HELICOPTER LOBED NOZZLE OPTIMIZATION WITH DIFFERENTIAL EVOLUTION METHOD

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

AKAY BAYAT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
AEROSPACE ENGINEERING

FEBRUARY 2021

Approval of the thesis:

HELICOPTER LOBED NOZZLE OPTIMIZATION WITH DIFFERENTIAL EVOLUTION METHOD

submitted by **AKAY BAYAT** in partial fulfillment of the requirements for the degree of **Master of Science** i**n Aerospace Engineering, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar

Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İsmail H. Tuncer

Head of the Department, **Aerospace Engineering** _____

Assist. Prof. Dr. Mustafa Perçin

Supervisor, **Aerospace Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Sinan Eyi

Aerospace Engineering Dept., METU _____

Assist. Prof. Dr. Mustafa Perçin

Aerospace Engineering Dept., METU _____

Assoc. Prof. Dr. Nilay Sezer Uzol

Aerospace Engineering Dept., METU _____

Assoc. Prof. Dr Utku Kanoğlu

Engineering Sciences Dept., METU _____

Assist. Prof. Dr Onur Baş

Mechanical Engineering Dept., TEDU _____

Date: 15.02.2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name :

Signature :

ABSTRACT

HELICOPTER LOBED NOZZLE OPTIMIZATION WITH DIFFERENTIAL EVOLUTION METHOD

Bayat, Akay
Master of Science, Aerospace Engineering
Supervisor : Assist. Prof. Dr. Mustafa Perçin

February 2021, 145 pages

The main purpose of this study is to develop a methodology for the optimization of an ejector system geometry that is used for the helicopter engine bay cooling. The performance of an ejector system is directly related to the entrainment ratio, which is the ratio of the secondary flow rate to the primary flow rate, and the turbine exit backpressure. In the literature, it is observed that the nozzle geometry used in the ejector system has a significant effect on the performance and that the most efficient geometry is reported to be the lobed shape. For this reason, the optimum lobed nozzle geometry is aimed to be obtained by changing the number of lobes, the exit diameter of the lobe, and the lobe tangent radius in this study. In addition to these parameters, the outlet diameter of the center body that is included in the nozzle structure is also considered as an optimization variable. With the help of a code written in Python, the geometry of the helicopter engine compartment was modeled with "CATIA," and a solution network was created with "Pointwise." The flow analysis of geometry is performed with "Ansys Fluent," and the entrainment ratio and turbine exit back pressure values are calculated. The differential evolution (DE) method is used as an optimization approach.

As a result of the optimization study, the best geometry is obtained with the minimum lobe number. Similarly, the lobe tangent radius is close to the lower bound of the given range. The nozzle exit diameter gets approximately the middle

value of the given range while the cone exit diameter having the maximum value of the given range for the best geometry.

In order to assess the effect of nozzle shape on flow physics and efficiency, the best and worst geometries in all optimization processes are compared. It is observed that the swirl effect that comes from the engine turbine is highly eliminated for the best configuration. Therefore, the cooling flow can be entrained more easily. However, this decreases the mixing efficiency of the flow.The turbine exit back pressure, on the other hand, is higher for this model since the nozzle exit diameter is relatively small.

In addition to the analysis of optimum geometry, effects of all optimization parameters on the performance are examined one by one. It is found that the most influential parameter in the optimization process is the lobe number. It is seen that the nozzle geometry with low-lobe numbers has better performance. On the other hand, the higher cone exit diameter affects the performance well. There is no net effect of the other parameters on the system performance.

Keywords: Helicopter, Ejector, Nozzle, Optimizer, Differential Evolution Method

ÖZ

DİFERANSİYEL EVRİM YÖNTEMİYLE LOPLU HELİKOPTER NOZUL OPTİMİZASYONU

Bayat, Akay
Yüksek Lisans, Havacılık ve Uzay Mühendisliği
Tez Yöneticisi: Dr. Öğr. Üyesi Mustafa Perçin

Şubat 2021, 145 sayfa

Bu çalışmada öncelikli amaç, helikopter motor bölümü soğutulmasında kullanılan ejektör sisteminin optimizasyonu için bir metot geliştirmektir. Ejektör sisteminin performansı sürüklenme oranın artışı ve türbin geri basıncının azalmasıyla doğru orantılıdır. Yapılan literatür taramasında ejektör sisteminde kullanılan lüle yapısının performans üzerinde önemli bir etkisi olduğu ve en verimli yapının loplu yapı olduğu gözlemlenmiştir. Bu sebeple, loplu lüle yapısının lop sayısı, lüle çıkış çapı ve lop tanjant yarıçapı değiştirilerek optimum geometri elde edilmeye çalışılmıştır. Bunlara ek olarak, lüle yapısına yerleştirilen merkezcil gövde çıkış çapı da parametrelere eklenmiştir. Python üzerinde yazılan bir kod yardımıyla helikopter motor bölgesinin geometrisi "CATIA" ile modellenmiş ve "Pointwise" ile çözüm ağı oluşturulmuştur. Oluşturulan geometri için akış analizi "Ansys Fluent" ile gerçekleştirilip sürüklenme oranı ve türbin geri basıncı hesaplanmıştır. Optimizasyon yöntemi olarak ise diferansiyel evrim metodu seçilmiştir.

Bu optimizasyon çalışmasının sonunda, en iyi geometri düşük lop sayılı lüle ile elde edilmiştir. Benzer şekilde, lop tanjant yarıçapı verilen aralıkta en düşük değere yakındır. Lüle çıkış çapı, verilen aralığın yaklaşık olarak orta değerini alırken, koni çıkış çapı, en iyi geometri için verilen aralığın maksimum değerine sahiptir.

Lüle geometrisinin akış fiziği ve verimlilik üzerindeki etkilerini değerlendirmek için, en iyi ve en kötü geometriler karşılaştırılmıştır. En iyi konfigürasyon için motor türbininden gelen dönen akış etkisinin azaldığı görülmüştür. Böylece, soğutma havası kolaylıkla çekilebilmektedir. Ancak, bu akış karışım verimini düşürmektedir. Ayrıca türbin çıkış basıncı lüle çıkış çapı daha düşük olduğu için daha yüksektir.

Optimum geometrinin analizine ek olarak, bütün optimizasyon parametrelerinin performans üzerindeki etkileri ayrı ayrı incelenmiştir. Lop sayısının optimizasyonda en etkili parametre olduğu bulunmuştur. Lop sayısı düşük lülelerin daha iyi bir performansa sahip olduğu görülmüştür. Öte yandan, daha yüksek merkezcil gövde çıkış çapı sistem performansını iyi yönde etkilemektedir. Diğer parametreler için ise böyle kesin bir yargı bulunmamaktadır.

Anahtar Kelimeler: Helikopter, Ejektör, Lüle, Optimizasyon, Diferansiyel Evrim Metodu

To my love and my family…

ACKNOWLEDGMENTS

I would like to express my gratitude to Asst. Prof. Dr. Mustafa PERÇİN for guiding this work.

I specially thank to the my colleagues and friends Onur BAŞ, Gökhan ASLAN, Emre SANCAR and Abdurrahman Burak DALDAL.

I also would like to state that this study could not be accomplished without the support and guidance of my chief engineer Kerem DENK and my manager Yüksel ORTAKAYA.

Above all, I would like to offer the most and my sincere gratitude to my love and wife Mrs. Buse ARAR BAYAT, my father Mr. Kemal BAYAT, and my mother Mrs. Meryem BAYAT.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xv

LIST OF FIGURES xvii

LIST OF ABBREVIATIONS xxii

LIST OF SYMBOLS xxiii

CHAPTERS

1 INTRODUCTION 1

1.1 Applications and Investigations of the Ejector Systems 2

1.2 Usage of Ejector Systems in Helicopter Engine Bay Cooling 4

1.3 Objective of the Thesis 9

2 COMPUTATIONAL METHODS AND TOOLS 11

2.1.1 Geometry Generation with CATIA 12

2.2 Grid Generation with Pointwise 13

2.3 CFD Analysis with Fluent 14

2.3.1 Turbulence Modeling and Simulation 15

2.3.1.1 Reynolds-averaged Navier Stokes (RANS) 15

2.3.2 Realizable k-ε Turbulence Model 16
2.3.3 k-ω SST Turbulence Model 17
2.3.4 Validation of the Turbulence Models 17
2.3.4.1 Validation Case 18
2.3.4.2 Numerical Setup and Grid Generation 19
2.3.4.3 Numerical Convergence 21
2.3.4.4 Results of the Validation Case Simulations 22
2.3.5 Grid Convergence 25
2.4 Computational Mesh Properties 28
2.5 Hardware Properties 30
3 OPTIMIZATION PROCESS 31
3.1 Differential Evolution Method with Python 31
3.2 Nozzle Optimizer Module 34
3.3 Objective Function Calculator 36
3.3.1 Geometry Generation 36
3.3.2 Grid Generation 38
3.3.3 Solver Functions 40
3.4 Objective Function 41
4 RESULTS AND DISCUSSION 43

4.1 Input Parameters 43
4.1.1 Flow Field in the Ejector System 44
4.1.2 Optimization Results for Three Nozzle Variables 50
4.1.2.1 Effects of the Optimization Variables on the Flow Physics with the Base Function 51
4.1.2.1.1 Comparison of the Best and Worst Configurations 53
4.1.2.2 Effects of the Optimization Variables on the Flow Physics with the Backpressure Dominated Function 60
4.1.2.2.1 Comparison of the Best and Worst Configurations 62
4.1.2.3 Comparisons of Two Different Optimization Results 66
4.1.3 Optimization Results for Four Nozzle Variables 68
4.1.3.1 Effects of the Optimization Variables on the Flow Physics with the Base Function 68
4.1.3.1.1 Comparison of the Best and Worst Configurations 69
4.1.4 Comparisons of Optimization Results for 3 and 4 Parameters 74
5 CONCLUSION AND FUTURE WORK 79
5.1 Conclusion 79
5.2 Future Work 81
REFERENCES 83
APPENDICES 93

A. SAMPLE CALCULATION for DIFFERENTIAL EVOLUTION METHOD 93

B. PYTHON CODE and INPUT LIST 99

C. ANALYSIS RESULTS 137

LIST OF TABLES

TABLES

Table 2.1 Surface and volume mesh numbers for different grid sizes..... 25

Table 2.2 Boundary conditions for simulated engine compartment 26

Table 2.3 Differences in the performance parameters of different mesh sizes..... 27

Table 4.1 Samples of parameter value round operation 44

Table 4.2 Results for different lobe numbers for the base cost function solution .. 51

Table 4.3 Results for different nozzle exit diameters for the base cost function solution..... 52

Table 4.4 Results for different lobe tangent radius values for the base cost function solution..... 53

Table 4.5 Parameters and results of the best and worst configurations for the base cost function solution..... 54

Table 4.6 Results for different lobe numbers for the backpressure dominated cost function solution 60

Table 4.7 Results for different nozzle exit diameters for the backpressure dominated cost function solution..... 61

Table 4.8 Results for different lobe tangent radiuses for the backpressure dominated cost function solution..... 61

Table 4.9 Parameters and results of the best and worst configurations for the backpressure-dominated cost function solution..... 62

Table 4.10 Parameters and results for the best cases of the base and backpressure-dominated cost function solutions 67

Table 4.11 Results for different cone exit diameters for the base cost function solution 69

Table 4.12 Parameters and results of the best and worst configurations for the base cost function solution 70

Table 4.13 Parameters and results for the best cases of the base function solutions with three and four parameters 75

LIST OF FIGURES

FIGURES

Figure 1.1. Flow motion through ejector system and related sections [1] 2

Figure 1.2. Representation of ejector system in the helicopter engine bay 5

Figure 1.3. Ejector flow vs. exhaust interfaces total pressure by changing ejector system parameters (1) [42] 6

Figure 1.4. Dimensions of different nozzle types [42] 8

Figure 1.5. Entrainment ratio versus backpressure for different formed nozzles [42] 8

Figure 1.6. The change of optimization parameters and related results, final configuration obtained end of the process [43]. 9

Figure 2.1. Working scheme of the optimization process 11

Figure 2.2. Helicopter engine compartment geometry 12

Figure 2.3. Dimensions of the helicopter engine compartment 13

Figure 2.4. A sectional view of the computational grid for analysis 14

Figure 2.5. Geometrical parameters of the test nozzle 18

Figure 2.6. Air supply system in the experiment 19

Figure 2.7. Experimental setup geometry used in the analysis 19

Figure 2.8. Experimental setup grid generation 20

Figure 2.9. The cross-sectional views of grids for k-ε (upper) and k-ω (lower) 21

Figure 2.10. Residuals for the realizable k-ε (left) and k-ω SST (right) turbulence models 22

Figure 2.11. Comparisons of experimental data (1st row) with the realizable k-ε (2nd row) and the k-ω SST (3rd row) models in terms of the in-plane velocity vectors and magnitude of the axial velocity (out-of-plane velocity) distributions at different streamwise (z/D) locations 23

Figure 2.12. The maximum value of the azimuth vorticity of the lobed nozzle for experimental data and analysis 24

Figure 2.13. Variation of the cooling inlet mass flow rate and turbine exit total pressure (gauge) for the last 250 iterations of the numerical solutions for different computational grids 26

Figure 2.14. Turbine exit backpressure and cooling inlet flow rates for medium size grid structures. 28

Figure 2.15. Cross-sectional views of the engine compartment critical regions 29

Figure 3.1. An optimization tool written in Python using and combining different software through macros and scripts and an objective function module 31

Figure 3.2. Two-dimensional example of a mutation step 33

Figure 3.3. Parameters to be optimized for nozzle and center body 34

Figure 3.4. Boundary conditions, mesh parameters, and solver parameters of the solution 36

Figure 3.5. Generic parameters used in CAD functions and an angled line generation 37

Figure 3.6. Angled line sub-function 38

Figure 3.7. Importing database to Pointwise in objective function calculator module 38
Figure 3.8. Subfunction for Importing database to Pointwise 39
Figure 3.9. Glyph file part of database importing to Pointwise..... 39
Figure 3.10. Writing of solver parameters, mesh scaling, and numerical solution method in solver subfunction. 40
Figure 4.1. Contours of velocity magnitude of three components plotted on xz plane for the sampe solution 45
Figure 4.2. Contours of axial velocity (x-velocity) with vector fields plotted on xz plane for the sample solution 46
Figure 4.3. Contours of velocity magnitude of three components plotted in xz and a number of yz planes downstream of the nozzle exit for the sample solution 47
Figure 4.4. Streamlines generated from the center of exit zone for the sample solution..... 48
Figure 4.5. Streamlines generated from the center body outlet for the sample solution..... 48
Figure 4.6. Streamlines generated from the secondary (cooling) inlet for the sample solution..... 49
Figure 4.7. Streamlines generated from the primary (nozzle) inlet for the sample solution..... 50
Figure 4.8. The geometries for the best and worst cases of the base cost function solution..... 53

Figure 4.9. Velocity distributions for cold regions for the best and worst cases of the base cost function solution 55
Figure 4.10. Contours of axial velocity (x-velocity) plotted on yz planes at different downstream positions from the nozzle exit for the best and worst cases of the base cost function solution 56
Figure 4.11. 58
Figure 4.12. Static pressure contours at the turbine exit for the best and worst cases of the base cost function solution 59
Figure 4.13. The geometries for the best and worst cases of the backpressure-dominated cost function solution 62
Figure 4.14. Velocity distributions for cold regions for the best and worst cases of the backpressure-dominated cost function solution 63
Figure 4.15. Contours of axial velocity (x-velocity) plotted on yz planes at different downstream positions from the nozzle exit for the best and worst cases of the backpressure-dominated cost function solution 64
Figure 4.16. Contours of temperature plotted on yz planes at different downstream positions from the nozzle exit for the best and worst cases of the backpressure-dominated cost function solution 65
Figure 4.17. Static pressure contours at the turbine exit for the best and worst cases of the backpressure-dominated cost function solution 66
Figure 4.18. The geometries for the best cases of the base and backpressure-dominated cost function solutions 67

Figure 4.19. The geometries for the best and worst cases of the base cost function solution 70
Figure 4.20. Axial velocity distributions for cold regions for the best and worst cases of the base cost function solution 71
Figure 4.21. Contours of axial velocity (x-velocity) plotted on yz planes at different downstream positions from the nozzle exit for the best and worst cases of the base cost function solution 72
Figure 4.22. Contours of temperature plotted on yz planes at different downstream positions from the nozzle exit for the best and worst cases of the base cost function solution 73
Figure 4.23. Static pressure contours at the turbine exit for the best and worst cases of the base cost function solution 74
Figure 4.24. The geometries for the best cases of the base function solutions with three and four parameters 75
Figure 4.25. Contours of axial velocity (x-velocity) plotted on yz planes at different downstream positions from the nozzle exit for the best cases of the base cost function solutions with three and four parameters 77
Figure 4.26 Contours of temperature plotted on yz planes at different downstream positions from the nozzle exit for the best cases of the base cost function solutions with three and four parameters 78

LIST OF ABBREVIATIONS

ABBREVIATIONS

CAD	Computer added drawing
CCW	Counterclockwise
CFD	Computational fluid dynamics
D	Diameter
DE	Differential evolution
DNS	Directly numerical solution
DP-SPIV	Dual-Plane Stereoscopic PIV
ER	Entrainment ratio
GE.	General electronics
F	Weight factor
K	Kelvin
L	Length [mm]
LES	Large-eddy simulation
NP.	Population size
Pa	Pascal
PLIF	Planar laser-induced fluorescence
PIV	Particle image velocimetry
POF	Objective function of population
RANS	Reynolds-averaged Navier Stokes

LIST OF SYMBOLS

SYMBOLS

F_{obj}	Objective function
k	Turbulent kinetic energy
$\dot{m}_{ci}$	Cooling inlet mass flow rate
$\dot{m}_{te}$	Turbine exit mass flow rate
n_l	Lobe number
P_{amb}	Ambient pressure
P_{te}	Turbine exit total pressure
y^+	Dimensionless wall distance
α	Angle (°)
∂	Partial derivative
μ_T	Turbulent viscosity
ε	Dissipation rate
ρ	Density
s	Mean rate of the stress tensor
U	Axial Velocity
W	Tangential Velocity

CHAPTER 1

INTRODUCTION

Helicopters are widely used in many areas such as the transportation of people and cargo, military applications, firefighting, or agriculture because they can operate in various ground and weather conditions. Therefore, the design of a helicopter is very challenging, especially for hot weather conditions. In hot weather, engine performance decreases, and cooling of the engine bay and other systems gets harder. Since the engine must always run, its compartment should be continuously cooled. For this reason, an ejector system is used for engine bay cooling due to its low complexity and cost.

The ejector system is mainly used for removing or transferring any fluid from a region to another one. According to the application, the working fluid can be selected as water, steam, refrigerant, or any other fluid type.

An ejector system comprises four sections: primary and secondary inlet, nozzle, mixing chamber, and diffuser. The high velocity (primary) fluid is ejected through a nozzle with high pressure and creates a suction flow from the secondary inlet. Two fluids are mixed in the mixing chamber and removed to the outside or other section by a diffuser. Flow motion and sections of an ejector system are shown in Figure 1.1

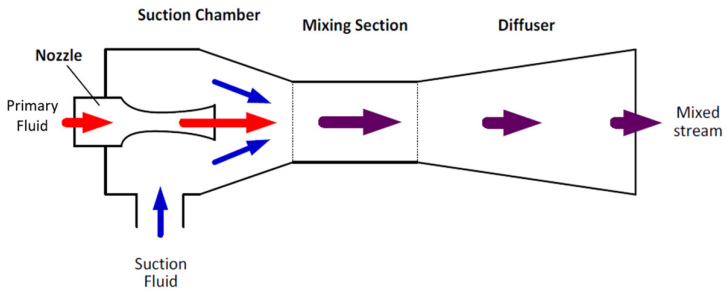


Figure 1.1. Flow motion through ejector system and related sections [1]

Figure 1.1 illustrates that primary fluid with high energy and pressure is supplied to a nozzle from a source like a condenser or an engine. A secondary fluid called suction fluid with low pressure is sucked with high velocity and primary fluid pressure. These two fluids are well mixed in the mixing section, and the resultant mixed stream is discharged outside via a diffuser. The type of fluids, shape, and dimensions of the system may show variations according to the application.

The efficiency of the system is assessed by using the entrainment ratio, which is the ratio of the suction mass flow rate to the primary mass flow rate and the backpressure that occurs at the exit of the nozzle [2].

1.1 Applications and Investigations of the Ejector Systems

Ejector systems have been used in many areas throughout history due to their simplicity and efficiency. The use of ejectors dates back to the middle of the 19th century. Firstly, a French inventor Giffard developed an ejector system to supply liquid water to steam generators in 1858, and his research was studied in detail by Kranakis [3]. Since the steam expansion, the water flow rate was controlled by changing the primary nozzle exit area using a needle. Because of these advantages, the usage of ejectors has been increased widely. By1860, systems have been used in locomotives and the French fleet [4]. In 1901, an ejector was invented to remove air from the steam engine condenser by Parsons [5]. Ejectors were first used in a jet refrigeration system to create a low-pressure container where water is evaporated and working fluid cools [6]. After the benefits of the ejector system in terms of cooling were observed, its use has become quite common in various areas such as water supply in nuclear reactors [7], fuel cell systems [8], and air conditioning of buildings [9], [10], [11], [12]. The first usage of ejector systems in aircraft applications for thrust augmentation was also seen around these dates [13]. In 1931, Gay [14] patented an ejector system to cool refrigerant in refrigeration

systems, unlike air conditioning. Until 1939, all studies were made with data obtained through trial and error. Flügel [15] developed a method using energy and impulse equations to design an ejector system. In 1946, Mcclintock and J. Hall Hood [16] proposed a complicated approach that includes non-uniformities of flow for aircraft ejector system design. These non-uniform effects are formed by using empirical equations.

Turbine engine cooling was firstly developed in the 1950s. J.W. Mitchell has derived a mixing constant, which shows the effectivity of mixing between primary and secondary flows [17]. These constants are still applicable in the SAE Aerospace Information Report, including calculation steps for 1-D preliminary ejector design [18].

Several experimental studies were conducted to develop previous expressions and observe flow fields after performing these analytical approaches. R. Porcar et JP [19] Panel visualized shock waves generated in a supersonic ejector by light scattering. Then they improved this method by adding polarization properties of the scattered light [20]. Similar studies to visualize flow fields were performed by different techniques such as laser-induced fluorescence [21], laser tomography technique [22], optical Schlieren [23]. Also, to see the mass and thrust effects of the system with multiple ejectors, M. S. Chandrasekhara et al. [24] carried out several experiments.

In addition to experimental studies, the flow field of the system is simulated by using CFD analysis. In 1993, R. S. Nave [25] analyzed diffusers' performance in gas-powered jet pumps in two dimensional. S. B. Riffat et al. [26] improved this by solving a three-dimensional axisymmetrical ejector geometry. After computational costs are decreased, and capabilities of computing machines are increased, such studies gained momentum. E. Rusty [27] et al. validated CFD results with experimental data, and the effect of ejector geometry is examined in terms of entrainment ratio. A similar study is conducted by T. Sriveerakul et al. [28], [29] for the steam ejector. They also studied the operation condition impacts. In addition

to subsonic flow solutions, supersonic conditions are analyzed in several studies [30], [31], [32].

In 1986, to solve a large class of supersonic ejector optimization problems, a technique based on the one-dimensional constant-field flow model was developed by Dutton and Carrol [33]. Bartosiewicz et al. [34] validated the k-omega SST model with experiments to calculate the flow field in an ejector system. Using this turbulence model, an optimized ejector model was developed for supersonic flow. Similar studies for the optimization of ejector systems and validation of analysis have been performed by, Manikanda et al. [35], Croquer et al. [36], Gagan et al. [37], and Maghsoodi et al. [38] with different approaches.

After these investigations, several optimization studies have been performed on either ejector system performance or validation of turbulence models on the system flow analysis.

1.2 Usage of Ejector Systems in Helicopter Engine Bay Cooling

In a helicopter, turboshaft engines are used to supply required shaft power to blades. Engine and its accessories such as oil tank, combustion chamber should be cooled to operate under their allowable component limits. Furthermore, the required air change in the engine compartment should be provided to prevent the accumulation of hazardous fluid vapors such as oil and fluid. Similar issues occur in an aircraft engine compartment. Since aircraft always have a forward flight, engine accessories can be cooled by bypass air. However, helicopters have many different operations without having a forward flight velocity, such as hover or ground operations. Therefore, a forced-air cooling system is used in the engine nacelle like a fan or an ejector system. The ejector system is widely used for cooling helicopter engine equipment due to its simple construction, high reliability, and low maintenance cost, as previously mentioned. The turbine exit flow is used as the primary flow, and cooling flow is supplied from cut-outs on compartment

nacelle by primary flow suction effect. A schematic representation of the engine compartment of a helicopter with an ejector system is given in Figure 1.2

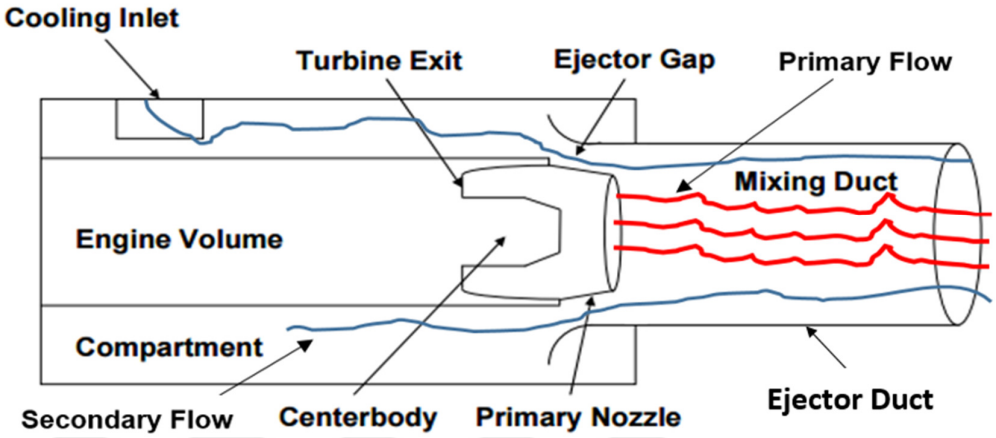


Figure 1.2. Representation of ejector system in the helicopter engine bay

Primary flow leaves the engine from the rotating airfoils called rotor blades. These rotating blades cause the increase of primary flow radial velocity components. This rotational motion is describe with the swirl number, which is the ratio of the axial flux of angular momentum to the axial thrust. Swirl number can be calculated by using the below formula:

$$S_n = \frac{\int_0^R \rho U W r^2 dr}{R \int_0^R \rho U^2 r dr} \quad (1)$$

where U and W are the axial and tangential velocity components, respectively, and R is the swirl radius [39].

Primary flow expands to the ejector ducts due to the swirl effect, blocks the secondary flow, and causes the decrease of entrainment ratio. Similar to the other application areas, the increase of the entrainment ratio is the main objective of the ejector system in helicopter nacelle cooling. Therefore, this swirl should be eliminated to increase the entrainment ratio. Besides, backpressure on engine turbine exit is of high importance for engine performance [40]. For this reason, several studies have been performed to investigate the effects of several parameters

on system performance. Kazlauskas [41] analyzed the impact of the primary nozzle size, mixing duct length, and cooling inlet size in terms of entrainment ratio and turbine exit backpressure. The summary of this study is represented in Figure 1.3

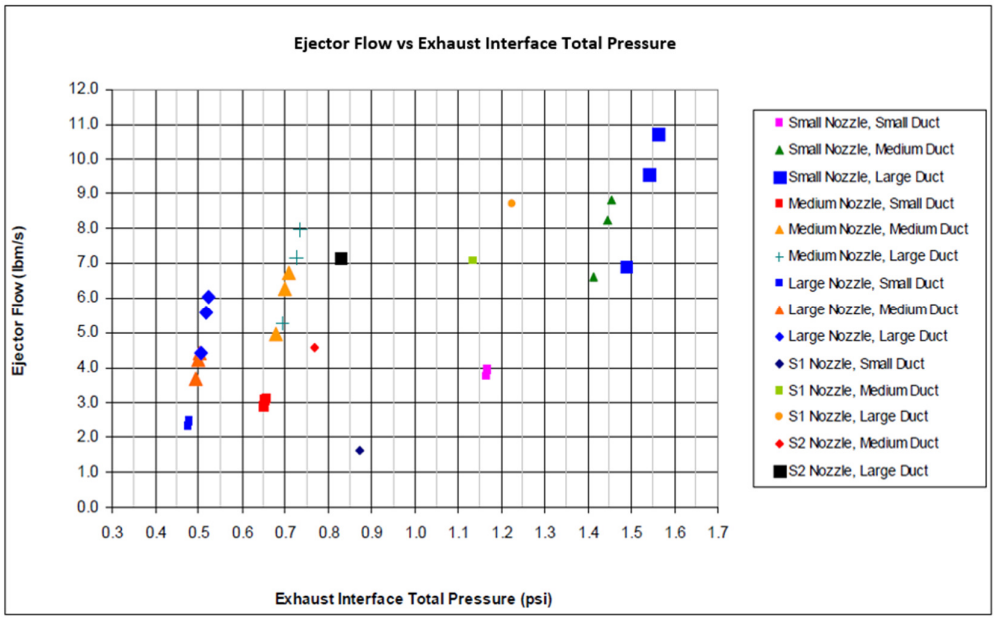


Figure 1.3. Ejector flow vs. exhaust interfaces total pressure by changing ejector system parameters [(1)] [42].

[(1)] S1 and S2 represent primary nozzles with a larger outer diameter, S2> S1 > S.

