7	0.7	$\lvert C_M^* \rvert \leq 0.06$	$\geq 5\%$

Table 6.2: Flight conditions at the supersonic and subsonic design points.

We then run the optimizer, as described in Section 6.2, to minimize $C_{D,sup}$ subject to constraints on supersonic lift target ($C_{L,sup}^*$), supersonic moment bounds ($C_{M,sup}^*$), subsonic lift target ($C_{L,sub}^*$), subsonic moment bounds ($C_{M,sub}^*$), and static margin bounds (K_n^*). Eight design variables are optimized: six geometric parameters and two angles of attack at subsonic and supersonic conditions AOA_{sub} and AOA_{sup}. Their bounds are given in Table 6.3.

6.2 Optimizer

A variety of solver choices can be used for the optimization problem discussed in Section 6.1. In prior work, Seraj and Martins employed the gradient-based optimizer `SNOPT` [35], leveraging the discrete-adjoint capability of `ADflow` to obtain sensitivities of the FFD design variables [113]. However, the current `SUAVE`-based multifidelity workflow is not adjoint-capable (this capability is currently under development in `RCAIDE`, the next-generation framework of

Design Variable	Upper	Lower
δ	7.0°	−8.0°
B	1300 [inches]	1100 [inches]
Λ_{in}	67°	47°
Λ_{out}	60°	35°
Λ_{te}	25°	0°
BW2	0.4	0.32
AOA_{sub}	21°	9°
AOA_{sup}	5°	0°

Table 6.3: List of design variables and their corresponding upper and lower bounds for the optimization problem.

`SUAVE`), so gradients must be approximated via finite differences. Constructing the Jacobian requires perturbing each of the eight design variables, as listed in Section 6.1, and evaluating the objective and all constraints, rendering the cost of each optimization iteration relatively high. Moreover, the surrogate model (`VortexNet`) was trained only to predict the pressure field, without enforcing accurate design parameter to pressure field gradients. Previous validation shows NRMSE of 4.1% and 1.9% for C_L and C_M respectively [121]. Considering the small step size that will be used in $C_{M,\alpha}$ and $C_{L,\alpha}$ computation, the finite-difference gradients computed within these error bounds may be too noisy to guide a gradient-based optimizer reliably.

Hence, instead of relying on gradient information, zero-order algorithms are adopted for the optimization problem described in Section 6.1. These methods explore the design space by tracking and updating a population of candidate solutions, eliminating the need for gradients computation. The stochastic nature, together with the board sampling space, avoids optimizer becoming trapped in local minima [57]. In this work, we select particle swarm optimization (PSO) [51], a stochastic, population-based method inspired by "swarm intelligence" [79]. In PSO, each candidate adjusts its position based on its current trajectory, its own best-found solution, and the swarm's global best, enabling efficient search without gradient evaluations. PSO has been successfully applied to aerodynamic shape optimization in previous studies [73], and its characteristics make it well suited to our problem's dimensionality and search-space requirements.

For a design point (particle) position $\mathbf{x_i} \in \mathbf{R}^6$, the following velocity v_i and position

update is used:

$$\begin{aligned} v_i^{t+1} &= w v_i^t + c_1 r_1 (p_i^t - \mathbf{x_i}^t) + c_2 r_2 (g^t - \mathbf{x_i}^t), \\ \mathbf{x_i}^{t+1} &= \mathbf{x_i}^t + v_i^{t+1}, \end{aligned} \tag{6.2}$$

where $\mathbf{x_i}^t$ is the current particle position, $\mathbf{x_i}^{t+1}$ is the position after 1 step update, v_i^t is the current particle velocity, and v_i^{t+1} is the velocity after 1 step update; the parameters r_1 and r_2 are random numbers drawn from the uniform distribution $\mathcal{U}(0,1)$; the coefficient w is the inertia coefficient, regulating how similar the current velocity to the velocity in the previous iteration; p_i is particle's best location, and g is the global best location; the coefficients c_1 and c_2 are the "memory" and "social" coefficients that pull particle towards its historical best and the global best identified. The initial coefficients are set to $w_{start} = 0.95$, $c_{1,start} = 1.6$ and $c_{2,start} = 1.0$ to allow each particle to explore the design space more independently. Throughout the optimization iterations, the following coefficient scheduling are used:

$$\begin{aligned} w &= w_{start} - 0.5\frac{t}{T}; \\ c_1 &= c_{1,start}; \\ c_2 &= c_{2,start}, \end{aligned} \tag{6.3}$$

where t is the current iteration index, and T is the prescribed total steps. The linear decay of inertia promotes social behavior of particles at later iterations, and encourages the balance between "exploration" and "exploitation" of the optimizer through iterations.

As PSO is designed for unconstrained optimization problems, constraints needs to be incorporated into the optimizer through penalty method. In addition to the supersonic drag coefficient $C_{D,sup}$, the total objective function that PSO optimizes is defined as:

$$f_{total} = f_{obj} + P_{total}, \tag{6.4}$$

where f_{obj} is a normalized supersonic drag count, defined as $f_{obj} = C_{D,sup}/C_{D,reference}$, and $C_{D,reference} = 0.02$, representing the nominal objective; P_{total} is the total penalty term that captures the deviation of a current design point to the constraints bounds, and is computed as:

$$P_{total} = w \sum_i P_i \tag{6.5}$$

where w is the constraint penalty weight and is set to be 2.0. The value of the penalty weight

is determined through trail-and-error by observing the relative scale of f_{obj} and P_{total}. The intention of setting $w = 2$ is to balance the penalty and objective scale.

Individual constraint penalties P_i are calculated as follows:

$$P_{C_L} = \left| \frac{C_L - C_L^*}{C_L^*} \right|; \tag{6.6}$$

$$P_{C_M} = \begin{cases} 0 & \text{if } |C_M| \leq C_M^* \\ \left| \frac{C_M - C_M^*}{C_M^*} \right| & \text{if } |C_M| \geq C_M^* \end{cases} \tag{6.7}$$

$$P_{K_n} = \begin{cases} 0 & \text{if } K_n \geq K_n^* \\ \left| \frac{K_n - K_n^*}{K_n^*} \right| & \text{otherwise,} \end{cases} \tag{6.8}$$

where, the superscript * denotes target (or reference) values.

Throughout the optimizations, particle may arrive at design points that produce unrealistic geometries, such that the tip chords of the wing are negative. In such cases, the particle is reset to a random location within the design space. In addition, if a particle's velocity would carry it beyond the design-space boundaries, boundary reflection is applied to reverse its velocity vector. In this work, the swarm comprises 30 particles and runs for 75 iterations.

6.3 Dataset

The `double-delta-aero` low-speed double-delta wing aerodynamics dataset is used for the surrogate model training. Specifically, we use the Level 3 dataset (as denoted in Table 4.2, which consists of 32 geometries. The specific geometry call signs and their corresponding design variables can be looked up by cross-referencing Table A.2 and Table A.3.

Within the Level 3 dataset, snapshots correspond to AOA at $9°, 11°, 13°, 15°, 17°, 19°$, and $21°$ are selected. Six geometries are selected as the validation set for hyper-parameter tuning, while the remaining geometries are exposed to the model training. In addition to the training set geometries, a hold-out set to evaluate model's geometric generalization performance is assembled using the LHS set discussed in Section 4.3. These geometries are listed in Table A.2.

6.4 Surrogate Model Results

The goal of this section is to quantify the prediction error of the surrogate model and to establish its suitability for conceptual-level aerodynamic shape optimization by demonstrating the geometric generalizability of the proposed surrogate model on random configurations within the design space. Two tests are presented. The first test evaluates the model's prediction accuracy on vehicle configurations from the training geometries set. Specifically, a qualitative discussion of the ΔC_p prediction quality is provided, along with a quantitative comparison between QOIs evaluated from the predicted ΔC_p and those from VLM. These results shall be taken as the reference model performance for the subsequent generalization test. The second test examines the surrogate model's ΔC_p and QOIs evaluations on configurations from the hold-out geometries set. It is concluded that the trained surrogate model provides improved analysis fidelity comparing against that from VLM, such that a prediction accuracy improvement on C_L, C_M, and their derivatives is observed, and the model is generalize well across the design space.

6.4.1 Training Geometries

We first evaluate the root mean square error (RMSE) of ΔC_p predicted by the GNN-based surrogate model and compare the results against those from VLM, to assess the aerodynamic prediction improvement achieved by adopting the surrogate model. In the RMSE computation, the ΔC_p from mapped CFD (ΔC_p^{HF}) is treated as the "ground truth," and the predicted ΔC_p (ΔC_p^{pred}) or VLM's ΔC_p evaluation are compared against it. The RMSEs for ΔC_p prediction on the training set from `VortexNet` is 0.1387. For comparison, the RMSEs of the ΔC_p computed by VLM for the same training set is 0.9070. On average, a six-fold accuracy improvement is achieved by using the GNN-based surrogate model for ΔC_p evaluation.

Figure 6.1 presents a qualitative comparison among ΔC_p obtained from VLM, the `VortexNet` surrogate model, and mapped CFD. The vehicle configurations are selected to provide a representative overview of ΔC_p across a broad design space. The corresponding vehicle configuration call sign and the angle of attack of the free-stream condition are annotated at the top of each ΔC_p figures. Four vehicle configurations, with AOAs ranging from 9° to 18°, are shown. The specific design variables for each configuration can be looked up from Table A.3.

When the ΔC_ps are compared column-wise (i.e., across different sources), it is evident that the surrogate model's predicted ΔC_p more closely resembles the HF reference data than the LF VLM field distribution. Specifically, under conditions where vortex lift is present, such as 07240101SK at 18°, the surrogate model successfully reconstructs the vortex lift suction effects, resulting in a pressure field that better matches the HF results. For conditions where vortex lift is less prominent, such as for 091200-7GZ at 12°, the ΔC_p distributions across VLM, `VortexNet`, and CFD appear similar. It is worth noting that the surrogate model does not introduce artificial vortex lift into the pressure field for such geometric and free-stream conditions. Overall, the GNN-guided surrogate model demonstrates accurate field prediction across a wide range of geometric configurations and AOAs.

Once the ΔC_ps are obtained, the corresponding QOIs can be computed by feeding the ΔC_ps back into `SUAVE`, as outlined in Section 2.3. Figure 6.2 presents the resulting QOIs and their derivatives as a function of AOA, ranging from 9° to 21°, for the vehicle configurations shown in Figure 6.1. The vehicle call signs are indicated at the top of each plots. From left to right, each figure shows the variation of C_L, C_M, $C_{L,\alpha}$, and $C_{M,\alpha}$ with respect to AOA. The QOIs predicted by the GNN-based surrogate model are plotted in blue, while results from VLM and CFD are shown in green and gold, respectively.

Overall, the surrogate model-augmented QOIs estimations are significantly more accurate compared to those from VLM. Specifically, for C_L and $C_{L,\alpha}$, the surrogate model not only reduces the prediction error in the absolute C_L values but also captures the lift curve slope $C_{L,\alpha}$ accurately while the VLM-based computation fails to predict reliably. More importantly, the surrogate model provides a notably improved estimation of C_M: while VLM fails to capture the overall trend of C_M and cannot correctly identify the saddle point, which is critical for the static margin evaluation, the surrogate model successfully reproduces these features. However, accuracy variation is observed among geometries. For example, the C_M prediction accuracy for the 07240101SK configuration does not follow the ground truth reference well. Such deviation is most likely due to the training sample distribution: configuration 07240101SK is one of the only two geometries within the training set that have Λ_{out} larger than Λ_{in}, and the only one between them to have a slightly positive δ, which induce more vortex lift. Such observation indicates that DOE shall be carefully designed for VortexNet-like domain specific surrogates. Nevertheless, when viewed across the results presented, the adoption of `VortexNet` still substantially enhances design fidelity compared to relying solely on VLM for QOIs quantification.

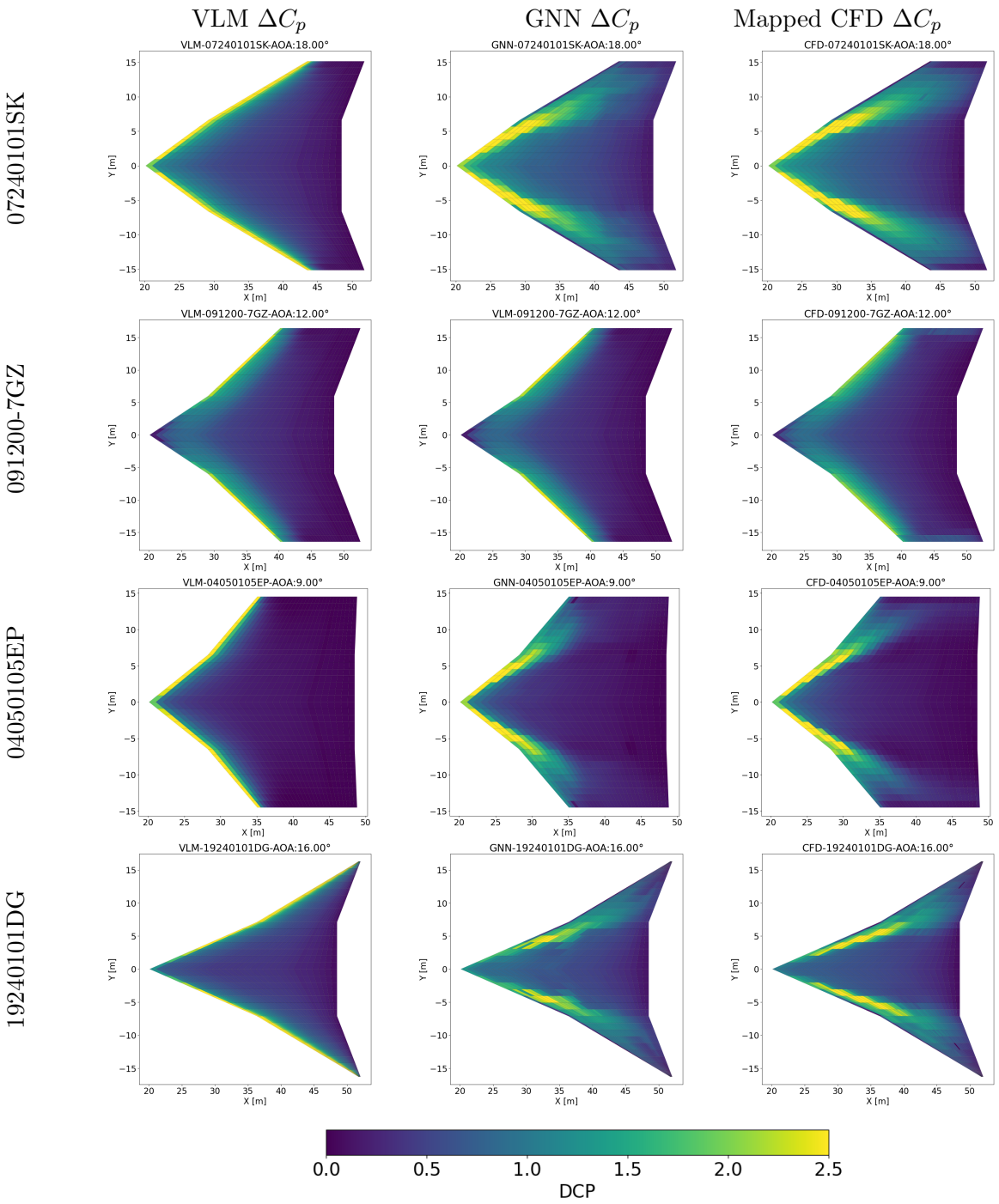


Figure 6.1: Lattice panels and local loading coefficient (ΔC_p) for various configurations and AOAs. Left to right: VLM, GNN, and reference CFD results.

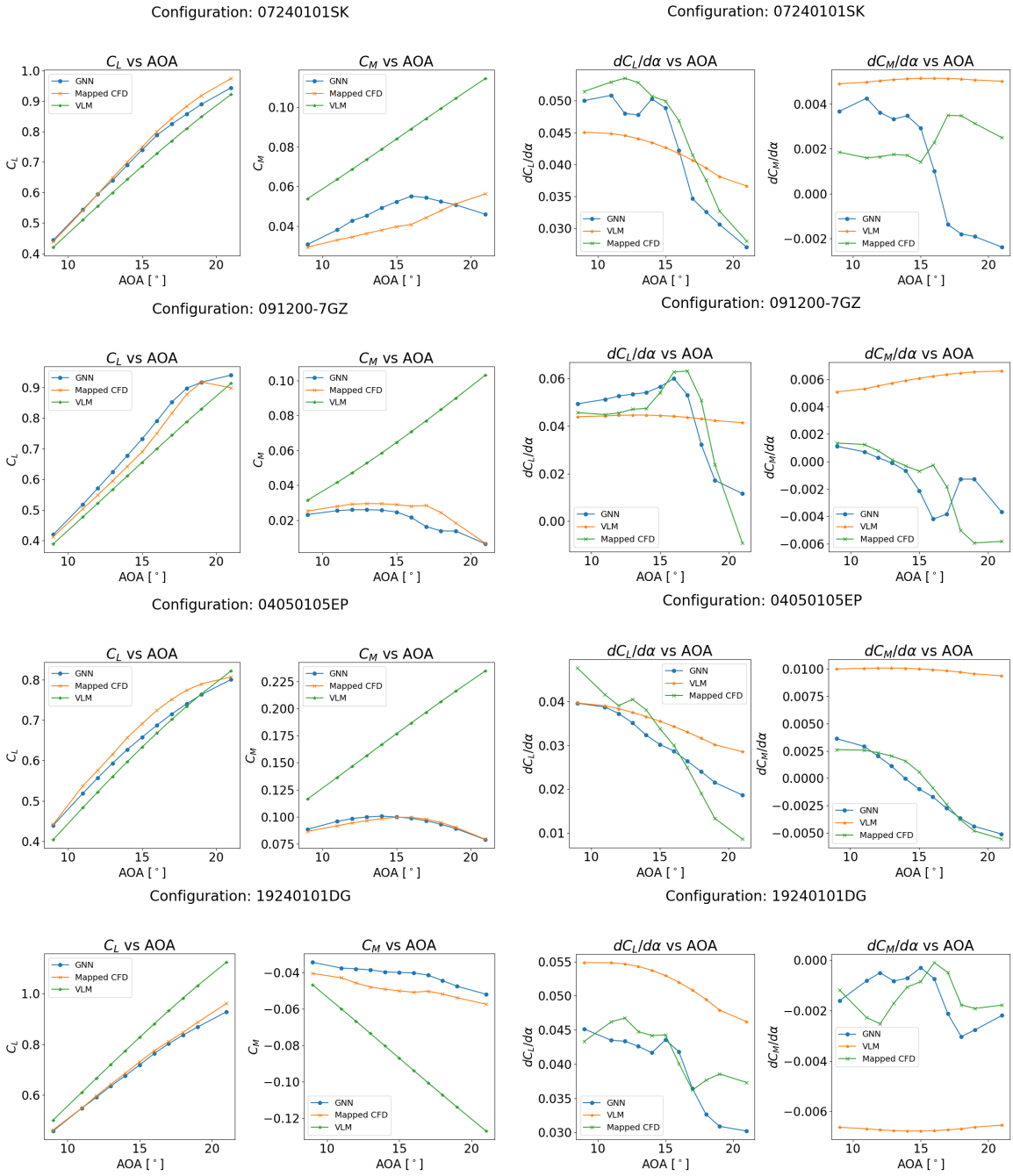


Figure 6.2: QOIs and their derivatives sweep for selected configurations from the training geometries set. AOA from 9° to 21°.

