

Universidad Autónoma de San Luis Potosí

Facultad de Ingeniería
Centro de Investigación y Estudios de Posgrado

Study of the influence of vortex generators on the performance of finned tube heat exchangers through the application of CFD and machine learning

T E S I S

que para obtener el grado de:

Maestría en Ingeniería Mecánica

P R E S E N T A

Ing. Carlos Tovar Martínez

ASESORA

Dra. Geydy Luz Gutiérrez Urueta

CO-ASESORA

Dra. Cintia Monreal Jiménez



San Luis Potosí, S.L.P., México, Noviembre de 2026

UASLP
Universidad Autónoma de San Luis Potosí

FACULTAD DE
INGENIERÍA

23 de abril de 2026

ING. CARLOS TOVAR MARTÍNEZ
P R E S E N T E.

En atención a su solicitud de Temario, presentada por las **Dras. Geydy Luz Gutiérrez Urueta y Cintia Monreal Jiménez,** Asesora y Coasesora de la Tesis que desarrollará Usted, con el objeto de obtener el Grado de **Maestro en Ingeniería Mecánica**. Me es grato comunicarle que en la sesión del H. Consejo Técnico Consultivo celebrada el día 23 de abril del presente, fue aprobado el Temario propuesto:

TEMARIO:

"Study of the influence of vortex generators on the performance of finned tube heat exchangers through the application of CFD and machine learning"

Introduction.
1. Theoretical foundations for numerical simulation and machine learning application for heat exchangers with vortex generators.
2. Design of experiments using CFD and machine learning.
3. Development of CFD simulations for heat exchangers with vortex generators.
4. Application of machine learning for predicting heat exchangers performance indicators.
5. Thermo-hydraulic optimization for the heat exchangers.
6. Analysis and discussion of results obtained using CFD and machine learning.

Conclusions.
References.

"MODOS ET CUNCTARUM RERUM MENSURAS AUDEBO"

A T E N T A M E N T E

DR. EMILIO JORGE GONZÁLEZ GALVÁN
DIRECTOR

UNIVERSIDAD AUTONOMA
DE SAN LUIS POTOSI
FACULTAD DE INGENIERIA
DIRECCION

Ccp. Archivo

-bvlg

www.uaslp.mx

Av. Manuel Nava 8
Zona Universitaria • CP 78290
San Luis Potosí, S.L.P.
tel. (444) 826 2330 al39
fax (444) 826 2336

UNIVERSIDAD AUTÓNOMA DE SAN LUIS POTOSÍ
FACULTAD DE INGENIERÍA
Área de Investigación y Estudios de Posgrado

D E C L A R A C I Ó N

El presente trabajo que lleva por título:

Study of the influence of vortex generators on the performance of finned tube heat exchangers through the application of CFD and machine learning.

se realizó en el periodo mayo de 2025 a octubre de 2026 bajo la dirección de la Dra. Geydy Luz Gutiérrez Urueta y de la Dra. Cintia Monreal Jiménez.

Originalidad

Por este medio aseguro que he realizado el trabajo reportado, y la escritura de este documento de tesis para fines académicos sin ayuda indebida de terceros y sin utilizar otros medios más que los indicados.

Las referencias e información tomadas directa o indirectamente de otras fuentes se han definido en el texto como tales y se ha dado el debido crédito a las mismas.

El autor exime a la UASLP de las opiniones vertidas en este trabajo escrito y asume la responsabilidad total del mismo.

Este trabajo no ha sido sometido como tesis o trabajo terminal a ninguna otra institución nacional o internacional en forma parcial o total, exceptuando el caso cuando existe un convenio específico de doble titulación celebrado entre ambas instituciones.

Se autoriza a la UASLP para que divulgue este documento para fines académicos, una vez que termine el periodo del embargo autorizado por las autoridades.

El autor del trabajo escrito, Carlos Tovar Martínez

CONTENTS

Agradecimientos v

Abstract vii

Resumen ix

List of Figures xi

List of Tables xiii

List of symbols xv

Introduction 1

Objectives . . . 2

General objectives . . . 2

Specific objectives . . . 2

Organization of this work . . . 2

1 Theoretical foundations for numerical simulation and machine learning application for heat exchangers with vortex generators 3

1.1 Governing Equations . . . 3

1.2 Heat Exchangers . . . 4

1.2.1 Heat transfer fundamentals for heat exchanger . . . 5

1.2.2 Heat exchanger performance parameters . . . 9

1.3 Heat transfer enhancement in FTHE . . . 14

1.3.1 Passive heat transfer enhancement techniques . . . 14
1.3.2 Features of vortex generators . . . 15
1.3.3 Tube bundles and VGs for performance enhancement . . . 16
1.4 Numerical modelling of FTHE . . . 19
1.4.1 Numerical methods for CFD . . . 20
1.4.1.1 Finite difference method . . . 20
1.4.1.2 Finite element method . . . 20
1.4.1.3 Finite volume method . . . 21
1.4.2 Time and space discretization . . . 21
1.4.3 Grid quality metrics . . . 25
1.4.4 Properties of CFD methods . . . 25
1.4.5 Workflow for CFD simulations . . . 26
1.5 Introduction to machine learning . . . 27
1.5.1 Basic concepts and classification . . . 27
1.5.2 Application to this work . . . 28

2 Design of experiments using CFD and machine learning 31
2.1 Case study description . . . 31
2.2 Methodology for computational experiments . . . 35
2.2.1 Experimental designs . . . 36
2.2.2 Sobol' sequences . . . 37
2.2.3 Active learning and surrogate models . . . 40
2.2.4 Workflow process overview . . . 40

3 Development of CFD simulations for heat exchangers with vortex generators 43
3.1 Preliminary modeling considerations . . . 43
3.2 A brief overview of turbulence models . . . 44
3.3 Validation of the numerical model . . . 48
3.3.1 Settings data . . . 49
3.3.2 Meshing process . . . 49
3.3.3 Solver Setup . . . 52
3.3.4 CFD validation results . . . 54
3.4 Simulations for the FTHE with bow-shaped VGs . . . 55
3.4.1 Grid generation through Fluent Meshing . . . 56
3.4.2 Grid independence study . . . 58

3.4.3 Automation of batch simulations . . . 61
3.4.3.1 Preparation of 3D models . . . 61
3.4.3.2 Simulations though batch processing . . . 62
3.4.3.3 Results . . . 62

4 Application of machine learning for predicting heat exchangers performace indicators 65
4.1 Machine learning algorithms for regression . . . 66
4.1.1 Gaussian Process Regressor . . . 66
4.1.1.1 Radial basis function and rational quadratic kernels . . . 67
4.1.1.2 Predictive variance . . . 67
4.1.2 Artificial Neural Networks . . . 68
4.1.3 Extreme Gradient Boosting . . . 69
4.2 Code development and testing . . . 70
4.2.1 Modifications for better predictions . . . 72
4.2.2 Sequential design . . . 74
4.2.3 Cross-validation . . . 75
4.3 Relationship between features and targets . . . 75

5 Thermo-hydraulic optimization for the heat exchangers 83
5.1 Optimization concepts . . . 84
5.1.1 Multi-objective optimization overview . . . 85
5.1.2 Non-dominated sorting genetic algorithm II . . . 86
5.2 Optimization over $\overline{Nu}$ and f . . . 87

Conclusions 95

References 99

AGRADECIMIENTOS

Quiero reconocer a María José por haber formado parte de la importante etapa de decisión de cursar una maestría. A Cristian Martínez por sus ingeniosas pláticas y su amigable presencia. A Ángel Lara por inspirarme artísticamente y recordarme que en la vida hay que buscar el balance. A Dulce Dávila por darme las mejores porras que pude tener en este largo proceso y por escucharme pacientemente. Gracias totales a Guillermo Ruiz por ser mi referente en la ingeniería y por orientarme en la introspección y la mejora continua.

Mi profunda gratitud al Dr. Dirk Frederik por su constante orientación, al Dr. Robert Jäckel por brindarme herramientas que me ayudaron al desarrollo de mi tesis, a la Dra. Cintia Monreal por su paciencia y perseverante apoyo, y a la Dra. Geydy Gutiérrez por pedirme siempre lo mejor de mí para buscar la perfección y por el más cálido acompañamiento que tuve en esta satisfactoria trayectoria de mi vida.

Y principalmente, mi más grande agradecimiento a mis padres, Rosa y Arnulfo, que siempre han velado por mi bienestar y su inmutable presencia a lo largo no solo de esta etapa de mi vida, sino de todas.

A todos, muchas gracias.

ABSTRACT

Heat exchangers operate with two working substances, the one that gives up energy and the one that receives it. In a heat exchanger that works with gas on one or both sides, the thermal resistance on the gas side is responsible for the majority of the total resistance. To decrease the thermal resistance on the gas side, techniques are used to increase heat transfer, such as using extended surfaces on tubes to increase the area exposed to energy transfer. These devices are called finned tube heat exchangers and usually use air on the external side of the tubes; thus, that is where the thermal resistance is the greatest. To reduce thermal resistance, a passive enhancement technique is employed by installing vortex generators on the surface of the fins that facilitate the interruption in the growth of the thermal boundary layer. Different shapes of vortex generators have been proposed; however, studies in the literature are restricted to one to three geometrical degrees of freedom, leading to limited analysis. In addition, these vortex generators are used in different heat exchangers, lacking standardization that would allow for a comparison of their impact on performance. The foremost objective of this qualitative project was to review the study of bow-shaped vortex generators by applying five degrees of freedom in their geometrical position on the fins when elliptical tubes are used instead of circular ones. Moreover, the study was developed by adopting the characteristics of heat exchangers that were already presented in the literature, in order to create standardization in the use of finned tube heat exchangers with vortex generators. The analysis of the geometrical arrangements and their outcomes was carried out by considering the thermo-hydraulic performance in the airside of the heat exchanger through machine learning regression models that used data obtained from computational fluid dynamics simulations. [redacted] The

findings refer to installing the vortex generators cut off from the tubes to allow the creation of longitudinal vortices, which promote heat transfer augmentation. A comparison between circular tubes and elliptical tubes suggests that elliptical tubes handle slightly more thermal energy, but decrease the pressure drop significantly.

RESUMEN

Los intercambiadores de calor operan con dos sustancias de trabajo: una que cede energía y otra que la recibe. En un intercambiador de calor que opera con gas en uno o ambos lados, la resistencia térmica del lado del gas es la responsable de la mayor parte de la resistencia total. Para reducir la resistencia térmica del lado del gas, se emplean técnicas destinadas a aumentar la transferencia de calor, como el uso de superficies extendidas en los tubos para incrementar el área expuesta a la transferencia de calor. Estos dispositivos se denominan intercambiadores de calor de tubos aletados y suelen utilizar aire en el lado exterior de los tubos; por lo tanto, ahí es donde la resistencia térmica es mayor. Para reducir la resistencia térmica, se emplea una técnica de mejora pasiva mediante la instalación de generadores de vórtices en la superficie de las aletas, lo que facilita la interrupción del crecimiento de la capa límite térmica. Se han propuesto diferentes formas de generadores de vórtices; sin embargo, los estudios publicados se limitan a entre uno y tres grados de libertad geométricos, lo que restringe el alcance del análisis. Además, estos generadores de vórtices se emplean en distintos intercambiadores de calor, por lo que carecen de una estandarización que permita comparar su impacto en el rendimiento.

El objetivo principal de este proyecto cualitativo fue revisar el estudio de los generadores de vórtices "bow-shaped", aplicando cinco grados de libertad en su posición geométrica respecto de las aletas cuando se emplean tubos elípticos en lugar de circulares. Además, el estudio se llevó a cabo tomando como referencia las características de los intercambiadores de calor ya presentadas en la literatura, con el fin de estandarizar el uso de intercambiadores de calor de tubos aletados con generadores de vórtices. El análisis de los arreglos geométricos y sus resultados se llevó a cabo considerando el rendimiento termohidráulico en el lado del aire del intercambiador de calor, mediante modelos de machine learning de regresión entrenados con datos provenientes de simulaciones de dinámica de fluidos computacional.

[redacted] Los resultados muestran que los generadores de vórtices deben estar separados de los tubos para permitir la formación de vórtices longitudinales, lo que favorece un mayor intercambio de calor. Y una comparación entre tubos circulares y elípticos sugiere que los elípticos disipan ligeramente más energía térmica, a la vez que reducen significativamente la caída de presión.

LIST OF FIGURES

1.1 Temperature distribution inside the thermal boundary layer. *Modified from [4].* . 8
1.2 Heat transfer between two fluid in an arbitrary flow orientation. 10
1.3 Representation of FTHE where air is the working fluid for the external side. *Modified from [2].* 11
1.4 VGs over a flat smooth surface. They can be punched directly on the fin or can be a protrusion installed over any surface. *Modified from [3].* 15
1.5 General representation of longitudinal vortices. 17
1.6 Cross-sectional view of the tubes: two different ways to arrange the tube bundles. 17
1.7 Different shapes of VGs. 18
1.8 Types of elements for 3D meshes. 22
1.9 Visual difference for a structured and unstructured mesh. 23
1.10 An example of non-conformal mesh. Note that some vertices do not coincide with the vertices of adjacent elements. 24
1.11 At the top and bottom regions wall boundaries are placed to illustrate the usage of inflation layers. Note after the last inflation layer the mesh continues with different type of elements. 24

2.1 Bow-shaped VG by Bai et al. [10]. 32
2.2 Geometry arrangement proposed by Bai et al. where the parameters x_1, y_1 and y_2 represent the design variables. *Modified from [10].* 33
2.3 Dimensions for the portion of the FTHE. Values in mm. 33
2.4 Concept of the portion of the FTHE under study. 35

2.5 Matrix scatterplot with histograms for the relationship between the design variables. . . . 39
2.6 General workflow process for the project development. . . . 41

3.1 Turbulence dissipation described by the cascade energy. *Modified from [27].* . . . 45
3.2 FTHE with staggered tubes that Sinha et al. studied. *Modified from [20].* . . . 48
3.3 Multibody representation of the fluid domain of the Sinha's 3D domain. . . . 50
3.4 Top view of a clipping x-z plane. Note the usage of multibody for mesh quality improvement. . . . 51
3.5 Comparison of two results. Average relative error for $\overline{Nu}$ and f: 3.3% and 3.8%, respectively. . . . 55
3.6 Blunt end for the bow-shaped VGs . . . 58
3.7 Dimensions for the extensions in the halved FTHE with bow-shaped VGs. . . . 58

4.1 Graphical concept of the MTNN model. . . . 73
4.2 Sorting of importance of each feature based on the target value. . . . 77
4.3 Sorting of importance of each feature based on the target value. . . . 80
4.4 Sorting of importance of each feature based on the target value. . . . 80

5.1 Pareto optimal set (in red) obtained by NSGA-II with the GPR model. . . . 87
5.2 Pareto fronts obtained from MOO using NSGA-II . . . 88
5.3 Sorting of importance of each feature based on the target value. . . . 89
5.4 Sorting of importance of each feature based on the target value. . . . 90
5.5 New Pareto fronts obtained from MOO using NSGA-II . . . 92

LIST OF TABLES

2.1 Range of values for the design variables. 34

3.1 Thermophysical properties for air at 315 K [5]. 44
3.2 Values of average velocity for u_x for each case. 49
3.3 Mesh statistics and quality metrics. 53
3.4 Data for grid independence study. 59
3.5 Summary of GCI analysis and asymptotic range verification. 60

4.1 Performance metrics for the 5-fold cross-validation and test datasets through ANN model. 76
4.2 Performance metrics for the 5-fold cross-validation and test datasets through GPR model. 76

5.1 Comparison of CFD and ML (GPR) predicted outputs. 91
5.2 Comparison of CFD and ML (MTNN) predicted outputs. 91
5.3 Maximum and minimum cases for η compared to the baseline values. 92
5.4 Geometrical arrangement that yield maximum and minimum scenarios for η. . . 92

LIST OF SYMBOLS

A_c	cross-sectional area	T_b	bulk temperature
A_s	surface area	t	height of VG or time
C	capacity rate	U	overall heat transfer coefficient, average absolute velocity magnitude
C_r	heat capacity ratio	u_i	velocity component in direction of x_i
c_p	specific heat	u_i'	fluctuating speed
D_h	characteristic length or hydraulic diameter	u_τ	friction velocity
e	energy	$\dot{V}$	volumetric flow
f	Darcy's friction factor	W	width of the channel
f_i	body force in direction of x_i	X_1	longitudinal separation of the VG from the entrance edge of the channel
H	height of the channel	x_i	spatial coordinates
h	enthalpy	Y_1	transversal separation of the first VG from the horizontal edge of the channel
h_c	heat coefficient	Y_2	transversal separation of the second VG from the horizontal edge of the channel
k	kinetic energy	y^+	y plus

k_f	thermal conductivity	$\langle ... \rangle$	average or mean magnitude
L	length of the channel	β	proportionality coefficient for heat transfer, angle for rectangular winglet
LMTD	logarithmic mean temperature difference	ΔT_M	effective mean temperature difference
$\dot{m}$	mass flow	δ	viscous boundary layer thickness
Ma	Mach number	δ_t	thermal boundary layer thickness
Nu	Nusselt number	ϵ	heat exchanger effectiveness or turbulent dissipation rate
$\overline{Nu}$	average Nusselt number	η	thermo-hydraulic performance
P	pumping power	λ	second viscosity
P_c	perimeter	μ	dynamic viscosity
p	pressure or apparent order	ν	kinematic viscosity
Pr	Prandtl number	Ω	streamwise averaged absolute vorticity flux
q	heat rate	ω	specific dissipation rate
q_i''	heat flux in direction of x_i	Φ	dissipation function
R	radius of the major axis	φ_1	orientation of the first VG with reference to the horizontal edge of the channel
R_s	thermal conductance	φ_2	orientation of the second VG with reference to the horizontal edge of the channel
r	radius of the minor axis or grid refinement factor	ρ	density of the fluid
Re	Reynolds number	τ_w	shear stress of the fluid over the surface
T	temperature		

INTRODUCTION

Historically, industrial development has been characterized by successive energy transitions as different natural resources have been harnessed. From coal to nuclear power, each era has required specific technologies to facilitate energy conversion and consumption. Energy consumption is primarily driven by electrical energy, which is obtained through a series of processes that transform energy from one form to another.

In recent decades, technological development has driven energy consumption toward more efficient resource utilization, alongside the pursuit of zero-emission industries leveraging renewable and clean resources. Consequently, several legacy technologies are being phased out. Nevertheless, a common denominator across all energy transformations is the direct or indirect involvement of thermal energy. Therefore, thermal equipment, particularly heat exchangers, plays a role at some stage in the energy transformation process. Since heat exchangers remain fundamental components, their optimization presents a significant opportunity to enhance overall energy efficiency.

One of the primary challenges in heat exchanger design is achieving an optimal balance between heat transfer enhancement and pressure drop. Passive techniques, such as the incorporation of vortex generators, have been proposed to enhance heat transfer by promoting flow disturbance; however, this enhancement is frequently accompanied by a undesirable increase in pressure drop.

Therefore, the present study focuses on analyzing the influence of vortex generators installed in finned tube heat exchangers, with the objective of identifying configurations that successfully balance heat transfer augmentation and pressure drop. This analysis is conducted utilizing advanced computational tools, specifically computational fluid dynamics and machine learning. The integration of these methodologies facilitates a comprehensive evaluation of flow behavior

and thermal performance, ultimately leading to the identification of optimal design parameters.

Objectives

General objectives

To study the influence of vortex generators on the performance of finned tube heat exchangers through the application of CFD and machine learning.

Specific objectives

1. To study the theoretical foundations relevant to heat exchangers, vortex generators, and machine learning applications in thermal systems.
2. To implement computational fluid dynamics simulations to analyze the performance yielded by each VG configurations in heat exchangers.
3. To perform and validate numerical simulations.
4. To develop and train machine learning models using the data obtained from CFD simulations.
5. To validate the trained machine learning models in predicting results by different VG configurations.
6. To generate predictions for VG design parameters and compare their performance in terms of heat transfer efficiency and pressure drop, identifying the configurations that yield optimal results.

Organization of this work

This thesis is organized by six principal chapters. The opening chapter is related to the introduction of the science that is utilized for the development of the project. For chapter 2 the methodology of the project is presented by adopting the design of experiments approach. Chapters 3 and 4 are developed based on the adoption of the computational fluid dynamics and machine learning, respectively, thus these chapters are the backbone for the remaining ones. Based on the two precedent chapters, the optimization for the thermo-hydraulic performance is obtained in chapter 5. Finally, for chapter 6, the results from all previous steps are gathered to yield a robust analysis of the influence of bow-shaped vortex generators installed within the finned tube heat exchanger.

CHAPTER 1

THEORETICAL FOUNDATIONS FOR NUMERICAL SIMULATION AND MACHINE LEARNING APPLICATION FOR HEAT EXCHANGERS WITH VORTEX GENERATORS

This chapter gathers the basic concepts required for developing the project, such as the governing equations, geometric parameters, simulation techniques, equations applied to heat exchangers (HEs), data reduction or constraints considered for the analysis or the study of physics, if it is appropriate. Additionally, the fundamentals of machine learning theory and their application to the present work are introduced.

1.1 Governing Equations

At the macroscopic scale, the continuum concept is adopted to derive the different equations that govern the fluid behavior by establishing a set of *field variables* that represents average molecular properties at each corresponding position and time. Such equations are *partial differential equations* (PDEs) obtained from the conservation laws of mass, energy and momentum. This derivation is complemented by the constitutive equations where transport coefficients are described either as constants or as a function of a certain field variable.

For the current project, only Newtonian fluids are considered. Using the Einstein summation convention, Eq. 1.1 describes the conservation of mass where ρ represents the density of the substance, and x_i is any of the Cartesian coordinates, while u_i is the velocity magnitude corresponding to the coordinate x_i.

The equations that describe the flow movement are condensed in Eq. 1.2 known as the Navier-Stokes equations, where p represents the pressure at certain point, μ is the dynamic viscosity

of the fluid (first viscosity coefficient) while λ corresponds to the second viscosity which relates viscous stress to the volumetric expansion of the fluid, and f_i represents body forces. The constitutive equation for the heat-flux vector (q_i'') is given by Fourier's law, represented in Eq. 1.3 where T is used to represent the temperature at a given point and k_f is a proportionality factor referred to as the thermal conductivity of the substance. Depending on the scenario, this factor may be constant or temperature-dependent. Eq. 1.4 is derived from the conservation of energy for the fluid flow where, Φ is the dissipation function that represents the conversion of mechanical energy into thermal energy due to viscosity, and e is the internal energy of the working fluid.

$$\frac{\partial \rho}{\partial t} + u_i \frac{\partial \rho}{\partial x_i} + \rho \frac{\partial u_i}{\partial x_i} = 0 \tag{1.1}$$

$$\rho \frac{\partial u_j}{\partial t} + \rho u_i \frac{\partial u_j}{\partial x_i} = -\frac{\partial p}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\lambda \frac{\partial u_i}{\partial x_i}\right) + \frac{\partial}{\partial x_k}\left[\mu\left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k}\right)\right] + \rho f_i \tag{1.2}$$

$$q_j'' = -k_f \frac{\partial T}{\partial x_j} \tag{1.3}$$

$$\rho \frac{\partial e}{\partial t} + \rho u_i \frac{\partial e}{\partial x_i} = -p \frac{\partial u_i}{\partial x_i} + \frac{\partial}{\partial x_j}\left(k_f \frac{\partial T}{\partial x_j}\right) + \Phi \tag{1.4}$$

The above equations model fluid flow physics in a general way [1], thus providing the foundation for further derivations as required for the current study.

1.2 Heat Exchangers

HEs are devices widely used as a part of HVAC (heating, ventilation, and air conditioning) systems in different industries such as pharmaceuticals, power generation, chemical processing, and refrigeration. The primary function of a HE is, in most cases, to either heat or cool a fluid by transferring heat to or from another fluid, due to the temperature difference between them. Usually, the fluids involved in HEs do not mix between each other but are separated by a solid interface. Depending on the application, these fluids can be either liquid or gas. Some HEs use a liquid on one side and gas on the other, or fluids in the same phase on both sides. It is also common that HEs use fluids undergoing phase change at certain stages. For unmixed fluids, there are two principal components for the solid interfaces that separate them: tubes and plates. Enhanced surfaces can be obtained by combining tubes and plates in different

geometrical configurations to increase the contact area to improve heat transfer. Based on the direction of the fluids inside the HE system, the flow can be classified as parallel, counter and cross flows. In general, the classification of HEs depends on their construction geometry, the heat transfer mechanisms and the applications. HEs are commonly classified into three primary categories [2][3], from which more specific classifications are derived.

- Tubular heat exchangers.
  - Shell-and-tube HEs.
  - Double pipe HEs.
  - Tube in spiral HEs.
- Plate heat exchangers.
  - Gasketed plate HEs.
  - Pillow plate HEs.
  - Louvered plate HEs.
- Extended surface heat exchangers.
  - Plate-fin HEs.
  - Externally finned tubes HEs.
  - Internally finned tubes HEs.
  - Finned annuli HEs.

The foregoing classifications are only some of the most known HEs types; however, many others exist that involve different construction geometries or combinations of the most common types. The size, cost, environment or the amount of energy that HE must handle are some parameters to take into account when selecting a HE. Even more parameters are needed when a new design is the target.

1.2.1 Heat transfer fundamentals for heat exchanger

To select a HE, one of the most important parameters to be considered is the HE performance. In addition to factors such as size, geometry, construction, and compatible applications, the performance provides the information needed to determine whether a given HE can handle a certain amount of energy transfer. This performance depends indirectly on the working fluid's physical properties, the flow type, the temperature difference, and the contact surface area. To

determine this parameter, it is necessary to analyze other related parameters. Consider Eq. 1.5, which gives the general heat transfer rate, where β is a proportionality coefficient and ΔT is the temperature difference.

$$q \propto \beta \Delta T \tag{1.5}$$

The value of β depends on the heat transfer mechanism: conduction, convection, radiation, or a combination of them. For HEs, heat transfer occurs primarily by convection because the working substances are fluids. Heat transfer by conduction must also be considered when the solid interface offers significant resistance to energy transfer, which depends on the thickness of the solid, the geometry, and the material properties.

Individual energy analysis in the working fluids is made by knowing their corresponding enthalpy differences. Ideally, the changes in enthalpy are the same for the cool and hot fluids in HEs; however, primarily due to the heat losses and complex flow in HEs such differences are not exactly the same. If the changes in potential and kinetic energy are neglected, for substances that consider variable specific heat, the appropriate math expression is shown in Eq. 1.6a where h represents the enthalpy per unit of mass at certain state. When specific heat variations are neglected, the expression can be written using the inlet and outlet temperatures, as shown in Eq. 1.6b. The parameter $\dot{m}$ denotes the mass flow rate and c_p is the specific heat of the substance. Subscripts i and o indicate the inlet and outlet conditions, respectively.

$$q = \dot{m}(h_o - h_i) \tag{1.6a}$$

$$q = \dot{m} c_p (T_o - T_i) \tag{1.6b}$$

Note that Eq. 1.6b only measures how the substance increases or decreases its energy based on temperature changes, provided that no phase change occurs. This expression only considers the inlet and outlet temperatures (or the initial and final temperatures of any process). To understand how these temperatures vary, it is necessary to consider the dominant heat transfer mechanisms. The first mechanism is *conduction*, which corresponds to energy transfer through molecular activity. In this process, energy is transferred from regions of higher temperature to neighboring regions through molecular motion and collisions. Conduction occurs in gases, liquids, and solids when a temperature difference exists within the domain. For heat transport by conduction, the term β in Eq. 1.5 can be replaced by the product of two parameters, k_f and

A_c, where A_c is the cross-sectional area for heat transport by conduction. In general terms, Eq. 1.7 represents the heat transfer rate.

$$q = -k_f A_c \frac{\partial T}{\partial x_i} \tag{1.7}$$

Another heat transfer mechanism is called *convection*, which refers to the energy transfer due to the combined mechanism of heat transport due to advection and molecular diffusion. Observations made by Isaac Newton established the Newton's Law of Cooling, which states that heat flows due to the differences between the surface temperature and the temperature of its surroundings (fluid or gas). A constant of proportionality h_c is used, called *convection coefficient* or *heat transfer coefficient* [4]. Unlike thermal conductivity, h_c is not a material property, but depends on geometry, fluid properties, motion, and temperature difference if changes in temperature significantly modify the main parameters. Eq. 1.8 depicts the simplest heat transfer expression by convection, where A_s is the contact surface area and the subscripts s and ∞ correspond to surface temperature and bulk fluid temperature, respectively. It is common to study the heat transfer rate per unit surface area (heat flux); in such cases, Eq. 1.8b is used to compute such a value.

$$q = h_c A_s (T_s - T_\infty) \tag{1.8a}$$

$$q''_s = h_c (T_s - T_\infty) \tag{1.8b}$$

Convection is a phenomenon that can occur in quiescent fluid where its density varies due to the temperature difference. In a gravitational field, these density variations can induce fluid motion if the temperature gradient allows it. This type of heat transfer in quiescent fluids is known as natural or free convection, where buoyancy forces are responsible for the fluid motion. For cases where the flow is created by external mediums such as fans, pumps or the wind a significant pressure gradient is induced, thus a force is applied to the fluid that makes it flow towards the decreasing pressure location. This mechanism is named forced convection.
Heat transfer by convection, either free or forced takes place in the interface between the fluid and the solid surface where a boundary layer is created as a consequence of the viscous effects of the fluid that imposes a no-slip condition. Heat transfer coefficient by itself quantifies how effectively a fluid transfers heat to/from a surface and is not a parameter that can be obtained directly for the majority of the cases.

