AIAA SciTech Forum
8-12 January 2024, Orlando, FL
AIAA SCITECH 2024 Forum

10.2514/6.2024-2013

Optimizing Diffusion to Diffuse Optimal Designs

Cashen Diniz* and Mark D. Fuge †
University of Maryland, College Park, Maryland, 20742

Generative models offer the possibility to accelerate and potentially substitute parts of the often expensive traditional design optimization process. However, current implementations of these models often ignore aspects of the optimization process that, among other benefits, could be used to improve model accuracy and design feasibility. We implement a latent denoising diffusion model (DDM) capable of generating airfoil geometries conditioned on flow parameters and an area constraint. Additionally, we create a novel, diverse dataset * of optimized airfoil designs that better reflects a realistic design space than has been done in previous work. The model is applied to this dataset, and key metrics are assessed both statistically and with an open-source computational fluid dynamics (CFD) solver to determine the performance of the generated designs. Our final contribution is the implementation of a diffusion model with a variance schedule based on the statistical changes in design variables along the aggregate optimization trajectories of our data. We find this model produces better-performing airfoils and improves design feasibility. We acknowledge that the verification and reasons behind this improvement remain unclear, but hope to present an initial step towards further investigations of optimization-informed generative models.

I. Introduction

A. Motivation and Related Works

Airfoil design is an approachable, yet sufficiently difficult and realistic problem. Traditionally, airfoil design problems have been tackled by applying adjoint-based optimization methods, which can be implemented relatively efficiently and robustly [1]. Despite this, these solvers can be difficult for a non-expert to use properly and importantly, still require much in the way of computational resources for any large-scale design space study. To address some of these concerns, some authors have presented data-driven approaches to this problem [2, 3].

More recently, generative machine learning models have garnered significant attention in both academic and commercial circles. Generative models, when applied to engineering problems allow for the modeling of diverse, multi-modal distributions, and depending on the model used, can do so in a relatively data-efficient manner. One particular type of generative model, the denoising diffusion model (DDM) has recently surpassed other models such as generative adversarial networks (GANs) in image generation tasks [4]. Notable examples of such models include OpenAI's DALL-E2 and Stability AI's Stable Diffusion [5, 6]. Generative models have also demonstrated their efficacy in various engineering optimization design problems, either as a replacement or complement to traditional approaches. This is because engineering problems often exhibit multimodal behavior, high dimensionality, and non-intuitive design spaces [7, 8]. However, applying generative models to engineering design problems presents unique challenges due to the need to satisfy physical constraints and simulate designs, making the task more complex than simpler image processing or generation tasks. Regenwetter *et al.* [9] identified four key challenges to applying generative models in engineering design, namely data sparsity, modeling design performance, feasibility, and sample diversity or model creativity.

DDMs have shown promise in image generation and processing tasks, outperforming GANs with generally higher quality and more diverse outputs [4]. DDMs are also relatively more stable to train than GANs, as they do not rely on an adversarial/game theoretic approach. Although DDMs are relatively new to the field of engineering design, early research suggests advantages in topology optimization. For example, in one study, DDMs reduced feasibility error by 11-fold and improved performance by a factor of 8 [10]. In this paper, we propose modifications that attempt to better align DDM training to the behavior of traditional gradient-based optimizers.

*Research Assistant and Ph.D. Student, Department of Mechanical Engineering.
†Associate Professor, Department of Mechanical Engineering.
*https://github.com/IDEALLab/OptimizingDiffusionSciTech2024

Our work focuses on the problem of airfoil shape optimization, in particular, we follow the conditional generative model work of Chen *et al.* who used a Conditional GAN to generate airfoil designs based on a given Mach number (Ma), Reynold's number (Re) and desired coefficient of lift (C_L) [11]. We add an additional minimum area constraint parameter (A_{min}) due to the diverse design space used.

Several relevant publications have also applied generative models to design engineering problems. Ghosh *et al.* presented a probabilistic framework for generating airfoil designs using an invertible neural network trained on data generated by a real-valued non-volume preserving (real NVP) model. In this way, they were able to replace a traditional gradient-based iterative optimization approach. The real NVP model applies a sequence of probabilistic transformations to a latent input [12]. However, most publications apply a GAN to the task airfoil shape optimization in favor of a probabilistic approach. For example, Du *et al.* applied a GAN to efficiently reconstruct the (unconditional) UIUC dataset of airfoil designs [13]. Achour *et al.* built on the airfoil-generating GAN architecture to add conditioning to generations, with an intent to replace aspects of the traditional optimization pipeline [14]. Separately, Morton *et al.* have applied conditional GANs to predict the flow around circles and half-cylinders [15]. Although this work was not focused on the design optimization process, it demonstrates the broad applicability of generative models to the more general issue of parameterized fluid flow problems.

B. Diffusion Model Theory

A denoising diffusion model is optimized by maximizing the Evidence Lower Bound (ELBO). ELBO can be derived from the definition of the probability $p(\mathbf{x})$ of all observed $\mathbf{x}$ which marginalizes the latent variables $\mathbf{z}$. Let $q_\theta(\mathbf{z}|\mathbf{x})$ be the variational distribution with model parameters θ which is used in the forward diffusion process [16].

$$p(\mathbf{x}) = \int p(\mathbf{x},\mathbf{z})d\mathbf{z} \tag{1}$$

$$\log\ p(\mathbf{x}) = \log \int p(\mathbf{x},\mathbf{z})d\mathbf{z} \tag{2}$$

$$= \log \int \frac{p(\mathbf{x},\mathbf{z})q_\theta(\mathbf{z}|\mathbf{x})}{q_\theta(\mathbf{z}|\mathbf{x})} \tag{3}$$

$$= \log\ \mathbb{E}_{q_\theta(\mathbf{z}|\mathbf{x})}\left[\frac{p(\mathbf{x},\mathbf{z})}{q_\theta(\mathbf{z}|\mathbf{x})}\right] \tag{4}$$

$$\geq \mathbb{E}_{q_\theta(\mathbf{z}|\mathbf{x})}\left[\frac{p(\mathbf{x},\mathbf{z})}{q_\theta(\mathbf{z}|\mathbf{x})}\right] \qquad \text{(Using Jensen's Inequality)} \tag{5}$$

Adapted from Luo: [16]

Diffusion models are unique in that the latent variable, z, is actually a vector with the same cardinality as the data x, but noised according to a variance, β_t, schedule. The amount of noise is notated by the index t ranging from 0 to T, where 0 represents the original, un-noised data (in implementation, the 0th step contains a small non-zero amount of noise), and T represents the data with the most amount of noise added. As T approaches infinity, the data approaches an isotropic Gaussian distribution.

Knowing this, the posterior conditioned on the un-noised data, x_0 can be written as a product over a Markov chain [16]:

$$q(\mathbf{x_{1:T}}|\mathbf{x_0}) = \prod_{t=1}^{T} q(\mathbf{x_t}|\mathbf{x_{t-1}}) \tag{6}$$

Letting $\alpha_t = 1 - \beta_t$, and assuming the isotropic Gaussian distribution above, the posterior conditioned on the previous step is:

$$q(\mathbf{x_t}|\mathbf{x_{t-1}}) = \mathcal{N}(\mathbf{x_t}; \sqrt{\alpha_t}\mathbf{x_{t-1}}, 1-\alpha_t) \tag{7}$$

This adequately describes the *forward* process. To describe the *reverse* process, we can rewrite the model probabilities as:

$$p_\theta(\mathbf{x_{0:T}}) = p(\mathbf{x_T}) \prod_{t=1}^{T} p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}) \tag{8}$$

with

$$p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}) = \mathcal{N}(\mathbf{x_{t-1}}; \mu_{theta}(x_t, t), \sigma_\theta(x_t, t)) \tag{9}$$

[16]. In practice and in our implementation, σ_θ is equivalent to β_t. Finally, the ELBO can be written concisely as

$$ELBO = \mathbb{E}_{q(\mathbf{x_{0:T}}|\mathbf{x_0})}\left[\frac{p(\mathbf{x_{0:T}})}{q(\mathbf{x_{1:T}}|\mathbf{x_0})}\right] \tag{10}$$

The ELBO can be practically maximized by calculating a metric such as MSE over the predicted and actual noise added to the data, which is the approach taken in this paper [17].

II. Airfoil Optimization Problem Setup

We formulate a diverse, multi-modal, optimization problem by first specifying the flow conditions; Mach number (M) and Reynolds's number (Re). In addition, we require the final design to attain the specified coefficient of lift (C_l) while keeping the area above the minimum area $\left(\left(\frac{A}{A_{init}}\right)_{min}\right)$ constraint. Thickness constraints are imposed based on the initial design—although we note that in practice, none of the adjoint solver iterations activated these constraints. We also impose no-twist leading and trailing edge conditions.

After defining the problem, we used Latin-hypercube sampling to generate the input parameter space, within the bounds specified in Table 1.

A. Design Space

Table 1 Sampled Parameter Bounds

Category	Parameter	Lower	Upper	Description
Flow Condition	M	0.4	0.9	Mach Number
	Re	1E6	10E6	Reynold's Number
Constraint	C_l^{con}	0.5	1.2	Coefficient of Lift
	$\left(\frac{A}{A_{init}}\right)_{min}$	0.75	1.0	Minimum Area Fraction

The bounds were chosen purposefully to cover part of the transonic regime which is known to be difficult to model with non-RANS approaches, such as panel methods [2].