One way to quantify the prediction accuracy across all AOAs for certain vehicle configuration is to use NRMSE, as defined in Equation 3.13. The NRMSE comparison results for C_L and C_M, computed using the surrogate model-augmented ΔC_p and using VLM, across all test set free-stream conditions for configurations within the training geometries set, are presented in Table 6.4. A z-score test with a two-standard-deviation threshold is applied to the GNN-based C_L and C_M NRMSE values to assess the consistency of the surrogate model's prediction accuracy across different geometries. It is observed that the C_L prediction for configuration 171200-7QF and 140500-3DA and the C_M prediction for configuration 17120000ME are considered as outliers. For the remaining configurations, the prediction quality is found to be relatively consistent. The deterioration in accuracy for these outliers is likely due to their design points are near the boundaries of the design space, as similar argument for the 07240101SK can be made.

Call Sign	**VLM C_L NRMSE**	**GNN C_L NRMSE**	**VLM C_M NRMSE**	**GNN C_M NRMSE**
19240101YI	0.1050	0.0197	0.5325	0.1928
19240101XC	0.0967	0.0234	0.3408	0.1016
19240101TK	0.1041	0.0246	0.4649	0.1123
19240101MB	0.1041	0.0246	0.4649	0.1123
19240101DG	0.1473	0.0189	0.8600	0.1645
17120000ME	0.0561	0.0199	10.3141	0.6937
09120000EF	0.0706	0.0263	1.6105	0.1132
07240101SK	0.0778	0.0232	1.0740	0.2192
04050105EP	0.0716	0.0396	0.9631	0.0251
02180104PM	0.0582	0.0332	0.7055	0.0597
192401-8QF	0.1726	0.0446	0.2439	0.0465
171200-7QF	0.1096	0.1117	1.2869	0.2558
141801-3YZ	0.0818	0.0277	3.7651	0.1552
141800-4CI	0.0551	0.0268	4.6750	0.1521
141800-3ZI	0.0610	0.0187	5.7377	0.1695
141800-3QO	0.0610	0.0187	5.7377	0.1695
141800-3KV	0.0649	0.0220	4.4802	0.1126

Continued on next page

Call Sign	**VLM C_L NRMSE**	**GNN C_L NRMSE**	**VLM C_M NRMSE**	**GNN C_M NRMSE**
140500-3DA	0.1715	0.1223	3.2116	0.2583
121800-3MJ	0.0445	0.0153	4.9927	0.1741
120501-4KR	0.1257	0.0346	1.0724	0.1230
091200-7VK	0.0591	0.0445	1.3082	0.2033
091200-7OI	0.0544	0.0465	1.3686	0.1860
091200-7KH	0.0665	0.0516	1.3816	0.1583
091200-7GZ	0.0731	0.0427	1.9054	0.2350
091200-7EY	0.0954	0.0488	0.8243	0.1744
072401-8UA	0.0564	0.0500	1.5375	0.1431
04180105MK	0.0875	0.0734	0.7034	0.1030
04050105NF	0.0571	0.0288	1.1898	0.0422
04050105IF	0.0735	0.0323	0.9852	0.0263
04050105CA	0.0735	0.0323	0.9852	0.0263
04050104PF	0.0914	0.0307	0.9045	0.0350
02050105BD	0.0861	0.0332	0.9563	0.0372

Table 6.4: Comparison of VLM and GNN NRMSE for C_L and C_M across all free-stream conditions for geometries within the training geometries set.

Finally, Table 6.5 summarizes the average NRMSE for each method, based on the data reported in Table 6.4. By incorporating the `VortexNet` into the aerodynamic evaluation workflow, a 2.2× improvement in C_L prediction and an 13.9× improvement in C_M prediction are achieved.

Methods	**C_L NRMSE**	**C_M NRMSE**
GNN	0.0378	0.1494
VLM	0.0848	2.0807

Table 6.5: Averaged NRMSE for the training geometries set, summarized by methods.

6.4.2 Test Geometries

The subsequent test examines the performance of the surrogate model on vehicle configurations that were withheld from training, as listed in Table A.2. The objective of this test is to evaluate the geometric generalizability of the surrogate model for accurate aerodynamic quantification under random geometries within a design space.

Similar to the previous test, the RMSE for ΔC_p evaluation is first investigated. For the hold-out set, the RMSE from `VortexNet` is 0.1863, representing a 34% increase in error. Such accuracy deterioration for interpolated geometric configurations is observed previously in Section 3.4.1. The performance decay is relatively benign, that we conclude the model's field prediction capability is preserved. The RMSE from VLM is 0.9364, which is similar to that presented in the previous test. Four geometries with AOA from 11° to 18° are presented in Figure 6.3 in a similar manner as the ΔC_p comparison in the previous section. The qualitative conclusion regarding the model's prediction capability remains unchanged.

The resulting QOIs, evaluated using VLM, the GNN-based surrogate model, and the CFD reference, for these selected geometries are shown in Figure 6.4. All data points presented were not included in the surrogate model's training. Across all cases, the QOIs predicted by the surrogate model consistently exhibit higher accuracy than those computed using VLM, although the prediction accuracy are relatively poorer than those presented in Figure 6.2. The VLM-based predictions fail to capture the transition behavior beyond linear aerodynamics at moderate-to-high AOAs, due to limitations in its underlying physical model. For some configurations, such as 191301-3JT, the VLM-predicted C_M trends are opposite to those obtained from CFD. As a consequence, the static margin evaluated from VLM will be opposite to those obtained at higher fidelity. In contrast, `VortexNet` is able to capture the trend-wise evolution of C_M and $C_{M,\alpha}$, although the prediction accuracy for certain configurations is relatively lower than that of the other configurations. We conducted a z-score test to identify prediction outliers. Among the 16 geometries, geometry 09190107DL and 1804004MO are believed to have higher prediction error in C_L and C_M, respectively. Individual C_L and C_M NRMSEs are reported in Table 6.6 for data completeness. Design variables for these configurations can be referenced in Table A.2.

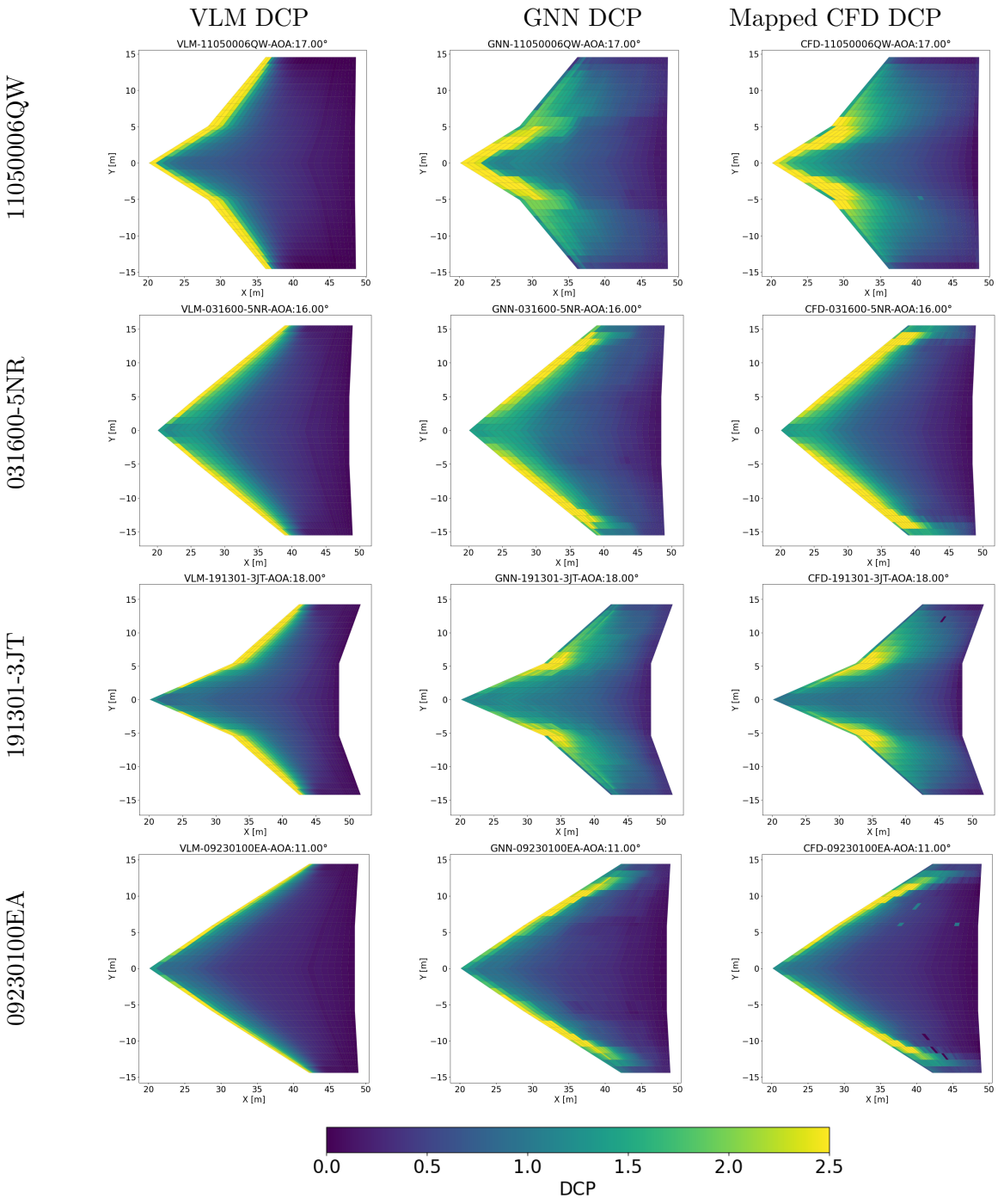


Figure 6.3: Test set ΔC_ps. Left to right: VLM, GNN, and reference CFD results.

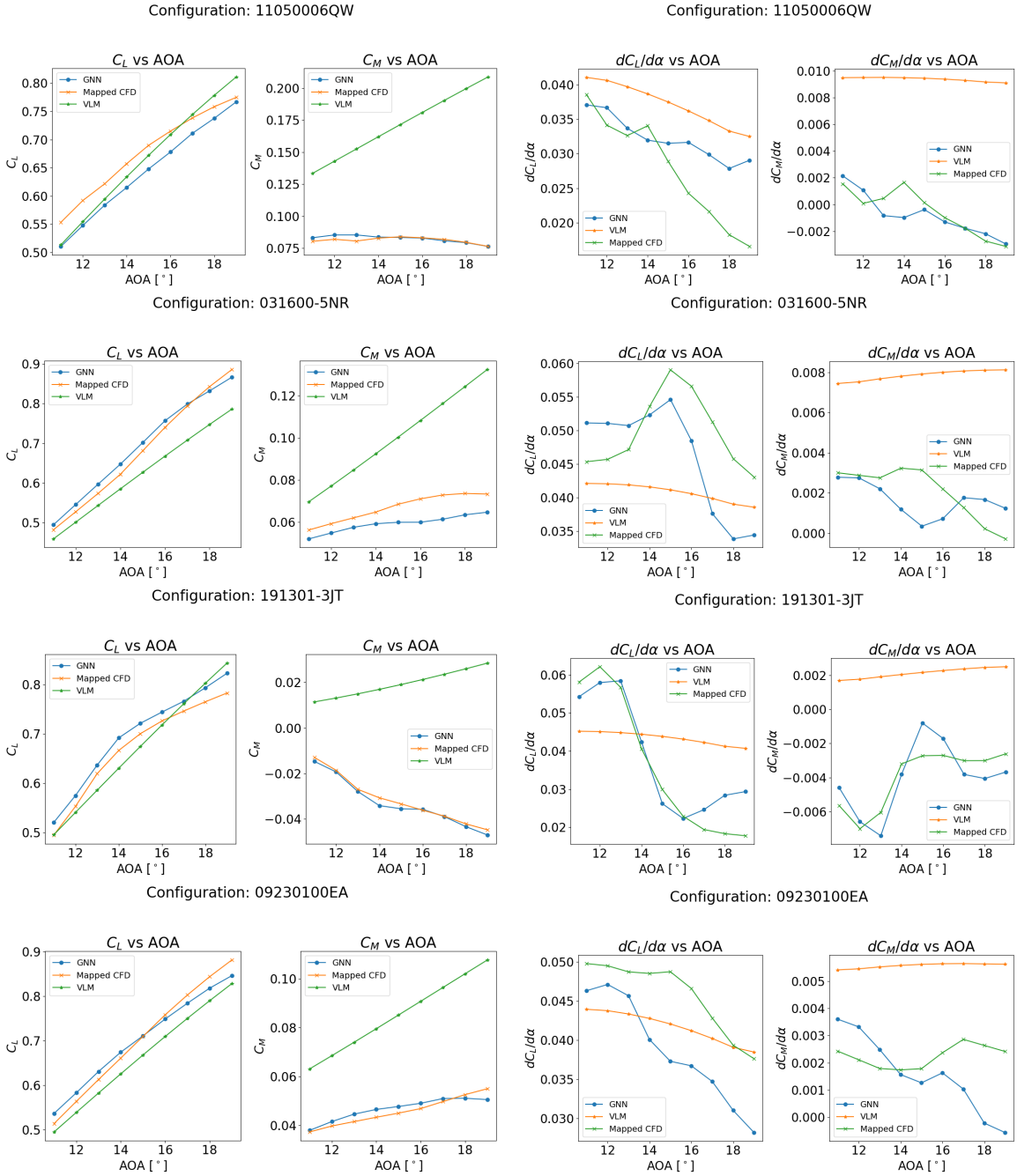


Figure 6.4: QOIs and their derivatives sweep for configurations from the test geometries set. AOA from 11° to 19°.

Call Sign	**VLM C_L NRMSE**	**GNN C_L NRMSE**	**VLM C_M NRMSE**	**GNN C_M NRMSE**
012100-6CB	0.0782	0.0486	0.6124	0.1062
09230100EA	0.0594	0.0290	0.8876	0.0561
050101-7LK	0.1180	0.0567	0.4950	0.1732
09190107DL	0.0590	0.1283	0.9744	0.2132
11050006QW	0.0393	0.0524	1.1519	0.0273
031600-5NR	0.0952	0.0262	0.5505	0.1219
13240101PQ	0.0091	0.0322	1.2861	0.4574
18040004MO	0.0716	0.0487	5.9194	0.7550
150201-4SN	0.1158	0.0916	5.1086	0.4158
060901-7YD	0.0986	0.0704	0.5253	0.1955
02120002AN	0.0945	0.0319	0.9615	0.0713
171901-1NW	0.0890	0.0116	0.8581	0.1619
14150105TT	0.0303	0.0978	2.1422	0.2781
191301-3JT	0.0460	0.0373	1.6878	0.0557
070801-2FS	0.0547	0.0342	1.3646	0.1036
12100003VH	0.1232	0.0552	1.1044	0.1227

Table 6.6: Comparison of VLM and GNN NRMSE for C_L and C_M for geometries within the hold-out test set.

Finally, we summarize the NRMSE values for C_L and C_M across all configurations and AOAs. The results are grouped by QOI evaluation method and are presented in Table 6.7. The GNN-based NRMSEs increase are found to be consistent with the performance deterioration evaluated through RMSE. When comparing the NRMSEs between the VLM and GNN-based methods, the incorporation of the GNN surrogate model in QOIs estimation yields a 1.4× improvement in C_L prediction and a 7.7× improvement in C_M prediction.

Methods	C_L **NRMSE**	C_M **NRMSE**
GNN-based	0.0535	0.2093
VLM-based	0.0739	1.6019

Table 6.7: Averaged NRMSE for the test geometries set, summarized by methods.

Based on both tests reported in this section, it is evident that the inclusion of the trained GNN-based surrogate model leads to significant improvements in aerodynamic estimation fidelity compared to VLM-based predictions. Furthermore, the GNN-based estimations of $C_{L,\alpha}$ and $C_{M,\alpha}$ match those obtained from mapped CFD results, enabling accurate quantification of static margin without requiring direct CFD evaluations within an aerodynamic shape optimization loop. Across all vehicle configurations, the GNN-based aerodynamic predictions exhibit relatively consistent accuracy. Therefore, we conclude that the trained surrogate model is likely to generalize well to other generated geometries within the current design space, that may encounter during the aerodynamic shape optimization.

