

# 다목적 최적화를 통한 추력 장치 익형 형상 최적화: CFD 수치해석과 머신러닝 융합 연구

서준영<sup>1\*</sup>  
광주과학기술원<sup>1</sup>

## Propeller Shape Optimization via Multi-objective Optimization: A Study Integrating CFD Analysis and Machine Learning

Joon Young Seo<sup>1\*</sup>

### 초 록

An integrated optimization framework is proposed for advanced propeller shape design by combining multi-objective optimization, steady-state FiPy CFD, and a UNet-to-GINN surrogate. The framework employs the NSGA-II algorithm to efficiently explore the design space and simultaneously optimize thrust, drag, and design constraint penalties. Candidate designs are evaluated via four strategies: (1) simplified drag estimation, (2) FiPy-based CFD, (3) simplified drag + UNet surrogate, and (4) FiPy CFD + UNet surrogate. Numerical results show that, relative to their base methods, the simple+UNet surrogate raises thrust value by 17.9 % but increases drag value by 57.0 %, while the FiPy+UNet hybrid boosts thrust by 59.2 % and reduces drag by 40.8 %. These findings demonstrate that coupling machine learning with physics-based simulation can accelerate design exploration and improve aerodynamic performance. Future work will refine the surrogate model, integrate additional metrics, and perform uncertainty analysis to enhance practical applicability.

**Key Words** : Propeller Shape Optimization, Multi-objective Optimization, Computational Fluid Dynamics, Machine Learning, Surrogate Model, Aerodynamic Performance, Design Optimization

### I. 서론

In aerospace engineering, optimal propeller design is essential for high-performance aerodynamics. Especially, the geometrical character of a propeller is directly linked to its efficiency. The intention of an engineer seeking for this specific enhancement is met by achieving a balance between the parameters of thrust and drag. Traditional single-objective methods often fail to capture these complex trade-offs, motivating the use of multi-objective techniques such as the Non-dominated Sorting Genetic Algorithm II (NSGA-II) for this study.

The framework presented introduces an empirical model of simple drag estimation, a partial differential equation solver FiPy with Computational Fluid Dynamics (CFD) under steady-state conditions, and machine learning approaches for surrogate modeling. While the empirical model goes through rapid assessments by decomposing the drag coefficients into three, the FiPy-based CFD approach analyzes fluid flow under steady-state conditions, and the UNet-to-GINN architecture predicts aerodynamic responses from past shape profiles and its corresponding drag value.

### II. 본론

#### 2.1 Optimization Problem Formulation

The propeller design problem is posed as a multi-objective optimization task where the primary objectives are to maximize thrust, minimize aerodynamic drag, and minimize penalties associated with design constraint violations.

Candidate designs are parameterized using asymmetric cubic Bézier curves to capture the complex geometric features of a propeller by using the Non-dominated Sorting Genetic Algorithm II (NSGA-II) is employed<sup>(2)</sup>.

##### 2.1.a The Cubic Bézier Curves

The cubic Bézier curve is expressed as:

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3, \quad (1) \\ \text{for } t \in [0,1],$$

where  $P_0$  and  $P_3$  represent the fixed endpoints and  $P_1$  and  $P_2$  modulate the curve's tangency and curvature. This parameterization provides the necessary flexibility to capture complex geometrical features.

##### 2.1.b Generation of Optimized Shapes

After the optimization process identifies optimal control point values through NSGA-II, a two-dimensional profile is generated by computing the corresponding upper and lower Bézier curves. This 2D representation forms the propeller profile<sup>(Fig. 1)</sup>, which

is then extended into three dimensions using edge guideline function generated as a 3D B zier spline (Fig. 2).