B. Problem Formulation

We define the specific optimization problem below in equation 11.

$$\begin{aligned}
\min_{\Delta y_i, \alpha} \quad & C_d \\
\text{s.t.} \quad & C_l = C_l^{con} \\
& -0.025 \leq \Delta y_i \leq 0.025 \\
& 0.0 \leq \alpha \leq 10.0 \\
& \left(\frac{A}{A_{init}}\right)_{min} \leq \frac{A}{A_{init}} \leq 1.2 \\
& 0.15 \leq \frac{t}{t_{init}} \leq 3.0 \\
& t_{TE} = t_{TE,init} \\
& y_{TE}^{upper} = -y_{TE}^{lower} \\
& y_{LE}^{upper} = -y_{LE}^{lower}
\end{aligned} \tag{11}$$

Where Δy_i is the difference in the y-coordinates of the Free-Form Deformation (FFD) cage from the initial FFD cage. As such, Δy_i, has a large impact on how diverse the resulting design space is, and its value can be thought of as being embedded in the initial airfoil shape. Table 2 describes each of the optimization parameters in detail.

Table 2 Optimization Problem Parameters

Category	Parameter	Quantity	Lower	Upper	Units	Description
Objective	C_d	1	-	-	Non-Dim./Counts	Coefficient of Drag
Variable	Δy_i	20	-0.025	0.025	m	Change from initial FFD cage y value: $\Delta y_i = y_i - y_{init}$
	α	1	0.0	10.0	Degrees	Angle of Attack
Constraint	$C_l = C_l^{con}$	1	0.0	0.0	Non-Dim.	Coefficient of Lift
	$\frac{A}{A_{init}}$	1	$\left(\frac{A}{A_{init}}\right)_{min}$	1.20	Non-Dim.	Area Fraction; Relative to Initial
	$\frac{t}{t_{init}}$	100	0.15	3.00	Non-Dim.	Thickness Fraction; Relative to Initial
	$t_{TE} = t_{TE,init}$	1	0.0	0.0	m	Trailing Edge Thickness: Blunted TE Condition
	$y_{TE}^{upper} = -y_{TE}^{lower}$	1	0.0	0.0	m	Trailing Edge FFD Point Shearing Twist Condition
	$y_{LE}^{upper} = -y_{LE}^{lower}$	1	0.0	0.0	m	Leading Edge FFD Point Shearing Twist Condition

In addition to the scalar optimization parameters, 1400 initial airfoil designs were taken from a random subset of the UIUC dataset [18]. Raw coordinates from the dataset were re-sampled, smoothed, and normalized as part of a pre-processing procedure. Samples from this dataset are shown in Figure 1.

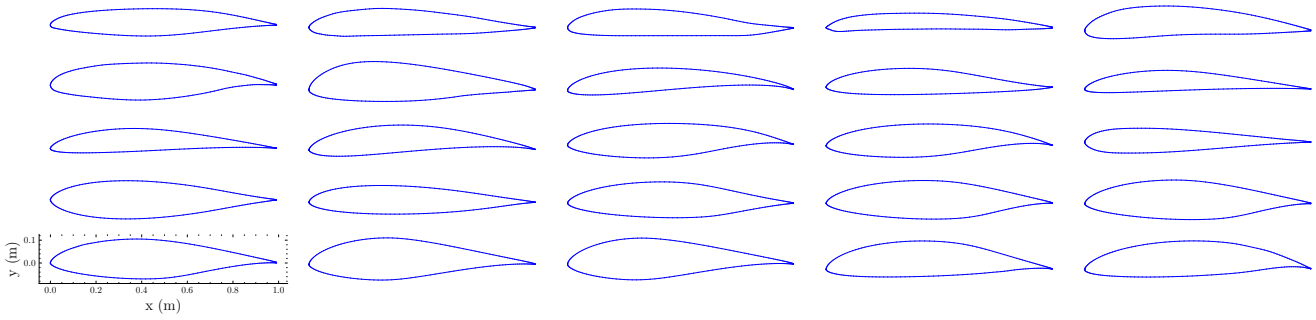


Fig. 1 Processed Samples From UIUC Dataset

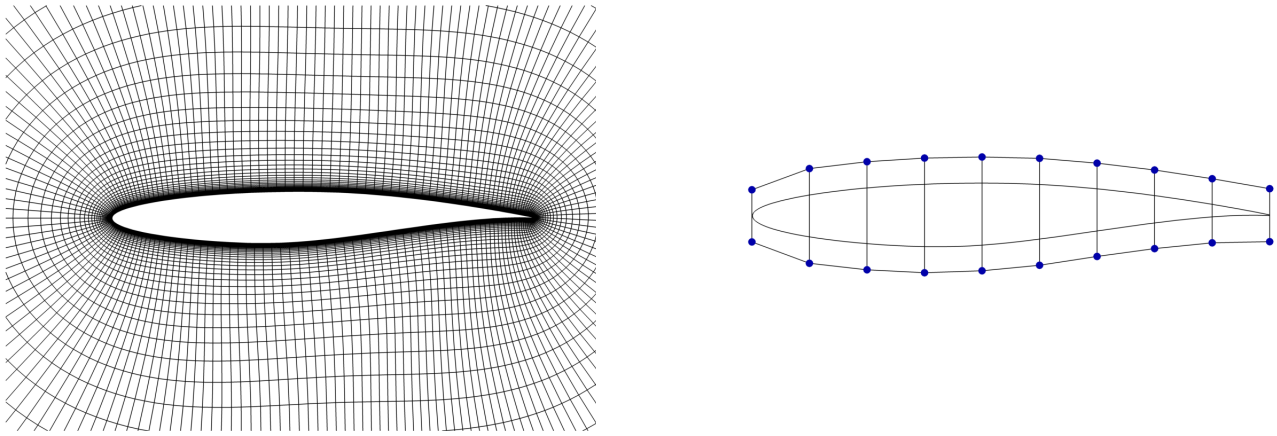
C. Solver Setup

We used the MACH-Aero framework for all discussed simulations [†]. Within the MACH-Aero framework, structured mesh generation was handled automatically by pyHyp, a hyperbolic mesh generator [19]. Additional details concerning the mesh setup are described in the next section. pyOptSparse, a sparse numerical optimization suite, was used to efficiently solve the nonlinear optimization problem [20]. Specifically, we use the sequential least squares programming algorithm (SLSQP). pyGeo, a geometry parameterization module, was used to parameterize and deform the FFD cages, with warping of the cages handled by the robust IDWarp framework [21, 22]. The flow simulation, and the corresponding adjoint solution, was computed using the open-source ADflow solver [23, 24]. Within the solver, we used the approximate Newton–Krylov (ANK) method for improved convergence and robustness [25]. ADflow was configured to run Reynolds Averaged Navier Stokes (RANS) simulations with the Spalart-Allmaras model for turbulence effects.

[†]https://github.com/mdolab/MACH-Aero

D. Geometry Parameterization and Meshing

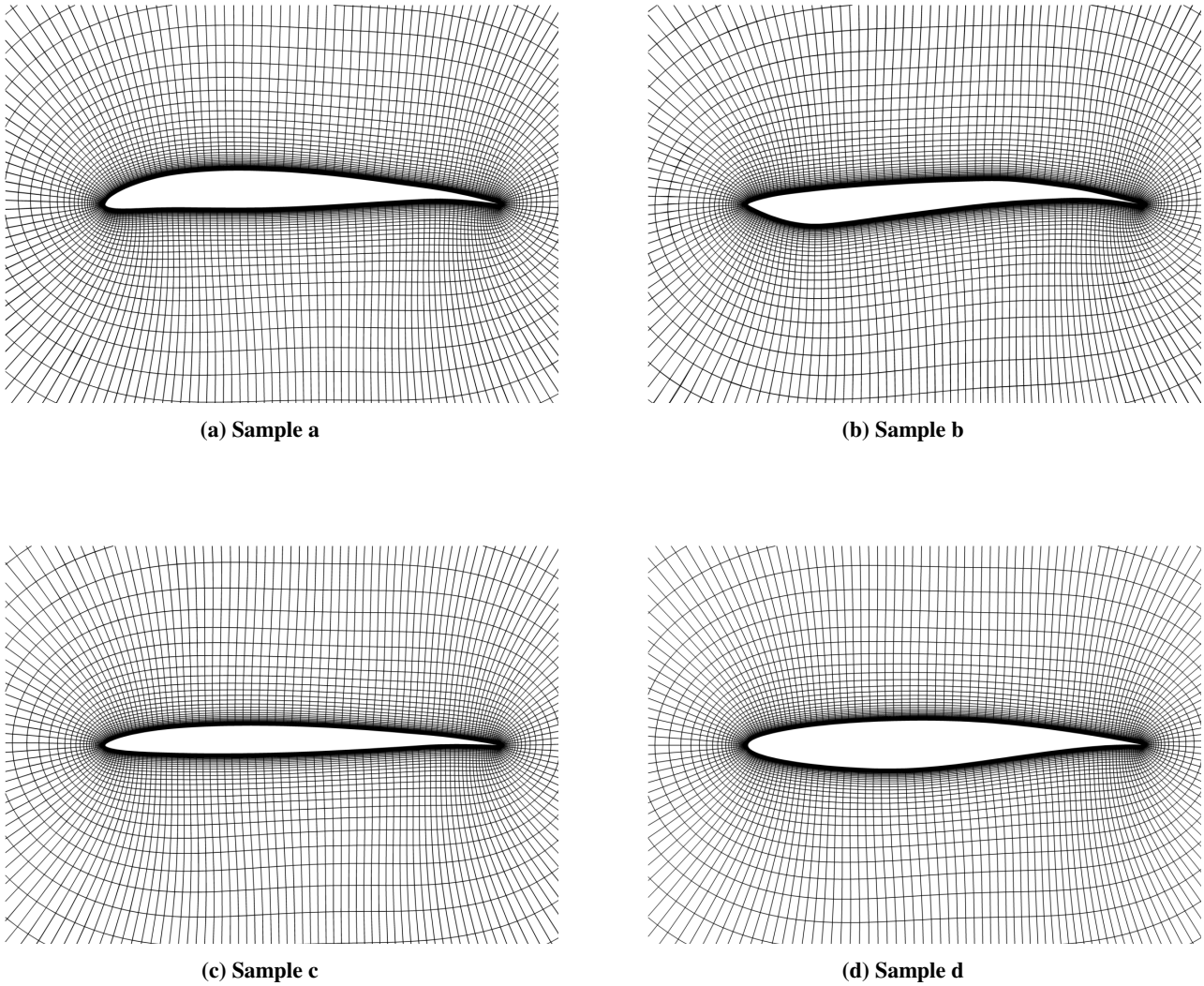
Since we had to run a diverse set of input flow conditions and also initial airfoils, we decided to use a meshing setup that was able to successfully match the results from the benchmark ADODG RAE2822 airfoil optimization problem. A more detailed description of this benchmark problem can be found in [1]. The mesh used to successfully replicate the results from this benchmark problem, as well as the corresponding FFD cage is shown in Figure 2.