6.5 Planform Optimization Results

6.5.1 Optimization Results

Two aerodynamic shape optimizations at the conceptual design level are conducted: one using the GNN-guided surrogate model for subsonic C_L, C_M, and K_n, while using VLM for supersonic aerodynamics evaluation, and one using only VLM for all aerodynamic coefficients without the surrogate augmentation. The optimization problem is defined in Section 6.1. For both cases, the optimization uses the same bounds for the design variables, as listed in Table 6.3. While the initial design variable values for each particle may differ due to the random generation of initial particles, all other hyperparameters remain consistent across both runs. Therefore, the differences in the optimized configurations can be attributed to the improved accuracy of subsonic QOIs and static margin evaluation enabled by the inclusion of the GNN-based surrogate model within the aerodynamic shape optimization workflow.

The optimization history for the six geometric design variables is shown in Figure 6.5. The evolution of design variables from the GNN-guided workflow is shown in red, while that from the VLM-based workflow is shown in blue. In Figure 6.5, the solid line represents the design variables corresponding to the global best design, while the dashed line indicates the average values of the design variables across all particles at each iteration. The semi-transparent shaded region illustrates the spread of the particles at each iteration.

Across iterations, the VLM-based optimization converges to an optimal design faster than the GNN-based shape optimization. The spikes in the spread of particles are a result of the optimizer occasionally discovering combinations of design variables that produce a

negative tip chord. When this occurs, the affected particle is reset to a random location within the initial design space, introducing additional challenges for convergence. Nevertheless, after 65 iterations, both optimization workflows yield sufficiently converged designs. The resultant wing shapes at the 1st, 25th, 50th, and the final iteration obtained from both workflows, represented by the camber plane shape and height, are shown in Figure 6.6 and Figure 6.7, respectively.

Design Variable	**VLM-Based**	**GNN-Guided**
δ	$-0.4119°$	$-0.0891°$
Λ_{in}	$65.5376°$	$60.1937°$
Λ_{out}	$56.4473°$	$44.6366°$
Λ_{te}	$11.1109°$	$12.4042°$
$BW2$	0.3706	0.3591
B	1278.3991 [in.]	1202.1273 [in.]

Table 6.8: Comparison of design variables for VLM-based and GNN-guided optimized designs.

The optimized designs from VLM-based and GNN-guided QOIs evaluation exhibit notable geometric differences. The VLM-based wing has a wider span than the GNN-guided case. The location of the inboard-outboard section break is more outboard for the VLM-based design. The leading-edge sweep angles of the inboard and outboard sections for the VLM-based design are higher than those from the GNN-guided approach. Finally, the GNN-guided optimal design has slightly higher trailing-edge sweep. The estimated center of gravity locations of the full vehicle for the VLM-based configuration is at 37.12 meters from datum, while the GNN-guided design has 36.97 meters. The planform area are 434.76[m^2] and 556.62[m^2] respectively, where the GNN-guided design has significantly large wing area. The optimal design variable values are reported in Table 6.8.

Table 6.9 summarizes the constraint satisfaction, the AOAs required for both designs at subsonic and supersonic conditions to meet those constraints, and the objective value of supersonic $C_{D,sup}$ and subsonic $C_{D,sub}$ (reported for reference). The subsonic QOIs for the VLM-based design are evaluated directly using VLM, while those for the GNN-guided design are obtained using the surrogate model-augmented workflow. The supersonic QOIs for both designs are evaluated using VLM. Both designs satisfy the specified constraints with comparable levels of constraint violation residuals.

To satisfy these constraints, the VLM-optimized vehicle operates at an AOA of $12.75°$

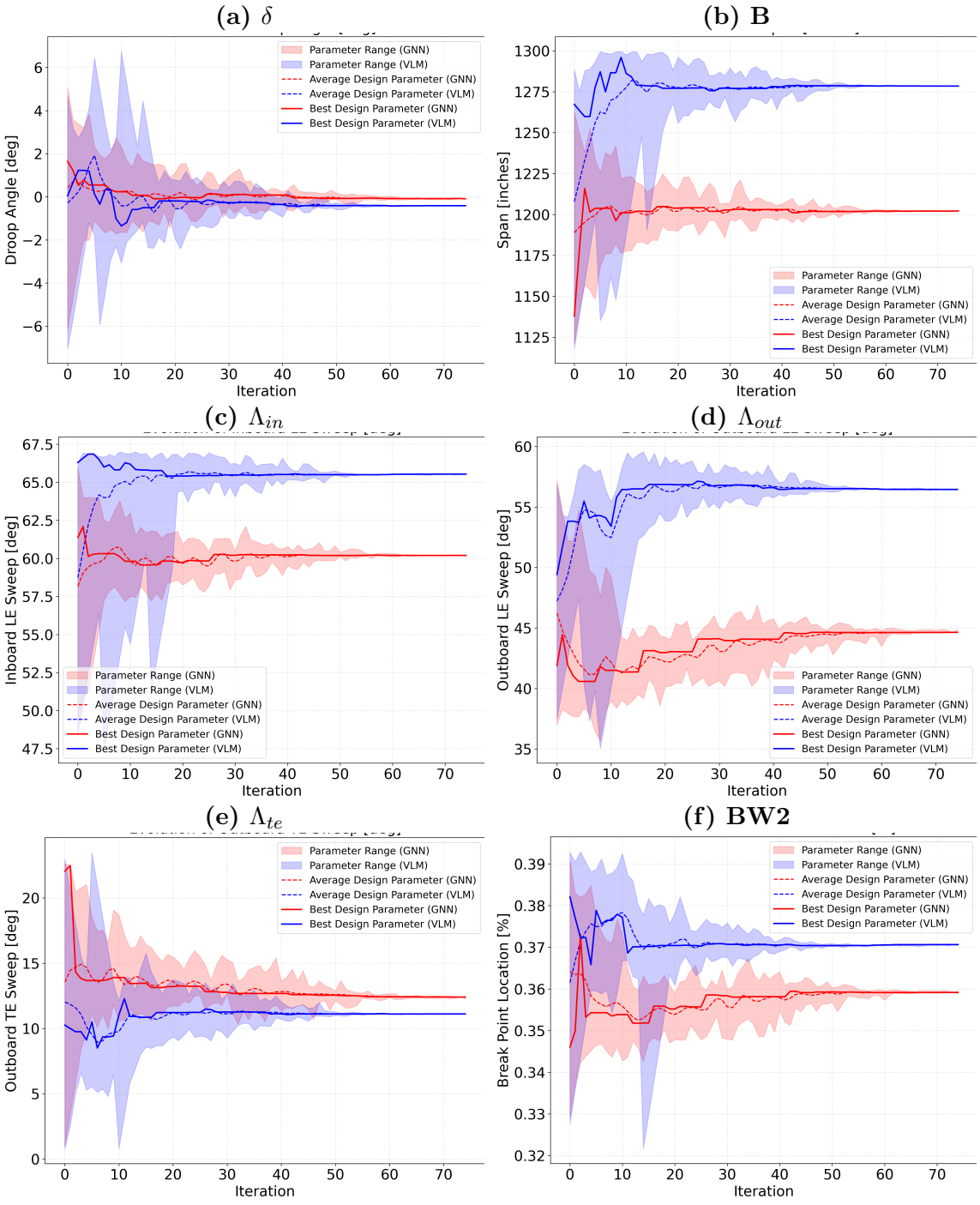


Figure 6.5: Evolution of design variables during optimization.

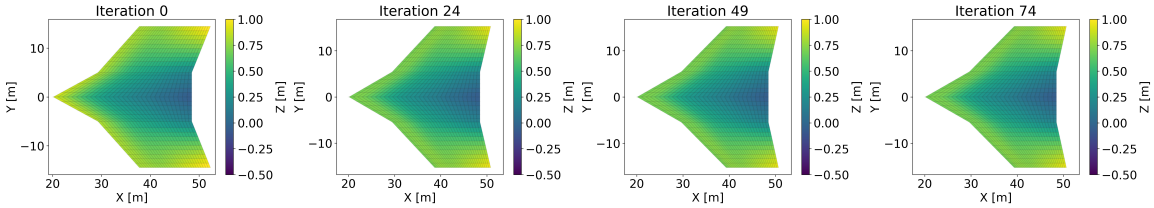


Figure 6.6: Evolution of GNN-guided wing configurations, visualized by the camber plane in the x-y plane with the z-coordinate indicated by color.

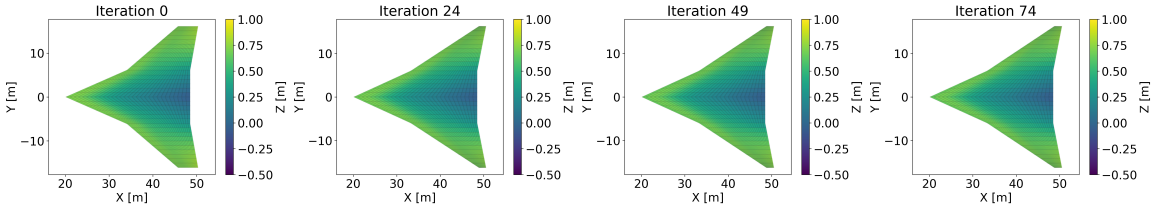


Figure 6.7: Evolution of VLM-guided wing configurations, visualized by the camber plane in the x-y plane with the z-coordinate indicated by color.

for the subsonic condition and 3.34° for the supersonic condition. In contrast, the GNN-guided vehicle operates at higher AOAs, specifically 15.52° and 3.96° for the subsonic and supersonic conditions, respectively. The resulting supersonic C_D, which is to be minimized in the aerodynamic shape optimization problem, is 0.0144 for the VLM-optimized vehicle and 0.0174 for the GNN-guided configuration. The inclusion of the GNN model for QOIs and constraints evaluation introduces a 20.83% increase in supersonic drag for the optimized configurations. On the other hand, the subsonic C_D evaluated for the GNN-guided configuration is 29.3% higher than that of the VLM-based design.

6.5.2 Validation

To further understand the aerodynamic differences between the optimal configurations from both workflows, we examine the ΔC_p at their corresponding subsonic AOAs that satisfy the design constraints (listed in Table 6.9). Figure 6.8(a) to Figure 6.8(b) show the comparison for the two optimal configurations, from the VLM-based and GNN-guided workflows, under two methods for ΔC_p computation: ΔC_p^{LF} evaluated from VLM and ΔC_p^{pred} evaluated from GNN. As shown in Figure 6.8(a), the lift is concentrated at the leading edge of the vehicle. The resultant center of pressure, calculated using VLM ΔC_p, is located at 38.40 meters, corresponding to a 1.28 meter behind the center of gravity.

Metric	VLM-Based	GNN-Guided
Constraints		
Supersonic C_L	0.2000	0.1986
Supersonic C_M	-0.0291	-0.0059
Subsonic C_L	0.7002	0.6983
Subsonic C_M	-0.0323	0.0485
Subsonic K_n	0.0555	0.0576
Design Variables		
Subsonic AOA	12.7539°	15.5215°
Supersonic AOA	3.3393°	3.9636°
Objectives		
Subsonic C_D	0.1582	0.2046
Supersonic C_D	**0.0144**	**0.0174**

Table 6.9: Comparison of performance metrics for VLM-based and GNN-guided optimized designs.

For the GNN-guided optimal configuration (Figure 6.8(b)), a less concentrated ΔC_p at the leading edge is observed. In addition to the primary vortex contribution, the vortex is creating lift at the outboard section of the wing. The resulting center of pressure, when evaluated from the GNN-predicted ΔC_p, is located at 36.00 meter, corresponding to a 0.96 meters before its center of gravity.

As a validation effort, we conducted two additional CFD simulations for the optimized configurations using the same CFD settings. Since a direct comparison between the CFD-computed C_p and the ΔC_p shown in Figure 6.8(a)–(b) would be both cumbersome and potentially misleading, the CFD surface pressure coefficients are projected onto the lattice panels to generate CFD-based ΔC_p ground truth. The resulting ΔC_p fields for the VLM-based and GNN-guided configurations are shown in Figure 6.8(c) and (d), respectively.

Examining the ground truth result for the VLM-based configuration in Figure 6.8(c), a strong primary vortex originating from the inboard section and extending outward is observed. It is observed that the VLM-based pressure field completely neglected this vortex system, indicating an insufficient modeling fidelity for the optimized configuration. When comparing the GNN-guided ΔC_p prediction (Figure 6.8(b)) to its CFD ground truth (Figure 6.8(d)), the surrogate model tends to have strong vortex dissipation, such that the region strongly impacted by vortex, as indicated by $\Delta C_p \geq 1.5$, is smaller than that of the ground truth. This discrepancy in ΔC_p magnitude may lead to inaccuracies in the magnitude estimation of C_M. Overall, however, the GNN-estimated ΔC_p is qualitatively similar

to that from CFD.

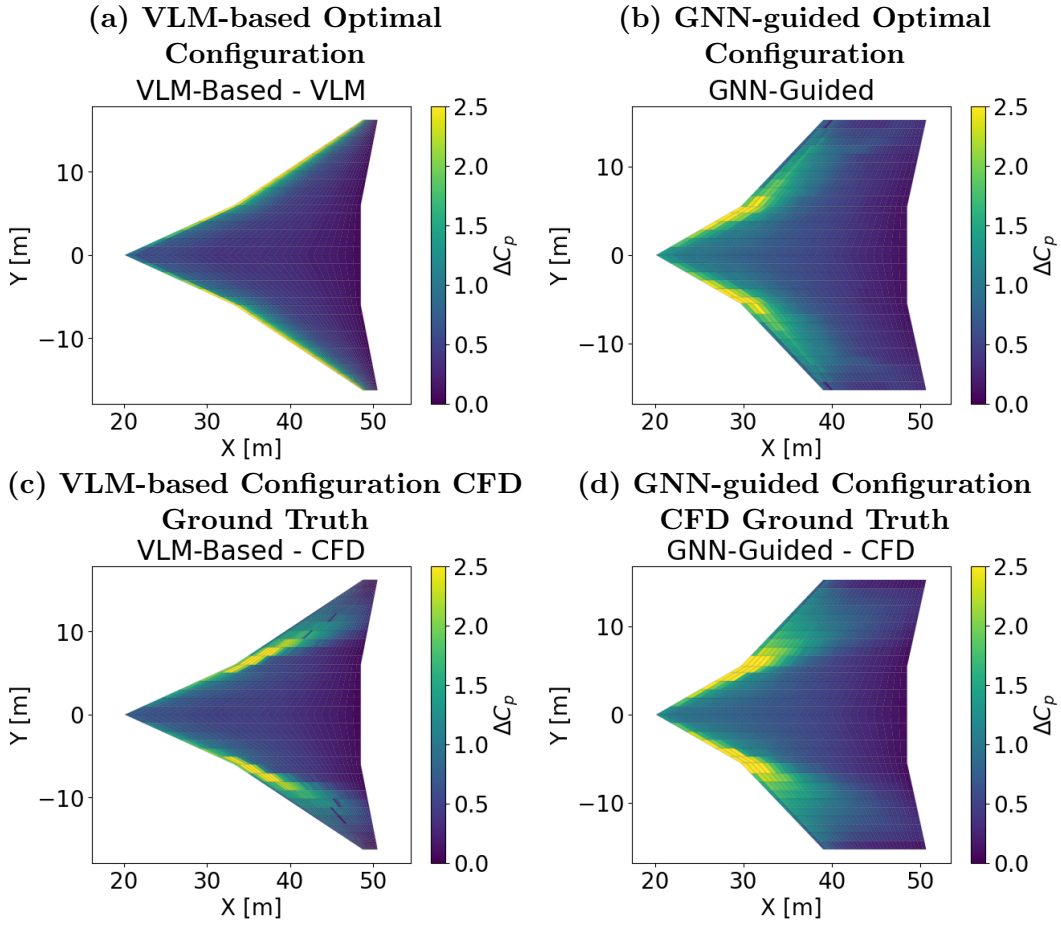


Figure 6.8: Optimal wing design ΔC_p distributions: (a) VLM-based configuration evaluated using VLM, (b) GNN-guided configuration evaluated using the GNN surrogate model, (c) VLM-based configuration evaluated using CFD, and (d) GNN-guided configuration evaluated using CFD.

Finally, the optimized design is validated to ensure that it meets the design requirements by presenting the AOA sweep results, with the QOIs evaluated using CFD. We ran CFD simulations for both the VLM-based and GNN-guided optimized configurations over AOA values ranging from 6° to 20°, with a one-degree interval. The C_L and C_M are presented in Figure 6.9. In addition to these coefficients, K_n is calculated from Equation 6.1 using $C_{L,\alpha}$ and $C_{M,\alpha}$ computed through central differences. The results is presented in Figure 6.10.

