



Computational Science and Engineering  
(International Master's Program)

Technische Universität München

Master's Thesis

**Black-Box Optimization for Engineering  
Systems with Score-Function Estimator**

Mohammad Anas Khan







# Computational Science and Engineering (International Master's Program)

Technische Universität München

Master's Thesis

## Black-Box Optimization for Engineering Systems with Score-Function Estimator

Author: Mohammad Anas Khan  
1<sup>st</sup> examiner: Prof. Dr. Hans-Joachim Bungartz  
2<sup>nd</sup> examiner: Prof. Dr. Faidon-Stelios Koutsourelakis  
1<sup>st</sup> Supervisor : M.Sc. Kislaya Ravi  
2<sup>nd</sup> Supervisor: M.Sc. Atul Agrawal  
Submission Date: July 19th,2024





I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

A handwritten signature in black ink, appearing to read 'Anas', with a horizontal line underneath it.

July 19th, 2024

Mohammad Anas Khan



---

## Acknowledgments

First and foremost, I would like to express my deepest gratitude to my thesis supervisors, M.Sc. Kislaya Ravi and M.Sc. Atul Agrawal, for their unwavering support, invaluable guidance, and constant encouragement throughout the duration of my research. Their profound knowledge and expertise have been instrumental in the successful completion of this thesis. Their patience, insight, and commitment to excellence have inspired me to strive for the best in my work.

I am also immensely grateful to my parents, whose unconditional love, patience, and understanding have been my greatest source of strength and motivation. Without their sacrifices and encouragement, this achievement would not have been possible.

Thank you all for your invaluable contributions.

---



---

## Abstract

Optimization of engineering systems is essential for efficient operation and enhanced lifespan of these systems. Owing to their complex designs, multiple design parameters, and their operation under stochastic conditions, optimization of such systems becomes difficult. The absence of a well-defined mathematical relation between the input and the output of such systems prevents the conventional gradient based methods from providing meaningful solutions. To optimize such problems, we use black-box optimization techniques. Multiple black-box optimization techniques have been developed, which have been shown to perform well for low and medium dimensional problems. ScoutND, a black-box optimization algorithm is shown to perform well for high dimensional problems as well as stochastic problems. The present work extends ScoutND to perform shape optimization of the PitzDaily problem, a test case problem in Computational Fluid Dynamics. We propose methods to perform Constrained and Unconstrained optimization of the PitzDaily problem. We perform optimization under stochastic and non-stochastic conditions. We compare the results of our shape optimization problem from ScoutND with other optimizers such as Nelder-Mead, SLSQP, COBYLA, and L-BFGS-B. We show that ScoutND successfully optimizes the problem for stochastic and non-stochastic conditions and outperforms all other optimizers in the case of stochastic conditions.



# Contents

<b>Acknowledgements</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>I. Introduction and Background Theory</b>	<b>1</b>
1. Introduction	3
2. Literature Review	5
<b>II. Method and Theory</b>	<b>9</b>
<b>3. ScoutND</b>	<b>11</b>
3.1. Monte Carlo Method . . . . .	13
3.2. Gradient Estimation . . . . .	13
3.3. Adam Optimizer . . . . .	15
3.4. Natural Gradients and Fisher Information Matrix . . . . .	16
3.5. Problem Definition . . . . .	16
3.6. Variance Reduction . . . . .	18
3.7. ScoutND Algorithm . . . . .	18
<b>4. PitzDaily</b>	<b>21</b>
4.1. CFD Simulation of PitzDaily . . . . .	22
4.1.1. SIMPLE Algorithm . . . . .	23
4.1.2. Turbulence Model . . . . .	26
4.1.3. Geomtric Algebraic Multi-Grid (GAMG) . . . . .	28
4.2. OpenFOAM Simulation . . . . .	30
4.3. Catmull-Clark Subdivision Curves . . . . .	32
4.4. Objective Function . . . . .	35
4.5. Constrained Optimization . . . . .	35
4.6. Normalization of Input . . . . .	36
4.7. Unconstrained Optimization . . . . .	37
4.8. Normalization of Output . . . . .	38
4.9. Stochasticity in Fluid flows . . . . .	38

4.10. Objective Function Evaluation . . . . .	38
<b>III. Results and Conclusion</b>	<b>41</b>
<b>5. Results</b>	<b>43</b>
5.1. Non-stochastic constrained optimization using ScoutND . . . . .	43
5.2. Stochastic constrained optimization using ScoutND . . . . .	51
5.3. Non-stochastic unconstrained optimization using ScoutND . . . . .	58
5.4. Stochastic unconstrained optimization using ScoutND . . . . .	63
5.5. Comparative Analysis . . . . .	67
<b>6. Conclusion</b>	<b>73</b>
<b>Bibliography</b>	<b>81</b>

**Part I.**

**Introduction and Literature Review**



# 1. Introduction

The use of engineering systems can be traced back to ancient civilizations where human beings developed tools to accomplish difficult tasks with ease. Over the years, the designing process has seen multiple phases, from designing new systems to improving the existing ones. The last century has seen rapid growth in the development and design of engineering systems. Modern engineering systems are often accompanied by complex design and multiple design parameters. Optimization of these design parameters becomes essential for efficient operation of the system. Therefore, optimization has become a topic of keen interest among engineers. Multiple optimization techniques have been developed to optimize mathematical problems, which have been extended to engineering systems as well. Structures, automotive systems, electronic circuits, transportation networks, power grids, and fluid systems are typical cases of engineering systems where optimization of design parameters is essential. With advent of computational techniques, engineering systems today can be digitally designed and simulated before they are prototyped. This makes the optimization process much more economical. There has been extensive research in developing softwares to simulate engineering systems. The present work deals with optimizing a fluid flow problem, the PitzDaily problem. Fluid flows are governed by Navier-Stokes equations, which are non-linear, coupled, partial differential equations [1]. The Navier-Stokes equation cannot be solved analytically. Computational Fluid Dynamics(CFD) is a numerical approach to simulating fluid flow and heat transfer problems in engineering systems. Multiple CFD simulation methods have been proposed which solve the Navier-Stokes equation. The solution of CFD problems is an iterative process that consumes significant time and computational power. Complex problems may run for hours or even a few days before providing some meaningful results. This becomes a major bottleneck in the optimization of such systems as optimization is also an iterative process that requires multiple evaluations of the objective function. Another issue with CFD problems is the absence of gradient information. Given these constraints, we make use of Black-Box optimization or Derivative-Free optimization [2] techniques to optimize CFD problems. Some CFD problems have been extensively modeled, tested, and validated over the years. The PitzDaily problem is a popular test case problem in Computational Fluid Dynamics. The aim of the current work is to minimize the objective function defined for our PitzDaily problem using ScoutND, a black-box optimization algorithm.





## 2. Literature Review

Optimization procedures can be traced back to the works of early mathematicians who sought to optimize functions. Its formal application began to emerge in the 20th century for industrial problems. Multiple optimization techniques have been developed in the past century. Optimization problems can be classified into different categories, some of which are, Constrained and Unconstrained optimization problems, Gradient based and Derivative-free optimization problems, Linear and non-Linear problems, Deterministic and Stochastic Optimization problems. Many optimization problems rely on gradients to calculate the minimum. An analogy for these methods explained by Vanderplaats [3] is a boy standing between two fences on a hillside, wearing a blindfold. To reach the top, the boy moves in the direction of maximum steepness of the terrain using fixed steps. At the end, the boy ends up on a higher terrain, where the two fences intersect, and cannot move further. His final position represents the local maxima, the hilltop being the global maxima. A problem with these methods is that the gradient cannot always be analytically calculated. Therefore, we require numerical methods to estimate the gradient. Many algorithms have been developed to deal with such problems, some of which will be discussed here.

The most basic of the gradient based optimization methods is the Gradient Descent method. Gradient Descent iteratively calculates the gradient and updates the value based on a defined step size by moving in the direction of steepest descent. It is easy to implement but has slow convergence and may converge locally. Multiple variants of Gradient-Descent have been proposed to counter these issues. Stochastic Gradient Descent (SGD) [4] updates the gradient using a single sample from the dataset. It has fast convergence when dealing with large datasets but is less accurate due to noise in gradient estimation. A trade-off between SGD and Gradient Descent is the Mini-batch Gradient Descent which works by using a small batch of samples to calculate the gradient [5]. It has faster convergence than Gradient Descent and better accuracy than SGD. Momentum techniques [6, 7] were introduced to accelerate the convergence of Gradient Descent methods. Momentum techniques keep a history of previously calculated gradients and assign a weight to the current and previously calculated gradients to calculate the momentum, which is used to accelerate convergence. Some common optimizers based on momentum techniques are Adagrad [8] and Adam [9]. Newton's method is a popular iterative optimization method [10] which uses the Hessian matrix to estimate the search direction and the updated value. Newton's method has fast, quadratic convergence. An essential condition of Newton's method is that the Hessian matrix should be positive definite. For high-dimensional problems, computing the inverse of the Hessian matrix can be expensive. Quasi-Newton's

methods address the limitations of Newton's methods. In Quasi-Newton's methods, the Hessian matrix is calculated at each iteration using information from previously calculated gradient. There are multiple variants of Quasi-Newton's methods, which mainly differ in the method used to update the Hessian matrix. Broyden-Fletcher-Goldfarb-Shanno(BFGS) [11, 12, 13, 14] is a popular Quasi-Newton method, named after its founders, which approximates the inverse of the Hessian matrix. Although BFGS is more computationally efficient than Newton's method, it still needs to store the matrix, which is memory intensive. An improvement to the BFGS method is the Limited memory-BFGS (L-BFGS) method [15]. Instead of storing the approximation of the inverse of the Hessian matrix, L-BFGS stores vectors to obtain the inverse of the Hessian matrix. These methods are generally employed for non-constrained, non-bounded problems. Limited-memory BFGS with Bounds (L-BFGS-B) is an extension to L-BFGS which is capable of implementing bounds to the optimization problem [16]. The Sequential Quadratic Programming (SQP) [17] method is a well-known technique used to solve constrained optimization problems. SQP method works by approximating the objective function into a quadratic problem and the constraints into a linear problem, which is solved iteratively by forming a Lagrangian, to obtain the updated value. Dieter and Kraft [18] introduced Sequential Least-Square Sequential Quadratic Programming (SLSQP), an extension of the SQP method which solves a Least-Squares problem instead of a Lagrangian. SQP and SLSQP can handle equality as well as inequality constraints.

In practical scenarios, the objective function is often not differentiable, or the gradient information is unavailable. Derivative-free optimization techniques are algorithms that do not rely on gradients for optimization. Genetic Algorithm (GA) [19] is a derivative-free optimization technique based on the natural selection principle. GA works by initializing a set of population, which is used to calculate the objective function. Crossover and Mutation operators are applied across generations to select the best individual. GA requires multiple evaluations of the objective function in a single generation, which makes it computationally expensive for real-life problems. Bayesian Optimization [20] is a derivative-free optimization method useful in scenarios where the objective function is expensive to calculate. It works by constructing a surrogate model to approximate the objective function. An acquisition function is used to obtain the next evaluation points for the optimization process. Dantzig proposed the simplex algorithm [21] to solve linear programming problems. Simplex algorithm works by evaluating the objective function on the vertices of a feasible region until an optimal value is obtained. This feasible region is bounded by constraints. Simplex can handle equality and inequality constraints. Simplex, although, was originally designed for linear problems, there are simplex based methods to optimize non-linear functions. Constraint Optimization By Linear Approximation (COBYLA) [22] is a derivative-free optimization technique used to solve Non-Linear Constrained problems. It is a trust-region based simplex method. It approximates the objective function and constraints into linear problems. COBYLA is not useful in cases of highly non-linear problems. The Nelder-Mead method [23] is a simplex based derivative-free optimization technique to optimize non-linear functions. A disadvantage of Nelder-Mead is that it may converge

---

to a local optimum and does not support constraints. Optimization methods discussed so far either require gradient information, converge locally, or do not perform well in the case of high dimensions[24]. Moreover, engineering systems are often subject to stochastic conditions. The algorithms discussed so far are not effective in optimizing problems under stochastic conditions. We will be implementing ScoutND, a black box optimization algorithm [25] for our optimization problem, which can address these issues.



**Part II.**

**Method and Theory**



### 3. ScoutND

Gradient based optimization methods cannot be used in the case of non-continuous functions, where the gradient cannot be calculated, or in the case of black-box functions, where the gradient is unavailable. The work of Staines and Barber [26] shows that it is possible to define an upper bound to the function, which is smooth and can be made differentiable under weak conditions. The basic idea behind Variational Optimization is based on the bound

$$\min_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x}) \leq \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})}[f(\mathbf{x})], \quad (3.1)$$

where  $\mathbf{x} \in \mathbb{R}^n$  is our input parameter,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is our objective function,  $\mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})}$  is the expectation with respect to the distribution  $p(\mathbf{x}|\boldsymbol{\theta})$  defined over solution space  $\mathcal{C}$  and  $\boldsymbol{\theta}$  is the distribution parameter. This bound becomes tight, as the variance reduces. A bound with a lower value of variance approximates the objective function closely but makes it difficult to find the optimum, whereas a bound with a higher value of variance does not approximate the function closely but its optimum is easier to find. Figure 3.1 shows a non-differentiable function  $f(x) = |x|$  and the approximation to the function using different variance values. From the figure, we observe that lower variance values approximate the function more closely than higher variance values. For optimization problems, a solution to this is to start with a higher variance value and reduce it iteratively.

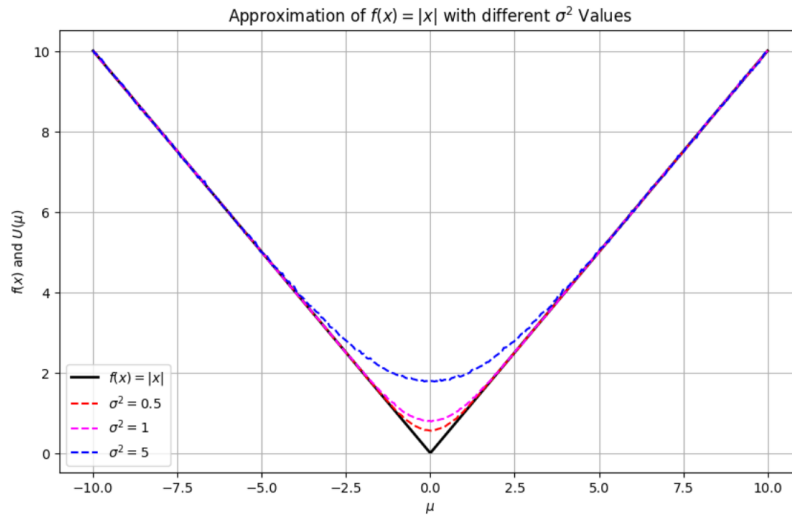


Figure 3.1.: Figure shows  $f(x) = |x|$  and the approximation to the function using different variance values. Lower variance values approximate the function closely as compared to higher variance values.

We defined a bound to our function using Variational Optimization in equation 3.1, which states that, for a set of sampled values, the expectation of function evaluations at these samples is greater than or equal to the minimum of these function evaluations. This expectation acts as a bound to our function. Instead of optimizing the function with respect to the input parameter, we optimize this bound with respect to the distribution parameter. Stochastic Variational Optimization [27] aims to optimize this bound by estimating the gradient of this bound using score-function estimator. ScoutND (Stochastic Constrained Optimization for N dimensions) [25] is a black-box optimization algorithm based on Stochastic Variational Optimization.

In this chapter, we describe the working of the ScoutND Algorithm. We begin this chapter by providing a general theory and the relevant equations of different techniques used in the ScoutND algorithm. Later in this chapter, we specify the relevant equations specifically for ScoutND and highlight the general algorithm of ScoutND. Section 3.1 describes the basics of Monte Carlo methods. In section 3.2, we describe the general method used for gradient estimation in ScoutND. Section 3.3 describes the Adam optimizer used in our optimization problem. Section 3.4 describes the concept of Natural gradients. Section 3.5 describes the problem statement that ScoutND solves. Section 3.6 describes the variance reduction techniques used in ScoutND. Section 3.7 outlines the general Algorithm of ScoutND.



### 3.1. Monte Carlo Method

Monte Carlo simulation [28] is used to estimate the value of an input function using random sampling. Suppose we generate  $N$  random variables having probability distribution  $p(\mathbf{x}|\boldsymbol{\theta})$  where  $\mathbf{x}$  is sampled from the distribution parameters  $\boldsymbol{\theta}$  and let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be a function. The expectation,  $\mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})}[f(\mathbf{x})]$ , is given as:

$$\mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})}[f(\mathbf{x})] = \int_{-\infty}^{\infty} f(x)p(x|\boldsymbol{\theta})dx \quad (3.2)$$

Using Monte Carlo, sampling we can approximate Equation (3.2) as

$$\hat{\mathbb{E}}_{p(\mathbf{x}|\boldsymbol{\theta})}[f(\mathbf{x})] = \frac{1}{N} \sum_{i=1}^N f(x_i), \quad (3.3)$$

where  $x_i \sim p(\mathbf{x}|\boldsymbol{\theta})$ .

where  $\hat{\mathbb{E}}_{p(\mathbf{x}|\boldsymbol{\theta})}[f(\mathbf{x})]$  is the approximate value of expectation of the function.

### 3.2. Gradient Estimation

For any optimization problem, gradient plays an essential role in the efficiency and stability of the optimization algorithm. The gradient estimates the direction of the optimum value and guides the optimization algorithm in that direction [10]. For analytical functions, gradients can be easily calculated by differentiating the objective function. Black-box functions on the other hand, owing to the ambiguity of the internal functioning of their functions and constraints, make it impossible to calculate the gradient analytically [29]. In such scenarios, we resort to numerical techniques to calculate the gradient. The two approaches to obtain the derivative of the bound in equation 3.1 include the Derivative of Paths approach, where the gradient can be obtained by differentiation  $f(\mathbf{x})$  with respect to the input  $\mathbf{x}$ , or the Derivatives of Measure approach, where the gradient can be obtained by differentiation of the distribution  $p(\mathbf{x}|\boldsymbol{\theta})$  with respect to the distribution parameter  $\boldsymbol{\theta}$  [30]. We will be using the Derivatives of Measure approach for our problem. ScoutND uses score function estimator to estimate the gradient. The score function estimator, also known as the REINFORCE estimator, is commonly used in reinforcement learning [31]. As mentioned, instead of calculating the derivative of  $f$  with respect to  $\mathbf{x}$ , we calculate the derivative of expectation of  $f$  with respect to distribution parameter  $\boldsymbol{\theta}$ . We begin by defining the bound of the objective function  $U(\boldsymbol{\theta})$  and its derivative with respect to  $\boldsymbol{\theta}$ ,

$$U(\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})}[f(\mathbf{x})], \quad (3.4)$$

$$\frac{dU}{d\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})}[f(\mathbf{x})]. \quad (3.5)$$

where  $\mathbf{x} \in \mathbb{R}^n$  is our input parameter,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is our objective function,  $\mathbb{E}_{p(\mathbf{x}|\theta)}$  is the expectation with respect to the distribution  $p(\mathbf{x}|\theta)$  and  $\theta$  is the distribution parameter. We expand the term on the right hand of equation 3.5,

$$\nabla_{\theta} \mathbb{E}_{p(\mathbf{x}|\theta)}[f(\mathbf{x})] = \nabla_{\theta} \int p(x, \theta) f(x) dx \quad (3.6)$$

We encounter an issue equation 3.6. In case of black-box functions, our objective function is not differentiable. In addition to that, using quadrature can be ineffective in case our input  $\mathbf{x}$  is of high dimension [30]. We can use Monte Carlo sampling to estimate the integral in equation 3.6.

$$\nabla_{\theta} \mathbb{E}_{p(\mathbf{x}|\theta)}[f(\mathbf{x})] = \int p(x|\theta) F(x) dx, \quad (3.7)$$

where  $F(x)$  is some function. We use the "log-derivative trick" here to show that,

$$\nabla_{\theta} \log p(x|\theta) = \frac{\nabla_{\theta} p(x|\theta)}{p(x|\theta)}. \quad (3.8)$$

Substituting equation 3.8 in 3.6, we get the following equation:

$$\nabla_{\theta} \mathbb{E}_{p(\mathbf{x}|\theta)}[f(\mathbf{x})] = \int \nabla_{\theta} \log p(x|\theta) p(x|\theta) f(x) dx. \quad (3.9)$$

On comparing equation 3.7 and equation 3.9, we have

$$F(x) = \nabla_{\theta} \log p(x|\theta) f(x) dx. \quad (3.10)$$

This allows us to evaluate the gradient using equation using Monte Carlo sampling. The gradient in equation 3.6 can be simplified and written as,

$$\nabla_{\theta} \mathbb{E}_{p(\mathbf{x}|\theta)}[f(\mathbf{x})] = \mathbb{E}_{p(\mathbf{x}|\theta)}[F(\mathbf{x})]. \quad (3.11)$$

To approximate the right hand side of equation 3.11 using Monte Carlo Sampling, we write it as,

$$\hat{\mathbb{E}}_{p(\mathbf{x}|\theta)}[F(\mathbf{x})] = \frac{1}{N} \sum_{i=1}^N F(x_i) \quad (3.12)$$

where  $x_i \sim p(\mathbf{x}|\theta)$ .

Equation 3.12 is the estimated gradient for our black-box objective function.

### 3.3. Adam Optimizer

Now that we have obtained our gradient, we can pair it with any gradient based optimizer. ScoutND comes with different choices of optimizers. For our problem, we use the Adam (Adaptive Moment Estimation) optimizer [9]. The most basic optimizer, the Gradient Descent method is defined as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \cdot \left[ \frac{\partial U(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_t} \right], \quad (3.13)$$

where  $\boldsymbol{\theta}$  is the input parameter and  $U(\boldsymbol{\theta})$  is our objective function. Adam optimizer is an upgrade to other adaptive learning rate optimizers like AdaGrad [8] and RMSProp [32]. The Adam optimizer combines two important aspects:

- Adam uses exponential weighted average of past gradients. The exponential weighted average of the gradients  $\mathbf{m}_t$  is calculated as follows:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} - (1 - \beta_1) \left[ \frac{\partial U(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_t} \right]. \quad (3.14)$$

- Adam uses exponential weighted average of the square of past gradients. The exponential weighted average of the squared gradients,  $\mathbf{v}_t$ , is calculated as follows:

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} - (1 - \beta_2) \left[ \frac{\partial U(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_t} \right]^2 \quad (3.15)$$

where  $\beta_1$  and  $\beta_2$  are the exponential decay rates. We cannot directly use the values of equation 3.14 and equation 3.15 to update our parameters. The values of  $\mathbf{m}$  and  $\mathbf{v}$  are initialized as zero. This makes these values to be biased towards zero initially. To counter this, we use bias corrector which is defined as follows:

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t}, \quad (3.16)$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t}. \quad (3.17)$$

The final equation for updating the parameters using the Adam optimizer is defined as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \hat{\mathbf{m}}_t \left( \frac{\alpha}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \right). \quad (3.18)$$

The commonly accepted values for constants in Adam algorithm are shown in Table 3.1.