(a) Example Mesh **(b) FFD Cage**

Fig. 2 Example Setup: RAE2822

We set the far-field distance to 100 chord lengths away from each airfoil, and extrude the mesh using 100 grid levels. For each optimization run, we set the number of FFD points to 20 based on the results in [1], which showed reduced benefit after about 20 points were used. Finally, to improve stability we blunt the trailing edge at the point 99% away from the leading edge. Figure 3 shows the meshes for four random samples in our dataset after applying the described setup.



(a) Sample a **(b) Sample b**

(c) Sample c **(d) Sample d**

Fig. 3 Four Random Samples: Meshing Setup

III. Results: Dataset

From the starting dataset of 1400 airfoils, we observed 935 successful optimizations, corresponding to a 67% success rate. The training, testing, and validation sets were split into 748, 140, and 47 airfoils respectively. We display a random subset of the resulting optimized airfoil dataset in Figure 4.

A. Optimization Results

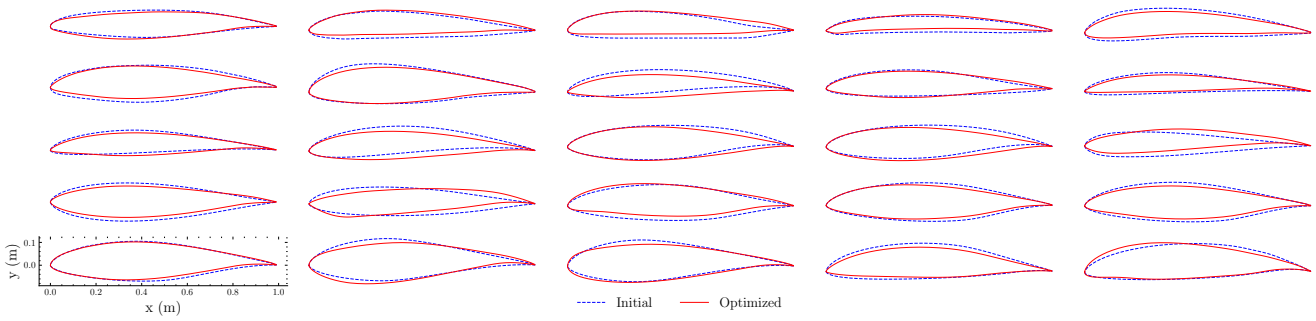


Fig. 4 Examples of Initial and Optimized Airfoils

Of note is the relatively slight change in shape from the initial design. This can be explained by the proportionately small deviation constraint (±0.025 m) from the initial FFD points we enforced on the optimizer. In testing, we found that making the range much larger made it difficult for the adjoint optimizer to converge consistently across the relatively wide range of flow parameters we sampled from. Techniques such as FFD point adaption, as described in [1], may be required for more dramatic changes to the design space.

Figure 5 presents more detailed results for the four aforementioned random samples in our dataset. In addition, Figure 6 shows the resulting pressure contours for the final, optimized designs. We note these samples display design characteristics one might expect for the given design parameters we used. For example, Figure 5 (b), corresponds to an optimized design with a relatively sharp, "diamond-shaped", leading-edge profile typical of high Mach number airfoil designs, as such designs can better take advantage of the shock-expansion patterns relevant to transonic regimes [26]. Finally, we present the optimization trajectory for the four random designs in Figure 7.

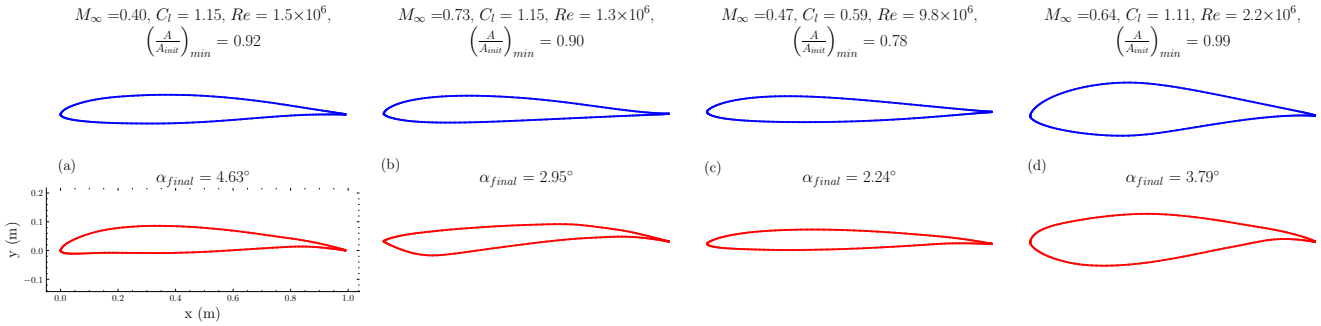
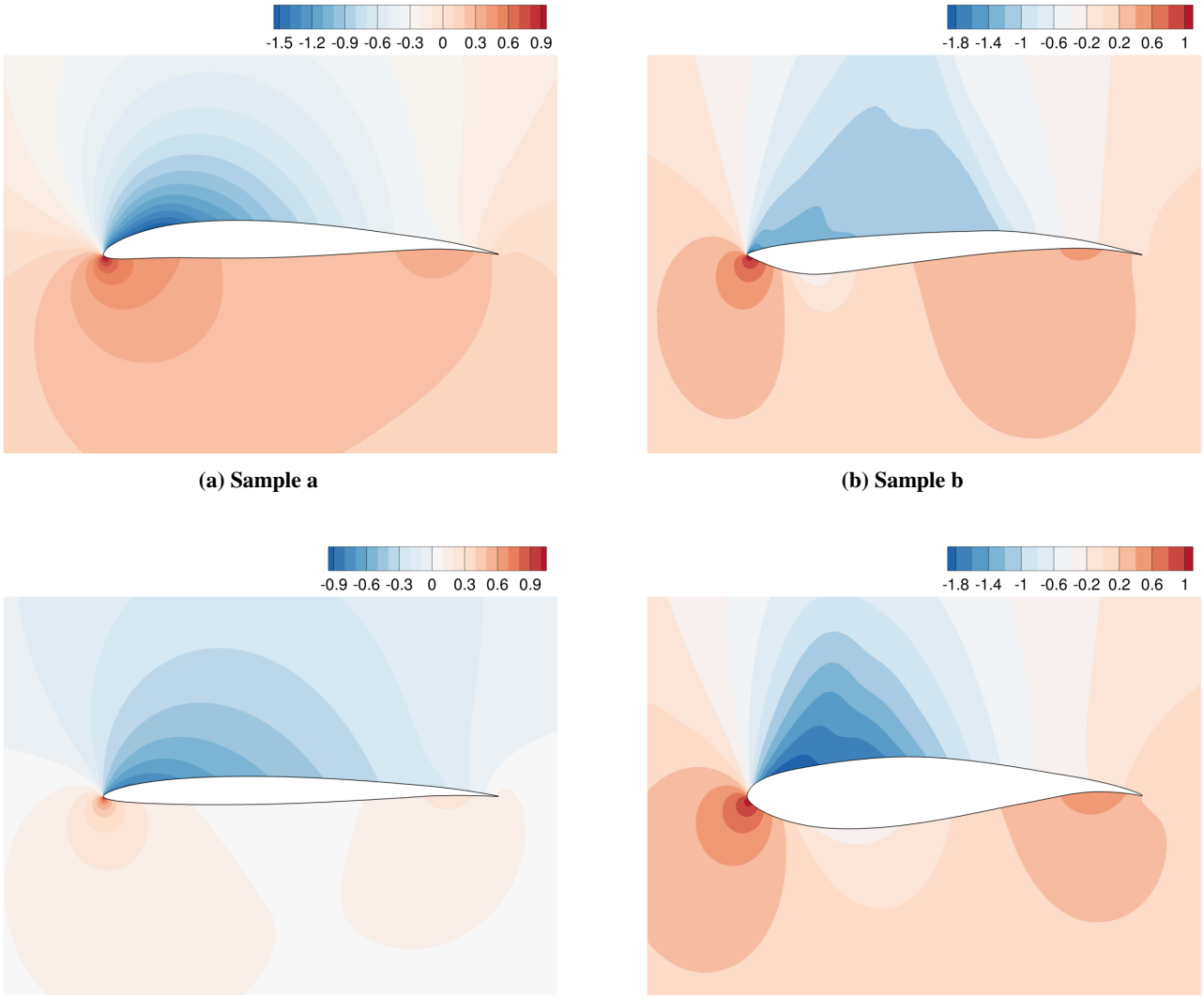