Across QOIs, the GNN-augmented evaluations resemble the CFD results more closely than the VLM-based assessment. For C_M (Figure 6.9(b)), although the magnitude of the

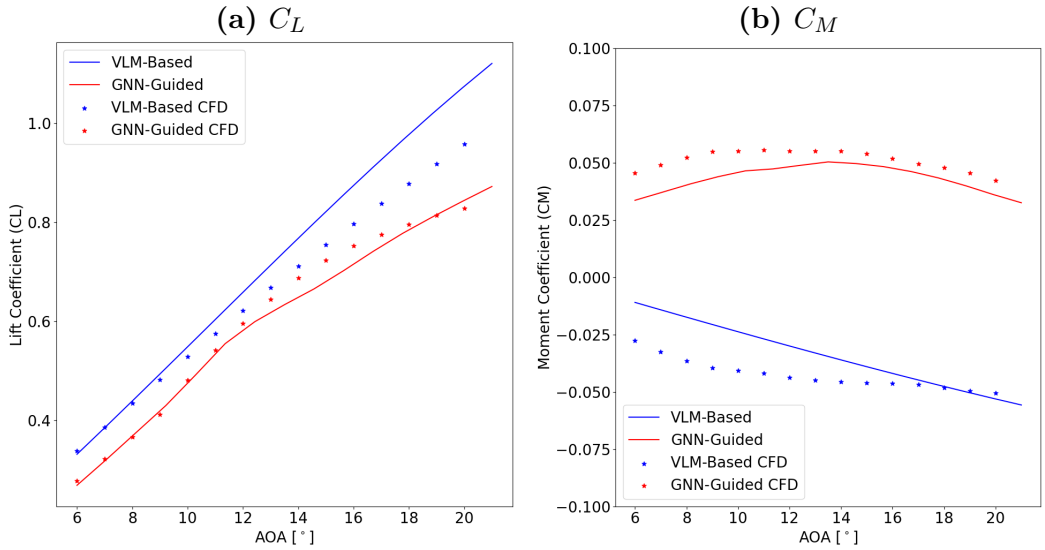


Figure 6.9: Comparison of QOIs for (a) C_L and (b) C_M for optimal configurations.

GNN-augmented evaluation differs slightly from the CFD reference, the deviation remains approximately constant across AOAs. Higher deviation at AOA less than 9° is observed, and this shall not be surprising as the surrogate model was not trained for these AOAs. The origin of the C_M magnitude deviation is most likely due to the discretization error in the surface pressure projection step and the modeling error introduced by evaluating C_M using the camber plane profile in the current workflow rather than computing surface integration as in CFD. C_M evaluated using VLM, by contrast, does not match the CFD reference well, as it fails to capture the C_M plateau from 13° to 17°. For C_L, it is observed that both methods predicts well for AOA $\leq 11°$. At higher AOAs, the VLM-based approach begins to exhibit larger errors as the reference $C_{L,\alpha}$ starts to decrease. In contrast, the GNN-guided approach is able to predict $C_{L,\alpha}$ with relatively good accuracy up to 18°. In addition, the CFD results reveal that both the VLM-based and GNN-guided configurations fall short of the $C^*_{L,\text{sub}}$ target. Specifically, at their respective operating conditions, the VLM-based configuration yields a C_L of 0.657, whereas the GNN-guided configuration yields a C_L of 0.739. Nevertheless, the deviation from C^*_L remains within 6%, which is acceptable in many conceptual design applications.

In Figure 6.10, the calculated K_n for the GNN-guided optimal configuration is plotted in red, where the solid line denotes the value obtained from the `VortexNet`-based workflow and the dashed line denotes the value obtained from CFD. The solid blue line shows the K_n

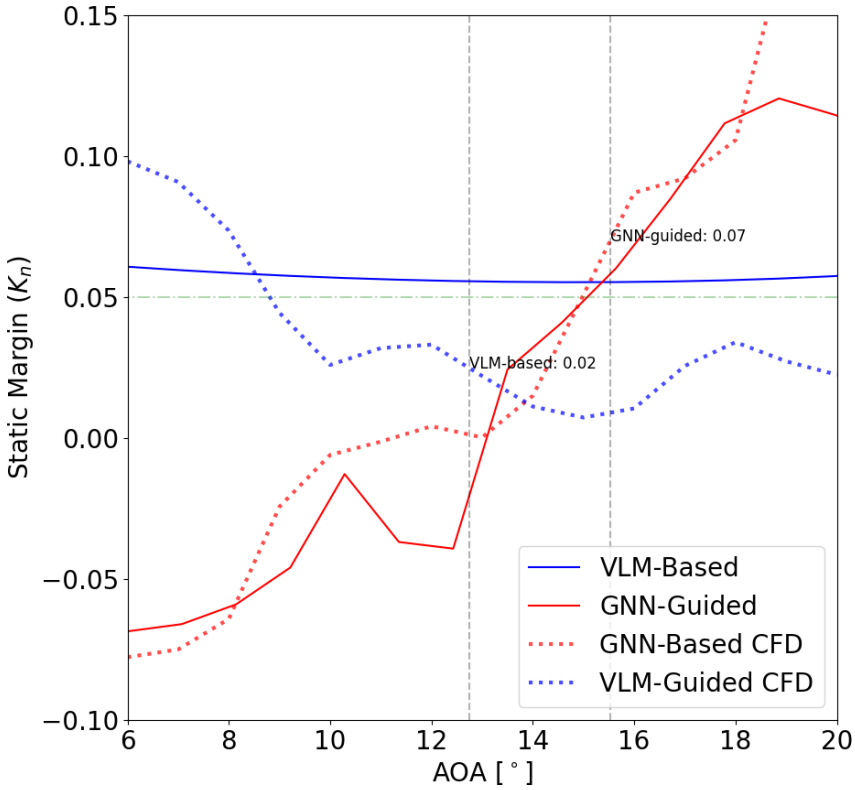


Figure 6.10: Comparison of static margin K_n for optimal configurations.

evolution predicted using VLM, whereas the dashed blue line shows the corresponding CFD result. The green dash-dot line indicates the 5% static margin requirement, and the two vertical dashed lines mark the corresponding subsonic operating conditions selected by the optimizer. From this figure, the VLM-based workflow fails to predict K_n accurately, as it suggests an unconditionally stable vehicle across all AOAs and does not identify the unstable region. Under CFD evaluation, the VLM-based configuration has $K_n \leq 5\%$ for AOA values beyond 9°. In contrast, the GNN-guided workflow successfully captures the K_n evolution within the operational range, where AOA remains below 18°. A small deviation is observed near an AOA of 12°. The cause of this deviation requires further CFD validation and could potentially be reduced through improvements to the GNN surrogate model training. Nevertheless, this improvement in K_n estimation, when applied to a stability-constrained aerodynamic shape optimization problem, can guide the optimizer toward configurations that better satisfy the design constraints and more closely align with the design intent,

thereby supporting subsequent stages of the design pipeline.

Finally, Table 6.10 presents the CFD-evaluated K_n values for intermediate optimal configurations obtained throughout the optimization process. Because the geometric changes from iteration 40 to 75 are subtle, only the intermediate optimal geometries at the 10th, 20th, 30th, and 40th iterations are selected for spot-check validation. The corresponding design parameters and CFD-evaluated K_n values, evaluated using central scheme with a 0.5° step length, are listed side by side. The results show that, when VLM-based evaluation is used, none of the intermediate optimal configurations satisfies the 5% static margin requirement, indicating that the constraint is ineffective due to modeling bias. In contrast, the GNN-guided approach consistently produces configurations that satisfy the static margin constraint even under CFD validation. This difference provides strong evidence of the benefit of increased fidelity in design optimization: while the low-fidelity analysis fails to enforce the static margin constraint effectively, the multi-fidelity approach enables such constraint to be effective during the optimization.

Method	Iter.	$\delta[^\circ]$	$\Lambda_{\text{in}}[^\circ]$	$\Lambda_{\text{out}}[^\circ]$	$\Lambda_{\text{te}}[^\circ]$	BW2	B [in.]	$\text{AOA}_{\text{sub}}[^\circ]$	$K_{n,\text{CFD}}$
VLM	10	-1.35	66.19	53.41	11.05	0.38	1286.10	12.83	3.30%
VLM	20	-0.19	65.41	56.86	11.22	0.37	1277.04	12.75	1.64%
VLM	30	-0.27	65.48	56.79	11.26	0.37	1276.97	12.74	2.80%
VLM	40	-0.35	65.51	56.60	11.12	0.37	1277.97	12.75	2.78%
GNN	10	0.24	59.81	41.50	13.90	0.35	1201.06	15.81	8.06%
GNN	20	-0.05	59.74	42.97	13.21	0.36	1203.88	15.77	9.45%
GNN	30	0.08	60.20	41.11	12.77	0.36	1202.74	15.63	9.97%
GNN	40	0.08	60.23	44.08	12.68	0.36	1202.99	15.59	6.62%

Table 6.10: CFD validation of K_n for intermediate optimal configurations obtained across VLM-based and GNN-guided optimization iterations.

6.6 Discussion

This chapter demonstrates the integration of the proposed MF surrogate modeling workflow into a conceptual ASO problem. The study is intended as a preliminary demonstration rather than a fully matured SST planform development case. It tests whether the fidelity improvement provided by `VortexNet` can meaningfully affect conceptual design decisions. The two-point optimization problem is meaningful because its design goals are competing:

the supersonic condition drives drag minimization, while the subsonic condition imposes lift, moment, and static-margin constraints sensitive to nonlinear vortex-dominated aerodynamics.

The validation results show that `VortexNet` provides useful fidelity augmentation within the sampled design space. Trained on 32 geometries and evaluated on both training and holdout geometries, the surrogate-augmented workflow predicts field quantities and integrated QOIs more accurately than the VLM-based workflow. This improvement is especially important for C_L, $C_{L,\alpha}$, C_M, and $C_{M,\alpha}$, which directly determine the static margin. Although accuracy varies across geometries, the model preserves the dominant aerodynamic trends needed for conceptual-level optimization.

The optimization results show that improved aerodynamic fidelity changes the resulting design. The VLM-based and GNN-guided workflows converge to noticeably different planforms. The VLM-based design achieves a lower predicted supersonic drag, but CFD spot-check validation shows that its static-margin constraint is not effectively enforced. In contrast, the GNN-guided workflow produces intermediate and final designs that consistently satisfy the static-margin requirement under CFD validation. This difference shows that the added fidelity does more than reduce prediction error: *it changes the optimization direction* by steering the search toward designs that are feasible under HF assessment.

Several limitations remain. The demonstration considered only wing planform variables, while the fuselage, propulsion system, and tail surfaces are held fixed. In addition, `VortexNet` introduces modeling errors. These errors are not explicitly modeled in the current optimization workflow, indicating the need for future studies on uncertainty quantification and robust shape optimization under surrogate-modeling error.

Despite these limitations, the results support the main claim of this chapter. Even on a limited-scale test problem, increased aerodynamic fidelity at conceptual-design cost enables a design task that is historically difficult to incorporate early in the aircraft design cycle: stability-constrained planform selection. In this sense, `VortexNet` acts as a geometrically generalizable correction knowledge base, allowing information learned from HF simulations to guide early design exploration. Rather than using HF analysis only for late-stage validation and modification, the proposed workflow allows HF information to influence the design search itself. This capability provides a promising path toward future MDO and product development workflows, where higher-fidelity analysis must be considered early, but repeated HF simulation remains prohibitively expensive.

Chapter 7

Conclusion

This dissertation addresses a critical limitation of current conceptual design tools for supersonic transport (SST) development: traditional conceptual design tools, which heavily utilize low-fidelity (LF) analysis due to budget limitations embedded in the product development cycle, fail to capture the nonlinear aerodynamics necessary for early-stage stability assessment and stability-constrained design. To bridge this fidelity–cost gap, a multi-fidelity, graph-neural-network-based field-prediction surrogate model, `MF-VortexNet`, is developed to assimilate LF vortex lattice method (VLM) predictions with high-fidelity (HF) RANS CFD data. The neural network architecture, data pipeline, and validation efforts are presented in Chapter 3. Subsequently, using this surrogate model, this dissertation examines model scaling behavior under increasing computational scale (Chapter 5) and integrates the surrogate model into an multidisciplinary design optimization (MDO) framework, while demonstrating the fidelity gains of the proposed modeling methodology in comparison with traditional LF methods (Chapter 6). Through these experiments, this dissertation demonstrates that fidelity enhancement can be introduced into early-stage design without sacrificing the computational efficiency required for design space exploration.

7.1 Thesis Contribution

1. A graph neural network (GNN)-based multi-fidelity surrogate model for field predictions. The first contribution of this dissertation is the development of `MF-VortexNet`, a GNN-based surrogate model for multi-fidelity aerodynamic field prediction. Unlike conventional surrogate approaches that target only scalar aerodynamic outputs, `VortexNet`

learns a mapping from low-fidelity lattice-wise aerodynamic loading to quasi-high-fidelity surface pressure fields. This field-prediction capability enables the recovery of arbitrary field-derived quantities of interests (QOIs) within the conceptual design environments, thereby moving beyond coefficient-level correction toward field-level fidelity enhancement. Architecturally, the model combines graph-based message passing for local stream-condition–flow–geometry correlation discovery. The architecture, data pipeline, and training protocol is specifically designed for domain-specific learning, such that model can be trained with less data while keeping consistent performance when measured in prediction error respect to CFD references. The results showed that the surrogate significantly improves lift and pitching-moment prediction relative to standalone VLM, while retaining inference costs comparable to low-fidelity analysis. In addition, the model demonstrated geometric generalizability to unseen geometric configurations and flow conditions. Such prediction merits are enabled by the nonlinear feature encoding and decoding of the geometry and flow features, as investigated by a latent space ablation study.

2. Creation of an open-source multi-fidelity aerodynamic dataset for field-based learning. The second contribution is the development and release of the open-source `double-delta-aero` dataset, designed to support field-based multi-fidelity learning for nonlinear aerodynamic flows. The dataset was constructed for a parametric family of double-delta wings and includes 272 unique geometries evaluated across multiple angles of attack (11° to 19°) at low-speed conditions (Ma = 0.3) using both VLM and RANS CFD. This dataset fills in a gap in open-domain SST low-speeds aerodynamics, featuring vortex-dominated flow behavior under a wide planform design space. By using a structured DOE that supports future sensitivity analysis, extension capability, and uncertainty quantification, the dataset provides a benchmark for future field-capable or scalar-valued surrogate model development under single or multiple fidelities.

3. Empirical scaling laws for multi-fidelity GNN surrogate. The third contribution is an empirical investigation of how `VortexNet` performance scales with both training data size, model size, and computation. Using hierarchical subsets of the `double-delta-aero` dataset, the dissertation examined training sets spanning multiple sampling densities and model sizes. The results show that generalization improves primarily through data scaling, with test loss decreasing up to approximately $D = 640$ and following a clear power-law-like

trend most consistently in the intermediate regime of $80 \leq D \leq 320$. In contrast, the largest dataset size ($D = 1280$) underperforms slightly, suggesting diminishing information gain from denser sampling within the same bounded design space. This study therefore provides a quantitative scaling assessments for a field-predicting, multi-fidelity GNN surrogate, while also showing that model-size scaling is comparatively weak in terms of absolute best test loss. Although moderately larger models exhibit somewhat higher data-utilization exponents, the results indicate that directly increasing latent feature dimension yields diminishing returns, and that a model of roughly 10^6 parameters is adequate under the present workflow and prediction task. Beyond reporting empirical scaling exponents, the analysis connected model performance to geometric sampling density and estimated that approximately 36 to 64 geometries, corresponding to about 180 to 360 total snapshots, are sufficient to achieve near-optimal performance for the current problem setting. The study also investigated the trade-off between data utilization efficiency and computational cost, including a compute-scaling law for the best achieved test loss, highlighting a broader compute–data tradeoff that has not yet been systematically characterized for this class of aerodynamic surrogate models.

4. Integration of the multi-fidelity surrogate into conceptual design and MDO environment. The fourth contribution is the integration of the `VortexNet` surrogate into the `SUAVE` conceptual design environment for aerodynamic shape optimization under stability constraints. By augmenting VLM-computed local pressure loading within the design loop, the framework enables fast online evaluation of higher-fidelity-informed aerodynamic quantities, such that an accurate and fast static stability evaluation became possible. Applied to a SST wing planform optimization problem, the integrated workflow achieved substantial error reduction in the estimation of lift, pitching moment, and static margin relative to a conventional LF-only workflow. More importantly, the dissertation showed that the fidelity gain materially changes optimization outcomes: without the surrogate augmentation, the LF workflow mis-characterizes stability-related derivatives and can converge to apparently optimal designs that do not satisfy constraints when assessed at higher fidelity in later aircraft design process. With the multi-fidelity surrogate in the loop, the optimization produces designs that remain consistent with the intended stability requirements. This demonstrates that the proposed framework can enable stability-constrained conceptual design in regimes where traditional low-fidelity tools are insufficient. More broadly, this result

suggests that geometrically generalizable MF surrogate models can act as reusable LF–HF correction knowledge bases, enabling higher-fidelity information to guide early-stage design search and support future MDO and product development workflows.

7.2 Future Work

With the recent efforts in tightening design, analysis, and optimization in new product development cycle through automation and knowledge extraction, as well as the industrial trend of pushing deeper into previously underexplored design space, the multi-fidelity method presented in this dissertation bridges the gap between solvers, and provides a mean to iterate design under an assimilated framework. From such perspective, several avenues for continued research exist. These directions span both immediate extensions of the present thesis and a broader research agenda in which multi-fidelity surrogates evolve into uncertainty-aware digital twins for engineering decision support.

1. Modeling robustness and generalizability improvement. Although `VortexNet` demonstrated strong performance and meaningful fidelity gains, degradation in field reconstruction quality was observed for unseen geometries, and optimization performance becomes sensitive when training data are sparse near the boundaries of the design space. Through this dissertation, it has been observed that the feature encoding and correlation robustness and efficiency are the key enabler of a robust and generalizable model performance. From the scaling experiments, it is also observed that the GNN architecture, along with the current data and training pipeline, have a relatively narrower power law scaling regime comparing with those obtained from transformer architecture. The model performance improvement remains architecture dependent, such that careful feature design is required to harvest full performance of the trainable weights. Such problem is less a concern for domain-specific surrogate modeling, but raises concerns on scaling up to larger scale modeling needs, such as knowledge transferability among products. Future work should therefore focus on improving feature-capturing efficiency and extrapolation robustness through richer graph constructions, refined edge-connectivity definitions, dimension reduction techniques, and hierarchical pooling mechanisms. Inspirations can be drawn from recent progresses in multi-grid GNNs, neural operators, and transformers. These efforts would strengthen the

model's ability to support broader design space and discretization types and more demanding aerodynamic regimes.