According to this study, the smaller the nozzle outlet diameter is, the more cooling mass flow rate is achieved. However, this configuration has the highest backpressure at the engine exit. Similarly, ejector duct length has the same effect on the system performance. Turbine exit backpressure and entrainment ratio increase with the ejector duct length. The most impressive result is that the cooling inlet size or other parameters are futile if the nozzle and the ejector gap are too small. This small gap causes choking of secondary flow and decreases the system performance. For this reason, the effects of system parameters are complicated to determine.

In the study conducted by Kazlauskas [42], it is concluded that the nozzle size has the most significant influence on the ejector system performance. However, the effect of the primary nozzle area is investigated instead of its shape. Yang et al. [42] compared five nozzles with different forms to examine the impact of the nozzle shape on entrainment ratio, backpressure, and mixing. In Figure 1.4, the profile and dimensions of the nozzles used in the study are shown.

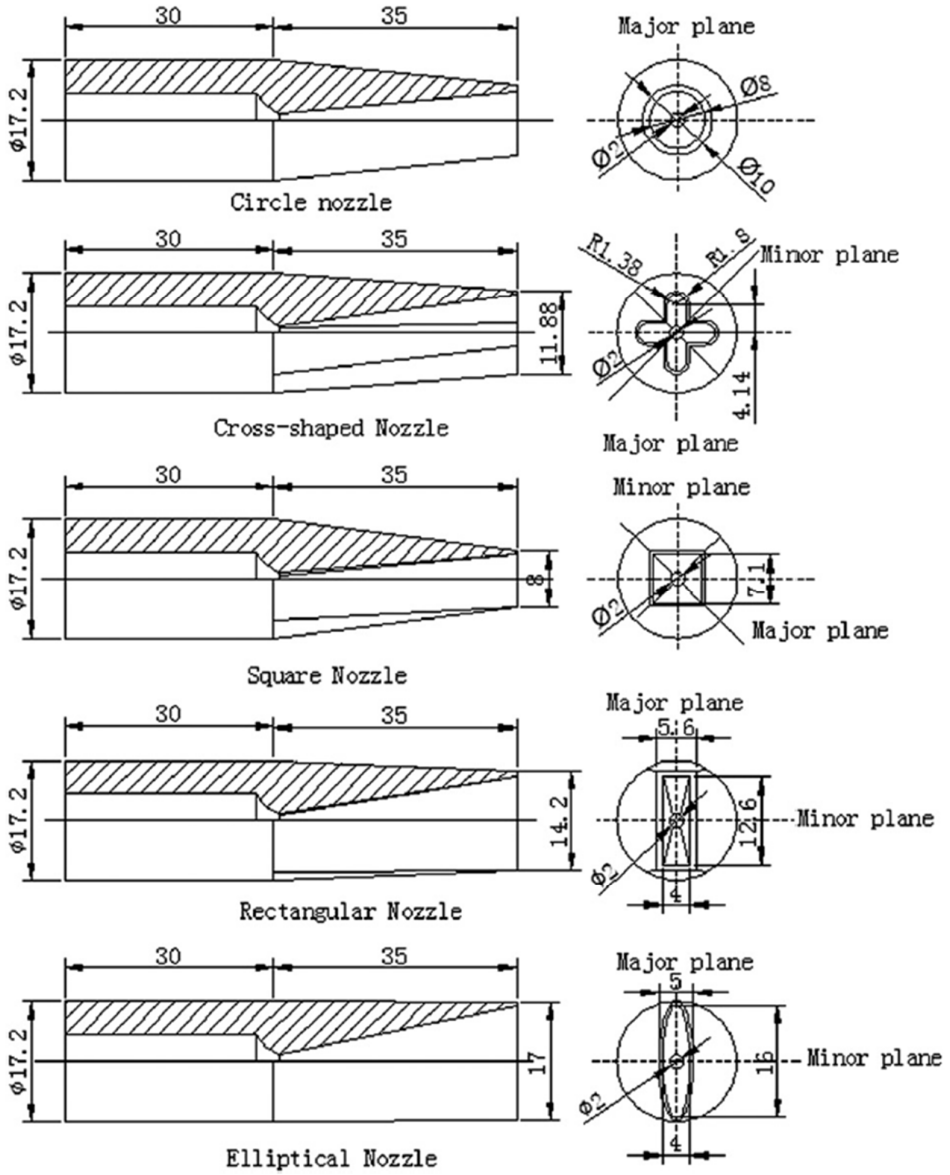


Figure 1.4. Dimensions of different nozzle types [42]

Performance comparison of these nozzles is given in Figure 1.5. As shown in Figure 1.5, a cross-shaped nozzle gives the highest secondary flow rate with low backpressure. The cross-shaped nozzle is named a lobed nozzle in literature.

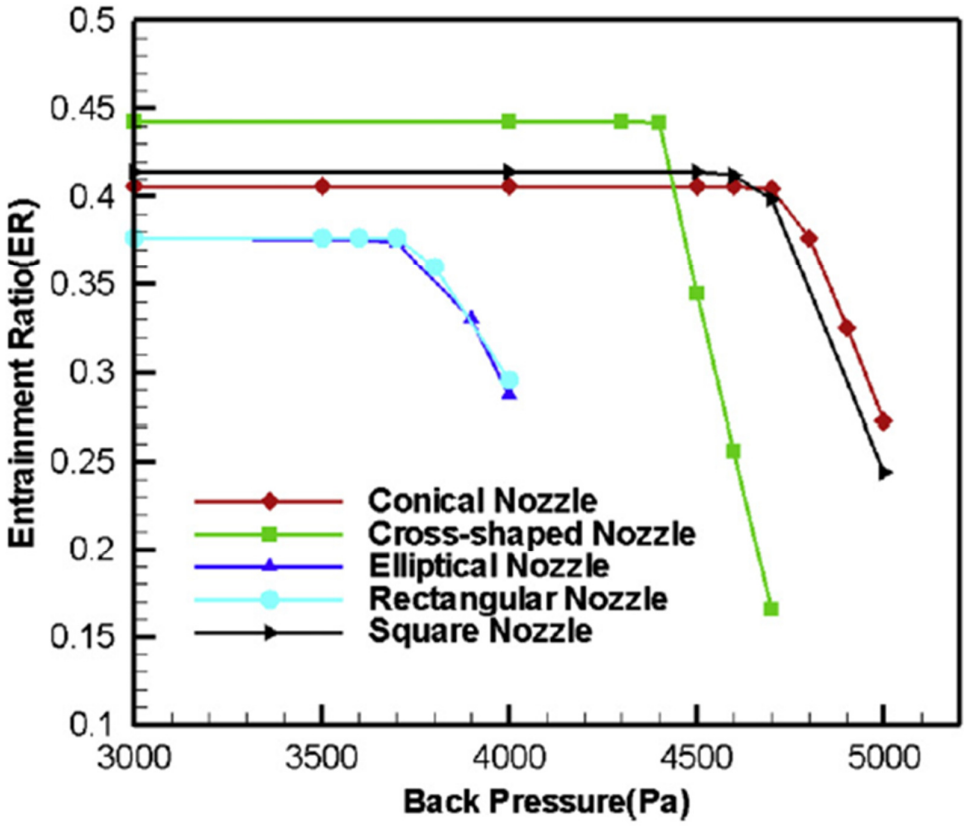


Figure 1.5. Entrainment ratio versus backpressure for different formed nozzles [42]

Dvorak [43] performed an optimization study on the lobed nozzle in order to increase the entrainment ratio. Lobe number, exit diameter, and the height of the lobed nozzle are selected as optimization parameters. Rosenbrock's method was chosen as an optimization algorithm in this study. While one parameter has been changed at each step, the others were kept constant. Optimization trials, analysis results, and characteristics of the final configuration are given in Figure 1.6. In the

given figure, lobe number, height, exit diameter, and secondary mass flow rate are represented with n, H, de, and m2, respectively.

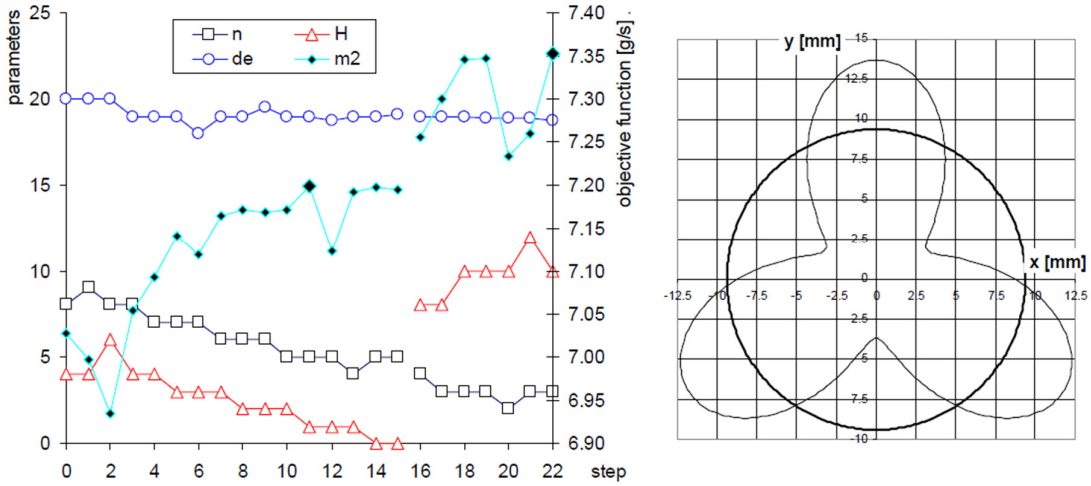


Figure 1.6. The change of optimization parameters and related results, final configuration obtained end of the process [43].

During the optimization process, it was observed that the secondary mass flow rate increases as the lobe number of the nozzle decreases. Unlike the lobe number, it was found that the entrainment ratio increases with the lobe height. The primary nozzle diameter remained nearly constant throughout the optimization process.

In this study, like the Dvorak approach, optimization of a lobed nozzle is performed by changing lobe number and dimensions of nozzle parameters. Unlike Dvorak's analysis, the back pressure at the nozzle is also taken into account in objective function with the entrainment ratio.

1.3 Objective of the Thesis

In the literature, several optimization approaches have been made on the ejector system. However, most of these studies have been done on a tailpipe shape rather than a nozzle geometry. The studies that focus on the optimization of the nozzle of an ejector system are scarce and generally focus on the limited number of

geometrical parameters. In these studies, it is aimed aim to improve either the backpressure condition or the entrainment ratio of the ejector system. In this respect, the main purpose of this thesis is to develop a methodology for the optimization of a helicopter ejector system, which is used in the cooling of the engine bay, with the objectives of increasing the entrainment ratio while decreasing the turbine exit backpressure by changing the geometry of the nozzle and the cone of the ejector system. The differential evolution (DE) method is utilized as the optimization algorithm.

In order to obtain the optimum nozzle geometry, a new geometry should be created and meshed by using the DE method outputs. Then, this geometry needs to be analyzed, and further trials should be determined by using results. This process is continued until the optimum geometry is reached. For this purpose, a Python code is developed that couples different software packages to create and analyze the new geometries.

In the next chapter of this study, the computational tools and the turbulence models used in the study are mentioned. Moreover, the DE method is briefly described. The structure of the developed Python code is explained in the third chapter by giving some examples. After this chapter, the results of the optimization process and the effects of nozzle parameters are shown.

CHAPTER 2

COMPUTATIONAL METHODS AND TOOLS

In this study, an ejector system used for helicopter nacelle cooling is optimized by changing the nozzle shape. Nozzle length, exit diameter, lobe number, and cone exit diameter is selected as optimization parameters. The differential evolution module of Python is chosen as the optimization algorithm. Therefore, all optimization cycle is established in Python. The working scheme of the process is given in Figure 2.1

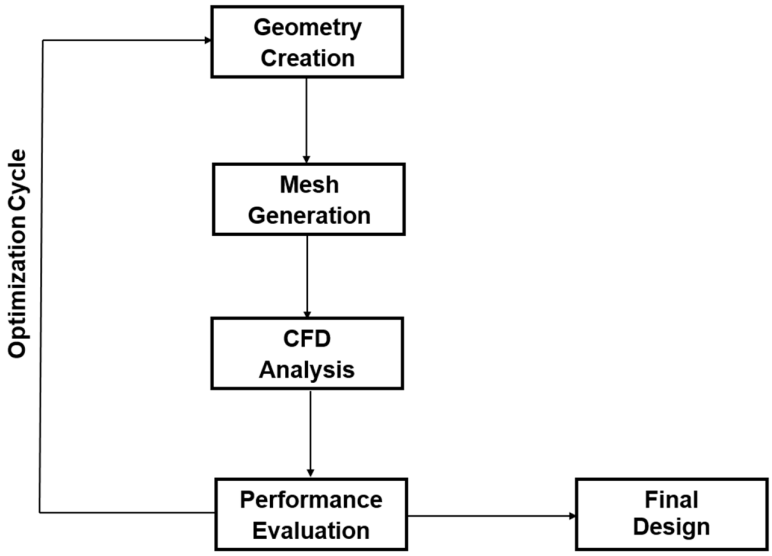


Figure 2.1. Working scheme of the optimization process

Engine compartment geometry is modeled with CATIA, and the mesh generation is made with Pointwise. Computational fluid dynamics (CFD) methods are used to calculate the entrainment ratio and turbine exit backpressure. Widely used commercial software, ANSYS Fluent, is selected for this process. Softwares are

managed by using Python code. These process steps are explained in the following sections in detail.

2.1.1 Geometry Generation with CATIA

Using the Python module, which manages different software applications, and macro commands given by the CATIA developers library [44], a helicopter engine compartment with a lobed nozzle is generated, as shown in Figure 2.2.

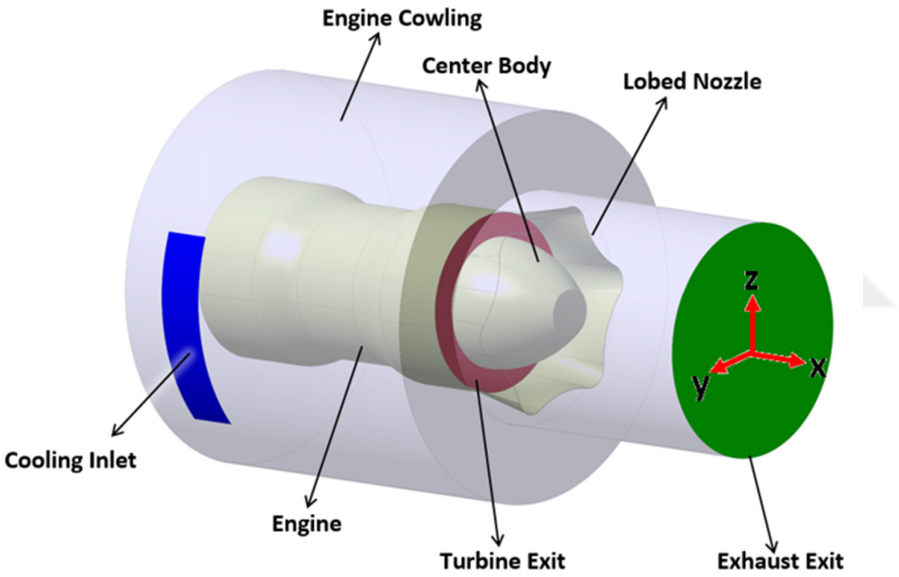
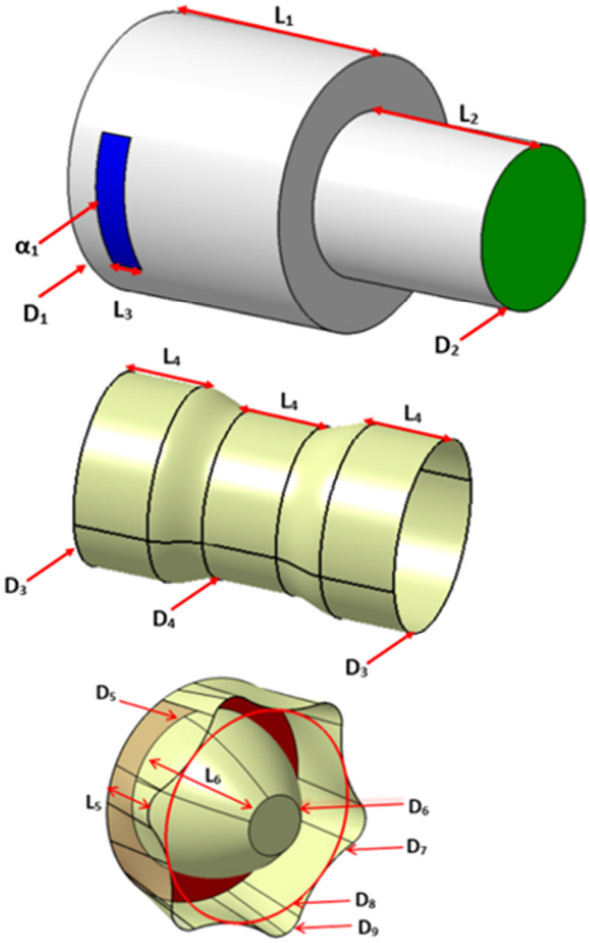


Figure 2.2. Helicopter engine compartment geometry

Dimensions of the engine compartment, engine section, and lobed nozzle used in the analysis are given in Figure 2.3.



L_1	650 mm
L_2	600 mm
L_3	100 mm
L_4	150 mm
L_5	115 mm
L_6	200 mm
D_1	800 mm
D_2	480 mm
D_3	370 mm
D_4	320 mm
D_5	270 mm
D_6	[80 150] mm
D_7	[100 180] mm
D_8	[330 370] mm
D_9	80 mm
α_1	60 °

Figure 2.3. Dimensions of the helicopter engine compartment

The number of lobes, center body exit diameter (D_6), lobe tangent radius (D_7), and nozzle exit diameter (D_8) are set as variables regarding the nozzle geometry in the geometry generation code.

2.2 Grid Generation with Pointwise

Pointwise Software is used to generate the computational grid in the engine compartment due to its ease of automation. Moreover, many sources and examples

can be reached quickly for the automated mesh generation in Pointwise [45]. A cross-section view of the computational grid that is generated in Pointwise is shown in Figure 2.4.

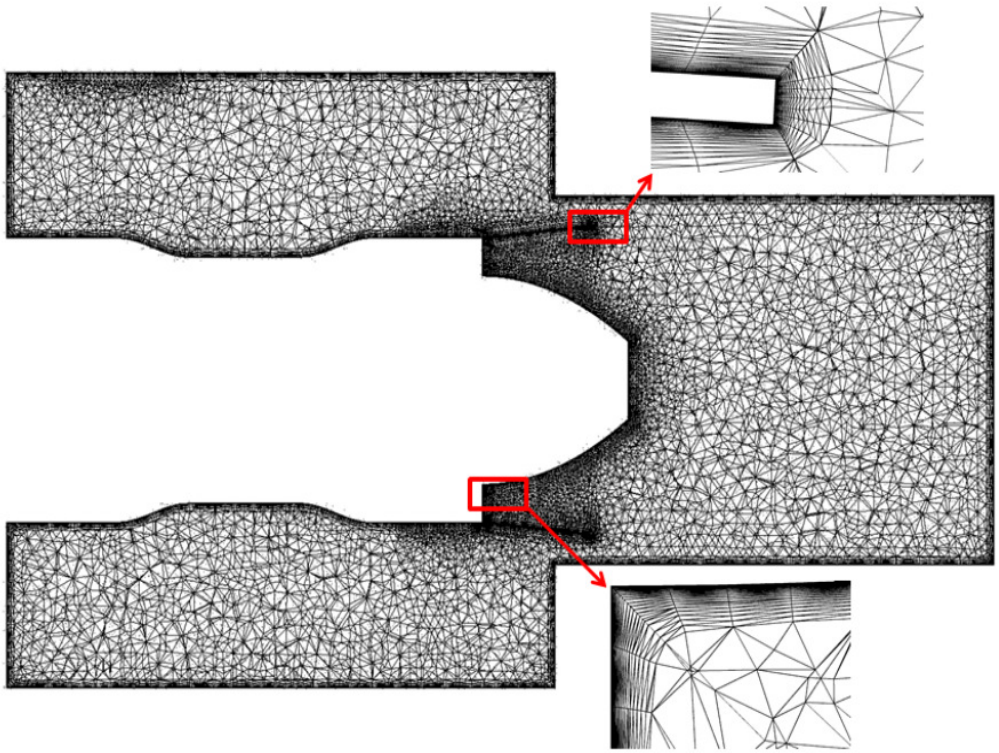


Figure 2.4. A sectional view of the computational grid for analysis

2.3 CFD Analysis with Fluent

ANSYS Fluent is used for the computation of the flow field in the engine compartment. Similar to Pointwise, automation of ANSYS Fluent is straightforward by using its "journal" file. In the solution of engine compartment flow, the k-ε realizable turbulence model is used.

2.3.1 Turbulence Modeling and Simulation

Three main governing equations: continuity, momentum, and energy equations are applied to a fluid element or flow to simulate flow motion. These equations can be solved analytically for some exceptional cases like Couette flow [46]. However, an analytical solution of these equations is not possible for such a complicated problem. To simulate flow for similar topics, these governing equations are numerically solved using some techniques, such as Directly Numerical Solution (DNS) or Large Eddy Simulation (LES) [47]. The most popular and useful of these methods is Reynolds-averaged Navier Stokes (RANS) due to its applicability on different types of problems and low computational cost. The most crucial difference between the RANS and other methods is that averaging of flow variables is performed over time in this model. Thus, flow variables do not depend on time.

2.3.1.1 Reynolds-averaged Navier Stokes (RANS)

As a basic concept of the RANS method, dependent variables are the sum of the mean and fluctuating components, as shown in Eqn. 1.

$$u_i(x,t) = U_i(x) + u_i'(x,t) \quad \textbf{(2)}$$

Here,

$u_i(x,t)$; flow velocity in x-direction,

$U_i(x)$; time-averaged flow velocity in x-direction,

$u_i'(x,t)$; fluctuating part.

When these equations are added to Navier Stokes equations, RANS equations can be shown as with tensor notation.

$$\rho\frac{\partial U_i}{\partial t} + \rho U_j\frac{\partial U_i}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}(2\mu S_{ij} - \rho\overline{u_j{}'u_i{}'}) \quad \textbf{(3)}$$

Where S_{ij} is the mean rate of stress tensor and equals to;

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \quad \textbf{(4)}$$

The term $\rho\overline{u_j{}'u_i{}'}$ is named as Reynolds stress tensor. To solve this fluctuating term, in 1877, Boussinesq assumed that the momentum transfer caused by turbulent eddies could be modeled with an eddy viscosity [48]. Then, the stress tensor can be solved by the following equation;

$$-\rho\overline{u_j{}'u_i{}'} = \mu_T S_{ij} - \frac{2}{3}\delta_{ij}\mu_T\left(\frac{\partial U_k}{\partial x_k} + \rho k\right) \quad \textbf{(5)}$$

Eddy viscosity, μ_T and turbulent kinetic energy, k, are required for the solution of these equations. There are different approaches that give the relation between eddy viscosity and turbulent kinetic energy. Two of the most common of these approaches are k-ε Realizable and k-ω SST models. In the following chapters, these methods are examined briefly.

2.3.2 Realizable k-ε Turbulence Model

B.E. Launder and D.B. Spalding describes the relation between eddy viscosity with turbulent kinetic energy and dissipation as [49];

$$\mu_T = \rho C_\mu \frac{k^2}{\varepsilon} \quad \textbf{(6)}$$

Tsan-Hsing Shih et al. [50] modeled transport equations for k and ε;

$$\rho\frac{\partial k}{\partial t} + \rho u_j \frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\right] + G_k + G_b - \rho\varepsilon - Y_M + S_k \quad \textbf{(7)}$$

$$\rho\frac{\partial \varepsilon}{\partial t} + \rho u_j \frac{\partial \varepsilon}{\partial x_j} = \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_\varepsilon})\frac{\partial \varepsilon}{\partial x_j}\right] + \rho C_1 S_\varepsilon - \rho C_2 \frac{\varepsilon^2}{k + \sqrt{\nu\varepsilon}} + C_{1\varepsilon}\frac{\varepsilon}{k}C_{3\varepsilon}G_b + S_\varepsilon \quad \textbf{(8)}$$

Terms and coefficients are described in the article of Tsan-Hsing Shih et al. [50].

2.3.3 k-ω SST Turbulence Model

F. R. Menter [51] developed a new turbulence model by adding the effect of the transport of the principal turbulent shear stress. The turbulent viscosity is defined as in this method;

$$\mu_T = \frac{k}{\omega} \tag{9}$$

They used the following two equations to solve turbulent viscosity;

$$\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = \mathrm{P_k} - \beta^* k\omega + \frac{\partial}{\partial x_j}\left[(\upsilon + \sigma_k \upsilon_T)\frac{\partial k}{\partial x_j}\right] \tag{10}$$

$$\frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} = \alpha \mathrm{S}^2 - \beta\omega^2 + \frac{\partial}{\partial x_j}\left[(\upsilon + \sigma_{\omega 1}\upsilon_T)\frac{\partial \omega}{\partial x_j}\right] + 2(1 - F_1)\sigma_{\omega 2}\frac{1}{\omega}\ \frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i} \tag{11}$$

Terms and coefficients are described in his article.

2.3.4 Validation of the Turbulence Models

Liu Dawei et al. [52] studied how the nozzle width and position affect the entrainment ratio and mixing of fluids by analyzing different cases. For the CFD simulations, they used the k-ω SST turbulence model validated with an experimental study conducted by Hu et al. [53].

A similar validation study is made for realizable k-ε and k-ω SST models. Standard wall functions are applied near the walls for the realizable k-ε model. Details of this study and a comparison of the experimental and numerical results are given in this section.

2.3.4.1 Validation Case

Hu et al. [53] examined the mixing performance of a lobed nozzle in their experimental study. In this study, jet air was exhausted from the lobed nozzle into a chamber where it mixed with ambient air. Planar laser-induced fluorescence (PLIF) and particle image velocimetry (PIV) measurements were performed in order to assess the mixing characteristics of the nozzle geometry.

In the study of Hu et al. [53], a lobed nozzle that has six lobes was used. Geometrical parameters for this nozzle are given in Figure 2.5. The width of each lobe is 6 mm, and the height of the lobe is 15mm (H=15mm). The inner and outer penetration angles of the lobed structures are 25° and 14°, respectively. The diameter of the lobed nozzle is D=40mm.

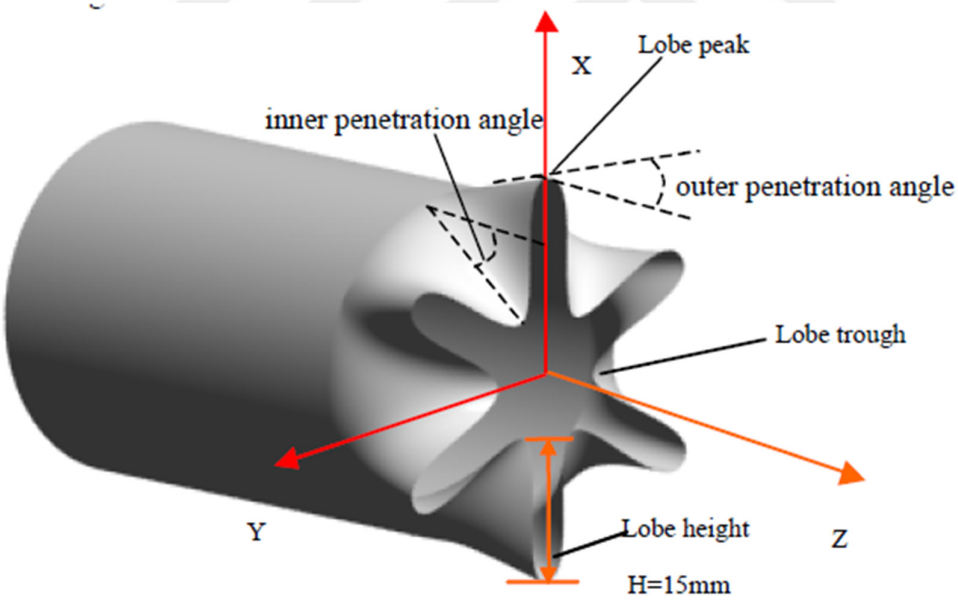


Figure 2.5. Geometrical parameters of the test nozzle

A centrifugal compressor was used to supply jet air from the air tank. Compressed air went through a cylindrical honeycomb chamber to decrease turbulence. Then, the settled airstream was accelerated via a convergent connection. Air velocity at

the nozzle exit can be adjusted to the desired value by a flow meter. In this experiment, that is settled as 20 m/s. Figure 2.6 shows the air supply system that was used in their study.

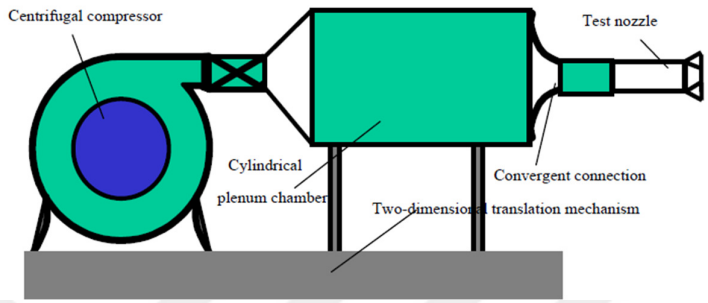


Figure 2.6. Air supply system in the experiment

2.3.4.2 Numerical Setup and Grid Generation

The three-dimensional geometry of the experimental setup [53] is generated by using commercial CAD software CATIA. As shown in Figure 2.7, the nozzle geometry with the reported dimensions is developed in a large control volume generated to obtain stable ambient air.