$\alpha = 0.001$	$\beta_1 = 0.9$	$\beta_2 = 0.999$
------------------	-----------------	-------------------

Table 3.1.: Values of constants in Adam Optimizer

### 3.4. Natural Gradients and Fisher Information Matrix

Natural gradient method is an extension to the gradient descent method which soughts to improve convergence. We defined the gradient descent method in equation 3.13. Amari [33] showed that the direction of steepest descent in a Riemannian space can be obtained using the following equation,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha G^{-1} \left[ \frac{\partial U(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_t} \right], \quad (3.19)$$

where  $G$  is the Riemannian metric tensor. The gradient descent method is a special case of equation 3.19, where  $G$  is an identity matrix, representing the gradient in the Euclidean space.

In the case of probabilistic models, we use the Fisher Information Matrix [33]. The Fisher Information matrix  $\mathbf{F}$  is the covariance of the likelihood is defined as,

$$\mathbf{F} = \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})} \left[ \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}|\boldsymbol{\theta})^{\top} \right]. \quad (3.20)$$

The modified gradient descent method is

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \mathbf{F}^{-1} \left[ \frac{\partial U(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_t} \right]. \quad (3.21)$$

Although we generalised this for the gradient descent algorithm, the new gradient can be substituted in Adam's algorithm described in section 3.3.

For a probability distribution  $p(\mathbf{x}, \boldsymbol{\theta})$ , the Fisher Information Matrix measures how well a random variable  $\mathbf{x}$  is aware of the distribution parameter  $\boldsymbol{\theta}$ . It stores the information about the curvature of the parameter space [34]. Fisher Information Matrix with higher eigenvalues indicates high curvatures, where small changes in parameters have a notable effect on the log-likelihood function and lower eigenvalues indicate flat curvatures where small changes in parameters have less effect on the log-likelihood function [35]. The Fisher Information Matrix can be ill-conditioned when the variance is high. Moreover, the Fisher Information Matrix calculation and inversion can be computationally expensive in case of higher dimension problems. ScoutND ensures that natural gradients are used after a certain degree of convergence when variance values are reduced.

### 3.5. Problem Definition

Now that we have highlighted the relevant theory behind Stochastic Variational Optimization, it is time to define the problem that ScoutND aims to solve. Consider the function  $f(\mathbf{x}, \mathbf{b})$  where  $\mathbf{x} \in \mathbb{R}^n$  is our deterministic input parameter,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is our objective function and  $\mathbf{b}$  is a random vector. The constrained optimization problem with objective function  $f(\mathbf{x}, \mathbf{b})$ , subject to constraints  $(\mathcal{C}(\mathbf{x}, \mathbf{b}))$  is defined as,

$$\begin{aligned} & \min_{\mathbf{x}} \mathbb{E}_{\mathbf{b}}[f(\mathbf{x}, \mathbf{b})], \\ & s.t \quad \mathbb{E}_{\mathbf{b}}[\mathcal{C}(\mathbf{x}, \mathbf{b})] \leq 0. \end{aligned} \quad (3.22)$$

To implement a constrained optimization, we use the Sequential Unconstrained Minimization Technique (SUMT) to convert our constrained optimization problem to an unconstrained optimization problem [36]. In SUMT, we define an augmented objective function ( $\mathcal{L}$ ) which penalizes the constraints using a penalty term ( $\lambda$ ) if they go out of bounds but leaves it unchanged if constraints are satisfied. We start by defining our penalty function ( $\mathcal{P}$ ) as,

$$\mathcal{P}(\mathbf{x}, \mathbf{b}) = \max(0, \mathcal{C}(\mathbf{x}, \mathbf{b})). \quad (3.23)$$

We define our augmented objective function  $\mathcal{L}$ , subject to  $K$  constraints as,

$$\mathcal{L}(\mathbf{x}, \mathbf{b}, \lambda) = f(\mathbf{x}, \mathbf{b}) + \lambda \cdot \mathcal{P}(\mathbf{x}),$$

Equation (3.24) represents the transformed unconstrained optimization problem of our original constrained optimization problem defined in equation 3.22. Therefore, our new optimization problem is defined as,

$$\min_{\mathbf{x}} \mathbb{E}_{\mathbf{b}}[\mathcal{L}(\mathbf{x}, \mathbf{b}, \lambda)]. \quad (3.24)$$

Using the gradient estimation method explained in section 3.2, the gradient for the problem is defined as,

$$\nabla_{\theta} U(\theta) = \mathbb{E}_{\mathbf{x}, \mathbf{b}} [\nabla_{\theta} \log p(\mathbf{x} | \theta) \mathcal{L}(\mathbf{x}, \mathbf{b}, \lambda)] \quad (3.25)$$

The gradient in equation 3.25 can be estimated using Monte Carlo sampling as follows:

$$\nabla_{\theta} U(\theta) \approx \frac{1}{N} \sum_{i=1}^N [\mathcal{L}(\mathbf{x}_i, \mathbf{b}_i, \lambda) \nabla_{\theta} \log p(\mathbf{x}_i | \theta)] \quad (3.26)$$

During optimization, when a constraint is violated, the value of  $\mathcal{L}$  increases due to the additional penalty terms. The magnitude of increase in  $\mathcal{L}$  is determined by the value of  $\lambda$  and the magnitude of constraint function  $\mathcal{C}$ . This increase in value of  $\mathcal{L}$  forces the optimizer to search within the constrained domain. The value of the penalty term ( $\lambda$ ) is another important factor to consider. Higher values of  $\lambda$  impose stricter penalties to the constraints. We start our optimization with a small value of  $\lambda$  and increase its value iteratively. This method allows our optimizer to explore the domain with less restriction initially, even if the constraints are violated. As the solution progresses, we increase  $\lambda$  to impose strict penalties, which helps in directing the optimizer towards a feasible solution.

Optimization using stochastic gradient methods has been performed before but for unconstrained problems [37, 38, 39, 40]. ScoutND extends the Stochastic Variational Optimization method to constrained optimization problems.

### 3.6. Variance Reduction

The gradient estimator explained in section 3.2 has a high variance [41]. Variance reduction techniques are used to improve the accuracy of the Monte Carlo estimator. There are many variance reduction techniques available. ScoutND uses two variance reduction techniques:

- Quasi-Monte Carlo methods (QMC) uses low discrepancy sequences to generate deterministic samples instead of pseudorandom samples. QMC samples are evenly distributed across the domain which reduces the variance. Common QMC sequences include Sobol and Halton sequences [42]. We use the Sobol sequence to generate QMC samples for our optimization problem. Figure 3.2 shows comparison between random samples and QMC samples.
- ScoutND uses the baseline [43] to reduce the variance. The derivative of the bound with respect to the distribution parameter is defined as,

$$\frac{\partial U}{\partial \theta} \approx \frac{1}{S} \sum_{i=1}^S \frac{\partial}{\partial \theta} \log p(\mathbf{x}_i | \theta) \left( \mathcal{L}(\mathbf{x}_i, \mathbf{b}_i, \lambda) - \frac{1}{S-1} \sum_{j=1, j \neq i}^S \mathcal{L}(\mathbf{x}_j, \mathbf{b}_j, \lambda) \right). \quad (3.27)$$

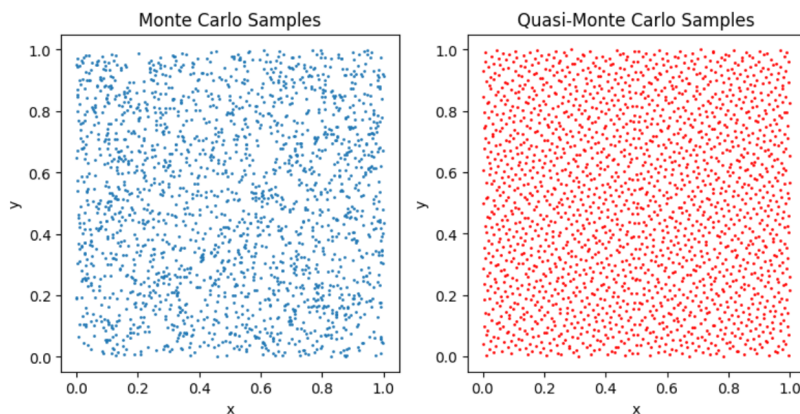


Figure 3.2.: Figure shows Monte Carlo(left) samples and Quasi-Monte Carlo samples(right) generated using Sobol sequences

### 3.7. ScoutND Algorithm

Now that we have highlighted the theory behind ScoutND, in this section, we present the general algorithm of ScoutND [25].

**Algorithm 1** ScoutND Algorithm

**Input:** Dimension( $d$ ), Number of samples  $N$ , Objective function  $f$ , Constraints ( $\mathcal{C}$ ), penalty parameter( $\lambda$ ), [Mean, Variance] ( $\theta$ ), Optimizer ( $\mathcal{G}$ )

```

1: Initialize  $\lambda, \theta, N, f, \mathcal{C}$ 
2: repeat
3:    $n \leftarrow 0$ 
4:   repeat
5:      $\mathbf{x}_i \sim p(\mathbf{x}|\theta_k^n) \mathbf{b}_i \sim q(\mathbf{b})$  //SAMPLE FROM DISTRIBUTION
6:     Evaluate  $\mathcal{L}(\mathbf{x}_i, \mathbf{b}_i, \lambda_k)$  //EVALUATE AUGMENTED OBJECTIVE FUNCTION
7:     Estimate  $\nabla_{\theta} U$  //EVALUATE GRADIENT
8:      $\theta_k^{n+1} \leftarrow \mathcal{G}(\theta_k^n, \lambda_k, \nabla_{\theta} U)$  //CALL OPTIMIZER
9:   until  $\|\theta_k^n - \theta_k^{n-1}\| > \epsilon_{\theta}$  //CONVERGENCE CRITERIA FOR DISTRIBUTION PARAMETERS

10:  $\theta_{k+1}^0 \leftarrow \theta_k^n; \{\mu, \sigma\} \leftarrow \theta_k^n$  and  $k \leftarrow k + 1$ 
11: until  $\|\sigma\| > \epsilon_{\sigma}$ 
12: return  $\{\mu, \sigma\}$ 

```





## 4. PitzDaily

In many flow systems, shape optimization plays a crucial role in components such as nozzles, diffusers, and different flow channels for better energy efficiency. Shape optimization of flow devices improves the life and reliability of such systems, which is critical in industries like aerospace and automotive, where operational costs are of significant importance. With strong resolutions taken towards shaping a sustainable environment, shape optimization of fluid flow systems can reduce the emission of harmful substances into the environment. With this motivation, we begin by introducing our shape optimization problem. The PitzDaily problem is a benchmark problem in CFD, based on an experiment by Pitz and Daily [44]. It is a two-dimensional CFD simulation problem with the fluid entering the computational domain, then going through sudden expansion due to a backward-facing step and moving forward to exit the domain through a converging passage. Figure 4.1 shows the domain of the PitzDaily problem, with the left upper boundary representing the inlet and the right boundary representing the outlet. In this chapter, we cover the CFD simulation setup for the PitzDaily problem for optimization. In section 4.1 we explain the theory behind the CFD simulation of our PitzDaily problem. Section 4.2 describes the OpenFOAM code structure used to carry out the CFD simulation. Section 4.3 describes the methodology used to generate the new boundary for our optimization problem. Section 4.4 defines the objective function of our optimization problem. Section 4.5 explains the methodology proposed to implement constraints for the problem. Section 4.6 explains the need to normalize the input. Section 4.7 explains the methodology used to convert the constrained optimization problem to an unconstrained optimization problem. Section 4.8 explains the need and methodology used to normalize the output. Section 4.9 explains the implementation of stochasticity in our optimization problem. Section 4.10 highlights the general algorithm used for the evaluation of the objective function and parallelization for the optimization process.

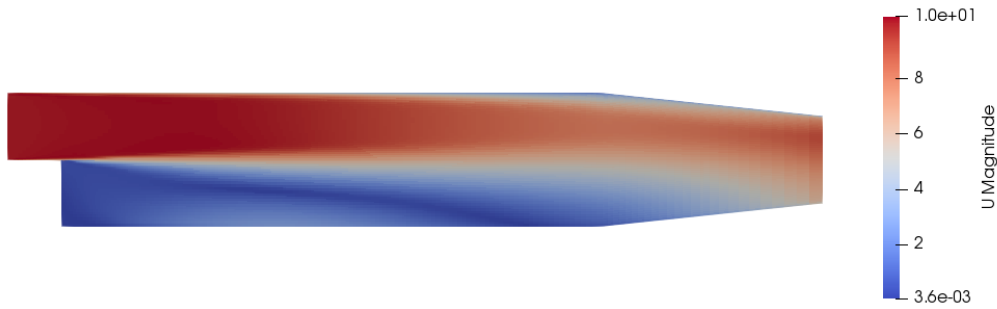


Figure 4.1.: Figure shows the domain of the PitzDaily problem.

## 4.1. CFD Simulation of PitzDaily

For the purpose of this study, the PitzDaily problem is simulated using OpenFOAM. OpenFOAM stands for Open Field Operation and Manipulation, which is an open-source software for CFD simulations [45]. It has different utilities to simulate simple as well as complex fluid flows. Being an open-source software, OpenFOAM provides the option to code and manipulate solvers. In the present study, we use the *simpleFoam* solver [46] provided by OpenFOAM as our base for the CFD simulation. We use other OpenFOAM utilities to modify the domain (to be discussed in section 4.2) which are all built upon the *simpleFoam* solver. The *simpleFoam* solver is a pressure-based solver, based on the SIMPLE algorithm. It is designed to simulate steady-state, incompressible turbulent flows. *simpleFoam* supports the use of multiple turbulence models. Table 4.1 shows the simulation settings for the PitzDaily problem.

<i>Field</i>	<i>Boundary Condition</i>
Inlet Velocity	10 m/s
Outlet Pressure	0 Pa (static)
Wall	No-slip Condition
Pressure-Velocity coupling scheme	SIMPLE
Turbulence model	$k - \epsilon$
Pressure Poisson Solver	<i>GAMG</i> with Gauss Seidel

Table 4.1.: Table shows the simulation settings for the PitzDaily shape optimization problem.

### 4.1.1. SIMPLE Algorithm

Solving the Navier-Stokes equation becomes particularly difficult due to strongly coupled pressure and velocity fields. For fluid flows, mass conservation is a strict constraint for the solution of momentum equations. Unlike incompressible flows, density can be directly related to the pressure using Equations of State in the case of compressible flows. The momentum equations in the Navier-Stokes equation require the pressure gradient at the grid point where the velocities are evaluated. Using a central differencing leads to numerical errors, known as the *checkerboard* problem, which leads to inaccurate values of pressure gradients [1]. To approach this problem, we use pressure correction methods, specifically the Semi-Implicit method for Pressure-Linked Equations (SIMPLE) Algorithm [47]. The SIMPLE algorithm is a predictor-corrector method. Figure 4.2 shows a staggered grid arrangement.

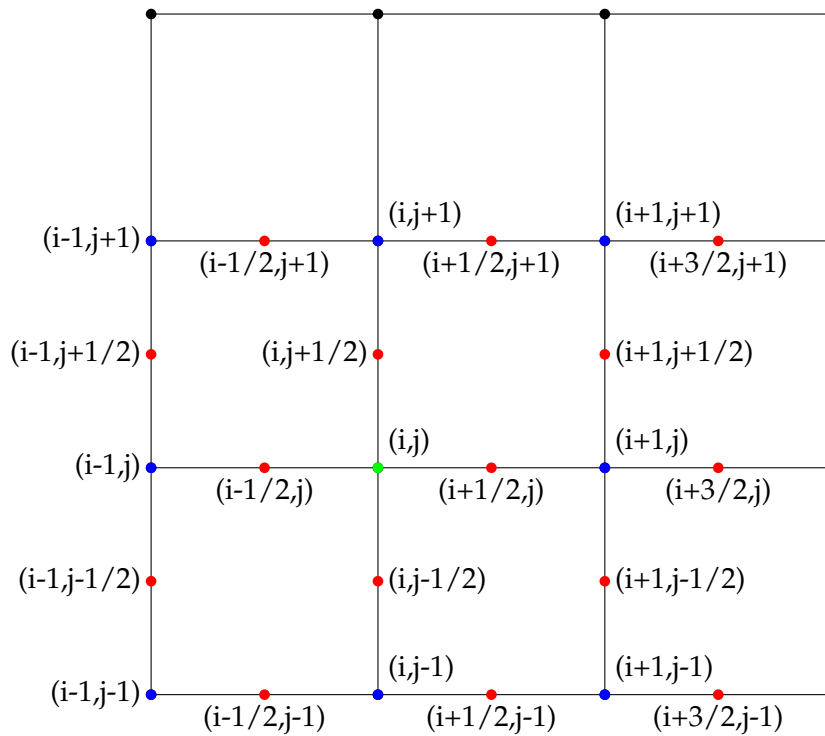


Figure 4.2.: Figure shows a staggered grid arrangement used for SIMPLE algorithm. Staggered grid mitigates the *checkerboard* board problem.

We start with the Navier-Stokes Equation in two-dimensions. The  $x$ -momentum and the  $y$ -momentum equations are

$$\frac{\partial(\rho u_x)}{\partial t} + \frac{\partial(\rho u_x^2)}{\partial x} + \frac{\partial(\rho u_x u_y)}{\partial y} = -\frac{\partial p}{\partial x} + \mu \left( \frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2} \right), \quad (4.1)$$

$$\frac{\partial(\rho u_y)}{\partial t} + \frac{\partial(\rho u_x u_y)}{\partial x} + \frac{\partial(\rho u_y^2)}{\partial y} = -\frac{\partial p}{\partial y} + \mu \left( \frac{\partial^2 u_y}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2} \right), \quad (4.2)$$

and the continuity equation is

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0, \quad (4.3)$$

where  $\rho$  is the density,  $p$  is the pressure,  $t$  is time,  $u_x$  and  $u_y$  represent the velocities in  $x$  and  $y$  directions. Consider the staggered grid arrangement shown in figure 4.2.

Discretizing the  $x$ -momentum equation at at the point  $(i + 1/2, j)$ , we can write,

$$\begin{aligned} \frac{(\rho u_x)_{i+1/2,j}^{n+1} - (\rho u_x)_{i+1/2,j}^n}{\Delta t} = & -\frac{p_{i+1,j}^n - p_{i,j}^n}{\Delta x} + \\ & - \left[ \frac{(\rho u_x^2)_{i+3/2,j}^n - (\rho u_x^2)_{i-1/2,j}^n}{2\Delta x} + \frac{(\rho u_x \bar{u}_y)_{i+1/2,j+1}^n - (\rho u_x u_y)_{i+1/2,j-1}^n}{2\Delta y} \right] \\ & + \mu \left[ \frac{(u_x)_{i+3/2,j}^n - 2(u_x)_{i+1/2,j}^n + (u_x)_{i-1/2,j}^n}{\Delta x^2} - \frac{(u_x)_{i+1/2,j+1}^n - 2(u_x)_{i+1/2,j}^n + (u_x)_{i+1/2,j-1}^n}{\Delta y^2} \right], \end{aligned} \quad (4.4)$$

On simplifying equation 4.4 we get,

$$(\rho u_x)_{i+1/2,j}^{n+1} = (\rho u_x)_{i+1/2,j}^n + A\Delta t - \frac{\Delta t}{\Delta x} (p_{i+1,j}^n - p_{i,j}^n), \quad (4.5)$$

where  $A$  is:

$$\begin{aligned} & - \left[ \frac{(\rho u_x^2)_{i+3/2,j}^n - (\rho u_x^2)_{i-1/2,j}^n}{2\Delta u_x} + \frac{(\rho u_x \bar{u}_y)_{i+1/2,j+1}^n - (\rho u_x u_y)_{i+1/2,j-1}^n}{2\Delta y} \right] \\ & + \mu \left[ \frac{(u_x)_{i+3/2,j}^n - 2(u_x)_{i+1/2,j}^n + (u_x)_{i-1/2,j}^n}{\Delta x^2} - \frac{(u_x)_{i+1/2,j+1}^n - 2(u_x)_{i+1/2,j}^n + (u_x)_{i+1/2,j-1}^n}{\Delta y^2} \right]. \end{aligned} \quad (4.6)$$

Similarly, we can show that,

$$(\rho u_y)_{i,j+1/2}^{n+1} = (\rho u_y)_{i,j+1/2}^n + B\Delta t - \frac{\Delta t}{\Delta y} (p_{i,j+1}^n - p_{i,j}^n), \quad (4.7)$$

where  $B$  is:

$$\begin{aligned}
 & - \left[ \frac{(\rho u_y^2)_{i,j+3/2}^n - (\rho u_y^2)_{i,j-1/2}^n}{2\Delta y} + \frac{(\rho u_y \bar{u}_y)_{i+1,j+1/2}^n - (\rho u_x u_y)_{i+1,j+1/2}^n}{2\Delta y} \right] \\
 & + \mu \left[ \frac{(u_y)_{i+1,j+1/2}^n - 2(u_y)_{i,j+1/2}^n + (u_y)_{i-1/2,j+1/2}^n}{\Delta x^2} - \frac{(u_y)_{i,j+3/2}^n - 2(u_y)_{i,j+1/2}^n + (u_y)_{i,j-1/2}^n}{\Delta y^2} \right].
 \end{aligned} \tag{4.8}$$

We start with  $p^*, u^*$  and  $v^*$ , as predicted values for pressure and velocity fields. We substitute these values in Equation 4.5 and Equation 4.7 to obtain the following equations:

$$(\rho u_x^*)_{i+1/2,j}^{n+1} = (\rho u_x^*)_{i+1/2,j}^n + A^* \Delta t - \frac{\Delta t}{\Delta x} ((p^*)_{i+1,j}^n - (p^*)_{i,j}^n), \tag{4.9}$$

$$(\rho u_y^*)_{i,j+1/2}^{n+1} = (\rho u_y^*)_{i,j+1/2}^n + B^* \Delta t - \frac{\Delta t}{\Delta x} ((p^*)_{i,j+1}^n - (p^*)_{i,j}^n). \tag{4.10}$$