2. Expand to more conceptual design disciplines. To promote wider adoption of the proposed framework, it is necessary to push the current conceptual planform design workflow toward full-vehicle and system engineering applications. The present optimization study models only the main wing and considers a limited set of operating conditions. Future work should incorporate additional aerodynamic surfaces, trim and control-surface sizing, and fuselage–wing interference effects, while expanding the flight-condition set toward a more representative mission profile, a systematic MDO direction that fully harness the capabilities of `SUAVE`. In addition, as the proposed surrogate modeling technique is field-generic, field variables beyond pressure can be incorporated for multi-disciplinary multi-fidelity problems. In this dissertation, the focus was on aerodynamic pressure-field correction in a VLM-to-RANS setting, but the same framework could be extended to other field quantities such as thermal and velocity fields, and additional fidelity hierarchies. Promising targets include wall-shear stress, local boundary-layer properties, and thermal loads, which are either poorly resolved or entirely unavailable in classical low-fidelity methods. Similarly, the framework should be generalized beyond two fidelity levels so that information can be interpolated across a richer hierarchy of analysis tools and development stages. This would move the framework closer to a unified engineering knowledge model rather than a task-specific aerodynamic regressor.

3. Cross-fidelity training sample allocation. One particular strength of multi-fidelity modeling, as discussed in Section 1.4, is that training data can be curated from multiple fidelity sources under a resource-allocation strategy. In practical surrogate-model construction, it is often desirable to leverage lower-fidelity data as much as possible to reduce the offline cost of dataset generation, resulting in an intentionally unbalanced dataset size between LF and HF samples. Within the scope of this dissertation, `VortexNet` was trained using a one-to-one paired LF–HF dataset, where each VLM solution was associated with a corresponding CFD reference. However, an important future direction is to extend the training pipeline so that the model can benefit from a larger amount of VLM data while requiring only sparse CFD supervision. The current multi-fold training strategy provides a possible pathway for such extension: the first stage could be trained primarily as a

low-fidelity self-regression task by increasing the physical loss weight, thereby allowing the model to learn geometry–flow correlations from abundant LF samples before being corrected by sparse HF data. Alternatively, transfer-learning strategies, such as the one presented by Shen et al. [123], could be explored by pretraining `VortexNet` on LF data only and subsequently fine-tuning it with limited HF samples. Techniques such as freezing shallow feature-encoding layers may help preserve globally learned geometric and flow representations while reducing the amount of HF data required for calibration. More broadly, future work should quantify the scaling behavior associated with LF–HF sample allocation, including how prediction accuracy changes with the ratio of HF to LF data. Such analysis would provide practical guidance for determining how much expensive HF data is needed when abundant LF data are available.

4. Enabling uncertainty quantification. All models are approximations, this holds true for LF, HF, and the surrogate model itself. While the present dissertation establishes the feasibility of cross-fidelity field synthesis, practical engineering deployment also requires calibrated estimates of predictive confidence. In the longer term, the surrogate framework should be extended with uncertainty quantification methods capable of producing uncertainty bounds for interpolated high-dimensional field predictions. Such uncertainty estimates would allow the model to distinguish between cases where available evidence is sufficient for design commitment and cases where additional HF analysis is needed. Such knowledge, in return, can be used to guide adaptive design-of-experiments strategies: With quantification of surrogate model's correction accuracy at a given design point, adaptive sampling strategies that identify the value of additional query point and suggest the fidelity levels required, can be developed to reduce the cost of HF dataset generation. In this setting, the surrogate modeling becomes an active knowledge-extraction problem, balancing the cost of new simulations or experiments against the expected reduction in model uncertainty. This step is central to transforming the surrogate from a predictive tool into a decision-support model aligned with human engineering workflows.

5. Human-centered digital twin across product development cycle. Beyond the previously mentioned immediate extensions, this thesis motivates a broader research program in which multi-fidelity surrogates become components of a human-centered digital twin

for engineering design. Building on the high-dimensional cross-fidelity knowledge correlation, a logical next step is the development of an uncertainty-aware digital twin framework that integrates assessments across fidelity levels and development phases into a unified and evolving model of the product and its design space. In this view, the surrogate is not only a regressor, but a reusable knowledge model that assimilates heterogeneous evidence across product development phases. With such developments, using a cross-fidelity field-surrogate model as a tool, engineering knowledge and abstractions can be preserved, even at a higher data generation throughput that is beyond human diagnostic capacity. Such a digital twin framework opens the door to agentic engineering workflows. With uncertainty-aware multi-fidelity models at the core, autonomous agents could coordinate design exploration, evidence acquisition, and natural-language knowledge summarization across the development life cycle, while keeping humans in the loop for judgment and commitment decisions. Although this vision lies beyond the scope of the present dissertation, the work in this dissertation establishes early-stage enabling elements: graph-based representation of field assessments, cross-fidelity information synthesis, and integration of learned models into design environments.

Appendix A

Dataset "double-delta-aero" Geometries

Using the design space defined in Chapter 4.2, with parameterization presented in Figure 4.1, the following 256 geometries are created with sampling method described in Chapter 4.3.

Table A.1: Design parameter table for the `double-delta-aero` dataset.

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
051800-7KM	-7.297	52.313	52.578	11.328	0.34109	1284.38
05180004HD	3.953	52.313	52.578	11.328	0.34109	1284.38
131800-7BE	-7.297	60.438	52.578	11.328	0.34109	1284.38
052500-7WB	-7.297	52.313	59.609	11.328	0.34109	1284.38
051800-7PG	-7.297	52.313	52.578	23.828	0.34109	1284.38
051801-7JY	-7.297	52.313	52.578	11.328	0.37859	1284.38
051800-7YZ	-7.297	52.313	52.578	11.328	0.34109	1290.63
13250104QV	3.953	60.438	59.609	23.828	0.37859	1290.63
15050100FX	0.203	62.313	40.078	23.828	0.41609	1184.38
150501-4UI	-3.547	62.313	40.078	23.828	0.41609	1184.38
03050100BB	0.203	50.438	40.078	23.828	0.41609	1184.38

Continued on next page

Table A.1 – continued from previous page

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
15120100OT	0.203	62.313	47.109	23.828	0.41609	1184.38
15050100ZA	0.203	62.313	40.078	11.328	0.41609	1184.38
15050100OS	0.203	62.313	40.078	23.828	0.45359	1184.38
15050100YO	0.203	62.313	40.078	23.828	0.41609	1190.63
031201-4RO	-3.547	50.438	47.109	11.328	0.45359	1190.63
00240104IW	3.953	47.313	58.828	5.078	0.45359	1134.38
00240100TK	0.203	47.313	58.828	5.078	0.45359	1134.38
08240104TO	3.953	55.438	58.828	5.078	0.45359	1134.38
00060104RV	3.953	47.313	40.859	5.078	0.45359	1134.38
00240104NM	3.953	47.313	58.828	5.078	0.41609	1134.38
00240104XM	3.953	47.313	58.828	5.078	0.45359	1140.63
08060100VX	0.203	55.438	40.859	5.078	0.41609	1140.63
101101-4PZ	-3.547	57.313	46.328	17.578	0.37859	1234.38
101101-7XB	-7.297	57.313	46.328	17.578	0.37859	1234.38
181101-4KP	-3.547	65.438	46.328	17.578	0.37859	1234.38
101801-4EL	-3.547	57.313	53.359	17.578	0.37859	1234.38
101100-4AC	-3.547	57.313	46.328	17.578	0.34109	1234.38
101101-4TC	-3.547	57.313	46.328	17.578	0.37859	1240.63
181800-7WN	-7.297	65.438	53.359	17.578	0.34109	1240.63
030200-2AC	-1.672	49.813	36.953	14.453	0.35984	1259.38
03020002AE	2.078	49.813	36.953	14.453	0.35984	1259.38
060200-2HR	-1.672	52.938	36.953	14.453	0.35984	1259.38
030300-2JI	-1.672	49.813	37.734	14.453	0.35984	1259.38
030200-2OT	-1.672	49.813	36.953	8.203	0.35984	1259.38
030201-2HU	-1.672	49.813	36.953	14.453	0.39734	1259.38
030200-2FA	-1.672	49.813	36.953	14.453	0.35984	1215.63
06030102MV	2.078	52.938	37.734	8.203	0.39734	1215.63
13140106EH	5.828	59.813	49.453	1.953	0.43484	1159.38
131401-5IZ	-5.422	59.813	49.453	1.953	0.43484	1159.38

Continued on next page

Table A.1 – continued from previous page

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
16140106MO	5.828	62.938	49.453	1.953	0.43484	1159.38
13150106ZX	5.828	59.813	50.234	1.953	0.43484	1159.38
13140106DW	5.828	59.813	49.453	20.703	0.43484	1159.38
13140006JW	5.828	59.813	49.453	1.953	0.32234	1159.38
13140106PE	5.828	59.813	49.453	1.953	0.43484	1115.63
161500-5CK	-5.422	62.938	50.234	20.703	0.32234	1115.63
08080102UZ	2.078	54.813	43.203	20.703	0.39734	1109.38
08080106WT	5.828	54.813	43.203	20.703	0.39734	1109.38
11080102JH	2.078	57.938	43.203	20.703	0.39734	1109.38
08210102HD	2.078	54.813	56.484	20.703	0.39734	1109.38
08080102QG	2.078	54.813	43.203	14.453	0.39734	1109.38
08080002EI	2.078	54.813	43.203	20.703	0.35984	1109.38
08080102RL	2.078	54.813	43.203	20.703	0.39734	1165.63
11210006UU	5.828	57.938	56.484	14.453	0.35984	1165.63
182100-5AY	-5.422	64.813	55.703	8.203	0.32234	1209.38
182100-2WL	-1.672	64.813	55.703	8.203	0.32234	1209.38
012100-5LU	-5.422	47.938	55.703	8.203	0.32234	1209.38
180900-5JB	-5.422	64.813	43.984	8.203	0.32234	1209.38
182100-5AD	-5.422	64.813	55.703	1.953	0.32234	1209.38
182101-5AL	-5.422	64.813	55.703	8.203	0.43484	1209.38
182100-5WA	-5.422	64.813	55.703	8.203	0.32234	1265.63
010901-2HR	-1.672	47.938	43.984	1.953	0.43484	1265.63
020701-4XG	-4.484	48.563	41.641	0.391	0.42547	1246.88
02070103WQ	3.016	48.563	41.641	0.391	0.42547	1246.88
070701-4LJ	-4.484	54.188	41.641	0.391	0.42547	1246.88
020101-4TM	-4.484	48.563	36.172	0.391	0.42547	1246.88
020701-4FU	-4.484	48.563	41.641	16.016	0.42547	1246.88
020701-4WF	-4.484	48.563	41.641	0.391	0.44422	1246.88
020701-4CE	-4.484	48.563	41.641	0.391	0.42547	1278.13

Continued on next page

Table A.1 – continued from previous page

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
07010103QE	3.016	54.188	36.172	16.016	0.44422	1278.13
12190003PS	3.016	58.563	54.141	12.891	0.35047	1146.88
121900-4LC	-4.484	58.563	54.141	12.891	0.35047	1146.88
17190003PH	3.016	64.188	54.141	12.891	0.35047	1146.88
12140003QZ	3.016	58.563	48.672	12.891	0.35047	1146.88
12190003EA	3.016	58.563	54.141	3.516	0.35047	1146.88
12190003MW	3.016	58.563	54.141	12.891	0.36922	1146.88
12190003IV	3.016	58.563	54.141	12.891	0.35047	1178.13
171400-4VR	-4.484	64.188	48.672	3.516	0.36922	1178.13
07000107KI	6.766	53.563	35.391	6.641	0.38797	1196.88
12000107PN	6.766	59.188	35.391	6.641	0.38797	1196.88
07200107GU	6.766	53.563	54.922	6.641	0.38797	1196.88
07000107ZH	6.766	53.563	35.391	9.766	0.38797	1196.88
07000007VS	6.766	53.563	35.391	6.641	0.33172	1196.88
07000107SN	6.766	53.563	35.391	6.641	0.38797	1128.13
12200007BH	6.766	59.188	54.922	9.766	0.33172	1128.13
171301-1NO	-0.734	63.563	47.891	19.141	0.46297	1296.88
021301-1FY	-0.734	49.188	47.891	19.141	0.46297	1296.88
170701-1RZ	-0.734	63.563	42.422	19.141	0.46297	1296.88
171301-1RP	-0.734	63.563	47.891	22.266	0.46297	1296.88
171301-1RQ	-0.734	63.563	47.891	19.141	0.40672	1296.88
171301-1DA	-0.734	63.563	47.891	19.141	0.46297	1228.13
020701-1VN	-0.734	49.188	42.422	22.266	0.40672	1228.13
092201-3TJ	-2.609	56.063	57.266	22.266	0.44422	1221.88
09220105HG	4.891	56.063	57.266	22.266	0.44422	1221.88
152201-3AO	-2.609	61.688	57.266	22.266	0.44422	1221.88
092301-3BX	-2.609	56.063	58.047	22.266	0.44422	1221.88
092201-3DS	-2.609	56.063	57.266	0.391	0.44422	1221.88
092200-3JE	-2.609	56.063	57.266	22.266	0.35047	1221.88

Continued on next page

Table A.1 – continued from previous page

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
092201-3MH	-2.609	56.063	57.266	22.266	0.44422	1203.13
15230005RY	4.891	61.688	58.047	0.391	0.35047	1203.13
19100005IJ	4.891	66.063	44.766	9.766	0.36922	1121.88
191000-3RW	-2.609	66.063	44.766	9.766	0.36922	1121.88
05100005IF	4.891	51.688	44.766	9.766	0.36922	1121.88
19110005IQ	4.891	66.063	45.547	9.766	0.36922	1121.88
19100005EY	4.891	66.063	44.766	12.891	0.36922	1121.88
19100105LV	4.891	66.063	44.766	9.766	0.42547	1121.88
19100005IN	4.891	66.063	44.766	9.766	0.36922	1103.13
051101-3JY	-2.609	51.688	45.547	12.891	0.42547	1103.13
04160001EV	1.141	51.063	51.016	16.016	0.33172	1171.88
10160001YK	1.141	56.688	51.016	16.016	0.33172	1171.88
04040001LH	1.141	51.063	39.297	16.016	0.33172	1171.88
04160001XJ	1.141	51.063	51.016	19.141	0.33172	1171.88
04160101YN	1.141	51.063	51.016	16.016	0.46297	1171.88
04160001KE	1.141	51.063	51.016	16.016	0.33172	1153.13
10040101SU	1.141	56.688	39.297	19.141	0.46297	1153.13
140401-6IU	-6.359	61.063	38.516	3.516	0.40672	1271.88
200401-6NM	-6.359	66.688	38.516	3.516	0.40672	1271.88
141701-6VS	-6.359	61.063	51.797	3.516	0.40672	1271.88
140401-6WW	-6.359	61.063	38.516	6.641	0.40672	1271.88
140401-6JN	-6.359	61.063	38.516	3.516	0.38797	1271.88
140401-6BD	-6.359	61.063	38.516	3.516	0.40672	1253.13
201701-6RE	-6.359	66.688	51.797	6.641	0.38797	1253.13
00240104TX	3.953	47.313	58.828	5.078	0.45359	1134.38
101101-4EP	-3.547	57.313	46.328	17.578	0.37859	1234.38
140401-6NF	-6.359	61.063	38.516	3.516	0.40672	1271.88
171301-1EW	-0.734	63.563	47.891	19.141	0.46297	1296.88
04160001TU	1.141	51.063	51.016	16.016	0.33172	1171.88

Continued on next page

Table A.1 – continued from previous page

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
07000107MF	6.766	53.563	35.391	6.641	0.38797	1196.88
091200-7VK	-6.594	56.375	46.719	8.594	0.36219	1293.75
09120000EF	-0.031	56.375	46.719	8.594	0.36219	1293.75
171200-7QF	-6.594	63.875	46.719	8.594	0.36219	1293.75
091200-7OI	-6.594	56.375	46.719	8.594	0.36219	1293.75
091200-7GZ	-6.594	56.375	46.719	21.094	0.36219	1293.75
091200-7KH	-6.594	56.375	46.719	8.594	0.33406	1293.75
091200-7EY	-6.594	56.375	46.719	8.594	0.36219	1181.25
17120000ME	-0.031	63.875	46.719	21.094	0.33406	1181.25
19240101MB	0.906	66.375	59.219	21.094	0.43719	1193.75
192401-8QF	-7.531	66.375	59.219	21.094	0.43719	1193.75
07240101SK	0.906	53.875	59.219	21.094	0.43719	1193.75
19240101TK	0.906	66.375	59.219	21.094	0.43719	1193.75
19240101YI	0.906	66.375	59.219	8.594	0.43719	1193.75
19240101XC	0.906	66.375	59.219	21.094	0.40906	1193.75
19240101DG	0.906	66.375	59.219	21.094	0.43719	1281.25
072401-8UA	-7.531	53.875	59.219	8.594	0.40906	1281.25
04050105IF	4.656	51.375	40.469	2.344	0.39969	1143.75
04050104PF	3.719	51.375	40.469	2.344	0.39969	1143.75
02050105BD	4.656	48.875	40.469	2.344	0.39969	1143.75
04180105MK	4.656	51.375	52.969	2.344	0.39969	1143.75
04050105CA	4.656	51.375	40.469	2.344	0.39969	1143.75
04050105EP	4.656	51.375	40.469	2.344	0.44656	1143.75
04050105NF	4.656	51.375	40.469	2.344	0.39969	1231.25
02180104PM	3.719	48.875	52.969	2.344	0.44656	1231.25
141800-3QO	-2.844	61.375	52.969	14.844	0.32469	1243.75
141800-4CI	-3.781	61.375	52.969	14.844	0.32469	1243.75
121800-3MJ	-2.844	58.875	52.969	14.844	0.32469	1243.75
140500-3DA	-2.844	61.375	40.469	14.844	0.32469	1243.75