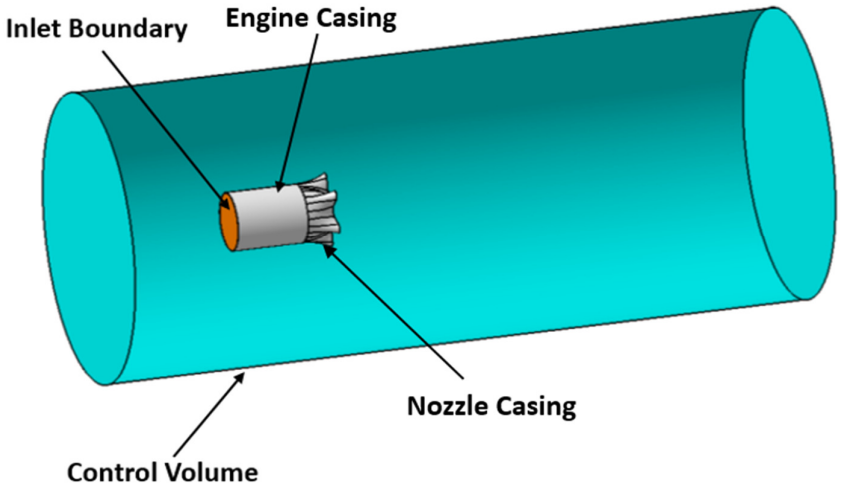


Figure 2.7. Experimental setup geometry used in the analysis

In the analysis, a computational grid made of three blocks and 2.1 million elements is used in the numerical solution of the realizable k-ε model validation case. The three mesh blocks correspond to the nozzle, transition volume, and control volume, as shown in Figure 2.8. For the k-ω SST model, approximately three times the number of realizable k-ε method elements are used with the same structure.

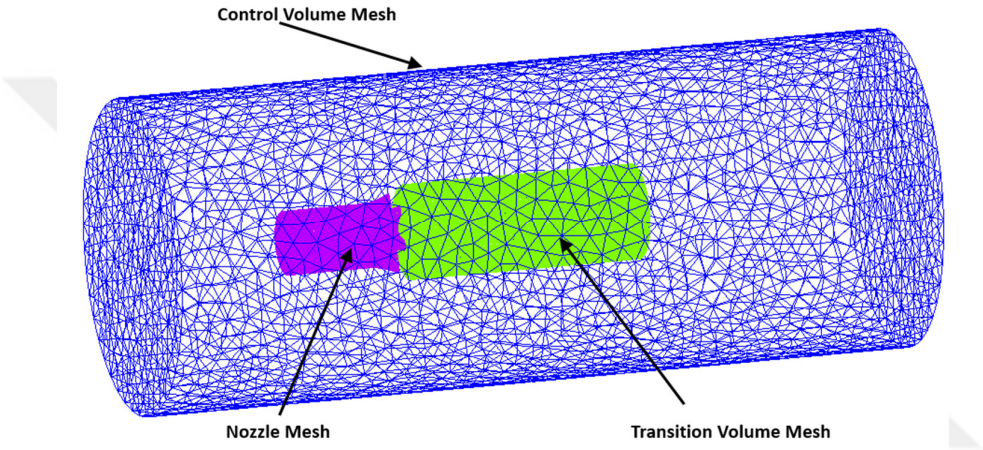


Figure 2.8. Experimental setup grid generation

The first cell height from the surface, called the wall thickness, should be chosen such that the y+ value falls in the range between ~ 30 and ~ 300 for the realizable k-ε model with standard wall function and less than 1 for the k-ω SST model [54]. Thus, the wall thickness from the nozzle inlet surface is adjusted to be 0.2 mm and 0.01 mm for the realizable k-ε and k-ω SST models, respectively. Also, the wall thickness is refined in some critical regions like control volume surface, nozzle wall, or transition volume mesh. Cross-sectional views of the computational grids are shown in Figure 2.9.

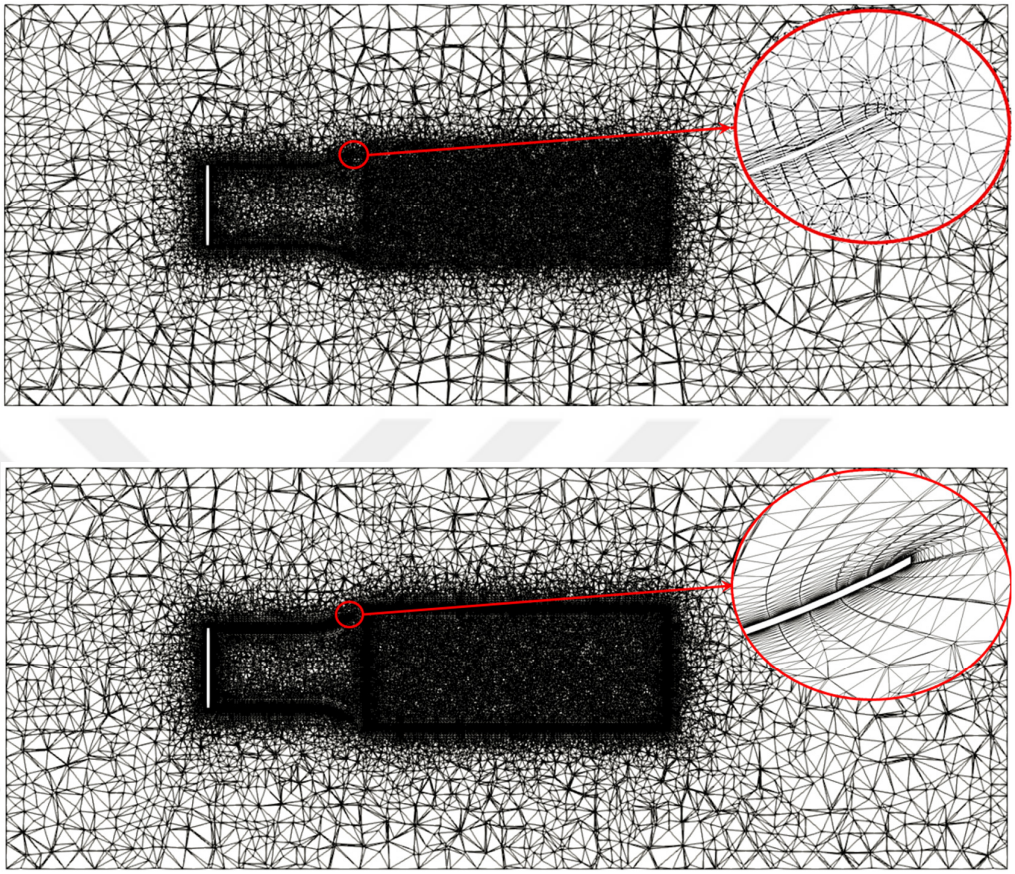


Figure 2.9. The cross-sectional views of grids for k-ε (upper) and k-ω (lower)

2.3.4.3 Numerical Convergence

Residual values of some flow variables are monitored during the solution. Convergence criteria for the flow variables are selected as 10^{-4} and are achieved after about 2250 iterations except for continuity for the realizable k-ε model with standard wall functions. This value is increased to 3250 for the k-ω SST. Although convergence criteria are not achieved for continuity, oscillations are too low for both cases. Residuals of flow variables for the last 250 iterations are shown in Figure 2.10.

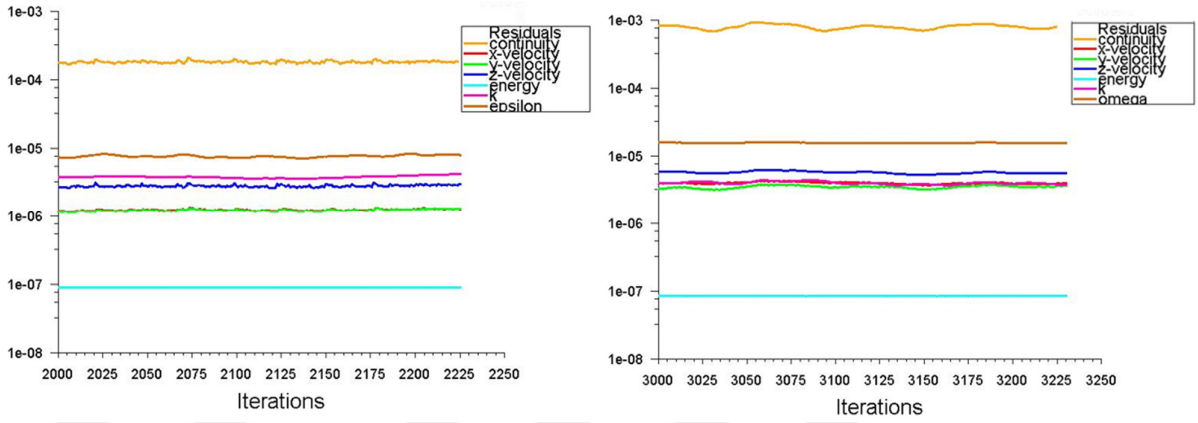


Figure 2.10. Residuals for the realizable k-ε (left) and k-ω SST (right) turbulence models

2.3.4.4 Results of the Validation Case Simulations

The results of the numerical simulations are compared with the experimental data reported by Hu et al. [54]. First, velocity fields at 0.5D, 1.5D, and 2D downstream positions from the nozzle exit are compared in Figure 2.11. It can be seen that numerical results compare well with the experimental data.

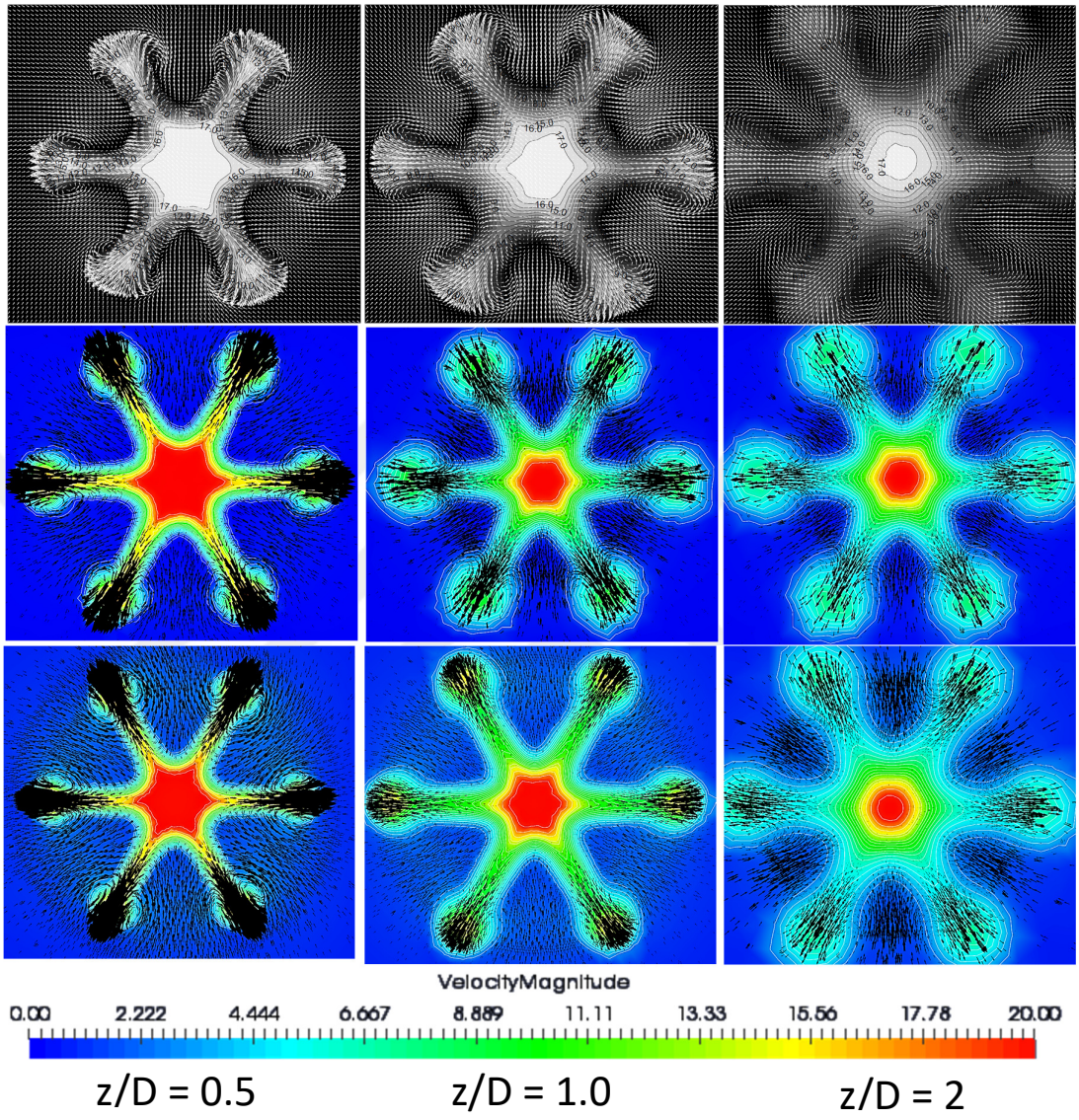


Figure 2.11. Comparisons of experimental data (1st row) with the realizable k-ε (2nd row) and the k-ω SST (3rd row) models in terms of the in-plane velocity vectors and magnitude of the axial velocity (out-of-plane velocity) distributions at different streamwise (z/D) locations

In order to have a more quantitative comparison, maximum azimuthal vorticity levels in cross-stream planes at different streamwise (z/D) positions are compared. The following formula describes the calculation of the azimuthal vorticity:

$$\omega_x = \frac{D}{U_0}\left(\frac{\partial w}{\partial y} - \frac{\partial y}{\partial z}\right) \quad \textbf{(12)}$$

$$\omega_y = \frac{D}{U_0}\left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right) \quad \textbf{(13)}$$

$$\omega_{azimuth} = \sqrt{\omega_x^2 + \omega_y^2} \quad \textbf{(14)}$$

where, U_0 is the nozzle exit velocity as 20 m/s, and D is the nozzle diameter as 40 mm.

As shown in Figure 3.12, the realizable k-ε and k-ω SST models perform very well in terms of maximum azimuthal vorticity. The difference between analysis results and experimental data is less than 5% at all points except the point 0.5D after the nozzle exit.

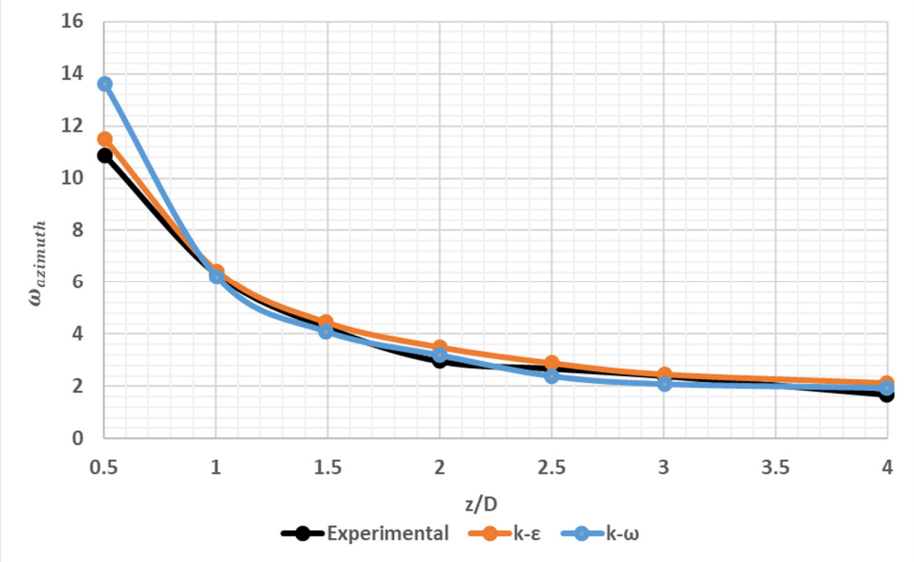


Figure 2.12. The maximum value of the azimuth vorticity of the lobed nozzle for experimental data and analysis

The above results show that the two models give very consistent results with the experimental data. Therefore, both can be used in the optimization process. For this

study, the realizable k-ε method with standard wall functions is chosen due to its relatively lower computational cost.

2.3.5 Grid Convergence

Three different grid structures with changing cell numbers are used to eliminate mesh dependency of the solution. Mesh structures are named as coarse, medium, and fine according to the number of cells. In order to generate different mesh structures, the edge size of cells on surfaces is changed while keeping the first node height constant. For example, the edge size of a cell on engine cowling is set as 30 mm for the coarse mesh and 6 mm for the fine mesh. The number of cells on surfaces and the compartment volume for different meshes are tabulated in Table 2.1.

Table 2.1 Surface and volume mesh numbers for different grid sizes

Mesh Region	*Coarse*	*Medium*	*Fine*	*Very fine*
Surface Cells	170786	192784	243594	309244
Volume Cells	2591976	3398194	4253339	5299978

Simulations for different grid sizes are performed with Ansys Fluent and by using the realizable k-ε turbulence model. The mass flow rate and temperature values at the GE T700 turboshaft engine turbine exit are used in the analysis [55]. Additionally, 10° swirl in the counterclockwise direction when looking from aft to forward is added to the primary flow. Flow conditions at boundary regions are given in Table 2.2.

Table 2.2 Boundary conditions for simulated engine compartment

Boundary Region	*Total Pressure (Pa)*	*Total Temperature (K)*	*Mass Flow Rate (kg/s)*
Turbine Exit Plane	-	884	4.6
Cooling Inlet	95891	289	-
Exhaust Exit	95891	289	-

Numerical convergence of flow variables and objective functions are achieved after approximately 1500 iterations for all computational grids. Variations of the cooling inlet mass flow rate and turbine exit backpressure for the last 250 iterations are shown in Figure 2.13.

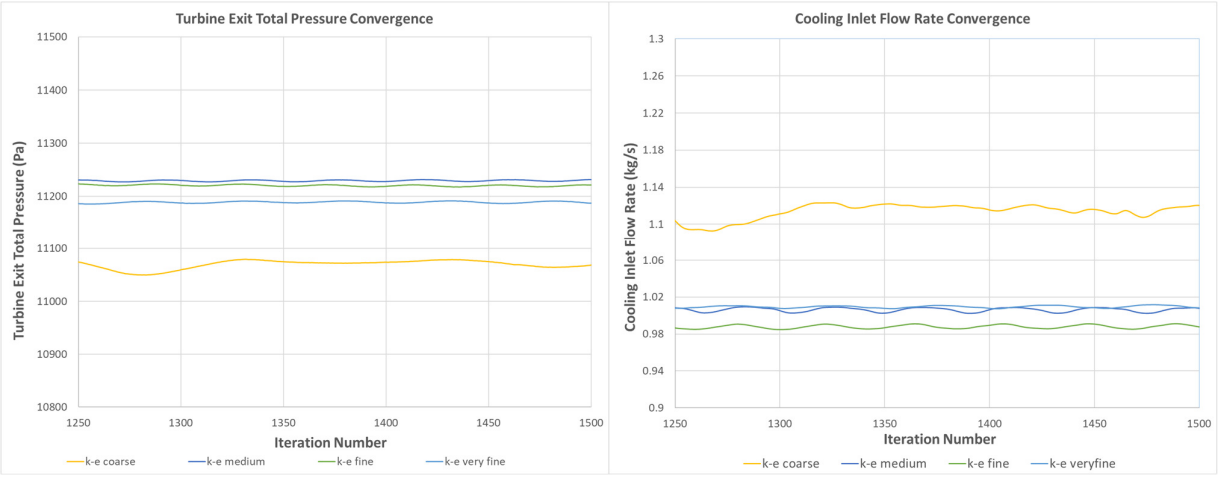


Figure 2.13. Variation of the cooling inlet mass flow rate and turbine exit total pressure (gauge) for the last 250 iterations of the numerical solutions for different computational grids

The differences in the performance parameters obtained with different grid sizes are calculated concerning the fine mesh structure to evaluate better. Calculations are made for an average of the last 50 iterations and given in percentage.

Table 2.3 Differences in the performance parameters of different mesh sizes

Mesh Sizes	*Cooling Flow Rate (%)*	*Turbine Exit Backpressure (%)*
Coarse	10.38	-1.08
Medium	-0.32	0.37
Fine	-2.12	0.28
Very Fine	0	0

It can be seen from Table 2.3 that the average turbine exit backpressures are very close to each other. Therefore, all grid structures can be used for optimization. However, coarse mesh performance is too low in terms of cooling inlet flow rate. It gives more than 10% erroneous results. On the other hand, there is a permissible difference between the medium and the fine mesh results in cooling inlet flow rates. The medium-sized grid structure can be used for all optimization process, considering the results of these two performance criteria

Also, the optimum iteration number should be decided to decrease power and time consumption. Figure 2.16 shows that the turbine exit backpressure and the cooling inlet flow rate values do not significantly change after 1000 iterations. Therefore, the maximum iteration number is set as 1000 for all cases.

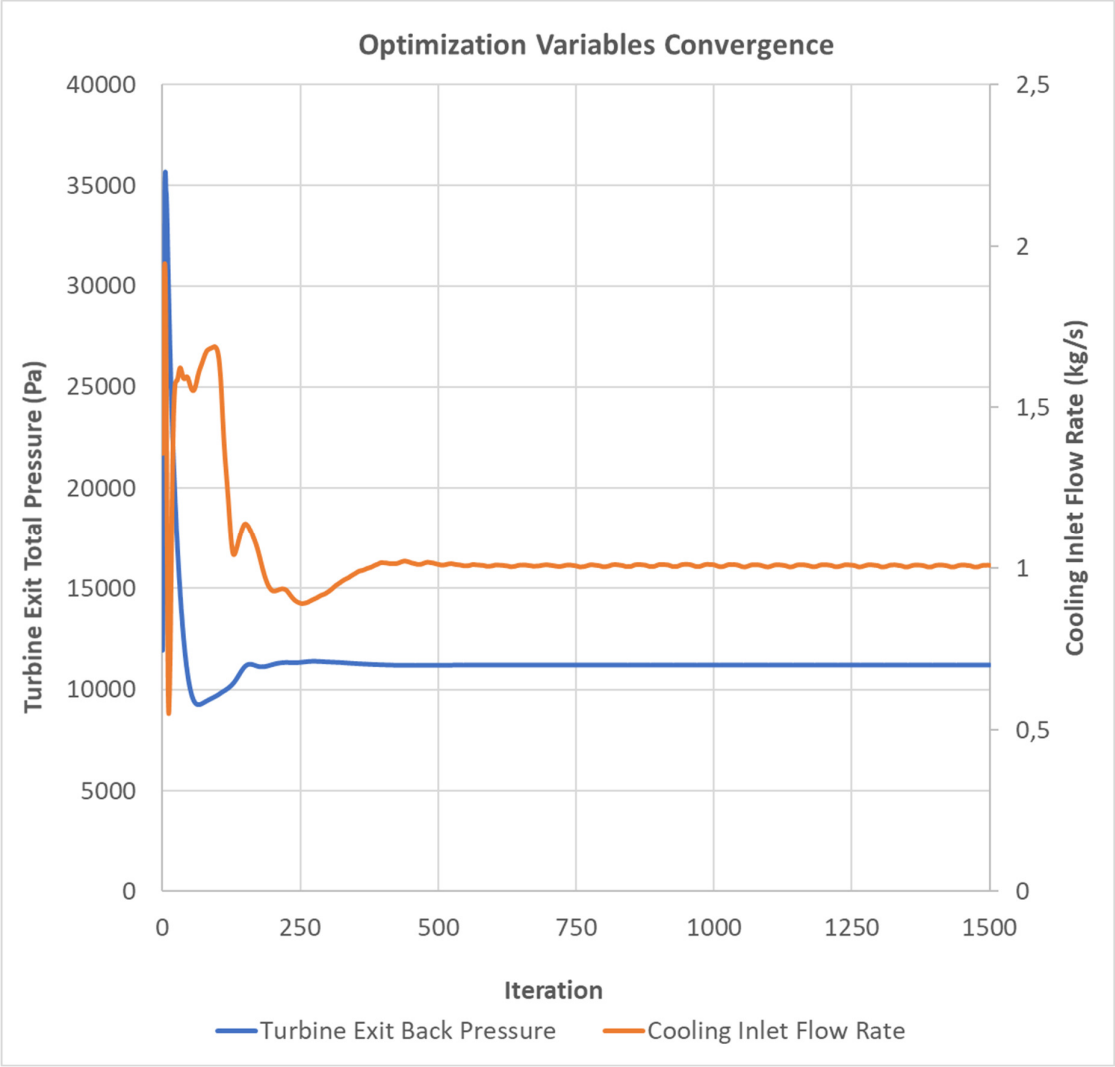


Figure 2.14. Turbine exit backpressure and cooling inlet flow rates for medium size grid structures.

2.4 Computational Mesh Properties

An unstructured tetrahedral mesh structure is used in all optimization processes. Grid size convergence is achieved for the medium-sized mesh structures, as

explained in 2.3.5. The side view of the mesh structure and a sectional view of the critical regions are shown for a six-lobed nozzle in Figure 2.15.

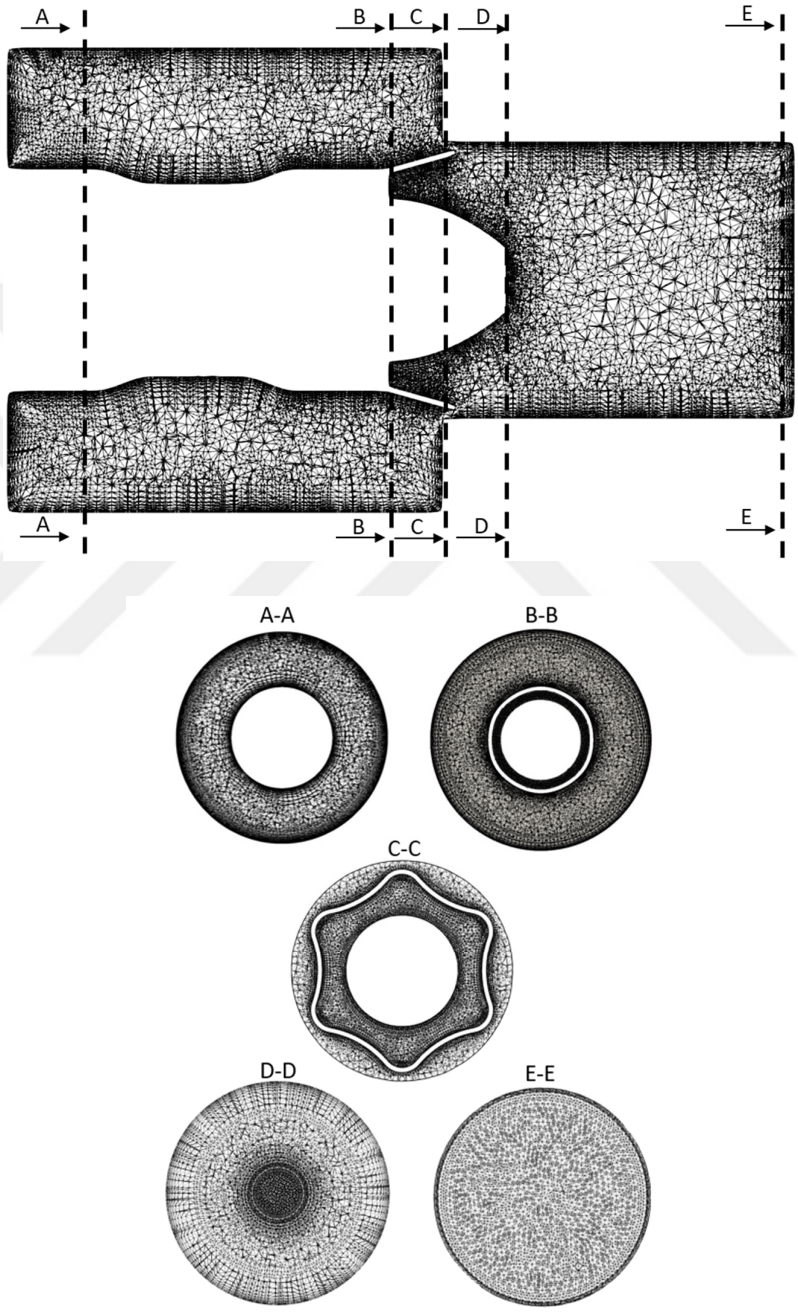


Figure 2.15. Cross-sectional views of the engine compartment critical regions

The sectional view of the mesh structure in the cooling inlet (A-A), the turbine exit (B-B), the lobed-nozzle exit (C-C), the center cone exit (D-D), and the ejector outlet (E-E) regions can be seen in Figure 2.15. The computation grid is refined in these critical regions. The first node height is kept constant as 0.2 mm for all walls to satisfy the y^+ value between 30 and 300. Since different lobed nozzle shapes are generated for further iterations, cell numbers on the nozzle surface are changed. For example, a higher lobe number results in a greater surface area. Therefore, this increases cell numbers on the nozzle surface and in the whole compartment volume. The number of surface cells varies between 114500 to 116400. Similarly, the total cell number varies from 3.4M to 3.9M.

2.5 Hardware Properties

Since the optimization of a system requires a lot of geometry analysis, this should be done with a computer with high CPU and RAM capacity For this purpose, a computer with 4CPU and 16GB RAM is used in all studies.

CHAPTER 3

OPTIMIZATION PROCESS

The optimization scheme of this study is previously described in section 2. In this chapter, the optimization process and Python code are explained in detail. The structure of the optimization code is given in Figure 3.1.