Heat transfer coefficient can be obtained analytically when Fourier's Law is applied at the boundary of the fluid that is in contact with the surface (Eq. 1.9). Solving Eq. 1.8b for h_c, the expression shown in Eq. 1.10 provides a way to compute h_c if the temperature distribution is known [4]. Note that the difference $(T_s - T_\infty)$ can be a constant or variable depending on the thermal behavior of the surface.

$$q''_s = -k_f \left.\frac{\partial T}{\partial x_2}\right|_{x_2=0} \tag{1.9}$$

$$h_c = \frac{-k_f \left.\frac{\partial T}{\partial x_2}\right|_{x_2=0}}{T_s - T_\infty} \tag{1.10}$$

In most applications, the analytical determination of h_c is not a simple task due to the complexity of knowing the temperature distribution because of either complex geometries or turbulent flows or both. In such cases heat transfer coefficient is obtained experimentally by measuring q''_s and the temperatures $(T_s - T_\infty)$. In the literature there are several empirical correlations obtained by experimental data that are applicable to certain cases. The term $\partial T/\partial x_2$ has strong variation near the surface where the thermal boundary layer (δ_t) appears (Fig. 1.1). Thermal boundary layer is proportional to the velocity boundary layer (δ), therefore the h value is directly related to the velocity flow. As the flow advances over the surface the thickness of the boundary layer increases, thus the temperature gradient tends to decrease which leads to a lower h_c value and consequently less heat flux q''_s.

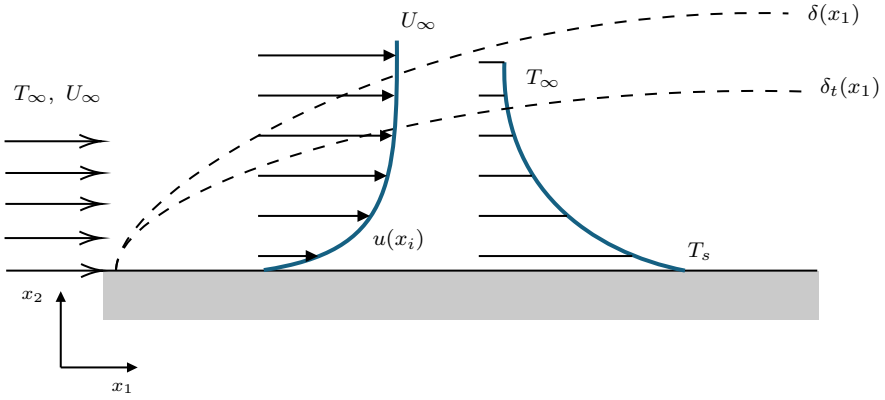


Figure 1.1: Temperature distribution inside the thermal boundary layer. *Modified from [4].*

As a consequence of this variation, the value of h_c is not the same for each location over the surface. Therefore, in order to determine the heat transfer coefficient over the entire surface,

it is necessary to compute an integration over the surface considering the dependence of h_c to the detected independent variables in the domain, such as the position variables x_i. The result of such operation will give an average convection coefficient $\bar{h}_c$, which can be substituted into Eq. 1.8 to obtain the total heat transfer rate or alternatively estimated through correlations.

1.2.2 Heat exchanger performance parameters

Based on the previous theory and under the consideration that a HEs commonly work with two fluids separated by a solid interface, Eq. 1.5 can be used to measure the heat transfer rate that a HE can handle between its working substances. The coefficient β in this equation can be replaced by the *overall heat transfer coefficient* (U) of the HE. Thus, neglecting fouling and gathering the coefficients for heat transfer between two fluids separated by a solid, in Eq. 1.11, $(U \cdot A)^{-1}$ represents the overall heat conductance, where A is a reference area to scale U. In this case, the value of R_{tot} is referred to as the overall heat resistance [5], where R_s is the thermal resistance for the solid, which depends on the geometry, thickness, and type of material. Subscripts c and h represent cool and hot fluids, respectively. For this equation, the values of A are the contact surface areas for each fluid, and these areas must belong to the shared interface between both fluids.

$$\frac{1}{U \cdot A} \equiv R_{tot} = \frac{1}{(h_c A)_c} + R_s + \frac{1}{(h_c A)_h} \tag{1.11}$$

Another consideration lies in the fact that for Eq. 1.5 requires the magnitude of the temperature difference, but attributed to the use of the overall heat transfer coefficient, also a temperature difference must be related to both working substances. Based on the temperature of the fluids at the ends of a HE an "overall" temperature difference can be computed. A general temperature difference is called *effective mean temperature difference* (EMTD) represented by ΔT_M which is an averaged value over the total surface area. A mathematical definition is presented by Eq. 1.12 where the value (ΔT) is a dependent variable of the spatial variables (x_i) and represent local temperature differences between hot and cold temperatures ($T_h - T_c$) at a point $\mathbf{x}$ on the surface.

$$\Delta T_M = \frac{1}{A} \int_A \Delta T(\mathbf{x}) dA \tag{1.12}$$

The complexity of this expression depends on the geometry of the HE and its flow type (parallel,

counter, cross, etc.). A general idea for this heat transfer between two fluids is depicted in Fig. 1.2 where the orientation of the flows is shown arbitrarily. Due to the cross-flows, multi-pass, and complex routing, Eq. 1.12 could not be computed analytically for the majority of cases, but an analytical result for this integral exists for parallel and counter flows called *logarithmic mean temperature difference* (LMTD). Several studies in the literature have presented methods to use LMTD for different flow types in HEs, such as the use of correction factors [2].

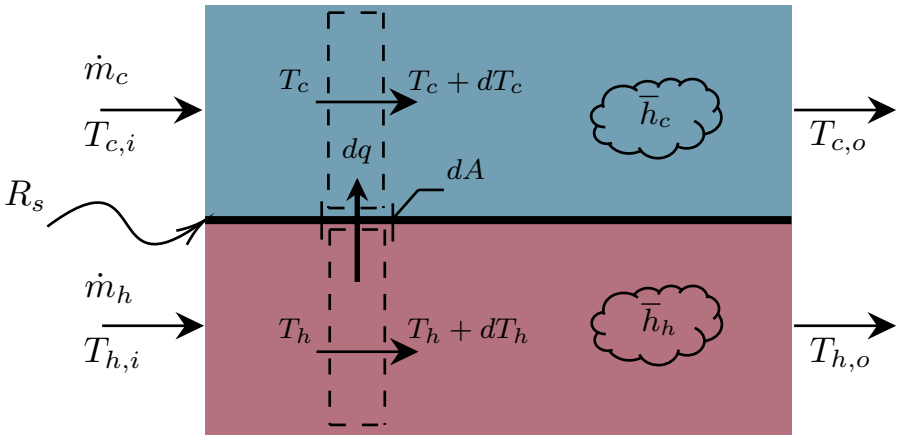


Figure 1.2: Heat transfer between two fluid in an arbitrary flow orientation.

For design or analysis purposes, the use of EMTD is widely used by the LMTD approach and correction factors that are obtained for specific flow arrangements. However, there is another method that uses a parameter called *heat exchanger effectiveness* (ϵ) defined as follows,

$$\epsilon = \frac{\text{actual heat transfer rate}}{\text{maximum possible heat transfer rate}} = \frac{q_{actual}}{q_{max}} \tag{1.13}$$

The maximum heat transfer rate is directly proportional to the product of the smaller *capacity rate* (C_{min}) and the maximum temperature difference between the fluids ($T_{h,i} - T_{c,i}$). The term capacity rate corresponds to the product of the mass flow rate $\dot{m}$ with its own specific heat value c_p. This parameter ϵ can be computed using the *number of transfer units* (NTU) which is a dimensionless parameter that quantifies the size or thermal size of a HE relative to the heat capacity rates of the fluids. Together form the effectiveness-NTU method (ϵ-NTU). The definition of NTU is represented by Eq. 1.14 where can be deduced that a higher

NTU means greater heat transfer capability. It is concluded that ϵ is a function of NTU and a parameter called *heat capacity ratio* (C_r) which represents the quotient C_{min}/C_{max}. The ϵ-NTU is a method used at HE designs stage, for comparing HE designs and as a tool for analyzing complex flow arrangements.

$$NTU = \frac{U \cdot A}{C_{min}} \tag{1.14}$$

These techniques are utilized as introductory tools for analysis and new designs of HEs. As complexity increases a general or overall attention in HEs must be reduced to study/analyze certain sole features to increase quality or enhance characteristics that positively impact the overall behavior. It was mentioned that β represents the overall conductance, however this value can be highly impacted by any single value of the right-hand side terms in Eq. 1.11. If the thermal resistance is high on one side, an effort in reducing such thermal resistance must be conducted. The current project focuses on *finned tube heat exchangers* (FTHE), which consists of a combination of two primary enhancement techniques: the usage of tubes and extended surfaces such as plates (fins). These FTHEs will be treated as devices that primarily work with air on the external side (outside the tubes as Figure 1.3 illustrates). It has been found that the airside thermal resistance contributes the most to the overall thermal resistance in these devices [3], therefore another enhancement techniques can be employed to reduce such thermal resistance.

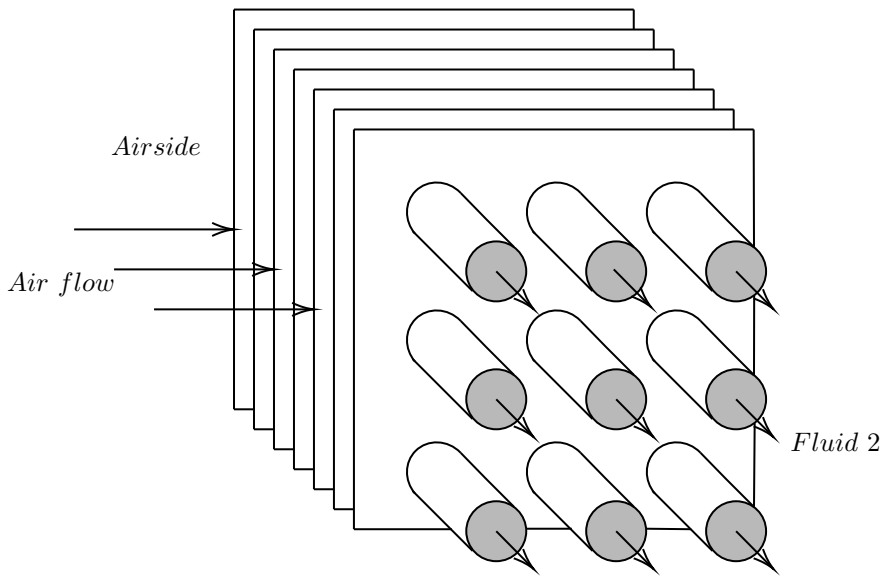


Figure 1.3: Representation of FTHE where air is the working fluid for the external side. *Modified from [2].*

Active and passive techniques exist to boost HE performance: active techniques consist of

external sources of movement or tools that induced "perturbations" such as surface vibrations, fluid vibrations, jet impingement, among others. These active techniques require additional energy consumption which must be balanced with the achieved enhancement; thus, they are not widely used. On the other hand, there are passive techniques that primarily consist of surface modifications (the surface that is in contact with the fluid) such as coated surfaces, displaced insert devices, coiled tubes, additives for liquids (like nanofluids techniques), modified shapes for extended surfaces, among others. Unlike active techniques, passive techniques do not require external energy sources for their operation, instead, it demands more pumping power. This results in a simpler relationship between energy consumption and the achieved improvement.

As heat exchangers are devices whose main characteristic is the transfer of thermal energy, specific enhancement techniques have been developed to increase this energy exchange. As mentioned beforehand, active or passive techniques must be balanced with overall energy consumption so that the use of such a technique is justified. Webb et al. [3] established the *performance evaluation criteria* (PEC) for different HEs designs. For the present work, only passive techniques are considered; therefore, the major drawback is that when an increase in h_c is achieved, an increase in pressure drop also exists, leading to an increase in pumping power (P). Pumping power is computed from the product of flow rate and the pressure drop ($\dot{V}\Delta p$), where the flow rate $\dot{V}$ is computed from the product of the air velocity (U) that crosses certain cross-sectional area (A_c). The pressure difference is computed as shown in Eq. 1.15.

$$\Delta p = f\frac{1}{2}\frac{L}{D_h}\rho U^2 \tag{1.15}$$

Where f is Darcy's friction factor, D_h is used to represent the characteristic length of the geometry through which the air passes, while L is the length or distance between the points where the pressure difference is computed. The average velocity is denoted by U. Therefore, if variations in the properties of the fluid are neglected,

$$P \propto f \cdot U^3$$

Keeping the same pumping power between the HE with and without the passive enhancement technique where the subscript b is the reference for the baseline HE, a comparison between the velocity pumping is related to the following relation,

$$\frac{U}{U_b} = \left(\frac{f_b}{f}\right)^{1/3}$$

Because the enhanced HE has higher friction than the baseline, the only way to keep the pumping power from increasing is to slow the fluid down compared to the smooth channel ($U < U_b$).

To compare the heat transfer enhancement, the *Nusselt number* (Nu) is directly proportional to the convective heat transfer coefficient; thus, the expression Nu/Nu_b is adequate to know how the heat transfer changes with the baseline as the reference. This ratio yields a way to compute the benefit of using an enhancement technique. In contrast, the friction ratio acts for the quantification of the penalty. A PEC that quantifies the benefit compared to the penalty is identified in different contexts as a quality factor, efficiency index, or thermo-hydraulic performance. For this project, the PEC is represented by the name of thermo-hydraulic performance (η), where the parameters with the subscript b refer to the values obtained from the HE without enhancement techniques [6]. This parameter is constructed by taking the pure heat transfer enhancement ratio and penalizing it by the respective velocity reduction,

$$\eta = \frac{Nu}{Nu_b}\left(\frac{U}{U_b}\right)$$

applying the relationship between the velocity and the friction factor, the equation 1.16 stands for the thermo-hydraulic performance

$$\eta = \frac{Nu}{Nu_b}\left(\frac{f_b}{f}\right)^{1/3} \tag{1.16}$$

The use of this relationship allows to compare how the use of a technique (mainly passive) can sufficiently increase the value of h_c without adversely affecting energy consumption to maintain the operation of the HE. If η <1 the enhancement technique is not adequate, the HE wastes energy. Therefore, the baseline configuration is better. On the other hand, if η >1 energy savings are achieved as consequence of the enhancement technique. It should be mentioned that Eq. 1.16 is not exclusively for the airside, but it can be used for the Fluid 2 side as well

where similar modifications are applied.

It was mentioned previously that, for the development of this project, only passive enhancement techniques are considered. Hence, based on the preceding theory, these techniques will be studied in greater depth.

1.3 Heat transfer enhancement in FTHE

Previously the concepts of active and passive techniques for heat transfer augmentation were introduced. For this subsection the theory around passive techniques is explained in more detail in order to understand why the application of such techniques is not a simple task. Furthermore, *vortex generators* (VGs) are selected to be the devices that build the passive technique for the enhancement of the performance of FTHE that will be described later on in this work.

1.3.1 Passive heat transfer enhancement techniques

In the context of FTHE, passive techniques are defined as methods that require no direct application of external power to affect the enhancement. These techniques rely solely on the geometrical modification of the heat transfer surface or the fluid path to alter the flow field. The fin itself is the primary passive technique. However, advanced passive techniques go beyond simple area extension by increasing the contact surface for heat transfer. For example, *wavy fins* increase the surface area per unit volume while also inducing steady recirculating flows in the troughs and crests of the waves. In plain fins, a thick thermal boundary layer develops as the fluid flows downstream, creating an "insulating" blanket that reduces heat transfer. Techniques like *offset strip fins* or *louvered fins* work by physically interrupting the surface. This forces the boundary layer to separate and "restart" at each new leading edge. The result is a series of thin, highly effective boundary layers. Specific geometry modifications can induce swirl and secondary flows, defined as rotation or secondary circulation within the bulk fluid, that facilitate thermal energy transfer. In some cases, *coating surfaces* or *treated surfaces* could boost heat transfer; however, they are not recommended for single-phase fluids because of their low thermal conductivity, which can induce a high conduction resistance [2]. On the other hand, this technique is highly recommended for boiling and condensation since the liquid film acts as an insulator; removing it exposes the bare metal to the vapor, thereby achieving extremely high heat transfer rates. Furthermore, passive techniques can be based on flow manipulation devices that are designed specifically for flow control rather than just area extension, such as *dimples* that are created from material removal to create a spherical or oval depression so that

the boundary layer separates at the leading edge and creates a recirculating vortex inside the cavity. This vortex ejects fluid upwards from the dimple, creating a mushroom-shaped mixing structure. Inside the tubes, *wire coils* can be inserted to intentionally eliminate laminar flow and force it to become turbulent immediately. Other alternatives for the flow within tubes include *broken* and *serrated twisted tapes* that create swirl flows due to centrifugal force that push cold fluid to the wall, also inducing chaotic mixing. In addition to all these techniques, *vortex generators* (VGs) are also flow manipulation devices that provide a heat transfer increase with a much smaller friction penalty compared to other techniques. VGs can be inserted over the fins or inside the tubes.

VGs are modern high-efficiency devices that lead to high performance attributed to the vortex generated throughout the primary flow. Continuous research into these devices exists for heat transfer augmentation with low pressure drop; therefore, to cover some unseen study areas, this project utilizes VGs as a passive enhancement technique.

1.3.2 Features of vortex generators

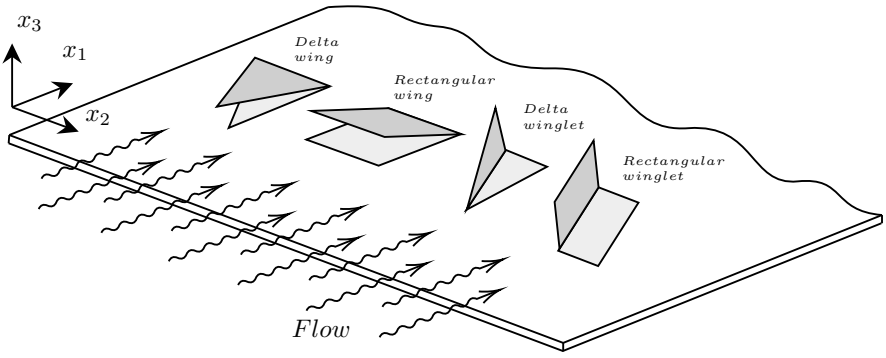


Figure 1.4: VGs over a flat smooth surface. They can be punched directly on the fin or can be a protrusion installed over any surface. *Modified from [3].*

VGs are widely used in aerodynamic applications such as aircraft wings, race cars, or wind turbine blades due to their influence on flow control. These devices are primarily utilized to delay flow separation, improve lift, and reduce drag due to the creation of swirls that aid in enhancing mixing between the slower boundary layer flow and the faster freestream flow. However, VGs are also used to disrupt the boundary layer intensification by the induction of secondary flows; thus, this behavior is exploited where flow stagnation is wanted to be prevented or minimized. The precursor geometries for vortex generators are slender shapes classified as

wings and winglets. Wings are geometries that are installed parallel to the spanwise direction but occasionally can be angled with respect to the transversal section of the flow. On the other hand, winglets are curved or flat geometries that are angled to the main flow and frequently are installed in pairs in a symmetrical way. Both wings and winglets are represented in Fig. 1.4.

HEs made of an arrangement of fins and tube bundles take advantage of the qualities that VGs provide. Vortices are induced in the flow domain between fins, nevertheless the vortices shape and spreading must be controlled. To do so it is important to distinguish between transverse and longitudinal vortices. While the axis of transverse vortices lies perpendicular to the main flow direction, longitudinal vortices are created with their axes parallel to the primary flow direction. Hence, transverse vortices can be studied as a two-dimensional flow, while for longitudinal vortices the study is always three-dimensional flow. Longitudinal vortices offer superior heat transfer enhancement compared to transverse vortices in practical applications due to their persistence and ability to influence flow over longer distances [7][8]. Unlike transverse vortices, which primarily enhance heat transfer through flow destabilization leading to self-excited fluctuating vortices, longitudinal vortices (Fig 1.5) contribute through their inherent swirling motion and ability to maintain their structure for extended lengths in relatively high flows regime which aids effectively mixing fluids that boost the heat transfer. This sustained vortex system promotes increased convective thermal transport also because longitudinal vortices are extremely persistent in both laminar and turbulent boundary layers. In contrast, transverse vortices often require flow to be unsteady to significantly enhance global heat transfer because stationary transverse vortices do not provide additional mechanisms for downstream thermal energy transport. The generation of vortex in HEs is preferred to be done by using winglets because they yield a major heat transfer contribution than wings.

1.3.3 Tube bundles and VGs for performance enhancement

Another important factor for FTHE performance is the shape of the tubes as well as their arrangement. Circular, elliptic, and flat tubes are the most used tube shapes, and their arrangement can be either in-line (Fig. 1.6a) or in a staggered (Fig. 1.6b) array. Winglet pairs are installed angled to be able to generate vortices; however, their position can be divergent or convergent downstream. Webb and Kim [3] summarize the characteristics of the VGs applied to FTHE. The protrusion height should be comparable to the local boundary layer thickness. If it is higher, it will cause a significantly higher pressure drop and affect the heat transfer enhancement achieved. Also, they mention that for circular finned tubes, a low heat transfer

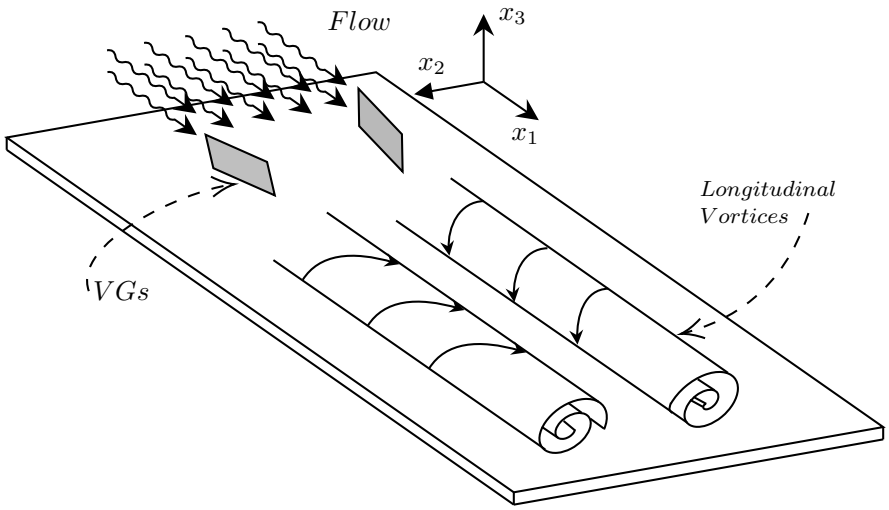
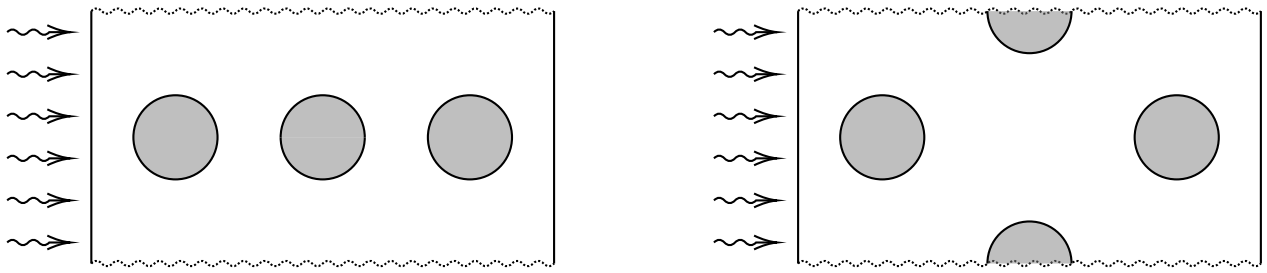


Figure 1.5: General representation of longitudinal vortices.

coefficient exists in the wake region behind the tubes, especially at low Reynolds numbers (Re). An in-line array causes the performance of the equipment to decrease due to the bypass effects that occur "behind" each tube. While staggered arrays generally yield superior thermal performance compared to in-line configurations, this enhancement incurs a substantial pressure drop penalty, which is counterproductive if excessive.



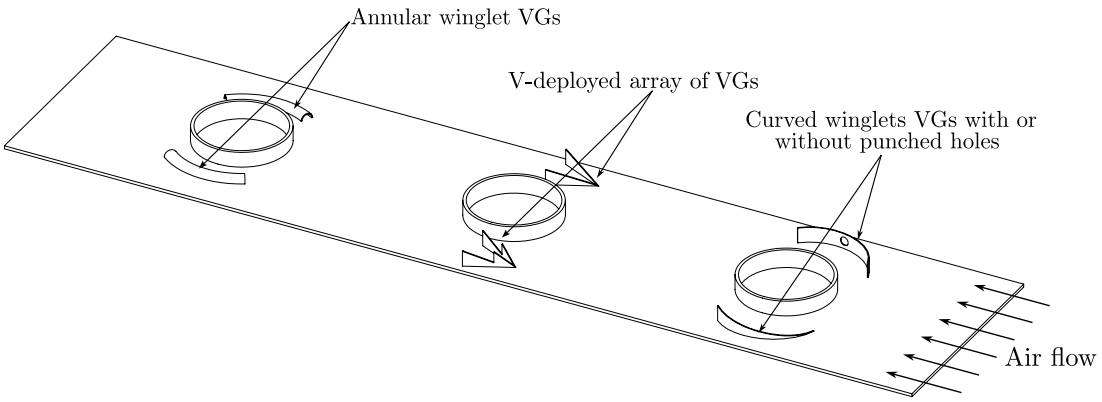
(a) In-line tubes arrangement.

(b) staggered tubes arrangement.

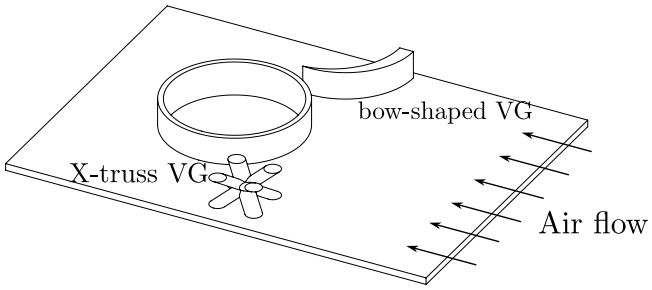
Figure 1.6: Cross-sectional view of the tubes: two different ways to arrange the tube bundles.

VGs are built based on different shapes. Fig. 1.4 depicts the most studied winglets; thus, based on these shapes, alternatives are presented by different authors. One initial adaptation involves punching holes in delta and rectangular winglet VGs to decrease pressure drop; however, related to these geometries, other different shapes are being studied as an iterative process to find the optimal geometric form [9]. In the literature, the most prevalent VG geometries include *annular winglet VGs*, *V-deployed VGs*, *curved winglet VGs* (see Fig. 1.7a), among others. Other studies utilize shapes that may not be called winglets due to their particular form that

diverges from the traditional winglet architecture, such as *bow-shaped VGs* [10] or the *X-truss VGs* [11], shown in Fig. 1.7b.