Continued on next page

Table A.1 – continued from previous page

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
141800-3ZI	-2.844	61.375	52.969	14.844	0.32469	1243.75
141801-3YZ	-2.844	61.375	52.969	14.844	0.37156	1243.75
141800-3KV	-2.844	61.375	52.969	14.844	0.32469	1131.25
120501-4KR	-3.781	58.875	40.469	14.844	0.37156	1131.25
022100-1NK	-0.969	48.875	56.094	17.969	0.34344	1268.75
02210006DX	5.594	48.875	56.094	17.969	0.34344	1268.75
042100-1EU	-0.969	51.375	56.094	17.969	0.34344	1268.75
021500-1AR	-0.969	48.875	49.844	17.969	0.34344	1268.75
022100-1AO	-0.969	48.875	56.094	11.719	0.34344	1268.75
022101-1JU	-0.969	48.875	56.094	17.969	0.46531	1268.75
022100-1NM	-0.969	48.875	56.094	17.969	0.34344	1106.25
04150106XY	5.594	51.375	49.844	11.719	0.46531	1106.25
12090107ZH	6.531	58.875	43.594	5.469	0.41844	1168.75
120901-2UI	-1.906	58.875	43.594	5.469	0.41844	1168.75
14090107FS	6.531	61.375	43.594	5.469	0.41844	1168.75
12020107SR	6.531	58.875	37.344	5.469	0.41844	1168.75
12090107DW	6.531	58.875	43.594	24.219	0.41844	1168.75
12090107ZJ	6.531	58.875	43.594	5.469	0.39031	1168.75
12090107SM	6.531	58.875	43.594	5.469	0.41844	1206.25
140201-2TH	-1.906	61.375	37.344	24.219	0.39031	1206.25
07150103FI	2.781	53.875	49.844	24.219	0.45594	1118.75
07150102QB	1.844	53.875	49.844	24.219	0.45594	1118.75
19150103DC	2.781	66.375	49.844	24.219	0.45594	1118.75
07090103CS	2.781	53.875	43.594	24.219	0.45594	1118.75
07150103DE	2.781	53.875	49.844	17.969	0.45594	1118.75
07150003WR	2.781	53.875	49.844	24.219	0.35281	1118.75
07150103LA	2.781	53.875	49.844	24.219	0.45594	1256.25
19090002LE	1.844	66.375	43.594	17.969	0.35281	1256.25
170201-5ZY	-4.719	63.875	37.344	11.719	0.38094	1218.75

Continued on next page

Table A.1 – continued from previous page

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
170201-6PI	-5.656	63.875	37.344	11.719	0.38094	1218.75
090201-5RA	-4.719	56.375	37.344	11.719	0.38094	1218.75
172101-5RM	-4.719	63.875	56.094	11.719	0.38094	1218.75
170201-5JQ	-4.719	63.875	37.344	5.469	0.38094	1218.75
170201-5ZQ	-4.719	63.875	37.344	11.719	0.42781	1218.75
170201-5AP	-4.719	63.875	37.344	11.719	0.38094	1156.25
092101-6SQ	-5.656	56.375	56.094	5.469	0.42781	1156.25
031301-6JA	-5.656	50.125	48.281	3.906	0.44656	1231.25
03130107LR	6.531	50.125	48.281	3.906	0.44656	1231.25
031301-6NN	-5.656	50.125	48.281	3.906	0.44656	1231.25
031301-6JK	-5.656	50.125	48.281	3.906	0.44656	1231.25
031301-6HP	-5.656	50.125	48.281	13.281	0.44656	1231.25
031301-6KQ	-5.656	50.125	48.281	3.906	0.41844	1231.25
031301-6CG	-5.656	50.125	48.281	3.906	0.44656	1193.75
03130107HX	6.531	50.125	48.281	13.281	0.41844	1193.75
13010102DH	1.844	60.125	35.781	16.406	0.37156	1131.25
130101-1BW	-0.969	60.125	35.781	16.406	0.37156	1131.25
13010102LQ	1.844	60.125	35.781	16.406	0.37156	1131.25
13010102PI	1.844	60.125	35.781	16.406	0.37156	1131.25
13010102QO	1.844	60.125	35.781	0.781	0.37156	1131.25
13010002ZL	1.844	60.125	35.781	16.406	0.34344	1131.25
13010102GV	1.844	60.125	35.781	16.406	0.37156	1293.75
130100-1XX	-0.969	60.125	35.781	0.781	0.34344	1293.75
08200006EU	5.594	55.125	54.531	10.156	0.33406	1181.25
08200003TU	2.781	55.125	54.531	10.156	0.33406	1181.25
18200006HW	5.594	65.125	54.531	10.156	0.33406	1181.25
08070006SZ	5.594	55.125	42.031	10.156	0.33406	1181.25
08200006RT	5.594	55.125	54.531	7.031	0.33406	1181.25
08200106EM	5.594	55.125	54.531	10.156	0.38094	1181.25

Continued on next page

Table A.1 – continued from previous page

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
08200006VE	5.594	55.125	54.531	10.156	0.33406	1243.75
18070103QT	2.781	65.125	42.031	7.031	0.38094	1243.75
180701-2QX	-1.906	65.125	42.031	22.656	0.40906	1281.25
180701-5JD	-4.719	65.125	42.031	22.656	0.40906	1281.25
080701-2UX	-1.906	55.125	42.031	22.656	0.40906	1281.25
182001-2KP	-1.906	65.125	54.531	22.656	0.40906	1281.25
180701-2VR	-1.906	65.125	42.031	19.531	0.40906	1281.25
180701-2TN	-1.906	65.125	42.031	22.656	0.45594	1281.25
180701-2TF	-1.906	65.125	42.031	22.656	0.40906	1143.75
082001-5RE	-4.719	55.125	54.531	19.531	0.45594	1143.75
060401-4QF	-3.781	52.625	38.906	19.531	0.42781	1206.25
06040101XP	0.906	52.625	38.906	19.531	0.42781	1206.25
160401-4FS	-3.781	62.625	38.906	19.531	0.42781	1206.25
061001-4CC	-3.781	52.625	45.156	19.531	0.42781	1206.25
060401-4XG	-3.781	52.625	38.906	3.906	0.42781	1206.25
060400-4AU	-3.781	52.625	38.906	19.531	0.36219	1206.25
060401-4IG	-3.781	52.625	38.906	19.531	0.42781	1118.75
16100001SY	0.906	62.625	45.156	3.906	0.36219	1118.75
16160004NQ	3.719	62.625	51.406	7.031	0.35281	1106.25
161600-7DW	-6.594	62.625	51.406	7.031	0.35281	1106.25
06160004BC	3.719	52.625	51.406	7.031	0.35281	1106.25
16230004UY	3.719	62.625	57.656	7.031	0.35281	1106.25
16160004OZ	3.719	62.625	51.406	16.406	0.35281	1106.25
16160104TO	3.719	62.625	51.406	7.031	0.43719	1106.25
16160004RA	3.719	62.625	51.406	7.031	0.35281	1218.75
062301-7KB	-6.594	52.625	57.656	16.406	0.43719	1218.75
01100100FQ	-0.031	47.625	45.156	13.281	0.39031	1156.25
01100105FM	4.656	47.625	45.156	13.281	0.39031	1156.25
01100100WS	-0.031	47.625	45.156	13.281	0.39031	1156.25

Continued on next page

Table A.1 – continued from previous page

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
01160100CV	-0.031	47.625	51.406	13.281	0.39031	1156.25
01100100CX	-0.031	47.625	45.156	22.656	0.39031	1156.25
01100100VN	-0.031	47.625	45.156	13.281	0.39969	1156.25
01100100BM	-0.031	47.625	45.156	13.281	0.39031	1268.75
01160105OJ	4.656	47.625	51.406	22.656	0.39969	1268.75
112301-8AC	-7.531	57.625	57.656	0.781	0.46531	1256.25
112301-3WH	-2.844	57.625	57.656	0.781	0.46531	1256.25
112301-8LP	-7.531	57.625	57.656	0.781	0.46531	1256.25
110401-8US	-7.531	57.625	38.906	0.781	0.46531	1256.25
112301-8WC	-7.531	57.625	57.656	10.156	0.46531	1256.25
112300-8YP	-7.531	57.625	57.656	0.781	0.32469	1256.25
112301-8VM	-7.531	57.625	57.656	0.781	0.46531	1168.75
110400-3DZ	-2.844	57.625	38.906	10.156	0.32469	1168.75

The following 16 geometries are created in addition to the 256 geometries using Latin Hypercube Sampling to serve as the hold-out test set.

Table A.2: Design parameter table for the hold-out test set in the `double-delta-aero` dataset.

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
12100003VH	3.058	59.326	45.284	17.249	0.33405	1110.78
02120002AN	2.293	48.531	47.236	24.560	0.36826	1284.99
171901-1NW	-1.208	64.273	54.330	14.406	0.39977	1206.77
191301-3JT	-2.567	66.406	48.424	20.081	0.38065	1122.63
18040004MO	4.092	65.174	38.789	11.433	0.36160	1245.04
070801-2FS	-1.735	53.930	42.804	13.094	0.44046	1299.67
060901-7YD	-7.290	52.605	43.893	17.174	0.46215	1199.12
050101-7LK	-6.845	51.980	35.542	21.451	0.39294	1269.96

Continued on next page

design-call-sign	$\delta[^\circ]$	$\Lambda_{in}[^\circ]$	$\Lambda_{out}[^\circ]$	$\Lambda_{te}[^\circ]$	BW2	B [in.]
13240101PQ	1.357	60.169	59.281	6.586	0.42903	1186.44
09190107DL	6.718	55.850	53.634	9.090	0.45063	1258.95
150201-4SN	-4.233	62.449	37.195	22.482	0.45858	1173.30
11050006QW	5.646	58.130	39.920	0.777	0.34876	1146.57
031600-5NR	-4.685	49.706	50.825	2.658	0.32025	1222.62
012100-6CB	-6.008	47.949	56.370	6.145	0.33920	1156.79
09230100EA	-0.207	55.726	57.604	3.265	0.40527	1135.19
14150105TT	4.780	61.454	50.410	10.170	0.41554	1230.50

The base airfoil used across all configurations, subject to camber deflection as discussed in Section 4.1, is obtained from the wing-root airfoil of the RSAC concept designed by Livne et al. and presented in [69].

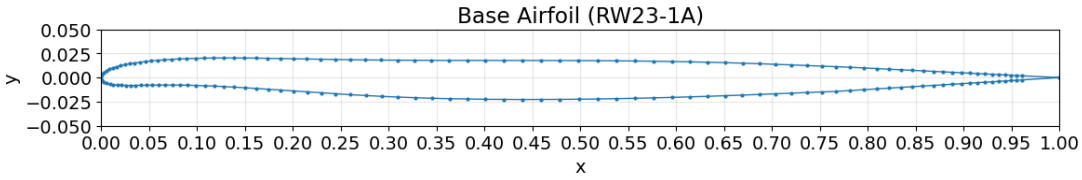


Figure A.1: Base airfoil used, obtained from [69].

The association between geometries and the corresponding dataset levels, as discussed in Chapter 4.2, is presented in the following table. The Level 6 dataset, which contains 256 geometries, includes all geometries presented in Table A.1. The remaining subsets, grouped by their sampling levels, are presented in Table A.3.

Table A.3: Association between geometries and the corresponding dataset levels.

Index	Level 1	Level 2	Level 3	Level 4	Level 5
1	091200-7VK	091200-7VK	091200-7VK	091200-7VK	091200-7VK
2	09120000EF	09120000EF	09120000EF	09120000EF	09120000EF
3	171200-7QF	171200-7QF	171200-7QF	171200-7QF	171200-7QF

Continued on next page

Index	Level 1	Level 2	Level 3	Level 4	Level 5
4	091200-7VK	091200-7VK	091200-7OI	091200-7OI	091200-7OI
5	091200-7GZ	091200-7GZ	091200-7GZ	091200-7GZ	091200-7GZ
6	091200-7KH	091200-7KH	091200-7KH	091200-7KH	091200-7KH
7	091200-7EY	091200-7EY	091200-7EY	091200-7EY	091200-7EY
8	17120000ME	17120000ME	17120000ME	17120000ME	17120000ME
9		19240101MB	19240101MB	19240101MB	19240101MB
10		192401-8QF	192401-8QF	192401-8QF	192401-8QF
11		07240101SK	07240101SK	07240101SK	07240101SK
12		19240101MB	19240101TK	19240101TK	19240101TK
13		19240101YI	19240101YI	19240101YI	19240101YI
14		19240101XC	19240101XC	19240101XC	19240101XC
15		19240101DG	19240101DG	19240101DG	19240101DG
16		072401-8UA	072401-8UA	072401-8UA	072401-8UA
17			04050105IF	04050105IF	04050105IF
18			04050104PF	04050104PF	04050104PF
19			02050105BD	02050105BD	02050105BD
20			04180105MK	04180105MK	04180105MK
21			04050105CA	04050105CA	04050105CA
22			04050105EP	04050105EP	04050105EP
23			04050105NF	04050105NF	04050105NF
24			02180104PM	02180104PM	02180104PM
25			141800-3QO	141800-3QO	141800-3QO
26			141800-4CI	141800-4CI	141800-4CI
27			121800-3MJ	121800-3MJ	121800-3MJ
28			140500-3DA	140500-3DA	140500-3DA
29			141800-3ZI	141800-3ZI	141800-3ZI
30			141801-3YZ	141801-3YZ	141801-3YZ
31			141800-3KV	141800-3KV	141800-3KV
32			120501-4KR	120501-4KR	120501-4KR
33				022100-1NK	022100-1NK

Continued on next page

Index	Level 1	Level 2	Level 3	Level 4	Level 5
34				02210006DX	02210006DX
35				042100-1EU	042100-1EU
36				021500-1AR	021500-1AR
37				022100-1AO	022100-1AO
38				022101-1JU	022101-1JU
39				022100-1NM	022100-1NM
40				04150106XY	04150106XY
41				12090107ZH	12090107ZH
42				120901-2UI	120901-2UI
43				14090107FS	14090107FS
44				12020107SR	12020107SR
45				12090107DW	12090107DW
46				12090107ZJ	12090107ZJ
47				12090107SM	12090107SM
48				140201-2TH	140201-2TH
49				07150103FI	07150103FI
50				07150102QB	07150102QB
51				19150103DC	19150103DC
52				07090103CS	07090103CS
53				07150103DE	07150103DE
54				07150003WR	07150003WR
55				07150103LA	07150103LA
56				19090002LE	19090002LE
57				170201-5ZY	170201-5ZY
58				170201-6PI	170201-6PI
59				090201-5RA	090201-5RA
60				172101-5RM	172101-5RM
61				170201-5JQ	170201-5JQ
62				170201-5ZQ	170201-5ZQ
63				170201-5AP	170201-5AP

Continued on next page

Index	Level 1	Level 2	Level 3	Level 4	Level 5
64				092101-6SQ	092101-6SQ
65					031301-6JA
66					03130107LR
67					031301-6NN
68					031301-6JK
69					031301-6HP
70					031301-6KQ
71					031301-6CG
72					03130107HX
73					13010102DH
74					130101-1BW
75					13010102LQ
76					13010102PI
77					13010102QO
78					13010002ZL
79					13010102GV
80					130100-1XX
81					08200006EU
82					08200003TU
83					18200006HW
84					08070006SZ
85					08200006RT
86					08200106EM
87					08200006VE
88					18070103QT
89					180701-2QX
90					180701-5JD
91					080701-2UX
92					182001-2KP
93					180701-2VR

Continued on next page

Index	Level 1	Level 2	Level 3	Level 4	Level 5
94					180701-2TN
95					180701-2TF
96					082001-5RE
97					060401-4QF
98					06040101XP
99					160401-4FS
100					061001-4CC
101					060401-4XG
102					060400-4AU
103					060401-4IG
104					16100001SY
105					16160004NQ
106					161600-7DW
107					06160004BC
108					16230004UY
109					16160004OZ
110					16160104TO
111					16160004RA
112					062301-7KB
113					01100100FQ
114					01100105FM
115					01100100WS
116					01160100CV
117					01100100CX
118					01100100VN
119					01100100BM
120					01160105OJ
121					112301-8AC
122					112301-3WH
123					112301-8LP

Continued on next page

Index	Level 1	Level 2	Level 3	Level 4	Level 5
124					110401-8US
125					112301-8WC
126					112300-8YP
127					112301-8VM
128					110400-3DZ

Bibliography

[1] Jennifer Abras, Shirzad Hosseinverdi, Dylan P. Jude, Jay Sitaraman, and Nathan S. Hariharan. Machine learning based surrogate model development using GPU driven computational simulations. In *AIAA SCITECH 2025 Forum*, 2025. doi: 10.2514/6.2025-0040.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. URL https://doi.org/10.1145/3292500.3330701.

[3] Benedikt Alkin, Maurits Bleeker, Richard Kurle, Tobias Kronlachner, Reinhard Sonnleitner, Matthias Dorfer, and Johannes Brandstetter. AB-UPT: Scaling neural cfd surrogates for high-fidelity automotive aerodynamics simulations via anchored-branched universal physics transformers. *arXiv preprint*, 2025. URL https://arxiv.org/abs/2502.09692.

[4] Apurva Anand, Muhammad Muneeb Safdar, Koushik Marepally, and James D. Baeder. A comprehensive review of neural network training approaches for airfoil design and optimization. In *AIAA SCITECH 2025 Forum*, 2025. doi: 10.2514/6.2025-0271. URL https://arc.aiaa.org/doi/abs/10.2514/6.2025-0271.

[5] Davide Anguita, Luca Ghelardoni, Alessandro Ghio, Luca Oneto, Sandro Ridella, et al. The 'K' in K-fold cross validation. In *ESANN*, volume 102, pages 441–446, 2012. URL https://www.esann.org/sites/default/files/proceedings/legacy/es2012-62.pdf.