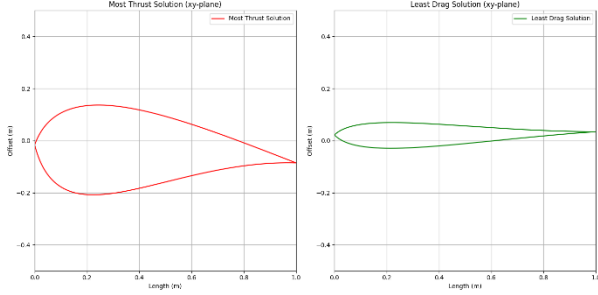


Fig. 1. Propeller Profile Generated via B zier Parametrization

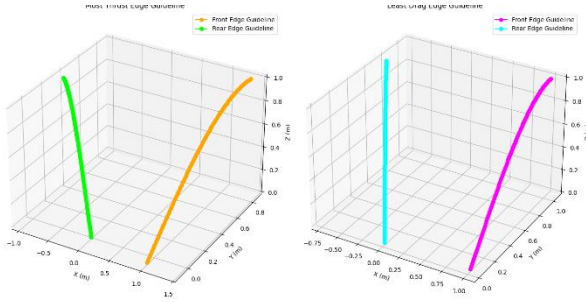


Fig. 2. 3D Edge Guideline for Propeller Design

### 2.1.c Objective Function

Mathematically, each candidate design is denoted by vector  $x = [dim_{char}, cp1_h, cp2_h, cp1_{lh}, cp2_{lh}]$ , where  $dim_{char}$  as the characteristic dimension and  $cp$  as control point scaling factors. Then the objective function can be expressed as:

$$f(x) = [T(x), -D(x), -P(x)], \quad (2)$$

where  $T(x)$  is the thrust,  $D(x)$  is the drag, and  $P(x)$  is the penalty function, ultimately generating propeller shapes with improved thrust, reduced drag and penalties (Fig.3).

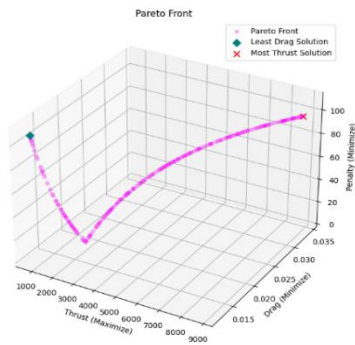


Fig. 3. 3D Pareto Front of Optimized Designs

## 2.2 Governing Equations and Analysis Methods

To evaluate each candidate design, three analysis methods are implemented: (a) A simplified drag estimation, (b) a FiPy-based CFD simulation, and (c) a machine learning surrogate model based on the UNet-to-GINN framework.

### 2.2.a Simplified Drag Estimation

For Simplified Drag Estimation, an empirical model offers a rapid, preliminary assessment by decomposing the drag coefficient into three components: pressure drag, friction drag, and wake drag<sup>(3)</sup>:

$$C_{D,pressure} = 0.5 \left( \frac{D_{eff}^2}{L_{eff}^2} \right), \quad (3)$$

$$C_{D,friction} = \begin{cases} \frac{24}{Re}, & \text{if } Re \leq 10^5, \\ \frac{0.074}{Re^{0.2}}, & \text{if } Re > 10^5, \end{cases} \quad (4)$$

with the effective dimensions defined as:

$$D_{eff} = D \sin \alpha, \quad L_{eff} = L \cos \alpha, \quad (5)$$

where  $D$  and  $L$  are characteristic dimensions and  $\alpha$  is the angle of attack.

$$C_{D,wake} = 0.1 \left( \frac{D}{L} \right). \quad (6)$$

The total drag coefficient is then approximated by:

$$C_D = C_{D,pressure} + C_{D,friction} + C_{D,wake}. \quad (7)$$

### 2.2.b FiPy-based CFD

In the FiPy-based CFD Simulation method, a Partial Differential Equation (PDE) solver using the Finite Volume Method (FVM) is used to obtain a surrogate pressure field by solving the following steady-state diffusion equation. In a general form, the scalar conservation equation governing transport phenomena is written as:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (u \phi) = \nabla \cdot (\Gamma \nabla \phi) + S, \quad (8)$$

where  $\phi$  is the scalar field used as a pressure surrogate,  $u$  is the convective velocity field,  $\Gamma$  is the diffusion coefficient typically set to 1.0, and  $S$  represents any source or sink terms.

For steady-state assumption, it is assumed that the system has reached  $\frac{\partial \phi}{\partial t} = 0$ . Convection is considered minimal, meaning the term  $\nabla \cdot (u \phi)$  is negligible. In this case, there is no additional generation or absorption of the scalar quantity, therefore  $S = 0$ . Under these assumptions, the conservation equation reduces to:

$$\nabla \cdot (\Gamma \nabla \phi) = 0, \quad (9)$$

which is the steady-state Laplace equation. In this implementation. In order to mimic the inflow and outflow conditions, the following boundary conditions are imposed:

$$\phi|_{inflow} = 1.0, \quad \phi|_{outflow} = 0.0 \quad (10)$$

Instead of performing a rigorous surface integration of pressure and shear stresses, the drag is approximated by summing the magnitudes of the gradients over the computational domain:

$$D_{CFD} \approx \sum_{i,j} |\nabla \phi_{i,j}| \quad (11)$$

Although simplified, empirical calibration has verified that this approach effectively ranks candidate designs by relative drag performance<sup>(4)</sup>.