Fig. 5 Four Random Samples: Optimized Airfoils



(a) Sample a **(b) Sample b**

(c) Sample c **(d) Sample d**

Fig. 6 Four Random Samples: C_p Contours

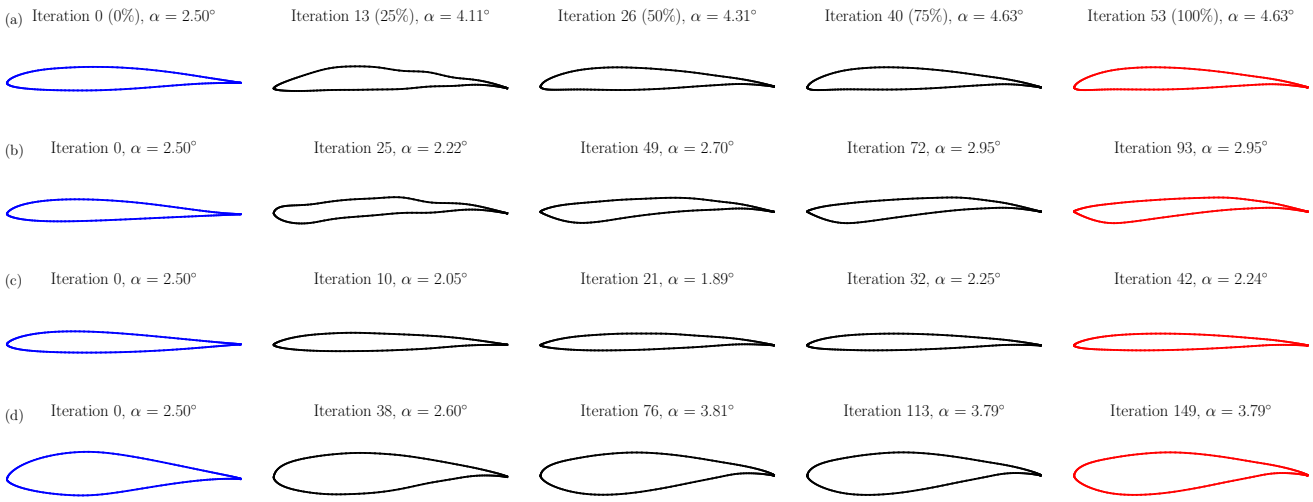


Fig. 7 Four Random Samples: Optimization Trajectories

We make note of the fact that, beyond 25% of the optimization trajectory, the design stays relatively the same, with the latter steps only introducing marginal changes with a correspondingly smaller effect on C_d.

B. Data Analysis

To interpret all of the variable trajectory-length optimization data, we linearly interpolated the trajectory data to 1000 samples for all key objective, constraint, and design variables. Figure 8 shows the aggregate mean and standard deviation of the difference from the optimal objective, C_d^*, in drag counts (1 drag count equals 0.0001). As was the case visually in Fig. 7, Fig. 8 confirms that the change in optimal C_d is most affected by changes occurring in the first 20-25% of the design trajectory.

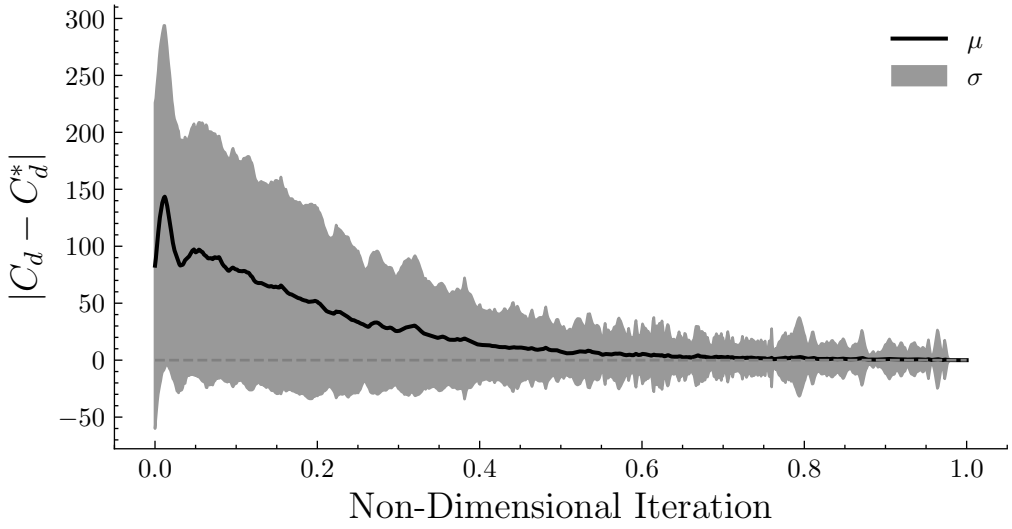
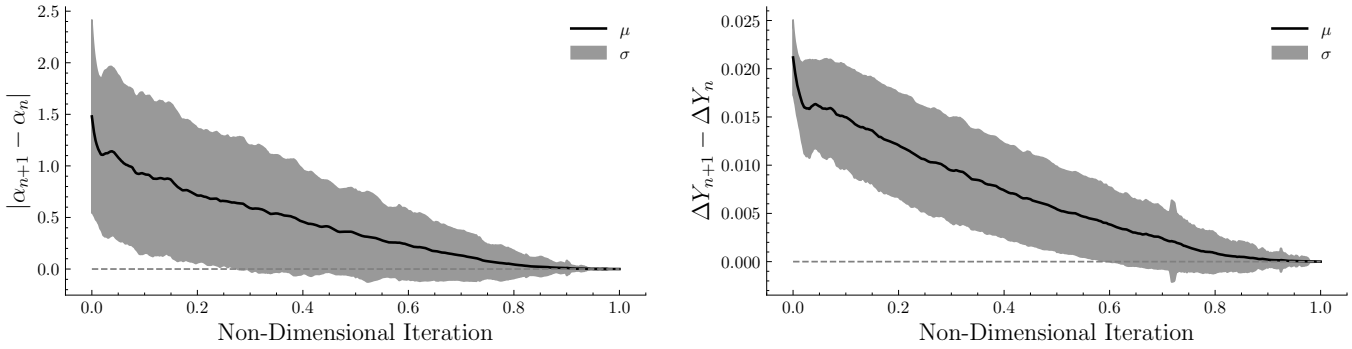


Fig. 8 Aggregate C_d Convergence

The design variable optimization trajectories present a more or less linear decrease in the average per-iteration change for the majority of the trajectory, as can be seen for both angle of attack and in the summation of over the FFD y-coordinates in Figure 9. Where $\Delta Y_{n+1} - \Delta Y_n = \frac{1}{N_{ffd}} \sum_i^{N_{ffd}} |\Delta y_{n+1}^i - \Delta y_n^i|$. Note that this is the absolute change from the initial FFD cage, and is not the same as the change in coordinate value, which would produce a mean around

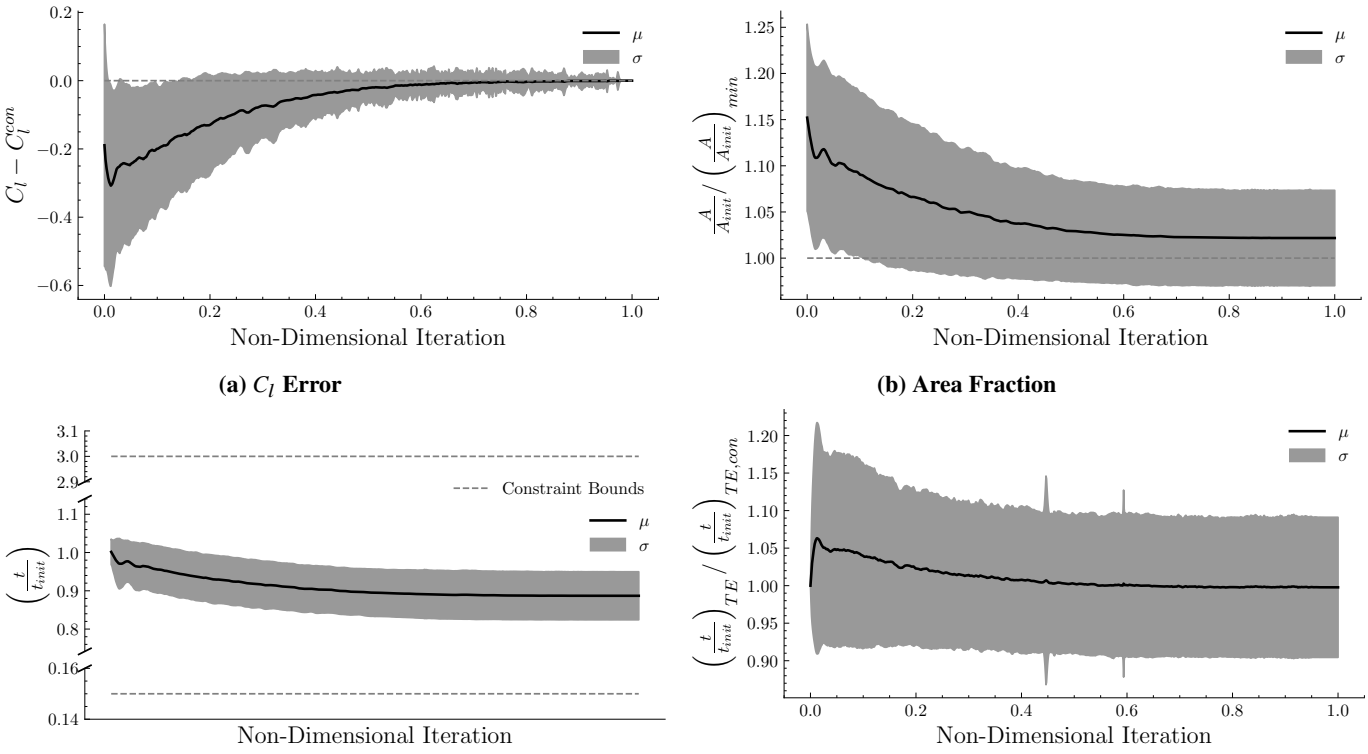
zero due to enforcing symmetry in the FFD point movement (see 11). In this sense, a linear rate of change is reasonable, as we are not measuring any form of cumulative statistic.



(a) Angle of Attack Step Size Convergence **(b) FFD Point Sum Step Size Convergence**

Fig. 9 Aggregate Design Variable Step Size Convergence

Next, we present the aggregate constraint satisfaction measures for C_l, minimum area fraction, thickness constraints, and blunted thickness constraints in Figure 10.