[6] Phuriwat Anusonti-Inthra and Richard C. Healy. Development of multi-fidelity surrogate models for rotor performance prediction. In *AIAA SCITECH 2025 Forum*, 2025. doi: 10.2514/6.2025-0490.

[7] Neil Ashton, Johannes Brandstetter, and Siddhartha Mishra. Fluid intelligence: A forward look on AI foundation models in computational fluid dynamics. *arXiv preprint*, 2025. URL `https://arxiv.org/abs/2511.20455`.

[8] Bo-Young Bae, Sangho Kim, Jae-Woo Lee, Nhu Van Nguyen, and Bong-Cheul Chung. Process of establishing design requirements and selecting alternative configurations for conceptual design of a VLA. *Chinese Journal of Aeronautics*, 30(2):738–751, 2017. ISSN 1000-9361. doi: https://doi.org/10.1016/j.cja.2017.02.018.

[9] William Bastedo. Commerical hypersonic transportation market study. NASA Contractor Report 20210014711, NASA Gleen Research Center, 2021.

[10] Gal Berkooz, Philip Holmes, and John L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25: 539–575, 1993. doi: https://doi.org/10.1146/ANNUREV.FL.25.010193.002543.

[11] John J Bertin and Russell M Cummings. *Aerodynamics for engineers*, pages 354–362. Cambridge University Press, Cambridge, England, 6 edition, August 2021. ISBN 978-1009098625.

[12] Nolan Black and Ahmad R. Najafi. Learning finite element convergence with the Multi-fidelity Graph Neural Network. *Computer Methods in Applied Mechanics and Engineering*, 397:115120, 2022. ISSN 00457825. doi: 10.1016/j.cma.2022.115120.

[13] W.B. Blake. USAF Stability and Control Digital Datcom. Technical report, McDonnel Douglas Auronautics Division, 1979. AFFDL-TR-79-3032.

[14] Benjamin S. Blanchard and Wolter J. Fabrycky. *Systems Engineering and Analysis, 5th edition*, chapter Ch.2. Pearson Education Limited, 2023. ISBN 978-0137980888.

[15] Emilio M. Botero, Matthew A. Clarke, Racheal M. Erhard, Jordan T. Smart, Juan J. Alonso, and Aaron Blaufox. Aerodynamic verification and validation of SUAVE. In *AIAA SCITECH 2022 Forum*, 2022. doi: 10.2514/6.2022-1929.

[16] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations (ICLR)*, 2022. URL `https://openreview.net/forum?id=F72ximsx7C1`.

[17] Kinga Budziak. Aerodynamic analysis with Athena Vortex Lattice (AVL). Technical report, Hamburg University of Applied Sciences, 2015. URL `https://www.fzt.haw-hamburg.de/pers/Scholz/arbeiten/TextBudziak.pdf`.

[18] Francesca Campolongo, Andrea Saltelli, and Jessica Cariboni. From screening to quantitative sensitivity analysis. a unified approach. *Computer Physics Communications*, 182(4):978–988, 2011.

[19] Sebastien Candel. Concorde and the future of supersonic transport. *Journal of propulsion and power*, 20(1):59–68, 2004.

[20] Giovanni Catalani, Siddhant Agarwal, Xavier Bertrand, Frédéric Tost, Michael Bauerheim, and Joseph Morlier. Neural fields for rapid aircraft aerodynamics simulations. *Scientific Reports*, 14(1):25496, Oct 2024. ISSN 2045-2322. doi: 10.1038/s41598-024-76983-w.

[21] Ching-Min Chang, Hsiang-Yu Tsai, Te-Yao Chiu, Che-Jung Cheng, Wei-Ann Lin, Ray-Bing Chen, and Yi-Ju Chou. Proper-orthogonal-decomposition–based emulation of spatiotemporal evolution of turbulent wakes. *AIAA Journal*, 63(9):3582–3594, 2025. doi: 10.2514/1.J064779.

[22] Smith W. Charles and Bhateley C. Ishwar. Application of the vortex-lattice technique to the analysis of thin wings with vortex separation and thick multi-element wings. In *Vortex-Lattice Utilization, a workshop held at Langley Research Center*, 1976. URL `https://ntrs.nasa.gov/api/citations/19760021075/downloads/19760021075.pdf`.

[23] J. Chen, E. Hachem, and J. Viquerat. Graph neural networks for laminar flow prediction around random two-dimensional shapes. *Physics of Fluids*, 33(12), 2021. ISSN 10897666. doi: 10.1063/5.0064108.

[24] C. Croke, Thomas. *Design of Aircraft*, chapter Ch.1.2 Design Process, pages 15–18. Prentice Hall, Upper Saddle River, NJ, 2003. ISBN 0130892343.

[25] Jennifer Dacles-Mariani, Gregory G. Zilliac, Jim S. Chow, and Peter Bradshaw. Numerical/experimental study of a wingtip vortex in the near field. *AIAA Journal*, 33 (9):1561–1568, 1995. doi: 10.2514/3.12826.

[26] Kenneth Decker, Henry D. Schwartz, and Dimitri Mavris. Dimensionality reduction techniques applied to the design of hypersonic aerial systems. In *AIAA AVIATION 2020 FORUM*, 2020. doi: 10.2514/6.2020-3003.

[27] Kenneth Decker, Nikhil Iyengar, Dushhyanth Rajaram, Christian Perron, and Dimitri Mavris. Manifold alignment-based nonintrusive and nonlinear multifidelity reduced-order modeling. *AIAA Journal*, 61(1):454–474, 2023. doi: 10.2514/1.J061720.

[28] Saurabh Deshpande, Stéphane P.A. Bordas, and Jakub Lengiewicz. MAgNET: A graph U-Net architecture for mesh-based simulations. *Engineering Applications of Artificial Intelligence*, 133:108055, July 2024. ISSN 0952-1976. doi: 10.1016/j.engappai.2024.108055.

[29] Mark Drela. *Flight Vehicle Aerodynamics*, chapter Ch.6 Aerodynamics of Aircraft in Maneuver, pages 127–135. The MIT Press, Cambridge, MA, 2014. ISBN 9780262526449.

[30] Thomas D. Economon, Francisco Palacios, Sean R. Copeland, Trent W. Lukaczyk, and Juan J. Alonso. SU2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, 54(3):828–846, 2016. doi: 10.2514/1.J053813.

[31] Hamidreza Eivazi, Mojtaba Tahani, Philipp Schlatter, and Ricardo Vinuesa. Physics-informed neural networks for solving reynolds-averaged navier–stokes equations. *Physics of Fluids*, 34(7):075117, 07 2022. ISSN 1070-6631. doi: 10.1063/5.0095270.

[32] M. Giselle Fernández-Godino. Review of multi-fidelity models. *Advances in Computational Science and Engineering*, 1(4):351–400, 2023. doi: 10.3934/acse.2023015.

[33] Hongyang Gao and Shuiwang Ji. Graph U-Nets. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning,ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 2083–2092. PMLR, 2019. URL `http://proceedings.mlr.press/v97/gao19a.html`.

[34] Limin Gao, Chi Ma, and Yutong Cai. A robust blade design method based on non-intrusive polynomial chaos considering profile error. *Journal of Thermal Science*, 28 (5):875–885, 2019.

[35] Philip E Gill, Walter Murray, and Michael A Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[36] A. A. Giunta, V. Balabanov, D. Haim, B. Grossman, W. H. Mason, L. T. Watson, and R. T. Haftka. Multidisciplinary optimisation of a supersonic transport using design of experiments theory and response surface modelling. *The Aeronautical Journal*, 101 (1008):347–356, 1997. doi: 10.1017/S0001924000066045.

[37] P. K. Green, M. Pacull, and H. D. Reimers. European 2nd generation supersonic commercial transport aircraft. In *Proceedings of the 20th International Congress of the Aeronautical Sciences (ICAS)*, Sorrento, Italy, 1996. URL `https://www.icas.org/icas\_archive/ICAS1996/ICAS-96-4.4.1.pdf`. ICAS-96-4.

[38] Andrew Grenville and Leonidas J. Guibas. Spline curves. Technical report, Stanford CS348 Course Reader, 1992. URL `https://graphics.stanford.edu/courses/cs348a-08-winter/`.

[39] Thiago A.M. Guimarães, Carlos E.S. Cesnik, and Ilya Kolmanovsky. Low Speed Aerodynamic Modeling for Control-related Considerations in Supersonic Aircraft Design. *AIAA Aviation and Aeronautics Forum and Exposition, AIAA AVIATION Forum 2021*, pages 1–18, 2021. doi: 10.2514/6.2021-2531.

[40] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint*, 2018. URL `https://arxiv.org/abs/1709.05584`.

[41] Richard C. Healy, Phuriwat Anusonti-Inthra, and Nathan Beals. Neural network modeling of airfoil aerodynamics for rotorcraft analysis. In *AIAA SCITECH 2025 Forum*, 2025. doi: 10.2514/6.2025-0270. URL `https://arc.aiaa.org/doi/abs/10.2514/6.2025-0270`.

[42] Jon Herman and Will Usher. SALib: An open-source python library for sensitivity analysis. *The Journal of Open Source Software*, 2(9), jan 2017. doi: 10.21105/joss.00097. URL `https://doi.org/10.21105/joss.00097`.

[43] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint*, 2017. URL `https://arxiv.org/abs/1712.00409`.

[44] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint*, 2022. URL `https://arxiv.org/abs/2203.15556`.

[45] Daniel Huynh, Davide Di Pasquale, Simon Prince, and Vivek Ahuja. Application of a semi-empirical method to model subsonic vortex lift over sharp leading-edge delta wings. In *AIAA SCITECH 2023 Forum*, 2023. doi: 10.2514/6.2023-2457.

[46] Gabriele Immordino, Andrea Vaiuso, Andrea Da Ronch, and Marcello Righi. Predicting transonic flowfields in non–homogeneous unstructured grids using autoencoder graph convolutional networks. *Journal of Computational Physics*, 524:113708, 2025. doi: 10.1016/j.jcp.2024.113708.

[47] A. Jameson, Wolfgang Schmidt, and Eli Turkel. Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes. In *14th Fluid and Plasma Dynamics Conference*, 1981. doi: 10.2514/6.1981-1259.

[48] Stephen Joe and Frances Y. Kuo. Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654, 2008. doi: 10.1137/070709359.

[49] Hemant Joshi and Peter Thomas. Review of vortex lattice method for supersonic aircraft design. *The Aeronautical Journal*, pages 1–35, 2023. doi: 10.1017/aer.2023.25.

[50] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint*, 2020. URL `https://arxiv.org/abs/2001.08361`.

[51] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995. doi: 10.1109/ICNN.1995.488968.

[52] Marc C. Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3): 425–464, 2001. doi: https://doi.org/10.1111/1467-9868.00294.

[53] Parisa Khodabakhshi, Karen E. Willcox, and Max Gunzburger. A multifidelity method for a nonlocal diffusion model. *Applied Mathematics Letters*, 121:107361, 2021. ISSN 0893-9659. doi: https://doi.org/10.1016/j.aml.2021.107361.

[54] Parisa Khodabakhshi, Olena Burkovska, Karen Willcox, and Max Gunzburger. Multifidelity methods for uncertainty quantification of a nonlocal model for phase changes in materials. *Computers & Structures*, 297:107328, 2024. ISSN 0045-7949. doi: https://doi.org/10.1016/j.compstruc.2024.107328.

[55] Dilan Kilic and Melike Nikbay. A comparative study on multi-fidelity machine learning modeling for aerospace problems. In *AIAA SCITECH 2025 Forum*, 2025. doi: 10.2514/6.2025-0272. URL `https://arc.aiaa.org/doi/abs/10.2514/6.2025-0272`.

[56] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015 Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

[57] Mykel J. Kochenderfer and Tim A. Wheller. *Algorithms for Optimization, Second Edition*. The MIT Press, 2025.

[58] Jiaqing Kou, Chenjia Ning, and Weiwei Zhang. Transfer learning for reduced-order modeling of transonic flows using multifidelity data. In *ICAS 2024*, 2024. URL `https://www.icas.org/icas\_archive/icas2024/data/papers/icas2024\_1110\_paper.pdf`.

[59] Ilan Kroo. *Aircraft Design: Synthesis and Analysis*, chapter Ch.6 Wing Design. Desktop Aeronautics, 2001.

[60] Howon Lee, Pranay Seshadri, and Juergen Rauleder. Finite wing pressure and lift coefficient prediction using bayesian inference on pressure data. In *AIAA SCITECH 2025 Forum*, 2025. doi: 10.2514/6.2025-0493. URL https://arc.aiaa.org/doi/abs/10.2514/6.2025-0493.

[61] Chaojian Li, Zhifan Ye, Massimiliano Lupo Pasini, Jong Youl Choi, Cheng Wan, and Prasanna Balaprakash. Scaling laws of graph neural networks for atomistic materials modeling. In *Proceedings of the 62nd Annual ACM/IEEE Design Automation Conference*, DAC '25. IEEE Press, 2025. ISBN 9798331503048. doi: 10.1109/DAC63849.2025.11132864.

[62] Jinxing Li, Yunzhu Li, Tianyuan Liu, Di Zhang, and Yonghui Xie. Multi-fidelity graph neural network for flow field data fusion of turbomachinery. *Energy*, 285(July): 129405, 2023. ISSN 03605442. doi: 10.1016/j.energy.2023.129405.

[63] Wu Li and Karl Geiselhart. Mtidisciplinary design optimization of low-boom supersonic aircraft with mission constraints. *AIAA Journal*, 59:1–15, 12 2020. doi: 10.2514/1.J059237.

[64] 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*, 2021. URL https://arxiv.org/abs/2010.08895.

[65] 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. *arXiv preprint*, 2023. URL https://arxiv.org/abs/2309.00583.

[66] Meng-Sing Liou and Christopher J. Steffen. A new flux splitting scheme. *Journal of Computational Physics*, 107(1):23–39, 1993. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1993.1122.

[67] Jingzhe Liu, Haitao Mao, Zhikai Chen, Tong Zhao, Neil Shah, and Jiliang Tang. Towards neural scaling laws on graphs. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2402.02054.

[68] Qiang Liu, Wei Zhu, Xiyu Jia, Feng Ma, and Yu Gao. Fluid simulation system based on graph neural network. *arXiv preprint*, 2022.

[69] Eli Livne, Chester P. Nelson, Reza M. Soltani, Kuang-Ying Ting, Nicolas Mavriplis, Kenneth Wiersema, Carlos E. S. Cesnik, Ilya Kolmanovsky, Joaquim R. R. A. Martins, Thiago A. M. Guimaraes, Torbjørn Cunis, Guilherme C. Barbosa, Rafael Bertolin, Sabet Seraj, Bernardo B. Monteiro, Juan J. Alonso, Ilan Kroo, Tejal K. Shanbhag, Yiren Shen, Todd E. Magee, Patrick J. Hayes, Andrew J. Dorgan, and Abdollah Khodadoust. Compilation of supersonic configurations at low speeds (SCALOS) final report and appendices A – F. Contractor or Grantee Report 20250006613, NASA - Langley Research Center, July 2025. URL `https://ntrs.nasa.gov/citations/20250006613`.

[70] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5.

[71] Trent Lukaczyk, Francisco Palacios, and Juan Alonso. Response surface methodologies for low-boom supersonic aircraft design using equivalent area distributions. In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2012. doi: 10.2514/6.2012-5705. URL `https://arc.aiaa.org/doi/abs/10.2514/6.2012-5705`.

[72] Trent W Lukaczyk, Andrew D Wendorff, Michael Colonno, Thomas D Economon, Juan J Alonso, Tarik H Orra, and Carlos Ilario. SUAVE: an open-source environment for multi-fidelity conceptual vehicle design. In *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 3087, 2015.

[73] Zhoujie Lyu, Zelu Xu, and Joaquim R. R. A. Martins. Benchmarking optimization algorithms for wing aerodynamic design optimization. In *Proceedings of the 8th International Conference on Computational Fluid Dynamics*, Chengdu, Sichuan, China, July 2014. ICCFD8-2014-0203.

[74] Timothy MacDonald, Matthew Clarke, Emilio M. Botero, Julius M. Vegh, and Juan J.

Alonso. SUAVE: An open-source environment enabling multi-fidelity vehicle optimization. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017. doi: 10.2514/6.2017-4437.

[75] J MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*, 1967. URL `https://zbmath.org/?q=an:0214.46201`.

[76] Charles A. Mader, Gaetan K. W. Kenway, Anil Yildirim, and Joaquim R. R. A. Martins. ADflow—an open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization. *Journal of Aerospace Information Systems*, 2020. doi: 10.2514/1.I010796.

[77] Alexandre N. Marques, Max M. J. Opgenoord, Remi R. Lam, Anirban Chaudhuri, and Karen E. Willcox. Multifidelity method for locating aeroelastic flutter boundaries. *AIAA Journal*, 58(4):1772–1784, 2020. doi: 10.2514/1.J058663.

[78] Joaquim R. R. A. Martins and Andrew B. Lambe. Multidisciplinary design optimization: A survey of architectures. *AIAA Journal*, 51(9):2049–2075, 2013. doi: 10.2514/1.J051895.

[79] Joaquim R.R.A. Martins and Andrew Ning. *Engineering Design Optimization*. Cambridge University Press, 2022.

[80] William H. Mason and Pradeep Raj. *Lecture Notes on Configuration Aerodynamics*, chapter Ch.9 Supersonic Aerodynamics. Kevin T. Crofton Department of Aerospace and Ocean Engineering in association with Virginia Tech Publishing, 2026. doi: 10.21061/configurationaerodynamics.

[81] Nicolas Mavriplis, Kuang-Ying Ting, Reza M. Soltani, Chester P. Nelson, and Eli Livne. Supersonic Configurations at Low Speeds (SCALOS): CFD Aided Wind Tunnel Data Corrections. In *AIAA Scitech 2024 Forum*, number January, pages 1–28, 2023. doi: 10.2514/6.2023-0231.