### 2.2.c UNet-to-GINN

To reduce the computational cost associated with full CFD simulations during iterative design evaluations, a machine-learning surrogate model is incorporated for rapid aerodynamic drag estimation based on documented archives of the past candidate propeller profiles and its corresponding drag values. This surrogate is constructed by combining a UNet architecture with a subsequent Geometric Inference Neural Network (GINN) analysis stage.

An input image  $X$  representing a candidate propeller profile is processed through a UNet, which employs an encoder-decoder structure with skip connections. In the encoder, a series of convolutional layers coupled with max pooling extracts hierarchical, multi-scale features from the image:

$$F_l = \sigma(W_l * X_l + b_l), \quad (12)$$

where  $W_l$  and  $b_l$  denote the weights and biases at layer  $l$ ,  $*$  represents the convolution operation, and  $\sigma$  is the non-linear activation function.

The decoder reconstructs the spatial resolution of the input by using transposed convolutions. Skip connections from the encoder layers are concatenated with the decoder outputs to refine the reconstruction. The final output of the UNet is a feature map  $U$  that has the same dimensions as the original image:

$$U = \text{Decoder}(F_0, F_1, \dots, F_L), \quad (13)$$

where  $F_0, F_1, \dots, F_L$  are encoder outputs from different resolution levels.

Map  $U$  is then processed using a GINN function to derive a quantitative drag estimate. The process aggregates the pixel-wise information from  $U$  and applies calibration scaling factors to obtain a surrogate drag value:

$$\hat{D} = \alpha \times \left( \sum_{i,j} U(i,j) \right) + \beta, \quad (14)$$

where  $U(i,j)$  is the UNet output at pixel  $(i,j)$ , and  $\alpha$  and  $\beta$  are scaling parameters determined through empirical calibration. In the current implementation, setting  $\alpha = 0.01$ ,  $\beta = 0$  and applying an overall scaling factor of 1000 yields:

$$D_{UNet-GINN} = 1000 \times \left( 0.01 \times \sum_{i,j} U(i,j) \right). \quad (15)$$

The UNet is trained using a dataset comprised of candidate designs and corresponding reference drag values  $D_{ref}$  calculated from CFD simulations. A Mean Squared Error (MSE) loss function is employed:

$$\mathcal{L} = \frac{1}{N} \sum_{k=1}^N \left( D_{UNet-GINN}^{(k)} - D_{ref}^{(k)} \right)^2, \quad (16)$$

where  $N$  is the number of training samples.

### 2.3. Terminal User Interface

The Wing Bakery<sup>(Fig. 4)</sup> user interface is designed as a terminal-based, text-driven system with engaging ASCII art that guides users through the optimization process. Upon launching the application, users see a main menu offering clear options, such as starting the optimization ("Start Baking"), accessing additional functionalities ("Visit the Great Wizard"), or exiting the program<sup>(Fig. 5)</sup>.

The interface prompts users for configuration inputs (e.g., environmental and optimization parameters), and it provides visual feedback during design evaluation and simulation with progress indicators and messages that mimic a culinary workflow. This intuitive layout ensures that even complex processes like CFD simulation and deep learning surrogacy are presented in an easily navigable and interactive format.



Fig. 4. Main Interface of the 'Wing Bakery'