(a) Some examples of VGs mentioned in [9].



(b) Novel VG shapes mentioned by [10][11].

Figure 1.7: Different shapes of VGs.

Furthermore, several studies consider VGs as flexible devices; unlike rigid VGs that produce a steady vortex, these elastic flaps are made from flexible materials that induce periodic flapping motion, creating high-strength vortices that penetrate the viscous sublayer and successfully disrupt the growth of the thermal boundary layer. Unfortunately, achieving the exact balance of fluid-structure interaction required for optimum heat transfer and acceptable pressure drop is difficult, as the specific geometries, sizes, shapes, and spacing of FVGs still require extensive optimization. [12]. There are many VG options for thermal energy transfer enhancement in HEs; nevertheless, some VGs are more suitable for low or high flow regimes, others may work better with more dense/viscous fluids, and differing thermal boundary conditions. All these alternatives can be combined with different aspects of the HEs, such as the fin pitch, the tube shapes, and the number of tubes per row, etc. [13]. All these alternatives and combinations introduce significant analytical complexity due to the degrees of freedom available; for this reason, selecting an FTHE design and the VGs has to be a prudent approach to prevent repetitive or unreliable research.

1.4 Numerical modelling of FTHE

To know the behavior of FTHE based on their features, a method can be selected between three principal strategies to quantify parameters. Knowing the governing equations, if analytical solutions exist, this is the first option to take into account because direct solutions can be obtained. However, the Navier-Stokes equations (Eq. 1.2) and similar governing equations lack exact analytical solutions for most practical cases. Furthermore, when the physical domain is not sufficiently simple to be solved analytically, alternative techniques must be adopted. The technique that has spanned history across different disciplines is experimentation. When a setup is mounted to study certain phenomena, the physics can be replicated in a controlled space to learn about the parameters of interest. In general, conclusions have been obtained for most of the heat exchangers with specific configurations by experiments . Although experiments give a high degree of certainty, such reliability lies in the quality of each setup and the instrumentation accuracy, leading, in some cases, to high-cost investment. A relatively novel technique is numerical simulation, where results similar to those of experiments are sought, but with the difference that instead of a complex setup, computing equipment is utilized alongside the respective governing equations. The principal drawback in numerical simulations is computational resources because, for some complex cases, high-performance computing systems are required that are not easily accessible. In the current work, numerical simulations are only considered to study the influence of VGs in FTHE. These simulations for fluid motion are called *computational fluid dynamics* (CFD), where the governing equations are solved by numerical methods.

Various techniques exist for modelling fluid physics. It was mentioned that equations like the Navier-Stokes equations are difficult to solve analytically, leading to a lack of exact solutions that can be interpreted as continuous functions. The most commonly used numerical methods offer discrete solutions for complex equations, complex geometries, or both. Such numerical methods work based on a discrete domain, either in space or time. When a discretization is taking place for a spatial domain, such a task is done over the geometry that represents the physical shape or environment where physics occurs. Once discretization has been carried out, certain properties must be present in each case. These properties lead to confidence in the results of each simulation. CFD yields approximate solutions, and accuracy depends on the selected method, the discretization task, and, due to the nature of the methods, how the solution is obtained at each iteration [14]. The previous topics are presented in more detail below.

1.4.1 Numerical methods for CFD

The main methods used in CFD are based on the discretization of a finite domain, where space or time are divided into small and simpler regions in which the governing equations, as well as the boundary and initial conditions, must be satisfied. These methods can be applied to either steady or transient phenomena.

1.4.1.1 Finite difference method

The oldest method to solve PDEs is the *finite difference method*, where Taylor series expansion or polynomial fitting is adopted to find approximations to the first and second derivatives in the governing equations. This method, like other methods, works by converting the domain under study into a grid to handle a discrete domain. This means that the continuous physical space, rather than being constructed by an infinite number of points, is replaced by a finite grid of discrete nodes where the partial differential equations are algebraically evaluated. Spatial and time domains are discretized into a system of linear equations that can be solved by matrix algebra techniques. The disadvantage of the finite difference method is that the conservation laws are not strictly enforced, in addition to the fact that this method is not well suited for mapping curved, complex, or irregular boundaries found in modern engineering.

1.4.1.2 Finite element method

The *finite element method* discretizes a continuous physical domain into a finite number of smaller, simpler sub-domains called elements. These elements are connected to specific points known as nodes to form a mesh or grid. The governing equations that are handled to solve a specific case are transformed into local algebraic matrices for each element. These local matrices are then assembled into a single global system of equations, which in turn is solved using matrix algebra to determine an approximate solution across all nodes. This allows accurate modeling of irregular geometries, complex boundary conditions, and non-homogeneous material properties, making it essential for structural analysis, heat transfer, and fluid dynamics. The key features of the finite element method are that the equations are multiplied by *weighted functions* in order to subsequently be integrated over the entire domain. The approximation is done by linear or quadratic shape functions within each element that, in a certain way, guarantees continuity of the solution across each element boundary. The biggest drawback for fluid dynamics is that this method is not locally conservative by default. While the global energy/mass might be conserved over the whole domain, there is no guarantee that the flux leaving Element A exactly equals

the flux entering Element B.

1.4.1.3 Finite volume method

The *finite volume method* (FVM) is similar to the finite element method, with the difference that this method uses the integral form of the conservation equations. The domain is divided into *control volumes* (CV) or *cells* where the conservation laws are enforced in each CV. The most common approach in FVM formulation is to solve the PDEs considering each node located at the centroid of each cell. Unlike the finite element method, FVM focuses on calculating the fluxes (flow of mass, momentum, or energy) crossing the faces of each control volume. Through the divergence theorem, it converts volume integrals into surface integrals (what crosses the boundary). While the finite difference method can easily achieve very high-order accuracy on structured grids, implementing high-order spatial discretization in FVM on complex, unstructured meshes is computationally expensive and complex. Consequently, most commercial FVM implementations default to second-order accuracy as the practical standard.

1.4.2 Time and space discretization

The previous finite methods are the most employed, in consequence, discretization methods concepts must be grasped for such numerical methods. When the solution can be done for a steady state case, only spatial discretization is needed, but when the physics has transient behavior, time discretization is needed as well. Nevertheless, for time discretization the employed technique is the finite difference method and does not need more than the specific care required by this method. On the other hand, spatial discretization does need special attention to some features. As mentioned before, the spatial discretization gives a medium where the domain is divided into small domains called elements, control volumes or cells. The set made by these elements is called *grid* or *mesh*.

The mesh can be built by subdomains that create structured or unstructured grids and for the topology, the mesh can be conformal or non-conformal. These options depend on the type of the elements that are used to construct the mesh. In Fig. 1.8 are presented the element types for 3D domains. These elements have their own advantages and disadvantages explained below.

- **Hexahedron (hex)** : A brick/cube (6 rectangular faces) used primarily for structured meshes and for flow alignment where numerical diffusion is low. This type of element is

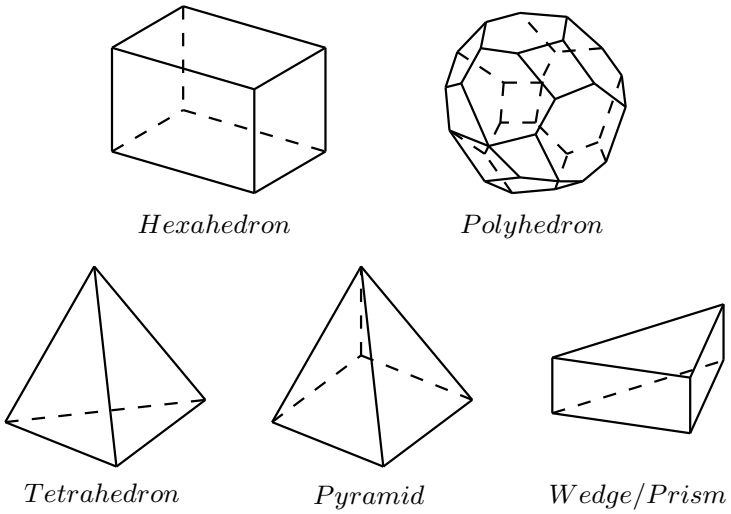


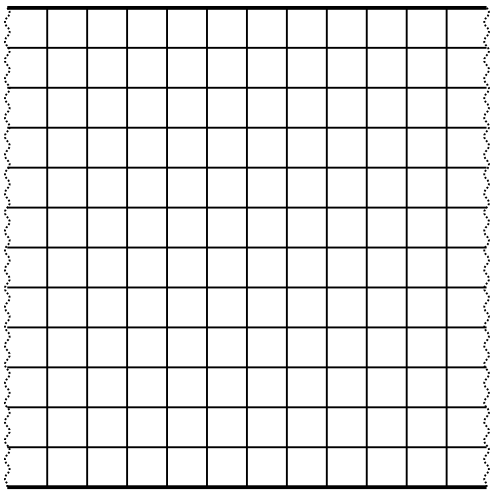
Figure 1.8: Types of elements for 3D meshes.

preferred to be always used when it is possible because it demands less computational effort than other elements. However, for complex geometries, this element is even impossible to employ.

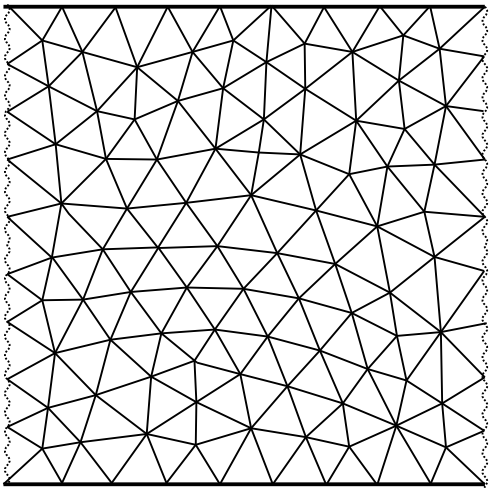
- **Tetrahedron** : A pyramid with triangular base (4 triangular faces), which is used to fill complex voids. The usage of this element is the default configuration for automatic meshing because it fits anywhere, but it is a bad choice to use this type of element for wall boundaries because it does not capture the physics correctly at these regions. Also, it requires more memory and computational effort.
- **Pyramid**: This element is constructed by a square-based pyramid used for transition elements, that is to say, it acts as an adapter to connect a hex mesh to a mesh composed of tetrahedrons.
- **Prism/wedge**: A geometry built by 2 triangles connected by 3 rectangles. Used primarily for boundary layer region where the velocity and thermal gradients are required to be captured.
- **Polyhedron (poly)**: Is an element with many faces (usually 12-30) used in modern fluent simulations since having different neighbors the gradients can be calculated very accurately. The use of this element leads to one-third of the elements having a mesh composed of tetrahedrons, which makes it faster.

A structured mesh is where the elements are arranged in ordered arrays with an indexing shared to adjacent elements (see Fig. 1.9a). It is used for simple geometries, boundary layers or

aerodynamic bodies. Some patterns built by structured mesh are called H-grid, C- grid and O-grid. While for unstructured mesh (see Fig. 1.9b) the elements are in an irregular arrangement with arbitrary node connections with different types of elements capable of discretize complex domains leading to an increment in the total number of elements in the mesh with more numerical diffusion, but with low-effort local refinement in critical regions at the expense of a lower accuracy compared to structured grids.



(a) Structured mesh. Regular grid made primarily by hex elements is 3D or quadrilateral shapes for 2D domains.



(b) Unstructured mesh. Built by a combination of different type of elements, primarily by tetrahedrons in 3D and triangle for 2D domains.

Figure 1.9: Visual difference for a structured and unstructured mesh.

Also, grids can be classified into conformal or non-conformal mesh. In conformal meshes the nodes match exactly at interfaces between regions, thus continuous meshes cross the boundaries with conservative flux transfer. For non-conformal meshes, independent meshing can be obtained for different regions with the capability of local refinement without global impact. These types of meshes are suitable for assemblies with relative motion. Unfortunately, interpolation is required at interfaces, thus they may not conserve fluxes perfectly. A non-conformal mesh is depicted in Fig. 1.10 where independent structured meshes are used to illustrate this idea.

When the elements are close to the boundary walls, and if the mesh is unstructured, prism elements are the best option to occupy in this region, but for structured mesh elongated hex elements are used. A first set of prisms or elongated hex elements are placed over the wall boundary like a “film” with a small height and the next set over the previous one can be placed with a larger height and so on until the total height of such sets cover the sufficient region where large gradients appear. This idea is illustrated in Fig. 1.11 where these described sets

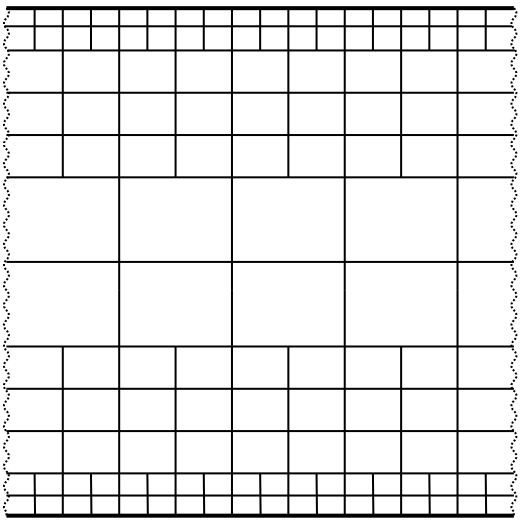


Figure 1.10: An example of non-conformal mesh. Note that some vertices do not coincide with the vertices of adjacent elements.

are called *inflation layers*. When the last layer is constructed a transition to the core of the mesh starts with different types of elements. In Fig. 1.11 the core is built by poly elements, but as mentioned before, the mesh can be built by different types of shapes or a combination of them.

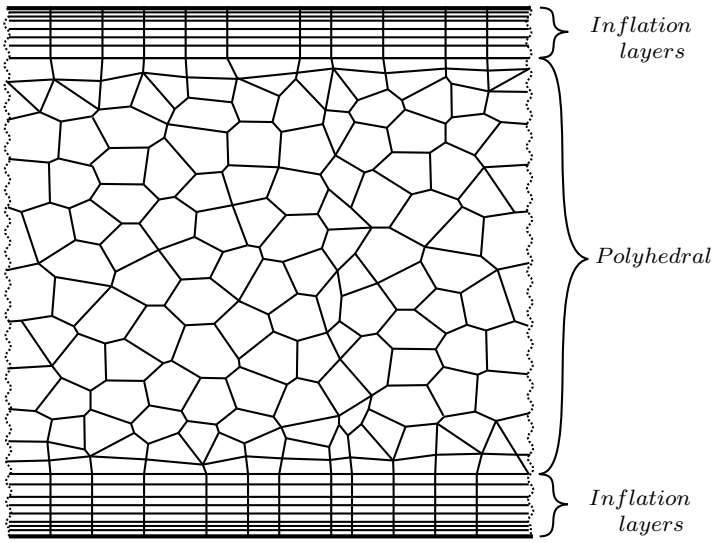


Figure 1.11: At the top and bottom regions wall boundaries are placed to illustrate the usage of inflation layers. Note after the last inflation layer the mesh continues with different type of elements.

1.4.3 Grid quality metrics

When the grid is constructed, the discretized domain must be analyzed by its quality metrics that serve as indicators of a good or bad mesh. Some metrics are more relevant than others, but such criteria depend on the physics to be modeled. For CFD, the most important quality metrics are *skewness*, *orthogonal quality*, and, to a lesser extent, *aspect ratio*, *Jacobian ratio*, and *warping factor*. The skewness measures how distorted an element is compared to the optimal geometry that is represented by a symmetrical shape like an equilateral triangle or a perfect square. The way to compute such metric varies depending on the type of element; for example, in some cases it measures the deviation between the face centroid and the intersection point of the vector connecting cell centers. The values of the skewness are situated between 0 and 1. Being 1 the worst case and when it is 0, the best case. When the skewness is close to 1, the results, at best, will be related to a divergence behavior in the numerical iterations. Orthogonal quality is crucial for solution stability. It measures how well the mesh elements align with the flow direction and the adjacent elements. It is computed using the vector connecting the cell centroids and the face normal vector; that is to say, it essentially checks if the center-to-center line passes perpendicularly through the shared face. The best case for orthogonal quality is when its value is 1, and the worst scenario is when the value is 0 [15]. The maximum allowable value of skewness is 0.85, while for orthogonal quality the minimum value must be 0.1 to ensure the solution will be obtained without any drawback. Aspect ratio compares the longest edge with the shortest one of an element. High aspect ratios (e.g., 100) are acceptable only in the inflation layers aligned with the flow. Jacobian ratio measures element distortion for structural mechanics but is sometimes used in CFD to check corner node distortion. Warping factor measures how much a 2D face is twisted out of plane (non-planar). In specialized CFD software, orthogonal quality is the most important quality parameter alongside skewness because frequently high orthogonal quality is caused by low skewness and vice versa. To obtain good grid quality metrics, the geometry that constitutes the domain could be built in a way that allows the mesh to be constructed in an appropriate manner. Even the choice of elements influences the quality result.

1.4.4 Properties of CFD methods

As well as a good grid giving results that reflect reality, the properties of numerical methods for CFD are a cornerstone of the reliability of results. Any adopted numerical method must have the following properties. The first property is *consistency*, which is related to the discretized

equation that must become mathematically identical to the exact governing equation as the element size approaches zero: The discretization scheme is consistent, ensuring that the truncation error vanishes as the mesh is refined. The next property is related to the *stability* of the numerical method. If a small perturbation is introduced in the solution, it must be damped out, not amplified, as the calculation proceeds. The previous properties lead to the property of *convergence*. A numerical method is convergent when the solution tends to the exact solution of the PDEs as the elements or volumes of the mesh tend to zero. Thus, when the mesh is refined, the physical result stops changing. To know how fine a mesh needs to be, a standard procedure called *mesh independence study* is used to ensure that the solution is independent of the spatial discretization. Due to the nature of the numerical method, also the convergence is related to the iterative error that are called *residuals*, meaning as the solver iterate, the error must drop to zero. The industry standard is to ensure that momentum and continuity residuals fall below 10^{-4} (or 10^{-5}), and the energy residual falls below 10^{-6} [16][14]. Additionally, if a transient simulation is taking place, temporal convergence is also required to be met. The *Courant number* is often used to estimate the required time step. It measures how many mesh cells the fluid passes through in a single time step.

1.4.5 Workflow for CFD simulations

The foregoing concepts are integrated into a simulation tool where the physics can be modelled by their governing equations. Such tool can be created by oneself, or a dedicated commercial/open-source software can be employed. In some cases, the software are divided to perform certain functions, for example from a *computer-aided design* (CAD) file a software can built the discretized geometry only, other software works only for the solving process (this is usually referred to as *solver*) and finally another software can work only for the data extraction and analysis. The execution of CFD simulation adheres to a rigorous, systematic workflow divided into three primary phases: *pre-processing*, *solving*, and *post-processing*. The process initiates with pre-processing, where the fluid domain is geometrically defined and discretized into a computational mesh while simultaneously assigning appropriate boundary conditions and fluid properties. Subsequently, the solving phase employs numerical algorithms to iteratively solve the discretized governing equations until the solution residuals diminish to an acceptable tolerance, ensuring the conservation of mass, momentum, and energy. Finally, the post-processing phase involves the qualitative visualization of flow structures and the quantitative extraction of physics data, allowing for the assessment of numerical accuracy and the physical validation of the simulated phenomena.

1.5 Introduction to machine learning

Since the middle of the last century, *artificial intelligence* (AI) has been developed in order to mimic human cognitive capabilities using computational systems. At present, AI is applied across various fields of study as a result of rapid advancements in computing power. AI is developed to perform tasks based on discovery, inference, and reasoning by distinguishing among different given inputs like sounds, images, text, movement, etc. Previously, the way AI systems were built was based on different techniques like *expert systems*, which are computational structures hard-coded with complex 'if-then' logic rules to make decisions. For example, for *natural language processing*, the ability of a computer to understand, interpret, and generate human language was required to create computational linguistics, syntax trees, and grammatical rules to create a functional system. These traditional AI systems are incredibly rigid. If they encounter a situation that falls even slightly outside their pre-programmed rules, they may produce an erroneous or nonsensical output, and this behavior will persist in similar cases unless the underlying rules or code are explicitly modified. To overcome such flaws, a tool named *machine learning* was created as a way to *train* AI systems.

1.5.1 Basic concepts and classification

Machine learning (ML), also known as *predictive analysis* or statistical learning [17], replaced the traditional programming process where hundreds of rules were needed to be written for the creation of an AI system. Instead, a ML algorithm finds the hidden correlations mathematically. Essentially, ML is a process that extracts knowledge from data. Several algorithms exist to be executed as a ML system that resolve diverse scientific problems.
Some algorithms rely on simple processes that facilitate the interpretation of their functionality, while others are created to solve highly complex problems that lead to a 'abstract' understanding about what is happening inside their code. An algorithm is defined as a *black box* when its underlying processes remain incomprehensible to human observers.

ML algorithms can adopt different approaches, for example, probabilistic, geometric, optimization, information-theoretic, or topological approaches. The selection of an algorithm requires a prior understanding of the available data and the specific problem to be solved. In addition, ML methodologies are fundamentally categorized into three distinct paradigms based on how algorithms interact with information. In *supervised learning*, algorithms are trained using meticulously labeled *datasets*, allowing the model to learn the mathematical mapping between known inputs and their corresponding correct outputs to accurately predict future events. Conversely, *unsupervised learning* algorithms are provided with raw, unlabeled data;

their primary objective is to autonomously discover hidden structures, inherent groupings, or anomalies within the dataset without any explicit human guidance or predefined answers. Finally, *reinforcement learning* operates on a behavioral level by placing an algorithmic agent into a dynamic environment where it must learn to make optimal sequential decisions through a continuous process of trial and error, adjusting its strategy to maximize cumulative rewards and minimize penalties based entirely on the consequences of its actions.

For either supervised or unsupervised learning, it is important the quality of the data that is used to feed the ML algorithm. Usually, the data is presented as sorted lists where each row is known as *sample* or *data point*, while the columns are called *features*, but if the dataset is destined to be used in a supervised ML algorithm, there must be a column with the output values that are related to the features. Such output values are called *targets*. Based on features and targets, the two principal tasks of a supervised ML algorithm are *regression* (the prediction of a number) or *classification* (the prediction of a category). For unsupervised learning the most common task is *clustering* (finding natural groups). Other activities are the *anomaly detection*, *dimensionality reduction* of the data, and for reinforcement learning, *optimization* and *control*.

1.5.2 Application to this work

The integration of ML with CFD is fundamentally driven by the need to overcome the waiting time and increased energy consumption that CFD could demand. One way to address this is by using results from previously completed CFD simulations to train a machine learning model. In this way, the model learns the relationship between the input variables and the flow response, and can later provide fast predictions without solving the full set of equations again. Since simulations and experiments can generate large amounts of data, ML becomes a useful tool to extract patterns, simplify complex problems, and support modeling tasks in fluid mechanics. However, because fluid behavior follows physical laws, these models cannot be used blindly [18]. It is necessary to ensure that their predictions remain consistent with the physics, either by validating them against CFD results or by incorporating known constraints into the model.

To understand in a deep manner how the vortex generators boost the heat transfer in FTHE, multiple simulations are needed to analyze how the variations in the characteristics of each VGs configuration modifies the behavior of the thermal energy exchange. When the study of such behavior depends on the number of simulations (or experiments, if that were the case), the study tends to be large or even unfeasible because it is highly time-consuming and costly. To overcome such drawbacks, ML is an excellent alternative to the high number of simulations. Given that for this project the dataset will be constructed without relying on preexisting data,

such data will be obtained in a way that the dataset include enough information to build a ML *model* capable of predicting new data for 'unseen' VGs configurations. This permits evaluating the effects of VGs while significantly reducing the required simulation overhead.

Chapter summary

Until now it has been reviewed that to increase the heat transfer in HEs the common strategy is to increase the surface area, but in some cases such heat transfer can be decreased due to the formation of a boundary layer that could act as a thermal resistance. This may reduce thermal performance in FTHE systems. Specifically, the air acting as the working fluid on the external side (Figure 1.3) has thermal characteristics that intensify the impact of the boundary layer. In a way to fix this, VGs are considered to increase the heat transfer, but due to the increment in the pressure drop, the application of VGs must be studied based upon the influence of the configurations that the FTHE with VGs can adopt. To accomplish this, a synergy with CFD and ML is sought to obtain the relationship between the input features and the target values.

CHAPTER 2

DESIGN OF EXPERIMENTS USING CFD AND MACHINE LEARNING

For this chapter the methodology for the development of the ongoing project is presented by adopting the *design of experiments* (DoE) approach focused for *computer experiments*. First, the overall idea introduced in the foregoing chapter is introduced for a specific case of study, and afterwards, the input parameters and bounds of this specific case are determined as well as the output variables. Knowing the problem to be resolved, DoE is then applied to build a series of steps that will lead to the integration between CFD and ML for the optimization and analysis processes for the HE. Throughout this project, diverse scripts were developed in Python using different libraries.

2.1 Case study description

Vortex generators augment heat transfer by disturbing the main flow, which in turn disrupts the boundary layer growth (see §1.3.2). Several shapes and sizes for VGs have been studied [9][13]. Most recently Luo et al. [19] has presented a greater review where it is mentioned the possibility of taking advantage of relatively new technologies like *shape memory alloys* or *additive manufacturing*, but also presented a list of possibilities and advantages obtained by the integration of ML and CFD, facilitating the execution of this project. The majority of cases presented in the reviews address a limited number of *design degrees of freedom* (DDoF), which correspond to the independent parameters that can be changed, for example, the position, height, or orientation of a given VG shape. Moreover, different studies carried out different analyses with similar VGs, diverging from a standardized approach.

Based on the recommendations given by the reviews, the studies of Bai et al. [10] and Xu et al. [11] were considered for their continuation by covering aspects they did not take into account. At this point, the main idea is to propose some changes in their designs while keeping important aspects of their HE model, for example, the size or geometrical construction. This aims to enable both studies to complement each other by standardizing such important aspects. It is important to mention that Bai et al. [10] retrieved the *baseline* HE from the work presented by Sinha et al. [20]. Such HE is a FTHE without any VGs. Thus, this is a starting point for standardization because even if different VGs are studied, the resulting behavior can be compared due to the related baseline HE.

Xu et al. [11] also used a baseline HE obtained from another study; however, there are more possibilities if the development of the current project is based on the work of [10] because their scope was limited. Therefore, the study of these latter authors is considered to be used as a reference for the current project, specifically utilizing their proposed VGs called bow-shaped VGs, whose representation is depicted in Fig. 2.1.

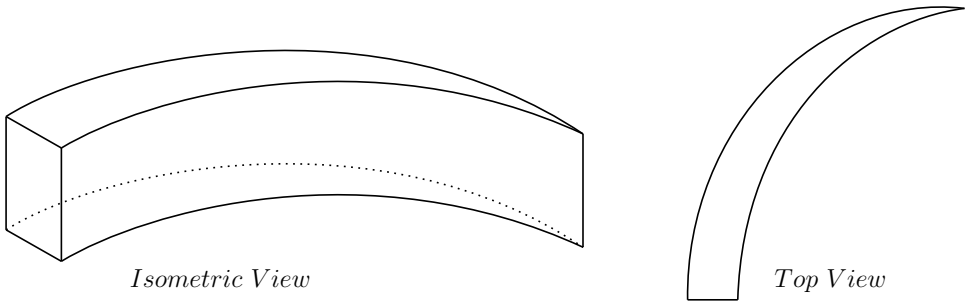


Figure 2.1: Bow-shaped VG by Bai et al. [10].

Their DDoF are based on the possibility of changing the position of the VGs for a fixed orientation, as shown in Fig. 2.2. Therefore, there are three DDoF for that study. Note that this figure represents a portion of the FTHE, where the circumferences represent the tubes that cross the fins perpendicularly.

For this reference, only an in-line tube arrangement was considered; however, due to the geometry of the VGs, if they are utilized for a staggered arrangement, the use of such VGs becomes impractical because the flow could stall in a recirculation zone caused by the VGs. Considering that only an in-line tube arrangement is the feasible option by using bow-shaped VGs, the tubes can be changed from circular to elliptical tubes, as [11] studied. Wang et al.