Subtracting the predicted pressure in equation 4.9 value from the true pressure value in equation 4.5, we get,

$$(\rho u_x')_{i+1/2,j}^{n+1} = (\rho u_x')_{i+1/2,j}^n + A' \Delta t - \frac{(p')_{i+1,j}^n - (p')_{i,j}^n}{\Delta x}, \tag{4.11}$$

where

$$(\rho u_x')_{i+1/2,j}^{n+1} = (\rho u_x)_{i+1/2,j}^{n+1} - (\rho u_x^*)_{i+1/2,j}^{n+1},$$

$$(\rho u_x')_{i+1/2,j}^n = (\rho u_x)_{i+1/2,j}^n - (\rho u_x^*)_{i+1/2,j}^n,$$

$$A' = A - A^*,$$

$$p'_{i+1,j} = p_{i+1,j} - p_{i+1,j}^*,$$

$$p'_{i,j} = p_{i,j} - p_{i,j}^*,$$

Similarly, we can show that,

$$(\rho u_y')_{i,j+1/2}^{n+1} = (\rho u_y)_{i,j+1/2}^n + B' \Delta t - \frac{\Delta t}{\Delta y} (p'_{i,j+1} - p'_{i,j}), \tag{4.12}$$

where,

$$(\rho u_y')_{i,j+1/2}^{n+1} = (\rho u_y)_{i,j+1/2}^{n+1} - (\rho u_y^*)_{i,j+1/2}^{n+1},$$

$$(\rho u_y')_{i,j+1/2}^n = (\rho u_y)_{i,j+1/2}^n - (\rho u_y^*)_{i,j+1/2}^n,$$

$$B' = B - B^*,$$

$$p'_{i,j+1} = p_{i,j+1} - p_{i,j+1}^*,$$

$$p'_{i,j} = p_{i,j} - p_{i,j}^*.$$

Using equations 4.11 and 4.12 and substituting the values  $A', B', (\rho u_x')_{i+1/2,j}^n$  and  $(\rho u_y')_{i,j+1/2}^n$  as zero [1], we obtain the following equations:

$$(\rho u'_x)_{i+1/2,j}^{n+1} = -\frac{\Delta t}{\Delta y}(p')_{i+1,j}^n - (p')_{i,j}^n, \quad (4.13)$$

$$(\rho u'_y)_{i,j+1/2}^{n+1} = -\frac{\Delta t}{\Delta y}(p'_{i,j+1} - p'_{i,j})^n. \quad (4.14)$$

On substituting the values, from equations (4.13) and (4.14) in the discretized continuity equation using central differencing scheme we obtain,

$$\frac{(\rho u_x)_{i+1/2,j} - (\rho u_x)_{i-1/2,j}}{\Delta x} + \frac{(\rho u_y)_{i,j+1/2} - (\rho u_y)_{i,j-1/2}}{\Delta y} = 0, \quad (4.15)$$

$$\begin{aligned} & \frac{(\rho u_x^*)_{i+1/2,j} - \Delta t/\Delta x(p'_{i+1,j} - p'_{i,j}) - (\rho u_x^*)_{i-1/2,j} + \Delta t/\Delta x(p'_{i,j} - p'_{i-1,j})}{\Delta x} \\ & + \frac{(\rho u_y^*)_{i,j+1/2} - \Delta t/\Delta y(p'_{i,j+1} - p'_{i,j}) - (\rho u_y^*)_{i,j-1/2} + \Delta t/\Delta y(p'_{i,j} - p'_{i,j-1})}{\Delta y} = 0. \end{aligned} \quad (4.16)$$

On solving the pressure Poisson equation (4.16), iteratively, we obtain the value of  $p'$ . The pressure value at the next time step is calculated as follows:

$$p^{n+1} = (p^*)^n + p'. \quad (4.17)$$

The updated velocity fields  $u^{n+1}$  and  $v^{n+1}$  are obtained by substituting  $p^{n+1}$  in the momentum equations.

The updated pressure value,  $p^{n+1}$  is our new  $p^*$  value. The equations are iteratively solved until convergence is achieved.

#### 4.1.2. Turbulence Model

Turbulence modeling is an important aspect of CFD simulations. A straightforward approach to model turbulence is to directly solve the Navier-Stokes equation using discretization schemes and iterative solvers, a method known as Direct Numerical Simulation (DNS) method [1]. Although accurate, DNS is computationally expensive to solve which makes it unsuitable for complex problems. The Reynolds Averaged Navier Stokes (RANS) is the most popular turbulence modeling technique. RANS works by expressing the velocity and pressure fields as the sum of its mean and fluctuating components [48]. It is less computationally intensive but less accurate. Some common RANS models include the Spalart-Allmaras mode, the  $k - \omega$  model, the  $k - \epsilon$  model, and the Shear-stress model(SST) [49, 50, 51]. Large Eddy Simulation (LES) [52] is a tradeoff between DNS and RANS. LES works by applying a filter to segregate large and small eddies, subsequently resolving the large eddies and modeling the smaller eddies using SGS models. LES is more accurate than RANS but less accurate than DNS. It is less computationally expensive than DNS but more than RANS. Among the different turbulence models available, we choose the

RANS  $k - \epsilon$  model for our PitzDaily problem. The  $k - \epsilon$  model [53] has become one of the most popular turbulence models. It is robust and capable of simulating a wide range of CFD problems. Turbulence modeling begins with the Reynolds Averaged Navier Stokes (RANS) equations. The velocity and pressure fields for a two-dimensional flow are expressed as,

$$\mathbf{u} = \mathbf{U} + \mathbf{u}'. \quad (4.18)$$

$$p = P + p'. \quad (4.19)$$

where

$$u_x = U_x + u'_x, \quad (4.20)$$

$$u_y = U_y + u'_y. \quad (4.21)$$

$$(4.22)$$

where  $\mathbf{U}$  is the mean velocity component and  $\mathbf{u}'$  is the fluctuating velocity component. The time-averaged Navier-Stokes equation is obtained by substituting the velocity and pressure fields defined in equation 4.18 and equation 4.19 in the Navier-Stokes equation. Equation 4.23 shows Reynold's Averaged Navier-Stokes in tensorial form.

$$\frac{\partial(\rho U_i)}{\partial t} + \frac{\partial(\rho U_i U_j)}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j} \left[ \mu \left( \frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) - \overline{\rho u'_i u'_j} \right]. \quad (4.23)$$

In equation 4.23, we have an additional Reynold's stress term which has fluctuating velocity components. We use the Boussinesq hypothesis to express Reynold's stress term as a function of the mean velocity. Equation 4.24 shows the Boussinesq's hypothesis.

$$-\overline{\rho u'_i u'_j} = \mu_t \left( \frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} - \frac{2}{3} \frac{\partial U_k}{\partial x_k} \delta_{ij} \right) - \frac{2}{3} \rho k \delta_{ij}, \quad (4.24)$$

where  $\mu_t$  is the eddy viscosity and is an unknown quantity. We need to calculate the eddy viscosity ( $\mu_t$ ) to close the RANS equations given by 4.23.

The standard  $k - \epsilon$  model is a two-equation model which solves equation 4.26 and equation 4.25 to obtain the Turbulent Kinetic Energy ( $k$ ) and the dissipation rate ( $\epsilon$ ).

The ( $k - \epsilon$ ) model uses the following transport equations to solve for  $k$  and  $\epsilon$ .

The transport equation for the turbulent kinetic energy ( $k$ ) is given as,

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho U_i k)}{\partial x_i} = \frac{\partial}{\partial x_j} \left[ \left( \mu + \frac{\mu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] + P_k + P_b - \rho \epsilon + S_k, \quad (4.25)$$

where,  $\sigma_k$  is the Prandtl number for  $k$ ,  $P_k$  is the production of turbulent kinetic energy due to mean velocity gradients,  $P_b$  is the production of turbulent kinetic energy due to buoyancy, and  $S_k$  is the source term.

The transport equation for the turbulent dissipation rate( $\epsilon$ ) is given as,

$$\frac{\partial(\rho\epsilon)}{\partial t} + \frac{\partial(\rho U_i \epsilon)}{\partial x_i} = \frac{\partial}{\partial x_j} \left[ \left( \mu + \frac{\mu_t}{\sigma_\epsilon} \right) \frac{\partial \epsilon}{\partial x_j} \right] + C_{1\epsilon} \frac{\epsilon}{k} (P_k + C_{3\epsilon} P_b) - C_{3\epsilon} \rho \frac{\epsilon^2}{k} + S_\epsilon \quad (4.26)$$

where,  $\sigma_\epsilon$  is the Prandtl number for  $\epsilon$ ,  $C_{1\epsilon}$ ,  $C_{2\epsilon}$ , and  $C_{3\epsilon}$  are model constants,  $P_k$  is the production of turbulent kinetic energy due to mean velocity gradients,  $P_b$  is the production of turbulent kinetic energy due to buoyancy, and  $S_\epsilon$  is the source term.

The eddy viscosity can be obtained by using equation 4.27, which is substituted in 4.24 and is subsequently used to solve RANS.

$$\mu_t = C_\mu \frac{\rho k^2}{\epsilon}. \quad (4.27)$$

The constants used in the equation have been extensively researched [54, 55] resulting in the standardised values shown in table 4.2:

$C_\mu = 0.09$	$\sigma_k = 1.00$	$\sigma_\epsilon = 1.30$	$C_{1\epsilon} = 1.44$	$C_{2\epsilon} = 1.92$
----------------	-------------------	--------------------------	------------------------	------------------------

Table 4.2.: Values of constants in  $k - \epsilon$  equations

### 4.1.3. Geomtric Algebraic Multi-Grid (GAMG)

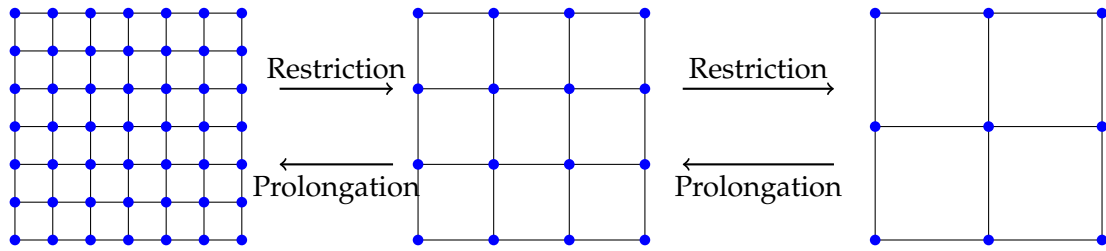


Figure 4.3.: Figure shows different levels in Multi-Grid from the finest(left) to the coarsest(right).

The idea of Multigrid solvers was proposed by Brandt [56] to accelerate the convergence of iterative methods. GAMG belongs to the class of Multigrid solvers[46]. In multi-grid solvers, the simulation is initially performed on a fine mesh, which reduces the high-frequency errors, leaving the low-frequency errors untouched. These low-frequency errors are transferred to a coarser mesh through a process known as *restriction*. The low-frequency errors behave as high-frequency errors on the coarse mesh and are resolved subsequently. The cycle continues iteratively until the coarsest mesh is reached. From here, the solution is interpolated back to the finer meshes in steps through the process known as *prolongation*. GAMG is not a solver in itself. We use the Gauss-Seidel solver on



each grid of the Multigrid meshes. In Multigrid methods, the solver used to reduce the errors is called the *smoother*.

According to traversal between different grids, the multigrid method can be categorized into different cycles[57].

- V-cycle: In the V-cycle, the algorithm descends to the coarsest grid by traversing through different levels and then returns back to the topmost grid in the same manner. In the V-cycle, each intermediate level is visited only twice, once while descending and once while ascending. Figure 4.4 shows a V-cycle.

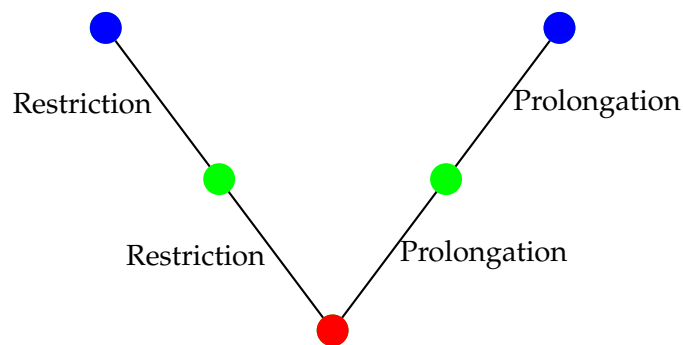


Figure 4.4.: Figure shows a V-cycle Multi-Grid

- W-cycle: In the W-cycle, the algorithm descends to intermediate levels and may revisit these levels multiple times before descending to the coarsest level and ascending to the finest level. It is used in case of more difficult problems. Figure 4.5 shows a W-cycle.

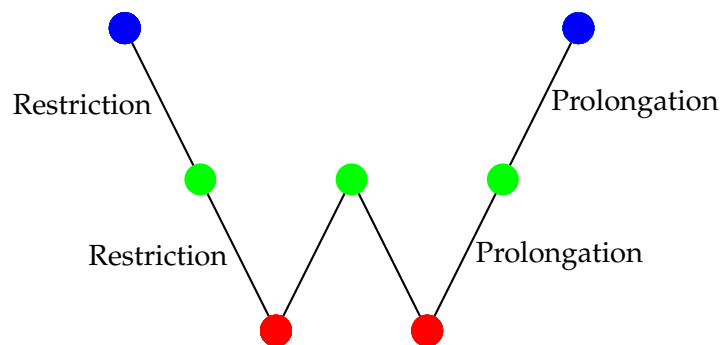


Figure 4.5.: Figure shows a W-cycle Multi-Grid

- Full-MultiGrid Cycle: In the Full-MultiGrid cycle, we start with the coarsest mesh and move to the next finer mesh. We perform a V-cycle and end up in the next

finer grid above the previous level. This cycle continues till we reach the finest grid. Figure 4.6 shows the full multigrid cycle.

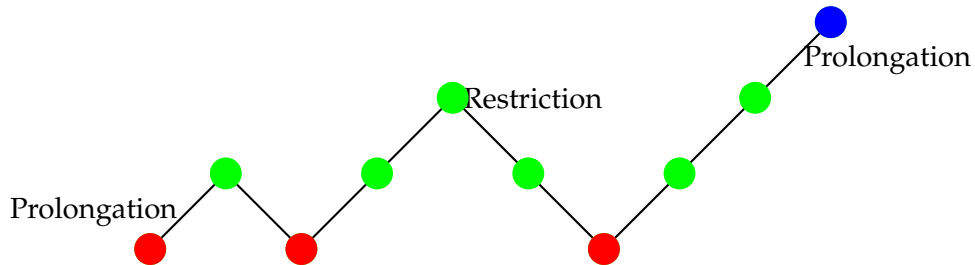


Figure 4.6.: Figure shows a Full Multi-Grid

## 4.2. OpenFOAM Simulation

This section explains the different libraries used in OpenFOAM for the PitzDaily problem. Figure 4.7 shows the code structure used for the Pitzdaily simulation. The PitzDaily folder consists of three main folders: *constant*, *system*, and *0*. We will be discussing the role of relevant files in these folders.

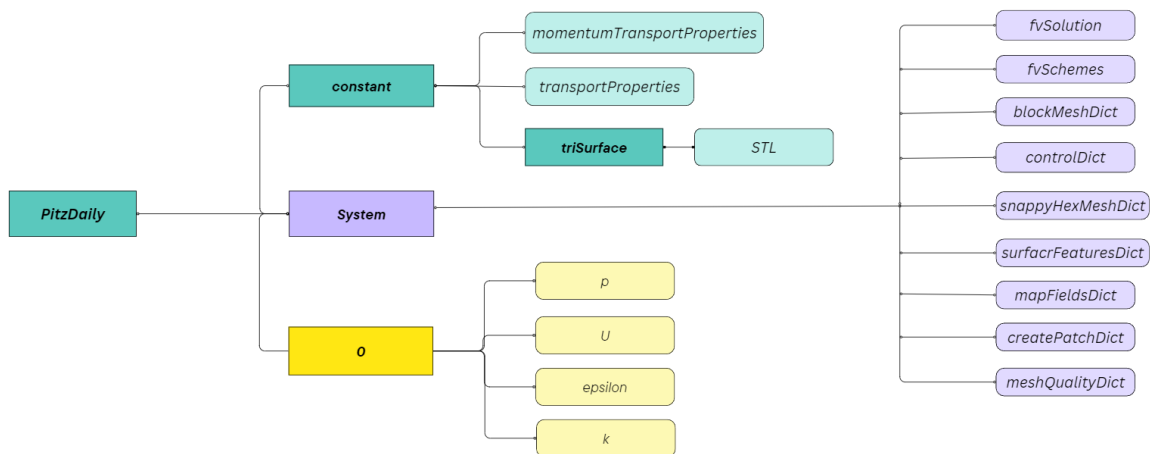


Figure 4.7.: Flowchart representing the OpenFOAM code structure for the PitzDaily problem

The **system** folder consists of files defining the parameters and simulation settings. We describe the important files used in the simulation. `fvSolution` file specifies settings relevant to the solution process. For our simulation, we use the *GAMG* solver for the pres-

sure field with the Gauss-Seidel method as smoother. We use the Gauss-Seidel method for the velocity fields ( $U$ ), the turbulent kinetic ( $k$ ) energy, and the dissipation rate ( $\epsilon$ ). The `fvSolution` file also specifies the residual settings for the SIMPLE algorithm like the reference pressure and the convergence criteria. The last thing we include in this file is the under-relaxation factors to ensure the stability of the solution. The `fvSchemes` file specifies the different discretization schemes for the solver. We use the `steadyState` discretization scheme for our time discretization since we are simulating a steady-state problem. The file also specifies the variable for which we calculate the flux, which in our case is the pressure. For the divergence of the convective flux of the velocity field  $U$ , the `"boundedGausslinearUpwindV grad(U)"` scheme is used. This scheme ensures numerical stability and accuracy by combining Gaussian quadrature with a bounded linear upwind interpolation that uses the gradient of the velocity field. For the turbulence kinetic energy  $k$  and the turbulence dissipation rate  $\epsilon$ , the `"boundedGaussupwind"` scheme is used. The interpolation scheme used for the simulation is `"linear"`. The `blockMeshDict` file specifies the measurement unit, points, and the different blocks of the simulation domain. It also specifies the boundary patches. It is used to generate a basic structured mesh using the `blockMesh` command. The `controlDict` file stores the simulation settings like the timestep, write interval, end time, etc. It also stores the list of functions used in post-processing. The PitzDaily case can be simulated without the `snappyHexMeshDict`, but this file becomes necessary for our simulation since we are creating a more complex geometry and modifying the original domain. The `snappyHexMesh` is a meshing tool in OpenFOAM used to generate hexahedral meshes. Using `castellatedMeshControls` utility, we define the maximum number of cells, minimum refinement of cells, and refinement of the edges and surfaces of the generated boundary during the optimization process. The `addLayersControls` utility is used to add and refine the boundary layer to capture the flow characteristics near the walls of the new boundary generated during optimization. `snapControls` utility is used to project and align the mesh with the domain geometry. To generate the mesh, `snappyHexMesh` command is used. The `surfaceFeatureDict` file is used to define the surface features. The surface is stored in the `constant/triSurface` folder. During optimization, the generated STL curve is copied into this folder. The `mapFieldsDict` file is used to map the fields from one mesh to another. It contains the `patchmap` utility, which maps the data from the source mesh to the target mesh, and the `cuttingPair` utility, which specifies the patch of the target mesh that cuts through the domain. In our case, this would be our lower wall surface generated during optimization. The `createPatchDict` file specifies the name, type, and the method using which the patch is created. The `meshQualityDict` file specifies and checks the quality of the mesh generated. The **constant** folder consists of the `transportProperties` file which specifies the physical properties of the fluid in the simulation. We specify the fluid type as Newtonian and the kinematic viscosity value. The `turbulenceProperties` file specifies the turbulence model used, which in our case is the  $k - \epsilon$  RANS model. The `0` folder contains files that specify the initial conditions for the velocity field ( $U$ ), pressure field ( $p$ ), kinematic viscosity ( $\nu$ ), turbulent kinetic energy ( $k$ ) and the dissipation rate ( $\epsilon$ ).

### 4.3. Catmull-Clark Subdivision Curves

For the current problem, the lower boundary of the PitzDaily problem needs to be changed in order to optimize the objective function. This requires generation of a curve and patching it to the lower boundary of the PitzDaily problem on each function evaluation. To do this, we define a bound around the lower boundary. Within this bound, we specify a set of control points. To create a curve using these control points, we use the Catmull-Clark Subdivision curve [58]. To generate the Catmull-Clark Subdivision curve, we start by setting the boundary points at their fixed positions using equation 4.28 and 4.29. We insert new points between two adjacent points using equations 4.30 and 4.31 at the  $j^{th}$  iteration. This algorithm is capable of generating a smooth curve in a few iterations. The governing equations to generate the curve are:

$$S^j[1] = S^{j-1}[1], \quad (4.28)$$

$$S^j[|S^j|] = S^{j-1}[|S^{j-1}|], \quad (4.29)$$

$$S^j[2i] = \frac{S^{j-1}[i-1] + 6S^{j-1}[i] + S^{j-1}[i+1]}{8}, \quad (4.30)$$

$$S^j[2i+1] = \frac{4S^{j-1}[i-1] + 4S^{j-1}[i+1]}{8}. \quad (4.31)$$

This generated curve is converted into an STL file which is mapped on the domain and replaces the lower boundary of the original PitzDaily problem. The simulation for the optimization process is carried out on this new domain. Figure 4.8 describes the problem and figure 4.9 shows the final domain on which the simulation is carried out.

Figure 4.8 shows a general outline of the PitzDaily domain. Figure 4.9 shows the final PitzDaily domain for the simulation. Figure 4.10 shows the generated STL curve. Figure 4.11 shows the modified PitzDaily domain after the STL is mapped on the old domain. Figure 4.12 shows the velocity contour when simulated on the newly generated domain.



Figure 4.10.: Figure shows the STL curve generated using the Catmull-Clark Subdivision curve.