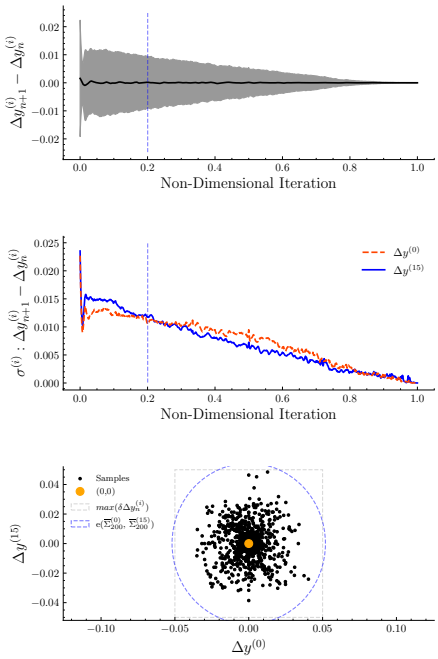


(a) C_l Error **(b) Area Fraction**

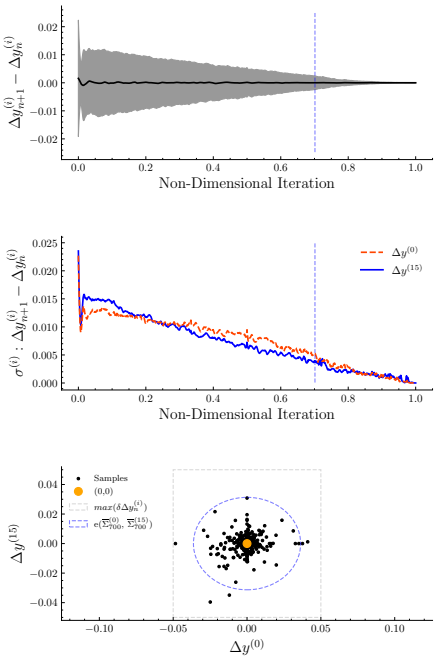
(c) Thickness Fraction **(d) Blunted Thickness Fraction**

Fig. 10 Aggregate Constraint Convergence

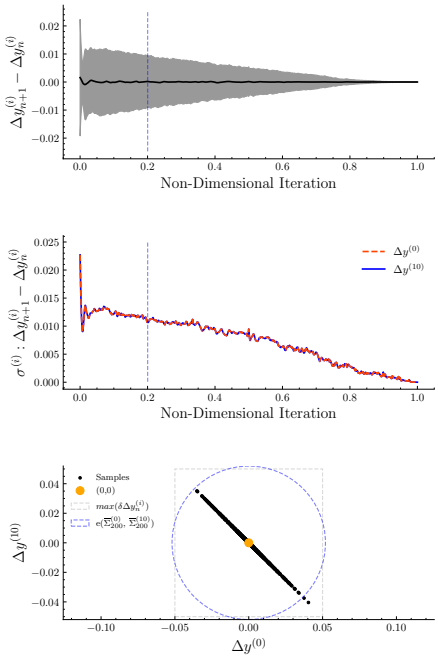
All constraints appear to decrease smoothly in the aggregate, and no abnormal behavior was observed. Additionally, we observed the thickness constraints were never violated in any of the optimization cases we ran. We also investigated how individual FFD points change per iteration, and whether they followed any sort of statistical pattern. We plotted the per iteration change in each FFD point against each other interactively as a function of optimization iteration. A static version of these results is presented in Figure 11.



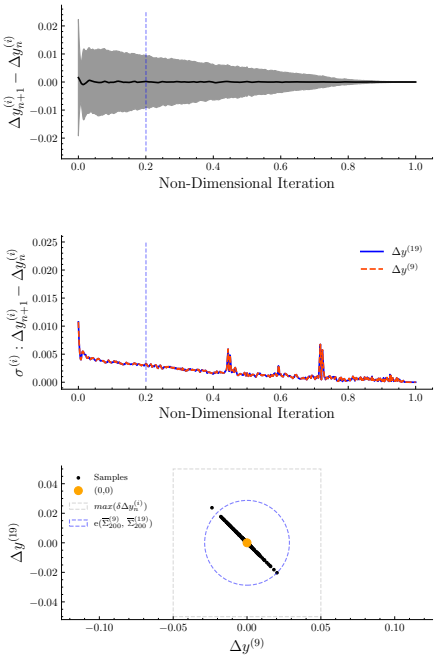
(a) First and 11th Points; 20% Along Trajectory



(b) First and 10th Points; 70% Along Trajectory



(c) Leading Edge Points; 20% Along Trajectory



(d) Trailing Edge Points; 20% Along Trajectory

Fig. 11 Per-FFD Point Variation as a Function of Iteration

A significant takeaway from Figure 11 is that the relative change in FFD points appears to follow a relatively Gaussian-shaped distribution with a linearly decreasing variance and a mean centered around zero. The difference between plots (a) and (b) in Figure 11 demonstrates how this Gaussian linearly shrinks as a function of trajectory. Finally, for validation purposes, we plotted the pairs of leading and trailing edge points (Figure 11 (c), (d)) which, as expected, follow a direct linear relationship, verifying the no-twist conditions we imposed.

Lastly, for reference, we present in more detail the standard deviations of the point changes for all 20 individual FFD points in Figure 12.

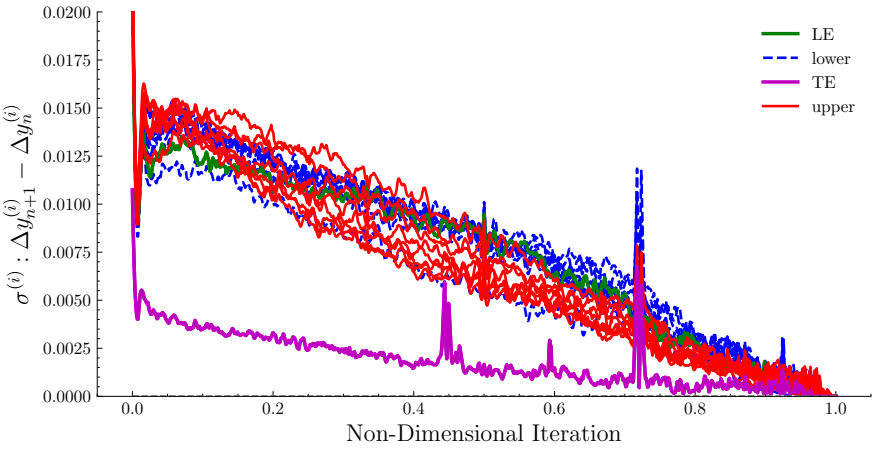


Fig. 12 Standard Deviations in Variation for All 20 FFD Points

Based on these results, it appears the upper surface experiences slightly less variation in FFD point position than the lower surface—however, this difference is not major. We also note how variation in the trailing edge is extremely limited due to the blunted trailing edge thickness constraint we imposed.

IV. Model Implementation Methodology

A. Architecture

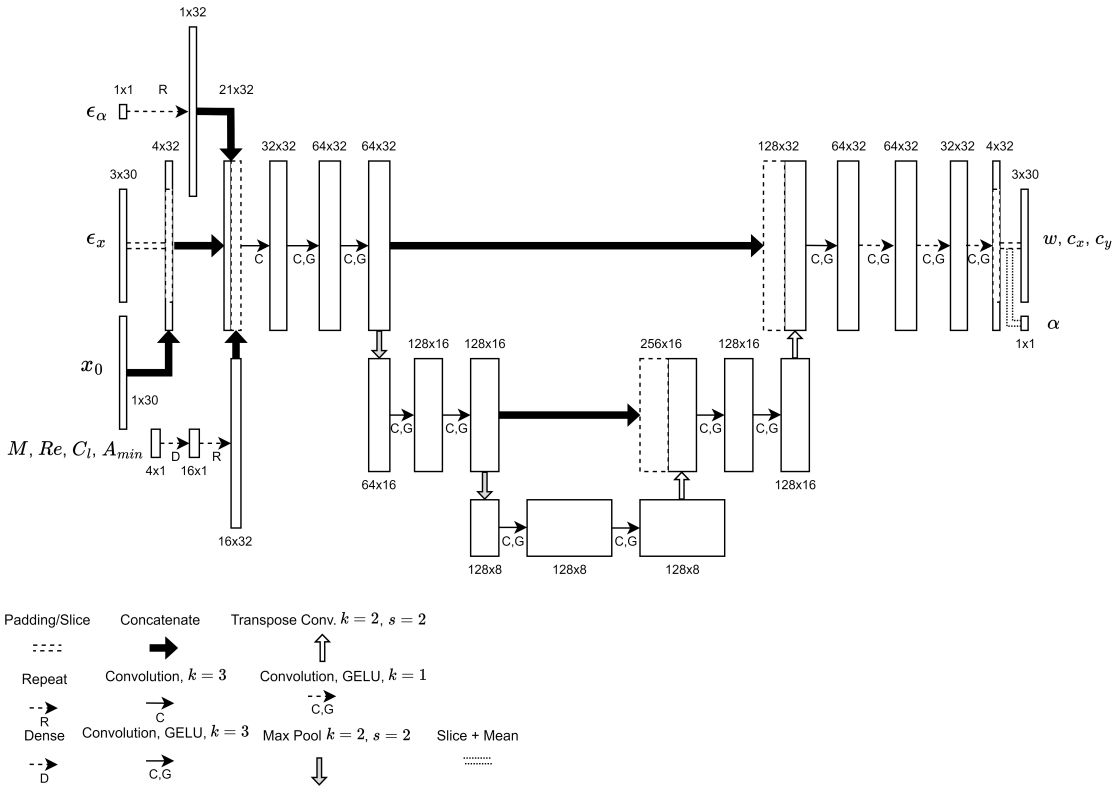


Fig. 13 U-Net Architecture

Both models implemented take on a standard U-Net architecture, which is commonly used in DDM implementations [17]. Our implementation differs in a few key aspects from the more standard image models. Most importantly, the model was trained in a latent space representing Bezier curve control points and weights (c_x, c_y, w), rather than in the raw coordinate space, following Chen *et al.* [8]. This latent space was achieved by pre-training an autoencoder on the training data. In this case, the training data also included all of the airfoils on the optimization trajectories. We use 32 Bezier control points, with 2 of the 32 control points being fixed at the trailing edge. Architecturally, 1-D convolutions with periodic padding were used in place of 2-D convolutions (which are better suited to pixel representations) to match the airfoil coordinate space. Conditioning on the flow parameters, (M, Re, C_l), the minimum area constraint, (A_{min}), and the initial airfoil design (x_0) was handled by a simple concatenation onto the latent space. Additionally, the scalar noise variable for angle of attack, (ϵ_α), was also repeated across the data dimension and concatenated in the same way. The model used was also relatively small, with less than 500,000 trainable parameters, not including the pre-trained auto-encoder. Training wall time is about 10 minutes on an NVIDIA A1000 GPU, and sampling time was about 7s for the entire training set. This computational efficiency in sampling is on par with other generative models, such as GANs, implemented for similar generative airfoil design problems, but much better than these models in terms of training time required [11].