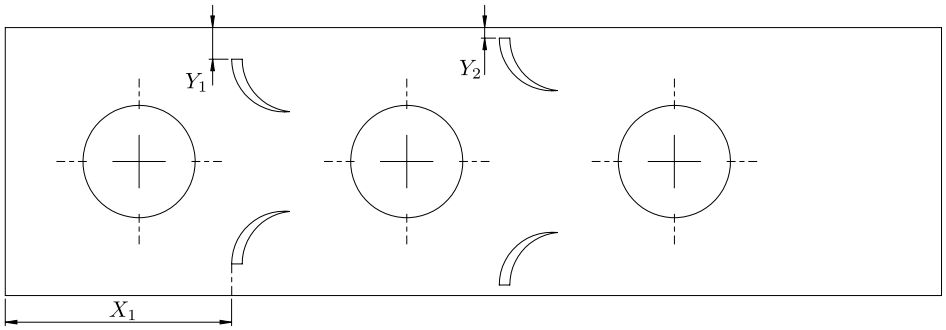


Figure 2.2: Geometry arrangement proposed by Bai et al. where the parameters x_1, y_1 and y_2 represent the design variables. *Modified from [10].*

[21] performed an investigation where an FTHE without VGs, but with elliptical tubes, offers a higher performance than the FTHE with circular tubes. Such a study also established the *tube ellipticity ratio* (the relation between the minor and major diameter of the ellipse) to be a variable, as well as the orientation of the tubes.

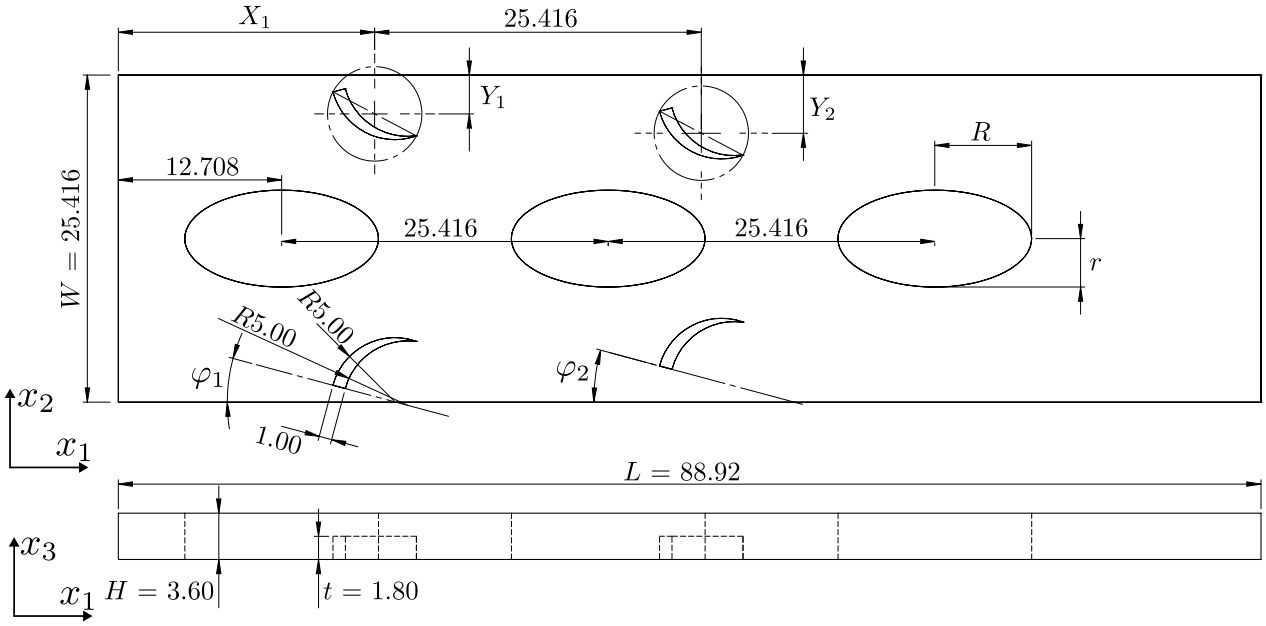


Figure 2.3: Dimensions for the portion of the FTHE. Values in mm.

In light of the previous context, the case of study for the project in progress will be performed using the baseline HE presented by Sinha et al. [20], which in turn is adopted by Bai et al. [10]. Based on this geometry, the first modification proposed is that the tube bundles will be elliptical

rather than circular. Furthermore, the bow-shaped VGs are adopted, keeping the same DDoF as shown in Fig. 2.2, but allowing the VGs to rotate. Hence, for this project, the number of DDoF is five. The dimensions of the geometry that represents the FTHE under study are depicted in Fig. 2.3, where the variables X_1, Y_1, Y_2, φ_1 and φ_2 are the design variables.

The diameter of the tubes in the original baseline is 10.656 mm, corresponding to a cross-sectional area of 89.182 mm^2. This value is preserved to determine the values of R and r for the ellipses when the value of the tube ellipticity ratio is equal to 0.5 (r/R). Under this condition, the value for R and r are equal to 7.535 mm and 3.768 mm, respectively. The positions and dimensions of the remaining elements are kept as the reference studies have established. The VGs are arranged in two successive columns along the flow direction. The first column corresponds to the pair of vertically aligned VGs encountered first by the incoming flow, while the second column corresponds to the downstream pair. The parameters Y_1 and Y_2 define the vertical positions of the first and second columns, respectively. Similarly, the angles φ_1 and φ_2 define the orientation of the VGs in each column, while the parameter X_1 simultaneously controls the streamwise position of both columns. When $\varphi_{1,2} = 0°$, the VG orientation corresponds to the baseline configuration studied by Bai et al. [10].

The constraints for this study are presented in Table 2.1. They were determined by considering the utility of having the VGs at certain positions. If any of the angles φ are greater than 45°, due to the shape of the VG, the flow will be reversed. With the same logic, if the VGs are positioned with a relatively large value of X_1, their effectiveness tends to decrease. This is because VGs are generally intended not only to enhance mixing, but also to reduce the wake region behind the tubes.

Table 2.1: Range of values for the design variables.

	range		
parameter	**min**	**max**	**units**
X_1	15.5	25	mm
Y_1	3	6	mm
Y_2	3	6	mm
φ_1	0	45	deg
φ_2	0	45	deg

After considering the aspects discussed above, this project will study a FTHE that uses elliptical tubes with bow-shaped VGs inserted over its fins with the dimensions presented in Fig. 2.3 and with the range of values for each design variable presented in Table 2.1. The concept of the FTHE portion is illustrated in Fig. 2.4.

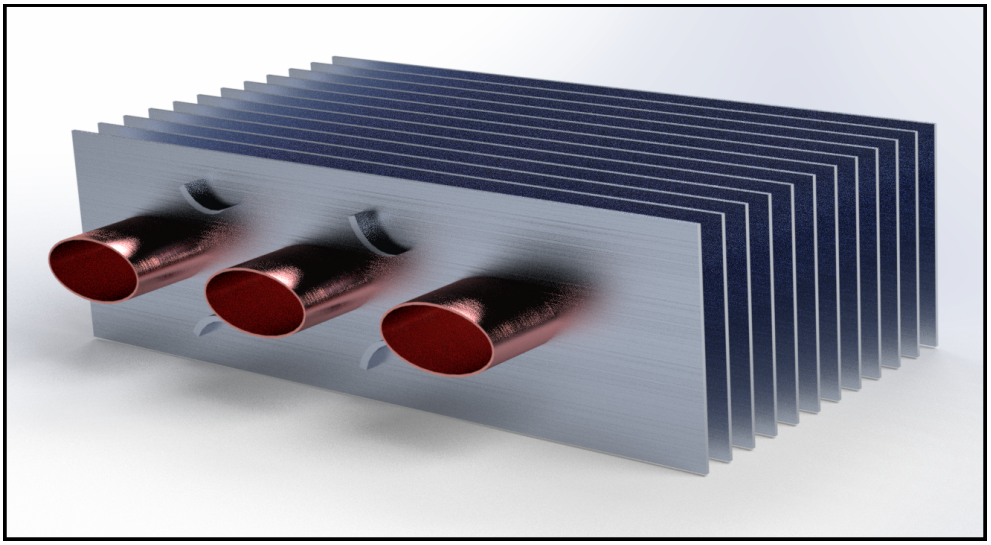


Figure 2.4: Concept of the portion of the FTHE under study.

2.2 Methodology for computational experiments

The study of experimental design is fundamental to the scientific and engineering methods, serving as a critical tool for technology commercialization, product realization, and process improvement. An experiment is formally defined as a series of tests or runs where purposeful changes are made to input variables to observe and identify changes in the output response [22]. Unlike simple observation, experimental design allows for the discovery of cause-and-effect relationships and the development of empirical models. The primary objective of statistically designed experiments is to optimize systems by identifying influential variables, reducing variability, and ensuring products or processes are robust against uncontrollable factors. By moving away from inefficient strategies such as "best-guess" or "one-factor-at-a-time" approaches and toward factorial designs, experimenters can achieve higher data efficiency and identify critical interactions between variables.

Experimental data can be generated through physical testing or computational models. In physical experiments, there could be too many factors that can introduce additional variations that are not desirable, like variations due to different time periods, weather conditions, locations, human bias, equipment reliability, and so on. These factors mean that the design of experiments must rely on three principles that must be applied to every physical experiment:

- *Randomization:* This averages out the effects of extraneous factors and ensures observa-

tions are independently distributed random variables.

- *Replication:* Independent repeat of each factor combination. It allows for the estimation of experimental error and increases the precision of the estimated mean response.
- *Blocking:* A technique used to reduce or eliminate variability from nuisance factors (e.g., different batches of raw material) that are not the primary focus of the study.

On the other hand, computer experimentation is feasible when physical experimentation is not, but it can be taken into account as a cheaper alternative or for exploratory analysis. The difference between physical and computational experiments lies in the fact that physical experiments measure stochastic responses while computational experiments obtain results from deterministic procedures [22]. Due to this difference, computational experiments frequently do not require randomization, blocking, or replication; however, some codes or solvers yield stochastic outputs because the outputs can be modeled as having a non-deterministic noise component when the analysis is carried out on a highly sensitive-to-initial-conditions or transient fluctuating experiment; thus, to separate the physical signal from noise, replication is reintroduced. Additionally, if the quality of the mesh is not well handled, the variations from grid to grid will introduce noise that can affect the computational responses.

For this project, computational experiments by CFD and ML are the backbone for the study of the influence of the bow-shaped VGs for the previously presented FTHE; hence, the methodology is based on these considerations. It was mentioned that the way to study the influence of VGs is by comparing the value of η (Eq. 1.16); therefore, the computational experiments must be focused on obtaining the values of Nu and f for each run or sample from CFD simulations to build a dataset for the ML training process.

2.2.1 Experimental designs

Using the previously defined parameter ranges in Table 2.1, a five-dimensional design space is established to generate the configurations evaluated in the CFD simulations. Based on this, it is easy to be tempted to create a list of configurations based on random positions for each parameter; however, this approach will lead to an uneven sampling that could yield an inadequate space mapping. That is to say that the sampling could be biased to a corner or directly at the center of the space due to the randomness neglecting important regions. As a result, the ML model may produce inaccurate predictions when evaluating configurations located in these unseen regions. To avoid this inconvenient phenomenon, a *space-filling* technique must

be embraced. The most used technique is called *latin hypercube sampling* (LHS) which ensures points are evenly spread across the range of each individual input. For N samples from a d-dimensional space, standard sampling (Monte Carlo) simply picks N points completely at random. LHS improves upon this by forcing stratification across every single dimension [22]. Stratification simply means dividing a large group or space into smaller, non-overlapping subgroups (called *strata*) before a sample is taken. The goal is to ensure that every subgroup is represented fairly in the final data. While LHS distributes each parameter uniformly, it does not always guarantee a good distribution of points in the complete design space as the number of variables increases. As a result, some regions may contain many samples while others remain unexplored, which can reduce the accuracy of the ML model in those poorly sampled regions.

Better alternatives to LHS are given from sampling techniques known as *Quasi-Monte Carlo* methods. Such methods do not use randomness at all. Instead, they use strictly defined mathematical formulas to generate *low-discrepancy* sequences usually referred as quasi-random sequences. From this class of techniques, Sobol' method stands out for this project.

2.2.2 Sobol' sequences

Sobol' method is mathematically engineered to generate low-discrepancy sequences (Sobol' sequences), meaning they are explicitly designed to fill a multi-dimensional space as uniformly as possible without leaving large gaps or creating dense clusters. Standard random number generators use complex formulas to create *pseudo-randomness*. Sobol' sequences do not require randomness at all; they are constructed by bisection operations. The math is entirely deterministic and relies heavily on base-2 or binary arithmetic, radical inversion and bit-by-bit exclusive-or operations. Through bisection, each dimension is separated evenly by converting an integer index into binary and mirrors it across the decimal point. This transforms a linear counting sequence into a fraction sequence that naturally seeks the center of the largest empty gaps. To prevent multiple dimensions from collapsing into a single diagonal line, each dimension is assigned a unique, mathematically irreducible *primitive polynomial*. The polynomials generate a unique set of fractions called *direction numbers*. Based on the complex math behind the Sobol' sequences generation an algorithm has been created to generate the sequences that maps the d-dimensional space as desired. The algorithm uses bit-by-bit exclusive-or operations to combine these numbers to act as a deterministic shuffle, ensuring that the bisection sequence of every parameter is mathematically independent [23]. This guarantees that if a sample size N (where N is a power of 2) is generated, the points will be so perfectly distributed that exactly

one point will fall into every single equal-volume subdivision of the space, regardless of whether the space is sliced into squares, long columns, or flat rows. If the sample size is not a power of 2, the resulting sampling is slightly uneven distributed.

The execution of the algorithm can be done by a library called *SciPy* available for Python. SciPy contains algorithms for scientific computing in Python, and it provides functions for statistics, optimization and signal processing [24]. Through SciPy, the function `qmc.Sobol()` is utilized to initialize the sequence generator, then considering the bounds that have been established in Table 2.1, the function `qmc.scale()` is called to create the list of the different points that will use to map the 5-dimensional space:

Listing 2.1: Output from the Sobol' sequence generation.

```
          X_1      Y_1      Y_2     phi_1     phi_2
0     18.8580   4.4747   4.6917   12.0642    3.8001
1     22.3938   4.9061   3.6822   33.5857   43.9684
2     24.5324   3.1976   5.9181    4.5520   24.2714
3     16.1262   5.4337   3.9776   39.7845   17.8720
4     16.7237   3.6355   3.2098   38.9658   14.9022
..        ...      ...      ...       ...       ...
251   16.7492   5.2485   5.4986   10.6438   30.8639
252   16.1516   3.8237   4.7780    0.6060   27.9480
253   24.4884   4.7956   3.7244   43.7281   19.9086
254   22.3496   3.5070   5.9871   16.5606    0.2552
255   18.8836   5.8617   3.9909   29.0966   41.9763

[256 rows x 5 columns]
```

A list of $2^8 = 256$ points was created. If additional points are required, it is possible to request $2^9 = 512$ points in the script, where the first 256 points will be the same as before, but with new points starting from the 257th point.

A way to prove that the list has been created by evenly relations between the dimensions of the space is through a matrix scatterplot that shows the space-filling efficiency of the Sobol' sequences. In Fig. 2.5 the relations between the design variables are shown. The diagonal shows the 1-Dimensional distribution of each individual variable independently. Each bar represents an interval within each parameter range. Since the bars tend to be flat, this confirms that

the Sobol' sequence is sampling the parameter space uniformly. The scatter plots show the 2-Dimensional projection of how two variables interact. Even though it can be seen that some points are overlapping, this is not a cause for concern due to the fact that this is an attempt to show the 5-dimensional interaction into a simpler 2-dimensional relationship.

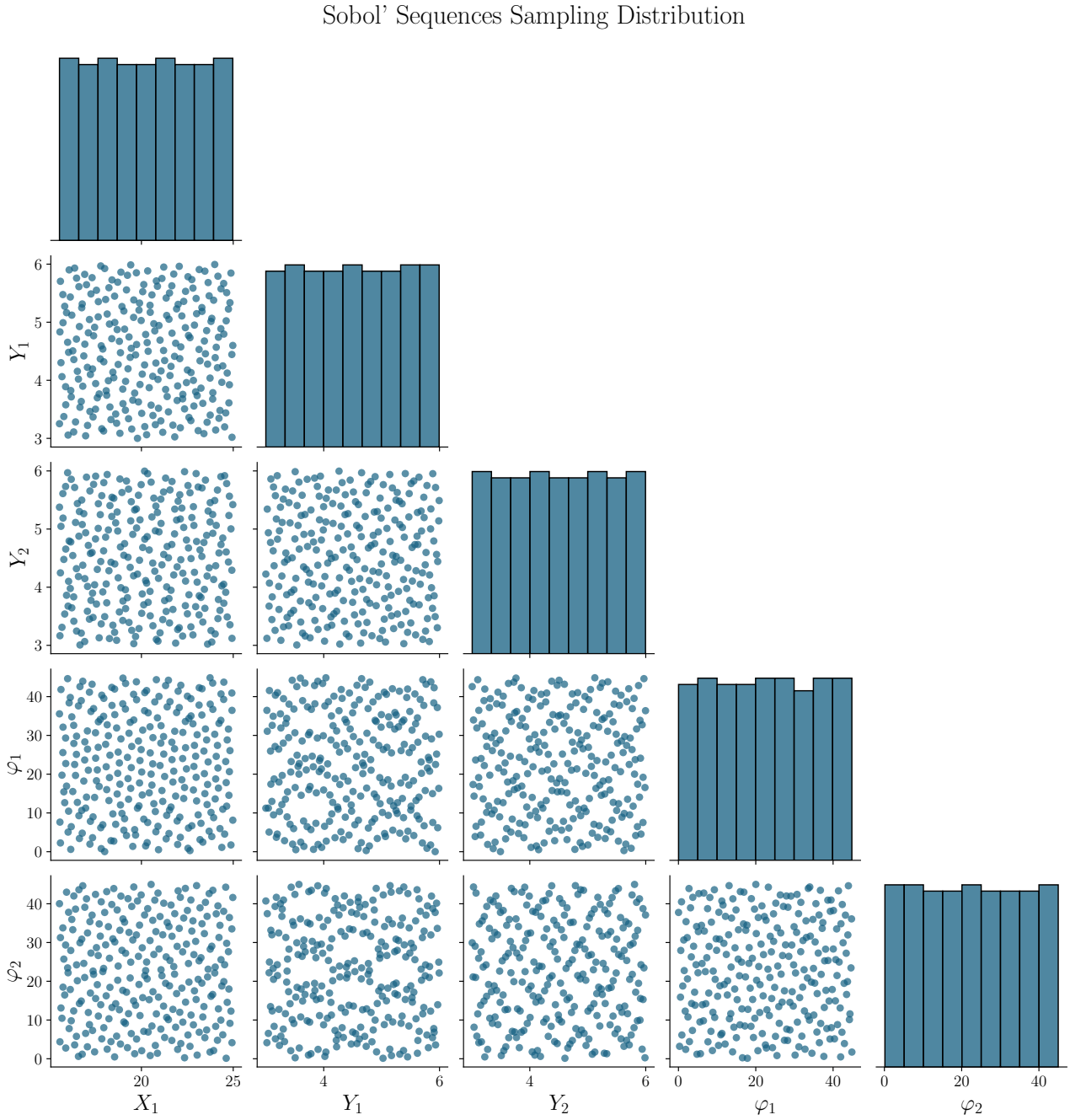


Figure 2.5: Matrix scatterplot with histograms for the relationship between the design variables.

2.2.3 Active learning and surrogate models

The points obtained from the Sobol' method will be used to run CFD simulations, and along with the outputs that these simulations will provide, some ML algorithms will be employed to create specific ML models. The idea is to simulate a reduced number of cases, but the ML statistics will define how many points are required for reliable predictions. If the simulations based on the points obtained from the Sobol' method are not sufficient, a technique called *active learning* can be employed in lieu of generating additional points by space-filling methods. Active learning is simply the machine learning term for sequential design. In this approach, the points where the model is expected to perform poorly are specifically selected creating a cycle between gathering new information and updating the model after further data from CFD simulations [25]. For this project the *Active Learning MacKay* method is adopted. This method evaluates the model and selects the next point exactly where the current *predictive variance* or *uncertainty* is the highest. The perfect ML algorithm for this method is through the *Gaussian Process Regressor* (GPR). A GPR calculates predictive uncertainty by taking the prior variance at a new point and subtracting the information provided by the existing training data. The details of the GPR will be covered in a future chapter.

After the active learning process, a reliable ML model can be obtained to be used as a *surrogate model*, which will serve as the 'calculator' for the optimization process where thousands of points are required by the optimization algorithm. In a similar way to the active learning, the results obtained from the optimization process through the ML model will be validated by CFD simulations as well. When this stage is reached, the general study of the influence of VGs in the behavior of FTHE can be carried out.

2.2.4 Workflow process overview

The overall workflow is shown in Fig 2.6 where the integration of CFD simulations and the usage of ML models is presented by close-loop processes. At this point, the generation of design points for sampling is already presented, thus the methodology for the simulations and ML setup are discussed in greater depth later on.

Chapter summary

The DoE has been adopted as a methodology applied to computational experiments where randomization, replication (if applicable) and blocking are ruled out, after that the case of study

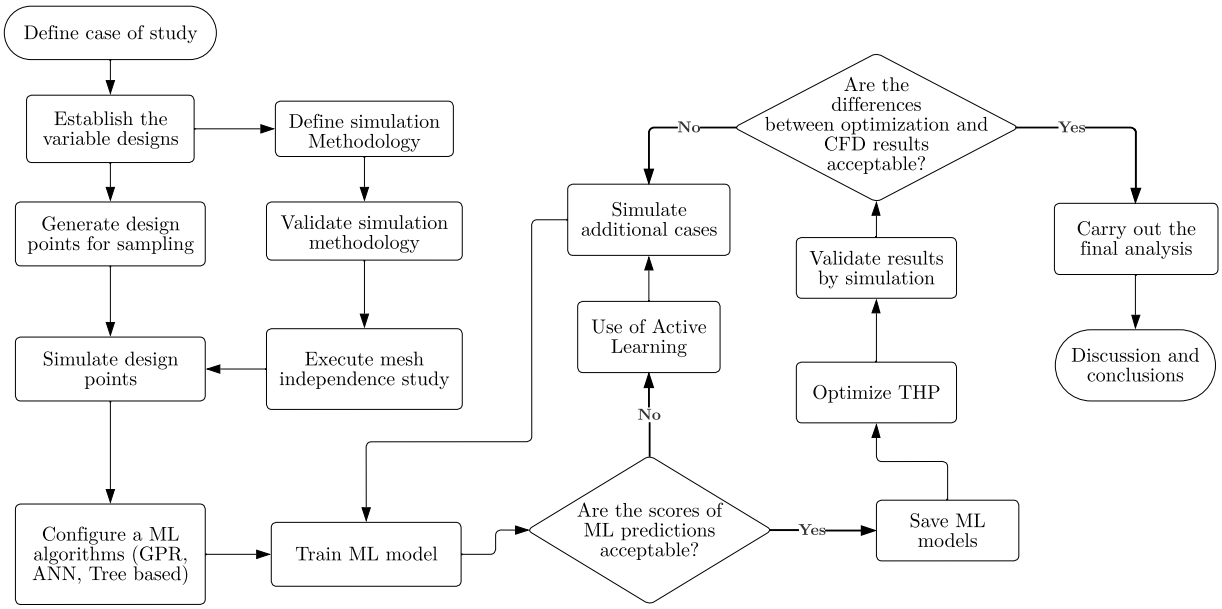


Figure 2.6: General workflow process for the project development.

has been defined along with its design variables. Five design variables have been established, forming a 5-dimensional space. Such space is planned to be mapped by a list of sampling points obtained by the Sobol' method. These points are used to create simulations cases to obtain the results that are utilized to build and train a ML model. The active learning approach takes action to map certain regions that were not covered by the former list. The details about CFD simulations, ML algorithms/models as well as the optimization are discussed in more depth later on.

CHAPTER 3

DEVELOPMENT OF CFD SIMULATIONS FOR HEAT EXCHANGERS WITH VORTEX GENERATORS

In this chapter a simulation methodology is developed for the FTHE by selecting and adapting the most suitable solver options for the present case. The simulation workflow is explained from the creation of the computer-aided design (CAD) files to the execution of the simulations. This workflow is carried out based on the validation results and on the data obtained from the mesh independence study. At the end of the process a dataset is obtained from the list of points created by the Sobol' method, which is used for the consecutive steps.

3.1 Preliminary modeling considerations

As part of the effort to maintain consistency with previous FTHE studies, the Re was fixed at 500, following the configuration presented by Bai et al. [10]. To compute Re, the hydraulic diameter (D_h) used for this type of study is the same as that utilized for the parallel plates that are extended infinitely along the x_2-axis in both directions. This D_h is described by Eq. 3.1 [26].

$$D_h = \frac{4A_c}{P_c} = \lim_{W\to\infty} \frac{4WH}{2W + 2H} = 2H \tag{3.1}$$

This result is used as the characteristic length in the Eq. 3.2 to compute the Re.

$$Re = \frac{2H\rho U}{\mu} \tag{3.2}$$

Continuing with the idea of standardization, the flow enters the computational domain at a temperature of 300 K, while the fins, tubes, and VGs are maintained as isothermal surfaces at 330 K. The variation in the air properties in this temperature range yields Mach numbers much less than the unity (Ma $\ll 1$); thus, considering the common engineering criterion that flows with Ma < 0.3 may be considered incompressible, the present flow was modeled as incompressible [26]. In addition to the above, the variation in the air properties can be neglected by fixing their values when the temperature is 315 K. The corresponding thermophysical properties of air are presented in Table 3.1 [5].

Table 3.1: Thermophysical properties for air at 315 K [5].

property	value	units
ρ	1.1115	$^{kg}/_{m^3}$
μ	1.917E-05	$^{N\cdot s}/_{m^2}$
c_p	1007.6	$^{J}/_{kg\cdot K}$
k_f	0.02741	$^{W}/_{m\cdot K}$
Pr	0.705	-

Based on the selected geometry and properties, the value of the average velocity profile must be equal to 1.1976 $^{m}/_{s}$ for $Re = 500$. From the literature for parallel plates, the flow is considered laminar if Re is less than 2000 [26].

The simulations were executed through Ansys® Fluent (hereinafter referred to as Fluent), version 2023 R1[1]. Fluent is known for its high capability in solving different types of flows through the FVM and its compatibility with different mesh file extensions. Due to its wide flexibility, the solver setup is carefully managed as mentioned later on in this chapter.

3.2 A brief overview of turbulence models

As previously established, the Re for this project is set at 500, which represents a laminar fluid flow; however, this criterion is valid for two parallel plates without any protuberance or obstacle along the flow direction that disturbs the uniform flow. The influence of the VGs and the presence of the tubes modify the flow in different manners; thus, further considerations must be made, like the use of *turbulence models*.

[1] Academic license held by the *Center for Research and Postgraduate Studies* (CIEP).

Turbulence is a three-dimensional phenomenon where unsteady random motion is observed. This occurs when the dissipation terms are overtaken by the advection terms in the Navier-Stokes equations. To solve directly these equations for a fully turbulent flow, the technique called *direct numerical simulation* (DNS) must be employed to capture all the nuances that the turbulent field contains. This technique is time-dependent and requires a fine mesh to capture all the turbulence scales. The scales are described through the *energy cascade* concept that is related to the eddies movement (spinning or swirling fluid structures) and how they evolve in time. The energy cascade refers to the situation where the kinetic energy (k) from the larger eddies is transferred onto smaller eddies, and from those smaller eddies, onto even smaller eddies, and so forth, until a physics-based minimum eddy length scale is reached. The context where the turbulence is taking place is important to establish a characteristic length which in turn defines the order of magnitude for the *integral*, *Taylor* and *Kolmogorov* scales that are represented in Fig. 3.1. In the integral region, integral eddies or large eddies can be as large as the characteristic length of a system, and even larger on occasion, up to about twice as large, while the Kolmogorov represents the smallest scale for the turbulence. Taylor scale is an intermediate scale where the size of the eddies can strongly influence the flow [27]. The smallest eddies that can be sustained by the turbulence are the Kolmogorov eddies, which, depending on the context, can be as small as the diameter of a human hair. In general, Kolmogorov eddies tend to be larger than the viscous boundary layer near the wall.