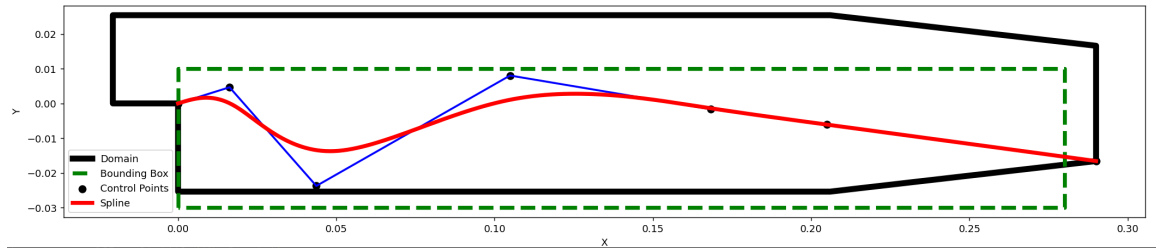


Figure 4.8.: Figure shows the new PitzDaily problem. The black line represents the original PitzDaily problem. The green box represents the rectangular bound within which the control points are generated. The blue points represent the control points. These points are used to generate the red curve which is the generated Catmull-Clark subdivision curve.

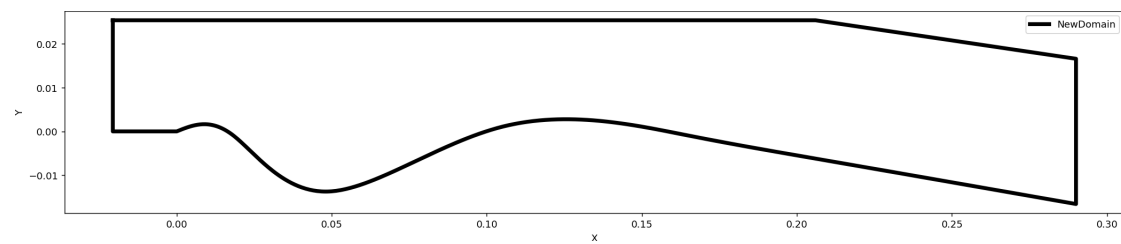


Figure 4.9.: Figure shows the final PitzDaily domain on which simulation is carried out for the shape optimization problem.

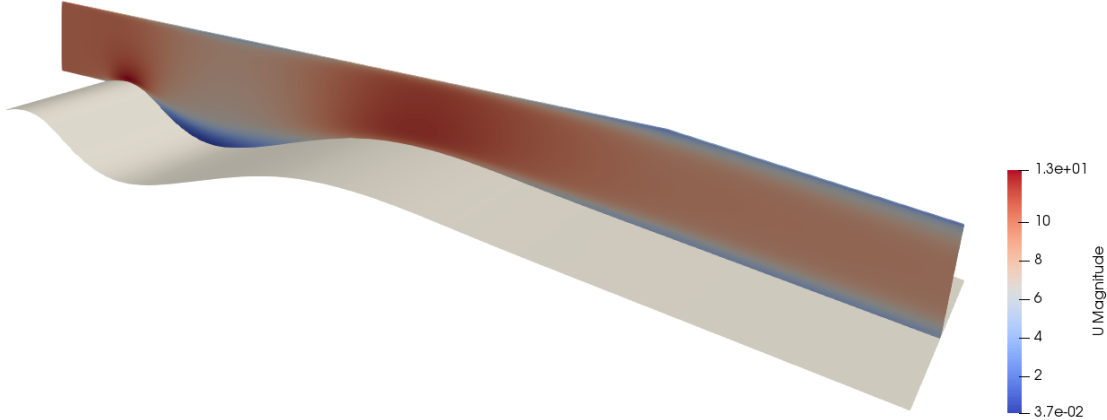


Figure 4.11.: Figure shows the STL curve mapped on the original PitzDaily problem.

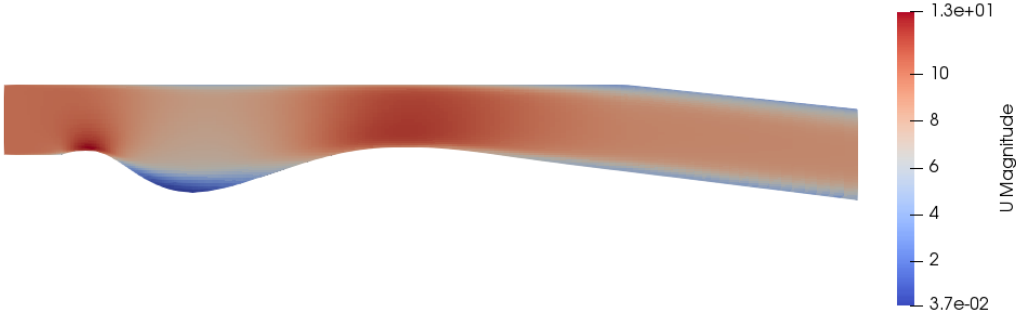


Figure 4.12.: Figure shows the final simulation on the New Generated Domain

## 4.4. Objective Function

The previous sections of this chapter described the setup required for the evaluation of the objective function for our optimization problem. CFD simulations are computationally expensive and time-consuming to evaluate. Optimization requires multiple calls of the objective function and this is a major drawback in the optimization of CFD systems. In addition to that, CFD systems are complex, non-linear systems which makes gradient calculation unfeasible. This drives our motivation to use black-box optimizers, such as ScoutND for optimizing CFD systems. The objective function of our optimization problem( $\xi$ ) inspired by the work of [58], is a non-dimensionalized value obtained by dividing the static pressure drop across the inlet and the outlet boundaries with the kinetic energy per unit volume of the fluid. It is defined as

$$\xi = \frac{2}{\rho \cdot U_{in}^2} \left[ \frac{1}{A_{in}} \int p_{in} \cdot dA_{in} - \frac{1}{A_{out}} \int p_{out} \cdot dA_{out} \right], \quad (4.32)$$

where,  $A_{in}$  and  $A_{out}$  are the areas of the inlet and the outlet boundaries, respectively,  $p_{in}$  and  $p_{out}$  are the static pressures at the inlet and the outlet, respectively,  $U_{in}$  is the velocity at inlet and  $\rho$  is the density of the fluid.

## 4.5. Constrained Optimization

To ensure the control points do not go out of the bound, we need to define constraints. Scout-ND requires the constraint's value to be less than or equal to zero if the constraint is satisfied and greater than zero if the constraint is not satisfied. Each control point has the following 4 constraints for its  $x$  and  $y$  coordinates:

$$x \geq x_{min} ; x \leq x_{max} ; y \geq y_{min} ; y \leq y_{max} \quad (4.33)$$

where  $x_{min}$ ,  $x_{max}$ ,  $y_{min}$  and  $y_{max}$  represent the minimum and maximum values of the bounds. For  $n$  control points, there are  $4n$  constraints. We propose a method to agglomerate all constraints into a single value.

Let's say a control point is randomly generated inside or outside the polygonal bound as shown in figure 4.13. If we join the control point to each corner of the polygonal bound, it leads to the formation of  $K$  triangles, where  $K$  is the number of vertices of the polygon. The vertices of each triangle are the two consecutive points of the polygonal bound, and one vertex being the control point itself. We calculate the sum of the area of  $K$  triangles. If the control point is inside the bound, the sum of the  $K$  triangles will be equal to the area of the polygonal bound and if the control point is outside the bound, the sum of the  $K$  triangles will be greater than the area of the polygonal bound. We replicate this for all  $M$  control points. The constraint value ( $\mathcal{C}$ ) is calculated as

$$C = \frac{\sum_{j=1}^M \left( \sum_{i=1}^K (A_{\Delta})_i \right)_j}{A_{bound} \times (M)} - 1, \quad (4.34)$$

where  $A_{\Delta}$  is the area of triangle,  $A_{bound}$  is the area of the bound,  $M$  is the number of control points and  $K$  is the number of vertices of the polygonal bound. Constraint value ( $C$ ) is zero if all control points are inside the domain. Constraint value ( $C$ ) is positive if even one control point is outside the domain. For our problem, we use,  $M = 5$  and  $K = 4$

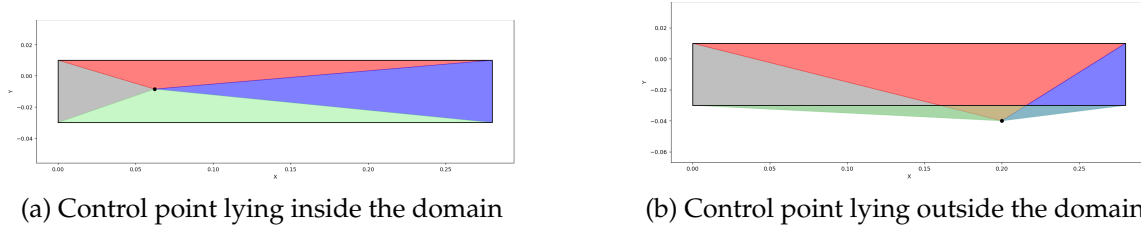


Figure 4.13.: Figure (4.13a) shows the control point inside the domain while (4.13b) shows the control point outside the domain. The value of constraint using equation (4.34) is equal to zero in case (4.13a) and greater than zero in case (4.13b).

## 4.6. Normalization of Input

Scout-ND generates samples for coordinates of the control points using the input mean and variance. An issue here is that if samples are generated in such a way that the control points are way outside the bound, it may happen that the generated curve leads to unfeasible CFD simulation. Figure 4.14 shows one such scenario where this might happen.

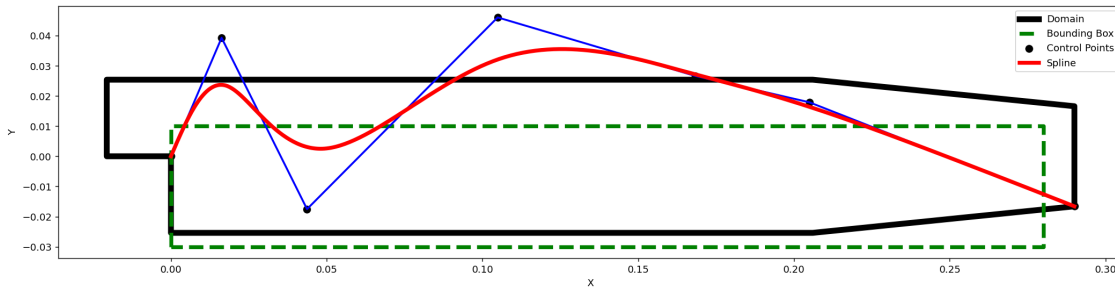


Figure 4.14.: Figure shows the control points going out of bound which causes issue in simulating the PitzDaily problem as the domain splits in between, making the simulation infeasible.

The red curve is the generated Catmull-Clark Subdivision curve. As seen from the figure, the red curve completely splits the PitzDaily domain, and in this case, the simulation



is unfeasible, and  $\xi$  value cannot be obtained. Using low variance values circumvents this issue but this leads the optimizer in a local minima without exploring the bound completely. In order to resolve this, the input is normalized to ensure that the mentioned situation does not arise and the optimizer can explore the domain without abruptly generating an unfeasible PitzDaily domain.

## 4.7. Unconstrained Optimization

We propose a simple method to convert our CFD problem to an unconstrained optimization problem without using SUMT. To achieve this, we apply the sigmoid transformation to the input coordinates of the control points. Sigmoid transform is given as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \quad (4.35)$$

Figure 4.15 shows a basic sigmoid Transformation function that restricts an input ( $X_{min}, X_{max}$ ) between (0.0, 1.0).

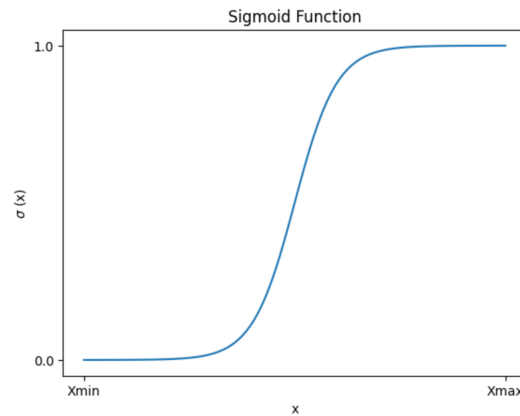


Figure 4.15.: Sigmoid Function

We modify the original sigmoid function to restrict the  $x$  and  $y$  coordinates of each control point to lie in the range defined in equation 4.33. The modified equations are defined as,

$$\sigma(x) = x_{\min} + \frac{x_{\max} - x_{\min}}{1 + e^{-(x) \cdot k_x}}, \quad (4.36)$$

$$\sigma(y) = y_{\min} + \frac{y_{\max} - y_{\min}}{1 + e^{-(y) \cdot k_y}}. \quad (4.37)$$

## 4.8. Normalization of Output

Normalization plays a crucial role in optimization. Normalization of output leads to faster convergence and prevents numerical instabilities. There are many ways to optimize the objective function. We randomly generate 50 samples of input coordinates and simulate them. We use the "Min-Max Normalization" technique defined as follows:

$$\xi' = \frac{\xi - \xi_{min}}{\xi_{max} - \xi_{min}}, \quad (4.38)$$

where  $\xi'$  is the normalized value of the objective function,  $\xi$  is the current value of the objective function,  $\xi_{max}$  is the maximum value of the objective function from the 50 simulated samples and  $\xi_{min}$  is the minimum value of the objective function from the 50 simulated samples.

## 4.9. Stochasticity in Fluid flows

Fluid flow problems in practical scenarios are inherently unpredictable due to their random behavior. Turbulence is a major cause of this randomness. These random parameters follow a distribution instead of being fixed. In such a scenario, it is crucial to account for stochasticity in order to predict more accurate results. ScoutND is capable of optimizing under stochastic conditions and can be easily implemented in the algorithm. To achieve this, instead of using a fixed value for the inlet velocity, we introduce a stochastic inlet velocity generated using random uniform sampling between a provided upper limit,  $U_{max}$  and lower limit  $U_{min}$ .

$$x \sim \mathcal{U}(U_{min}, U_{max}). \quad (4.39)$$

Equation 4.39 is used to sample an inlet velocity value at each function evaluation. This sampled inlet velocity is subsequently used to simulate the PitzDaily problem.

## 4.10. Objective Function Evaluation

Now that we have highlighted the necessary steps for shape optimization of our PitzDaily problem, we outline the algorithm used to calculate the objective function and the constraints in this section. Algorithm 2 presents the algorithm to calculate the objective function and Algorithm 3 presents the algorithm to calculate the constraints.

**Algorithm 2** Objective Function Algorithm

**Input:** Input Array  $\theta$  of  $x$  and  $y$  coordinates of size  $2N$ , where  $N$  is the number of control points

**Output:** Mechanical Energy Loss Factor,  $\xi$

```

1:  $\theta \leftarrow \text{Reshape}(\theta)$  // ARRAY FROM SIZE (1,N) TO (N/2 , 2)
2:  $\theta \leftarrow \text{Sort}(\theta)$  // SORT IN ORDER OF INCREASING X-COORDINATES
3:  $\theta \leftarrow \text{Vstack}(\theta)$  // CONNECT CURVE TO FIXED POINTS IN THE DOMAIN
4:  $\theta \leftarrow \text{GenCurve}(\theta)$  // GENERATE CATMULL-CLARK SUBDIVISION CURVE
5:  $\text{GenSTL}(\theta)$  // CONVERT TO STL FILE
6:  $\xi \leftarrow \text{Solve}()$  // RUN OPENFOAM COMMANDS TO CALCULATE ( $\xi$ )
7:  $\xi' \leftarrow \text{Normalize}(\xi)$  // NORMALIZE THE OUTPUT, ( $\xi$ )
8: return  $\xi'$ 

```

**Algorithm 3** Constraint Function Algorithm

**Input:** Input Array  $\theta$  of  $x$  and  $y$  coordinates of size  $2N$ , where  $N$  is the number of control points

**Output:** Constraint Value,  $\mathcal{C}$

```

1:  $\theta \leftarrow \text{Reshape}(\theta)$  // ARRAY FROM SIZE (1,N) TO (N/2 , 2)
2:  $\theta \leftarrow \text{Sort}(\theta)$  // SORT IN ORDER OF INCREASING X-COORDINATES
3:  $sumArea \leftarrow 0$ 
4: for  $i$  in Control point do
5:    $ar(\Delta)_i = \text{calcArea}(i, domainPoints)$  // CALCULATE AREA OF EACH TRIANGLE
    $sumArea \leftarrow sumArea + ar(\Delta)_i$  // SUM OF ALL TRIANGLES
6: end for
7:  $\mathcal{C} \leftarrow \text{Constraint}(sumArea)$  // CALCULATE CONSTRAINT VALUE USING EQUATION 4.34
8: return  $\mathcal{C}$ 

```

The average evaluation time of the objective function using OpenFOAM is 40 seconds. An optimization problem with 5 control points requires 128 sample evaluations per iteration. With an average compute time of 40 seconds per sample, a single iteration may take up to 90 minutes. Scout-ND is based on Monte Carlo sampling for gradient estimation. This makes the evaluation of the samples "embarrassingly parallel". To address the problem of long computation times, a parallel setup of function evaluations is employed.



## **Part III.**

# **Results and Conclusion**



## 5. Results

In this chapter, we present the results of our optimization problem. Section 5.1 presents the results of non-stochastic constrained optimization using ScoutND. In section 5.2, we present the results of stochastic constrained optimization using ScoutND. Section 5.3 presents the results of non-stochastic unconstrained optimization using ScoutND. Section 5.4 presents the results of stochastic unconstrained optimization using ScoutND. In section 5.5, we present a comparison of results of different ScoutND simulations and also compare the results of ScoutND with the results of different optimizers techniques such as Nelder-Mead, L-BFGS-B, COBYLA, and SLSQP.

### 5.1. Non-stochastic constrained optimization using ScoutND

In this section, we present the results of optimization with ScoutND. We define the dimensions of the bound for the control points as:

$$x_{min} = 0.0 ; x_{max} = 0.28 ; y_{min} = -0.03 ; y_{max} = 0.01. \quad (5.1)$$

The next step is to provide an input array as an initial guess for our optimization problem. Since we are using five control points to generate our boundary and each control point is defined by  $x$  and  $y$  coordinates, we need to provide a mean array of size 10 with every two consecutive elements representing the coordinates of a single control point and an array of initial values of variance for each element of our mean array. Therefore, our initial guess array is of size 20. As discussed in section 4.6, we provide normalized values of  $x$  and  $y$  coordinates as our initial guess which are normalized in the range (-1,1). In case of variance, the initial guess value we provide is the value of exponent of  $e$  i.e. we provide the values of  $\ln(\sigma)$ . We start with a variance of  $\sigma = e^{-1}$ . Therefore, we assign an initial value of  $-1$  to initialise the variance for all 10 input parameters of the variance array. The initial guess,  $\theta = \{\mu, \ln(\sigma)\}$ , used for our optimization problem is defined as,

$$\mu = [-0.9, 0.0, -0.45, 0.0, 0.0, 0.0, 0.45, 0.0, 0.9, 0.0], \quad (5.2)$$

$$\ln(\sigma) = [-1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0]. \quad (5.3)$$

The next step is to initialise our penalty term ( $\lambda$ ). A violated constraint in our problem does not return a very high magnitude. The magnitude order can be around  $10^{-3}$ . In such

a scenario, it only makes sense to start with a value of  $\lambda$  that scales the penalty component to be in the same order of magnitude as the objective term. ScoutND initializes penalty term using powers of  $e$ . We start with an initial value of  $\lambda = e^0$ . As discussed in section 2.3, we use the Adam optimizer with a fixed learning rate. A very high learning rate might cause oscillations and a small learning rate slows down convergence. To counter this, we start with a higher value of learning rate of 0.1, reduce it to 0.01, and then to 0.001 as the solution converges. This prevents the optimizer from overshooting the optimum and helps in improving the quality of the optimum without compromising on the rate of convergence. The non-normalized, true optimum value of the objective function for our PitzDaily problem is  $\xi^* = 0.0$ , which indicates zero pressure drop. Figures 5.1 and 5.2 show the velocity and the pressure contours for the simulation carried out using the initial guess value from equation 5.2. The value of the normalized objective function ( $\xi'$ ), from equation 4.38, for our initial guess is 0.7863.

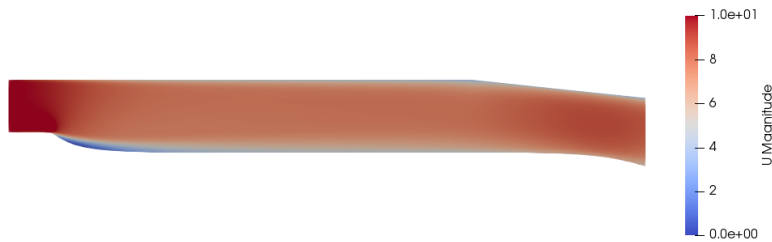


Figure 5.1.: Velocity Contour obtained from simulation using initial values from equation 5.2.



Figure 5.2.: Pressure contour obtained from simulation using initial values from equation 5.2.

Figures 5.3 and 5.4 show the simulation carried out for the optimized values. Our objec-



tive function is a non-dimensionalised quantity which is mainly dependent on the static pressure difference between the inlet and the outlet. It can be easily observed, that the optimized domain enforces this condition. The fluid enters the domain and goes through a gradual expansion. This increases the pressure in the area following the inlet boundary. In addition to this, we observe that near the outlet boundary, the generated curve slightly reduces the cross-section area near the outlet, which increases the pressure in the region preceding the outlet boundary. This further increases the pressure near the inlet. A thing to note here is that the static pressure at the outlet is fixed due to the outlet boundary condition. Therefore, in the original PitzDaily problem, we have a lesser static pressure value at the inlet, which is the reason our optimized curve tries to increase the static pressure at the region preceding the outlet boundary. Figure 5.6 shows the pressure plotted on a horizontal line, extending from the inlet to the outlet, along the flow i.e. in the X-direction. We observe that the pressure difference between the inlet and the outlet is negligible for the optimized curve, which directly influences the objective function. Figure 5.5 shows the control points for the optimized values of the non-stochastic constrained optimization problem.