B. Diffusion Parameters and Scheduling

For the baseline model, we adopted a linear variance (β) schedule for the latent Bezier points ranging from 1E-4 to 2E-2, with a total of 1000 steps where the 1st step represents the lowest amount of noise, and 1000 is the highest. This range was proposed by Ho *et al.* in their original diffusion paper [17]. A sample noised by the forward diffusion schedule for the baseline model is shown in Figure 14. Note we show latent Bezier control points (colored by weight) in the top row and the raw airfoil coordinates in the bottom row.

For α we chose a different schedule range of 1E-4 to 1E-1, as we determined higher levels of noise earlier in the schedule were necessary to avoid over-fitting, wherein the model would produce biased α values closer to the statistical mean of the dataset. This is possibly because early on in the schedule, α has little correlation to the actual airfoil shape, and so the loss can be minimized by simply predicting the statistical mean α value if not noised sufficiently—although further investigation may be required to fully validate this hypothesis.

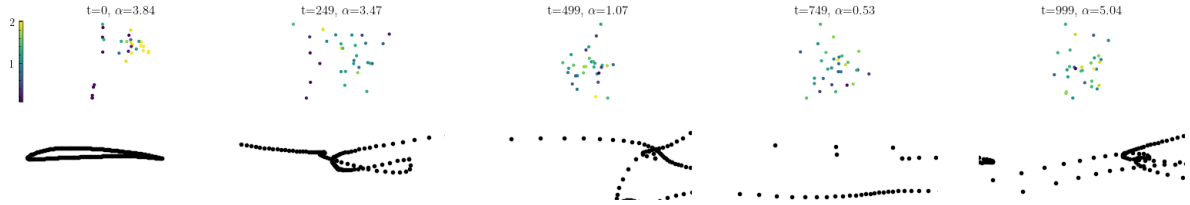
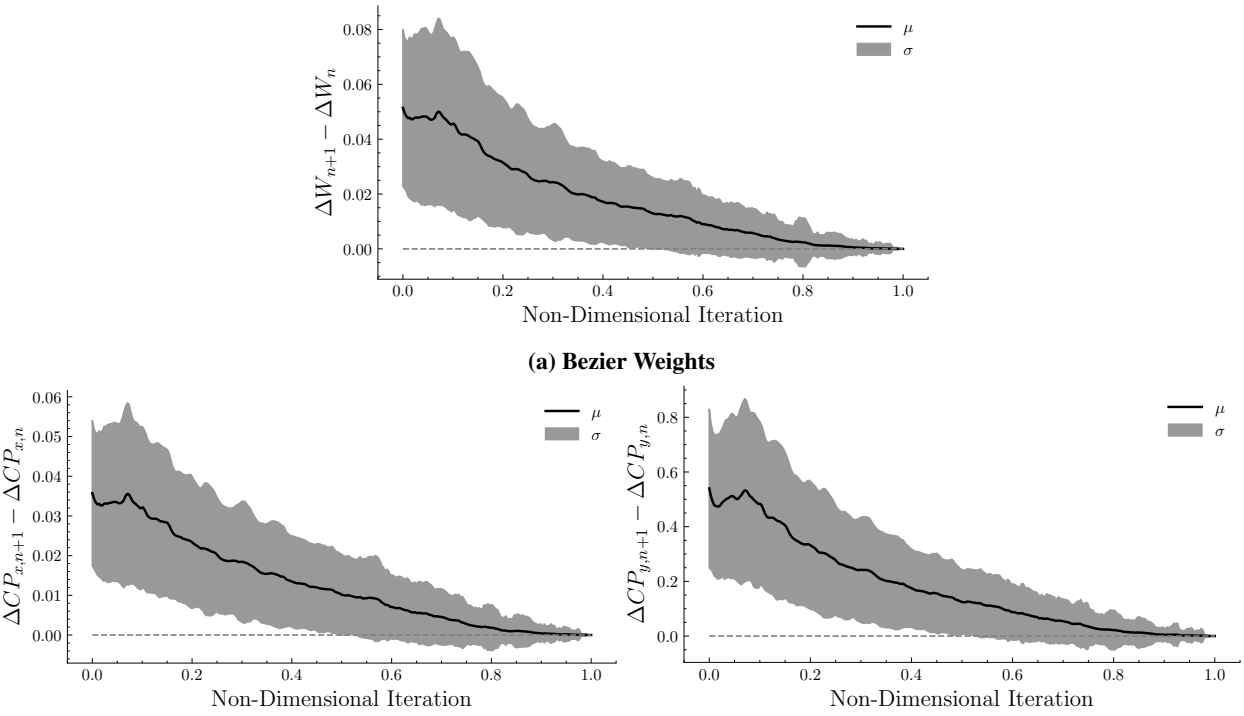


Fig. 14 Baseline Forward Diffusion Schedule

C. Variance Schedule Modeling

The second model we implemented was inspired by the similarity between the variance schedule parameter, β, and the variance in FFD point movement we observed during our statistical analysis of the data. We hypothesized that by mimicking the relative movement of each FFD point, we would be able to attain a more stable diffusion process with better constraint satisfaction and superior performance. We also note that the general idea of modifying diffusion models to mimic optimization processes was attempted in parallel by Giannone *et al.* in, at the time of writing, unpublished work [27]. In their work, the trajectory is aligned by enforcing a penalty term in the loss function that punishes deviation from the corresponding design on the optimization trajectory. In that work, corresponding designs were spaced linearly to line up with the diffusion schedule itself, and no attempt was made to match the actual variance of the design trajectory, as we have done in this work.

In our work, we attempt to take a different approach by modeling individual variance schedules for each design variable based on their statistical variation in the optimization trajectory. To do this we first transformed the airfoil coordinate variation into the latent Bezier space with the trained autoencoder. The statistical optimization trajectory, analogous to Figure 9b in Bezier space is presented in Figure 15.



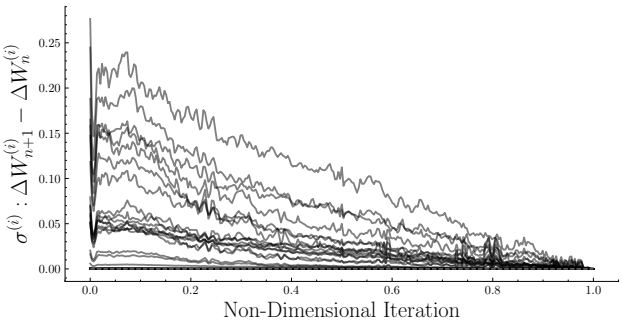
(a) Bezier Weights

(b) X Control Points

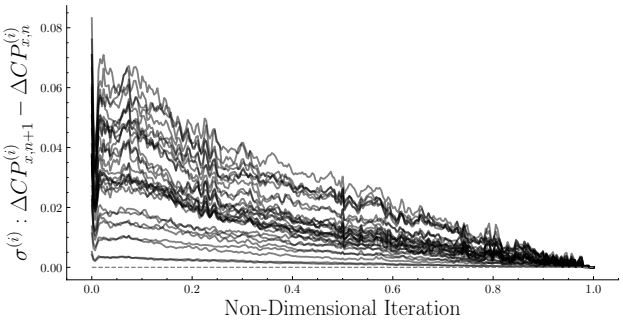
(c) Y Control Points

Fig. 15 Aggregate Bezier Point Step Size Convergence

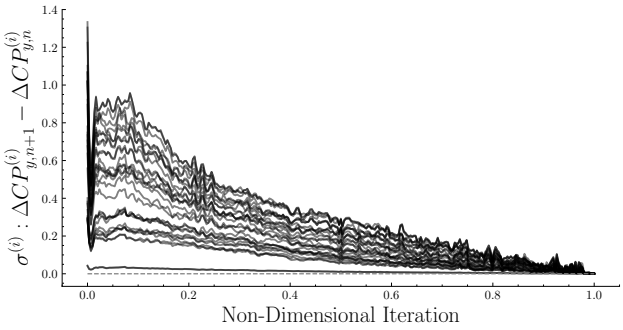
Next, we present the individual variation in each control point and weight in Figure 16. To implement our idea we wish to simply replace each element of the variance matrix with the corresponding statistically modeled variance on the optimization trajectory, such that $\beta_i = \sigma(X_i)$. Where X_i is a placeholder for each Bezier variable W_i, $CP_{x,i}$, and $CP_{y,i}$.