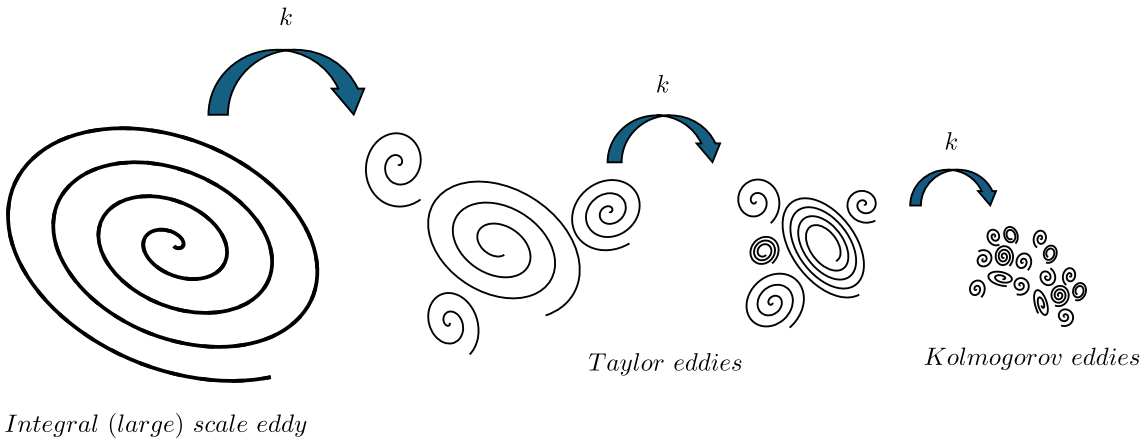


Figure 3.1: Turbulence dissipation described by the cascade energy. *Modified from [27].*

The usage of DNS is translated as a high computational power requirement because in order to capture the physics in the Kolmogorov scale, the size of the cells in the mesh must be considerably smaller than the smallest eddy. To avoid this drawback, *Large Eddy Simulation* (LES) has been introduced as a reliable option when the turbulence movement is required for study.

LES can capture the integral and Taylor eddies, but if the Kolmogorov scale still represents importance in the flow, it can be modeled by a *eddy viscosity* model. Even though LES is less expensive than DNS, it still requires high computational cost. Eddy viscosity (ν_t) is a core concept in turbulence modeling. It is not a true physical property of a fluid, but a property of the flow state analogous to the kinetic viscosity (ν), a highly effective mathematical tool invented to simplify the chaotic equations of turbulent flow. It depends entirely on the size and speed of the turbulent eddies. For example, a raging river has a massive eddy viscosity, while the exact same water sitting still in a glass has an eddy viscosity of zero.

Through these considerations, the turbulence can be studied as a representation of *time-average* of the entire flow. When the flow is averaged over time, the result completely blurs out all the chaotic, swirling eddies and only calculates the smooth, steady mean flow. But because all the turbulence was averaged out, the model must use eddy viscosity to account for the whole turbulent mixing at desired scale. To model turbulence by this mean, the flow is studied as an aggregate of two velocities: the mean velocity and the fluctuating velocity through the *Reynolds decomposition* (Eq. 3.3).

$$u(x_i, t) \equiv \langle u \rangle (x_i) + u'(x_i, t) \tag{3.3}$$

Where $\langle u \rangle$ is the mean velocity field, and u' is the velocity fluctuations. On account of the working fluid is considered as incompressible, the Eq. 1.2 can be simplified as follows,

$$\rho \frac{\partial u_j}{\partial t} + \rho u_k \frac{\partial u_j}{\partial x_k} = -\frac{\partial p}{\partial x_j} + \mu \frac{\partial^2 u_j}{\partial x_i \partial x_i} \tag{3.4}$$

After averaging the right-hand side of the Eq. 3.4 and applying the averaged form of the *material derivative* that is obtained by the application of the Reynolds decomposition in the left-hand side of the former equation, the Navier-Stokes equations can be transformed into the *Reynolds-averaged Navier–Stokes equations* (RANS) (Eq. 3.5).

$$\rho \frac{\partial \langle u_j \rangle}{\partial t} + \rho \langle u_k \rangle \frac{\partial \langle u_j \rangle}{\partial x_k} = -\frac{\partial \langle p \rangle}{\partial x_j} + \mu \frac{\partial^2 \langle u_j \rangle}{\partial x_i \partial x_i} - \rho \frac{\partial}{\partial x_k} \langle u'_k u'_j \rangle \tag{3.5}$$

The terms that are closed by the angle brackets $\langle \dots \rangle$ represent average values. Note the simi-

larity between the Eq. 3.4 and Eq. 3.5 where all velocity and pressure terms are averaged in the latter equation, but the major distinction is the apparition of the last term in the right-hand side. This last term ($-\rho\langle u'_k u'_j\rangle$) represents the *Reynolds stresses*. Because of the nonlinearity of the Navier-Stokes equations, the Reynolds stress appears after the averaging of such equations. Originally, the conservation of mass equation and momentum equations contain four unknowns, but the Reynolds stress adds 6 more, thus the number of unknowns amount to 10 knowns for only four equations, this is called the *closure problem*. Here is where eddy viscosity comes into being. Different techniques to model the eddy viscosity have been yielded, but the most significant proposal was through the Boussinesq approximation or sometimes referred to as the Boussinesq hypothesis. This theory states that the chaotic turbulent mixing acts mathematically just like regular fluid friction, only much stronger. This is where turbulence models (like $k-\epsilon$, $k-\omega$, or Spalart-Allmaras) come in. These models provide a separate, simplified equation to guess the eddy viscosity based on the flow conditions [14][27].

The $k-\epsilon$ model serves as a foundational two-equation turbulence model in computational fluid dynamics, solving transport equations for turbulent kinetic energy and the turbulent dissipation rate (ϵ). While it demonstrates robust performance in free-shear flows it exhibits significant inaccuracies in near-wall regions and flows subject to adverse pressure gradients, frequently necessitating the application of empirical wall functions. To address these near-wall deficiencies, the $k-\omega$ model was developed. By transporting the specific dissipation rate (ω) instead of ϵ, this formulation provides superior resolution of the boundary layer down to the viscous sublayer, yielding highly accurate predictions for aerodynamic drag and flow separation. However, the standard $k-\omega$ model is highly sensitive to free-stream inlet conditions, which can introduce artificial dependencies into the computational domain. To resolve these inherent limitations, the Shear Stress Transport (SST) $k-\omega$ model was introduced as a hybrid formulation. This approach employs a mathematical *blending function* that utilizes the $k-\omega$ equations within the inner boundary layer to accurately capture wall physics, while smoothly transitioning to the $k-\epsilon$ formulation in the free stream to circumvent inlet sensitivity. Due to this robust integration, the SST $k-\omega$ model has become the prevailing standard for complex fluid and thermal engineering simulations.

Because no single model is universally accepted for all classes of problems, the choice of each model depends on the physics of the flow. For exploratory analysis the SST $k-\omega$ model is the first option to use because it is considered for most engineering flows. For this reason, Fluent

set this model as default model for the solver setup [16].

3.3 Validation of the numerical model

A necessary step in numerical simulation is the validation of the methodology employed to simulate a certain case. The simulations can be compared either against validated numerical data or against empirical data. The present study validates its methodology by comparing the results with the data that Sinha et al. [20] provided in their work, which is a benchmark widely used in several studies. For the validation, the FTHE with tubes in staggered arrangement this reference provided is used as reference. Such FTHE is presented in Fig. 3.2. A sole geometrical parameter value was assigned as variable, presented by β, which represents the orientation of the rectangular winglet VGs. The most detailed case in this study is related when $\beta = 165°$, thus this is the case that is replicated to perform the validation of the CFD process. Additionally, Re was established as a design variable as well, varying it from 250 up to 1500 for each β value. Eight Re values were selected to be replicated to ensure coherence in comparisons: 250, 300, 500, 750, 900, 1000, 1250, 1500.

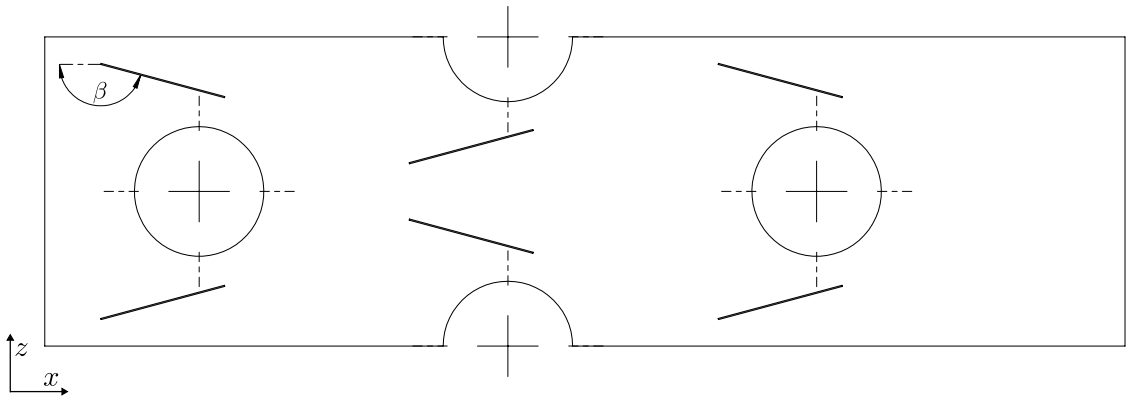


Figure 3.2: FTHE with staggered tubes that Sinha et al. studied. *Modified from [20].*

It must be borne in mind that the general dimensions of this FTHE have been adopted for the present project; therefore, the differences from the benchmark design lie primarily in the tubes and the VGs. The benchmark study specified the boundary conditions as well as the equations to be solved, and the solver setup. Unfortunately, they did not furnish the thermophysical properties of the air. The thermal conditions they used are the same to the ones that were explained in the previous section (see §3.1). Thus, the replications were carried out by adopting the same values presented in Table 3.1.

3.3.1 Settings data

Sinha et al. [20] performed a series of transient simulations with the *SIMPLE* algorithm for the solver computations. For the convection term, a second-order upwind scheme with fixed time step of 0.001 s for 5000-time steps was used. Further information was not provided; therefore, some assumptions were required. The velocity boundary conditions at the entrance were defined as follows:

$$u_x = u(y),\ u_y = 0,\ u_z = 0$$

Where $u(y)$ is the fully developed velocity profile. On the walls (fins, tube and winglet surfaces) they applied the no-slip condition. As mentioned, the temperature of the fluid at the entrance is kept at 300 K while for the walls at 330 K. At the exit of the domain the gauge pressure is set as 0. Each horizontal border is set as periodic boundary (Fig. 3.2), and zero mass flow rate in the spanwise direction was assumed. It was also reported that allowing mass flow in the z-direction did not produce a significant effect on the heat transfer distribution.

The eight simulations are performed by the same mesh; the only parameter that changes is the value of the velocity inlet u_x which variation depends on the Re. In Table 3.2 the values of the average velocity are described.

Table 3.2: Values of average velocity for u_x for each case.

Re	$u_{x,avg}$ [m/s]
250	0.5988
300	0.7186
500	1.1976
750	1.6284
900	2.1557
1000	2.3952
1250	2.9940
1500	3.5928

3.3.2 Meshing process

For the grid construction, the case of $Re = 1500$ is essential because at this regimen the flow is near to the turbulent phenomenon in two parallel flat plates, thus it is imperative to refine the mesh to meet the convergence requirements at this scenario. The computational domain was discretized using the *Ansys Meshing* module as this tool provides greater control over the

meshing process.

The correct way to create the 3D files to be meshed is by modeling the fluid domain instead of the solid interface. Therefore, the modeled volume represents the region occupied by the flowing fluid, that is the empty space surrounded by the solid geometry. The CAD files were created by SOLIDWORKS® (SW). In CFD, the computational domain must be extended in such way that numerical error propagation does not occur. If the physics beyond the artificial boundaries influence the modeled domain, numerical error can appear, and in consequence, the results may not be reliable. For this particular case, the inlet and outlet could require an extension to capture complex movement that could cause inconsistent calculations. A general guideline is to extend the axial length of the inlet region from 5 to 10 times the fins gap (H). The outlet length is extended between $10H$ and $50H$.

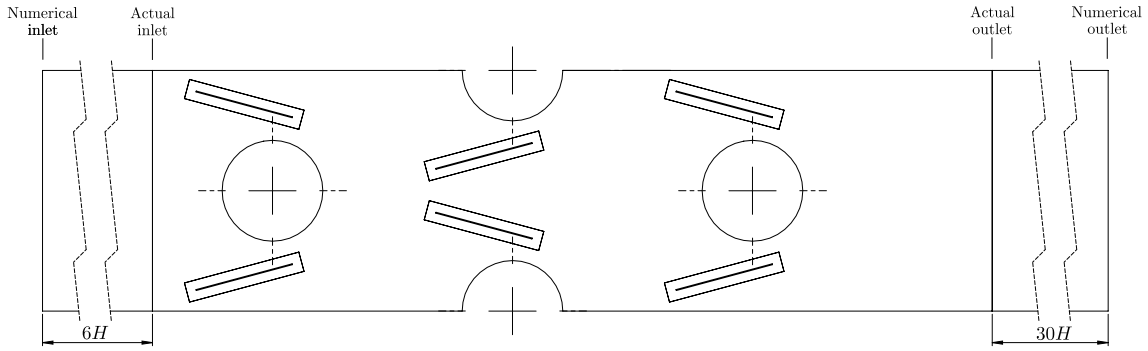


Figure 3.3: Multibody representation of the fluid domain of the Sinha's 3D domain.

These recommendations vary depending on the type of the flow. However, in [20] details were not provided. To address this issue conservatively, the inlet was extended. It was considered extending the inlet entrance up to $6H$ and for the outlet $30H$ as Fig. 3.3 depicts. The main drawback of these extensions is that they increase the number of cells, hence the computational effort is increased. In addition, due to the fact that these extensions do not correspond to the actual physical FTHE construction, heat transfer does not occur from these regions to the flow, thus adiabatic boundary condition is set. Under the same idea, the pressure drop is computed between the *actual inlet* and the *actual outlet*.

Figure 3.3 represents the 3D domain built by different bodies. This technique is used to mesh the domain in a controlled sequence so that the quality of the grid is within acceptable limits. Because the domain is built by different bodies, a setting called *shared topology* must

be enabled to avoid that the solver treats each edge as a boundary.

The important aspect of the mesh is the orthogonal quality and the skewness, thus through the multibody technique different type of elements can be employed to create the grid. The regions surrounding the VGs are particularly prone to generating low quality elements. Therefore the bodies that contain the VGs are built as a structured mesh with a small element size to keep the VGs detail. For the tube regions, it is convenient to build the grid as a structured mesh as well, but this demands to keep the same element size as the VGs, and this will lead to a high number of cells in the whole domain. Instead, tetrahedrons are selected to mesh this region. The inlet and outlet extensions were meshed using structured grids. The transition between the fine mesh surrounding the VGs and the larger structured regions was handled through unstructured mesh zones, allowing a gradual variation in element size across the domain.

Twenty inflation layers were established to capture the physics near the walls. In Fig 3.4 a section plane shows the arrangement of the cells. Note the tiny elements that enclose the vortex generators in a structured way. The tetrahedrons near the walls are built in a small size to settle into the inflation layers and the physics requirements.

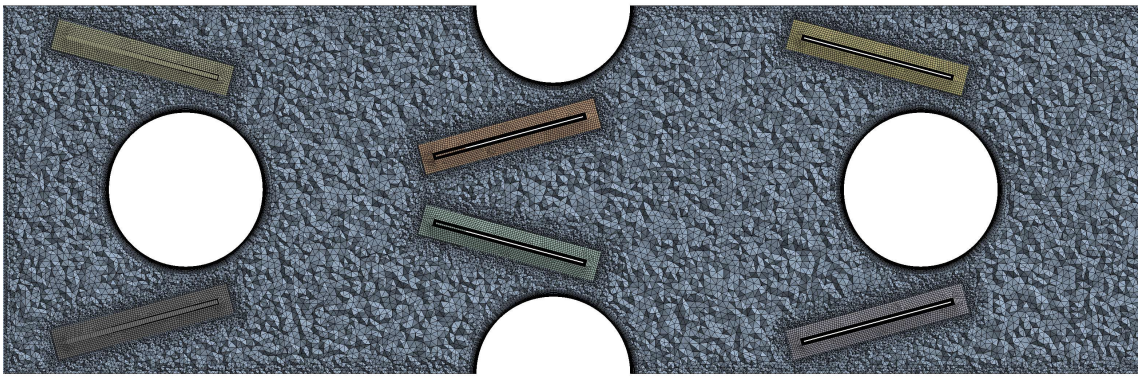


Figure 3.4: Top view of a clipping x-z plane. Note the usage of multibody for mesh quality improvement.

When the mesh was generated, a parameter called y *plus* (y^+) was carefully supervised. This is a dimensionless wall distance, and it is a fundamental parameter in CFD used to characterize the relative mesh resolution near a solid boundary. It is defined mathematically as the ratio of the product of the absolute distance from the wall and the friction velocity to the kinematic viscosity of the fluid. Physically, this parameter normalizes the turbulent boundary layer into distinct universal sub-regions to determine whether viscous forces or turbulent inertial forces

are dominant at a specific distance from the surface. The precise calculation and control of y^+ is strictly required for the proper application of turbulence models, as it dictates the necessary grid spacing adjacent to a wall. Models like the SST $k-\omega$ are explicitly designed to resolve the physics entirely down to the wall. To function correctly, these models mandate a highly refined near-wall mesh where the center of the first grid cell strictly requires a y^+ value of approximately 1. Consequently, matching the y^+ value of the computational mesh to the specific mathematical requirements of the chosen turbulence model is essential to guarantee the physical accuracy of any aerodynamic or thermal simulation. The value of y^+ is defined as follows,

$$y^+ = \frac{yu_\tau}{\nu} \tag{3.6}$$

where u_τ is the friction velocity that can be computed as the following equation depicts,

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{3.7}$$

the parameter τ_w represents the wall shear stress. These parameters are computed by Fluent, and for the created grid, the value of y^+ is met as the turbulence models require.

To make the mesh dependence study, the mesh was refined in succession comparing results by Eq. 1.15 and 3.8. When the mesh size was increased from 6.5 to 8.6 million cells, the difference for $\overline{Nu}$ and f were only 0.01% and 0.13%, respectively. The mesh with 8.6 million cells was used to execute the remaining cases. The characteristics of this mesh are summarized in Table 3.3.

3.3.3 Solver Setup

It was mentioned that the fluid flow in these simulations is considered incompressible because of the relatively low velocity; therefore, pressure-based solvers must be used. These solvers are designed to solve the flow problem through pressure-velocity coupling algorithms. Segregated and coupled algorithms exist for these procedures [14][28]. The segregated algorithms solve the governing equations sequentially (decoupled from one another). An example of segregated algorithms is the SIMPLE algorithm which first solves the momentum equation using a guessed

Table 3.3: Mesh statistics and quality metrics.

Statistics	
Cells	8,594,216
Nodes	5,812,172
Max. Size	0.24 mm
Min & Max y^+	0.18, 0.46
Orthogonal Quality	
Max	1.000
Min	0.181
Average	0.868
Skewness	
Max	0.824
Min	0
Average	0.139
Inflation layers	
Layers	20
Growth ratio	1.09

pressure field. Because the resulting face mass flux generally does not satisfy the continuity equation, a correction is added to the face. This substitution yields a discrete equation for the pressure correction in the cell, which is then solved, allowing the cell pressure and face flux to be updated and corrected. In lieu of solving the momentum equation and pressure correction equations separately, the Coupled algorithm solves the momentum and pressure-based continuity equations simultaneously. Full implicit coupling is achieved through an implicit discretization of the pressure gradient terms in the momentum equations and an implicit discretization of the face mass flux. Because it solves the equations in a closely coupled manner, it offers superior performance and a significantly improved rate of solution convergence for steady-state flows compared to the segregated solution schemes. Additionally, for transient flows, the Coupled algorithm is necessary when the mesh quality is poor or when large time steps are utilized.

Sinha et al. [20] performed transient simulations and reported a considerable computational time consumption for each case, even when the simulations were executed in high-performance computing systems. The exact replication of these simulations is not feasible due to hardware limitations; therefore, exploratory steady-state simulations were run to verify that the transient effect can be neglected. The SIMPLE algorithm was employed. Even though the convergence was achieved, the solver struggled to reach convergence with this algorithm. Instead, the coupled algorithm was used, which exhibited acceptable convergence behavior. As mentioned, these simulations were only executed as preliminary tests to become familiar with the effect that

these algorithms yield. The definitive simulations utilized a tailored configuration. Specifically, the solver was configured based on the recommendations given by Menter et al. [29] which are listed below,

- SST $k-\omega$ model.
- Coupled solver.
- Pseudo-transient with default proposed time step.
- Second order momentum.
- Second order turbulence.

It is mentioned that set second order turbulence is helpful for unstructured grids, in addition to this, it does not have an overall negative influence on convergence. These settings are recommended for steady-state simulations.

3.3.4 CFD validation results

The simulation results from each case presented in Table 3.2 were used to compute the values of Nu and f in order to compute the difference against the results of the benchmark. In [20] the value of f is computed through Eq. 1.15. For Nu, they considered the surface area of the fins exclusively (A_{fins}) to compute the value, but because the comparison is valid only by the average value over the whole domain, the value $\overline{Nu}$ is computed as follows,

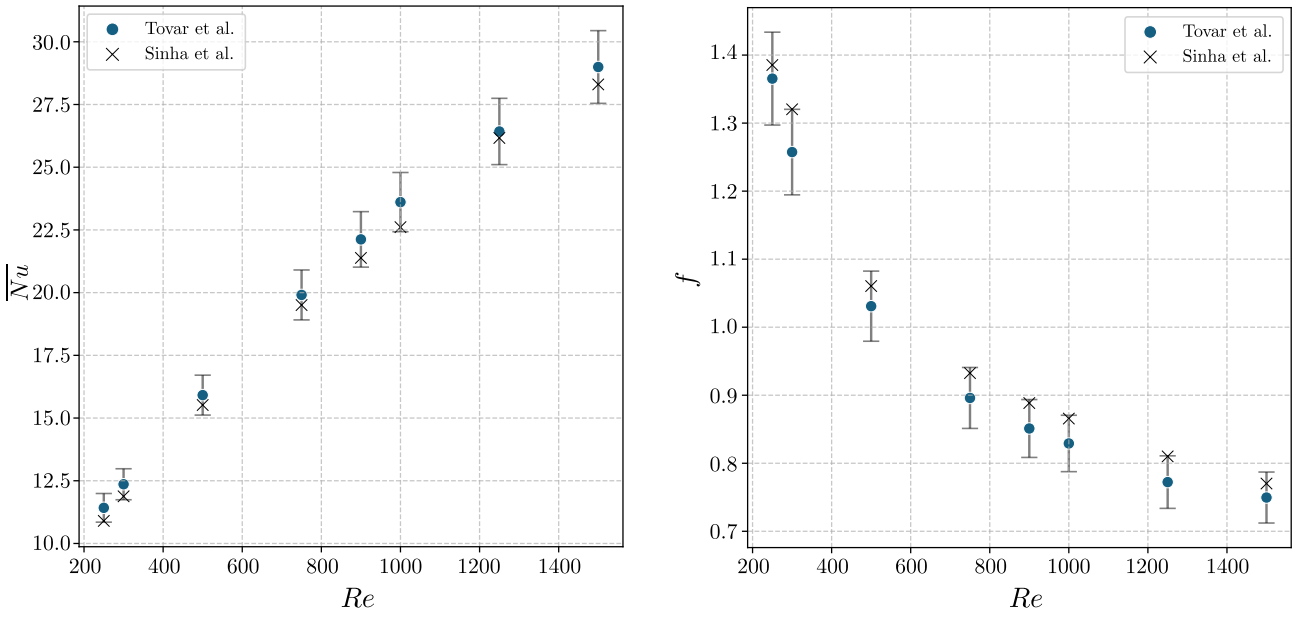
$$\overline{Nu} = \frac{2H \cdot q}{k_f \cdot LMTD \cdot A_{fins}} \tag{3.8}$$

where q is the heat added to the air (between the actual inlet and actual outlet), and it is worked out by Eq. 1.6b. For this case, $LMTD$ is calculated by the Eq. 3.9 through the wall temperature (T_w) and through T_i and T_o, the bulk temperature at the actual inlet and actual outlet of the air, respectively. The bulk temperature is computed in Fluent as Eq. 3.10 depicts, where $\vec{A}$ and $\vec{u}$ represent the cross-sectional area where the bulk temperature is evaluated and the corresponding flow velocity through that section, respectively.

$$LMTD = \frac{(T_w - T_o) - (T_w - T_i)}{\ln\left(\frac{T_w - T_o}{T_w - T_i}\right)} \tag{3.9}$$

$$T_{bulk} = \frac{\int_A T \left|\rho\vec{u} \cdot d\vec{A}\right|}{\int_A \left|\rho\vec{u} \cdot d\vec{A}\right|} \tag{3.10}$$

Fig. 3.5 presents the comparison of the $\overline{Nu}$ and f obtained during the validation process. The bars express an interval of ± 5% of difference. For the case of $\overline{Nu}$, the validation results were slightly higher than the benchmark data. Nevertheless, none of the cases exceeded the defined percentage variation interval. For the friction factor, the validation results were slightly lower, with deviations below 5%. Since some assumptions were made for the fluid properties of the air, and because the present simulations used a different solver configuration and steady-state conditions, deviations below 5% are considered acceptable for validation purposes.



(a) Simulation results: Comparison with data provided by [20] for average Nusselt number ($\overline{Nu}$).

(b) Simulation results: Comparison with data provided by [20] for friction factor (f).

Figure 3.5: Comparison of two results. Average relative error for $\overline{Nu}$ and f: 3.3% and 3.8%, respectively.

3.4 Simulations for the FTHE with bow-shaped VGs

After a successful validation, the simulations for the present project are carried out. For this scenario the points for sampling, obtained in the previous chapter, are considered to build the dataset. For this purpose, it is necessary to automate the whole process, from the CAD files

creation to the post-processing stage. The main reason for this is reducing the manual workload due to the amount of simulations to be executed. In addition, automatization diminishes the likelihood of human error. To accomplish this objective some options were considered. The unavailability of flexible and well documented automation process for Ansys Meshing ruled out its usage, thus other options were evaluated. Fluent, apart from the solver, includes a grid generator intended solely for CFD. Moreover, Ansys have developed a library for Python that allows the user write scripts to connect Fluent in a bidirectional communication way. This library is called *PyFluent* [30] which supports meshing/simulation setup, execution, monitoring, and results extraction.

Because the grid at this stage is automatically generated by Fluent, a specialized meshing methodology must be designed to accommodate the complex geometry of the FTHE with bow-shaped VGs.

3.4.1 Grid generation through Fluent Meshing

Even though the grid is generated through Fluent, the essence of the mesh that was created for the validation stage is reproduced here. It is required to construct the mesh considering that a high fidelity CFD simulations are sought but without requiring high computational effort. The FTHE presented in Fig 2.3 is modeled as a fluid domain where the inlet and outlet regions are extended to avoid numerical errors, just as done for the validation geometry (see §3.3.2).

In Fluent the workflow for the grid generation is called *Watertight Geometry*, where the meshing process is well structured by an operations tree. After importing the previously prepared CAD file, the first configuration is for *local sizing*; however, this step is bypassed because the grid is created only by global settings. The next step is to generate the surface mesh, where the minimum and maximum sizes for the cells are set. The minimum size is established at 5×10^{-5} m, while the maximum size is 3×10^{-4} m. This later parameter is available for surface and volume grid, which is essential for controlling the total cell count in the mesh. In this same segment, *size function* is set as *Curvature and Proximity* which means the surface mesh generation uses a function to refine the mesh based on the underlying curve and surface curvature, as well as the number of cells per gap specified. The curvature is modified by a specified angle that controls the maximum allowable angle that one element edge is allowed to span given a particular geometry curvature. For this case, the curvature angle is set at 10°. The cells per gap were set at 2 and the scope proximity is established for faces and edges. This configuration

allows the algorithm to refine when edges and/or faces are close enough to each other. The remaining parameters were left at their default values.