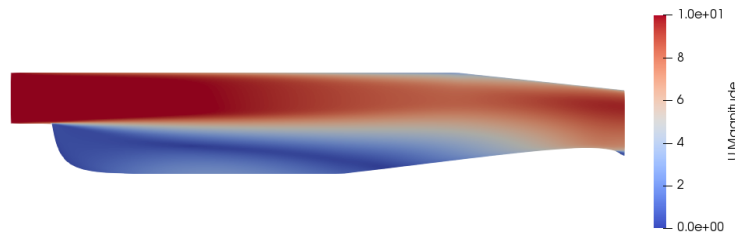


Figure 5.3.: Velocity Contour obtained from simulation of optimized values of non-stochastic constrained optimization problem.

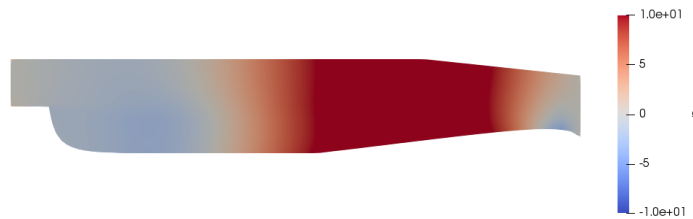


Figure 5.4.: Pressure Contour obtained from simulation of optimized values of non-stochastic constrained optimization problem. We observe that the pressure at the inlet and outlet boundaries have the same color, indicating very close pressure values.

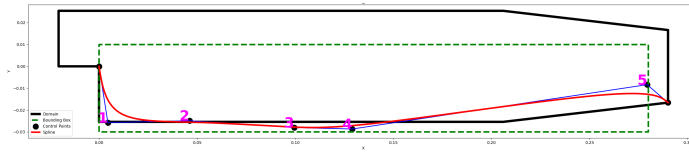


Figure 5.5.: Figure shows the domain and the control points for the non-stochastic constrained optimization problem. The numbers next to the point denote the point number.

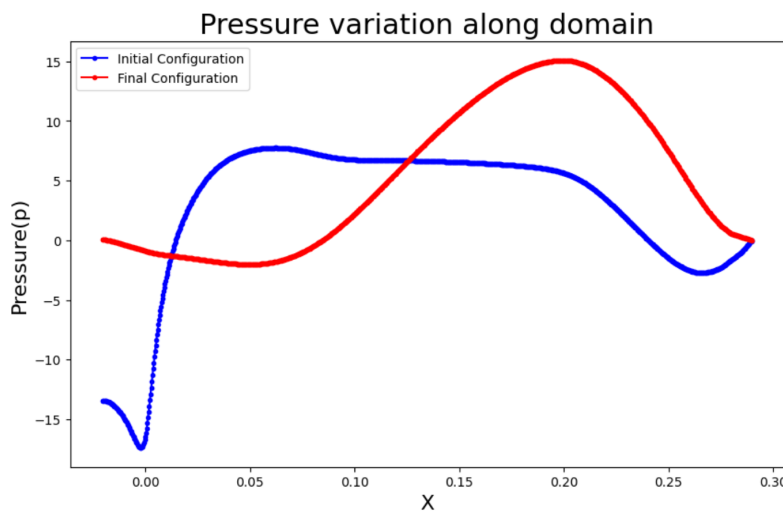
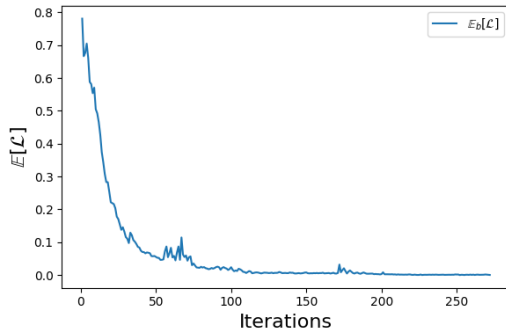
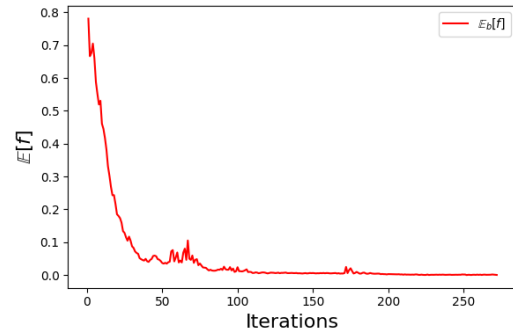


Figure 5.6.: Figure shows the pressure variation along the flow, starting from the inlet boundary to the outlet boundary, for the initial and the final configuration for the non-stochastic constrained optimization problem. We can observe that the pressure difference between the inlet and the outlet is almost negligible for the final configuration.

Figures 5.7a and 5.7b show the evolution of the augmented objective function and the evolution of the objective function with iterations. We observe that during the initial iterations, the graph of the augmented objective function and the objective function are slightly different as the optimizer searches for the minimum value throughout the domain as we start with a lower value of penalty term and higher value of variance. As the solution converges towards the minima, the graphs show a similar trend.



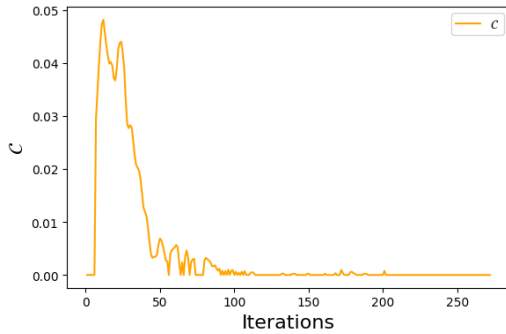
(a) Figure shows evolution of  $\mathbb{E}[\mathcal{L}]$  with iterations.



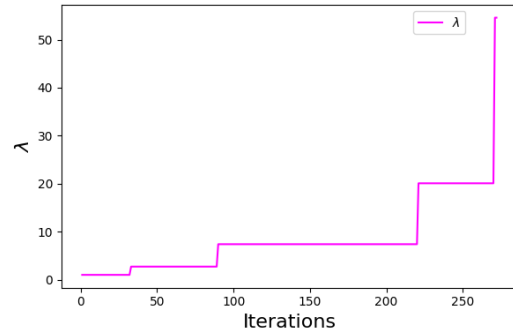
(b) Figure shows evolution of  $\mathbb{E}[f]$  with iterations.

Figure 5.7.: Figure shows evolution of  $\mathbb{E}[\mathcal{L}]$  and  $\mathbb{E}[f]$  with iterations for the non-stochastic constrained optimization problem. The two graphs are different initially as the optimizer searches for the minimum throughout the domain because of lower value of  $\lambda$  and higher value of  $\ln(\sigma)$ , but align as the solution converges.

Figure 5.8a shows the evolution of the constraint value with iterations. We observe that the constraints initially have a high value due to a lower value of the penalty term and decreases as the solution progresses due to strict penalisation and convergence of solution towards the minimum value within the bound. Figure 5.8b shows the evolution of penalty term with iterations. We observe that the penalty term increases in steps as the solution progresses.



(a) Figure shows evolution of  $\mathcal{C}$  with iteration.



(b) Figure shows evolution of  $\lambda$  with iteration.

Figure 5.8.: Figure shows evolution of constraint ( $\mathcal{C}$ ) and the penalty term ( $\lambda$ ) for the non-stochastic constrained optimization problem with iterations. The constraints during the initial iterations have a higher magnitude due to lower value of  $\lambda$ .

Figures 5.9 and 5.10 show the evolution of the mean and the variance of each variable

with iterations, respectively. We started with the values shown in equations 5.2 and 5.3. We pointed out that the objective function decreases as the region near the outlet restricts the flow. This makes the objective function sensitive to the control point 5, which is the control point nearest to the outlet boundary. We observe from figure 5.3 that the  $(\ln(\sigma_{X5}))$  and  $(\ln(\sigma_{Y5}))$  have the least values as the solution converges. This shows that the control point 5 is the most sensitive point around the optimum. Following them is  $(\ln(\sigma_{X1}))$ , a parameter of the control point 1, which shows that the shape of the lower boundary in the region where the fluid goes through expansion, has a notable effect on the objective function.

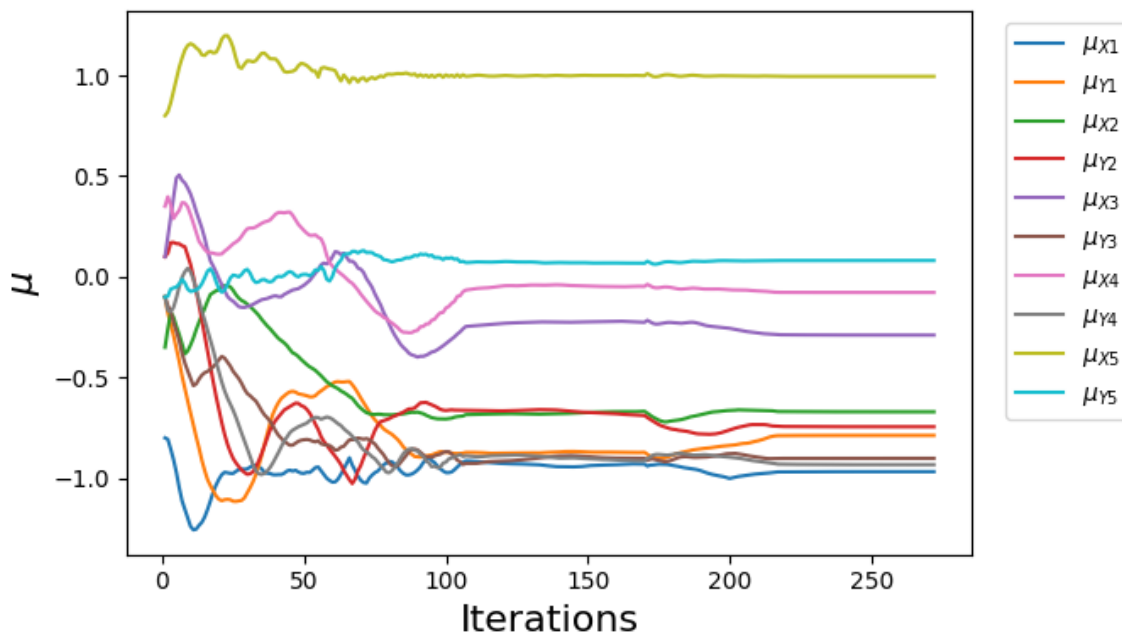


Figure 5.9.: Figure shows the evolution of  $\mu$  of each input variable for the non-stochastic constrained optimization problem.

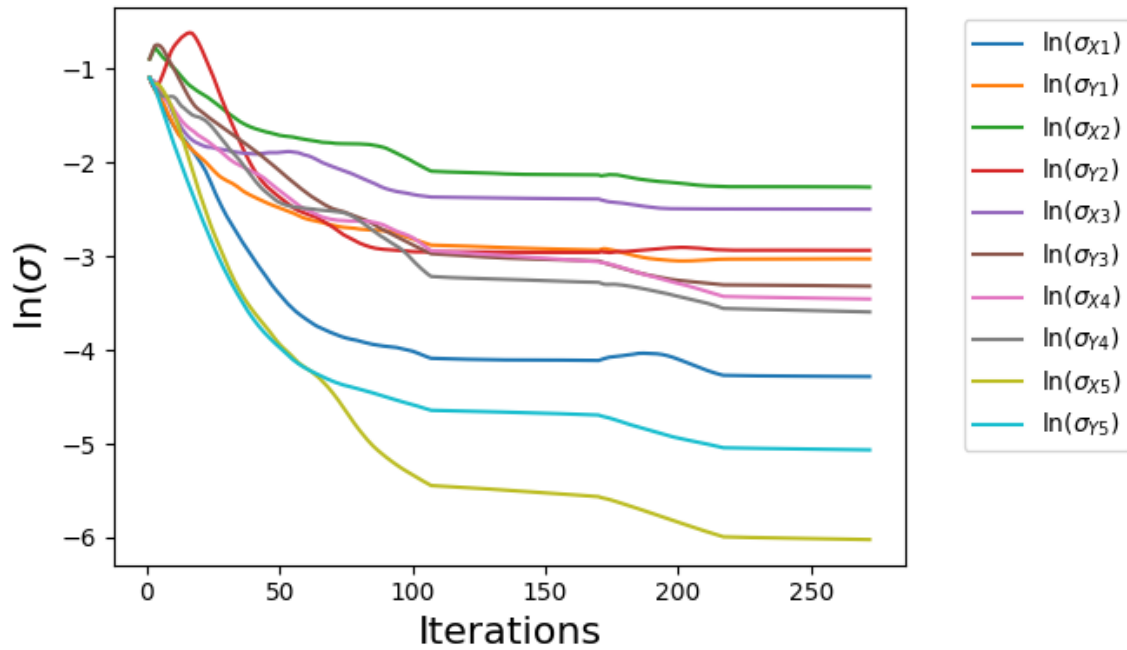


Figure 5.10.: Figure shows the evolution of  $\ln(\sigma)$  of each input variable with iterations for the non-stochastic constrained optimization problem. It is observed that the points near the boundary, with minimum cross-section area, have the least variance as these points are critical in the optimization of the objective function. The point near the expansion area also has a notable effect on the objective function.

Figure 5.11 shows the evolution of the learning rate with iterations. The effect of change in learning rate can be seen in the evolution of  $\mu$  and  $\ln(\sigma)$ . As the learning rate reduces, the convergence of  $\ln(\sigma)$  slows down, the oscillations reduce and the solution converges gradually. Similarly, the oscillations in  $\mu$  reduce as the learning rate drops.

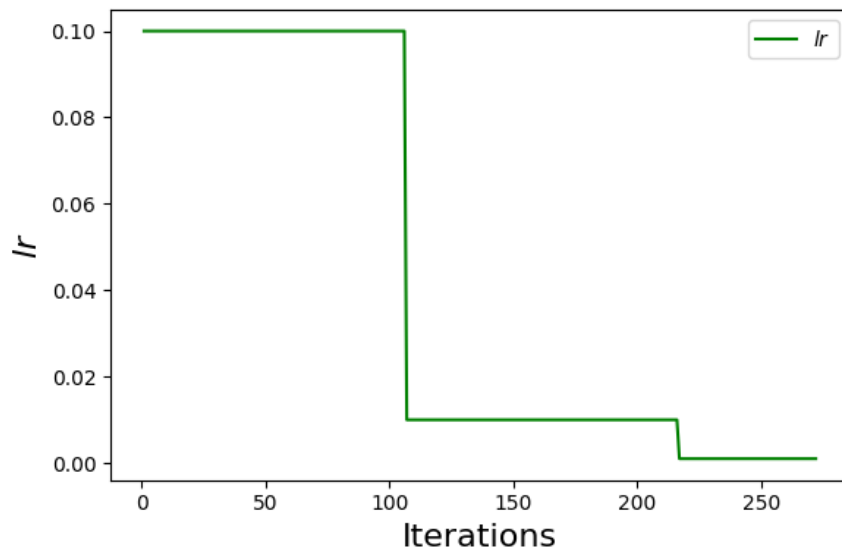
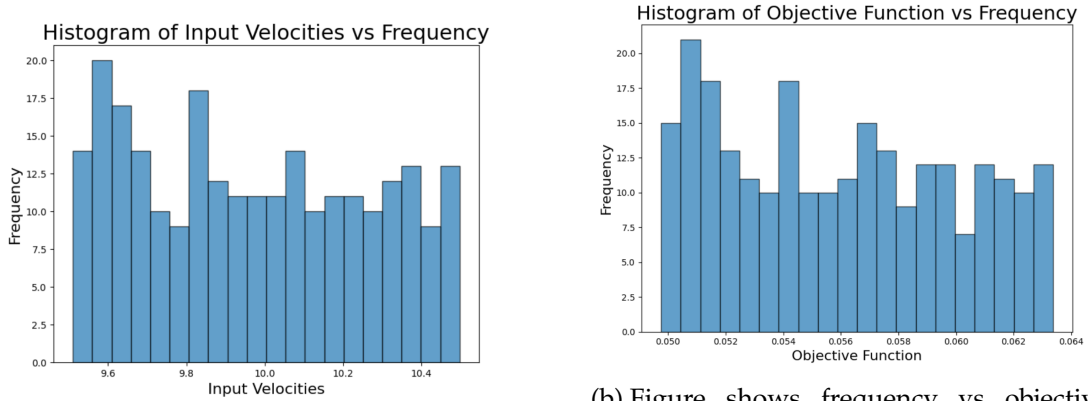


Figure 5.11.: Figure shows the evolution of learning rate with iterations. The effect of the drop-in learning rate is evident in the evolution of  $\mu$  and  $\ln(\sigma)$  as the oscillations reduce and stable convergence is observed.

## 5.2. Stochastic constrained optimization using ScoutND

We discussed in section 4.9, the necessity to consider stochasticity in optimizing engineering systems. In this section, we present the results of optimization under stochastic conditions. For our optimization problem in section 5.1, we used a constant inlet velocity of  $10 \text{ m s}^{-1}$ . We now sample our velocity using a uniform sampling as follows:

$$U_{in} \sim \mathcal{U}(9.5 \text{ m s}^{-1}, 10.5 \text{ m s}^{-1}).. \quad (5.4)$$



(a) Figure shows frequency vs input velocity.

(b) Figure shows frequency vs objective function.

Figure 5.12.: Figure shows input velocity samples(left) and the corresponding objective function value(right) for the PitzDaily problem evaluated for 250 samples.

We wanted to ensure that the choice of input velocity distribution does not introduce high variance in output. To do so, we use a simple sampling study by drawing histograms of input and output. Figure 5.12b shows the output of the PitzDaily problem for 250 sampled velocities in figure 5.12a, using the uniform distribution in equation 5.4. The histogram of the output objective function resembles the input velocity histogram. We see that the variance is not too large. However, it is not small either. We observe from figure 5.12b that the spread of the uniform distribution is approximately 25% of the median. So, we cannot deal with this as a deterministic problem and we must take stochasticity into consideration.

We keep all other simulation settings the same as described in section 5.1. Figures 5.13 and 5.14 show the pressure and velocity contours of the simulation carried out for the optimized values. The final PitzDaily domains for stochastic and non-stochastic problems are similar in the sense that both try to reduce the cross-section area near the outlet which increases the pressure between the domain, reducing the pressure difference between the inlet and the outlet. The reduced cross-section area near the outlet in this problem is more prominent, designed to accommodate for stochastic conditions. Figure 5.16 shows the

## 5. Results

---

pressure plotted on a line along the flow i.e in the X-direction, extending from the inlet and outlet. We observe from figure 5.16 that the pressure difference between the inlet and the outlet is negligible for the optimized curve, which directly influences the objective function.

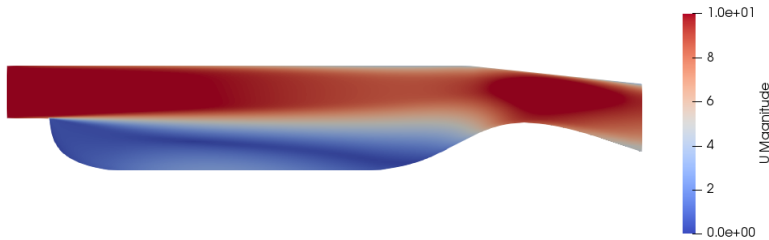


Figure 5.13.: Velocity Contour obtained from simulation of optimized values of stochastic constrained problem.

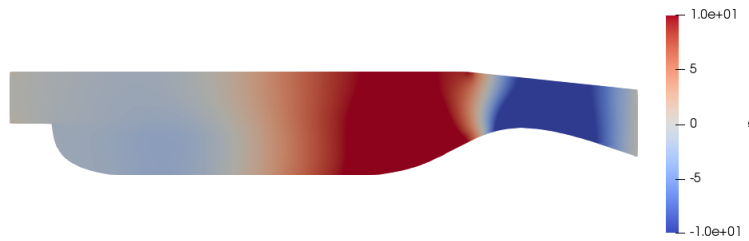


Figure 5.14.: Pressure Contour obtained from simulation of optimized values of stochastic constrained problem. We observe that the pressure at the inlet and outlet boundaries have the same color, indicating very close pressure values.



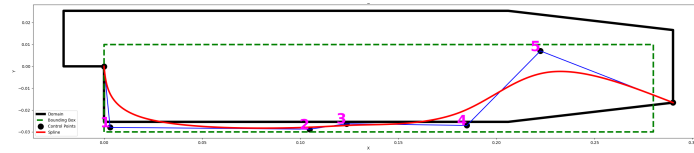


Figure 5.15.: Figure shows the domain and the control points for the stochastic constrained optimization. The numbers next to the point denote the point number.

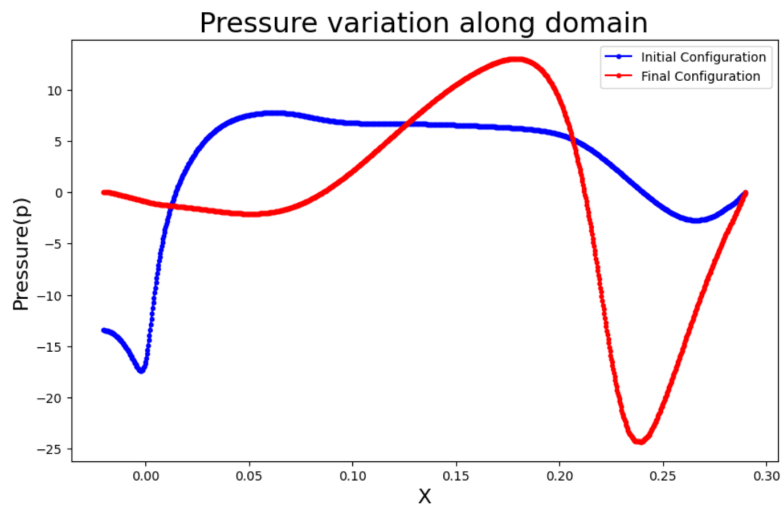
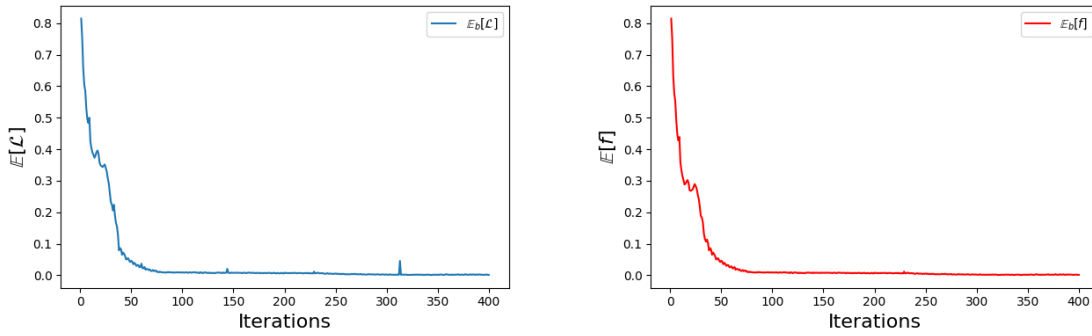


Figure 5.16.: Figure shows the pressure variation along the flow, starting from the inlet boundary to the outlet boundary, for the initial and the final configuration for the stochastic constrained optimization problem. We can observe that the pressure difference between the inlet and the outlet is almost negligible for the final configuration.