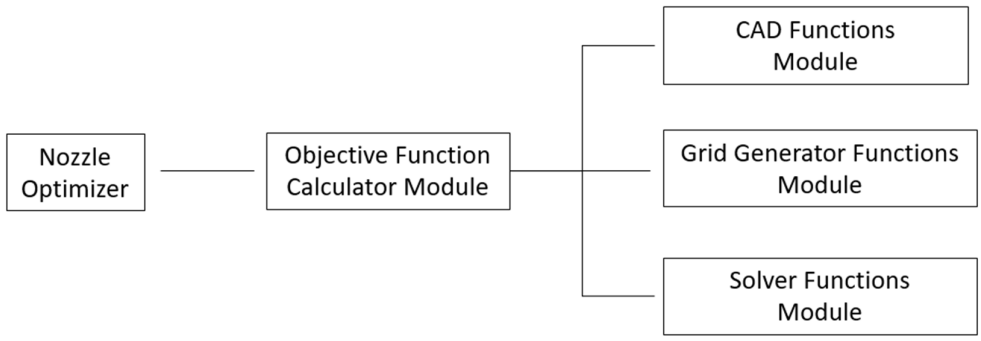


Figure 3.1. An optimization tool written in Python using and combining different software through macros and scripts and an objective function module

As shown in Figure 3.1, the nozzle optimizer module, which includes the DE function, uses an objective function calculator for the optimization. In the objective function calculator module, the engine compartment is modeled using input values defined in the nozzle optimizer. Moreover, mesh generation and analysis of compartments are performed in this module by using three sub-modules. These modules and optimization methods are mentioned in the following sections in detail with examples. The whole code is given in the Appendix B.

3.1 Differential Evolution Method with Python

The DE method developed by Storn and Price [56] is a well-known global optimization method due to its simple usage and accuracy. This method is used in

various optimization applications such as dynamic economic dispatch problems [57] or finding transient stability constrained optimal power flow [58]. Besides, the DE method is widely used in CFD studies in shape optimization. Optimization of high-resolution CFD applications [59] and race car lift [60] are some examples.

The primary phenomenon in the DE method is generating new trial parameter vectors by using randomly selected trials. The vector length is determined by the population size (NP) number. Firstly, initial attempts are randomly generated by the number of NP. According to Storn's study [56], it is suggested that NP should be between three and ten times the number of parameters. In all processes, NP is used as the default value of 15 in the Python module, and it satisfies Storn's requirement.

$$x_{i,G} \text{ where } x \text{ is the target vector and } i = 0, 1, 2, \ldots, \mathrm{NP} - 1. \quad \textbf{(15)}$$

After initial values are determined, objective functions are evaluated with these guesses. Then, new parameter vectors are generated by adding the weighted difference between two of the initial parameters to the target parameter value. These three parameter values are selected from the initial guesses randomly. This step is called mutation and described as;

$$u_{i,G+1} = x_{r_1,G} + F.(x_{r_2,G} - x_{r_3,G}\ ) \quad \textbf{(16)}$$

where u is the mutant vector, x is the parameter vector, and F is a constant weight factor between 0 and 2. The default value of F in the Python module is 0.5, and this value was used in the optimization process. Subscripts r indicates that randomly selected trial values.

It should be checked whether the newly created parameters are within the desired range. If it is not, new random numbers are selected to generate new trials. Figure 3.2 shows a two-dimensional example of this process.

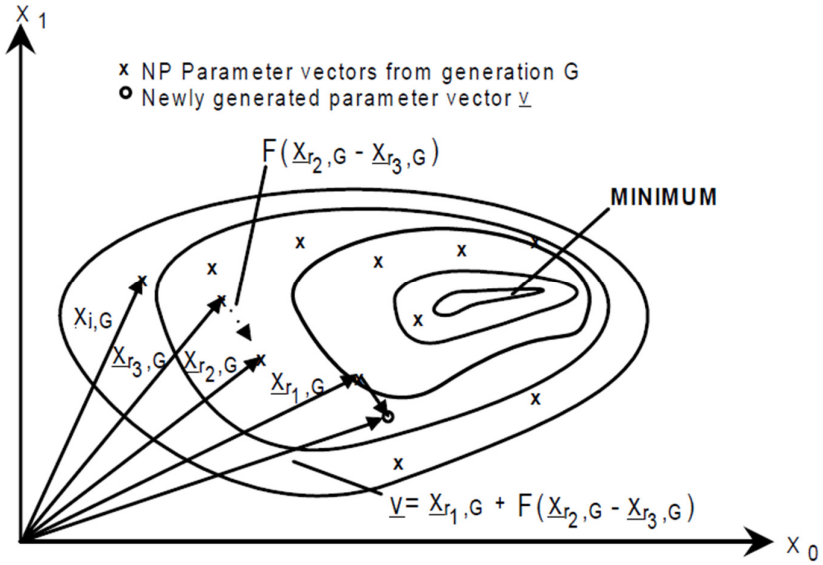


Figure 3.2. Two-dimensional example of a mutation step

The next step in DE is the crossover, determining whether newly generated trials are used or not. This selection is made by comparing the recombination number and a randomly generated number. If the random number is lower than the recombination value, a new parameter generated in the mutation step is used as a trial. The recombination number is adjusted as 0.7 for the optimization study. After newly generated number usage is determined, it should be decided whether the new or old parameters will live in the parameter vector. Therefore, the objective function for newly generated parameters is calculated, and the parameter which gives the lowest objective function is selected for the next iteration vector.

The new parameter set is generated by using an updated parameter set, and this iteration step continues until the success criterion is achieved. In Objective Function Module in the Optimization Tool developed in this thesis, optimization is completed when the following criteria are fulfilled.

$$POF_{std} <= atol + tol * abs(POF_{mean}) \quad \textbf{(17)}$$

where POF_{std} is the standard deviation of the objective function of population, POF_{mean} is average of the objective function of population, $atol$ is absolute tolerance as an input parameter, tol is tolerance as an input parameter. $atol$ and tol are adjusted as 10 and 1, respectively, for this study. Sample optimization by using the DE method is given in the Appendix.

3.2 Nozzle Optimizer Module

Nozzle optimizer module is the main module that controls all other functions. In this module, the input parameters used in the objective function calculator and differential evolution optimizer are defined. Firstly, the ranges of the parameter to be optimized are explained, as shown in Figure 3.3.

```
#####################################################################
#                          AKAY BAYAT                               #
#               METU Aerospace Engineering MS Student               #
#HELICOPTER NOZZLE OPTIMIZING WITH DIFFERENTIAL EVOLUTION METHOD#
#####################################################################

From powerplant_drawing_with_rotate_module import nozzle_solution
from scipy.optimize import differential_evolution
from os import path

#################### PARAMETERS RANGES ##########################

inp=[]
inp.append((5,12))      # Lobed number ranges
inp.append((100,180))   # Nozzle lobed tangent radius ranges in mm
inp.append((330,370))   # Nozzle exit diameter ranges in mm
inp.append((80,150))    # Centerbody exit diameter ranges in mm
```

Figure 3.3. Parameters to be optimized for nozzle and center body

After that, a function to be used in differential evolution method is defined. Boundary conditions, mesh parameters and solver parameters are given in this function.

```
def f(inp):
    lobed_number=round(inp[0],0)
    nozzle_lobed_tangent_radius=round(inp[1],1)
```

```
    nozzle_exit_diameter=round(inp[2],1)
    centerbody_exit_diameter=round(inp[3],1)

    ################## BOUNDARY CONDITIONS ##################

    turbine_exit_mass_flow_rate=4.6
    turbine_exit_temperature=884
    ambient_pressure=95891
    turbine_exit_swirl_deg=10

    ##################### MESH PARAMETERS ####################

    connector_dim=5
    extrude_initial_step_size=0.2
    extrude_growth_rate=1.2
    extrude_layer_number=20
    extrude_skew_angle_limit=0.9
    trex_growth_rate=1.3
    trex_layer_number=10
    trex_skew_limit=0.85
    trex_spacing=2

    ################### SOLVER PARAMETERS ####################

    max_iteration_number=1000
    number_of_processors=4
    turbulence_model="ke-realizable"
    average_iter_number=50

    ########### CALL OBJECTIVE FUNCTION CALCULATOR ###########

    dir_path = path.abspath(path.curdir)
    objective_function_result=objective_function(dir_path,
    lobed_number,nozzle_lobed_tangent_radius,
    nozzle_exit_diameter,centerbody_exit_diameter,
    connector_dim,extrude_initial_step_size,
    extrude_growth_rate,extrude_layer_number,
    extrude_skew_angle_limit,trex_growth_rate,
    trex_layer_number,trex_skew_limit,trex_spacing,
    turbine_exit_mass_flow_rate,turbine_exit_temperature,
    ambient_pressure,turbine_exit_swirl_deg,turbulence_model,
    max_iteration_number,number_of_processors,
    average_iter_number,func_eval)

    ################# PRINTING of RESULTS ####################

    form="%s\t"
    with open("optimization_trials.txt", "a") as txt_file:
        for i in range(len(inp)):
            txt_file.write(form % round(inp[i],0))
        for i in range(len(opt)):
            txt_file.write(form % objective_function_result[i])
        txt_file.write("\n")

    ########### RETURN OBJECTIVE FUNCTION RESULT #############
```

```
    return float(objective_function_result[2])
res =
differential_evolution(f,inp,strategy='best1bin',atol=10,tol=1)
```

Figure 3.4. Boundary conditions, mesh parameters, and solver parameters of the solution

3.3 Objective Function Calculator

The analysis of the engine compartment and the calculation of the objective function are performed in this module. Geometry generation, grid generation, and flow analysis are made, respectively, with the inputs defined in the previous section.

3.3.1 Geometry Generation

Engine compartment geometry is directly generated in CATIA by using the Python "win32com.client" module. In order to decrease the complexity of code, sub-functions are created in the "CAD_functions" module, and these functions are used in the objective function calculator. Some parameters are defined in the primary function and used in all sub-functions. As an example, the drawing of an angled line with code is shown in the below figure.

```
############################################################

#                        AKAY BAYAT                        #
#           METU Aerospace Engineering MS Student          #
#           Objective Function Calculator Module           #
    ############## GENERIC PARAMETERS IN CAD ###############

    global hybridShapes1
    global hybridShapeFactory1
    global part

    CATIA=win32com.client.Dispatch("CATIA.Application")
    CATIA.Visible=True
    part1=CATIA.documents.add("Part").part
    part=CATIA.ActiveDocument.part
    sel=CATIA.ActiveDocument.Selection
```

```
    sel.clear
    body=CATIA.ActiveDocument.part.mainbody
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Add()
    hybridBody1.name="Nozzle"
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapeFactory1 = part1.HybridShapeFactory
    parameters1=part.Parameters
    sel = CATIA.ActiveDocument.Selection
    visPropertySet1 = sel.VisProperties
    visPropertySet1.Parent.SetShow(1)

    ################ REFERENCE PLANES ####################

    xyplane=part1.createreferencefromgeometry(part1.
    originelements.planexy)
    zxplane=part1.createreferencefromgeometry(part1.
    originelements.planezx)
    yzplane=part1.createreferencefromgeometry(part1.
    originelements.planeyz)

    ################ REFERENCE DIRECTIONS ##################
    x = part1.HybridShapeFactory.AddNewDirectionByCoord(1,0,0)
    y = part1.HybridShapeFactory.AddNewDirectionByCoord(0,1,0)
    z = part1.HybridShapeFactory.AddNewDirectionByCoord(0,0,1)

    addlineangle("center_line","nozzle_center",0,500,
    starting_angle,"line1",hybridShapes1,hybridShapeFactory1,
    hybridBody1,part1,yzplane)
```

Figure 3.5. Generic parameters used in CAD functions and an angled line generation

Using sub-function in the example is given in the below figure.

```
############################################################

#                        AKAY BAYAT                        #
#           METU Aerospace Engineering MS Student          #
#                  CAD Functions Nodule                    #


    ################ ANGLED LINE GENERATION ##################

def addlineangle(centerln,centerpt,start,end,angle,lnname,
    hybridShapes1,hybridShapeFactory1,hybridBody1,part1,yzplane):
    hybridShapeLinePtDir1 = hybridShapes1.Item(centerln)
    ref1 = part1.CreateReferenceFromObject(hybridShapeLinePtDir1)
    ref2 = yzplane
    centerpoint = hybridShapes1.Item(centerpt)
    ref3 = part1.CreateReferenceFromObject(centerpoint)
    hybridShapeLineAngle1 = hybridShapeFactory1.AddNewLineAngle \
```

```
    (ref1, ref2, ref3, False,start,end,angle,False)
    hybridShapeLineAngle1.name=lnname
    hybridBody1.AppendHybridShape(hybridShapeLineAngle1)
    part1.InWorkObject = hybridShapeLineAngle1
    part1.Update()
```

Figure 3.6. Angled line sub-function

Similarly, all geometric shapes are generated in the objective function calculator. After geometry generation is completed, the engine compartment is exported in two different “igs” files that are named “lobed_nozzle” and “engine_section.”

3.3.2 Grid Generation

Unlike the generation of the CAD drawing, grid generation software Pointwise is not controlled directly by Python. However, Pointwise offers an option to generate parametric grid structures by a glyph file. A generic glyph file is generated and operated in the Python module. Similar to the CAD generation, Pointwise sub-functions are created in another module to decrease complexity. As an example, importing the engine geometries to Pointwise is given in the following figures. A part of the glyph file generated by using this example is shown in the following figures.

```
##########################################################

#                        AKAY BAYAT                      #
#           METU Aerospace Engineering MS Student        #
#           Objective Function Calculator Module         #
    ############## GLYPH FILE GENERATION #################

    if os.path.isfile("./nacelle.glf"):
        remove(meshing_path)
    os.chdir(solution_path)

    drawing_path_nozzle=solution_path+"\lobed_nozzle_cb.igs"
    drawing_path_engine=solution_path+"\engine_section.igs"

    adddatabase(drawing_path_nozzle,"nacelle.glf")
    adddatabase(drawing_path_engine,"nacelle.glf")
```

Figure 3.7. Importing database to Pointwise in objective function calculator module

```
############################################################
#                        AKAY BAYAT                        #
#            METU Aerospace Engineering MS Student         #
#                Grid Generator Functions                  #
############################################################

global file1
quotes=""'
def openfile(filename):
    global file1
    file1=open(filename,"a")
    return

def adddatabase(drawing_path,filename):
    openfile(filename)
    lines_of_file=["package require PWI_Glyph 2.4\n", \
    "pw::Application setUndoMaximumLevels 5\n",\
    "set var1 [pw::Application begin DatabaseImport]\n", \
    "$var1 initialize -type Automatic" + " {" + \
    drawing_path + "}\n","  $var1 setAttribute  \
    ModelAssembleTolerance 1.0\n"," $var1 setAttribute \
    QuiltAssembleAngle 1.0\n" \
    " $var1 read\n"," $var1 convert\n","$var1 end\n"]
    file1.writelines(lines_of_file)
    file1.close()
    return
```

Figure 3.8. Subfunction for Importing database to Pointwise

```
############################################################
#                        AKAY BAYAT                        #
#            METU Aerospace Engineering MS Student         #
#                Grid Generator GLYPH File                 #
############################################################


package require PWI_Glyph 2.4
pw::Application setUndoMaximumLevels 5
set var1 [pw::Application begin DatabaseImport]
$var1 initialize -type Automatic
{C:\Users\akay\Desktop\Thesis\Full Loop 4
parameter\20200518185901\lobed_nozzle_cb.igs}
  $var1 setAttribute ModelAssembleTolerance 1.0
 $var1 setAttribute QuiltAssembleAngle 1.0
 $var1 read
 $var1 convert
$var1 end
```

Figure 3.9. Glyph file part of database importing to Pointwise

3.3.3 Solver Functions

Similar to the grid generation, a macro file named "journal" is generated for ANSYS Fluent. In this macro file, solver parameters such as boundary conditions or maximum iteration number are defined. The solver parameters, mesh scaling parameters, and the numerical solution method in the solver subfunction is shown in Figure 3.10.

```
############################################################
#                        AKAY BAYAT                        #
#           METU Aerospace Engineering MS Student          #
#                     Solver Functions                     #
############################################################

import math
quotes='"'

def openfile(filename):
    global file1
    file1=open(filename,"a")
    return

def create_journal_file(case_file_path,solution_method, \
    te_wrate,te_temp,te_swirl_deg,sw_ctr,ci_gage, \
    ci_temp,eo_gage,eo_temp,iter_num,filename):

    openfile(filename)
    te_wrate=str(te_wrate) #Turbine exit mass flow rate
    te_temp=str(te_temp)   #Turbine exit temperature
    ci_gage=str(ci_gage)   #Cooling inlet gage pressure
    ci_temp=str(ci_temp)   #Cooling inlet temperature
    eo_temp=str(ci_temp)   #Ejector out temperature
    eo_gage=str(ci_gage)   #Ejector out gage pressure
    sw_ctr=str(sw_ctr)     #Swirl center location
    te_swirl=str(round(math.tan(te_swirl_deg* \
    math.pi/180),2))       #Turbine exit swirl degree
    iter_num=str(iter_num) #Maximum iteration number

    lines_of_file=["/file read-case " + quotes +  \
    case_file_path +quotes+ "\n", \
    "/file/set-batch-options yes yes no\n", \
    "/mesh/scale 0.001 0.001 0.001\n","/define/models/" +
    "/define/models/viscous/"+ solution_method
```

Figure 3.10. Writing of solver parameters, mesh scaling, and numerical solution method in solver subfunction.

3.4 Objective Function

As previously mentioned, the primary purpose of an ejector system is the cooling of a designated region. Therefore, increasing the secondary flow rate is the main purpose of the optimization process. In this study, the base objective function is set such that the cooling airflow rate is a more dominant factor than the turbine outlet backpressure. As the nature of an optimization process, the objective function must decrease when the cooling airflow rate increases and the turbine exit backpressure decreases. Therefore, the following formula is used as the cost function;

$$F_{obj} = [(\dot{m}_{te}/\dot{m}_{ci}) + (P_{te} + P_{amb})/P_{amb}] \times 100 \quad \textbf{(18)}$$

where;

$\dot{m}_{te}$ is the turbine exit mass flow rate and 4.6 kg/s,

$\dot{m}_{ci}$ is the cooling inlet mass flow rate,

P_{te} is turbine exit backpressure,

P_{amb} is ambient pressure and 95891 Pa.

CHAPTER 4

RESULTS AND DISCUSSION

In order to ensure the performance of the differential evolution optimization method in nozzle design, two different optimizations are performed with different objective functions for three parameters. In general, the objective is to decrease the turbine exit backpressure and increase the cooling mass flow rate. In the first optimization approach, the base cost function described in Section 3.4 is used. Then, the effect of the turbine backpressure on the cost function is increased, and final parameter values are investigated. After that, the nozzle optimization is repeated with four optimization variables by using the base cost function.

4.1 Input Parameters

Dimensions of the geometry and the boundary conditions are given in Figure 2.3 and Table 2.2, respectively. In this study, the nozzle lobe number, the nozzle exit diameter, the lobe tangent radius, and the center cone exit diameter are considered as the optimization variables.

Following ranges of variables are investigated in this study;

- n_l Lobe Number → from 5 to 12,
- D_8 Nozzle Exit Diameter → from 330 to 370,
- D_7 Lobe Tangent Radius → from 100 to 180,
- D_6 Cone Exit Diameter → from 80 to 150.

The differential evolution method creates new parameter values by using random numbers and weight factors. Therefore, the generated parameter values may not always be integers. However, since the number of lobes of the nozzle must be an integer, the generated parameter values are rounded to integer values before the calculation of the objective function. Similarly, values of the other optimization variables are rounded to one decimal number. Samples of this rounding operation are given in Table 4.1.

Table 4.1 Samples of parameter value round operation

Parameter Name	*DE Calculation*	*Rounded Value*
Lobe number	5.187956	5
Nozzle Exit Diameter [mm]	350.875258	350.9
Lobe Tangent Radius [mm]	140.145368	140.1
Cone Exit Diameter [mm]	90.875965	90.9

4.1.1 Flow Field in the Ejector System

In this chapter, the general flow topology of the ejector system is described for a sample case. For this purpose, the flow regime in the ejector system with an 8-lobed nozzle is simulated. The turbine exit diameter, the lobed tangent radius, and the cone exit diameter are 330 mm, 160 mm, and 100 mm, respectively. First, to illustrate the flow field, primary and secondary flow regions are shown, taking a cross-sectional view of the flow in Figure 4.1.

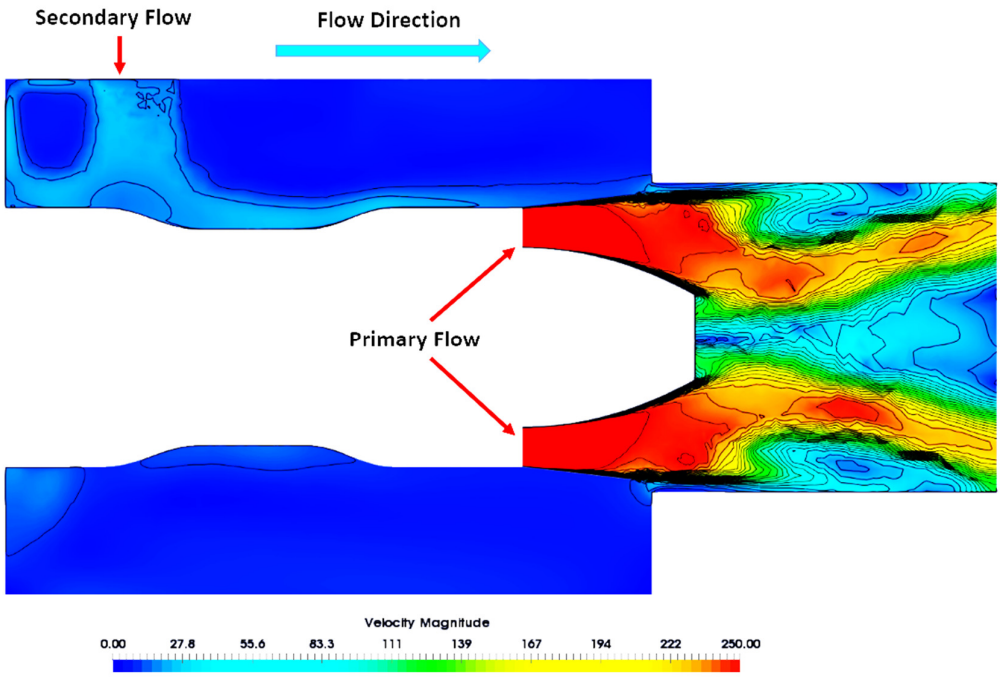


Figure 4.1. Contours of velocity magnitude of three components plotted on xz plane for the sampe solution

As seen in the above figure, the primary flow is mixed with the secondary flow and ejected outside. In order to display the general flow behavior in the system more clearly, velocity vectors are added to the cross-sectional view (Figure 4.2). Instead of the velocity magnitude, onlycontours of x-velocity is used in the cross-sectional view to understand the direction of the flow more easily.

As shown in Figure 4.2 A, recirculation zones are present in the engine bay, especially near the secondary flow inlet. Since the engine always generates heat, the temperature of the recirculated flows increases. Therefore, recirculations in the engine bay are not desired. However, these recirculation zones often occur in the engine's cold areas, and this situation is feasible.

In the region between the nozzle exit and ejector wall, there is a high-velocity gradient present. Due to the swirling primary flow of the nozzle, the flow passage

for the secondary flow is reduced. As a result, the secondary flow accelerates, which is also evident from Figure 4.2 B.

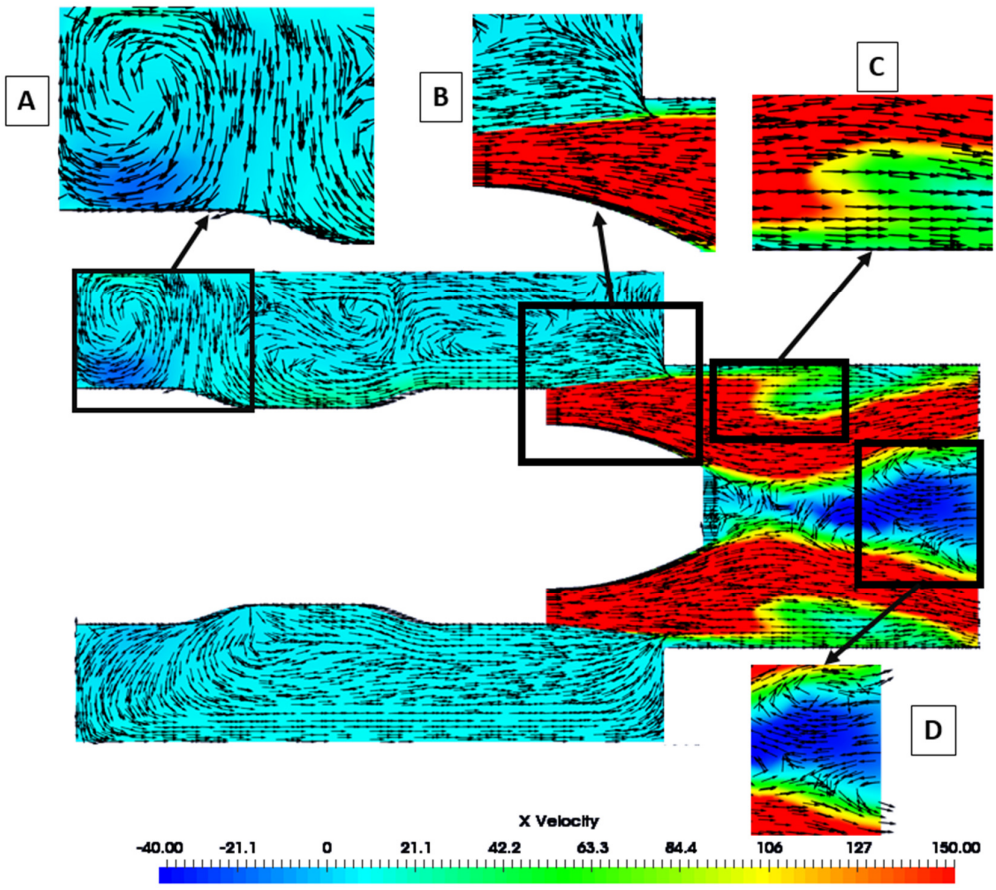


Figure 4.2. Contours of axial velocity (x-velocity) with vector fields plotted on xz plane for the sample solution

From Figure 4.2 C, it is seen that there is a sudden velocity drop after nozzle exit. The main reasons for this situation are the primary flow swirl and mixing of flows. The primary flow starts to mix with the secondary flow after nozzle exit due to the swirl effect. This mixing is not uniform for all nozzle regions due to the lobe effect. Since mixing flow rotates along with the ejector, this causes the mentioned

situation in cross-sectional view. It can be seen clearly by taking a cross-stream view of the flow in the xz and yz planes with Figure 4.3.

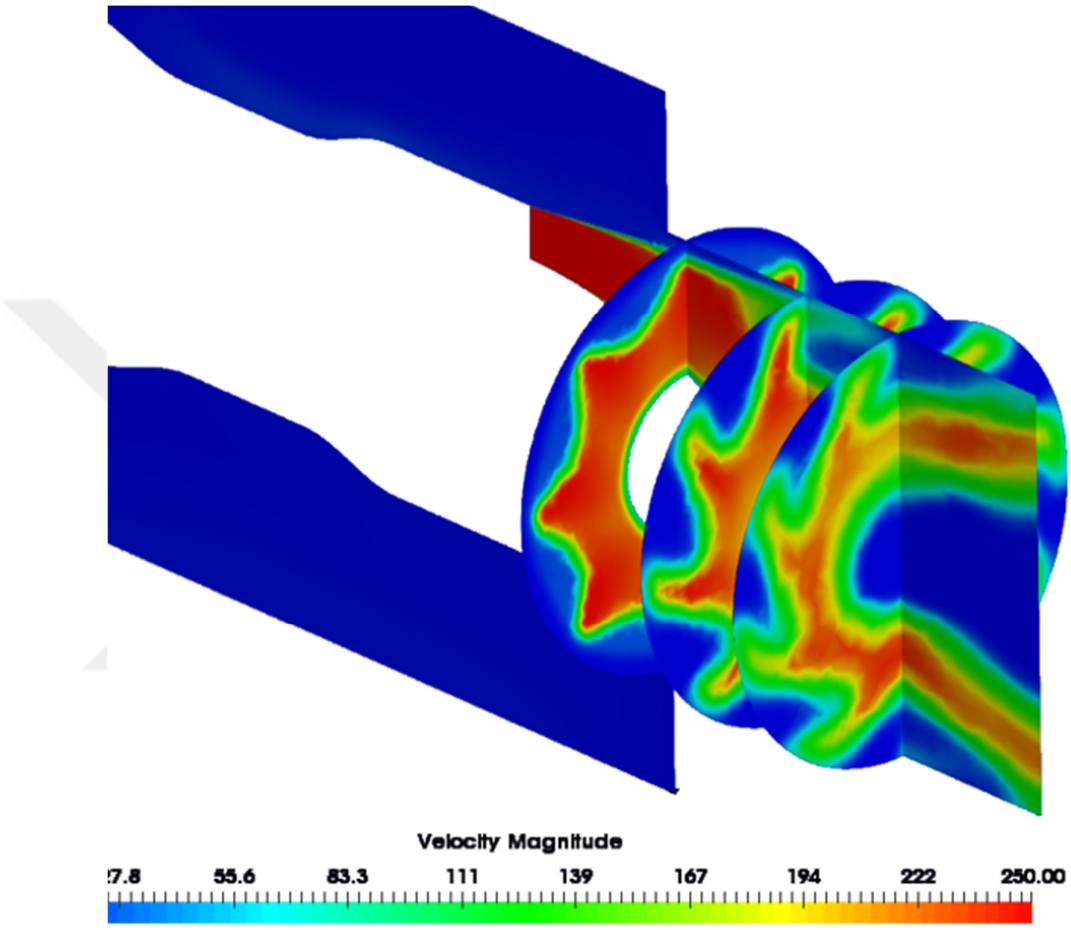


Figure 4.3. Contours of velocity magnitude of three components plotted in xz and a number of yz planes downstream of the nozzle exit for the sample solution

It can be easily observed that recirculation zones occur at the exit of the ejector since the swirl effect on primary flow is not eliminated (Figure 4.2 D). By changing the nozzle and the center cone geometries, these size of these zones can be decreased. In addition to these recirculation zones, it is expected that similar flow behaviors should occur at the near center body outlet. However, this situation cannot be observed clearly from Figure 4.2. In order to make a better interpretationof the flow behavior, streamlines plots are generated in these regions.