The next segment of the grid generation describes the geometry. For this project, the geometry represents a fluid domain only, and since the CAD file is built by different bodies, the share topology option is enabled. Subsequently, the boundaries were defined (inlet, outlet, wall, etc.) and for this case, the lateral edges are set as *symmetry* boundary condition. The next step is to set up the boundary layers. As the validation grid, 20 inflation layers on the walls were established, but with a growth ration of 1.1 with *smooth transition* option. Finally, for the 3D mesh, the minimum and maximum sizes for the internal cells are set to the same values used for the surface mesh.

Here it is important to highlight that the mesh is constructed using the *poly-hexcore* method which combines the ideas presented by Fig 1.10 and 1.11. Polyhedrons are used for 'complex' regions in the domain, such as around the VGs and tubes, then a transition occurs to change from polyhedrons to hexahedrons with a non-conformal construction in the core of the domain. This is an option unavailable in Ansys Meshing, thus besides of the automation capability, this method is another advantage of using Fluent.

For this work the acceptable mesh quality is when an orthogonal quality is between 0.15 and 0.20, but a minor issue was detected. The sharp end of the bow-shaped VG represents a difficult region to mesh, thus the solution to this geometric constraint was to truncate the tip to get a blunt end. In order to ensure this blunt end does not affect the results significantly; the end was truncated as Fig 3.6 depicts. The angle shown in the drawing corresponds to the internal circumference of the VGs.

A preliminary simulation was run utilizing the same setup as the validation stage with the difference that, for the subsequent simulations, the inlet condition for the flow is a uniform velocity profile in direction to the x_1-axis ($u_1 = U$). With this setup, the simulation time for a couple of test runs was around of 1 to 1.5 h. Since the extensions for the inlet and outlet regions add a considerable amount of cells to the domain, instead of following the thumb rule for these regions, the length of the inlet extension was set at $5H$, while for the outlet extension was set at $12H$ shown in Fig. 3.7.

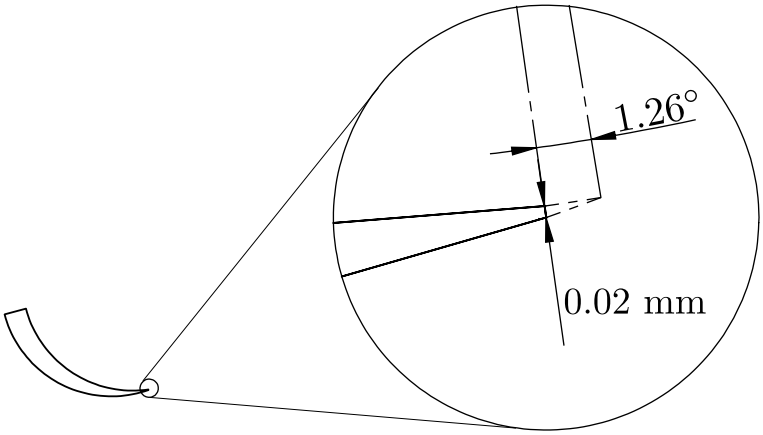


Figure 3.6: Blunt end for the bow-shaped VGs

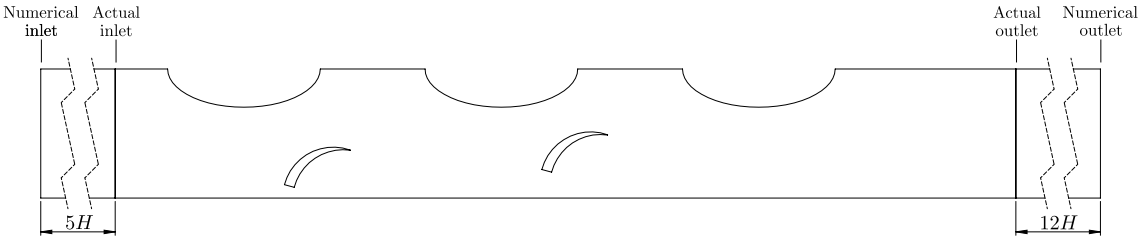


Figure 3.7: Dimensions for the extensions in the halved FTHE with bow-shaped VGs.

To confirm that these lengths for the extensions are valid for the current project, the results obtained from the preliminary simulation were used as reference, as the reduction of the extensions takes place in further simulations. The criterion is to obtain results as close as possible to the reference. With a difference of less than 0.5% the design is established as above mentioned. In addition, as shown in the same figure, the domain was halved. This action was considered because the flow behavior is symmetrical with respect to the line connecting the foci of the ellipses. With these adjustments the number of cells for each case is reduced by keeping the quality of the data obtained after each simulation. The next step is to carry out a mesh independency study.

3.4.2 Grid independence study

For the validation process the mesh independency study was conducted by evaluating the percentage difference in results as the mesh density was progressively increased. This is carried out by a technique called *Grid Convergence Index* (GCI). The GCI is an objective metric based on the generalized theory of *Richardson Extrapolation* that is used to estimate and quantify numerical uncertainty in CFD and related computational disciplines. The primary purpose of the

GCI is to provide a standardized, uniform method for reporting the results of grid refinement studies [31]. The GCI was designed to resolve this issue, providing a straightforward, well-justified method for mesh independency studies. After this philosophy was presented, Celik et al. [32] provide a highly structured, five-steps recommended procedure for estimating and reporting discretization error, explicitly designed to serve as a uniform guideline for publications.

Before applying GCI, Celik et al. [32] mandate a preliminary check: it must be ensured that iterative convergence has been achieved, typically indicated by at least a three (preferably four) orders of magnitude decrease residuals for each equation solved. Three simulations were performed and the convergence criterion for them (and for all subsequent simulations) was to set the residuals of the continuity equation to 1×10^{-6}. When the residuals of the continuity equation meet this criterion, the residuals of the remaining equations also meet this requirement.

The difference between these three simulations is the number of cells. The details are shown in Table 3.4. The case of the Mesh 1 is the finest, while Mesh 3 is the coarsest. The refinement followed the rule proposed by the method: the cube root of the ratio between a finer mesh and its coarser predecessor must be greater than 1.3. This ratio is called *grid refinement factor* (r). Note that the number of cells for this study is significantly lower than the number of cells used in the validation stage. This is a consequence of the simplification of the 3D domain related to its symmetric flow behavior.

Table 3.4: Data for grid independence study.

Mesh	**No. Cells**	$\overline{Nu}$	f
1	3,811,327	9.073044	0.544440
2	1,698,797	9.059260	0.545995
3	836,970	9.025515	0.547972

Two grid refinement ratio are calculated; the first one compares Mesh 1 with Mesh 2 and the second one compares Mesh 2 with 3. The values of $\overline{Nu}$ and f serve as control values for the CGI; if these values do not change significantly with refinement, the physics convergence is said to be inside an asymptotic region. The next step is to calculate a parameter called *apparent order* (p), that is the actual, empirically observed rate at which the numerical error decreases as the computational grid is refined for a specific calculation. To ensure that the refinement is enough for $\overline{Nu}$ and f, two GCI procedures are carried out. Once the apparent order is

computed, extrapolated values are calculated. These values are used to know the numerical uncertainty. At this point the last step is to compare two GCI index: coarse-to-medium GCI (GCI_{23}) against medium-to-fine GCI (GCI_{12}). Then to make sure thar the asymptotic region is reached, the following expression must be satisfied,

$$GCI_{23} \approx r^p GCI_{12}$$

If this approximate equality holds true, it indicates that the error is decreasing at the expected theoretical rate and that the grids are within the asymptotic range of convergence.

The method proposed by Celik et al. [32] is replicated through a Python script. With the data shown in Table 3.4, the results obtained after following the GCI method confirm a convergence as the refinement takes place. These results are presented in Table 3.5 where it is observed that the convergence occurs within an asymptotic region as the theory indicates.

Table 3.5: Summary of GCI analysis and asymptotic range verification.

Criteria	Asymptotic Check	Fine Grid GCI	Medium Grid GCI
$\overline{Nu}$	1.0015	0.09%	0.28%
f	0.9972	0.73%	1.08%

This table shows the certainty percentage of the convergence. The GCI cannot provide a strict, mathematically guaranteed maximum bound on the error, instead, it acts more like a statistical confidence interval.

Since the values of these reported GCI are low, the refinement of the Mesh 2 can be adopted to keep a reduced number of cells; however, to retain the uncertainty under 1% for f, a mesh with refinement controls that yield results between Mesh 1 and 2 will be tailored for the remaining cases. Thus, because the independency study is successfully concluded with the proposed refinement, the mesh for the future simulations is generated with over 2 million cells with an average orthogonal quality of 0.17. In addition, it is estimated that, on average, the value of y^+ is around 0.28 for all cases.

3.4.3 Automation of batch simulations

The automation for the sequence of each simulation is done by a script in Python. This script connects to Fluent by the library previously introduced: PyFluent, which is used for the CFD process only, thus the automation for the creation of the CAD file for each case is out of scope of this Python script. The creation of the 3D files for each case is also considered as part of the CFD process, but for this project generating these files is not part of the sequential workflow for individual simulations. Instead, the CAD files are created in batches to ensure they are fully prepared before the simulation automation script is executed. This methodology is explored in greater detail below.

3.4.3.1 Preparation of 3D models

The list of points created in Section 2.2.2 was used in this stage. Such list was previously prepared in a spreadsheet by Microsoft Excel® (hereinafter referred to as Excel), which is an essential step for exporting the coordinates of each point into SW. With a parametrized template created in SW, each CAD file is generated by extracting the information of each point from Excel by *Visual Basic for Applications* (VBA). The file is then saved with a characteristic name that serves to identify the specific case.

At this point a minor geometric constraint is found. For each meshing process all the faces and bodies that define the 3D domain must be labeled so that the solver can identify the regions in the domain and then apply the boundary conditions correctly, which also facilitates the creation of the automation script. In Ansys, these labels are called *named selections* and cannot be created by SW. To overcome this drawback, *SpaceClaim* is employed. This tool is a CAD modeler developed by Ansys that, unlike SW, is a direct-modeling tool that is used primarily to quickly edit, repair, or remove features of existing 3D models to prepare them for structural or fluid analysis. To complete de CAD processing, another script is built to add the labels to the 3D domains (previously created by SW) by SpaceClaim, but this time through *IronPython* which is used as the native scripting environment inside SpaceClaim. Through this software new CAD files were saved, which are fully compatible with Fluent because they contain all the required information to mesh the three-dimensional domain neatly.

3.4.3.2 Simulations though batch processing

As above mentioned, the simulations were run as an automated batch process, evaluating each configuration one after the other. Once the 3D files are correctly prepared, the meshing methodology is captured by logging the user interactions within the *graphical user interface* (GUI) of the meshing software. This process writes the actions onto a sequence of commands that represent the interaction between the user and the software. Because all the CAD files are constructed and labeled in the same way, the commands obtained work for all the cases. When the grid is constructed, the file is saved with its particular name and then read it in the solver environment.

PyFluent allows changing from the meshing to the solver environment, where the governing equations are solved. In the solver, in lieu of using the GUI, the actions applied by the user through the *text user interface* (TUI). This is an advantage because the syntax for the TUI is quite similar to the one used in PyFluent. These settings correspond to the boundary conditions, material properties, turbulence model, convergence criteria, monitoring, post processing data. These actions are manually executed once in order to identify the specific commands required to create the tailored script, which is subsequently used to control, simulate, and analyze all the samples.

For the post processing step of each case, leveraging the scripting capabilities, one hundred evenly spaced planes perpendicular to the x_1-axis (x_2-x_3 planes) have been created to serve as a reference for any specific data extraction that may be useful during the analysis stage.

3.4.3.3 Results

After extracting the sequence of commands, finally a tailored script is built to be used for all the required sampling points. When each simulation is completed, specific data is extracted by the script. The main extracted data corresponds to the temperature and pressure at the actual inlet and outlet. This information is used to compute the values of f and $\overline{Nu}$. For these simulations, the value of $\overline{Nu}$ is computed using Eq. 3.11, where A_s corresponds to the total area where the heat transfer takes place (fins, tubes and VGs).

$$\overline{Nu} = \frac{2H \cdot q}{k_f \cdot LMTD \cdot A_s} \tag{3.11}$$

Across the defined spanwise planes, the bulk temperature is computed to analyze its evolution throughout the domain. In addition, the quantification of the intensity of secondary flows generated by VGs is strongly linked to the overall heat transfer. Chang et al. [33] proposed a mathematical definition to compute the intensity of such secondary flows. Instead of computing the *spanwise average vorticity flux*, which leads to a value equal to zero due to the solid boundaries and the existence of multiple swirl motions rotating in opposite directions, they proposed computing the *absolute vorticity flux* ($\omega_{x_1,abs}$) defined by Eq. 3.12. This parameter has been identified to be used to quantify the intensity of the secondary flows in the spanwise orientation, in addition to having a direct relationship with the local Nu value along the streamwise direction. However, $\omega_{x_1,abs}$ is specifically designed to measure the intensity of streamwise swirling, the secondary flow created by the VGs. At the very beginning of the channel, the flow is mostly straight, and the VGs have not yet fully tripped the fluid into robust vortices.

$$\omega_{x_1,abs} = \frac{1}{2}\frac{1}{A_{2,3}}\int_{A_{2,3}}\left|\frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3}\right| dx_2 dx_3 \tag{3.12}$$

Expanding the relationship of the secondary flows with the heat transfer, [33] demonstrated that the overall heat transfer intensity represented by $\overline{Nu}$ changes proportionately with a parameter called *streamwise averaged absolute vorticity flux* (Ω) as Re at the entrance of the channel is modified. Therefore, Ω can be linked to the tendency of $\overline{Nu}$ as the geometrical arrangement of the VGs is changed. Over each plane, the average of the absolute vorticity flux is computed through commands in Fluent (area weighted-average). Based on this data, Eq. 3.13 is used to compute the cross-averaged absolute vorticity flux that defines an average vorticity over the domain of interest.

$$\Omega = \frac{1}{L}\int\left(\frac{1}{2}\frac{1}{A_{2,3}}\int_{A_{2,3}}\left|\frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3}\right| dx_2 dx_3\right) dx_1 \tag{3.13}$$

The data extracted after each simulation is built up in a spreadsheet where additional information is computed. This spreadsheet gives the required information to create the preliminary dataset to train ML models.

Chapter summary

The methodology developed for the simulations carried out for the present project was successfully validated by comparing the results with a benchmark study widely cited in the literature. After this step, the 3D files are generated using a custom VBA script that connects the data of each sampling point to a parametrized CAD template modeled in SW. To complete the preparation of each three-dimensional model, SpaceClaim is automated with another script in the IronPython environment. Finally, after some adjustments and by using Fluent for meshing, PyFluent is utilized to automate all the remaining process where even the post processing steps are programmed.

CHAPTER 4

APPLICATION OF MACHINE LEARNING FOR PREDICTING HEAT EXCHANGERS PERFORMACE INDICATORS

As previously established, the purpose of applying machine learning (ML) is to accelerate the study of the FTHE with the bow-shaped VGs through reliable predictions that replace the CFD simulations. In particular, Nusselt number ($\overline{Nu}$) and the friction factor (f) are the primary parameters to be predicted by a ML model. Additionally, it is explored how additional parameters can augment the accuracy of the predictions.

With the data obtained in the previous chapter, a preliminary dataset is built to be used for training ML models based on three different algorithms for regression. One of these algorithms is the Gaussian Process Regressor, fundamentally used for the active learning step. The three algorithms are configured to obtain the best predictions that they can yield. Based on their metrics; a single ML model is used for the remaining process of the current project.

The dataset is structured as a table containing both features and targets. The features are shown in Listing 2.1, while the targets ($\overline{Nu}$, f and η) are added to their corresponding row in each case. $\overline{Nu}$ is computed via Eq. 3.11, while f is obtained through Eq. 1.15 where Δp is the differential pressure between the outlet and the inlet. For the case of η, the Eq. 1.16 is applied, utilizing Nu_b and f_b, obtained based on the data yielded by the simulation of the FTHE without VGs (baseline FTHE). The values of η are added to explore the possibility of predicting it in lieu of $\overline{Nu}$ and f.

All the scripts created for ML purposes use a fundamental library called *scikit-learn* [34], which is designed to apply ML in Python. With this library, the dataset is handled in any convenient way for the algorithms processing. Furthermore, scikit-learn integrates the GPR algorithm and provides different manners to compute the metrics of each model. Further libraries are introduced later on in this chapter, which are required for the remaining ML algorithms and for advanced data processing.

4.1 Machine learning algorithms for regression

As aforementioned, three algorithms are adopted to be used for regression. GPR, *artificial neural networks* (ANNs), and *Extreme Gradient Boosting* (XGBoost). Following, a review of the characteristics of each of these algorithms is provided.

4.1.1 Gaussian Process Regressor

GPR is a non-parametric Bayesian approach that places a prior distribution directly over an infinite space of functions, rather than restricting the model to a specific parametric form like linear regression. The model treats the unknown input-output relationship as a realization of a Gaussian stochastic process, which is completely specified by a mean function and a *covariance function*, or *kernel* [35]. In GPR, the kernel serves as the primary mathematical mechanism for quantifying the similarity between distinct data points within the input space. It explicitly encodes prior assumptions regarding the behavior of the underlying system, dictating critical characteristics such as the smoothness, periodicity, and variance of the predictive model. By evaluating the spatial distance and correlation between known training samples and unseen target coordinates, the kernel enables the algorithm to construct a precise interpolation matrix. Consequently, it operates on the foundational probabilistic principle that inputs located closely together in the parameter space will naturally exhibit highly correlated and predictable output values.

The exact behavior of the covariance function is controlled by its hyperparameters. The most critical of these is the *characteristic length-scale*, which determines how rapidly the correlation between points decays as the distance between their inputs increases. Another common hyperparameter is the *signal variance*, which controls the overall amplitude or magnitude of the function. Additionally, a *noise variance* parameter is frequently included to handle noisy observations or simply to ensure numerical stability during computation [25]. Different kernels are available to be employed for GPR [34], but two of them were found useful for the development

of this thesis. Regardless of the adopted kernel, GPR is capable of computing the uncertainty of the prediction of any given point. This characteristic is beneficial when additional points are required to enhance the predictions accuracy.

4.1.1.1 Radial basis function and rational quadratic kernels

The *radial basis function* (RBF), also known as the *squared exponential* kernel, is used as default when GPR is employed. For the purpose of the present work, this kernel is an adequate option because it is infinitely differentiable, meaning it mathematically assumes the underlying physics are infinitely smooth and continuous, which is a very safe guess for many natural phenomena. It calculates correlation solely based on the radial geometric distance between two input points, making it mechanically stable to guarantee a valid, positive-definite covariance matrix regardless of how many dimensions are introduced. It works as an exponential function that converts the distance of the points into a *similarity score*, which means that when the distance is 0, the result is the unit (100% similar points). As the distance grows, the score exponentially decays towards 0 [35].

The fundamental limitation of the RBF kernel is that it only has one length scale; in consequence, it forces assuming that the physics operates at exactly one specific magnitude (i.e. speed, frequency). The *rational quadratic* (RQ) kernel was invented to solve this. Mathematically, the RQ equation is actually the equivalent of taking an infinite number of RBF kernels, all with different length scales, and adding them together. Because it blends these different scales together, it is exceptionally good at modeling data that exhibits dynamics across multiple length-scales simultaneously. Similar to the RBF kernel, it is infinitely differentiable and assumes the underlying physical process is extremely smooth, and as its shape parameter approaches infinity, the QR kernel perfectly mimics the RBF kernel.

4.1.1.2 Predictive variance

In GPR, the *posterior variance* at a given test point, predictive variance or uncertainty, is computed by subtracting the information acquired from the known data from the baseline assumption made by the model. Mathematically, this is achieved by evaluating the kernel function at the test coordinate to establish the prior variance and subsequently subtracting a reduction term defined by the product of the cross-covariance vector and the inverse of the training covariance matrix. Crucially, this uncertainty calculation is entirely independent of the observed target values, relying exclusively on the spatial geometry of the input features. Consequently,

the algorithm provides a rigorously quantified measure of confidence. This advantage permits establishing regions within the 5-dimensional space where points are found with high relative uncertainty for the model.

4.1.2 Artificial Neural Networks

ANNs construct ML models inspired by the biological structure of the human brain. The architecture is defined by layers: input layer, hidden layers, and output layer. Each layer contains a set of nodes (the neurons) that are interconnected by branches to the next (or previous) layers, forming an *acyclic graph*. Each neuron receives input numbers from the previous layer, and subsequently, it multiplies those numbers by their specific connection weights (contained by the branches), adds them altogether into a single sum, and adds a small constant called a *bias*. It passes that final sum through an *activation function*, a mathematical rule that decides what the final output number should be before passing it forward to the next layer [36]. If an ANN only has one or two hidden layers, it is considered a *shallow* network, but if the ANN has three or more hidden layers, it officially crosses the threshold into *deep learning*. When a network is "deep", it does not try to solve the entire problem at once. Instead, each successive layer learns to recognize increasingly complex and abstract features based on what the previous layer figured out.

The training of the model requires a decision maker defined as the activation function. This function introduces non-linearity into the network to decide whether the neuron is turned on or remains 'silent'. Without activation functions, an ANN would mathematically collapse into a single, basic linear regression model. Activation functions (like ReLU, which turns all negative numbers to zero, or Sigmoid, which squishes numbers between 0 and 1) allow the network to draw complex, curved, non-linear boundaries around data. Activation functions are a scalar-to-scalar functions for the hidden neurons, and the selection of an activation function depends on the purpose of the ANN and the available data.

As the information is conveyed forward through the network, this information is "learned" by using an algorithm called *backpropagation* to adjust the weights. Backpropagation is the messenger that calculates the error, but the *optimization algorithm* is the engine that actually decides how to fix the weights. For the case of the activation functions, an optimization algorithm must be selected for specific scenarios. The way that the optimization is carried out is by a *loss function* that is used to quantify the accuracy of the predictions compared to actual

data. The most common loss functions for regression are mean squared error (MSE), mean absolute error (MAE), mean squared log error (MSLE), and mean absolute percentage error (MAPE) [36].

The development of ANN for this thesis is through a library for Python called *PyTorch* [37]. This library is widely used open-source deep learning framework that leverages *graphics processing unit* (GPU) for tensor computations and dynamic computational graphs. The principal hyperparameters controlled are the number of layers and their number of neurons. Different activation functions are tested, as well as the optimizer and a parameter called *learning rate* (*alpha*) that controls the internal parameter updates during the optimization. The learning rate dictates how aggressively the network changes its internal weights to fix the error computed by the loss function.

4.1.3 Extreme Gradient Boosting

XGBoost is the result of an evolutionary development of the *decision tree* technique that was created to fix some flaws that can be encountered in its predecessor algorithms.

Decision trees are based on if-then-else decision rules, which limit the capabilities of the algorithm to learn complex patterns. A single decision tree is notorious for having extremely high variance, but if the tree is created with a considerable amount of branches, it will perfectly memorize the training data, capturing every piece of noise and failing on new, unseen data. To control overfitting, a technique called *random forest* is developed to use several decision trees in parallel and fit them on various sub-samples of the dataset. The principal advantage of random forests is that they do not overfit because they average the predictions of each tree, but if the data is highly complex, subtle patterns cannot be learned even if more trees are added because the trees are built completely independently; thus, they cannot learn from the mistakes of the other trees. To capture complex information, a sequential philosophy has been developed and named *gradient boosting* that creates models based on shallow trees. Gradient boosting builds trees one at a time. Tree 1 makes a crude prediction. Tree 2 looks exclusively at what Tree 1 mispredicted (the residuals) and tries to fix it. Tree 3 looks at what Tree 2 failed to predict, and so on. To make a final prediction, the algorithm sums the answers of all the trees together. Unfortunately, this technique is prone to overfitting the data because each new tree is aggressively trying to fix the residual errors of the last one. To fix the overfitting in the gradient boosting, *lasso* and *ridge regularization* are considered to build the XGBoost

algorithm.

In XGBoost, the final prediction of a leaf node is called its weight, and this weight is optimized through an objective function inside the algorithm of the XGBoost. Regularization is a technique that is utilized to modify the way that the algorithm learns the patterns of the given data. This technique is used to avoid overfitting, especially when the information in the dataset has a certain level of noise. Regularization forces the algorithm to build safe, generalizable rules. These controls are separated into two different approaches. Lasso regularization is used to ignore variables that are not significantly correlated with the target and then simplifies the model. It acts like a feature selector. It mathematically forces the importance of ineffectual or redundant variables to exactly zero. If two variables are highly correlated, lasso will pick the best one and completely delete the other. On the other hand, ridge regularization acts like a shock absorber. It does not delete any variables, but it heavily penalizes extreme predictions. It forces the model to share the predictive weight across many different variables rather than trusting one single erratic parameter. This results in a highly stable, smooth model that does not jump violently when the input data changes slightly. For further information regarding the regularization, it is recommended to refer to the work of Müller and Guido [17]. Among other characteristics, XGBoost uses a second-order Taylor approximation; thus, this algorithm can take faster and more accurate steps toward the optimal solution.

A library developed for Python called *XGBoost* [38] is adopted to build the codes that leverage this algorithm. The most important hyperparameters utilized are the number of trees (*No. of estimators*), a parameter called *learning rate* that, in this context, shrinks the feature weights at each boosting step to prevent overfitting. It determines how aggressively the model corrects errors made by previous trees. Another parameter is related to the maximum number of levels a single tree is allowed to grow (*max depth*). The default value for this parameter is 6, and higher values capture extremely specific nuances but drastically increase the risk of the model memorizing pure noise. Finally, the ridge and lasso regularization (represented by *lambda* and *alpha*, respectively, in the Python code) are modified for more accurate predictions.

4.2 Code development and testing

For training ML models, the dataset with 256 data points is split into two subsets: training and test. This action is performed by the function `train_test_split()`, a built-in function of scikit-learn, where the test size is set to be the 15% of the whole dataset. That is to say, 218

points are used to train the model, and 38 points are used to test the model. This split is done randomly, but for more control an argument of the function called `random_state` is used to set a seed[1], thus every time the code is run, the dataset will be split in the same way.

For GPR and ANN the data must be pre-processed to be correctly utilized by the algorithms. The data is transformed into a scaled version of itself by standardization. This standardization sets the values to have the mean at zero with the unit as variance. Through this action the data is scaled to be the same order of magnitude, thus the weights can be assigned perfectly to each case with low bias. For decision trees-based algorithms, this action is not required. The construction of the codes for each case is explained below,

- **GPR:** With an RBG kernel, the length scale is set to be within a bounds from 0.01 to 100 with a starting point at 1.0 (the optimizer of the algorithm finds the best one). The noise parameter is set at 0.03. This means that the dataset is accepted to have ± 0.03 of variance of the standardized data (percentage of the total dataset variance). This later parameter was found adequate after a sequence of grid search.

- **ANN:** In the case of the ANN, a multi-layer perceptron for regression was built with two hidden layers. The first hidden layer is built with 40 neurons, while for the second one 20 neurons are established. The adequate activation function for this case is `tanh` with L-BFGS[2] as optimization algorithm for a learning rate of 0.01. Five neurons are set for the input (because of the number of features) and only one neuron at the output (for $\overline{Nu}$, f or η).

- **XGBoost:** Even though this algorithm does not require normalized or standardized dataset, to keep a consistency with the other algorithms, the dataset is also standardized for this case. The model is constructed though 200 estimators with a maximum depth of 3. The learning rate for this context is 0.3 with ridge regulation of 1.5 and 0.1 for lasso.

To compare the accuracy of the predictions made by each model, *coefficient of determination* (R^2) and RMSE scores are computed. R^2 indicates the amount of variation in the predictions by a regression model, explained by the actual values. This score is computed by Eq. 4.1 where y_i represents an actual value, $\overline{y}$ is the average of the actual values recorded, and $\widehat{y_i}$ corresponds

[1] In programming, a seed is an initial starting value used to generate a sequence of numbers or outcomes. It is most commonly used in random number generators to ensure that "random" sequences can be controlled, repeated, or tested.