(a) Bezier Weights



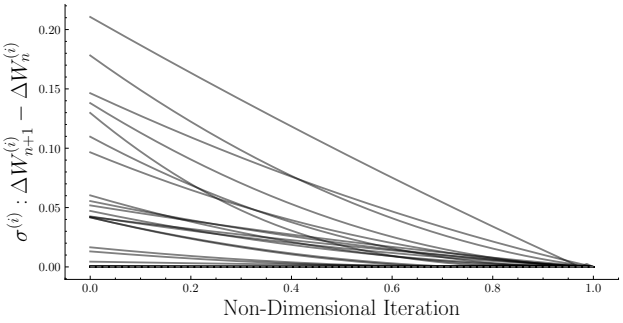
(b) X Control Points



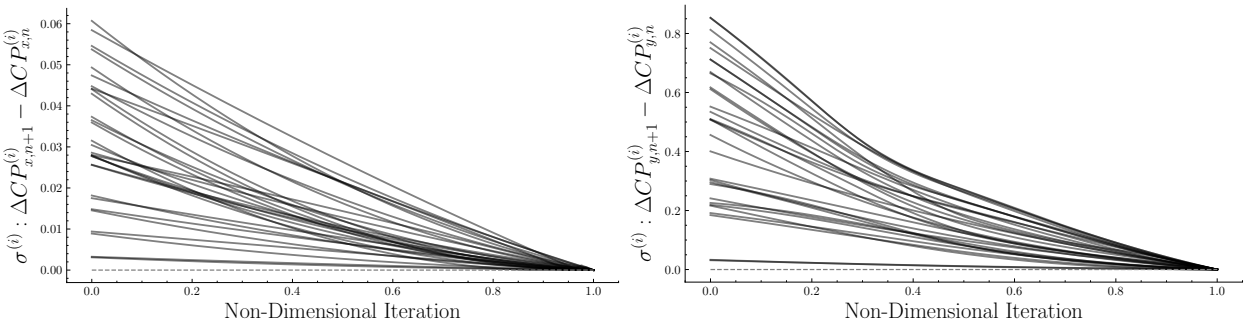
(c) Y Control Points

Fig. 16 Aggregate Per-Iteration Deviation in Bezier Points

Since the statistical data is noisy, we apply smoothing and cut off the first 10% of the trajectory, replacing the cut-off data with a line from the maximum variance to the cut-off point. The results of this processing are displayed in Figure 17. This final processed version of the data is used directly as the variance schedule for each design variable. Note we do not attempt to model the variance of the angle of attack (α) in this work.



(a) Bezier Weights



(b) X Control Points **(c) Y Control Points**

Fig. 17 Processed Aggregate Per-Iteration Deviation in Bezier Points

Finally, the last additional change required is to ensure that during sampling, the input to the forward diffusion schedule has a similar distribution as the input to the optimizer, since starting from pure noise would produce completely infeasible designs, owing to the wide variation in the starting points and scaling of the modeled variance schedule. This problem is simply addressed by using the initial design as the initial input, in place of pure random noise, to the diffusion model during sampling.

Putting all of these modifications in place leads to a forward variance schedule that is visually much more stable in the decoded coordinate space than a traditional, linear variance schedule. This is illustrated in Figure 18. As a note, we did attempt to use the initial airfoil with the baseline model as well, but the results were essentially unchanged, with the first few steps quickly destroying the structure of the initial design, which was expected since the baseline model was trained to start from inputs much closer to pure noise.

Fig. 18 Modeled Variance Forward Diffusion Schedule

V. Model Implementation Results

A. Reverse Diffusion Process

As a point of reference, we present the reverse diffusion processes, in Figures 19, and 20, as learned by each model, noting that they mimic their respective forward schedules as expected.

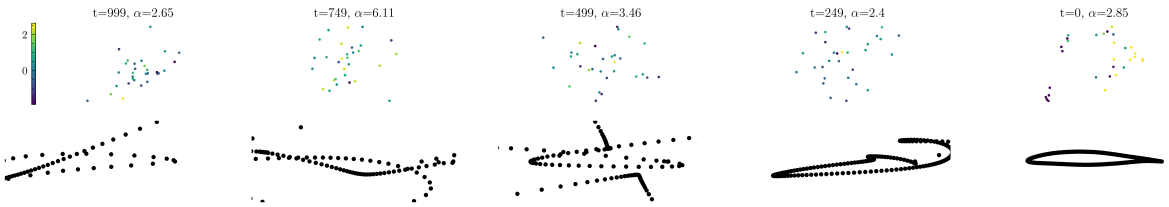


Fig. 19 Baseline Reverse Diffusion Process

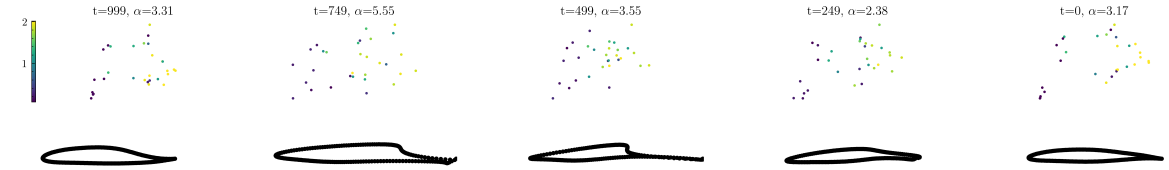
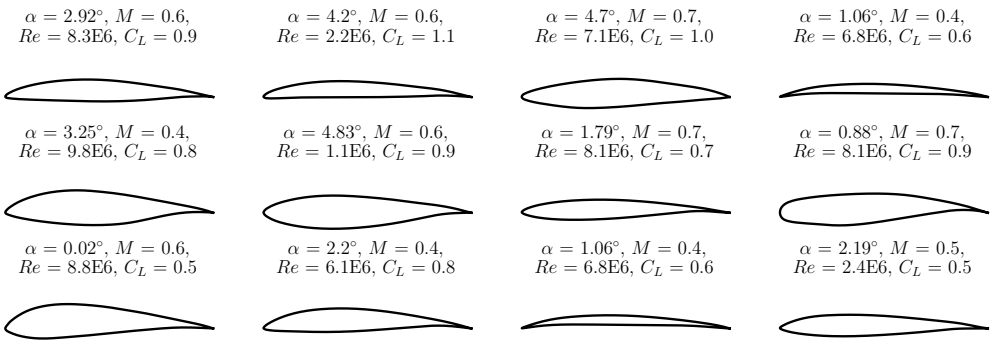


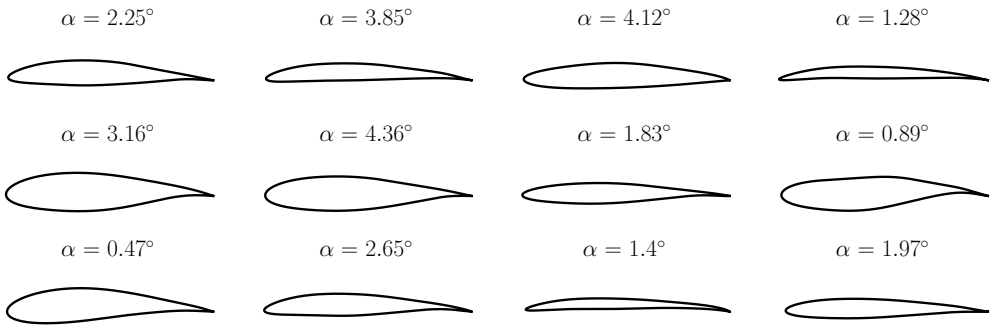
Fig. 20 Modeled Variance Reverse Diffusion Process

B. Sample Generations

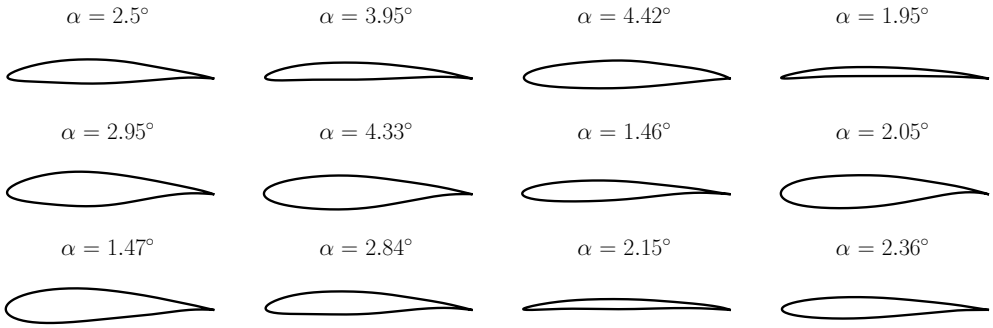
Input parameters from the held-out validation set were used to statistically validate generations produced by each model. A grid of sample generations for the baseline and modeled variance approaches is presented in Figure 21, along with the corresponding ground truth validation data.



(a) Validation Data



(b) Generated, Baseline



(c) Generated, Modeled Variance

Fig. 21 Sample Generations

We observed that both models produced relatively accurate generations that appear to match the unseen validation data visually. However, it is difficult to determine the difference in sample quality between the two models themselves, owing to how minor the changes are. It does appear that the baseline model produces a much more accurate angle of attack generations than the modeled variance approach does. This may be because the angle of attack variance schedule, which was the same for both approaches, was somehow overfitted to the training data, but the reason as to why this is the case remains an open question for future investigation.

C. Validation

To measure model performance, we used two key statistical metrics; MSE, and maximum mean discrepancy (MMD), the latter of which was averaged over 10 passes through the validation dataset. In addition, to measure area constraint satisfaction, we calculate and sum the MSE of any generated airfoils with areas less than the required minimum area

constraint. The results are collated in Table 3.