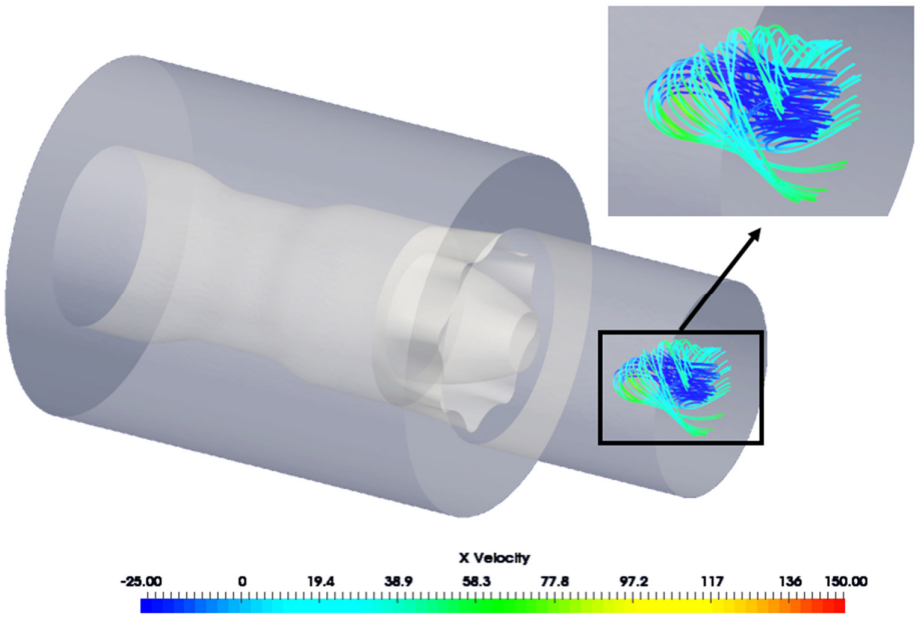


Figure 4.4. Streamlines generated from the center of exit zone for the sample solution

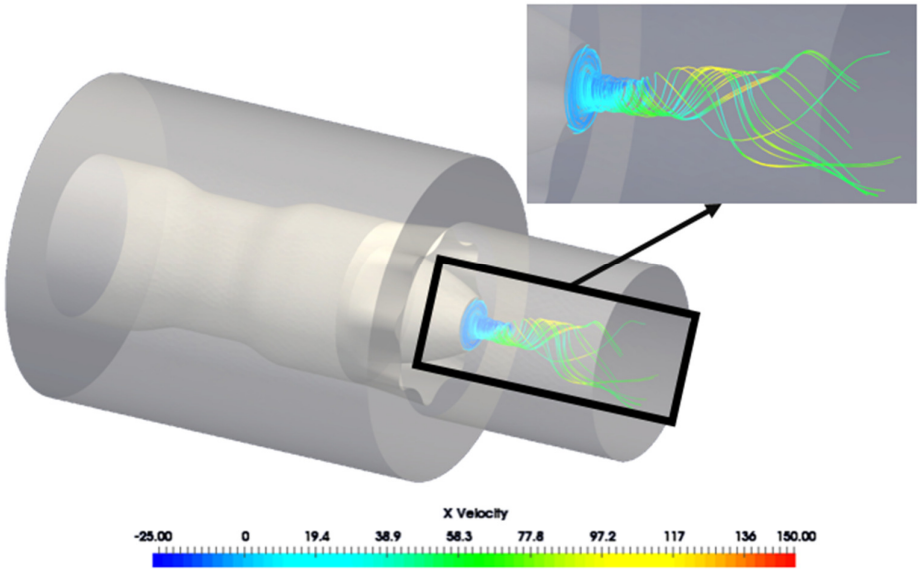


Figure 4.5. Streamlines generated from the center body outlet for the sample solution

It is seen that recirculation occurs at the middle region of the ejector exit. Ambient air goes through for about one-third of the ejector. It mixes with the primary and secondary flows and it is thrown out from the side of the ejector exit. Recirculations behind the center body can be observed easily from Figure 4.5. This torus is created due to the swirl-induced radial pressure gradient balancing the centrifugal forces [61]. In addition to recirculation zones, primary and secondary flow streamlines are generated to understand flow physics (Figure 4.6).

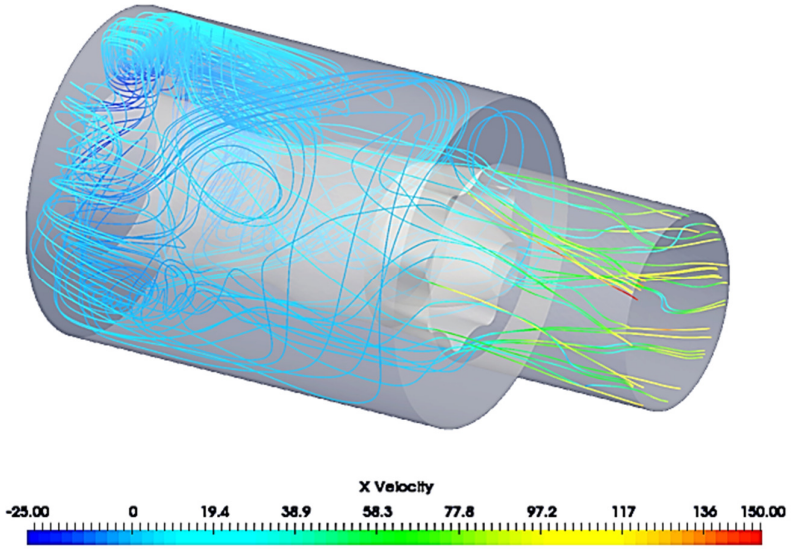


Figure 4.6. Streamlines generated from the secondary (cooling) inlet for the sample solution

It is clear that the nozzle flow is sufficient to entrain the cooling flow; besides, the cooling flow circulates the engine bay, which reduces the risk of local temperature increase.

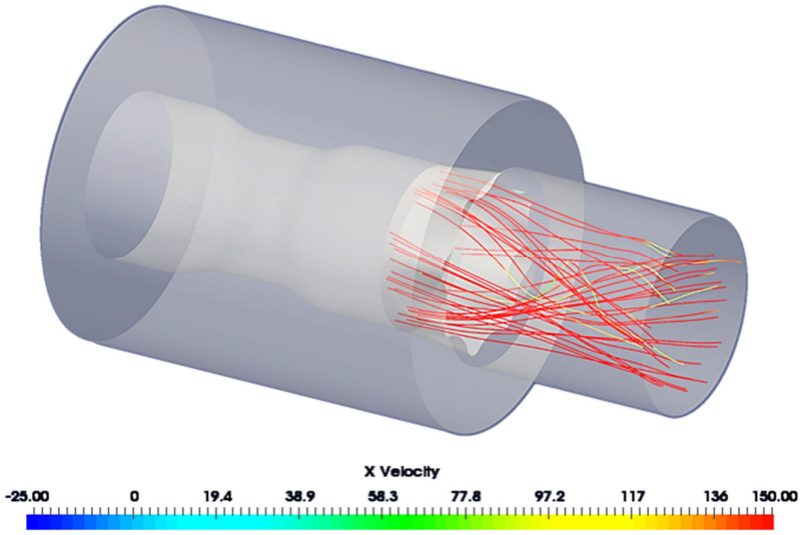


Figure 4.7. Streamlines generated from the primary (nozzle) inlet for the sample solution

Primary (nozzle) flow streamlines are also generated with a similar approach (Figure 4.7). The swirl component can be observed from these streamlines.

4.1.2 Optimization Results for Three Nozzle Variables

Optimization of the lobe number, the lobe tangent radius, and the nozzle exit diameter is performed in this section. These parameters are optimized with two different cost function definitions. In the first case, the base objective function defined in 3.4 is used. This process is repeated with another objective function in which the weight of the backpressure is increased. In the three-variable optimization processes, the cone exit diameter is defined as 100 mm.

4.1.2.1 Effects of the Optimization Variables on the Flow Physics with the Base Function

Optimization of the lobe number, the nozzle exit diameter, and the lobe tangent radius is achieved for the base objective function. The optimization process is completed in 90 iterations. In this section, the effects of varying optimization variables on the cooling mass flow rate and the turbine backpressure are investigated. Moreover, results of the best and worst cases are given together with some flow field characteristics.

Although it is often difficult to distinguish between the effects of optimization variables, some general trends may indicate the influence of each variable on flow physics. For example, configurations with lower lobe numbers generally show better performance, as shown in Table 4.2. The nozzle exit area decreases with the decrease of lobe number while other parameters are kept constant. This area reduction helps to eliminate the swirl effect and raises the cooling inlet flow rate. On the other hand, the turbine exit backpressure is negatively affected.

Table 4.2 Results for different lobe numbers for the base cost function solution

Case	n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$	F_{obj}
1	42	7	114.3	337.7	1.58	6787	397.9
	87	9	114.3	337.7	1.47	6094	419.4
2	60	5	136.2	366.6	1.83	9213	361.5
	13	9	134.4	366.6	1.26	6169	472.9

Similar to the lobe number effect section, the results where only the nozzle outlet diameter changes are given below.

Table 4.3 Results for different nozzle exit diameters for the base cost function solution

Case	n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$	F_{obj}
1	24	5	178.7	332.7	1.51	9362	414.4
	69	5	178.7	366.5	1.70	8393	378.7
2	32	8	166.8	340.2	1.49	6277	415.3
	77	8	166.8	354.2	1.51	6145	411.0
3	25	11	106.9	335.1	1.34	5768	450.1
	70	11	106.9	355.1	1.21	6056	485.1

As can be seen from the results, there is a direct relationship between the nozzle outlet diameter and the performance of the system. Unlike the previous case, the increase of the exit diameter affects system performance positively, although the nozzle exit area increases. However, this expression is not valid for the 25th and 70th iterations in the turbine exit backpressure. For this case, it is seen that the lobe number is about the maximum, and the lobe tangent radius is close to the minimum value in the range. Therefore, the nozzle exit shape is similar to a circle, and the swirl value is high at the nozzle exit. For this type of geometry, the increase of the exit diameter adversely affects the swirl elimination. Thus, the primary flow expands and blocks some parts of the secondary flow. This condition influences the system efficiency adversely.

A similar procedure is applied to see the effects of lobe tangent radius. The increase of the lobe tangent radius adversely affects the cooling inlet flow rate, as shown in Table 4.4. The reason is that the lobe depth decreases with the increase of the tangent radius, and flow swirl cannot be eliminated effectively. On the other hand, there is no net effect in terms of turbine exit backpressure.

Table 4.4 Results for different lobe tangent radius values for the base cost function solution

Case	n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$	F_{obj}
1	4	7	118.2	362.7	1.59	6387	395.6
	49	7	178.8	362.7	1.51	6216	411.1
2	5	11	142.4	343.2	1.27	5919	469.7
	50	11	155.6	343.2	1.13	7233	513.7

4.1.2.1.1 Comparison of the Best and Worst Configurations

Qualitative and quantitative comparisons of the best and worst cases in terms of the objective function are given in this section. The geometries of the best and worst geometries are presented in Figure 4.8.

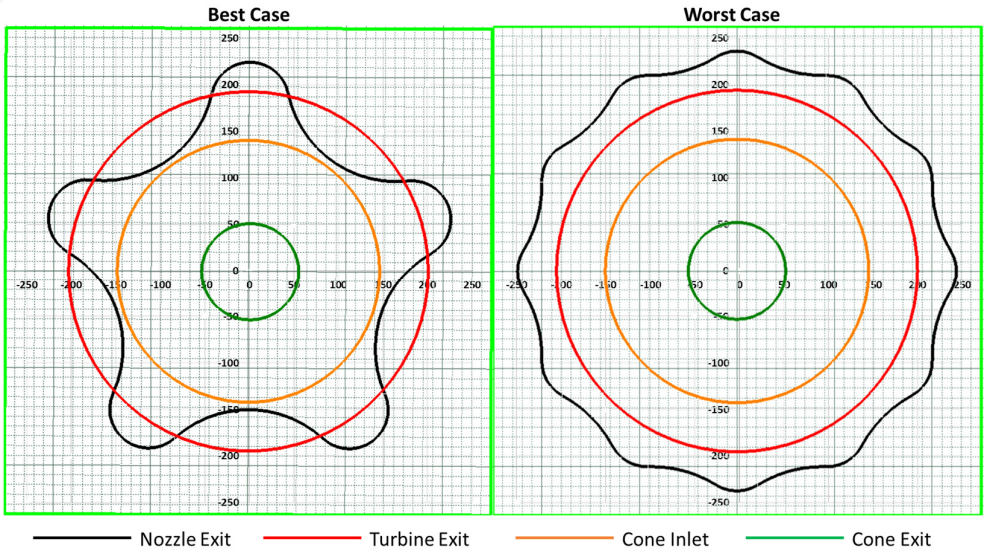


Figure 4.8. The geometries for the best and worst cases of the base cost function solution

From Figure 4.8, it can be seen that the lobe shapes are more evident in the best case. On the other hand, the nozzle exit shape is almost a circle for the worst case. Parameters for the best and the worst configurations are compared in Table 4.5

Table 4.5 Parameters and results of the best and worst configurations for the base cost function solution

Case	n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$
62	5	113.5	351.2	1.90	10856	353.7
61	12	125.6	369.8	1.12	6822	516.9

There is an improvement of approximately 70% in the amount of cooling air compared to the worst case in the best configuration. However, the turbine exit backpressure is 60% higher. The best geometry is obtained for the minimum number of lobes, i.e., five lobes. On the other hand, the worst performance is observed for the maximum lobe number, i.e., 12, while the nozzle outlet diameter is approximately at the maximum value of the given range for the worst case. Similar to the nozzle exit diameter, the lobe tangent radius is smaller for the best configuration.

To compare the flow fields for the two cases, contours of x-velocity, static temperature, and static pressure on yz plane of the critical regions are plotted in Fig. 4.9 – 4.12.

The axial velocity distribution of cold regions from the cooling inlet to the turbine exit region is plotted in Figure 4.9.

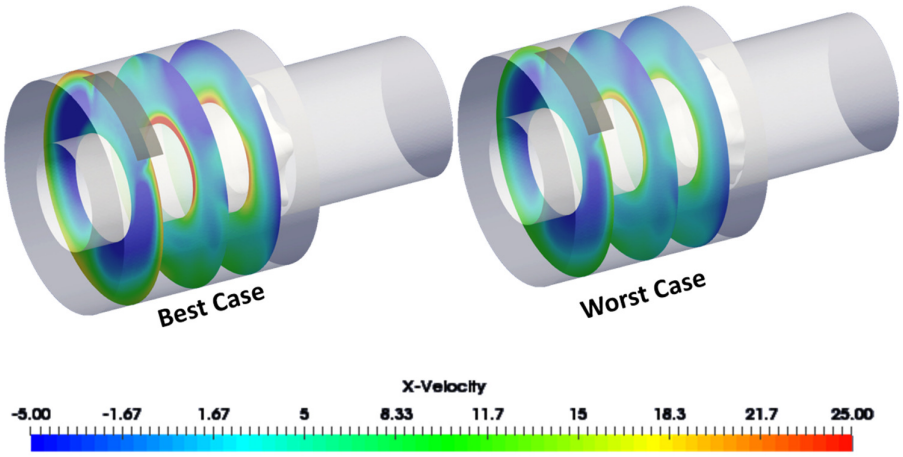


Figure 4.9. Velocity distributions for cold regions for the best and worst cases of the base cost function solution

The axial velocity of secondary flow is higher in the best configuration asexpected due to the increased mass flow rate. Also, the cooling flow is significant throughout the engine case in the best design. Thus, all engine parts are cooled more effectively.

In order to observe the effect of the nozzle parameters, contours of x-velocity on yz planes at x/L_2 =0.05, 0.4, and 1 (ejector exit) after the turbine exit plane are given in Figure 4.10.

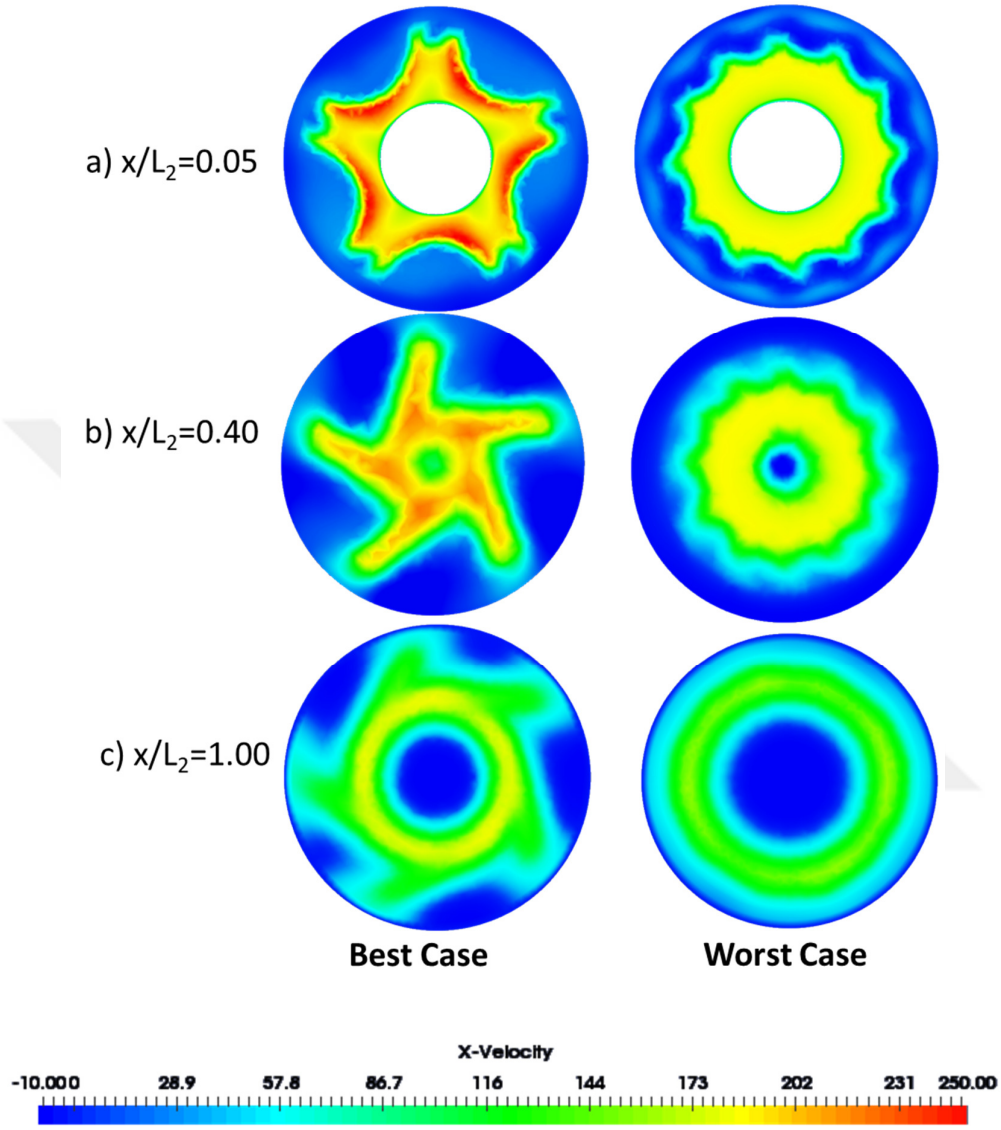


Figure 4.10. Contours of axial velocity (x-velocity) plotted on yz planes at different downstream positions from the nozzle exit for the best and worst cases of the base cost function solution

It can be observed that velocity magnitudes in the axial direction are higher for the best case due to low nozzle exit area and more cooling inlet flow. On the other hand, mixing efficiency seems to be high for the worst configuration. It is considered that the reason for this is the swirl of primary flow could not be

eliminated with the worst nozzle. This facilitates the mixing of primary and secondary flows. However, this also causes the formation of a prominent recirculation zone present at the center of the exit region.

The mixing efficiencies of the cases are numerically compared in terms of the maximum azimuthal vorticities at the ejector exit as in the validation case. The maximum azimuthal vorticities are calculated by using the below formula:

$$w_{azi} = \frac{D_{te}}{U_{te}} \max\left(\sqrt{\left[\left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)^2\right]}\right) \quad \textbf{(19)}$$

where;

D_{te} is the diameter of turbine exit which is 370 mm and

U_{te} is the primary flow velocity at the turbine exit by using ambient pressure (95481 Pa) and 245 m/s.

The maximum azimuthal vorticities of the best and worst cases are 6.65 and 4.66, respectively, and this supports that the mixing is better in the worst case. In order to obtain more detailed information, the temperature variations for hot regions are studied and presented in Figure 4.11.

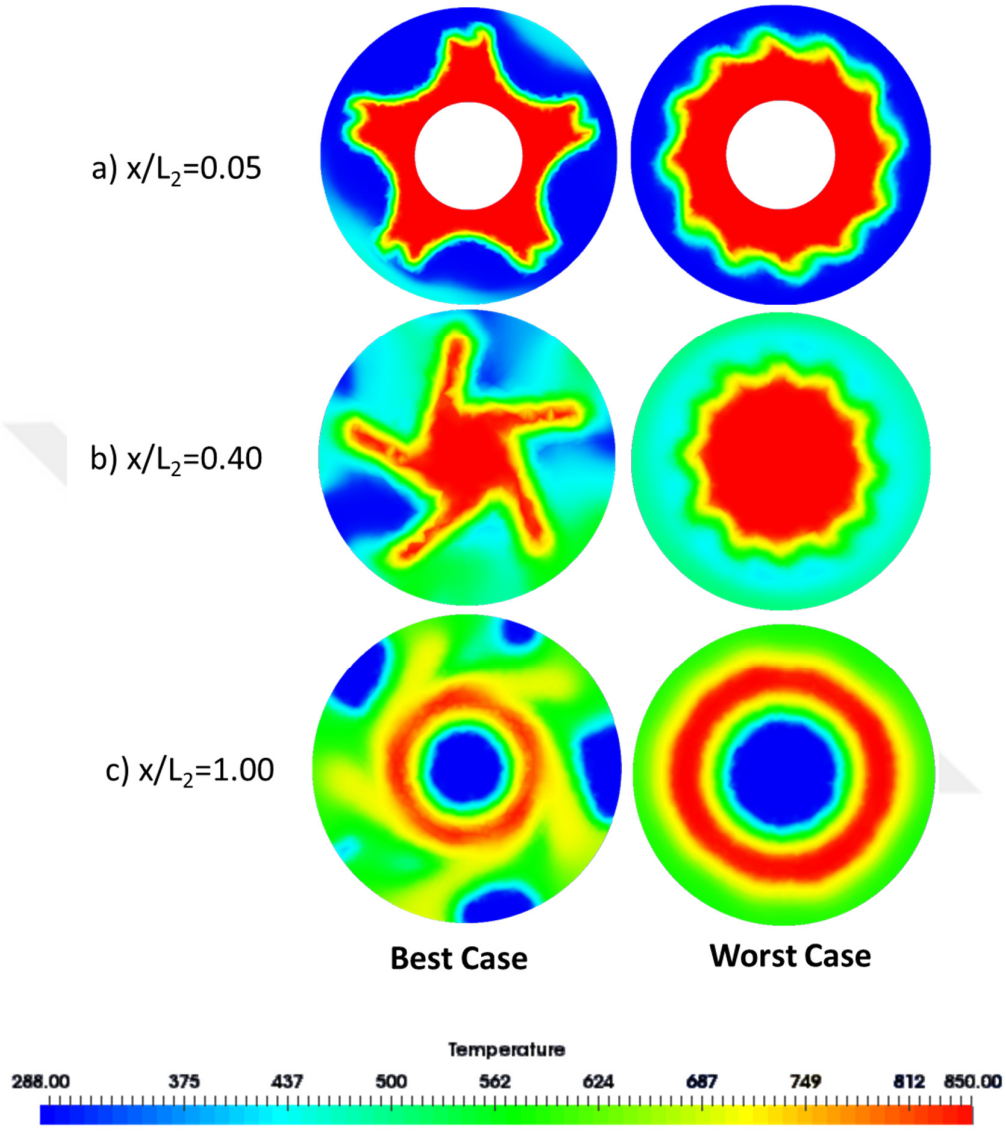


Figure 4.11.

Contours of temperature plotted on yz planes at different downstream positions from the nozzle exit for the best and worst cases of the base cost function solution

Temperature values are lower for all sections of the best case due to the higher cooling mass flow rate. Similar to the velocity profiles, effects of recirculation zones can be observed at the middle regions of ejector exits.

Pressure distribution of different nozzle types is also studied, and related contours are generated within the region near the turbine exit plane. As previously mentioned, turbine exit backpressure is higher for the best configuration due to the low nozzle exit area. Also, the effect of the lobe structure is more visible for this configuration.

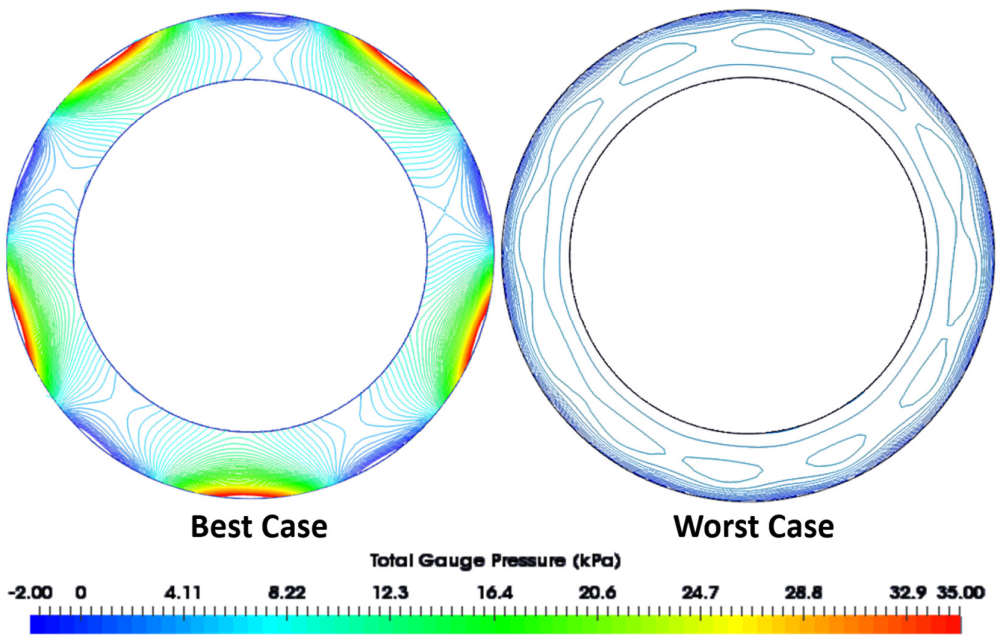


Figure 4.12. Static pressure contours at the turbine exit for the best and worst cases of the base cost function solution

4.1.2.2 Effects of the Optimization Variables on the Flow Physics with the Backpressure Dominated Function

The impact of backpressure on the objective function is increased four times to observe the effect of the objective function and the DE method's efficiency. For this purpose, the following objective function is used;

$$F_{obj} = [(\dot{m}_{te}/\dot{m}_{ci}) + 4 \times (P_{te} + P_{amb})/P_{amb}] \times 100 \quad \textbf{(20)}$$

A dataset is created according to parameter number in the differential evolution method, then results for this parameter set are calculated. According to analysis results, parameters that give minimum objective function are selected as optimum parameters. Therefore, the optimization process is completed in 90 iterations, similar to the previous study. The effects of parameters and peak results are studied in the following sections.

Table 4.6 displays variation of the objective function and other optimization criteria as a functioın of the lobe number. Similar to the previous study, fpr lower lobe numbers, the objective function and cooling inlet flow rate values are better, while the turbine exit backpressure is relatively higher. Although the most influential parameter is the lobe number, it is less effective on objective function for the pressure-dominated case.

Table 4.6 Results for different lobe numbers for the backpressure dominated cost function solution

Case	n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$	F_{obj}
1	72	6	151.4	341.8	1.55	7319	727.1
	27	11	151.4	339.2	1.26	5691	790.2
2	85	9	120.7	360.5	1.27	6086	788.9
	40	10	119.7	360.5	1.19	6046	810.5

It is expected that the increase of nozzle exit diameter should decrease the turbine exit backpressure. However, this is not valid for all cases, as shown in Table 4.7. It is observed that this parameter has the most influence on the objective function after the lobe number.

Table 4.7 Results for different nozzle exit diameters for the backpressure dominated cost function solution

Case	n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$	F_{obj}
1	79	8	114.4	335.2	1.49	6110	734.3
	57	8	114.4	338.1	1.47	6233	739.1
2	78	6	154.7	340.8	1.55	7264	726.9
	33	6	154.7	359.5	1.68	6444	700.1

The previous study stated that there is a direct proportional relationship between the lobe tangent radius and the amount of cooling air, but not a clear relationship in terms of turbine outlet backpressure. However, in this part of the study (i.e., for the backpressure dominated solutions), it is observed that this situation is reversed. Therefore, the effects of the lobe tangent on the result cannot be clearly stated. The least significant parameter is lobe tangent radius according to the given consequences in Table 4.8

Table 4.8 Results for different lobe tangent radiuses for the backpressure dominated cost function solution

Case	n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$	F_{obj}
1	34	8	100.3	335.2	1.47	6278	739.2
	79	8	114.4	335.2	1.49	6110	734.3
2	45	6	114.7	341.8	1.53	7820	733.2
	72	6	151.4	341.8	1.55	7319	727.1

4.1.2.2.1 Comparison of the Best and Worst Configurations

Similar to the previous section, the geometries for the best and worst cases are given. Also, the best and worst results for pressure-dominated objective function are numerically compared in the following sections, and the velocity and pressure distributions for critical areas are given.