[2] A second-order optimization algorithm that approximates curvature information for precision (see [36]).

to the value predicted by the model. The RMSE represents an intuitive quantity of the average error of the predictions. This value is represented by Eq. 4.2

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2} \tag{4.1}$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \widehat{y}_i)^2} \tag{4.2}$$

The mentioned values of the hyperparameters for each algorithm are found adequate after adjusting such values to find the best results in the predictions. In general, the models based on GPR and ANN yielded the best results, while XGBoost does not predict correctly the majority of the test points, even for different seeds. None of the models predict correctly η, but this is not a problem due to the fact this parameter can be computed by $\overline{Nu}$ and f. At this stage the parameter f is readily predicted, but for the case of $\overline{Nu}$, the conclusion is to run more simulations to enhance the accuracy, and automatically to also improve the predictions for f.

4.2.1 Modifications for better predictions

In the previous chapter, it was reviewed that the streamwise averaged absolute vorticity flux can be harnessed because it was found that this parameter is correlated with the heat transfer augmentation when enhancement techniques are employed. Thus, adding this value to each case to bolster the predictions of $\overline{Nu}$ is a logical conclusion. The value that is added to each row of the dataset is computed by Eq. 3.13. Now, adjusting the input values in the respective codes, it was found that Ω is actually useful to improve $\overline{Nu}$. However, Ω is also an unknown; thus, this parameter must be predicted as well.

To predict Ω the same techniques used for $\overline{Nu}$ and f were employed. Again, the worst predictions are made by the model based on XGBoost, while the level of accuracy made by the models based on GPR and ANN is similar, with a tendency to improve as more data points are included. Based on these observations, only GPR and ANN are considered from now on.

Because, in the case of GPR, the model can only predict one output at a time. Thus, the idea

of predicting $\overline{Nu}$ by adding Ω is to primarily predict this latter parameter and then, with a different model, predict the former one. While for the model based on ANN, the number of outputs is more flexible due to the structure of the model. The technique used to handle two or more outputs is called *multi-task learning*, and the structure of this technique with ANN is called *multi-task neural network* (MTNN).

In the case of Ω, it has been found that its predictions are improved if f is used as part of the inputs. Hence, three individual models are created using GPR and a unique model with ANN. For the model based on GPR, a couple of lines in the code are modified to allow more or fewer inputs; consequently, the main structure remains the same. For the case of MTNN, Fig. 4.1 depicts the overall idea of this model. The inputs are kept at 5 (the original features), but the network is arranged to have three hidden layers with 64 neurons in the first one and 32 in the second. These first layers are shared by all the parameters, but a third layer is created for each case. 32 neurons are assigned to the last hidden layer of each parameter.

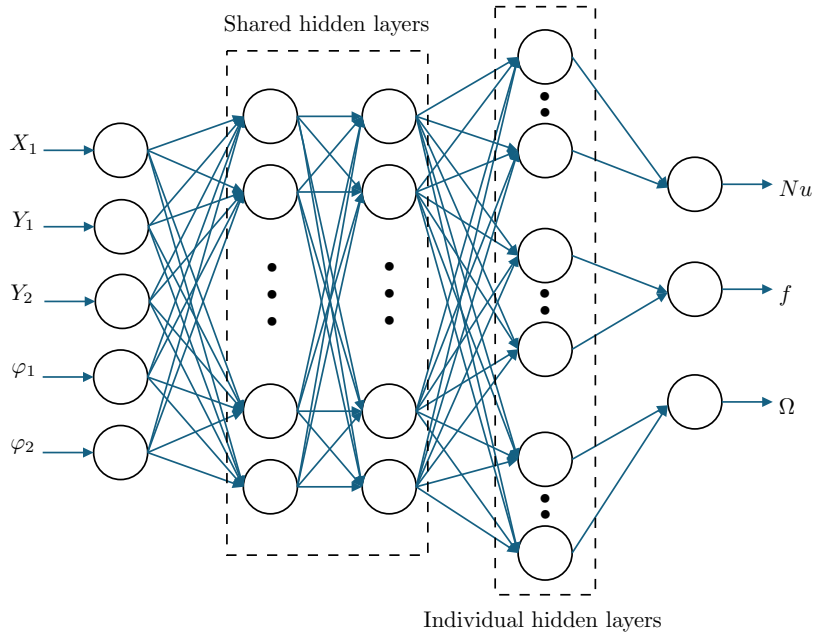


Figure 4.1: Graphical concept of the MTNN model.

This model now uses the residuals of the three target values to adjust the weights, but now the model handles three outputs with different order of magnitude between them. The order of magnitude of f is the lowest at 0.1, thus the values of $\overline{Nu}$ and Ω are adjusted by normalization to have the same order of magnitude. For the case of $\overline{Nu}$ the normalization is performed by Eq. 4.3, while for Ω (the value with the greatest order of magnitude), Eq. 4.4 is used where

the value of 250 is used to ensure the values of Ω^* do not exceed the unit.

$$\overline{Nu}^* = \frac{\overline{Nu} - 10}{\overline{Nu}_b - 10} \tag{4.3}$$

$$\Omega^* = \frac{\Omega - 250}{100 - 250} \tag{4.4}$$

4.2.2 Sequential design

For this part, GPR is the preferred model due to the capability to yield the uncertainty of the unseen points that are given to predict their respective targets. In this stage, active learning replaces the sequence generated by the Sobol' method. The active learning is defined by a creation of a huge amount of points that are sequentially tested in the GPR model. With the available model trained with the 256 first samples, an uncertainty is computed for each evaluated point. To create these test points, two approaches were selected: random generation and through the Sobol' method. When these techniques are employed, the new points must be carefully selected because similar points could be generated. To only consider new points that map unseen regions of the domain, an algorithm was created to filter points that are close to the already processed points. When these points are given to the model to compute the uncertainty, the values are sorted from the largest to the smallest, and only the largest ones are considered to be part of the dataset.

The generation of new points is separated by batches that contain ten sampling points. Generation by the random and by the Sobol' methods are alternate to create the unseen points with high uncertainty computed by the current GPR model. In general the batches are built by five points that will be used to improve the accuracy of the predictions for $\overline{Nu}$, three for Ω and two for f; however, this idea was not always met because sometimes the selected points turned out to be shared between parameters, thus additional different points were added to complete a sampling of ten points. Each batch of points is added to the list of new sampling points and then used to run the CFD process. The results yielded by the simulations were compared against the preliminary predictions to track the evolution in the capabilities of the model, then were added to their corresponding rows in the list that forms the dataset. Here is the beginning of an iterative process. After each batch of simulations were completed and the results are added to the dataset, the newest dataset is used to re-train the GPR model. With the newest model, another batch of new points is generated and tested to capture new unseen points, and so forth.

This procedure was repeated several times until 244 samples were collected. Twenty points from such set of samples are out of the established bounds for the parameters because high uncertainties frequently appeared at the maximum and minimum values of each parameter, thus in order to reduce the uncertainties, those twenty points contributed to move from extrapolation to interpolation. At this point, the dataset had an aggregate of 500 points, and the dataset was considered ready to train GPR and MTNN models and validate them through a technique called *cross-validation*.

4.2.3 Cross-validation

The principal objective of cross-validation (CV) is to ensure that the models do not overfit to the data under the established settings. To accomplish this, *five-fold cross-validation* is employed for both selected algorithms, where the idea of 'five-fold' stands for the separation of the dataset into five subsets. Before this separation, the whole dataset is split, immediately it is loaded, into 10% for test and 90% for the five-fold exercise. When the developed code for the CV is executed, five training process occur iteratively excluding one of the five subsets each time to evaluate the model performance. For each iteration, the metrics are computed and printed, thus the models can be compared to each other. At the end of this exercise, the model now is trained with all the five subsets and tested with the 10% of the whole dataset that was separated at the beginning of the data processing.

The results of the five-fold CV are averaged and showed in Table 4.1 for the case of the ANN model, and for the GPR model, the results are displayed in Table 4.2. Alongside each CV results, the test metrics are presented with an additional column that contains the average value of the *absolute percentage difference* between the actual and predicted values of each parameter. The ANN model is slightly better than the GPR one for $\overline{Nu}$ and Ω, while for f the GPR model is better. The results of this latter model for each parameter are obtained only considering the five geometric parameters as features. However, when the actual values of Ω are included as additional inputs for predicting Nu, the average prediction error decreases to 0.37

4.3 Relationship between features and targets

The ML model created using the first 256 data points was evaluated to determine the importance of each feature in influencing the predictions. It was found that, the values of φ_1 and φ_2

Table 4.1: Performance metrics for the 5-fold cross-validation and test datasets through ANN model.

Parameter	5-fold CV		Test		
	R^2	RMSE	R^2	RMSE	abs. perc. diff
$\overline{Nu}$	0.8653	0.0782	0.9396	0.0220	0.562%
f	0.9829	0.0100	0.9902	0.0039	0.568%
Ω	0.9635	3.5251	0.9801	2.5115	1.168%

Table 4.2: Performance metrics for the 5-fold cross-validation and test datasets through GPR model.

Parameter	5-fold CV		Test		
	R^2	RMSE	R^2	RMSE	abs. perc. diff
$\overline{Nu}$	0.8362	0.3227	0.9141	0.0750	0.613%
f	0.9844	0.0691	0.9939	0.0031	0.465%
Ω	0.9200	2.3540	0.9567	3.8568	1.651%

are highly important for both $\overline{Nu}$ and f. The remaining input variables exhibit different levels of importance depending on each target. Some parameters have a significant influence on $\overline{Nu}$, while their effect on f is negligible.

To interpret the predictive model, feature importance is calculated using a technique obtained from cooperative game theory. The technique calculates coefficients referred to as *Shapley values*, which provide a method for fairly distributing the total compensation of a game among its players based on their individual contributions. For the current context, the features are considered to be the players, and their influence in the predictions is analogous to the contributions; thus, the compensation for each feature is computed based on how the input values affect the outputs. Rather than evaluating features in isolation, the Shapley value calculates the marginal contribution of a single feature across every possible permutation of the other variables. This guarantees a mathematically fair attribution that isolates exactly how much each parameter changed the specific prediction away from the average base value based on the values presented by the dataset, capturing all complex feature interactions. The computation of these parameters is executed using the library *SHAP* available for Python [39].

To know the importance of each parameter, the library calculates the mean absolute value of the Shapley values for each feature across the entire evaluated dataset. The scoring is sorted from top to bottom in descending order based on this calculated mean absolute value. That is to say, the parameter with the longest bar is placed at the very top of the vertical axis. The importance of each feature for $\overline{Nu}$ is depicted in Fig. 4.2a, while for f, the information

is presented in Fig. 4.2b. The values on the horizontal axis represent how many each feature alters the output by an absolute magnitude of x units away from the baseline mean (on average across the entire dataset). These values are measured in the exact same units as the output variable, which, for f and $\overline{Nu}$, are dimensionless. In conclusion, these bars depicted in Fig. 4.2 represent the expected influence that each parameter provided to the final value of the predictions. These bars are plotted using the GPR model, but if the MTNN model is used, there are no significant differences.

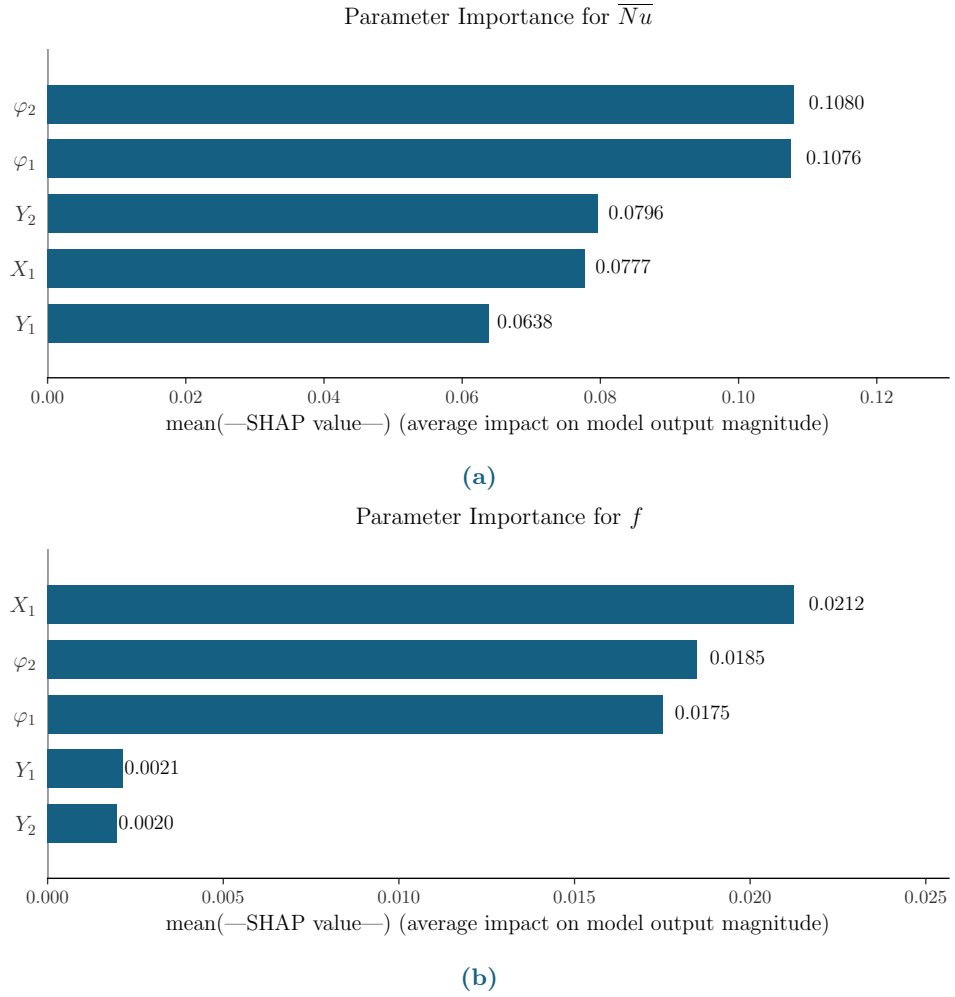


Figure 4.2: Sorting of importance of each feature based on the target value.

In general, the variance across all features is critical for maintaining predictive accuracy; therefore, throughout the active learning stage, dimensionality reduction or feature exclusion was not feasible because of the complex relationship between features and targets, necessitating a

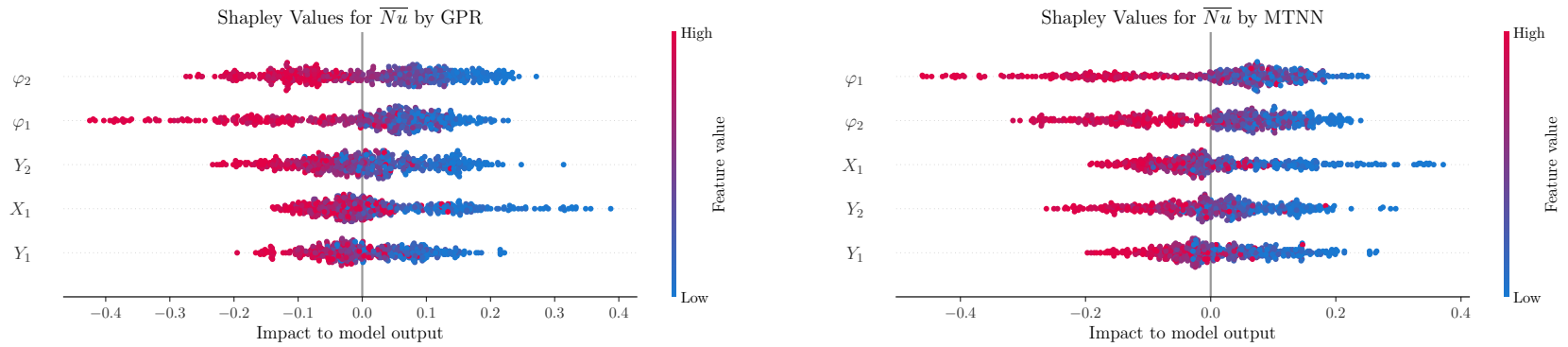
high number of simulations.

In Fig. 4.3, the Shapley values are plotted for two cases: when such values for $\overline{Nu}$ are computed from the GPR model and from the MTNN model. The plots presented here can be interpreted as follows. The vertical axis is sorted in the same way as the importance charts, but the horizontal axis is adjusted to present the variation of each output in accordance with the mean value computed from the dataset. The value of 0 on this axis is related to the fact that the feature does not add any significant deviation from the mean value. Each plotted point is colored based on the color map that appears on the right side of each plot. The color map represents the scaled values that can be assigned to each feature. For example, as illustrated in Fig. 4.3a, the value of φ_2 is the most important parameter for $\overline{Nu}$ based on the GPR model. This parameter tends to decrease the value of $\overline{Nu}$ as it increases (as shown by the red points). That is to say, as φ_2 approaches 45° (its maximum permitted value), $\overline{Nu}$ decreases independently of the values of the remaining features. On the contrary, if φ_2 decreases, $\overline{Nu}$ could increase. This idea is replicated for all figures that show the Shapley values.

As expected, the dots for both Fig. 4.3a and 4.3b tend to be similar to each other; however, the parameter φ_1 is more important when the MTNN model is used. This occurs because, as shown in Fig. 4.2a, the level of influence of both φ_1 and φ_2 is comparable; thus, due to the subtle differences between models, these parameters are interchangeable in terms of significance. The overall observation in the relationship between features and $\overline{Nu}$ is that the combinations of features do not tend to follow a coherent interplay, because in some regions blue and red dots are mixed. This can be interpreted as it is immaterial to have a high value of φ_2 as long as the remaining features are arranged in a way that compensates for the expected influence of φ_2. This observation reflects a non-monotonic trend between the position of the VGs and the value of $\overline{Nu}$ as a consequence of complex feature interactions precluding any direct analytical correlation.

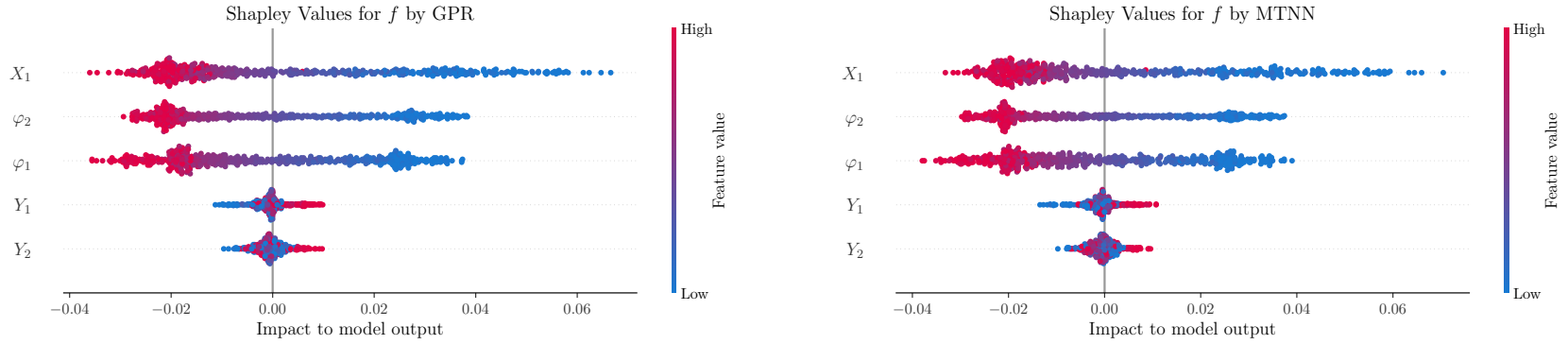
For the case of f, in Fig. 4.4, the Shapley values obtained from both models also tend to be similar. Here is depicted that the values of parameters Y_1 and Y_2 must be in the extremes of their assigned bounds to significantly influence the outcome of f. For the remaining features, in contrast with $\overline{Nu}$, the relationship is, indeed, coherent. For example, if X_1 is 25, f is expected to be low, and it increases as X_1 decreases to 15.5. This reasoning is also applied to φ_1 and φ_2.

In conclusion, GPR and MTNN models are suitable to predict $\overline{Nu}$ and f under the specific conditions explained thus far. It has been reported that there is a percentage difference of less than 1% between predicted and actual values, in addition to the demonstrations of similarity between them through the usage of Shapley values. At this stage, the developed methodology is considered validated and appropriate to move on to the next stage of the present project.



(a) **(b)**

Figure 4.3: Sorting of importance of each feature based on the target value.



(a) **(b)**

Figure 4.4: Sorting of importance of each feature based on the target value.

Chapter summary

The selected algorithms were reviewed and compared through different strategies to find the appropriate algorithm. The tree decision-based algorithm was ruled out because of its inaccurate results, while GPR and ANN algorithms were kept to continue with the project. GPR model serves as a primordial tool when a refinement in the accuracy is required. Instead of using more sampling points created by the Sobol' method, the capability of the model to compute uncertainty was leveraged to actively find the unseen regions in the 5-dimensional space. Based on the findings presented in the literature, the value of Ω was added to improve the predictions for $\overline{Nu}$. While for GPR this parameter was dismissed, it was beneficial for the model based on ANN. To incorporate the value of Ω, the MTNN strategy was employed. Although this parameter was utilized to improve the values of $\overline{Nu}$ predicted by the MTNN model, essentially, the predictions of Ω are not required for any other purpose.

Once 500 samples for the dataset were reached, these models were tested and, with favorable results, were compared through the Shapley values. These coefficients were used to demonstrate the intricate relationship between the features and targets, and why such a phenomenon ruled out the possibility of using the dimensionality reduction approach. While for f the predictions lie primarily on X_1 and $\varphi_{1,2}$, the variation of the whole set of the design variables is significantly correlated with $\overline{Nu}$, resulting in a relationship that is by no means linear.

Finally, both models are considered to be capable of predicting any value correctly as long as the input values are kept within the established boundaries.

CHAPTER 5

THERMO-HYDRAULIC OPTIMIZATION FOR THE HEAT EXCHANGERS

Up to this point, it was observed that the mapping between the 5-dimensional design space and $\overline{Nu}$ is characterized by highly complex feature interactions, as Fig. 4.3 depicts. This observation leads to the conclusion that the resulting $\overline{Nu}$ from each combination of the geometrical arrangement is highly uncertain; thus, an efficient technique must be adapted to find the geometrical arrangement that yields the highest $\overline{Nu}$. On the other hand, the values of f are expected to be low as the angles of the VGs remain near to 45°, and high when the angles tend to be at 0°. Among the values provided by the simulations, a case has been found where η is equal to 1.168, the greatest improvement found thus far. The worst scenario is found to be $\eta = 0.986$ when the value of f is the highest. Because of the observations made previously with the Shapley values, it can be expected to find different arrangements that yield almost the same scenarios.

To validate if the aforementioned values of η represent the maximum and minimum cases for the 5-dimensional space, an optimization task will be performed. Because the values of $\overline{Nu}$ are required to be high with relatively low values for f, multi-objective optimization (MOO) is considered to be employed for the ongoing work. Therefore, optimization is the main topic to be developed throughout the current chapter.

5.1 Optimization concepts

Optimization is used to find the best scenario within a set of alternatives available based on specific constraints and following a certain criteria. For this stage, the design process can be treated as an optimization problem because it shares part of its sequence. For the majority of optimization problems, a five-step procedure can be followed [40].

- Description of the problem/project.
- Collection of data and information related to the problem.
- Definition of the design variables.
- Criteria for the optimization task.
- Formulation of constraints.

At the beginning of this chapter, the objective of the optimization is mentioned: according to Eq. 1.16, it is required to maximize η through maximizing $\overline{Nu}$, keeping f as low or small as possible. These requirements must meet the constraints presented in Table 2.1. The first three steps from the aforementioned five-step procedure are already completed at the DoE stage. Therefore, the last two steps are the ones that need to be carried out.

Optimization problems can be categorized into *continuous optimization* and *discrete optimization* for single or multiple objectives. Based on the characteristics of the problem, a method can be selected and tailored to the specific characteristics analyzed with the five-step procedure. For this particular project, the optimization of the geometrical arrangement takes place in a continuous space within the range established in Table 2.1.

The value of η can be maximized by a single-objective optimization (SOO) procedure; however, this strategy seeks to find the single best solution to the problem by reducing all metrics into one mathematical score, then it evaluates the entire design space purely to maximize or minimize this single parameter. This yields one definitive optimal geometry, but fundamentally masking the underlying physical compromises required to achieve it. Because different scenarios where η is maximum are expected, MOO is preferred to identify all possible optimal configurations for η. Thus, $\overline{Nu}$ will be maximized and f minimized simultaneously.

5.1.1 Multi-objective optimization overview

MOO operates in two completely separate mathematical domains [40],

- **Decision variables (the design space):** These are the physical parameters allowed to change. This is the multi-dimensional space defined by the continuous longitudes (X_1, Y_1, Y_2) and angles (φ_1, φ_2).
- **Objective functions (the objective space):** These are the performance metrics currently being evaluated. For this case, the objective space is a 2-dimensional plane mapping $\overline{Nu}$ against f. The entire goal of MOO is to find how points in the design space map to ideal locations in the Objective Space.

The objectives $\overline{Nu}$ and f are conflicting parameters because physics dictates that improving one inherently degrades the other. That is to say, it cannot be forced to achieve maximum heat transfer without introducing flow resistance; therefore, it is possible that no single mathematical "perfect" geometry exists. Thus, an approach called *Pareto dominance* must be employed. This is a primary logic rule that a specific algorithm uses to rank designs without combining them into a single equation. This logic is explained by the following cases: Design A strictly dominates Design B if it is better in at least one objective and no worse in any others. If Design A yields a higher $\overline{Nu}$ and a lower f than Design B, then Design B is completely obsolete (dominated), and the algorithm discards it. If Design A has a higher $\overline{Nu}$, but Design C has a lower f, neither dominates the other. They are both classified as *non-dominated*; then they are kept alive in the population.

Iteratively, a list is created with the geometric configurations that survived the dominance checks; this is called the *Pareto optimal set*. When the Pareto optimal set is plotted in the objective space, it forms a distinct curve. This curve represents the physical limit of the system predicted for the 5-dimensional space defined by the surrogate model. The optimum values cannot move past this boundary without breaking the laws of physics or expanding the parameter bounds.

In traditional optimization, the selected algorithm for optimization evaluates an explicit algebraic equation. However, when the underlying physics are too complex to be captured by a simple formula or when the governing equations do not have an analytical solution, the surrogate model is used as the objective function. During the optimization loop, algorithms do not calculate values using a mathematical function. Instead, the surrogate model is consulted.

When the optimizer generates a new potential design, it feeds the continuous parameters directly into the MTNN or GPR model. The surrogate instantly predicts $\overline{Nu}$ and f, allowing the optimizer to evaluate the fitness. In the literature, this type of procedure is frequently referred to as *surrogate-based optimization* (SBO).

Several techniques or algorithms have been tailored to be used in MOO problems. For the current project, *genetic algorithms* represent the best option because they stand out as an approach for SBO.

5.1.2 Non-dominated sorting genetic algorithm II

The *non-dominated sorting genetic algorithm II* (NSGA-II), introduced by Deb et al. [41], is an evolutionary algorithm, meaning it mimics the process of natural selection to iteratively 'breed' better solutions over multiple generations. NSGA-II relies on three distinct operational pillars that set it apart from standard single-objective genetic algorithms,

- **Fast non-dominated sorting:** In a standard genetic algorithm, individuals are ranked by a single numerical fitness score. In NSGA-II, the algorithm evaluates every design in the current population and sorts them into distinct "fronts" based on the concept of Pareto dominance. During the selection process, the algorithm heavily favors designs from the lower-numbered fronts, ensuring that the population continually marches toward the true optimal boundary.
- **Crowding distance calculation:** If an algorithm only prioritized dominance, the population might eventually converge into a tight cluster around one single, highly dominant point, failing to map the rest of the optimal curve. NSGA-II solves this by calculating the *crowding distance* for every individual. This metric measures the rectangular volume of empty space surrounding a specific design in the objective space. When the algorithm must choose between two designs that sit on the exact same Pareto front, it acts as a 'tiebreaker', explicitly selecting the design with the larger crowding distance (the one sitting in a more isolated, less populated area). This mechanism guarantees that the final Pareto Front will be broadly and evenly distributed.
- **Elitism:** When moving from one generation to the next, it combines the current parent population with the newly created *offspring* population to form one massive, combined pool. It then performs the non-dominated sorting and crowding distance calculations on this combined group, keeping only the absolute best half to become the parents for the

next generation. This ensures that a high-quality solution, once discovered, can never be lost or accidentally discarded in subsequent generations.