Table 3 Validation Data Metrics

	MMD (Airfoil)	MMD (α)	MSE (Airfoil)	MSE (α)	Area Constraint MSE
Baseline	0.0755 ± 0.00094	**0.0740 ± 0.0031**	4.45E-5	**0.152**	2.06E-5
Modeled Variance	**0.0692 ± 0.0013**	0.112 ± 0.0066	**2.48E-5**	0.349	**3.87E-6**

In general, the modeled variance approach appears to produce superior-quality airfoil shapes that better match the real data distribution—however, we are cautious as to interpreting this improvement to be a result of our implementation, as the difference is relatively slight. Even so, these results are not insignificant and do justify further investigation. At the same time, the modeled variance approach also appears to produce worse α generations, which was also evident from the visual comparisons. A relatively more significant result is the order of magnitude improvement in area constraint satisfaction, which could indicate the modeled variance approach produces more feasible designs due to the relative stability in airfoil shape in the modeled variance schedule. We note similar claims of increased feasibility in the conclusions from the only other known optimization trajectory mimicking approach in literature [27].

Finally, we benchmarked the generated designs using the same ADflow CFD solver and meshing procedure used to create the dataset. These results are collated in Table 4. We note the mean C_d of the original adjoint-optimized test and validation set is 123.5 and 122.3 counts respectively. Likewise, the mean C_l error was less than 1E-8 for both the testing and validation sets.

Table 4 CFD Benchmarks

	Test		Validation	
	Mean C_l Error	Mean C_d (Counts)	Mean C_l Error	Mean C_d (Counts)
Baseline	0.012	138.5	0.011	130.5
Modeled Variance	**0.005**	**137.5**	**0.005**	**122.9**

Based on these results it appears that the modeled variance approach demonstrates superior performance across all metrics, with particularly good accuracy on the validation set, almost approaching the performance of the original optimized validation set in drag counts. These differences require further validation, as the test set performance is about on par with the baseline model. Even so, the mean C_l constraint error is consistently more than half that of the baseline model, lending more credence to the idea that mimicking the optimization trajectory may help with constraint satisfaction and therefore design feasibility. Additionally, we leave a more rigorous analysis of the use of these generations to warm-start adjoint optimizations as future work.

VI. Conclusion

In this paper, we have presented several key contributions which will help inform the current use, and future study, of generative models for aerodynamic design purposes. First, we have presented a novel dataset of 935 optimized airfoils obtained from a diversely sampled design space. Based on statistical analysis of this dataset, we have determined that changes in airfoil design parameters follow an approximately Gaussian distribution. Next, we demonstrated success using a standard denoising diffusion model with a custom U-Net architecture and simple conditioning on the initial airfoil design and flow parameters. Finally, we explored a modification to the diffusion model wherein each design variable is noised according to a schedule based on the design variable's statistical change in the optimization trajectory. We demonstrated this approach can produce airfoils that better match the data distribution, and additionally perform marginally better when benchmarked in a CFD solver. Our strongest observation was a drastic increase in constraint satisfaction of anywhere between 50% for C_l satisfaction, and up to an order of magnitude for area constraint satisfaction. We acknowledge the magnitude and reason behind this improvement is an open question, but we believe our initial exploration provides sufficient evidence to merit more rigorous future study of optimization-informed diffusion models.

Acknowledgments

We acknowledge the support from the National Science Foundation through award #1943699. The authors acknowledge the University of Maryland supercomputing resources (http://hpcc.umd.edu) made available for conducting the research reported in this paper. We would also like to acknowledge help from the University of Michigan Multidisciplinary Design Optimization (MDO) Lab member Alasdair C. Gray with using the MACH-Aero framework in our work.

References

[1] He, X., Li, J., Mader, C. A., Yildirim, A., and Martins, J. R. R. A., "Robust aerodynamic shape optimization—from a circle to an airfoil," *Aerospace Science and Technology*, Vol. 87, 2019, pp. 48–61. https://doi.org/10.1016/j.ast.2019.01.051.

[2] Li, J., Bouhlel, M. A., and Martins, J. R. R. A., "Data-Based Approach for Fast Airfoil Analysis and Optimization," *AIAA Journal*, Vol. 57, No. 2, 2019, pp. 581–596. https://doi.org/10.2514/1.J057129, URL https://doi.org/10.2514/1.J057129.

[3] Yonekura, K., and Suzuki, K., "Data-driven design exploration method using conditional variational autoencoder for airfoil design," *Structural and Multidisciplinary Optimization*, Vol. 64, No. 2, 2021, pp. 613–624.

[4] Dhariwal, P., and Nichol, A., "Diffusion models beat gans on image synthesis," *Advances in Neural Information Processing Systems*, Vol. 34, 2021, pp. 8780–8794.

[5] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M., "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

[6] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B., "High-resolution image synthesis with latent diffusion models," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10684–10695.

[7] Panchal, J. H., Fuge, M., Liu, Y., Missoum, S., and Tucker, C., "Special Issue: Machine Learning for Engineering Design," *Journal of Mechanical Design*, Vol. 141, No. 11, 2019. https://doi.org/10.1115/1.4044690, URL https://doi.org/10.1115/1.4044690, 110301.

[8] Chen, W., Chiu, K., and Fuge, M. D., "Airfoil Design Parameterization and Optimization Using Bézier Generative Adversarial Networks," *AIAA Journal*, Vol. 58, No. 11, 2020, pp. 4723–4735. https://doi.org/10.2514/1.J059317, URL https://doi.org/10.2514/1.J059317.

[9] Regenwetter, L., Nobari, A. H., and Ahmed, F., "Deep Generative Models in Engineering Design: A Review," *Journal of Mechanical Design*, Vol. 144, No. 7, 2022. https://doi.org/10.1115/1.4053859, URL https://doi.org/10.1115/1.4053859, 071704.

[10] Mazé, F., and Ahmed, F., "Diffusion Models Beat GANs on Topology Optimization," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[11] Chen, Q., Wang, J., Pope, P., (Wayne) Chen, W., and Fuge, M., "Inverse Design of Two-Dimensional Airfoils Using Conditional Generative Models and Surrogate Log-Likelihoods," *Journal of Mechanical Design*, Vol. 144, No. 2, 2021. https://doi.org/10.1115/1.4052846, URL https://doi.org/10.1115/1.4052846, 021712.

[12] Ghosh, S., Padmanabha, G. A., Peng, C., Atkinson, S., Andreoli, V., Pandita, P., Vandeputte, T., Zabaras, N., Wang, L., and Khan, G., "Pro-ml ideas: A probabilistic framework for explicit inverse design using invertible neural network," *AIAA Scitech 2021 Forum*, 2021, p. 0465.

[13] Du, X., He, P., and Martins, J. R., "A B-spline-based generative adversarial network model for fast interactive airfoil aerodynamic optimization," *AIAA Scitech 2020 Forum*, 2020, p. 2128.

[14] Achour, G., Sung, W. J., Pinon-Fischer, O. J., and Mavris, D. N., "Development of a conditional generative adversarial network for airfoil shape optimization," *AIAA Scitech 2020 Forum*, 2020, p. 2261.

[15] Morton, J., Kochenderfer, M. J., and Witherden, F. D., "Parameter-Conditioned Sequential Generative Modeling of Fluid Flows," *AIAA Journal*, Vol. 59, No. 3, 2021, pp. 825–841. https://doi.org/10.2514/1.J059315, URL https://doi.org/10.2514/1.J059315.

[16] Luo, C., "Understanding diffusion models: A unified perspective," *arXiv preprint arXiv:2208.11970*, 2022.

[17] Ho, J., Jain, A., and Abbeel, P., "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, Vol. 33, 2020, pp. 6840–6851.

[18] Michael Selig, A. A. G., University of Illinois Urbana-Champaign, "UIUC Dataset," , 2023. Data retrieved from , https://m-selig.ae.illinois.edu/ads/coord_database.html.

[19] Secco, N., Kenway, G. K. W., He, P., Mader, C. A., and Martins, J. R. R. A., "Efficient Mesh Generation and Deformation for Aerodynamic Shape Optimization," *AIAA Journal*, 2021. https://doi.org/10.2514/1.J059491.

[20] Wu, N., Kenway, G., Mader, C. A., Jasa, J., and Martins, J. R. R. A., "pyOptSparse: A Python framework for large-scale constrained nonlinear optimization of sparse systems," *Journal of Open Source Software*, Vol. 5, No. 54, 2020, p. 2564. https://doi.org/10.21105/joss.02564.

[21] Hajdik, H. M., Yildirim, A., Wu, N., Brelje, B. J., Seraj, S., Mangano, M., Anibal, J. L., Jonsson, E., Adler, E. J., Mader, C. A., Kenway, G. K. W., and Martins, J. R. R. A., "pyGeo: A geometry package for multidisciplinary design optimization," *Journal of Open Source Software*, Vol. 8, No. 87, 2023, p. 5319. https://doi.org/10.21105/joss.05319.

[22] Secco, N., Kenway, G. K. W., He, P., Mader, C. A., and Martins, J. R. R. A., "Efficient Mesh Generation and Deformation for Aerodynamic Shape Optimization," *AIAA Journal*, 2021. https://doi.org/10.2514/1.J059491.

[23] Mader, C. A., Kenway, G. K. W., Yildirim, A., and Martins, J. R. R. A., "ADflow—An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization," *Journal of Aerospace Information Systems*, 2020. https://doi.org/10.2514/1.I010796.

[24] Kenway, G. K. W., Mader, C. A., He, P., and Martins, J. R. R. A., "Effective Adjoint Approaches for Computational Fluid Dynamics," *Progress in Aerospace Sciences*, Vol. 110, 2019, p. 100542. https://doi.org/10.1016/j.paerosci.2019.05.002.

[25] Yildirim, A., Kenway, G. K., Mader, C. A., and Martins, J. R., "A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations," *Journal of Computational Physics*, Vol. 397, 2019, p. 108741.

[26] El-Gohary, M., and Ahmed, M., "Investigating the design parameters of a diamond-shaped supersonic airfoil," *International Conference on Aerospace Sciences and Aviation Technology*, The Military Technical College, 2013, pp. 1–18.

[27] Giannone, G., Srivastava, A., Winther, O., and Ahmed, F., "Aligning Optimization Trajectories with Diffusion Models for Constrained Design Generation," *arXiv preprint arXiv:2305.18470*, 2023.