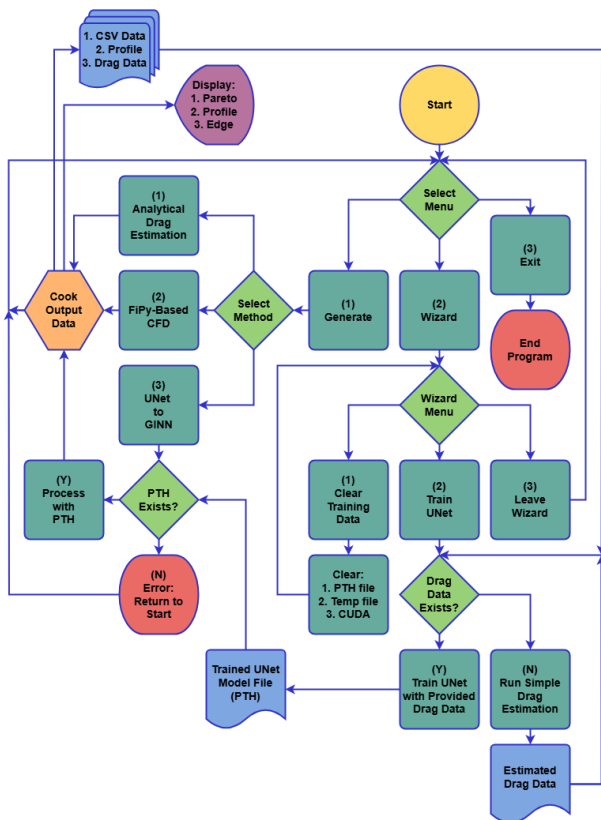


Fig. 5. Workflow Diagram of the 'Wing Bakery'

## 2.4 Comparison in Performance by Methods

The final step is to evaluate four analysis strategies: (1) Simple drag estimation, (2) FiPy-based CFD, (3) Simple estimation augmented with a UNet surrogate, and (4) FiPy CFD combined with a UNet surrogate. To isolate the effect of the machine-learning surrogate, we compare each base method (1 or 2) with its UNet-hybrid counterpart (3 or 4). Since minimizing drag is our primary design objective, we designate the Least-Drag Solution as the ideal baseline and use the Most-Thrust Solution, which remains shape-invariant across all methods, as our reference standard. The surrogate-augmented Least-Drag Solutions yield:

- Method 3 (Simple + UNet): Thrust ↑ 17.88 % but Drag ↑ 57.01 % (vs. Method 1)
- Method 4 (FiPy + UNet): Thrust ↑ 59.24 % and Drag ↓ 40.83 % (vs. Method 2)

These results show that, while a UNet-only surrogate can boost thrust, it does so at the cost of substantially higher drag; by contrast, the FiPy + UNet hybrid approach simultaneously increases thrust and reduces drag.

## III. 결 론

The comparative study confirms that integrating machine learning with CFD simulation yields the be

st overall aerodynamic performance. While combining simple drag estimations with UNet surrogate can accelerate design iterations and improve thrust, it tends to increase drag. In contrast, the hybrid FiPy + UNet framework not only delivers a 59.24 % thrust gain but also reduces drag by 40.83 %, demonstrating the synergistic benefit of coupling data-driven models with physics-based simulation.

Future work will validate these computational findings against experimental measurements and explore additional performance metrics (e.g., efficiency, noise). Such validation is essential to assess real-world applicability and to further refine the surrogate's predictive accuracy.

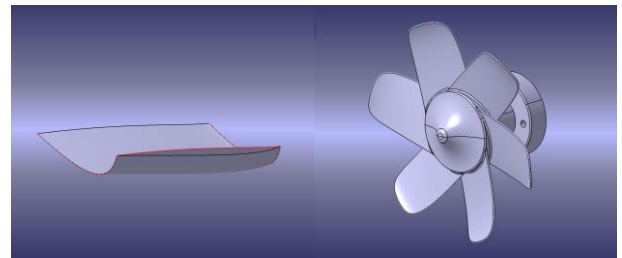


Fig. 6. 3D Representation and Example of Final Product using Generated Data

## 후 기

I would like to express my deepest gratitude to Professor Seong-Im Choi and the members of the Data-Driven Simulation and Design Optimization Laboratory, Gwangju Institute of Science and Technology (GIST), for every invaluable guidance and support throughout this research.

I also extend my sincere thanks to Mitsui E&S Co., Ltd. (Tokyo, Japan), whose creativity and collaborative spirit in marine engineering and precision technology inspired the development of the "Wing Bakery" propeller-generating tool.

## 참고문헌

- Anderson, J. D., *Fundamentals of Aerodynamics* (5th ed.), McGraw-Hill, New York, 2010.
- Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2002, pp. 182-197.
- Walker, R. E., Stone, A. R., and Shandor, M., "Secondary Gas Injection in a Conical Rocket Nozzle," *AIAA Journal*, Vol. 1, No. 2, 1963, pp. 334-338.
- Guyer, J. E., Wheeler, D., & Warren, J. A., "FiPy: Partial Differential Equations with Python," *Computing in Science & Engineering*, Vol. 11, No. 3, 2009, pp. 6-15.