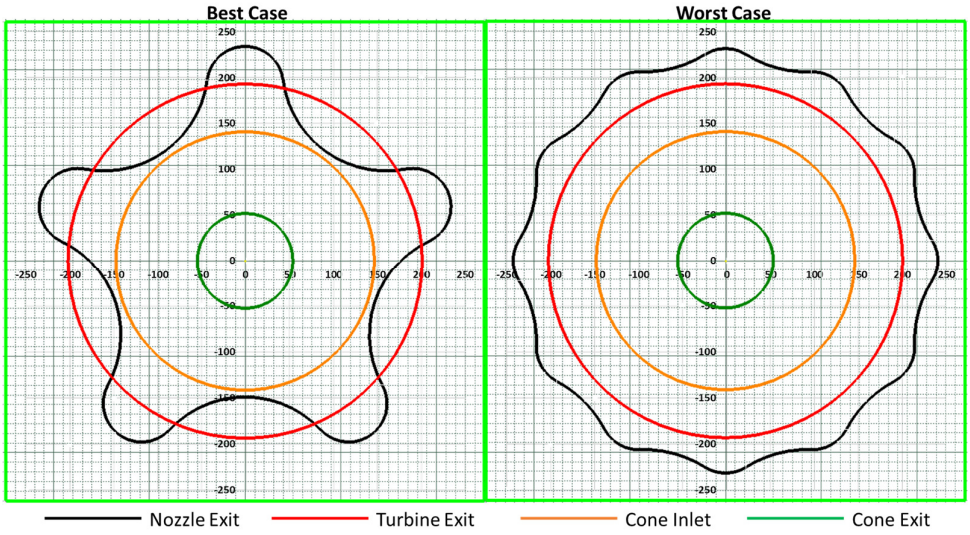


Figure 4.13. The geometries for the best and worst cases of the backpressure-dominated cost function solution

As in the previous case, the lobe shapes of the best case are more effective than the worst case. Parameters for the best and worst configurations are compared in Table 4.9.

Table 4.9 Parameters and results of the best and worst configurations for the backpressure-dominated cost function solution

Case	n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$
38	5	108.4	368.4	2.04	10057	667.4
47	12	171.6	365.8	1.11	6600	841.1

Similar to the base cost function case, the best configuration is achieved with fewer lobe numbers and smaller lobe tangent radius. However, the turbine outlet diameter is set at 368.4 mm, which is almost the maximum value.

Axial velocity distributions of cold regions from cooling inlet to turbine exit region are plotted in Figure 4.14.

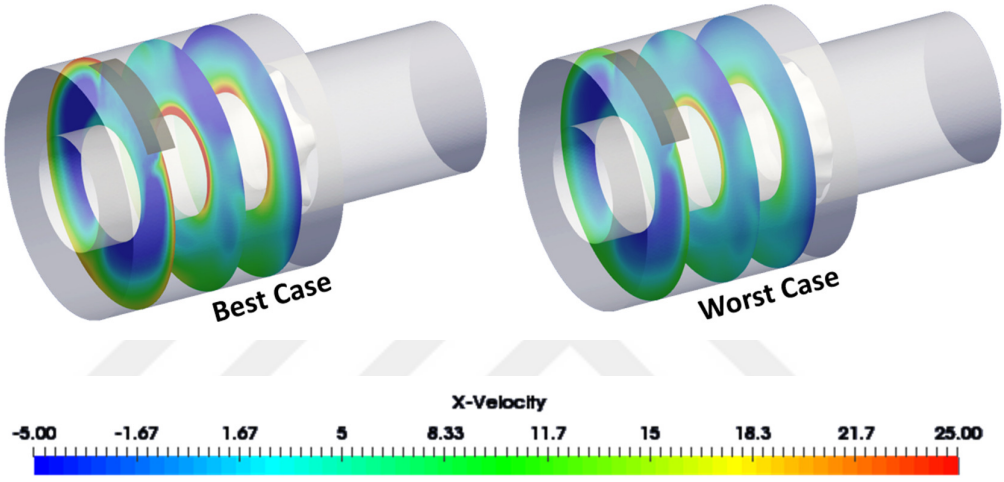


Figure 4.14. Velocity distributions for cold regions for the best and worst cases of the backpressure-dominated cost function solution

The secondary flow rate around the engine case is more effective in the best configuration, similar to the previous study. However, velocities in best configuration are higher than the base cost function case due to a large amount of cooling air.Also, velocity flow fields of analyses are generated for the same regions of the base cost function case.

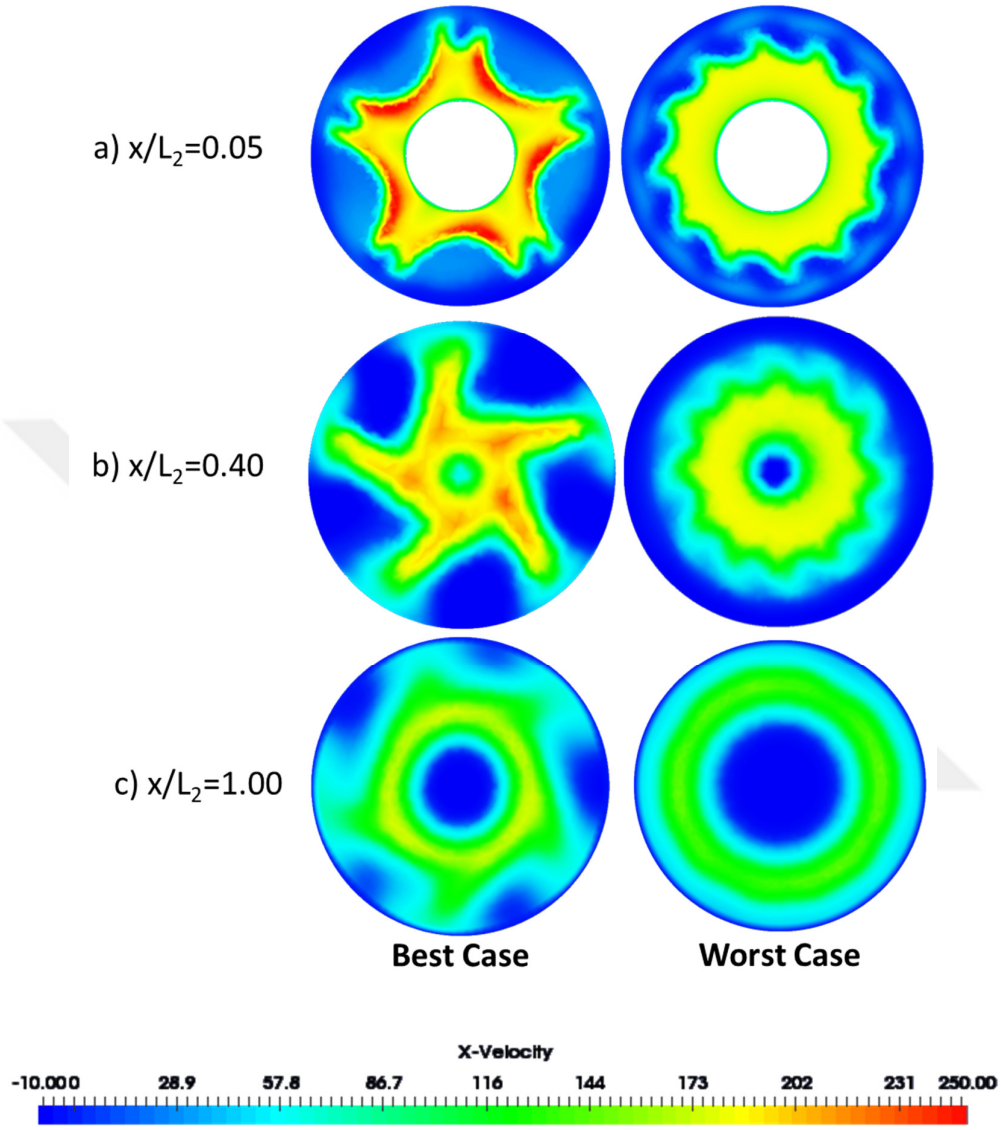


Figure 4.15. Contours of axial velocity (x-velocity) plotted on yz planes at different downstream positions from the nozzle exit for the best and worst cases of the backpressure-dominated cost function solution

From Figure 4.15, the best configuration has higher velocities due to a large amount of cooling air. However, the mixing of two flows is better for the other model, as seen from near the ejector wall. Compared to the base function case, it is seen that mixing is slightly better for the pressure-weighted case. The maximum

azimuthal vorticities for the best and worst cases are 6.02 and 4.79, respectively. The higher maximum azimuthal vorticities are observed in the best case as expected.

Temperature distributions downstream of the nozzle exit are shown below.

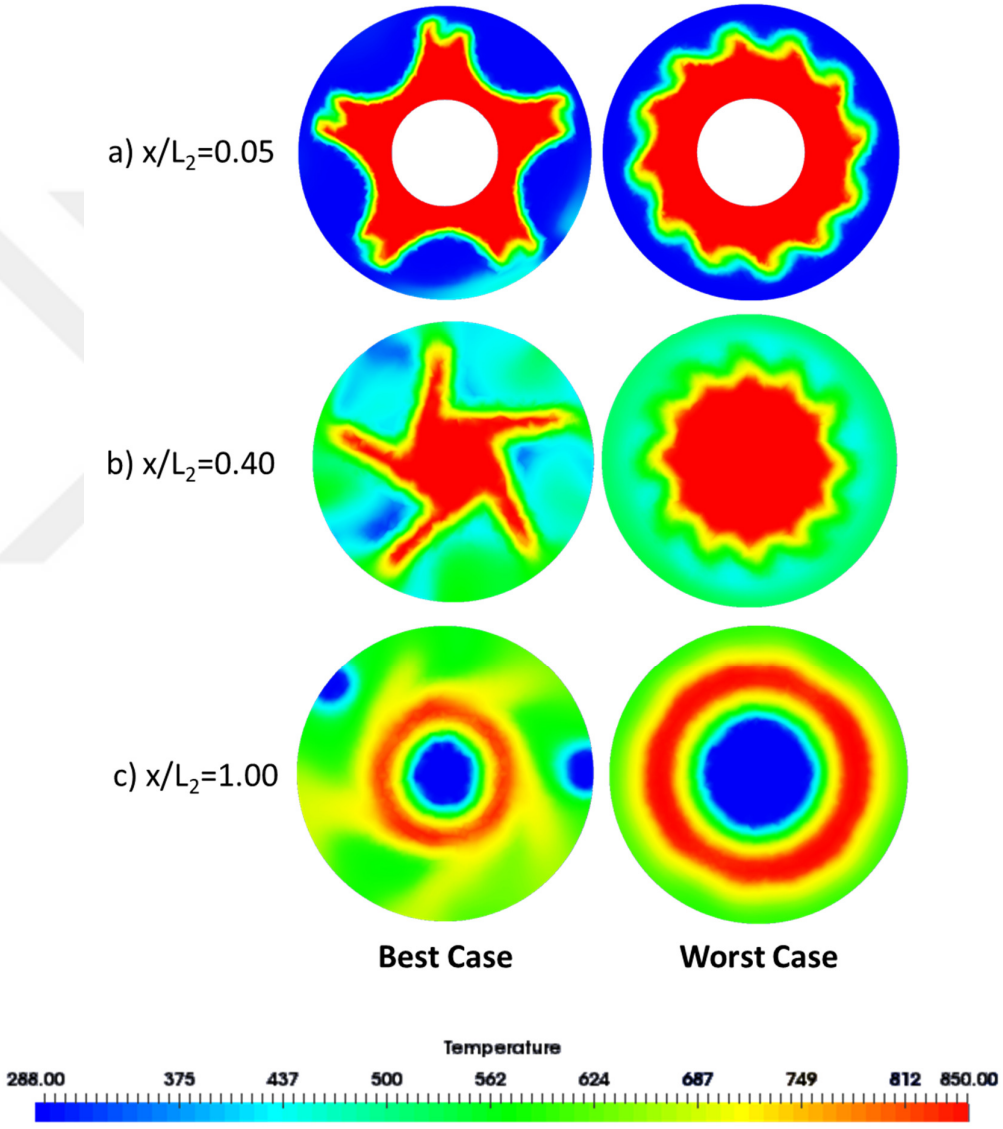


Figure 4.16. Contours of temperature plotted on yz planes at different downstream positions from the nozzle exit for the best and worst cases of the backpressure-dominated cost function solution

Temperature values for the best configuration are low due to secondary flow increment in some regions at the exit of the ejector. Moreover, temperature distributions also support that mixing of flows is better for the worst case.

Also, pressure isobars at the turbine exit plane are generated. The lobe shape effect is seen as the best configuration as in the previous study. However, pressure values are too high in this model.

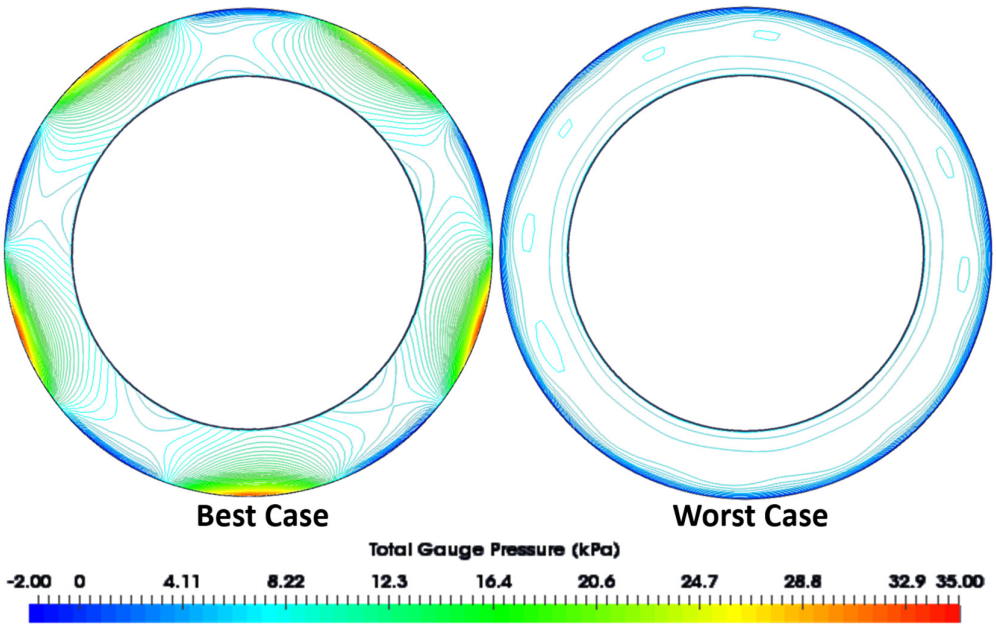


Figure 4.17. Static pressure contours at the turbine exit for the best and worst cases of the backpressure-dominated cost function solution

4.1.2.3 Comparisons of Two Different Optimization Results

The geometries and results of two different studies are compared in this section to assess the effect of the objective function definition.

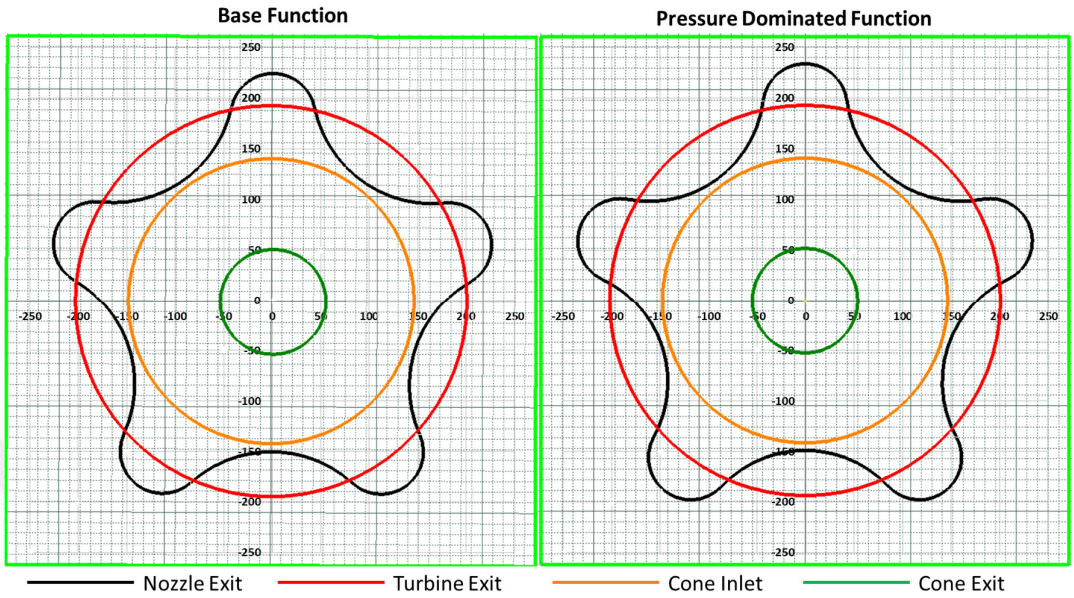


Figure 4.18. The geometries for the best cases of the base and backpressure-dominated cost function solutions

From Figure 4.18, there is a minor difference between the two shapes. The nozzle exit diameter is larger for the best case obtained with pressure dominated case. The numerical values of the geometries and results are given in Table 4.10 Parameters and results for Table 4.10 to allow for a better comparison.

Table 4.10 Parameters and results for the best cases of the base and backpressure-dominated cost function solutions

Case	n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$	Solution Time [day]
Base	62	5	113.5	351.2	1.90	10856	5.63
Pressure Dominated	38	5	108.4	368.4	2.04	10057	5.59

As seen in the above table, the turbine exit backpressure value is less for the pressure-dominated objective function, as expected. Besides, there is an improvement in the cooling inlet flow rate. Therefore, an optimized configuration for a pressure-based solution is better for both parameters.

When two results are compared in terms of geometries, it is observed that lobe numbers are the same, and the lobe tangent radiuses are similar. However, the nozzle exit diameter of the pressure-based configuration is higher as expected. Although it cannot be said any net effect of turbine exit diameter from previous assessments, the increase in it generally causes the turbine exit backpressure to decrease. For this reason, it can be inferred that the optimization method gives consistent results. Also, the computation times of the two solutions are similar because the same number of parameters are used.

4.1.3 Optimization Results for Four Nozzle Variables

In this chapter, lobed nozzle geometry is optimized with base objective function by adding a cone exit diameter. As the number of parameters increased to four, 30 new steps are added to the data set for optimization, and the process is completed with 120 iterations.

4.1.3.1 Effects of the Optimization Variables on the Flow Physics with the Base Function

It is observed in the previous sections that the lobe number, the nozzle exit diameter, and the lobe tangent radius used in optimization have no net effect on the cooling flow rate and the turbine exit backpressure. It can only be said that lower lobe numbers affect objective function well. Therefore, the results of these parameters are not examined for the new optimization process. Only the effects of the new parameter and the flow simulations are investigated in this chapter. A comparison of different cone exit diameters is given in Table 4.11.

Table 4.11 Results for different cone exit diameters for the base cost function solution

Case	n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	D_6 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$	F_{obj}
1	13	5	177.2	356.3	90.2	1.66	8418	385.3
	45	5	174.5	355.5	147	1.72	9100	376.2
2	9	7	129.7	341.9	94.8	1.47	6659	420.0
	69	7	129.7	341.9	122.7	1.50	6788	413.7

The increase of cone exit diameter causes a small nozzle exit area. Therefore, the primary flow velocity is increased, and this provides a more cooling inlet flow rate. Also, this raises backpressure at the turbine exit. This effect can be observed from the above results, but these effects are marginal.

4.1.3.1.1 Comparison of the Best and Worst Configurations

The details of the best and worst nozzle shapes are given in the following sections. Besides, the best and worst results for the four parameters are numerically compared in the following sections. Also, velocity and pressure distribution for critical areas are given.

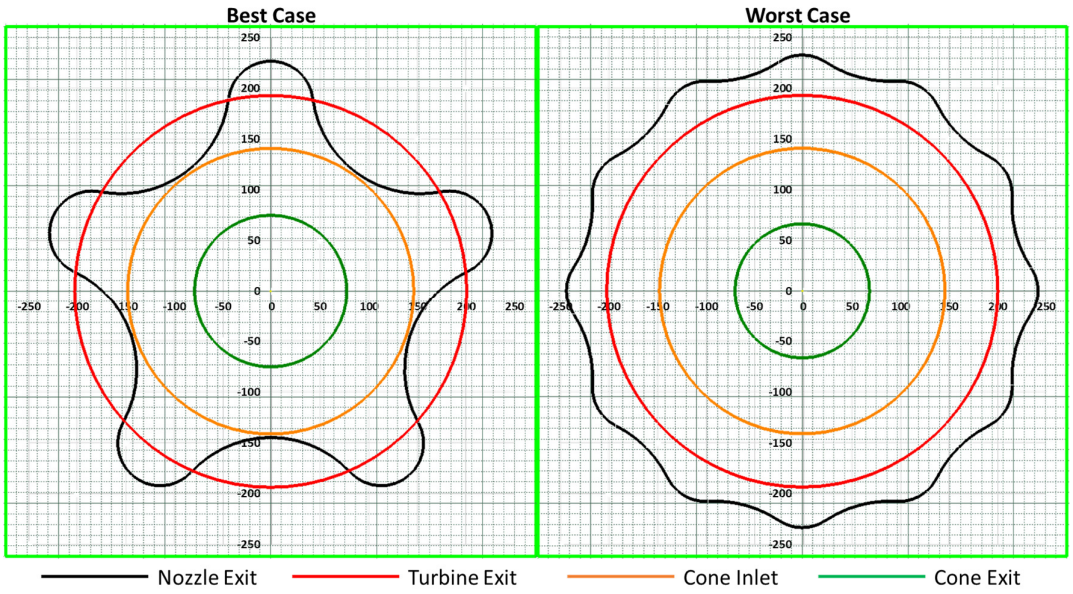


Figure 4.19. The geometries for the best and worst cases of the base cost function solution

In the worst-case, the lobe depth is almost zero, and the nozzle shape looks like a circle. The cone exit diameter is slightly higher in the best case. Parameters of the best and worst configurations are compared in Table 4.12.

Table 4.12 Parameters and results of the best and worst configurations for the base cost function solution

n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	D_6 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$	F_{obj}
114	5	102.5	354.9	143.9	2.41	11228	302.7
81	12	108.9	366.2	127.4	1.13	6460	512.9

There is a significant improvement of in the amount of cooling air in the best configuration compared to the worst case. On the other hand, the turbine exit backpressure is 70% less in the worst geometry. Similar to the previous cost function solutions, the best structure is achieved with the lowest lobe number and

almost the lowest tangent radius. As expected, the nozzle exit diameter is higher, almost maximum, for the worst configuration. The cone exit diameter is near the maximum value in range for the best design as the new parameter. Contours of axial velocity (x-velocity) on a number of yz planes in the cold flow region are shown in Figure 4.20.

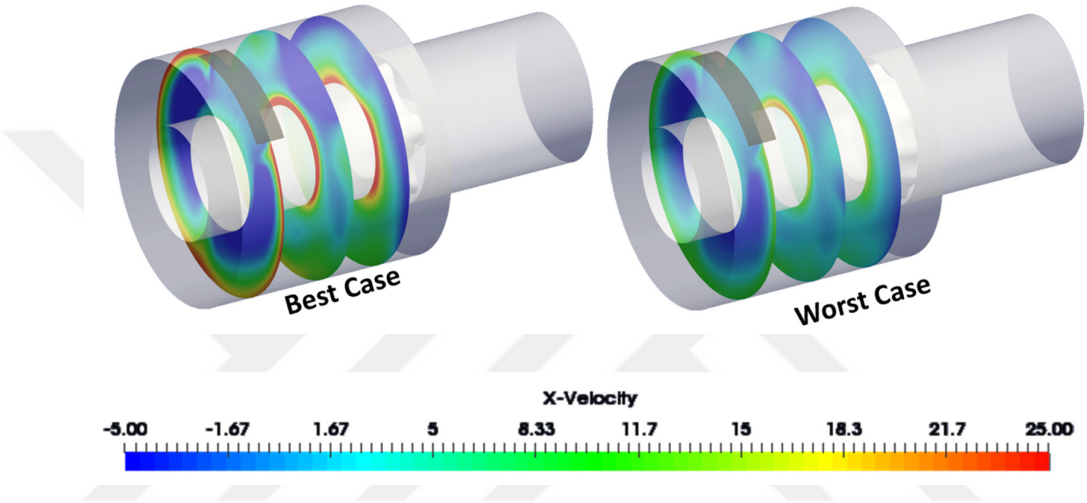


Figure 4.20. Axial velocity distributions for cold regions for the best and worst cases of the base cost function solution

In the best configuration, the cooling flow circulates the whole engine case with a high velocity due to higher primary flow. Also, it is well distributed in the engine bay compared to the worst case.

Axial velocity fields at some regions after nozzle exit are represented in Figure 4.21. According to this figure, primary nozzle flow and the cooling flow mix more rapidly in the worst case. The maximum azimuthal vorticities are calculated for the best and worst cases to compare the mixing efficiencies of the two geometries quantitatively. The maximum azimuthal vorticities for the best and worst cases are 6.38 and 4.70, respectively It is found that the azimuthal vorticity is lower for the worst condition, and this supports that the mixing efficiency is well.

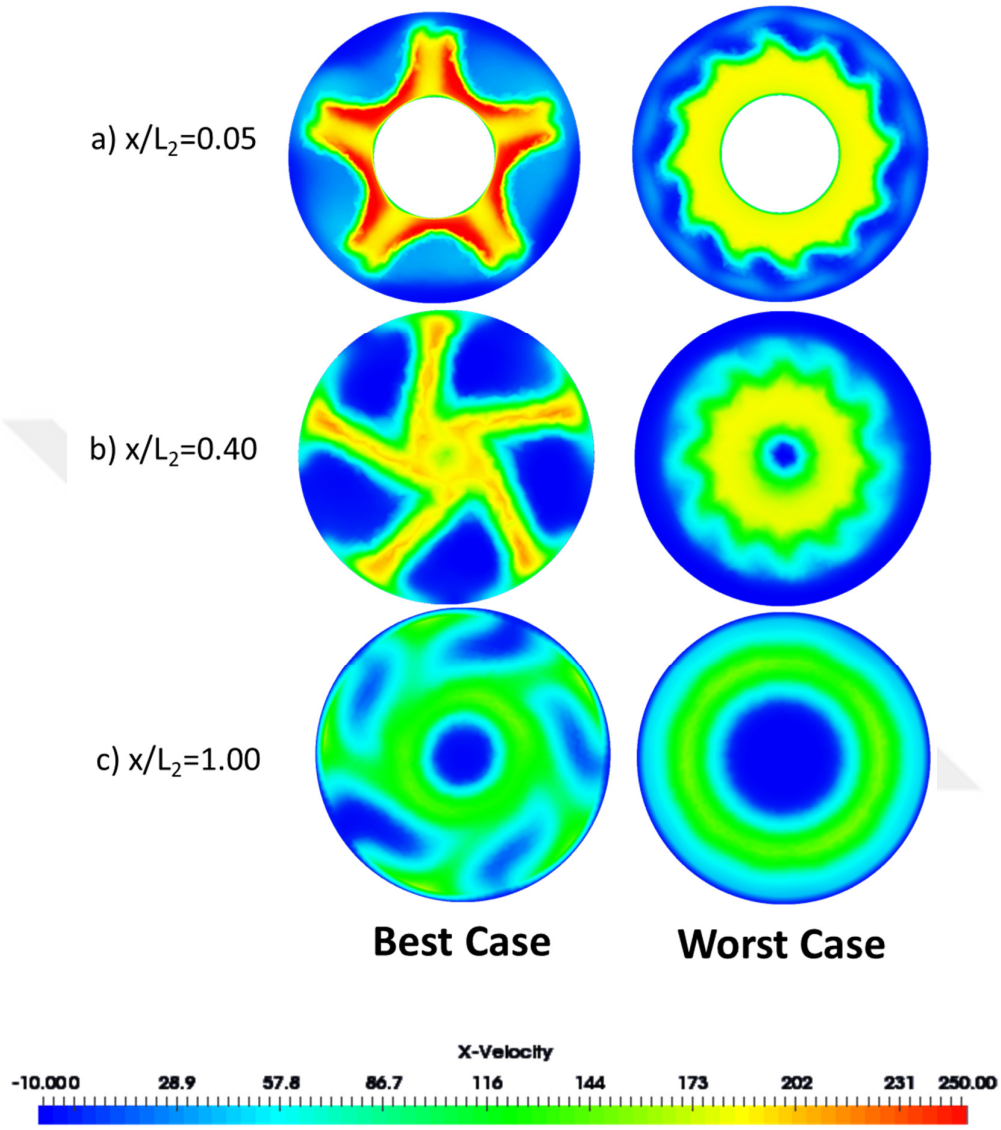


Figure 4.21. Contours of axial velocity (x-velocity) plotted on yz planes at different downstream positions from the nozzle exit for the best and worst cases of the base cost function solution

Temperature distributions are also plotted for the 4-parameter optimization study in Figure 4.22..