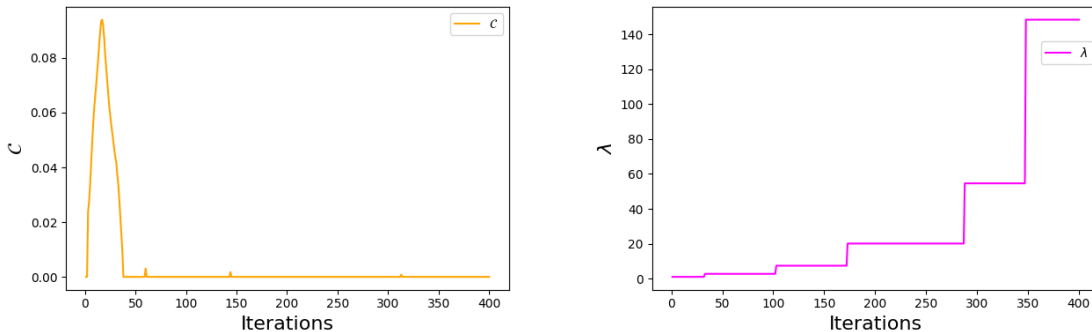
Figures 5.17a and 5.17b show the evolution of the augmented objective function and the objective function with iterations. Similar to section 5.1, we see slight differences in both plots due to lower penalty term and higher variance value initially. We also observe a slight difference in both the plots after 300 iterations due to a violated constraint but dies out quickly due to strict penalisation because of the high penalty term.



(a) Figure shows evolution of  $\mathbb{E}[\mathcal{L}]$  with iteration under stochastic input velocity. (b) Figure shows evolution of  $\mathbb{E}[f]$  with iteration under stochastic input velocity.

Figure 5.17.: Figure shows evolution of  $\mathbb{E}[\mathcal{L}]$  and  $\mathbb{E}[f]$  for stochastic constrained optimization problem with iterations. The two graphs are different initially due to higher constraint violations because of lower value of  $\lambda$  and higher value of  $\ln(\sigma)$ .

Figures 5.18a and 5.18b show the evolution of constraints and the penalty term. We see a similar trend of high constraint value during initial iterations, as in section 5.1, but the constraints die out due to strict penalisation as the penalty term increases. On comparing figure 5.8a with 5.18a, we observe that during initial iterations, constraint violation is higher in the present case due to stochastic conditions.



(a) Figure shows evolution of  $\mathcal{C}$  with iteration under stochastic input velocity. (b) Figure shows evolution of  $\lambda$  with iteration under stochastic input velocity.

Figure 5.18.: Figure shows evolution of  $\mathcal{C}$  and  $\lambda$  for the stochastic constrained optimization problem with iterations. The constraints during the initial iterations is more due to lower value of  $\lambda$ .

Figures 5.19 and 5.20 show the evolution of the mean and the variance of each variable with iterations, respectively. We started with the values shown in equations 5.2 and 5.3.

Similar to section 5.1, we observe from figure 5.20 that  $(\ln(\sigma_{X_5}))$  and  $(\ln(\sigma_{Y_5}))$ , the variance of the point closest to the outlet boundary, i.e. control point 5, which influences the objective function the most, have the least variance values as the solution converges. Moreover, the parameters of the first point,  $(\ln(\sigma_{X_1}))$  and  $(\ln(\sigma_{Y_1}))$ , have a lower variance as well, indicating that the shape of the lower boundary in the region where the fluid goes through expansion, has a notable effect on the objective function. Figure 5.21, shows the evolution of the learning rate with iterations. As explained in section 5.1 we see a similar reduction in oscillations and smooth convergence of  $\mu$  and  $\ln(\sigma)$  as the learning rate drops.

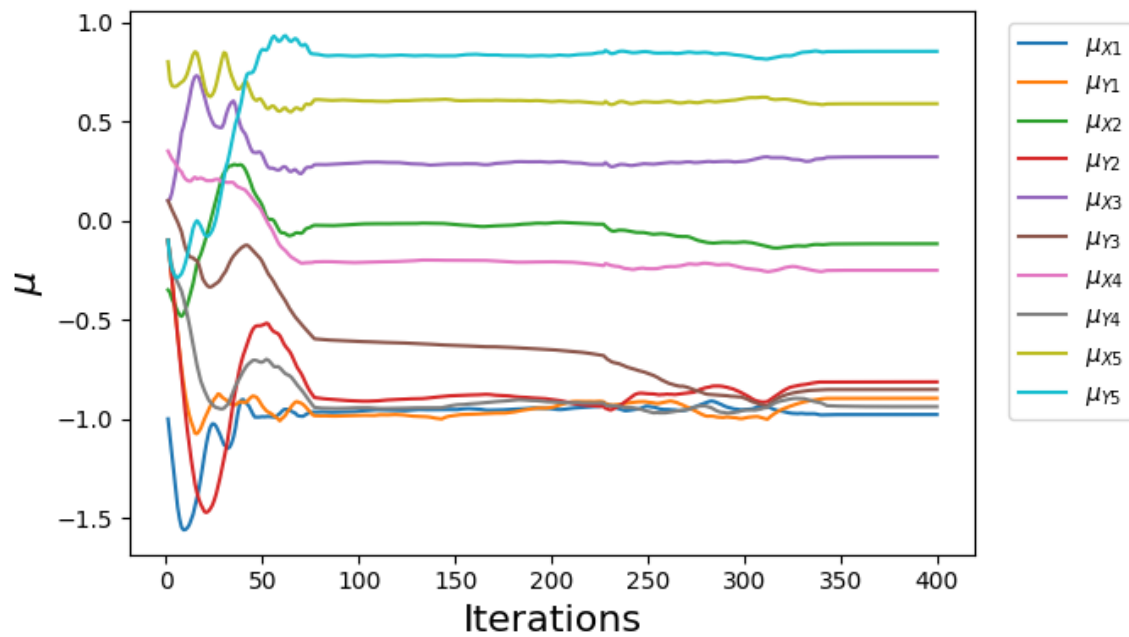


Figure 5.19.: Figure shows the evolution of  $\mu$  of each input variable with iterations for the stochastic constrained optimization problem.

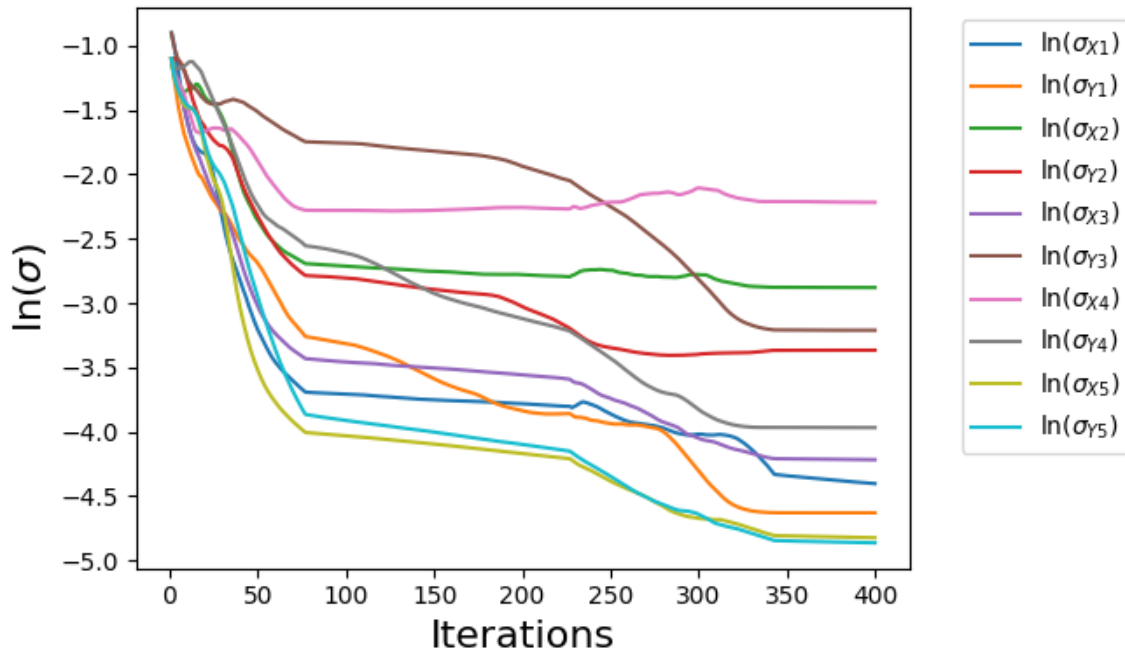


Figure 5.20.: Figure shows the evolution of  $\ln(\sigma)$  of each input variable with iterations for stochastic constrained optimization problem. It is observed that the point near the boundary i.e. point 5, with the minimum cross-section area, has the least variance as these points are critical in the optimization of the objective function. Variance of parameters of Point 1 have the next lowest values, indicating that the shape of the lower boundary in the region where the fluid goes through expansion, has a notable effect on the objective function.

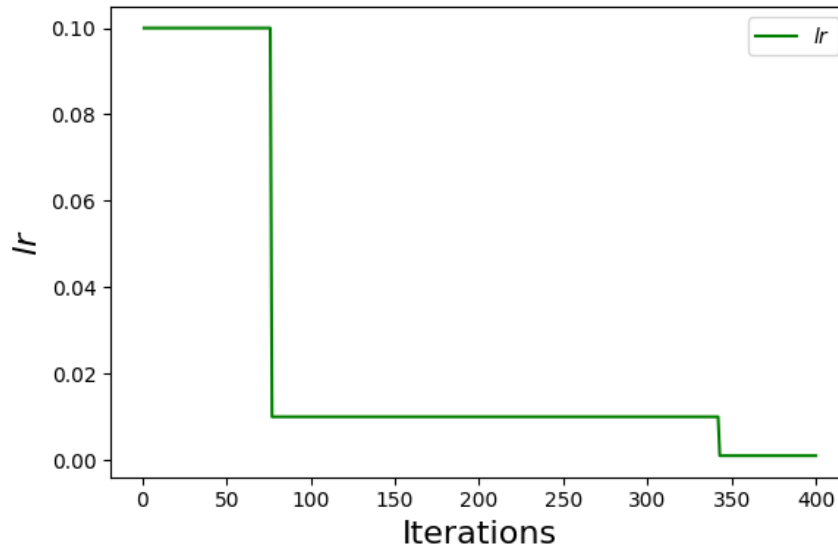


Figure 5.21.: Figure shows evolution of learning rate with iterations for stochastic constrained optimization problem. The effect of drop in learning rate is evident in the evolution of  $\mu$  and  $\ln(\sigma)$  as the oscillations reduce and stable convergence is observed.

### 5.3. Non-stochastic unconstrained optimization using ScoutND

In this section, we present the results of non-stochastic unconstrained optimization explained in section 4.7. We defined the formula for obtaining the transformed control points in equation 4.35. We observed during the initial tests that the values of  $k_x$  and  $k_y$  can influence the convergence. Higher values of  $k_x$  and  $k_y$  make the sigmoid function steeper, and the sampled points can move to regions of vanishing gradients. Moreover, a steeper sigmoid function increases sensitivity to small changes in input. We choose  $k_x = k_y = 3$  to ensure an effective transformation of sampled values. The initial guess,  $\theta = \{\mu, \ln(\sigma)\}$ , used for our unconstrained optimisation problem is defined as,

$$\mu = [-1.0, 0.0, -0.5, 0.0, 0.0, 0.0, 0.5, 0.0, 1.0, 0.0], \quad (5.5)$$

$$\ln(\sigma) = [-1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0]. \quad (5.6)$$

Figures 5.22 and 5.23 show the velocity and pressure contours for the optimized Pitz-Daily problem. The PitzDaily domain for the unconstrained problem is significantly different than the stochastic constrained and the non-stochastic constrained problems presented in section 5.1 and 5.2 but follows the same principle of increasing the pressure near the inlet by restricting the flow in between the domain. Figure 5.25 shows the pressure plotted on a line along the flow i.e. in the X-direction, extending from the inlet to the outlet, for the initial and the optimized values. We observe from figure 5.25 that the pressure difference between the inlet and the outlet is negligible for the optimized curve, which directly influences the objective function. In addition to that, the variation in pressure values along the domain is higher in the present case than the stochastic constrained and non-stochastic constrained problems. We observe from figure 5.23 that the point of the minimum cross-section has shifted from the end of the domain towards the centre. We also observe that in the present case, the minimum cross-section area is the least among the cases discussed previously. We mentioned that the critical factor responsible for minimizing the objective function is the cross-section area. This makes the y-coordinates of the control points a more sensitive parameter than the x-coordinates. The sigmoid transformation has a higher gradient in the middle region, which starts to vanish towards the outer extremes. When dealing with sigmoid transformations, the total gradient of the objective function with respect to the input is represented as a product of two; the derivative of the objective function with respect to the sigmoid transformation function and the derivative of the sigmoid transformation function with respect to the input. The total gradient starts to vanish if this product is zero, i.e. if one of the derivatives from the product becomes zero. At the outer extremes of the sigmoid transformation, the derivative of the sigmoid transformation function with respect to the input starts to vanish, which eventually makes the total gradient approach zero. As the total gradient becomes zero, moving towards the minimum becomes infeasible and the value of our input parameters does not change. This causes the the graphs of our input parameter, i.e. the mean and the variance to flatten

out. Since the y-coordinates are the more sensitive parameters, the optimizer is bound to guide these parameters towards the extremes of the sigmoid function in order to change the cross-section area, and we observe that except for one, all the y-coordinates get stuck in the upper and the lower bounds of the bound box. The x-coordinates, on the other hand, being less sensitive parameters do not move towards the extremes of the sigmoid function in search for the optimum, as changes in x-coordinates do not significantly affect the objective function. Figure 5.24 shows that most of the y-coordinates move towards the extremes of the bound except for Y5, while the x-coordinates do not, affirming our statement that the y-coordinates being the more sensitive parameter, will move towards the upper extreme of the sigmoid function to reduce the cross-section area despite the decreasing gradients of the sigmoid function. As the y-coordinates move to the upper extremes of the sigmoid function, the cross-section area reduces significantly. In order to equalize the pressure at the inlet and outlet boundaries, it was necessary for the point of minimum cross-section area to shift towards the centre, thereby moving the x-coordinates of the minimum cross-section area towards the centre instead of being near the outlet boundary.



Figure 5.22.: Velocity Contour obtained from simulation of optimized values of the non-stochastic unconstrained optimization problem.

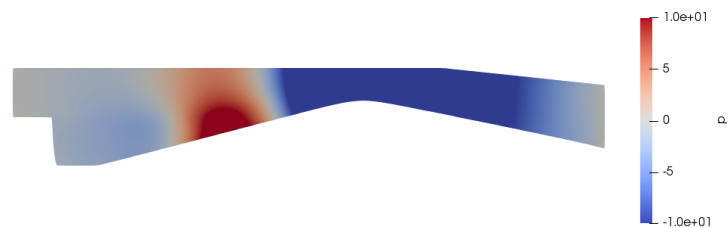


Figure 5.23.: Pressure Contour obtained from simulation of optimized values of the non-stochastic unconstrained optimization problem. We observe that the pressure at the inlet and outlet boundaries have the same color, indicating very close pressure values.

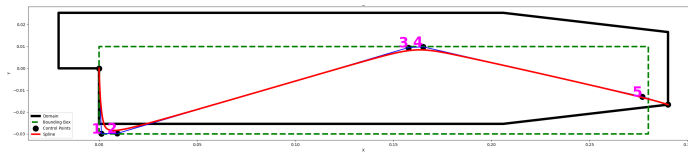


Figure 5.24.: Figure shows the domain and the control points of the non-stochastic unconstrained optimization problem. The numbers next to the point denote the point number. We observe that most of the y-coordinates are stuck in the upper and the lower bound except for Y5, which are the regions where gradients vanish.

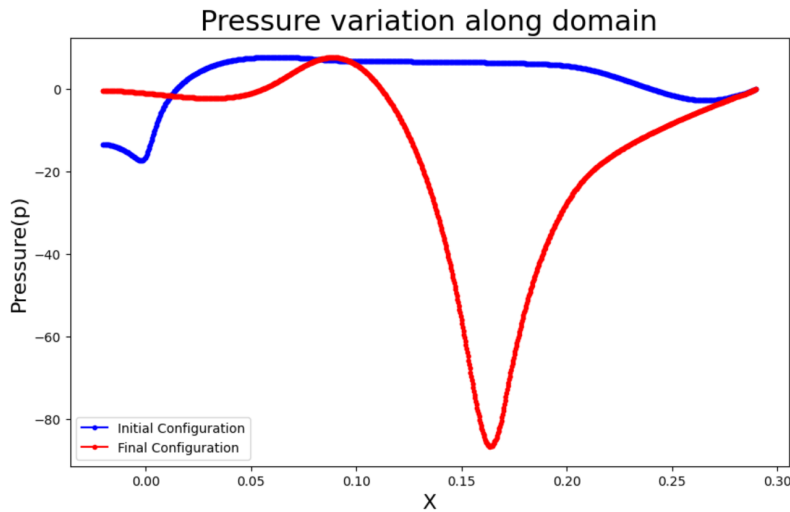
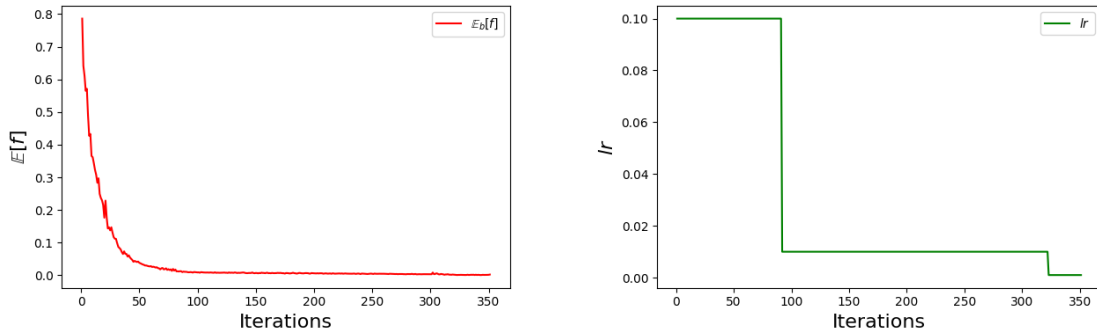


Figure 5.25.: Figure shows the pressure variation along the flow, starting from the inlet boundary to the outlet boundary, for the initial and the final configurations for the non-stochastic unconstrained optimization problem. We can observe that the pressure difference between the inlet and the outlet is almost negligible for the final configuration. The variation in pressure is higher than the Constrained problems.

Figures 5.26a and 5.26b show the evolution of objective function and learning rate with iterations. We observe that the objective function converges faster at higher learning rates. Similar to section 5.1, we see a reduction in oscillations and observe smooth convergence of  $\mu$  and  $\ln(\sigma)$  as the learning rate drops.





(a) Figure shows evolution of  $\mathbb{E}[f]$  with iteration.

(b) Figure shows evolution of learning rate with iteration.

Figure 5.26.: Figure shows evolution of  $\mathbb{E}[f]$  and learning rate with iterations for the non-stochastic unconstrained optimization. The effect of drop in learning rate is evident in the evolution of  $\mu$  and  $\ln(\sigma)$  as the oscillations reduce and stable convergence is observed.

Figures 5.27 and 5.28 show the evolution of mean and the variance with iterations. We observed in the constrained optimization problems that the points near the minimum cross-section area have the least variance values at the end of the optimization, as change in these parameters significantly affect the objective function. In the present case, points 3 and 4, which are present near the minimum cross-section area should have followed the same trend. Instead, as the solution progressed, the y-coordinates of these points were stuck in regions of vanishing gradients. For this reason, further changes in these values have a negligible effect on the objective function and their variances do not reduce as much as in the constrained optimization problem. We also observe that the variance ( $\ln(\sigma_{Y5})$ ) is the least. This is expected because this parameter is not stuck in the regions of vanishing gradients and is in the proximity of the outlet boundary, which can significantly affect the objective function. Moreover, as we observed in previous cases, the points near the outlet can significantly affect the objective function. The plots lying in the middle of the region of figure 5.27 have the least variance, as seen in figure 5.28. This is because these parameters do not get stuck in regions of vanishing gradients. Moreover, we observe that in the constrained optimization problems, the point near the expansion area also had less variance values at the end of optimization. This is not observed in the unconstrained cases as this point moves to the regions of vanishing gradients and has a high variance at the end of the optimization. The non-stochastic unconstrained problem converged to a completely different optimum, indicating that the shape optimization problem is multi-modal.

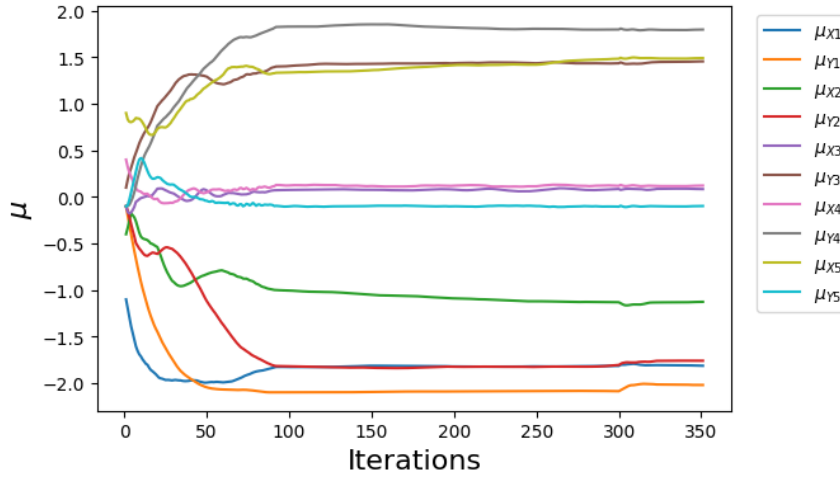


Figure 5.27.: Figure shows evolution of  $\mu$  of each input variable with iterations for the non-stochastic unconstrained optimization.