[82] David S. Miller and Richard M. Wood. Leeside flows over delta wings at supersonic speeds. *Journal of Aircraft*, 21(9):680–686, 1984. doi: 10.2514/3.45014.

[83] Luis R Miranda. Extended applications of the vortex lattice method. *NASA. Langley Res. Center Vortex-Lattice Utilization*, 1976.

[84] Luis R. Miranda, Robert D. Elliott, and William M. Baker. A generalized vortex lattice method for subsonic and supersonic flow applications. Technical report, NASA. Langley Res. Center, 1977. URL `https://ntrs.nasa.gov/citations/19780008059`.

[85] Anthony M Mitchell, Scott A Morton, James R Forsythe, and Russell M Cummings. Analysis of delta-wing vortical substructures using detached-eddy simulation. *AIAA journal*, 44(5):964–972, 2006. doi: https://doi.org/10.2514/1.755.

[86] John Morgenstern, Michael Buonanno, Jixian Yao, Mugam Murugappan, Umesh Paliath, Lawrence Cheung, Ivan Malcevic, Kishore Ramakrishnan, Nikolai Pastouchenko, Trevor Wood, Steve Martens, Phil Viars, Trevor Tersmette, Jason Lee, Ron Simmons, David Plybon, Juan Alonso, Francisco Palacios, Trent Lukaczyk, and Gerald Carrier. Advanced concept studies for supersonic commercial transports entering service in the 2018-2020 period phase 2. Technical Report June, NASA Gleen Research Center, 2015. URL `https://ntrs.nasa.gov/citations/20150015837`.

[87] Markus Mrosek, Carsten Othmer, and Rolf Radespiel. Reduced-order modeling of vehicle aerodynamics via proper orthogonal decomposition. *SAE Int. J. Passeng. Cars - Mech. Syst.*, 12:225–236, 2019. doi: 10.4271/06-12-03-0016.

[88] Jayant Mukhopadhaya, Aashwin Mishra, Gianluca Iaccarino, and Juan Alonso. Design exploration under epistemic uncertainties. *Bulletin of the American Physical Society*, 63, 2018.

[89] Jayant Mukhopadhaya, Brian T. Whitehead, John F. Quindlen, Juan J. Alonso, and Andrew W. Cary. Multi-fidelity modeling of probabilistic aerodynamic databases for use in aerospace engineering. *International Journal for Uncertainty Quantification*, 10(5):425–447, 2020. ISSN 21525099. doi: 10.1615/Int.J.UncertaintyQuantification.2020032841.

[90] Chester P. Nelson, Kuang Ying Ting, Nicolas Mavriplis, Reza M. Soltani, and Eli Livne. Supersonic Configurations at Low Speeds (SCALOS): Project Background and Progress at University of Washington. *AIAA SCITECH 2022 Forum*, 2022. doi: 10.2514/6.2022-1803.

[91] Michel Nöding, Martin Schuermann, Lothar Bertsch, Marc Koch, Martin Plohr, Robert Jaron, and Jeffrey J. Berton. Simulation of landing and take-off noise for supersonic transport aircraft at a conceptual design fidelity level. *Aerospace*, 9(1): 1–23, 2022. ISSN 22264310. doi: 10.3390/aerospace9010009.

[92] Octavi Obiols-Sales, Abhinav Vishnu, Nicholas P. Malaya, and Aparna Chandramowlishwaran. Surfnet: Super-resolution of turbulent flows with transfer learning using small datasets. In *2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 331–344, 2021. doi: 10.1109/PACT52795.2021.00031.

[93] OpenVSP. Open vehicle sketch pad. `https://openvsp.org/`, 2025. Accessed: 2025-06-01.

[94] D. Bruce Owens. Weissinger's model of the nonlinear lifting-line method for aircraft design. In *36th AIAA Aerospace Sciences Meeting and Exhibit*, 1998. doi: 10.2514/6.1998-597.

[95] Andres S. Padron, Juan J. Alonso, and Michael S. Eldred. Multi-fidelity methods in aerodynamic robust optimization. In *18th AIAA Non-Deterministic Approaches Conference*. AIAA, 2016. doi: 10.2514/6.2016-0680.

[96] Francisco Palacios, Juan Alonso, Karthikeyan Duraisamy, Michael Colonno, Jason Hicken, Aniket Aranake, Alejandro Campos, Sean Copeland, Thomas Economon, Amrita Lonkar, et al. Stanford university unstructured (SU2): an open-source integrated computational environment for multi-physics simulation and design. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 287, 2013. doi: https://doi.org/10.2514/6.2013-287.

[97] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60 (3):550–591, 2018. doi: 10.1137/16M1082469.

[98] Christian Perron, Dushhyanth Rajaram, and Dimitri Mavris. Development of a multi-fidelity reduced-order model based on manifold alignment. In *AIAA AVIATION 2020 FORUM*, 2020. doi: 10.2514/6.2020-3124.

[99] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint*, 2021.

[100] Pointwise. VSP2CFD. `https://github.com/pointwise/VSP2CFD`, 2021. Accessed Github Repository: 2025-09-30.

[101] Pointwise. Pointwise, cadence design systems, retrived 26 April 2023. URL `http://www.pointwise.com/`.

[102] Edward Charles Polhamus. A concept of the vortex lift of sharp-edge delta wings based on a leading-edge-suction analogy. Technical report, NASA. Langley Res. Center, 1966. URL `https://ntrs.nasa.gov/citations/19670003842`.

[103] Stefanos Pyrialakos, Anirban Chaudhuri, and Karen E. Willcox. Multi-resolution digital twins under uncertainty with applications to multiscale systems. In *AIAA SCITECH 2026 Forum*, 2026. doi: 10.2514/6.2026-1799.

[104] Contributors PyTorch. SmoothL1loss, retrived 30 November 2024. URL `https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html`.

[105] Rishikesh Ranade, Mohammad Amin Nabian, Kaustubh Tangsali, Alexey Kamenev, Oliver Hennigh, Ram Cherukuri, and Sanjay Choudhry. DoMINO: A decomposable multi-scale iterative neural operator for modeling large scale engineering simulations. *arXiv preprint*, 2025. URL `https://arxiv.org/abs/2501.13350`.

[106] Daniel P Raymer. *Aircraft design: A conceptual approach*, chapter 2 & 12. AIAA education series. American Institute of Aeronautics and Astronautics, Reston, VA, 4. ed. edition, 2006. ISBN 1563478293.

[107] Jonathan S. Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. *arXiv preprint*, 2019. URL `https://arxiv.org/abs/1909.12673`.

[108] Mohammad H. Sadraey. *AIRCRAFT DESIGN: A Systems Engineering Approach*, chapter 1-3. John Wiley & Sons, Ltd, 2013. ISBN 9781119953401.

[109] Andrea Saltelli, Stefano Tarantola, and KP-S Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1):39–56, 1999.

[110] Paul Schwerdtner and Benjamin Peherstorfer. Greedy construction of quadratic manifolds for nonlinear dimensionality reduction and nonlinear model reduction. *arXiv preprint*, 2024. doi: 10.48550/arXiv.2403.06732.

[111] Vignesh Sella, Julie Pham, Anirban Chaudhuri, and Karen E. Willcox. Projection-based multifidelity linear regression for data-poor applications. In *AIAA SCITECH 2023 Forum*, 2023. doi: 10.2514/6.2023-0916.

[112] Sabet Seraj. *Aerodynamic Design Optimization of a Supersonic Transport Aircraft Considering Low-Speed Stability*. PhD thesis, University of Michigan, 2024. URL `https://dx.doi.org/10.7302/22987`.

[113] Sabet Seraj and Joaquim R. Martins. Aerodynamic shape optimization of a supersonic transport considering low-speed stability. In *AIAA SCITECH 2022 Forum*, 2022. doi: 10.2514/6.2022-2177.

[114] Sabet Seraj and Joaquim R. R. A. Martins. Predicting the high-angle-of-attack characteristics of a delta wing at low speed. *Journal of Aircraft*, 59(4):1071–1081, 2022. doi: 10.2514/1.C036618.

[115] Sabet Seraj and Joaquim R. R. A. Martins. Minimum trim drag for a three-surface supersonic transport aircraft. *Journal of Aircraft*, 62(3):1–12, 2025. doi: 10.2514/1.C037899. URL `https://doi.org/10.2514/1.C037899`.

[116] Tejal K Shanbhag, Beckett Y Zhou, Eduardo Molina, and Juan Alonso. A comparison of jet acoustic analysis methods. In *AIAA Aviation 2021 Forum*, page 2102, 2021.

[117] Tejal K. Shanbhag, Beckett Y. Zhou, Carlos Ilario, and Juan J. Alonso. Adjoint-based jet noise minimization using geometrical acoustics. In *30th AIAA/CEAS Aeroacoustics Conference (2024)*, 2024. doi: 10.2514/6.2024-3068.

[118] Xuqiang Shao, Zhijian Liu, Siqi Zhang, Zijia Zhao, and Chenxing Hu. PIGNN-CFD: A physics-informed graph neural network for rapid predicting urban wind field defined on unstructured mesh. *Building and Environment*, 232:110056, 2023. ISSN 0360-1323. doi: https://doi.org/10.1016/j.buildenv.2023.110056.

[119] Yang Shen, Wei Huang, Zhen-guo Wang, Da-fu Xu, and Chao-Yang Liu. A deep learning framework for aerodynamic pressure prediction on general three-dimensional

configurations. *Physics of Fluids*, 35(10):107111, 10 2023. ISSN 1070-6631. doi: 10.1063/5.0172437. URL https://doi.org/10.1063/5.0172437.

[120] Yiren Shen and Juan Alonso. Graph neural network-guided aerodynamic shape optimization for conceptual design of supersonic transport wings. In *AIAA AVIATION FORUM AND ASCEND 2025*, 2025. doi: 10.2514/6.2025-3228.

[121] Yiren Shen and Juan J. Alonso. Performance evaluation of a graph neural network-augmented multi-fidelity workflow for predicting aerodynamic coefficients on delta wings at low speed. In *AIAA SCITECH 2025 Forum*, 2025. doi: 10.2514/6.2025-2360.

[122] Yiren Shen and Juan J. Alonso. A multi-fidelity double-delta wing dataset and empirical scaling laws for gnn-based aerodynamic field surrogates. In *AIAA SCITECH 2026 Forum*, 2026. doi: 10.2514/6.2026-0686.

[123] Yiren Shen, Harsh C. Patel, Zan Xu, and Juan J. Alonso. Application of multi-fidelity transfer learning with autoencoders for efficient construction of surrogate models. In *AIAA SCITECH 2024 Forum*, number January, pages 1–19, 2024. ISBN 9781624107115. doi: 10.2514/6.2024-0013.

[124] Yiren Shen, Jacob T. Needels, and Juan J. Alonso. Vortexnet: A graph neural network-based multi-fidelity surrogate model for field predictions. In *AIAA SCITECH 2025 Forum*, 2025. doi: 10.2514/6.2025-0494.

[125] Yiren Shen, Jacob T. Needels, and Juan J. Alonso. A multi-fidelity workflow for conceptual design using graph neural network-based field prediction surrogate model. *Computers & Fluids*, page 107153, 2026. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2026.107153.

[126] Ralph C. Smith. *Uncertainty Quantification, Theory, Implementation, and Applications*, chapter Ch.1 Introduction, pages 2–10. SIAM, Philadelphia, PA, 2014. ISBN 978161973211.

[127] Dong H. Song and Daniel M. Tartakovsky. Transfer learning on multi-fidelity data. *Journal of Machine Learning for Modeling and Computing*, 3(1):31–47, 2022. doi: 10.48550/ARXIV.2105.00856. URL https://arxiv.org/abs/2105.00856.

[128] P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th Aerospace Sciences Meeting and Exhibit*, 1992. doi: 10.2514/6.1992-439.

[129] William Spitz, Frank Berardino, Richard Golaszewski, and Jesse Johnson. Development cycle time simulation for civil aircraft. Technical report, NASA, 2001. URL `https://ntrs.nasa.gov/citations/20010032406`.

[130] Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2019.112732.

[131] William Sutherland. LII. The viscosity of gases and molecular force. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 36(223):507–531, 1893. doi: 10.1080/14786449308620508.

[132] Renee Swischuk, Laura Mainini, Benjamin Peherstorfer, and Karen Willcox. Projection-based model reduction: Formulations for physics-based machine learning. *Computers and Fluids*, 179:704–717, 2019. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2018.07.021.

[133] Maciej Sypetkowski, Frederik Wenkel, Farimah Poursafaei, Nia Dickson, Karush Suri, Philip Fradkin, and Dominique Beaini. On the scalability of GNNs for molecular graphs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=klqhrq7fvB`.

[134] Kuang-Ying Ting, Chester P. Nelson, Kenneth Wiersema, and Eli Livne. Supersonic Configurations at Low Speeds (SCALOS): Progress at the University of Washington. In *AIAA SCITECH 2024 Forum*, number January, 2024. doi: 10.2514/6.2024-0898.

[135] Kuang-Ying Ting, Chester Nelson, Kenneth Wiersema, and Eli Livne. Supersonic configurations at low speeds (SCALOS): Wind tunnel progress at university of washington. In *AIAA SCITECH 2025 Forum*, 2025. doi: 10.2514/6.2025-2359.

[136] Magee Todd, Hayes Patrick, and Khodadoust Abdollah. Dynamic Stability Characteristics for Commercial Supersonic Configurations at Low-Speed Flight Conditions – Part II. In *AIAA SCITECH 2024 Forum*, number January, pages 1–48, 2024. doi: https://doi.org/10.2514/6.2024-0900.

[137] Egbert Torenbeek. *Essentials of Supersonic Commercial Aircraft Conceptual Design*. Wiley, 2020. ISBN 9781119667001.

[138] Egbert Torenbeek, Evert Jesse, and Martin Laban. Conceptual design and analysis of a mach 1.6 airliner. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2004. doi: 10.2514/6.2004-4541.

[139] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(86):2579–2605, 2008. URL `http://jmlr.org/papers/v9/vandermaaten08a.html`.

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

[141] Li Wang, Boris Diskin, Robert T. Biedron, Eric J. Nielsen, and Olivier A. Bauchau. High-fidelity multidisciplinary sensitivity analysis and design optimization for rotorcraft applications. *AIAA Journal*, 57(8):3117–3131, 2019. doi: 10.2514/1.J056587.

[142] Simon Wassing, Stefan Langer, and Philipp Bekemeyer. Adopting computational fluid dynamics concepts for physics-informed neural networks. In *AIAA SCITECH 2025 Forum*, 2025. doi: 10.2514/6.2025-0269.

[143] Lan Wei and Nikolaos M. Freris. Multi-scale graph neural network for physics-informed fluid simulation. *The Visual Computer*, May 2024. ISSN 1432-2315. doi: 10.1007/s00371-024-03402-6.

[144] Harry R. Welge, John Bonet, Todd Magee, Daniel Chen, Steve Hollowell, Aaron Kutzmann, Alan Mortlock, Josh Stengle, Chet Nelson, Eric Adamson, Steve Baughcum, Robert T. Britt, Gregory Miller, and Jummy Tai. N+2 supersonic cencept development and system integration. Contractor Report NASA/CR-2010-216842, NASA Langley Research Center, 2010.

[145] A. Wendorff, A. Variyar, C. Ilario, E. Botero, F. Capristan, J. Smart, J. Alonso, L. Kulik, M. Clarke, M. Colonno, M. Kruger, J. M. Vegh, P. Goncalves, R. Erhard, R. Fenrich, T. Orra, T. St. Francis, T. MacDonald, T. Momose, T. Economon, T. Lukaczyk, and W. Maier. SUAVE: An aerospace vehicle environment for designing future aircraft. GitHub repository, 2020. URL `https://github.com/suavecode/SUAVE`.

[146] Markus Widhalm, Mario Stradtner, Andreas Schütte, Mehdi Ghoreyshi, Adam Jirásek, and Jürgen Seidel. Stability derivatives prediction using reduced-order models on triple delta wing. *Journal of Aircraft*, 0(0):1–14, 2025. doi: 10.2514/1.C038321.

[147] Kenneth Wiersema, Kuang-Ying Ting, Chester Nelson, and Eli Livne. Supersonic configurations at low speeds (SCALOS): Correlation of CFD and wind tunnel test results. In *AIAA SCITECH 2025 Forum*, 2025. doi: 10.2514/6.2025-2362.

[148] M.G. Wilde and G. Cormery. The aerodynamic derivation of the concorde wing. In *11TH Anglo-American Aeronautical Conference*, 09/1969.

[149] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, January 2021. ISSN 2162-2388. doi: 10.1109/tnnls.2020.2978386.

[150] Sunwoong Yang, Ricardo Vinuesa, and Namwoo Kang. Enhancing graph U-Nets for mesh-agnostic spatio-temporal flow prediction. *arXiv preprint*, 2024.

[151] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *arXiv preprint*, 2019.

[152] Jianshe Zhang, Lin Li, Xu Dong, Ziqing Zhang, Yanfeng Zhang, and Xingen Lu. A discrete adjoint framework coupled with adaptive pce for robust aerodynamic optimization of turbomachinery under flow uncertainty. *Aerospace Science and Technology*, 142:108592, 2023.

[153] Tun Zhao, Weiqi Qian, Jie Lin, Hai Chen, Houjun Ao, Gong Chen, and Lei He. Learning mappings from iced airfoils to aerodynamic coefficients using a deep operator network. *Journal of Aerospace Engineering*, 36(5):04023035, 2023. doi: 10.1061/JAEEEZ.ASENG-4508.

[154] Yingxue Zhao, Haoran Li, Haosu Zhou, Hamid Reza Attar, Tobias Pfaff, and Nan Li. A review of graph neural network applications in mechanics-related domains. *Artificial Intelligence Review*, 57(11):315, Oct 2024. ISSN 1573-7462. doi: 10.1007/s10462-024-10931-y.