Because NSGA-II requires evaluating thousands of individuals across hundreds of generations, using the CFD process directly to compute each case leads to highly expensive computation tasks. This problem is avoided through the use of a surrogate model, just like the ML models created in the previous chapter. Fortunately, there is no need to write code for this specific algorithm; a library for Python called *pymoo* [42] contains several algorithms for optimization, including NSGA-II.

5.2 Optimization over $\overline{Nu}$ and f

As mentioned beforehand, objective functions are not used for the optimization problem. In lieu thereof, any ML model can be used. Both GPR and MTNN models are used to find the arrangements that provide the best combination of $\overline{Nu}$ and f that yield the maximum values of η. Two similar scripts were made to execute the MOO, one for each ML model. The ranges of the design variables presented in Table 2.1 were utilized as the constraints.

Because the NSGA-II is used only to minimize objectives, the values calculated for $\overline{Nu}$ are set as negative while the algorithm is running. Once the solutions are generated, these values are converted back to their original form. Just 100 candidate points were allowed to exist for

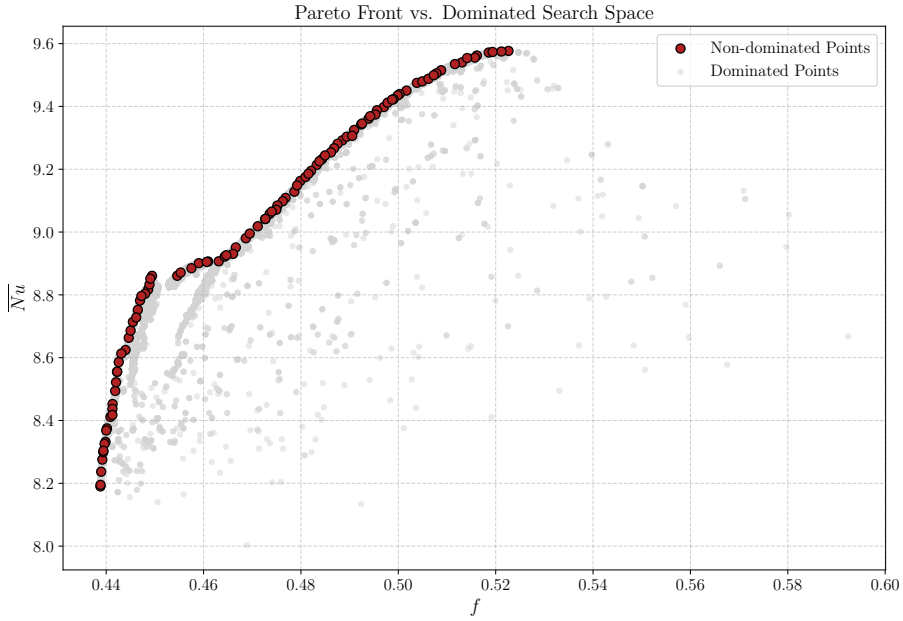


Figure 5.1: Pareto optimal set (in red) obtained by NSGA-II with the GPR model.

each generation. The MOO script is set to analyze a maximum of 2000 evolutionary cycles or generations (mating, mutating, and applying the non-dominated sorting). Convergence is reached if the Pareto optimal set stops improving by less than 0.25%. With both ML models, convergence is achieved after 1000 cycles.

Several points were ruled out as the iterations of the algorithm took place. Tracking those points gives a general idea of the path that the algorithm followed, providing an overall idea of the evolutionary concept applied to the ongoing project. The most significant way to show how the combinations of the objectives were improving after each iteration of the algorithm is through displaying the points that were considered and then discarded. In Fig. 5.1, such points are shown alongside the non-dominated points that build the Pareto front. The *feasible objective space*, defined as the complete set of all physically achievable performance outcomes, can be considered the space where the gray points dwell, with the Pareto front as the boundary that separates this region from the physically improbable combinations.

In Fig. 5.2, the Pareto fronts obtained by using each ML model are displayed. These fronts depict the similarities between the two ML models when the predictions are asked for the best combinations of the objectives. The most noticeable difference between these two fronts is evident near the lower left corner, where the MTNN model predicts better combina-

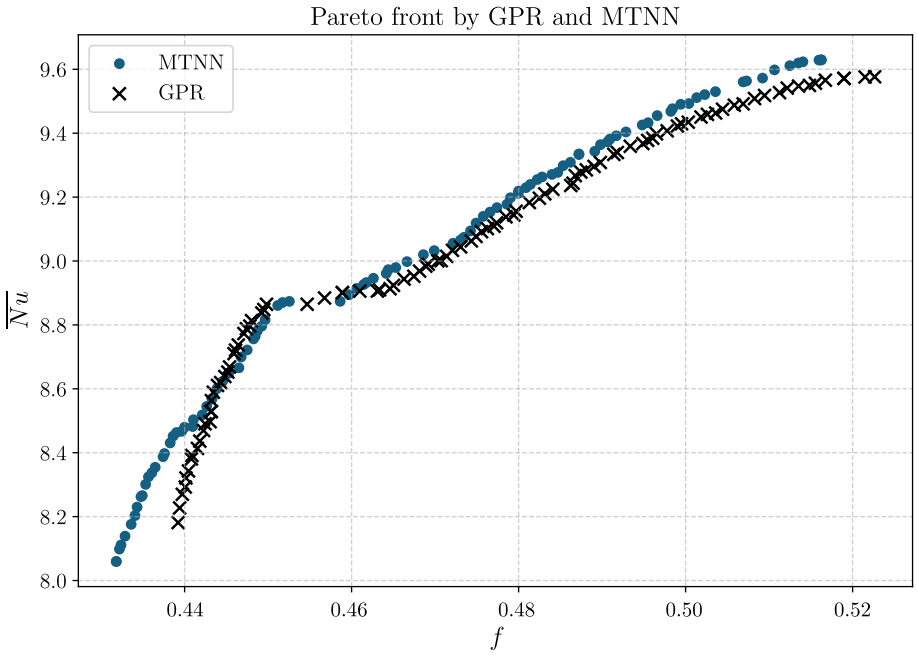


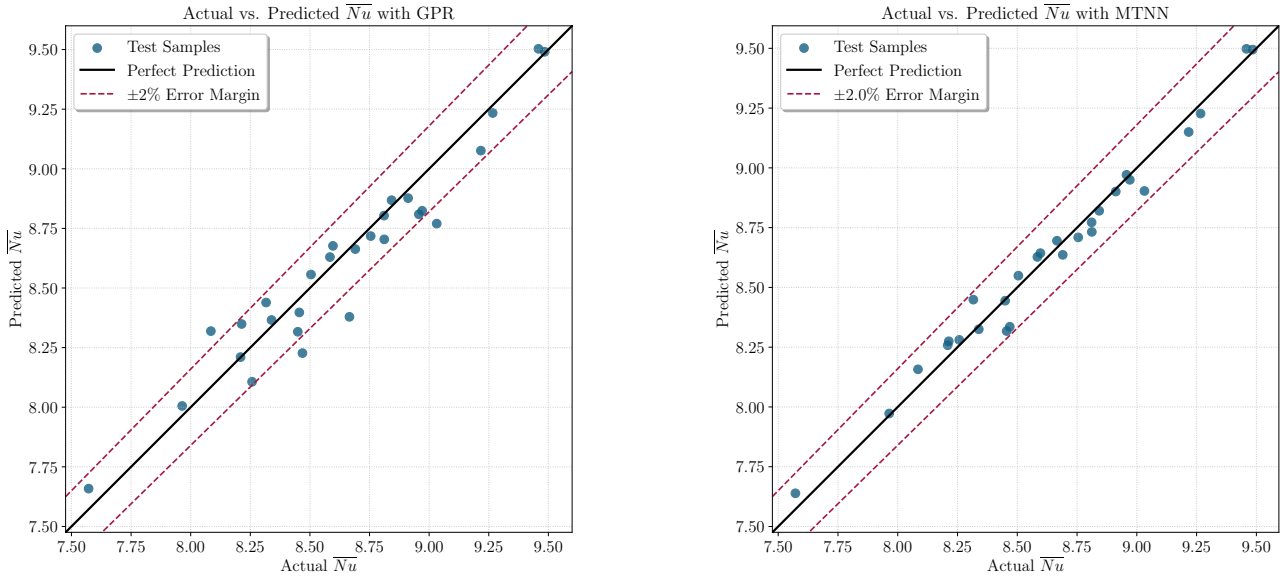
Figure 5.2: Pareto fronts obtained from MOO using NSGA-II

tions than those provided by the GPR model. In this same figure, the relationship between both objectives is described by a *positive covariance* when the optimal set is sought. Therefore, using bow-shaped VGs to increase the heat transfer inevitably increases the pressure drop.

Even though the ML models have been validated, the MOO yield values of $\overline{Nu}$ were not captured for the dataset; hence, a validation of cases that built these Pareto fronts was required. To carry out this activity, random points along the fronts were selected to compute their actual values by CFD simulations.

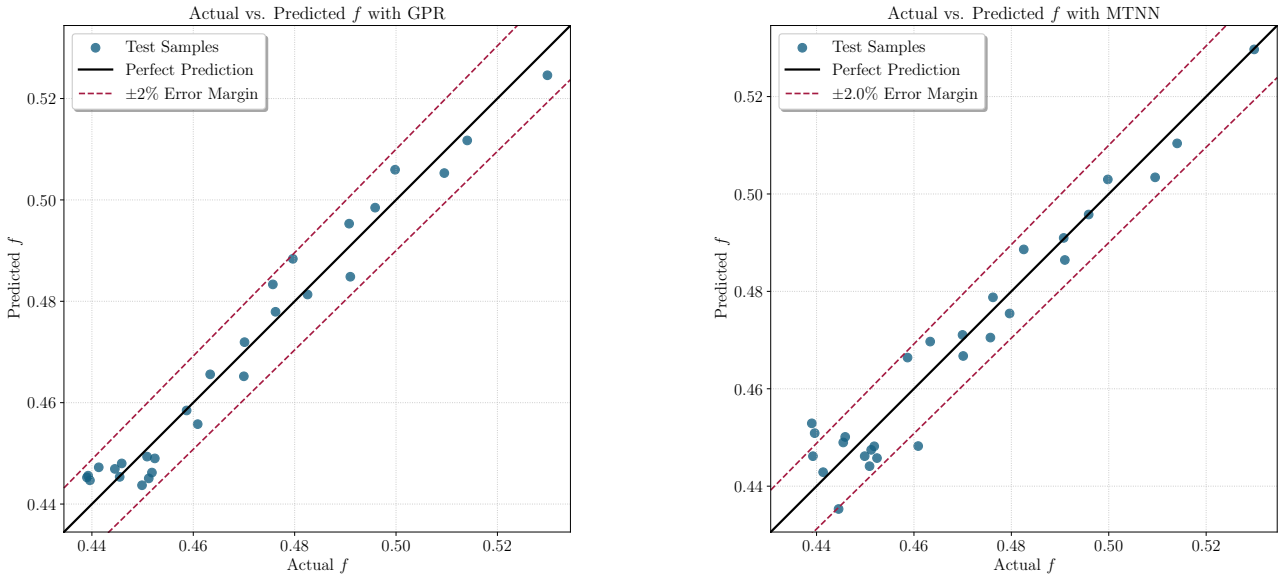
The values obtained from the simulations are used to create a small set for testing. The test results are shown graphically in Fig. 5.3 and in Fig. 5.4, wrapped in two bands that represent ±2% of difference. The tested data with the MTNN model is shown in the chart in Fig. 5.3b, which shows less discrepancy for $\overline{Nu}$ than the data depicted by the use of the GPR ML. While the opposite scenario is presented in Fig. 5.3. The overall observation is that both models are capable of correctly predicting the values that construct the Pareto optimal set.

The data obtained from this last batch of simulations was included in the dataset, which was used to re-train both ML models in order to get better results for the MOO. The optimization problem was solved anew using the updated ML models, providing new Pareto fronts shown



(a) **(b)**

Figure 5.3: Sorting of importance of each feature based on the target value.



(a) **(b)**

Figure 5.4: Sorting of importance of each feature based on the target value.

in Fig. 5.5, where the values of $\overline{Nu}$ are not significantly different from the first MOO results. However, the values related to f are significantly different when they are provided by the MTNN model. Just as was done previously, random points were selected to validate them from CFD simulations. The Tables 5.1 and 5.2 present some of the values obtained from the CFD simulations alongside the predicted values by GPR and MTNN, respectively. In general, both models were improved after being trained with additional points; however, on this occasion, the GPR model was slightly better than the MTNN one.

The additional points obtained from the last execution of the MOO were added to the dataset, completing 557 sampling points. Because of the probabilistic characteristics of the GPR model, it is considered smoother than the model based on ANNs. In consequence, the GPR model will be used for the next steps of the current project.

The maximum value of $\overline{Nu}$ obtained from MOO, combined with its respective f that form a non-dominated point, yielded the best improvement for η. None other unique geometrical arrangement was found for maximum performance. On the other hand, two different cases were found for the worst scenario for η. The overall results from this optimization activity are depicted in Table 5.3, while their respective geometrical arrangements are presented in Table 5.4.

Table 5.1: Comparison of CFD and ML (GPR) predicted outputs.



Table 5.2: Comparison of CFD and ML (MTNN) predicted outputs.



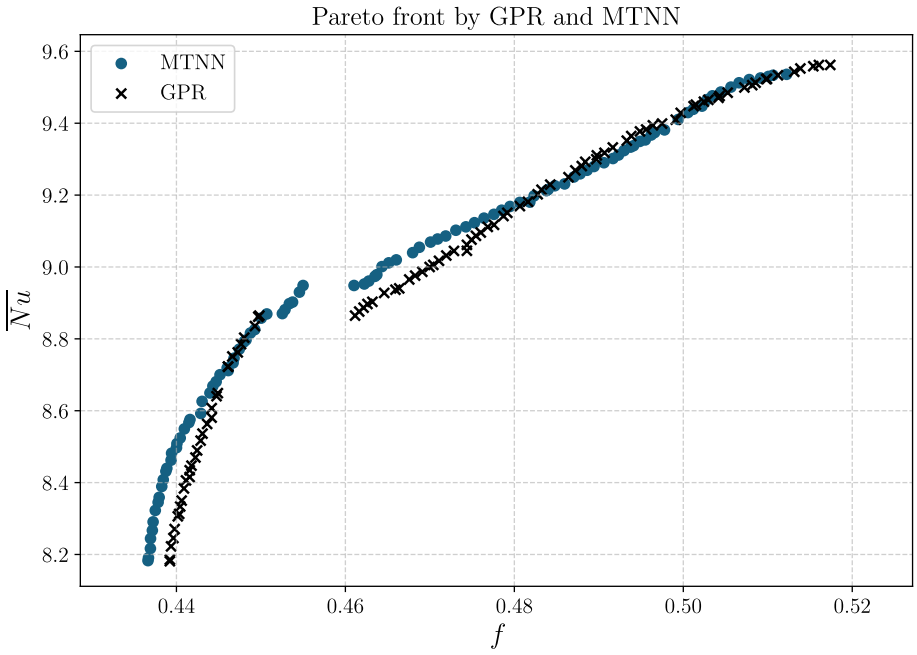


Figure 5.5: New Pareto fronts obtained from MOO using NSGA-II

Table 5.3: Maximum and minimum cases for η compared to the baseline values.

[redacted]

Table 5.4: Geometrical arrangement that yield maximum and minimum scenarios for η.

[redacted]

Chapter summary

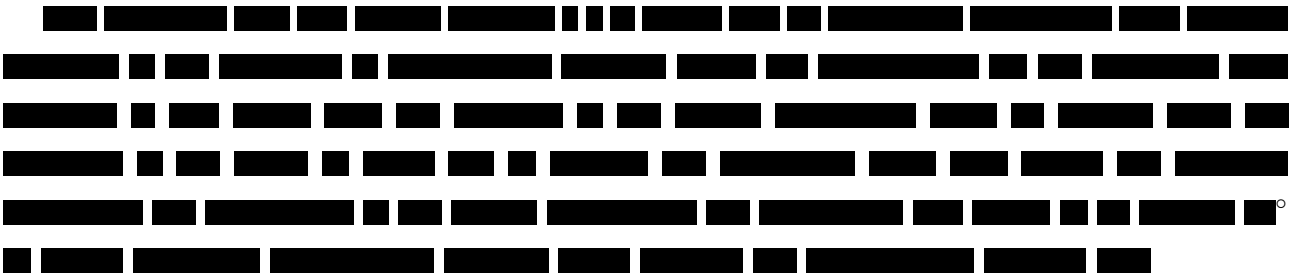
This chapter details the MOO of the FTHE's geometrical arrangement to maximize the overall thermo-hydraulic performance by simultaneously maximizing $\overline{Nu}$ and minimizing f. To bypass the prohibitive computational costs of direct CFD evaluations, surrogate-based optimization was conducted using previously trained GPR and MTNN ML models. The NSGA-II was applied to map the objective space, generating Pareto optimal sets that demonstrated a positive covariance between heat transfer augmentation and pressure drop. Following validation against additional CFD simulations, the dataset was expanded to 557 points, and the GPR model was ultimately selected to represent the system due to its smoother probabilistic characteristics after retraining.

CONCLUSIONS

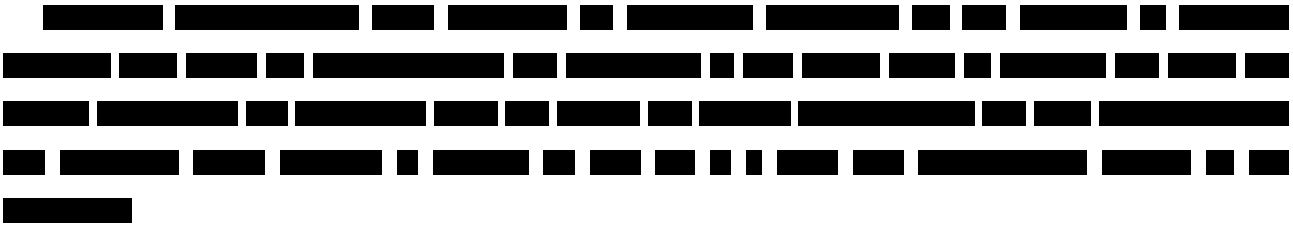
Throughout this document, the theory of analysis and design for finned tube heat exchanger development has been applied to the airside, the region that contributes the most to the thermal resistance. Focusing on this side of the heat exchanger, computational fluid dynamics and machine learning technologies were explored and tailored to the proposed heat exchanger design to study how elliptical tubes and bow-shaped vortex generators can enhance thermo-hydraulic performance.

After successfully validating the methodology of the CFD simulations, over 500 unique cases were simulated to explore the influence of the vortex generators on the surfaces between fins qualitatively. If the unique objective of the project were to find the optimal geometric arrangement of the vortex generators, the number of simulations would have been less than 500 because an active analysis of the simulations would have yielded a tendency of which arrangements provided the best results. Nevertheless, to achieve the general objective of the thesis, machine learning algorithms were selected to explore over the whole domain that the five design variables built, in consequence, requiring a considerable amount of data.

Even though a tree-based decision model was tested with different configurations, only the models based on Gaussian Process Regressor and Artificial Neural Networks were capable of creating reliable predictions. After these models were subjected to Cross-Validation, they proved to be comparable to each other and valid for their intended use; however, the settings for the learning stage for the MTNN model can be configured further to minimize the error in the predictions for Ω, thus proving more accurate predictions for $\overline{Nu}$.



The worst cases are related to a defective position of the vortex generators, where their position is close to the tubes. This arrangement causes the influence of vortex generators and tubes to interfere with one another, leading to the inability to create longitudinal vortices and/or significantly increasing the pressure drop.



Comparing the optimal results with the baseline presented in the benchmark, the improvement was found to be 44% for the thermo-hydraulic performance, which is reflected in an increment of 23% in the heat rate, requiring 7% less pumping power. This result is a combined consequence of the reduction in the friction factor due to the shape of the tubes and the increment in the heat transfer because of the creation of longitudinal vortices.

Assuming isothermal surfaces is a simplification that was adopted to reduce the time and complexity of the simulations. Therefore, while the analyses presented above demonstrate that using bow-shaped VGs in conjunction with elliptical tubes provides a considerable improvement over using rectangular winglets in an HE with circular tubes, the improvement percentages would not be those reported in this study; rather, they would likely be lower in a real-world scenario. That is to say, although the pressure drop values may indeed be equivalent in a measurement of a real FTHE, it must be remembered that $\overline{Nu}$, and in consequence, h_c, depend on the temperature difference between the fluid and the solid interface with which the flow interacts. Consequently, the values of η would differ due to the different temperature gradients.

On the other hand, the reported results would be limited if they were not extrapolated to cases where the fin spacing differs from 3.6 mm, since this distance between surfaces influences the formation and maintenance of longitudinal vortices. Like this separation that forms the channel, the flow regime plays an important role in the formation of vortices and, of course, in heat transfer. This does not mean that using bow-shaped VGs is not recommended for FTHE with different flow regimes and/or different fin spacings (or perhaps with tubes of different sizes); rather, further study is needed to help determine the optimal geometric position of the VGs installed in a different FTHE design. Despite these limitations, using elliptical tubes and bow-shaped vortex generators is highly recommended regardless of the FTHE construction.

The machine learning models, developed in this project, demonstrate the reliable capability to predict values that were not included in the dataset with a maximum difference of 2% from the true value. Keeping the standardization proposed in this project, the machine learning models can be modified to predict results for more variations in the design variables. Additionally, if more degrees of freedom are added to the position of the vortex generators, the structure of the algorithms can remain the same; however, if additional design variables are added, it will be advisable to make the structure of the multi-task neural network more robust to handle the additional relationships between independent variables.

REFERENCES

[1] I.G. Currie. *Fundamental Mechanics of Fluids*. 4th ed. Taylor & Francis, 2013.

[2] Sadik Kakaç, Hongtan Liu, and Anchasa Pramuanjaroenkij. *Heat Exchangers: Selection, Rating, and Thermal Design*. 3rd. CRC Press, 2012.

[3] Ralph Webb and Nae-Hyun Kim. *Principles of Enhanced Heat Transfer*. 2nd ed. Taylor & Francis, 2005.

[4] Latif M. Jiji. *Heat Convection*. 2nd ed. Springer, 2009.

[5] Frank P. Incropera et al. *Fundamentals of Heat and Mass Transfers*. 6th ed. John Wiley & Sons, Jan. 1, 2010.

[6] Keke Xu, Bo Ruan, and Hua Meng. "A thermal performance factor for evaluation of active engine cooling with asymmetric heating". In: *Applied Thermal Engineering* 73.1 (Dec. 5, 2014), pp. 351–356.

[7] M. Fiebig. "Vortices, Generators and Heat Transfer". In: *Chemical Engineering Research and Design*. 5th UK National Heat Transfer Conference 76.2 (Feb. 1, 1998), pp. 108–123.

[8] A. M. Jacobi and R. K. Shah. "Heat transfer surface enhancement through the use of longitudinal vortices: A review of recent progress". In: *Experimental Thermal and Fluid Science*. Generation and Structure of Vortical Flows for Heat Transfer Enhancement 11.3 (Oct. 1, 1995), pp. 295–309.

[9] Vaibhav Harishchandra Bansode et al. "Air-Side Heat Transfer Enhancement Using Vortex Generators on Heat Transfer Surfaces- A Comprehensive Review". In: *Journal of Heat and Mass Transfer Research* 11.2 (Nov. 1, 2024), pp. 255–272.

[10] Zhiqing Bai et al. "Bow-shaped vortex generators in finned-tube heat exchangers; ANN/GA-based hydrothermal/structural optimization". In: *Case Studies in Thermal Engineering* 55 (Mar. 1, 2024), p. 104135.

[11] Liang Xu et al. "Heat analysis and optimal design of fin-and-tube heat exchanger with X-truss vortex generators". In: *International Communications in Heat and Mass Transfer* 152 (Mar. 1, 2024), p. 107248.

[12] Samer Ali et al. "Heat transfer and mixing enhancement by using multiple freely oscillating flexible vortex generators". In: *Applied Thermal Engineering* 105 (July 25, 2016), pp. 276–289.

[13] Shailesh Kumar Sarangi and Dipti Prasad Mishra. "A comprehensive review on vortex generator supported heat transfer augmentation techniques in heat exchangers". In: *Journal of Thermal Analysis and Calorimetry* 149.15 (Aug. 1, 2024), pp. 7839–7867.

[14] Joel H. Ferziger, Milovan Perić, and Robert L. Street. *Computational Methods for Fluid Dynamics*. Cham: Springer International Publishing, 2020.

[15] ANSYS, Inc. *ANSYS Meshing User's Guide*. Release 2023 R2. ANSYS, Inc. Canonsburg, PA, July 2023.

[16] *Ansys Fluent User's Guide*. Release 2023 R1. ANSYS, Inc. Canonsburg, PA, Jan. 2023.

[17] Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. 1st ed. Sebastopol, CA: O'Reilly Media, Sept. 22, 2016.

[18] Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. "Machine Learning for Fluid Mechanics". In: *Annual Review of Fluid Mechanics* 52 (Volume 52, 2020 Jan. 5, 2020), pp. 477–508.

[19] Yidie Luo et al. "Heat Transfer Enhancement in Heat Exchangers by Longitudinal Vortex Generators: A Review of Numerical and Experimental Approaches". In: *Energies* 18.11 (Jan. 2025), p. 2896.

[20] Anupam Sinha et al. "Enhancement of heat transfer in a fin-tube heat exchanger using rectangular winglet type vortex generators". In: *International Journal of Heat and Mass Transfer* 101 (Oct. 1, 2016), pp. 667–681.

[21] Pengfei Wang et al. "An investigation of influence factor including different tube bundles on inclined elliptical fin-tube heat exchanger". In: *International Journal of Heat and Mass Transfer* 142 (Oct. 1, 2019), p. 118448.

[22] Thomas J. Santner, Brian J. Williams, and William I. Notz. *The Design and Analysis of Computer Experiments*. Springer Series in Statistics. New York, NY: Springer, 2018.

[23] Paul Bratley and Bennett L. Fox. "Algorithm 659: Implementing Sobol's quasirandom sequence generator". In: *ACM Trans. Math. Softw.* 14.1 (Mar. 1, 1988), pp. 88–100.

[24] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272.

[25] Robert B. Gramacy. *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*.

[26] Frank W. White. *Fluid Mechanics*. 7th. McGraw-Hill Series in Mechanical Engineering. McGraw Hill, 2011.

[27] Sal Rodriguez. *Applied Computational Fluid Dynamics and Turbulence Modeling: Practical Tools, Tips and Techniques*. Cham: Springer International Publishing, 2019.

[28] *Ansys Fluent Theory Guide*. Release 2023 R1. ANSYS, Inc. Canonsburg, PA, Jan. 2023.

[29] F.R. Menter, R. Lechner, and A. Matyushenko. "Best Practice: RANS Turbulence Modeling in Ansys CFD". In: 1 (2021).

[30] ANSYS Inc. *PyFluent Documentation*. Release 2023 R1. 2025.

[31] P. J. Roache. "Perspective: A Method for Uniform Reporting of Grid Refinement Studies". In: *Journal of Fluids Engineering* 116.3 (Sept. 1, 1994), pp. 405–413.

[32] Ismail B. Celik et al. "Procedure for Estimation and Reporting of Uncertainty Due to Discretization in CFD Applications". In: *Journal of Fluids Engineering* 130.78001 (July 22, 2008).

[33] Li-Min Chang et al. "Numerical study of the relationship between heat transfer enhancement and absolute vorticity flux along main flow direction in a channel formed by a flat tube bank fin with vortex generators". In: *International Journal of Heat and Mass Transfer* 52.7 (Mar. 1, 2009), pp. 1794–1801.

[34] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[35] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 23, 2005.

[36] Josh Patterson and Adam Gibson. *Deep Learning*. 1st ed. Sebastopol, CA: O'Reilly Media, July 27, 2017.

[37] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[38] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 785–794.

[39] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: Curran Associates, Inc., 2017.

[40] Jasbir Singh Arora. "Introduction to Optimum Design". In: *Introduction to Optimum Design*. 4th ed. Academic Press, 2017.

[41] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (Apr. 2002), pp. 182–197.

[42] J. Blank and K. Deb. "pymoo: Multi-Objective Optimization in Python". In: *IEEE Access* 8 (2020), pp. 89497–89509.