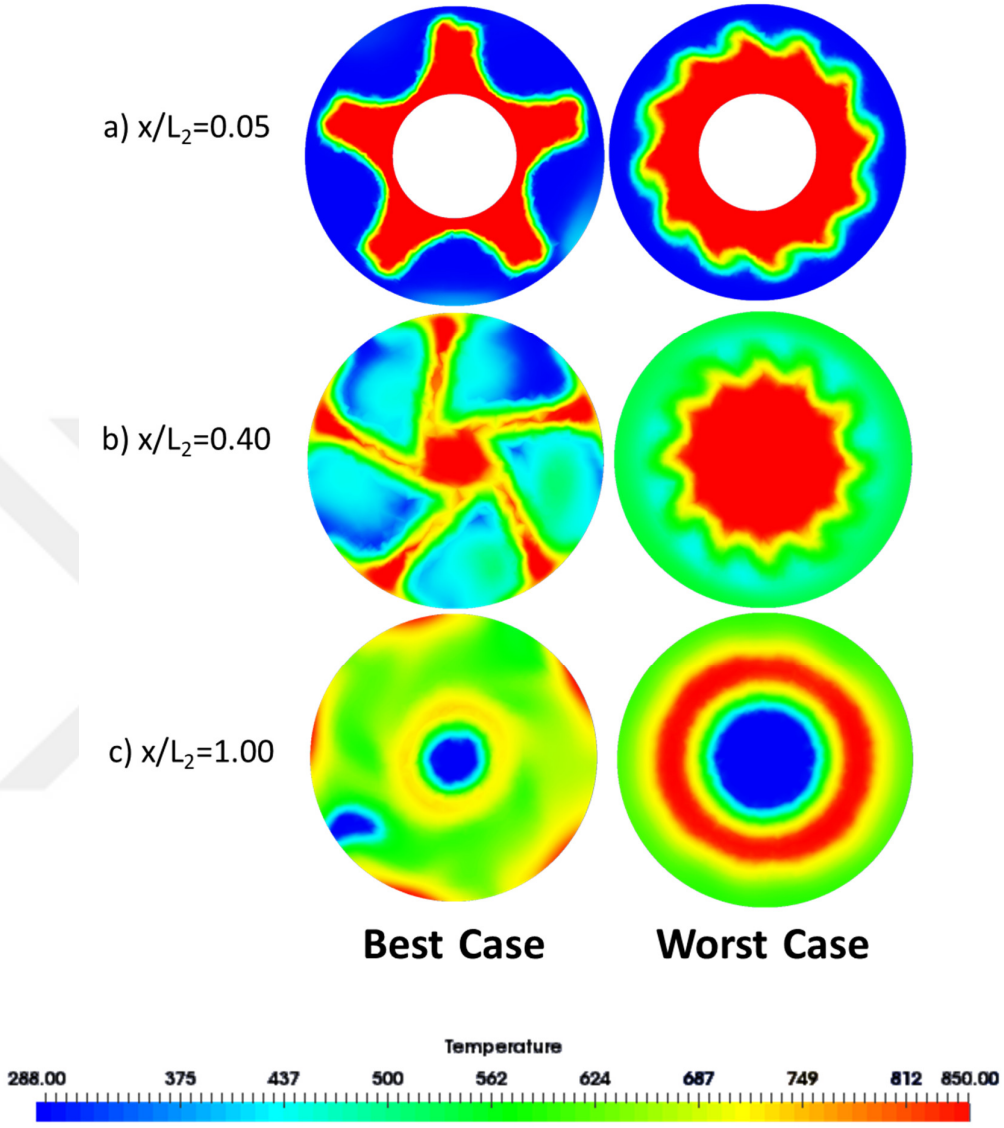


Figure 4.22. Contours of temperature plotted on yz planes at different downstream positions from the nozzle exit for the best and worst cases of the base cost function solution

As seen in Figure 4.22, the temperatures are lower for the best case. Also, it can be said that flows are completely mixed except for the middle region.

Isobaric contours at the turbine exit planes are compared to see the effects of configurations on pressure distribution.

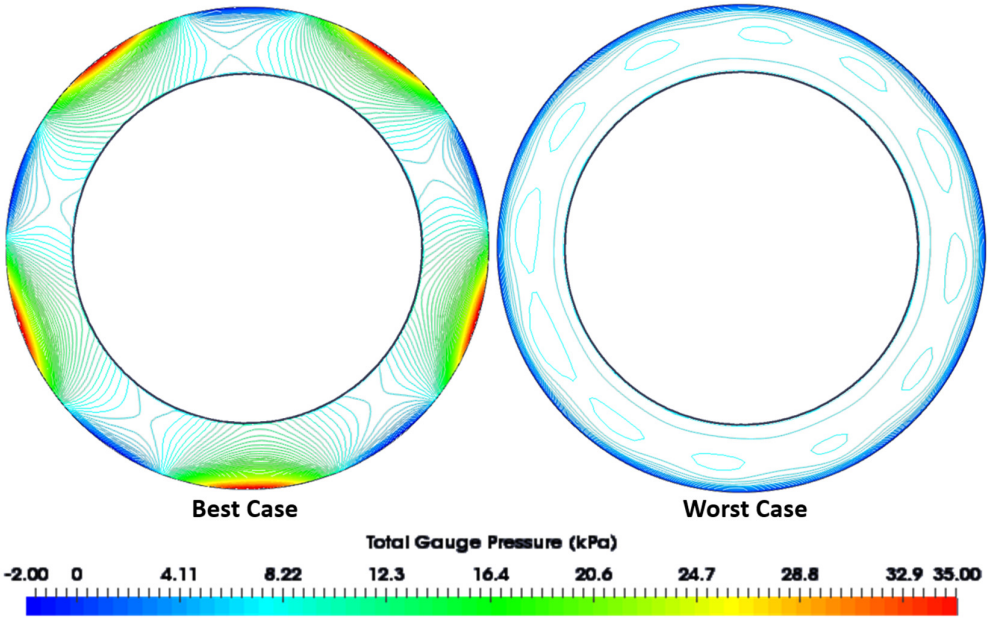


Figure 4.23. Static pressure contours at the turbine exit for the best and worst cases of the base cost function solution

Similar to previous cost function solutions, the effects of lobe shape are more visible for the best configuration. In contrast, the pressure is equally distributed for the 12-lobe nozzle, and this causes less distortion.

4.1.4 Comparisons of Optimization Results for 3 and 4 Parameters

Optimized parameter values and related results are demonstrated to see the consistency of the optimization method and the effect of adding a new parameter.

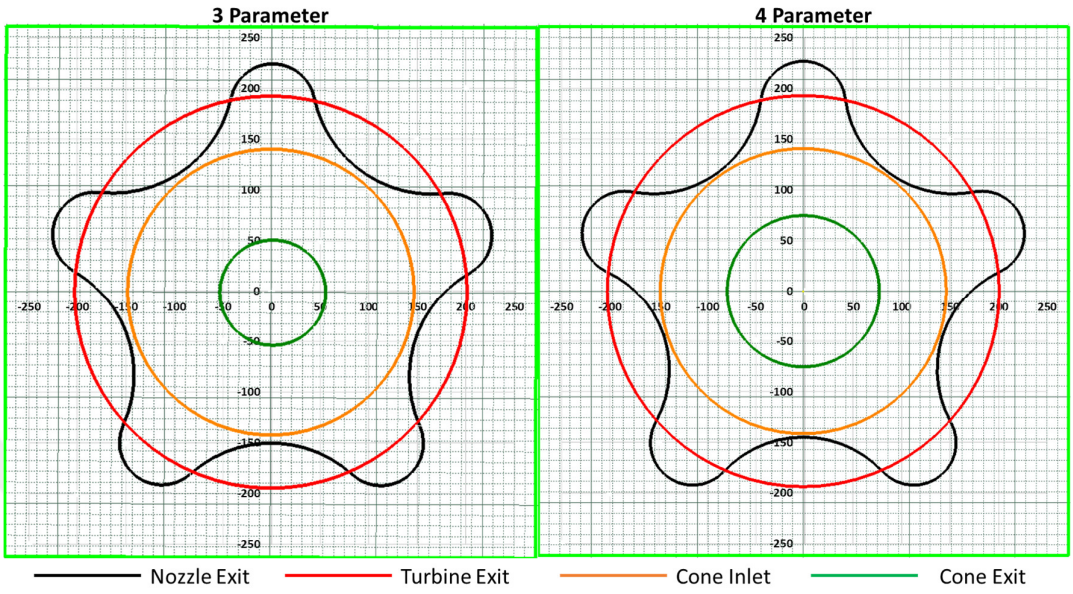


Figure 4.24. The geometries for the best cases of the base function solutions with three and four parameters

The nozzle shapes are similar for both cases, as can be seen from Figure 4.24. The main difference between the two different cases is seen in cone exit diameter. The cone exit diameter is increased with the addition of this parameter to the optimization. The numerical values of the results are also compared in Table 4.13.

Table 4.13 Parameters and results for the best cases of the base function solutions with three and four parameters

n_{para}	n_{iter}	n_l	D_7 $[mm]$	D_8 $[mm]$	D_6 $[mm]$	$\dot{m}_{te}$ $[kg/s]$	P_{te} $[Pa]$	F_{obj}	*Solution Time [day]*
3	62	5	113.5	351.2	100	1.90	10856	353.7	5.63
4	114	5	102.5	354.9	143.9	2.41	11228	302.7	7.69

Optimization is achieved with five lobe nozzle, which is the lowest value in the range, for both cases. Besides, lobe tangent radii and nozzle exit diameters are so

close to each other. This situation shows that the DE method is consistent and reliable for the optimization of the ejector geometry.

Another important inference is that the addition of a new parameter provides a significant improvement in the system performance. As previously mentioned, the increase of cone exit diameter helps to achieve a highersecondary flow rate. As shown in Table 4.12, the cooling inlet flow rate obtained with four parameters is about 25% higher than the other process. This improvement is achieved with only 3% turbine backpressure penalty. Therefore, the addition of a new parameter helps to procure better-lobed nozzle geometry. Besides, the solution time is higher for the four-parameter cases due to the addition of the new parameter It can be said that the solution time is directly proportional to the parameter number.

The distributions of the axial velocities and temperatures downstream the nozzle exit are compared in the following figures to see the effects of parameter addition on flow physics. From these figures, it is seen that the addition of a parameter provides a significant improvement in velocity and mixing. Moreoever, the temperature values are lower for the solution achieved by four parameters. The improvement of the mixing can be observed by comparing the maximum azimuthal vorticity values of the best cases of different solutions. The maximum azimuthal vorticities of the best cases are 6.65 for the solution with three parameters and 6.38 for the 4-parameter case. This shows that flows are mixed better for the solution with four parameters.

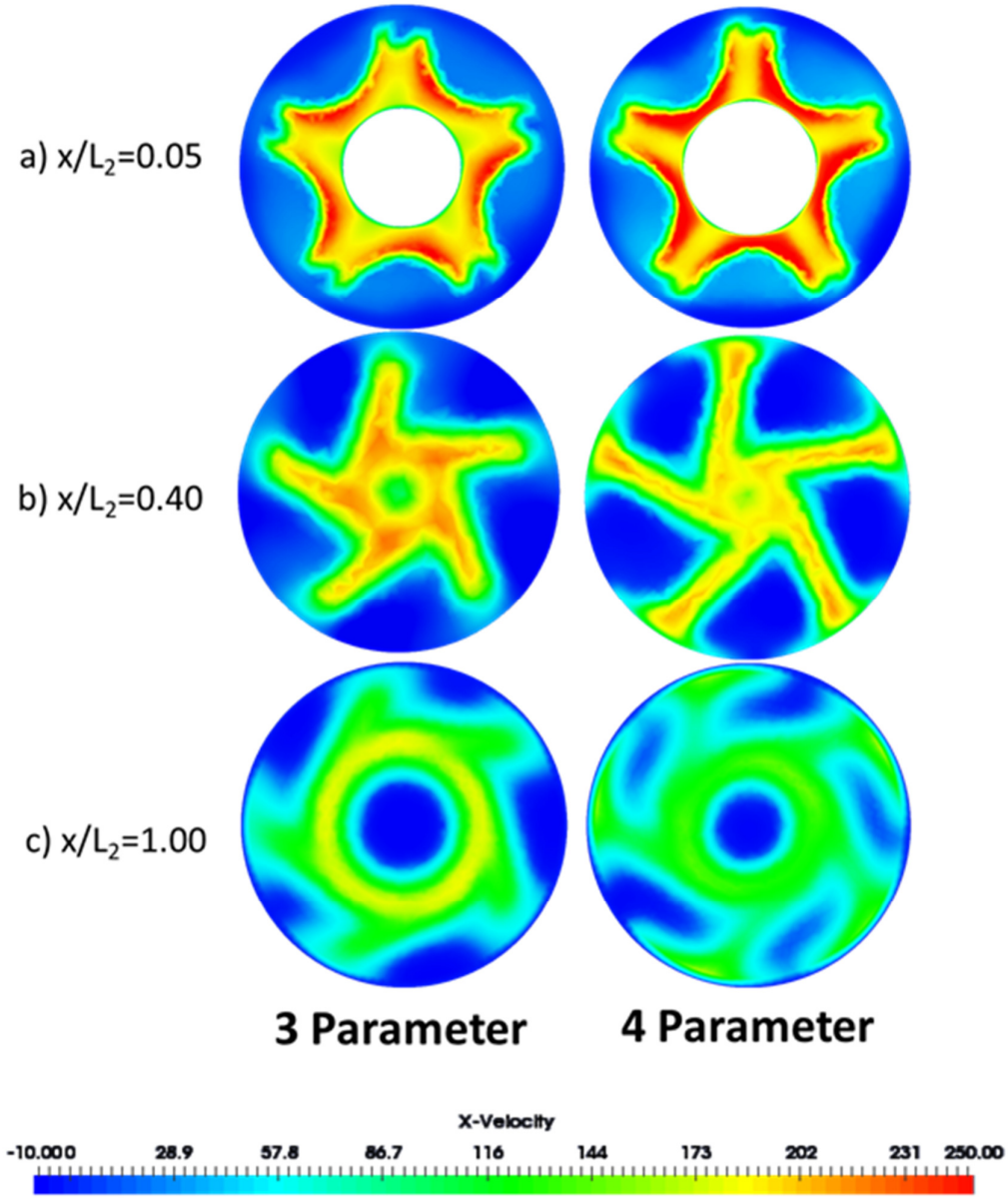


Figure 4.25. Contours of axial velocity (x-velocity) plotted on yz planes at different downstream positions from the nozzle exit for the best cases of the base cost function solutions with three and four parameters

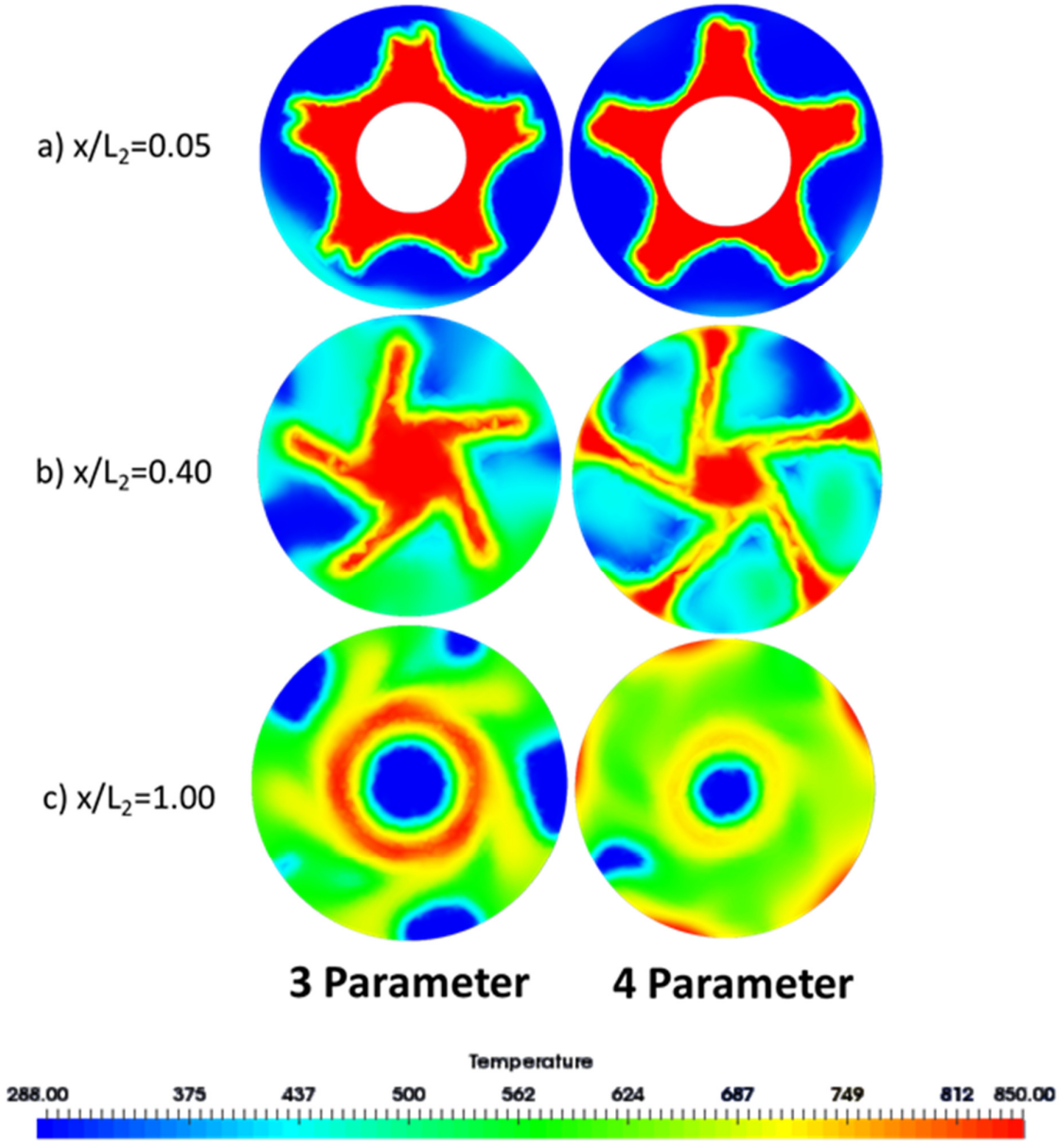


Figure 4.26 Contours of temperature plotted on yz planes at different downstream positions from the nozzle exit for the best cases of the base cost function solutions with three and four parameters

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The main objective of this thesis is to develop a methodology for the optimization of an ejector system used for helicopter engine bay cooling with changing nozzle and center body geometry. To determine the performance of an ejector system, an objective function is used, which includes entrainment ratio and turbine exit backpressure. The optimum geometry of a lobed nozzle is obtained by using the Differential Evolution (DE) method.

The optimization process is performed using a Python code, which combines the drawing, mesh, and flow simulation software. Firstly, a helicopter engine bay that uses a generic ejector system with a lobed nozzle is modeled in CATIA by using this code. Then, an unstructured mesh, according to the turbulence model, is generated with POINTWISE. Finally, the engine bay is analyzed by using FLUENT software. The differential evolution method is used to generate new parameter values in each iteration. By using these new parameter values, new geometry is modeled, meshed, and analyzed with code. According to the results, a new parameter set is generated, and this process continues until the whole parameter set is completed.

In the first instance, the methods and tools to be used had to be validated. Experimental results of a lobed nozzle performed by Hu et al. [53] are used. Flow analysis of this experimental setup is simulated with the realizable k-ε with standard wall function and k-ω SST turbulence method. These simulation results are compared with the study of Hu et al. [53] in terms of velocity fields at some specific cross-sectional regions. Moreover, azimuthal vorticities downstream of the

nozzle exit are checked for numerical comparisons. According to comparisons, it is seen that both models and FLUENT can be used for this optimization process. Realizable k-ε turbulence model with standard wall function is selected for the analysis due to its low time consumption.

After the validation phase is completed, the next step is making the solution independent from the mesh structure. For this reason, four mesh structures with different cell sizes are generated and analyzed. The results are compared with each other. It is determined that the moderate mesh structure is adequate for this study since the difference between medium and fine meshes is less than 3% in terms of turbine exit backpressure and cooling inlet flow rate. On the other hand, this difference for the coarse mesh is 11% for turbine exit backpressure, while 2% for cooling inlet flow rate.

Optimization of the lobe number, nozzle exit diameter and lobe tangent radius, of the lobed nozzle is performed using the Differential Evolution method. In the first step, the effectivity and consistency of the optimization method are investigated. For this reason, the best geometry is achieved for the base and pressure-dominated objective functions. As expected, the nozzle exit diameter is increased for pressure-dominated function in order to decrease the blockage effect of the nozzle. On the other hand, other parameters are similar for both cases. It shows that the Differential Evolution method is consistent and applicable in this study.

Finally, the optimization of the lobed nozzle is performed by the addition of a new parameter, i.e., the cone exit diameter. The addition of the new parameter resulted in a 25% increase in the cooling inlet flow rate with only a 3% turbine outlet backpressure increase. The best geometry is achieved with five lobes which is the minimum allowed lobe number in the optimization study. Similar to the lobe number, the tangent radius is very close to the minimum of the defined range. On the other hand, the turbine outlet diameter is approximately the average value of the defined range. The cone exit diameter, which is the last parameter, is approximately at its maximum value.

The velocity and temperature distributions of the best and worst configurations for different areas are compared to assess the effect of nozzle shape on the flow topology. It is seen that the swirl effect that caused by the engine turbine is highly eliminated for the best configuration. Therefore, more cooling flow can be entrained. Also, this also resulted in a substantial reduction in the recirculation zone formed in the center of the exit zone. However, the reduction of the swirl number decreases the mixing efficiency of the flow. The turbine exit back pressure, on the other hand, is higher for this model since the nozzle exit diameter is relatively small.

When all parameter effects are examined one by one, it can be said that lobe number is the most influential parameter. Low-number lobe models show better performance in terms of turbine exit backpressure and cooling inlet flow rate. However, there is no clear trend observed for the other parameters.

As a result, it can be concluded that the nozzle has a pronounced effect on ejector performance. Therefore, one can reach the desired ejector system performance by small modifications.

5.2 Future Work

For the future enhancements of the study, the following subjects can be studied:

- The optimization can be repeated by the addition of new nozzle and ejector parameters such as:
    - Nozzle length
    - Center body length
    - Lope diameter
    - Ejector diameter and length
- The new design targets can be added to objective function such as:
    - Mixing efficiency of flows

    - The cooling efficiency of secondary flow
    - Temperature distribution at the ejector exit
- Optimization can be performed with different global optimizers such as:
    - Basinhopping method
    - Dual Annealing method

REFERENCES

[1] N. D. Lawrance, "Experimental And Numerical Investigation Of The Design And Control Of Vapor-Compression Systems With Integration Of Two-Phase Ejectors For Performance Enhancement Through Expansion Work Recovery," University of Illinois at Urbana-Champaign, Urbana, Illinois, 2016.

[2] H. Cao and H. T. Brake, "Progress And Challenges In Utilization Of Ejectors For Cryogenic Cooling," Elsevier Ltd, Beijing, 2020.

[3] E. F. Kranakis, "The French Connection: Giffard's Injector and the Nature of Heat," *Technology and Culture,* vol. 23, no. 1, pp. 3-38, 1982.

[4] Z. Zhang, X. Feng, D. Tian, J. Yang and L. Chang, "Progress in ejector-expansion vapor compression refrigeration and heat," Department of Architecture and Civil Engineering, North China University of Science and Technology, Tangshan, 2020.

[5] C. Parsons and M. Stoney, "Trials of steam turbines for driving dynamos, Proc. Inst. Mech. Eng. 61," 1901.

[6] M. Leblanc, "Refrigerating-machine". Patent US1029201A, 1912.

[7] J. K. Nam, C. N. Kim and W. Chun, "Using the condenser effluent from a nuclear power plant for Ocean Thermal Energy Conversion (OTEC),"

International Communications in Heat and Mass Transfer, vol. 36, pp. 1008-1013, 2009.

[8] L. Jun-Hee, A. Sameen, V. S. Kumar, H. Kim, C. Bo-Gyu and K. Kim, "Studies on Ejector Systems for Hydrogen Fuel Cell," in *Joint Propulsion Conference & Exhibit*, Arizona, 2005.

[9] W. F. Stoecker and J. W. Jones, Refrigeration and Air Conditioning, New York, 1958.

[10] B. Dmytro, N. Sergii and S. Vitaliy, "Solar Ejector Refrigerating And Air-Conditioning System (SERAS) For Tower Buildings," in *International High Performance Buildings*, Purdue, 2010.

[11] B. Giorgio, M. Riccardo and I. Fabio, "Ejector refrigeration: A comprehensive review," *Renewable and Sustainable Energy Reviews,* vol. 53, pp. 373-407, 2015.

[12] V. Szabolcs, O. Armando C. and D. Bogdan, "Analysis of a solar-assisted ejector cooling system for air conditioning," *International Journal of Low-Carbon Technologies,* vol. 4, pp. 2-8, 2009.

[13] M. Moujahid, "Compressible Flow Analysis of Thrust Augmenting Ejectors," Embry-Riddle Aeronautical University, Daytona Beach, Florida, 1995.

[14] N. Gay, "Refrigerating system". Patent 1836318, 1931.

[15] G. Flügel, Design of jet, Berlin: VDI-Forschungsheft, 1939.

[16] F. A. Mcclintock and J. H. Hood, "Aircraft Ejector Performance," *Journal Of The Aeronautical Sciences,* vol. 13, no. 11, 1946.

[17] J. W. Mitchell, "Design Parameters For Subsonic Air-Air Ejectors," *Armed Services Technical Information Agency,* no. 40, 1958.

[18] S-12 Powered Lift Propulsion Committee, "Performance of Low Pressure Ratio Ejectors for Engine Nacelle Cooling," *SAE Aerospace Information Report AIR,* 2004.

[19] R. Porcar and J. Prenel, "Visualisation Des Ondes De Choc Dans Un Ejecteur Supersonique," *Optics Communications,* vol. 14, no. 1, pp. 104-107, 1975.

[20] J. Prenel and R. Porcar, "Visualisation Des Ondes De Choc Dans Un Ejecteur Supersonique Exploitation De La Polarisation De La Lumiere Diffusee," *Optics Communications,* vol. 17, no. 3, pp. 346-349, 1976.

[21] P. Desevaux, "A method for visualizing the mixing zone between two co-axial flows in an ejector," *Optics and Lasers in Engineering,* vol. 35, pp. 317-323, 2001.

[22] A. Bouhanguel, P. Desevaux and E. Gavignet, "Flow visualization in supersonic ejectors using laser tomography techniques," *International Journal of Refrigeration,* vol. 34, pp. 1633-1640, 2011.

[23] Z. Yinhai and J. Peixue, "Experimental and analytical studies on the shock wave length in convergent and convergent–divergent nozzle ejectors," *Energy Conversion and Management,* vol. 88, pp. 907-914, 2014.

[24] M. S. Chandrasekhara., A. Krothapalli and D. Baganoff, "Performance Characteristics of an Underexpanded Multiple Jet Ejector," *J. Propulsion,* vol. 7, no. 3, pp. 462-464.

[25] R. S. Nave, "Computational fluid dynamics analysis of diffuser performance in gas-powered jet pumps," *Int. J. Heat and Fluid Flow,* vol. 14, no. 4, pp. 401-407, 1993.

[26] B. Riffat, S. G. Gan and S. Smith, "Computational Fluid Dynamics Applied To Ejector Heat Pumps," *Applied Thermal Engineering,* vol. 13, no. 4, pp. 291-297, 1995.

[27] E. Rusly, L. Aye, W. Charters and A. Ooi, "CFD analysis of ejector in a combined ejector cooling system," *International Journal of Refrigeration,* vol. 28, pp. 1092-1101, 2005.

[28] T. Sriveerakul, S. Aphornratana and K. Chunnanond, "Performance prediction of steam ejector using computational fluid dynamics: Part 1. Validation of the CFD results," *International Journal of Thermal Sciences,* vol. 46, pp. 812-822, 2007.

[29] T. Sriveerakul, S. Aphornratana and K. Chunnanond, "Performance prediction

of steam ejector using computational fluid dynamics: Part 2. Flow structure of a steam ejector influenced by operating pressures and geometries," *International Journal of Thermal Sciences,* vol. 46, pp. 823-833, 2007.

[30] A. Hemidi, F. Henry, S. Leclaire, J.-M. Seynhaeve and Y. Bartosiewicz, "CFD analysis of a supersonic air ejector. Part I: Experimental validation of single-phase and two-phase operation," *Applied Thermal Engineering,* vol. 29, pp. 1523-1531, 2009.

[31] M. Federico and M. Adriano, "Performance analysis of a supersonic ejector cycle working with R245fa," *International of Refrigeration,* vol. 49, pp. 79-92, 2015.

[32] A. Bouhanguel, P. Desevaux and E. Gavignet, "Visualization of flow instabilities in supersonic ejectors using Large Eddy Simulation," *The Visualization Society of Japan,* vol. 18, pp. 17-19, 2014.

[33] J. C. Dutton and F. Carroll, "Optimal Supersonic Ejector Designs," *ASME,* vol. 108, pp. 414-420, 1986.

[34] Y. Bartosiewicz, Z. Aidoun, P. Desevaux and Y. Mercadier, "Numerical and experimental investigations on supersonic ejectors," *International Journal of Heat and Fluid Flow,* vol. 26, pp. 56-70, 2005.

[35] R. M. Kumaran, P. K. Vivekanand and T. Sundararajan, "Optimization of Second Throat Ejectors for High-Altitude Test Facility," *Journal Of*

Propulsion And Power, vol. 25, no. 3, 2009.

[36] C. Sergio, P. Sebastien and G. Nicolas, "Coupled Thermodynamic and CFD Approaches Applied to A Supersonic Air Ejector," in *16th International Refrigeration and Air Conditioning Conference*, Purdue, 2016.

[37] G. Jerzy, S. Kamil, B. Dariusz and K. Jaroslaw, "Comparative Study Of Turbulence Models In Application To Gas Ejectors," *International Journal of Thermal Sciences,* vol. 78, pp. 9-15, 2014.