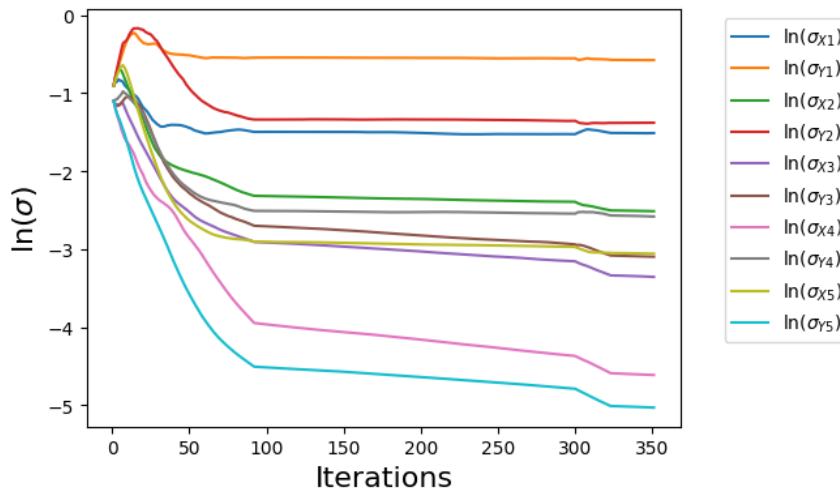


Figure 5.28.: Figure shows evolution of  $\ln(\sigma)$  of each input variable with iterations for the non-stochastic unconstrained optimisation problem. It is observed that  $\ln(\sigma_{Y3})$  and  $\ln(\sigma_{Y4})$ , despite being the variances of the points near the minimum cross-section area, do not have much less variance values as they get stuck in regions of vanishing gradients. The variance  $\ln(\sigma_{Y5})$  has the least variance due to its proximity to the outlet boundary and its presence in the middle region of the sigmoid function. Unlike constrained optimization problems, the point near the expansion region has the highest variance.

## 5.4. Stochastic unconstrained optimization using ScoutND

In this section, we present the results of unconstrained optimization under stochastic conditions. We begin with the initial guess defined in equation 5.5 and 5.6. Figures 5.29 and 5.30 show the velocity and pressure contours for the optimized PitzDaily problem. The PitzDaily domain for the stochastic unconstrained problem is similar to the results in the constrained problems. Figure 5.31 shows the position of control points for the present case. Similar to the non-stochastic unconstrained problem where most of the points move towards the extremes of the sigmoid function, the control points in the stochastic problem follow the same trend but the introduction of stochasticity leads the optimizer to a different profile, designed to accommodate for stochastic conditions. Figure 5.32 shows the pressure plotted on a line along the flow i.e. in the X-direction, extending from the inlet to the outlet, for the initial and the optimized values. We observe that the pressure difference between the inlet and the outlet is negligible for the optimized curve, which directly influences the objective function.

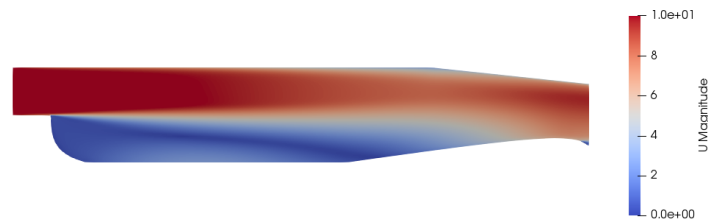


Figure 5.29.: Velocity Contour obtained from simulation of optimized values of the stochastic unconstrained optimization problem.

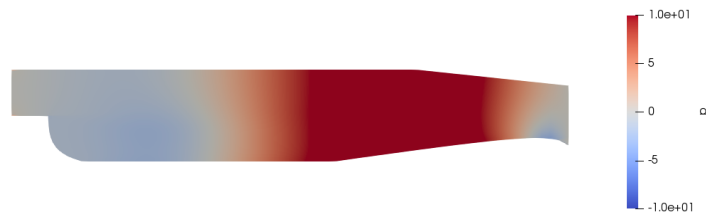


Figure 5.30.: Pressure Contour obtained from simulation of optimized values of the stochastic unconstrained optimization problem. We observe that the pressure at the inlet and outlet boundaries have the same color, indicating very close pressure values.

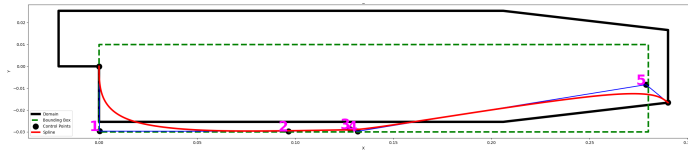


Figure 5.31.: Figure shows the domain and the control points of the stochastic unconstrained optimization problem. The numbers next to the point denote the point number. We observe that many of the y-coordinates are stuck in regions of vanishing gradients except point 5.

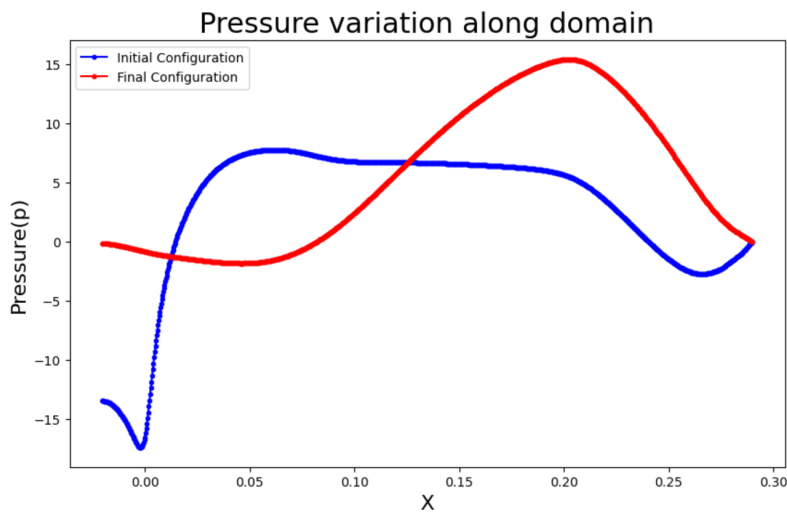
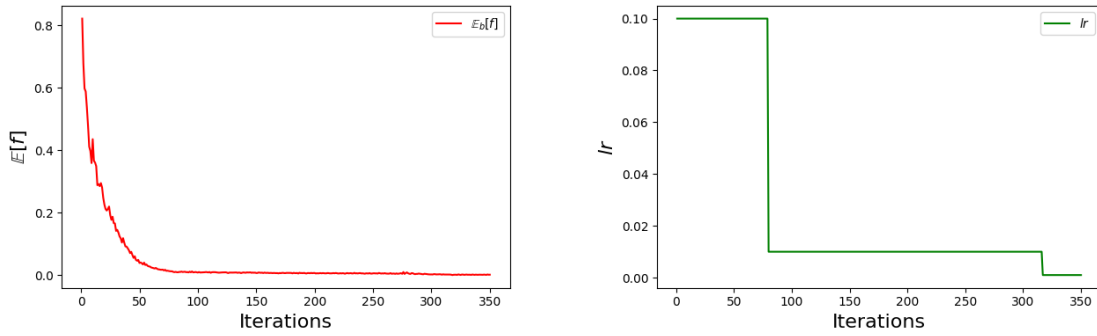


Figure 5.32.: Figure shows the pressure variation along the flow, starting from the inlet boundary to the outlet boundary, for the initial and the final configurations for the stochastic unconstrained optimization problem. We can observe that the pressure difference between the inlet and the outlet is almost negligible for the final configuration.

Figures 5.33a and 5.33b show the evolution of objective function and learning rate with iterations. We observe that the objective function converges faster at higher learning rates. Similar to section 5.1, we observe a reduction in oscillations and smooth convergence of  $\mu$  and  $\ln(\sigma)$  as the learning rate drops.



(a) Figure shows evolution of  $\mathbb{E}[f]$  with iteration.

(b) Figure shows the evolution of learning rate with iteration.

Figure 5.33.: Figure shows evolution of  $\mathbb{E}[f]$  and learning rate with iterations for the stochastic unconstrained optimization. The effect of drop in learning rate is evident in the evolution of  $\mu$  and  $\ln(\sigma)$  as the oscillations reduce and stable convergence is observed.

Figures 5.34 and 5.35 show the evolution of the mean and the variance with iterations. We observe that the parameters that do not get stuck in the extremes of the sigmoid function have less variance values as compared to the ones that get stuck in those regions. Moreover,  $(\ln(\sigma_{Y_5}))$  has the least value because this parameter is not stuck in regions of vanishing gradients and is in the proximity of the outlet boundary. We also observe that, unlike the constrained optimization problems, the point near the expansion area has high variance values at the end of optimization as it gets stuck in regions of vanishing gradients. Although the present case is very similar to the non-stochastic unconstrained problem, the convergence of variance is slower in the present case because of parameters getting stuck in regions of vanishing gradients.

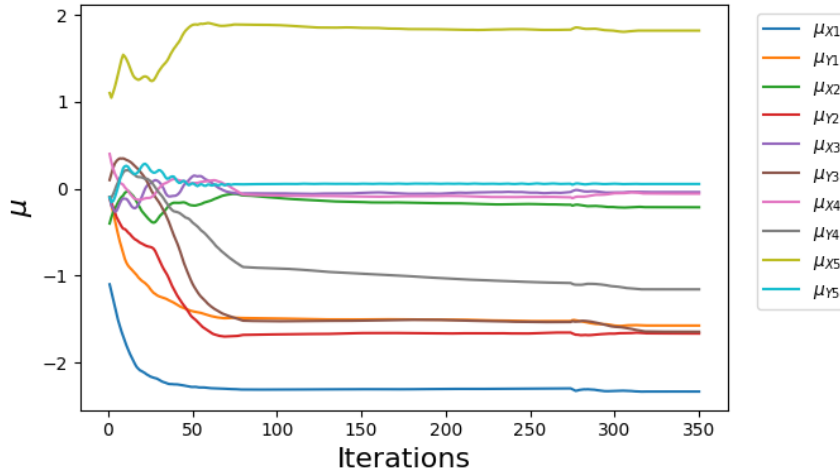


Figure 5.34.: Figure shows evolution of  $\mu$  of each input variable with iterations for stochastic unconstrained optimisation problem.

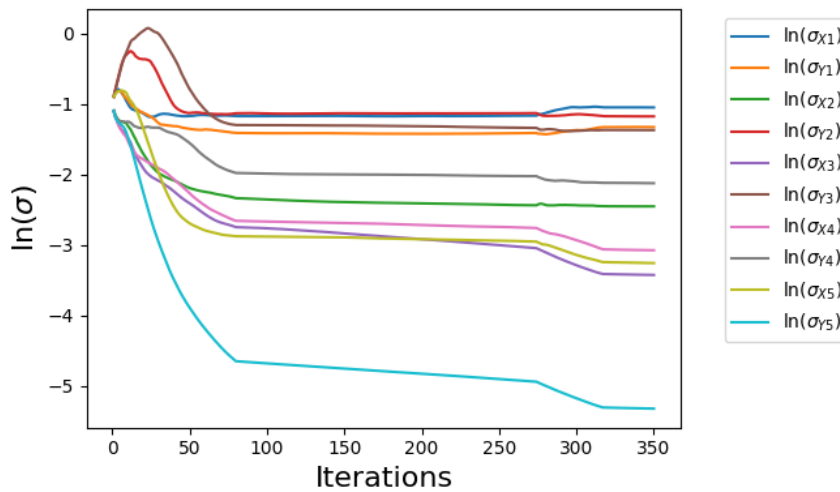


Figure 5.35.: Figure shows the evolution of  $\ln(\sigma)$  of each input variable with iterations for stochastic unconstrained optimization problem. We observe that the points that do not reach the extremes of the sigmoid have low variance. It is observed that the parameter of the point lying near the outlet boundary, where the cross-section area is minimum, has the least variance as this point is critical in the optimization of the objective function. We also observe that, unlike the constrained optimization problems, the point near the expansion area has high variance values at the end of optimization as it gets stuck in regions of vanishing gradients.

## 5.5. Comparative Analysis

In this section, we present a comparison between different simulations of the PitzDaily problem carried out using ScoutND. Figure 5.36 shows the evolution of the objective function with iterations of different optimizations carried out using ScoutND. We observe that the non-stochastic constrained problem converges fastest. We observe some peaks in the plot after 50 iterations as the optimizer searches through the domain, but it converges the fastest as the solution progresses. The constrained stochastic has the slowest convergence among others. This is expected as the introduction of stochasticity allows the optimizer to explore the domain more effectively for the minimum, which delays convergence. The non-stochastic unconstrained converges faster than stochastic unconstrained problem initially but their convergence plots gradually align. This is expected because as the solution progresses, many parameters of the unconstrained problem get stuck in the regions of vanishing gradients.

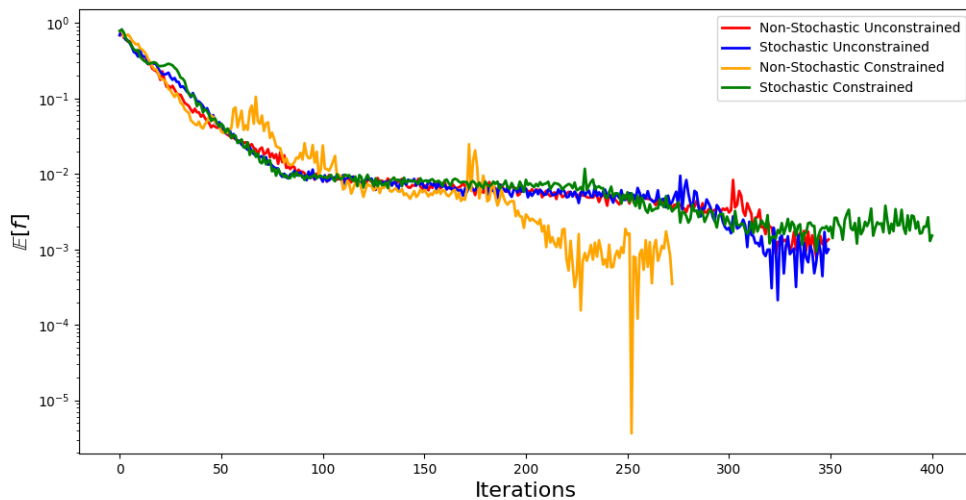


Figure 5.36.: Figure shows a comparison of stochastic and non-stochastic, constrained and unconstrained optimization problems using ScoutND. We observe that the non-stochastic constrained problem converges fastest. The constrained stochastic has the slowest convergence among others. This is expected due to the introduction of stochasticity. The non-stochastic unconstrained converges faster than stochastic unconstrained initially but their convergence rate gradually aligned. This is expected because as the solution progresses, many parameters of the unconstrained problem get stuck in the regions of vanishing gradients.

## 5. Results

---

Table 5.1 highlights the initial and optimized values for the non-stochastic optimizations carried out using ScoutND. We observe that the non-stochastic unconstrained problem has the minimum value, despite many of its parameters getting stuck in regions of vanishing gradients. Table 5.1 highlights the initial and optimized values for the stochastic optimizations carried out using ScoutND. We observe that the stochastic unconstrained problem has a lesser value, but it is very close to the stochastic constrained problem.

<i>Optimizer</i>	<i>Initial Value</i>	<i>Optimised Value(<math>\xi</math>)</i>	<i>Percentage Reduction</i>
Non-stochastic constrained	0.14047	0.00065	99.53
Non-stochastic unconstrained	0.12364	0.00007	99.94

Table 5.1.: Table shows a comparison of initial and optimized objective values for non-stochastic problems.

<i>Optimizer</i>	<i>Initial Value</i>	<i>Optimised Value(<math>\xi</math>)</i>	<i>Percentage Reduction</i>
Stochastic constrained	0.14047	0.00058	99.58
Stochastic unconstrained	0.12364	0.00045	99.63

Table 5.2.: Table shows a comparison of initial and optimized function values for stochastic problems.

We compare the results from the constrained optimization problem from ScoutND with Nelder-Mead, COBYLA, SLSQP and L-BFGS-B. We use non-normalized objective function values for our comparison. To compare ScoutND with other optimizers, we evaluate the function at the mean arrays obtained at each iteration. Table 5.3 summarizes the results of each optimizer for non-stochastic constrained problem. Among the listed optimizers, Nelder-Mead and L-BFGS-B do not support constraints. Instead of using constraints, we use bounds defined in equation 4.33. Figure 5.37 shows the convergence of the error with iterations. We observe that the Nelder-Mead runs upto significant number of iterations but fails to converge effectively. The COBYLA, SLSQP and L-BFGS-B methods performed significantly better, bringing the solution very close to the true optimum value. ScoutND significantly reduced the function value. Our optimization problem has 10 dimensions which makes it a medium dimensionality problem. It has been stated in the work of [25] that COBYLA performs better than ScoutND for relatively low dimensional problems but fails in the case of high dimensional problems.



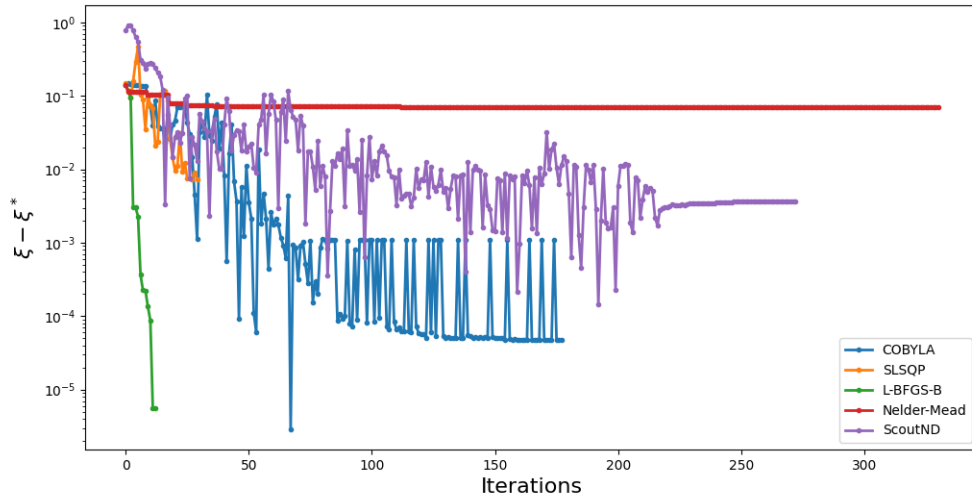


Figure 5.37.: Figure shows the error vs iterations for different optimizers for non-stochastic constrained problems. COBYLA, SLSQP, L-BFGS-B, and ScoutND effectively optimize the problem but Nelder-Mead fails.

We compare the results of stochastic constrained optimization of different optimizers with ScoutND. Table 5.3 summarizes the results of each optimizer for the stochastic constrained problem. Although, stochastic optimization of engineering systems is necessary, as there are always some uncertain parameters but in general, they are often expected to operate at some fixed parameter. We use the mean of our stochastic velocity distribution as a baseline, to compare the results of different stochastic optimization problems. We simulate our function using the control points obtained at each iteration of the optimizers, at the mean velocity. We observe that Nelder-Mead, SLSQP, and L-BFGS-B fail to optimize the PitzDaily problem under stochastic conditions. This is because the objective function is noisy, so the approximated Hessian does not guarantee a positive definite matrix. SLSQP requires gradient information for convergence, which is difficult to obtain in the case of noisy functions. COBYLA performs significantly better than the aforementioned optimizers. ScoutND proves to be the most effective among all optimizers under stochastic conditions, minimizing the function very close to the true optimum. Figure 5.38 presents the results of stochastic constrained optimization of different optimizers. Table 5.4 shows the profiles of the PitzDaily problem for different optimizers. We observe different profiles for each optimizer, indicating that this optimization problem is multimodal.

## 5. Results

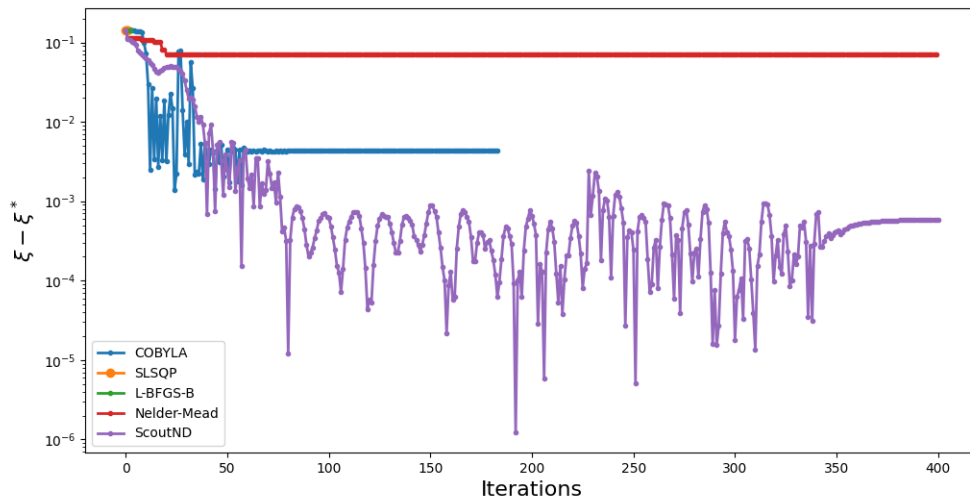


Figure 5.38.: Figure shows the Error vs Iterations for stochastic constrained problems. SLSQP, L-BFGS-B and Nelder-Mead fail to optimize the problem but COBYLA performs significantly better. ScoutND outperforms all other optimizers in case of stochastic problems.

<i>Optimizer</i>	<i>Initial Value</i>	<i>Optimised Value Non – Stochastic</i>	<i>Percentage Reduction Non – Stochastic</i>	<i>Optimised Value Stochastic</i>	<i>Percentage Reduction Stochastic</i>
ScoutND	0.14047	0.00065	99.53	0.00058	99.58
Nelder-Mead	0.14047	0.07060	49.74	0.07164	49.0
COBYLA	0.14047	0.000005	99.99	0.00428	96.9
SLSQP	0.14047	0.000001	99.99	0.14047	0.0
L-BFGS-B	0.14047	0.000005	99.99	0.14047	0.0

Table 5.3.: Table shows a comparison of initial and optimized function values for different optimizers for stochastic and non-stochastic problems.

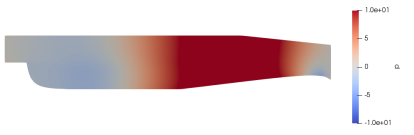
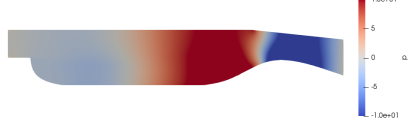




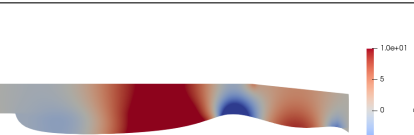



Optimizer	<i>non – stochastic constrained</i>	<i>stochastic constrained</i>
ScoutND		
Nelder-Mead		
COBYLA		
SLSQP		
L-BFGS-B		

Table 5.4.: Table shows the optimized profiles for non-stochastic and stochastic optimization problems of different optimizers. Different profiles indicate that this optimization problem is multimodal.



## 6. Conclusion

The present work focuses on the optimization of engineering systems using black-box optimization. ScoutND, a black-box optimization algorithm has been proven to perform well for high dimensional and stochastic problems. The present work extends the use of ScoutND to optimize engineering systems. We perform shape optimization of the Pitz-Daily problem, a test case CFD problem. We used OpenFOAM, an open-source CFD software to simulate the CFD problem. We used the *simpleFoam* solver, an OpenFOAM library to solve steady state, incompressible, turbulent flows, and modified it to accommodate for the shape optimization of the PitzDaily problem. To solve the problem, we modify the lower boundary of the original PitzDaily problem. We define a bound around the lower boundary. Within this bound, we generate a set of control points, which are used to generate a curve that replaces the lower boundary in the original PitzDaily problem. Any point outside this bound is treated as a constraint violation in our optimization problem. These control points are provided as an input to the optimizer. We use Python's PyFoam library to link our optimizer to OpenFOAM. ScoutND uses Monte Carlo simulation to estimate the gradient. This allows us to evaluate the functions in parallel which significantly reduces the computation time. We proposed methods to implement constrained and unconstrained optimization of the PitzDaily problem under deterministic and stochastic conditions. In the case of non-stochastic constrained optimization, we observed that the convergence was the fastest. Moreover, the control points near the minimum cross-section area have the least variance values, implying that these points have the most significant effect in minimizing the problem. For the stochastic constrained optimization, we observed a slower convergence when compared with the non-stochastic case due to the introduction of stochasticity. The shape profile also turned out to be slightly different. Similar to the non-stochastic case, we observed that the points near the minimum cross-section area have the least variances. For the unconstrained optimization problem, we used sigmoid transformation to convert our problem to an unconstrained optimization problem. The sigmoid transformation has an issue of vanishing gradients, which causes the control points to get stuck into regions of vanishing gradients. We observed this issue in our unconstrained optimization problem where some input parameters were stuck in regions of vanishing gradients which slowed down convergence(explained in section 5.3). The stochastic unconstrained optimization faces a similar issue of parameters getting stuck in regions of vanishing gradients, but the introduction of stochasticity leads the optimizer to a different solution. From the simulations, we observed that a control point's proximity to the outlet boundary, and the minimum cross-section area, is a critical factor in minimizing the objective function. A general observation is that the parameters of

these points have the highest convergence. The point near the region of expansion plays a critical role in minimizing the objective function. This was evident in the constrained minimization problems. However, in the case of unconstrained problems, this effect was not observed because the point was stuck in regions of vanishing gradients. Tables 5.1 and 5.2 compare the different optimization problems carried out using ScoutND. We compare our results from ScoutND with other optimizers such as Nelder-Mead, SLSQP, COBYLA, and L-BFGS-B for stochastic and non-stochastic conditions. We observe that SLSQP, COBYLA, and L-BFGS-B minimize the problem effectively with a 99.99% reduction in the objective function under non-stochastic conditions however, Nelder-Mead turns out to be ineffective with a 49.74% reduction. ScoutND effectively minimizes the problem with a 99.53% reduction. Under stochastic conditions, ScoutND outperforms all other optimizers with a 99.58% reduction. COBYLA reduces the objective function by 96.9%. SLSQP and L-BFGS-B completely fail to optimize the problem and Nelder-Mead proves to be ineffective with 49.0% reduction. The optimizers that successfully minimized the problem led to different shape profiles of the PitzDaily problem, indicating that the present shape optimization problem is multimodal. To conclude the present work, we found that ScoutND is effective in optimizing high-dimensional problems effectively. Moreover, for engineering systems, where stochasticity is an important factor to consider, ScoutND proves to be the most effective.

# List of Figures

3.1.	Figure shows $f(x) =  x $ and the approximation to the function using different variance values. Lower variance values approximate the function closely as compared to higher variance values. . . . .	12
3.2.	Figure shows Monte Carlo(left) samples and Quasi-Monte Carlo samples(right) generated using Sobol sequences . . . . .	18
4.1.	Figure shows the domain of the PitzDaily problem. . . . .	22
4.2.	Figure shows a staggered grid arrangement used for SIMPLE algorithm. Staggered grid mitigates the <i>checkerboard</i> board problem. . . . .	23
4.3.	Figure shows different levels in Multi-Grid from the finest(left) to the coarsest(right). . . . .	28
4.4.	Figure shows a V-cycle Multi-Grid . . . . .	29
4.5.	Figure shows a W-cycle Multi-Grid . . . . .	29
4.6.	Figure shows a Full Multi-Grid . . . . .	30
4.7.	Flowchart representing the OpenFOAM code structure for the PitzDaily problem . . . . .	30
4.10.	Figure shows the STL curve generated using the Catmull-Clark Subdivision curve. . . . .	32
4.8.	Figure shows the new PitzDaily problem. The black line represents the original PitzDaily problem. The green box represents the rectangular bound within which the control points are generated. The blue points represent the control points. These points are used to generate the red curve which is the generated Catmull-Clark subdivision curve. . . . .	33
4.9.	Figure shows the final PitzDaily domain on which simulation is carried out for the shape optimization problem. . . . .	33
4.11.	Figure shows the STL curve mapped on the original PitzDaily problem. . . . .	34
4.12.	Figure shows the final simulation on the New Generated Domain . . . . .	34
4.13.	Figure (4.13a) shows the control point inside the domain while (4.13b) shows the control point outside the domain. The value of constraint using equation (4.34) is equal to zero in case (4.13a) and greater than zero in case (4.13b). . . . .	36
4.14.	Figure shows the control points going out of bound which causes issue in simulating the PitzDaily problem as the domain splits in between, making the simulation infeasible. . . . .	36
4.15.	Sigmoid Function . . . . .	37

5.1. Velocity Contour obtained from simulation using initial values from equation 5.2. . . . .	44
5.2. Pressure contour obtained from simulation using initial values from equation 5.2. . . . .	44
5.3. Velocity Contour obtained from simulation of optimized values of non-stochastic constrained optimization problem. . . . .	45
5.4. Pressure Contour obtained from simulation of optimized values of non-stochastic constrained optimization problem. We observe that the pressure at the inlet and outlet boundaries have the same color, indicating very close pressure values. . . . .	45
5.5. Figure shows the domain and the control points for the non-stochastic constrained optimization problem. The numbers next to the point denote the point number. . . . .	46
5.6. Figure shows the pressure variation along the flow, starting from the inlet boundary to the outlet boundary, for the initial and the final configuration for the non-stochastic constrained optimization problem. We can observe that the pressure difference between the inlet and the outlet is almost negligible for the final configuration. . . . .	46
5.7. Figure shows evolution of $\mathbb{E}[\mathcal{L}]$ and $\mathbb{E}[f]$ with iterations for the non-stochastic constrained optimization problem. The two graphs are different initially as the optimizer searches for the minimum throughout the domain because of lower value of $\lambda$ and higher value of $\ln(\sigma)$ ,but align as the solution converges. . . . .	47
5.8. Figure shows evolution of constraint ( $\mathcal{C}$ ) and the penalty term ( $\lambda$ ) for the non-stochastic constrained optimization problem with iterations. The constraints during the initial iterations have a higher magnitude due to lower value of $\lambda$ . . . . .	47
5.9. Figure shows the evolution of $\mu$ of each input variable for the non-stochastic constrained optimization problem. . . . .	48
5.10. Figure shows the evolution of $\ln(\sigma)$ of each input variable with iterations for the non-stochastic constrained optimization problem. It is observed that the points near the boundary, with minimum cross-section area, have the least variance as these points are critical in the optimization of the objective function. The point near the expansion area also has a notable effect on the objective function. . . . .	49
5.11. Figure shows the evolution of learning rate with iterations. The effect of the drop-in learning rate is evident in the evolution of $\mu$ and $\ln(\sigma)$ as the oscillations reduce and stable convergence is observed. . . . .	50
5.12. Figure shows input velocity samples(left) and the corresponding objective function value(right) for the PitzDaily problem evaluated for 250 samples. . . . .	51
5.13. Velocity Contour obtained from simulation of optimized values of stochastic constrained problem. . . . .	52



---

5.14. Pressure Contour obtained from simulation of optimized values of stochastic constrained problem. We observe that the pressure at the inlet and outlet boundaries have the same color, indicating very close pressure values. . . .	52
5.15. Figure shows the domain and the control points for the stochastic constrained optimization. The numbers next to the point denote the point number. . . .	53
5.16. Figure shows the pressure variation along the flow, starting from the inlet boundary to the outlet boundary, for the initial and the final configuration for the stochastic constrained optimization problem. We can observe that the pressure difference between the inlet and the outlet is almost negligible for the final configuration. . . . .	53
5.17. Figure shows evolution of $\mathbb{E}[\mathcal{L}]$ and $\mathbb{E}[f]$ for stochastic constrained optimization problem with iterations. The two graphs are different initially due to higher constraint violations because of lower value of $\lambda$ and higher value of $\ln(\sigma)$ . . . . .	54
5.18. Figure shows evolution of $\mathcal{C}$ and $\lambda$ for the stochastic constrained optimization problem with iterations. The constraints during the initial iterations is more due to lower value of $\lambda$ . . . . .	54
5.19. Figure shows the evolution of $\mu$ of each input variable with iterations for the stochastic constrained optimization problem. . . . .	55
5.20. Figure shows the evolution of $\ln(\sigma)$ of each input variable with iterations for stochastic constrained optimization problem. It is observed that the point near the boundary i.e. point 5, with the minimum cross-section area, has the least variance as these points are critical in the optimization of the objective function. Variance of parameters of Point 1 have the next lowest values, indicating that the shape of the lower boundary in the region where the fluid goes through expansion, has a notable effect on the objective function. . . . .	56
5.21. Figure shows evolution of learning rate with iterations for stochastic constrained optimization problem. The effect of drop in learning rate is evident in the evolution of $\mu$ and $\ln(\sigma)$ as the oscillations reduce and stable convergence is observed. . . . .	57
5.22. Velocity Contour obtained from simulation of optimized values of the non-stochastic unconstrained optimization problem. . . . .	59
5.23. Pressure Contour obtained from simulation of optimized values of the non-stochastic unconstrained optimization problem. We observe that the pressure at the inlet and outlet boundaries have the same color, indicating very close pressure values. . . . .	59
5.24. Figure shows the domain and the control points of the non-stochastic unconstrained optimization problem. The numbers next to the point denote the point number. We observe that most of the y-coordinates are stuck in the upper and the lower bound except for Y5, which are the regions where gradients vanish. . . . .	60

5.25. Figure shows the pressure variation along the flow, starting from the inlet boundary to the outlet boundary, for the initial and the final configurations for the non-stochastic unconstrained optimization problem. We can observe that the pressure difference between the inlet and the outlet is almost negligible for the final configuration. The variation in pressure is higher than the Constrained problems. . . . .	60
5.26. Figure shows evolution of $\mathbb{E}[f]$ and learning rate with iterations for the non-stochastic unconstrained optimization. The effect of drop in learning rate is evident in the evolution of $\mu$ and $\ln(\sigma)$ as the oscillations reduce and stable convergence is observed. . . . .	61
5.27. Figure shows evolution of $\mu$ of each input variable with iterations for the non-stochastic unconstrained optimization. . . . .	62
5.28. Figure shows evolution of $\ln(\sigma)$ of each input variable with iterations for the non-stochastic unconstrained optimisation problem. It is observed that $\ln(\sigma_{Y3})$ and $\ln(\sigma_{Y4})$ , despite being the variances of the points near the minimum cross-section area, do not have much less variance values as they get stuck in regions of vanishing gradients. The variance $\ln(\sigma_{Y5})$ has the least variance due to its proximity to the outlet boundary and its presence in the middle region of the sigmoid function. Unlike constrained optimization problems, the point near the expansion region has the highest variance. . . . .	62
5.29. Velocity Contour obtained from simulation of optimized values of the stochastic unconstrained optimization problem. . . . .	63
5.30. Pressure Contour obtained from simulation of optimized values of the stochastic unconstrained optimization problem. We observe that the pressure at the inlet and outlet boundaries have the same color, indicating very close pressure values. . . . .	63
5.31. Figure shows the domain and the control points of the stochastic unconstrained optimization problem. The numbers next to the point denote the point number. We observe that many of the y-coordinates are stuck in regions of vanishing gradients except point 5. . . . .	64
5.32. Figure shows the pressure variation along the flow, starting from the inlet boundary to the outlet boundary, for the initial and the final configurations for the stochastic unconstrained optimization problem. We can observe that the pressure difference between the inlet and the outlet is almost negligible for the final configuration. . . . .	64
5.33. Figure shows evolution of $\mathbb{E}[f]$ and learning rate with iterations for the stochastic unconstrained optimization. The effect of drop in learning rate is evident in the evolution of $\mu$ and $\ln(\sigma)$ as the oscillations reduce and stable convergence is observed. . . . .	65
5.34. Figure shows evolution of $\mu$ of each input variable with iterations for stochastic unconstrained optimisation problem. . . . .	66

---

5.35. Figure shows the evolution of $\ln(\sigma)$ of each input variable with iterations for stochastic unconstrained optimization problem. We observe that the points that do not reach the extremes of the sigmoid have low variance. It is observed that the parameter of the point lying near the outlet boundary, where the cross-section area is minimum, has the least variance as this point is critical in the optimization of the objective function. We also observe that, unlike the constrained optimization problems, the point near the expansion area has high variance values at the end of optimization as it gets stuck in regions of vanishing gradients. . . . .	66
5.36. Figure shows a comparison of stochastic and non-stochastic, constrained and unconstrained optimization problems using ScoutND. We observe that the non-stochastic constrained problem converges fastest. The constrained stochastic has the slowest convergence among others. This is expected due to the introduction of stochasticity. The non-stochastic unconstrained converges faster than stochastic unconstrained initially but their convergence rate gradually aligned. This is expected because as the solution progresses, many parameters of the unconstrained problem get stuck in the regions of vanishing gradients. . . . .	67
5.37. Figure shows the error vs iterations for different optimizers for non-stochastic constrained problems. COBYLA, SLSQP, L-BFGS-B, and ScoutND effectively optimize the problem but Nelder-Mead fails. . . . .	69
5.38. Figure shows the Error vs Iterations for stochastic constrained problems. SLSQP, L-BFGS-B and Nelder-Mead fail to optimize the problem but COBYLA performs significantly better. ScoutND outperforms all other optimizers in case of stochastic problems. . . . .	70



# List of Tables

3.1. Values of constants in Adam Optimizer . . . . .	15
4.1. Table shows the simulation settings for the PitzDaily shape optimization problem. . . . .	22
4.2. Values of constants in $k - \epsilon$ equations . . . . .	28
5.1. Table shows a comparison of initial and optimized objective values for non-stochastic problems. . . . .	68
5.2. Table shows a comparison of initial and optimized function values for stochastic problems. . . . .	68
5.3. Table shows a comparison of initial and optimized function values for different optimizers for stochastic and non-stochastic problems. . . . .	70
5.4. Table shows the optimized profiles for non-stochastic and stochastic optimization problems of different optimizers. Different profiles indicate that this optimization problem is multimodal. . . . .	71



# Bibliography

- [1] J. D. Anderson, *Computational Fluid Dynamics: The Basics with Applications*. McGraw-Hill Education, 2010.
- [2] C. Audet and W. Hare, *Derivative-Free and Blackbox Optimization*. Springer, 2017.
- [3] G. Vanderplaats, *Multidiscipline Design Optimization*. Vanderplaats Research & Development, Incorporated, 2007.
- [4] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Physica-Verlag HD, 2010, pp. 177–186.
- [5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [6] B. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.
- [7] Y. Nesterov, "A method for solving the convex programming problem with convergence rate  $o(1/k^2)$ ," pp.543 – 547, 1983.
- [8] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.
- [9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.
- [10] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2006.
- [11] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms 1. general considerations," *IMA Journal of Applied Mathematics*, vol. 6, no. 1, pp. 76–90, 1970.
- [12] R. Fletcher and M. J. D. Powell, "A new approach to variable metric algorithms," *The Computer Journal*, vol. 13, no. 3, pp. 317–322, 1970.
- [13] D. Goldfarb, "A family of variable-metric methods derived by variational means," *Mathematics of Computation*, vol. 24, no. 109, pp. 23–26, 1970.
- [14] D. F. Shanno, "Conditioning of quasi-newton methods for function minimization," *Mathematics of computation*, vol. 24, no. 111, pp. 647–656, 1970.

- [15] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.
- [16] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large-scale bound-constrained optimization," pp. 550–560, 1997.
- [17] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. Academic Press, 1981.
- [18] D. Kraft, *A Software Package for Sequential Quadratic Programming*, ser. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988.
- [19] J. H. Holland, "Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence," 1992.
- [20] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.
- [21] H. Karloff, *The Simplex Algorithm*. Birkhäuser Boston, 1991.
- [22] M. J. D. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," 1994.
- [23] J. A. Nelder and R. Mead, *A Simplex Method for Function Minimization*. Oxford University Press, 1965, vol. 7, no. 4.
- [24] J. J. Moré and S. M. Wild, "Benchmarking derivative-free optimization algorithms," *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 172–191, 2009.
- [25] A. Agrawal, K. Ravi, P.-S. Koutsourelakis, and H.-J. Bungartz, "Multi-fidelity constrained optimization for stochastic black box simulators," 2023.
- [26] J. Staines and D. Barber, "Variational optimization," 2012.
- [27] T. Bird, J. Kunze, and D. Barber, "Stochastic variational optimization," 2018.
- [28] C. P. Robert and G. Casella, "Monte carlo statistical methods," New York, NY, 2004.
- [29] S. Alarie, C. Audet, A. E. Gheribi, M. Kokkolaras, and S. Le Digabel, "Two decades of blackbox optimization applications," 2021.
- [30] S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih, "Monte carlo gradient estimation in machine learning," 2020.
- [31] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Machine Learning*, 1992, pp. 229–256.



- [32] G. Hinton, "Neural networks for machine learning." 2012.
- [33] S. I. Amari, "Natural Gradient Works Efficiently in Learning," *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.
- [34] R. Shrestha, "Natural gradient methods: Perspectives, efficient-scalable approximations, and analysis," 2023.
- [35] R. Karakida, S. Akaho, and S. ichi Amari, "Universal statistics of fisher information in deep neural networks: Mean field approach," 2019.
- [36] A. V. Fiacco and G. P. McCormick, "The sequential unconstrained minimization technique (sumt) without parameters," pp. 820–827, 1967.
- [37] G. Louppe, J. Hermans, and K. Cranmer, "Adversarial variational optimization of non-differentiable simulators," 2020.
- [38] G. C. Pflug, "Optimization of stochastic models: The interface between simulation and optimization," 2012.
- [39] S. Shirobokov, V. Belavin, M. Kagan, A. Ustyuzhanin, and A. G. Baydin, "Black-box optimization with local generative surrogates," 2020.
- [40] N. Ruiz, S. Schuler, and M. Chandraker, "Learning to simulate," 2019.
- [41] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," 2017.
- [42] R. G. McClarren, "Uncertainty quantification and predictive computational science: A foundation for physical scientists and engineers," 2018.
- [43] W. Kool, H. van Hoof, and M. Welling, "Buy 4 reinforce samples, get a baseline for free!" in *DeepRLStructPred@ICLR*, 2019.
- [44] R. W. Pitz and J. W. Daily, "Combustion in a turbulent mixing layer formed at a rearward-facing step," pp. 1565–1570, 1983.
- [45] H. Jasak, A. Jemcov, and Željko Tuković, "Openfoam: A c++ library for complex physics simulations," 2007.
- [46] "Openfoam v9 user guide," 2021.
- [47] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Corporation, 1980.
- [48] H. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, 2nd ed. Pearson Education Limited, 2007.

- [49] F. R. Menter, "Two-equation eddy-viscosity turbulence models for engineering applications," *AIAA Journal*, vol. 32, pp. 1598–1605, 1994.
- [50] P. Spalart and S. Allmaras, "A one-equation turbulence model for aerodynamic flows," *AIAA*, vol. 439, 01 1992.
- [51] D. C. Wilcox, "Reassessment of the scale-determining equation for advanced turbulence models," *AIAA Journal*, vol. 26, pp. 1299–1310, 1988.
- [52] L. H. Hodges, W. Reichelderfer, J. E. Caskey, and Smagorinsky, "General circulation experiments with the primitive equations i . the basic experiment," 1962.
- [53] B. Launder and B. Sharma, "Application of energy dissipation model of turbulence to the calculation of flow near spinning disc," *Letters Heat Mass Transfer*, vol. 1, pp. 131–137, 11 1974.
- [54] W. Jones and B. Launder, "The prediction of laminarization with a two-equation model of turbulence," *International Journal of Heat and Mass Transfer*, vol. 15, pp. 301–314, 1972.
- [55] B. E. Launder and D. B. Spalding, "The numerical computation of turbulent flows," pp. 269–289, 1974.
- [56] A. Brandt, "Multi-level adaptive solutions to boundary-value problems math compt," 1977.
- [57] T. J. Chung, "Computational fluid dynamics," 2010.
- [58] S. J. Daniels, A. A.-A. M. Rahat, R. M. Everson, G. R. Tabor, and J. E. Fieldsend, "A suite of computationally expensive shape optimisation problems using computational fluid dynamics," 2018.