[38] A. Maghsoodi, M. Dadvar and H. Ahmadikia, "A New 1d Model for Supersonic Ejector Performance Evaluation At Critical And Sub-Critical Operational Modes," *International Journal of Air-Conditioning and Refrigeration,* vol. 22, no. 1, 2014.

[39] E. Yehia A., A. Hossam S., S. Khalid M. and J. Mohammad Nazri Mohd, "A Multiple Inlet Swirler for Gas Turbine Combustors," *World Academy of Science, Engineering and Technology,* vol. 53, pp. 1260-1263, 2009.

[40] V. K. Cheeda, R. V. Kumar and G. Nagarajan, "Design And Cfd Analysis Of A Regenerator For A Turboshaft Helicopter Engine," *Aerospace Science and Technology,* vol. 12, pp. 524-534, 2008.

[41] M. Kazlauskas, "Parametric Study of an Ejector-Cooled Turbine Engine Compartment Using CFD," Rensselaer Polytechnic Institute , Hartford, Connecticut, 2009.

[42] X. Yang, X. Long and X. Yao, "Numerical investigation on the mixing process in a steam ejector with different nozzle structures," *International Journal of Thermal Sciences,* vol. 56, pp. 95-106, 2012.

[43] V. Dvořák, "Study of Optimization of Lobed Nozzle For Mixing," in *Colloquium Fluid Dynamics*, Prague, 2007.

[44] "Catia Automation Home Page," Dassault Systems, [Online]. Available: http://catiadoc.free.fr/online/CAAScdBase/CAAScdAutomationHome.htm.

[45] I. Pointwise, "Pointwise Glyph," Pointwise, Inc, [Online]. Available: https://www.pointwise.com/glyph/files/Glyph/cxx/GgGlyph-cxx.html.

[46] Y. Demirel, "Using the second law of thermodynamics," in *Transport and Rate Processes in Physical and Biological Systems*, Blacksburg, VA, Department of Chemical Engineering, 2007, pp. 131-135.

[47] B. P. Stephen, Turbulent Flows, Cambridge University Press.

[48] J. Boussinesq, "Essai sur la théorie des eaux courantes," *Mémoires présentés par divers savants à l'Académie des Sciences,* vol. 23, pp. 1-680, 1987.

[49] D. Spalding and B. Launder, "The Numerical Computation Of Turbulent Flows," *Computer Methods In Applied Mechanics And Engineering,* vol. 3, pp. 269-289, 1974.

[50] S. Tsan-Hsing, L. William W., S. Aamir, Y. Zhıgang and Z. Jıang, "A New Kt

Eddy Viscosity Model For High Reynolds Number Turbulent Flows," *Compurers Fluids,* vol. 24, no. 3, pp. 227-238, 1995.

[51] F. R. Menter, "Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications," *AIAA,* vol. 32, no. 8, pp. 1598-1605, 1994.

[52] L. Dawei, H. Jun, S. Zhiqiang and J. Jinzu, "The Effects of Scalloping Width and Position on Jet Mixing of Lobed Nozzles," *J. Aerosp. Technol. Manag.,* vol. 7, no. 4, pp. 425-431, 2015.

[53] H. Hui, S. Tetsuo and K. Toshio, "Dual-Plane Stereoscopic PIV Measurements in a Lobed Jet Mixing Flow," in *Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 2005.

[54] "Turbulence Modeling," in *Introduction to ANSYS Fluent* , ANSYS, Inc., 20113, p. 12.

[55] B. Ernesto and M. Gianluigi Alberto, "Performance Of A Turboshaft Engine For Helicopter Applications Operating At Variable Shaft Speed," in *Proceedings of the ASME 2012 Gas Turbine India Conference*, Mumbai, Maharashtra, India, 2012.

[56] P. Kenneth and S. Rainer, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization Over Continious Spaces," *Journal of Global Optimization,* no. 11, pp. 341-359, 1997.

[57] C. Guangyu and D. Xiaoqun, "An Improved Differential Evolution Method Based on the Dynamic Search Strategy to Solve Dynamic Economic Dispatch Problem with Valve-Point Effects," *Hindawi Publishing Corporation Abstract and Applied Analysis,* 2014.

[58] H. R. Cai, C. Y. Chung and K. P. Wong, "Flow, Application of Differential Evolution Algorithm for Transient Stability Constrained Optimal Power," *Ieee Transactions On Power Systems,* vol. 23, 2008.

[59] J.-S. Wu and M. R. Smith, "Differential Evolution Algorithms in High Resolution CFD Applications," in *Joint workshop on Numerical Analysis and Scientific Computation*, Taiwan-Japan, 2009.

[60] O. Suchomel and T. Hyhlík, "Optimisation methods in CFD – Differential Evolution," *Faculty of Mechanical Engineering of Czech Technical University.*

[61] M. Percin., M. Vanierschot and B. W. v. Oudheusden, "Analysis of the pressure fields in a swirling annular jet flow," *Experiments in Fluids,* vol. 58, no. 12, 2017.

APPENDICES

A. SAMPLE CALCULATION for DIFFERENTIAL EVOLUTION METHOD

The working procedure of differential evolution (DE) can be explained in 4 parts: problem definition, DE parameters, initialization, and DE position update. DE is described here by considering a simple optimization problem (sphere function) with two decision variables. There are many variants of DE, which are called DE strategies. In this example, only DE/1/rand/bin strategy is considered for explanation purposes.

1) Problem Definition

Consider an optimization problem:

$$Minimize\ f(x) = x_1^2 + x_2^2 \quad -5 \leq x_1, x_2 \leq 5$$

2) Differential Evolution Method Parameters

$$Population\ size\ N = 5$$

$$Dimension\ of\ the\ problem = 2$$

$$Stopping\ Criteria = Maximum\ Number\ of\ Iterations = 2$$

$$Scaling\ Factor\ F = 0.5$$

$$Crossover\ Probability\ P_{cr} = 0.7$$

3) Initialization

The initial dataset is prepared by selecting parameter values randomly.

Position	Function Value
$x_1 = (1.7667, -4.1337)$	20.2089
$x_2 = (4.8071, 0.6642)$	23.5502

Position	Function Value
$x_3 = (-2.9232, -4.0439)$	24.8983
$x_4 = (-4.3747, -4.7421)$	41.6270
$x_5 = (-1.6587, 0.5680)$	3.07408

4) Differential Evolution Position Update

Generation – 1

In DE, position update is carried out by two operators, mutation and crossover. It is different from the evolutionary algorithms where the solution is mutated after the crossover. Nevertheless, here, the mutation is the first operator, and then crossover is applied.

Mutation

In differential evolution, the mutation operator is used to create a trial vector, named as u, for each solution. It is done by mutating a target vector.

$$Trial\ Vector = Target\ Vector + Scale\ Factor\ x\ (Randomly\ Selected\ Solution_1 - Randomly\ Selected\ Solution_2)$$

Here it should be noted that selected variables should be different from each other

Now, to begin with mutation, select the first solution as the parent vector.

Say $x_1 = (1.7667, -4.1337)$ is the $Parent\ Vector$.

The corresponding function value is $f(x_1) = 20.2089$

For mutation, let target vector (selected randomly from the current population) is

$x_4 = (-4.3747, -4.7421)$

$Randomly\ Selected\ Solution_1 = x_5$

$Randomly\ Selected\ Solution_2 = x_3$

Let $Scale\ Factor = 0.5$

Now the $Trial\ Vector\ u_1$ is calculated as below:

$$u_{11} = x_{41} + 0.5 * (x_{51} - x_{31}) = -4.3747 + 0.5 * \big(-1.6587 - (-2.9232)\big)$$
$$= -3.7425$$

$$u_{12} = x_{42} + 0.5 * (x_{52} - x_{32}) = -4.7421 + 0.5 * \big(0.5680 - (-4.0439)\big)$$
$$= -2.4362$$

$$Trial\ Vector\ u_1 = (-3.7425, -2.4362)$$

At this point, it should be checked whether the trial vector u_1 is within the search range or not. As it is seen, $-5 \leq x'_{11}, x'_{12} \leq 5$, it can be accepted.

Crossover

In the DE crossover, an offspring is generated using the discrete recombination of the parent vector and trial vector. Consider the crossover probability $P_{cr} = 0.7$. Since we are going to calculate offspring corresponding to the parent x_1, so let us denote the offspring by x'_1.

$$Offspring\ x'_{1j} = \begin{cases} u_{1j}, & if\ j \in I \\ x_{1j}, & otherwise \end{cases}$$

Here I is the set of crossover points which depends upon the choice of crossover-type and the crossover probability. Binomial crossover is quite popular in DE, which is explained below:

In Binomial crossover, crossover points are selected from the set {1, 2, ……, problem dimension} with probability P_{cr}. This may lead to the situation when no point is selected for I (more probable for the small dimension problems, e.g., this problem is the only 2-dimensional problem). If this happens, i.e., $= \emptyset$, then there will not be any change in the offspring, and it will be the same as the parent. To

avoid this situation, I is always considered as a non-empty set by including a random point (say j_0) from the set {1, 2, ……, problem dimension} initially.

So let $I = \{2\}$

Now, other crossover points are selected using the following algorithm:

$$for\ j\ \in \{1,2,\dots, problem\ dimension\}$$

$$if\ rand(0,1) <\ P_{cr}\ and\ j \neq j_0$$

$$I = I\ \cup \{j\}$$

$$end$$

For this example, let $rand(0,1) = 0.67. Then\ I = \{1,2\}$. If the random number was greater than probability, 1 could not be added to the binominal vector, and the trial vector is eliminated.

Now the offspring x_1' will be formed from parent vector $x_1 = (1.7667, -4.1337)$ and trial vector $u_1 = (-3.7425, -2.4362)$. Obviously, since the set of crossover points, I has both the dimensions, both the variables of the offspring x_1' is are selected from the trial vector $u_1 = (-3.7425, -2.4362)$.

Thus the offspring $x_1' = (-3.7425, -2.4362)$.

At this point, we will decide whether in the population parent vector x_1 and offspring x_1' will survive. For that, we will compare the function values of x_0 and x_1'.

Since $f(x_1') = 19.9417$, which is better than $f(x_1) = 20.2089$, therefore for the next generation, in the population offspring x_1' will survive and offspring x_1 will die out or, in other words, x_1' will replace x_1. Thus new $x_1 = (-3.7425, -2.4362)$.

For solution x_2

Mutation

$$Target\ Vector =\ x_5$$

$$Randomly\ Selected\ Solution_1 = x_3$$

$$Randomly\ Selected\ Solution_2 = x_1$$

After applying the same procedure, the trial vector $u_2 = (-1.2491, -0.2358)$ which will be accepted as both the variables are within the search space.

Crossover

Let $I = \{1,2\}$ using the binomial crossover.

Therefore, offspring $x_2' = (-1.2491, -0.2358). f(x_2') = 1.6158.$

$f(x_2') < f(x_2)$ so the new $x_2 = (-1.2491, -0.2358)$

For solution x_3

Mutation

$$Target\ Vector = x_4$$

$$Randomly\ Selected\ Solution_1 = x_1$$

$$Randomly\ Selected\ Solution_2 = x_5$$

After applying the same procedure, the trial vector $u_3 = (-5.4167, -6.2443)$. Now since u_{31} and u_{32} (both are beyond the search space; therefore, we will pull the solution to the corresponding boundary of the search space, i.e.)

Trial vector becomes $u_3 = (-5, -5)$.

Crossover

Let $I = \{1,2\}$ using the binomial crossover.

Therefore, offspring $x_3' = (-5, -5). f(x_3') = 50.$

$f(x_3') > f(x_3)$ so the new $x_3 = (-2.9232, -4.0439)$ which is the parent vector.

Applying the same procedure to the remaining two solutions, we get

$$new\ x_4 = (-4.3748, -4.3402)\ with\ f(x_4) = 37.9765$$

$$new\ x_5 = (-1.6587, 0.5681)\ with\ f(x_5) = 3.0741$$

After the first generation, the updated population is;

Position	Function Value
$x_1 = (-3.7425, -2.4362)$	19.9417
$x_2 = (-1.2491, -0.2358)$	1.6158
$x_3 = (-2.9232, -4.0439)$	24.8982
$x_4 = (-4.3748, -4.3402)$	37.9765
$x_5 = (-1.6587, 0.5681)$	3.0741

The best solution after the first generation is $x_2 = (-1.2491, -0.2358) with\ f(x_2) = 1.6158$

Generation – 2

Position	Function Value
$x_1 = (-1.3603, -1.9917)$	5.8173
$x_2 = (-1.2491, -0.2358)$	1.6158
$x_3 = (-2.9232, -4.0439)$	24.8982
$x_4 = (0.5233, -0.0876)$	0.2815
$x_5 = (-0.4676, 0.7903)$	0.8433

It can be observed that solutions x_1, x_4 and x_5 are modified. Also, after the second generation, the best solution is $x_4 = (0.5233, -0.0876)$ with $f(x_4) = 0.2815$, which is better than that of first-generation.

B. PYTHON CODE and INPUT LIST

Main Code

```
#####################################################################
#                              AKAY BAYAT                           #
#                                                                   #
#                METU Aerospace Engineering MS Student              #
#                                                                   #
#HELICOPTER NOZZLE OPTIMIZING WITH DIFFERENTIAL EVOLUTION METHOD#
#####################################################################

from powerplant_drawing_with_rotate_module import nozzle_solution
from scipy.optimize import differential_evolution
from os import path

################## PARAMETERS RANGES ############################

inp=[]
inp.append((5,12))       #Lobed number ranges
inp.append((100,180))    #Nozzle lobed tangent radius ranges in mm
inp.append((330,370))    #Nozzle exit diameter ranges in mm
inp.append((80,150))     #Centerbody exit diameter ranges in mm

def f(inp):

    lobed_number=round(inp[0],0)
    nozzle_lobed_tangent_radius=round(inp[1],1)
    nozzle_exit_diameter=round(inp[2],1)
    centerbody_exit_diameter=round(inp[3],1)

    ################## BOUNDARY CONDITIONS ######################

    turbine_exit_mass_flow_rate=4.6
    turbine_exit_temperature=884
    ambient_pressure=95891
    turbine_exit_swirl_deg=10

    ################## MESH PARAMETERS ######################

    connector_dim=5
    extrude_initial_step_size=0.2
    extrude_growth_rate=1.2
    extrude_layer_number=20
    extrude_skew_angle_limit=0.9
    trex_growth_rate=1.3
    trex_layer_number=10
    trex_skew_limit=0.85
    trex_spacing=2
```

```
    ##################### SOLVER PARAMETERS ######################

    max_iteration_number=1000
    number_of_processors=4
    turbulence_model="ke-realizable"
    average_iter_number=50

    ############ CALL OBJECTIVE FUNCTION CALCULATOR #############

    dir_path = path.abspath(path.curdir)

    objective_function_result=objective_function(dir_path,
lobed_number,nozzle_lobed_tangent_radius,nozzle_exit_diameter,
centerbody_exit_diameter,connector_dim,extrude_initial_step_size,
extrude_growth_rate,extrude_layer_number,extrude_skew_angle_limit,t
rex_growth_rate,trex_layer_number,trex_skew_limit,trex_spacing,turb
ine_exit_mass_flow_rate,turbine_exit_temperature,
ambient_pressure,turbine_exit_swirl_deg,turbulence_model,
max_iteration_number, number_of_processors,
average_iter_number,func_eval)

    ################ PRINTING of RESULTS ########################

    form="%s\t"
    with open("optimization_trials.txt", "a") as txt_file:
        for i in range(len(inp)):
            txt_file.write(form % round(inp[i],0))
        for i in range(len(opt)):
            txt_file.write(form % opt[i])
        txt_file.write("\n")

    ############ RETURN OBJECTIVE FUNCTION RESULT ################

    return float(objective_function_result[2])


    ############### CALL OPTIMIZER ##############################

res = differential_evolution(f, inp,
strategy='best1bin',atol=10,tol=1,disp=True)
res=str(res)
print(res)
text_file = open("res.txt", "w")
n = text_file.write(res)
text_file.close()
```

Objective Function Calculator

```
#################################################################
#                            AKAY BAYAT                         #
#                 METU Aerospace Engineering MS Student         #
```

```
#                  Objective Function Calculator Module                #
########################################################################

from catia_functions import *
from pointwise_functions import *
from fluent_functions import *
import os
import time
import numpy as np
from os import path
from os import remove
from datetime import datetime
from os import listdir
import subprocess

def objective_function(dir_path,lobed_number, \
    nozzle_lobed_tangent_radius,nozzle_exit_diameter, \
    centerbody_exit_diameter,connector_dim, \
    extrude_initial_step_size,extrude_growth_rate, \
    extrude_layer_number,extrude_skew_angle_limit, \
    trex_growth_rate,trex_layer_number,trex_skew_limit, \
    trex_spacing,turbine_exit_mass_flow_rate, \
    turbine_exit_temperature, ambient_pressure, \
    te_swirl_deg,turbulence_model,max_iteration_number, \
    number_of_processors,average_iter_number,func_eval):

    original_path=dir_path
    os.chdir(dir_path)
    name= datetime.now().strftime("%Y%m%d%H%M%S")
    os.mkdir(name)
    solution_path=dir_path+"\\"+ name
    os.chdir(solution_path)
    nozzle_exit_lobed_diameter=80.0
    nozzle_length=115.0
    centerbody_length=200

    # Constant Parameters
    starting_angle=0.0
    lobed_deg=360.0/lobed_number
    nozzle_thickness=0.8
    turbine_exit_center=800.0
    turbine_exit_center_shift=turbine_exit_center+0.3
    turbine_exit_diameter=370
    centerbody_inlet_diameter=turbine_exit_diameter*0.73

    # Engine Dimensions
    compressor1_diameter=370
    compressor1_length=150
    compressor2_diameter=320
    compressor2_length=150
    gas_generator_turbine_diameter=370
    stage1_length=150
    stage2_length=100
    stage3_length=150
    stage4_length=100
```

```
    stage5_length=150

    #Nacelle Dimensions
    nacelle_center=turbine_exit_center-stage1_length-stage2_length-
stage3_length-\
                    stage4_length-stage5_length
    nacelle_diameter=800
    nacelle_length=750

    #Cooling Inlet
    cooling_inlet_center=nacelle_center+100
    cooling_inlet_deg=60
    cooling_inlet_width=100

    #Ejector Dimensions
    ejector_center=nacelle_center+nacelle_length
    ejector_diameter=nacelle_diameter*0.6
    ejector_length=600

    cooling_inlet_gage_pressure=0
    cooling_inlet_temperature=288
    ejector_exit_gage_pressure=0
    ejector_exit__temperature=288

    #GEOMETRY GENERATION

    ############# GENERIC PARAMETERS IN CAD ###############

    global hybridShapes1
    global hybridShapeFactory1
    global part
    CATIA=win32com.client.Dispatch("CATIA.Application")
    CATIA.Visible=True
    part1=CATIA.documents.add("Part").part
    part=CATIA.ActiveDocument.part
    sel=CATIA.ActiveDocument.Selection
    sel.clear
    body=CATIA.ActiveDocument.part.mainbody
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Add()
    hybridBody1.name="Nozzle"
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapeFactory1 = part1.HybridShapeFactory
    parameters1=part.Parameters
    sel = CATIA.ActiveDocument.Selection
    visPropertySet1 = sel.VisProperties
    visPropertySet1.Parent.SetShow(1)

    ################ REFERENCE PLANES ####################


xyplane=part1.createreferencefromgeometry(part1.originelements.plan
exy)

zxplane=part1.createreferencefromgeometry(part1.originelements.plan
```

```
ezx)

yzplane=part1.createreferencefromgeometry(part1.originelements.plan
eyz)

    ################ REFERENCE DIRECTIONS ##################
    x = part1.HybridShapeFactory.AddNewDirectionByCoord(1,0,0)
    y = part1.HybridShapeFactory.AddNewDirectionByCoord(0,1,0)
    z = part1.HybridShapeFactory.AddNewDirectionByCoord(0,0,1)
    revx = part1.HybridShapeFactory.AddNewDirectionByCoord(-1,0,0)
    revy = part1.HybridShapeFactory.AddNewDirectionByCoord(0,-1,0)
    revz = part1.HybridShapeFactory.AddNewDirectionByCoord(0,0,-1)

    hybridShapeFactory1.GSMVisibility(xyplane, 0)
    hybridShapeFactory1.GSMVisibility(zxplane, 0)
    hybridShapeFactory1.GSMVisibility(yzplane, 0)
    hybridShapePointCoord1 =
hybridShapeFactory1.AddNewPointCoord(0,0,0)
    hybridShapePointCoord1.name="refpt"
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Item("Nozzle")
    part1.InWorkObject = hybridShapePointCoord1
    hybridBody1.AppendHybridShape(hybridShapePointCoord1)


addpointcoord(turbine_exit_center+nozzle_length,0,0,"refpt","nozzle
_center",hybridShapeFactory1,part1,hybridShapes1)

addpointcoord(turbine_exit_center_shift,0,0,"refpt","split_plane_ce
nter",hybridShapeFactory1,part1,hybridShapes1)

addcirclecenter("nozzle_center",yzplane,nozzle_exit_diameter/2,"noz
zle_circle",hybridShapes1,part1,hybridShapeFactory1,hybridBody1)

addlinedir("nozzle_center",z,0,500,"center_line",hybridShapes1,hybr
idShapeFactory1,hybridBody1,part1)
    lnname=[]
    intname=[]
    scrcname=[]
    tangentname=[]
    splitname=[]
    splitdummyname=[]
    wshshell=win32com.client.Dispatch("WScript.Shell")
    wshshell.AppActivate("CATIA V5")

addlineangle("center_line","nozzle_center",0,500,starting_angle,"li
ne1",hybridShapes1,hybridShapeFactory1,hybridBody1,part1,yzplane)
    CATIA.StartCommand("Fit All In")

addlineangle("center_line","nozzle_center",0,500,lobed_deg+starting
_angle,"line2",hybridShapes1,hybridShapeFactory1,hybridBody1,part1,
yzplane)

addpointintersect("nozzle_circle","line1","int1",hybridShapes1,part
1,hybridShapeFactory1,hybridBody1)
```

```
addpointintersect("nozzle_circle","line2","int2",hybridShapes1,part
1,hybridShapeFactory1,hybridBody1)

addcirclecenter("int1",yzplane,nozzle_exit_lobed_diameter/2,"scrc1"
,hybridShapes1,part1,hybridShapeFactory1,hybridBody1)

addcirclecenter("int2",yzplane,nozzle_exit_lobed_diameter/2,"scrc2"
,hybridShapes1,part1,hybridShapeFactory1,hybridBody1)

addcirclebitangent("scrc1","scrc2",nozzle_lobed_tangent_radius,yzpl
ane,"nozzle_center","tangent1",hybridShapes1,part1,hybridShapeFacto
ry1,hybridBody1)

rotate("tangent1",lobed_number,lobed_deg,x,"tangent_rotate",part1,h
ybridBody1)

join(["tangent1","tangent_rotate"],"jointangent",hybridShapes1,hybr
idShapeFactory1,hybridBody1,part1)

    if lobed_number==4:
        split("scrc1","jointangent",-
1,"split1",hybridShapes1,hybridShapeFactory1,hybridBody1,part1)
    else:
        split("scrc1","jointangent",-
1,"split1",hybridShapes1,hybridShapeFactory1,hybridBody1,part1)

rotate("split1",lobed_number,lobed_deg,x,"split_rotate",part1,hybri
dBody1)

join(["split1","split_rotate","jointangent"],"nozzle_exit",hybridSh
apes1,hybridShapeFactory1,hybridBody1,part1)
    CATIA.StartCommand("Fit All In")

addpointcoord(turbine_exit_center,0,0,"refpt","turbine_exit_center"
,hybridShapeFactory1,part1,hybridShapes1)

addcirclecenter("turbine_exit_center",yzplane,turbine_exit_diameter
/2,"turbine_exit",hybridShapes1,part1,hybridShapeFactory1,hybridBod
y1)

addlinedir("turbine_exit_center",z,0,500,"turbine_center_line",hybr
idShapes1,hybridShapeFactory1,hybridBody1,part1)

addpointintersect("turbine_center_line","turbine_exit","closingpt1"
,hybridShapes1,part1,hybridShapeFactory1,hybridBody1)

addpointintersect("center_line","nozzle_exit","closingpt2",hybridSh
apes1,part1,hybridShapeFactory1,hybridBody1)

blend("turbine_exit","nozzle_exit","closingpt1","closingpt2","nozzl
e_inner",hybridBody1,part1)

thicksurface("nozzle_inner",nozzle_thickness,0,"nozzle",hybridShape
s1,part1,hybridShapeFactory1)
```

```
addplane("split_plane_center",yzplane,"turbine_exit_ref_plane",hybr
idShapeFactory1,part1)
    try:

splitbody("turbine_exit_ref_plane","nozzle_splitted",part1,hybridSh
apeFactory1)

extract("PartBody","Nozzle","nozzle_splitted","nozzle_extracted",pa
rt1,hybridShapeFactory1)
    except (RuntimeError, TypeError, NameError,Exception):

extract("PartBody","Nozzle","nozzle","nozzle_extracted",part1,hybri
dShapeFactory1)
    hide(["refpt"],CATIA)
    part1.update()


addpointcoord(turbine_exit_center_shift,0,0,"refpt","centerbody_inl
et_center",hybridShapeFactory1,part1,hybridShapes1)

addcirclecenter("centerbody_inlet_center",yzplane,centerbody_inlet_
diameter/2,"centerbody_inlet_circle",hybridShapes1,part1,hybridShap
eFactory1,hybridBody1)

addpointcoord(turbine_exit_center_shift+centerbody_length,0,0,"refp
t","centerbody_exit_center",hybridShapeFactory1,part1,hybridShapes1
)

addcirclecenter("centerbody_exit_center",yzplane,centerbody_exit_di
ameter/2,"centerbody_exit_circle",hybridShapes1,part1,hybridShapeFa
ctory1,hybridBody1)

multisectionssurface("centerbody_inlet_circle","centerbody_exit_cir
cle","centerbody_section",part1)

fill("centerbody_exit_circle","centerbody_exit_plane",hybridShapeFa
ctory1,part1)

addcirclecenter("split_plane_center",yzplane,turbine_exit_diameter/
2,"turbine_exit_shift",hybridShapes1,part1,hybridShapeFactory1,hybr
idBody1)

fill("turbine_exit_shift","turbine_exit_plane",hybridShapeFactory1,
part1)
    split("turbine_exit_plane","centerbody_inlet_circle",-
1,"turbine_exit_region",hybridShapes1,hybridShapeFactory1,hybridBod
y1,part1)

join(["turbine_exit_region","centerbody_section","centerbody_exit_p
lane"],"centerbody",hybridShapes1,hybridShapeFactory1,hybridBody1,p
art1)


    drawing_path_nozzle=solution_path+"\lobed_nozzle_cb.igs"
```

```
    if os.path.isfile("./lobed_nozzle_cb.igs"):
        remove(drawing_path_nozzle)
    CATIA.ActiveDocument.ExportData(drawing_path_nozzle, "igs")

hide(["nozzle_extracted","turbine_exit_region","centerbody","center
body_exit_plane"],CATIA)


addpointcoord(turbine_exit_center,0,turbine_exit_diameter/2+nozzle_
thickness,"refpt","engpt1",hybridShapeFactory1,part1,hybridShapes1)
    addlinedir("engpt1",x,0,-
stage1_length,"sec1",hybridShapes1,hybridShapeFactory1,hybridBody1,
part1)
    addpointcoord(turbine_exit_center-
stage1_length,0,turbine_exit_diameter/2+nozzle_thickness,"refpt","e
ngpt2",hybridShapeFactory1,part1,hybridShapes1)
    addpointcoord(-stage2_length,0,-
25,"engpt2","engpt3",hybridShapeFactory1,part1,hybridShapes1)

addspline("engpt2","engpt3","sec1","sec1","sec2",hybridShapeFactory
1,part1)
    addlinedir("engpt3",x,0,-
stage3_length,"sec3",hybridShapes1,hybridShapeFactory1,hybridBody1,
part1)
    addpointcoord(-
stage3_length,0,0,"engpt3","engpt4",hybridShapeFactory1,part1,hybri
dShapes1)
    addpointcoord(-
stage4_length,0,25,"engpt4","engpt5",hybridShapeFactory1,part1,hybr
idShapes1)

addspline("engpt4","engpt5","sec1","sec1","sec4",hybridShapeFactory
1,part1)
    addlinedir("engpt5",x,0,-
stage5_length,"sec5",hybridShapes1,hybridShapeFactory1,hybridBody1,
part1)

join(["sec1","sec2","sec3","sec4","sec5"],"engineline",hybridShapes
1,hybridShapeFactory1,hybridBody1,part1)
    addlinedir("refpt",x,0,-
stage3_length,"revaxis",hybridShapes1,hybridShapeFactory1,hybridBod
y1,part1)
    revolve("engineline","revaxis","engine_casing",CATIA)
    addpointcoord(-
stage1_length,0,0,"centerbody_inlet_center","sec1_center",hybridSha
peFactory1,part1,hybridShapes1)

addcirclecenter("sec1_center",yzplane,turbine_exit_diameter/2+nozzl
e_thickness,"sec1_circle",hybridShapes1,part1,hybridShapeFactory1,h
ybridBody1)

split("engine_casing","sec1_circle",1,"engine_casing_aft",hybridSha
pes1,hybridShapeFactory1,hybridBody1,part1)
    split("engine_casing","sec1_circle",-
```

```
1,"engine_casing_forward",hybridShapes1,hybridShapeFactory1,hybridB
ody1,part1)


addpointcoord(nacelle_center,0,0,"refpt","nacelle_center_pt",hybrid
ShapeFactory1,part1,hybridShapes1)

addcirclecenter("nacelle_center_pt",yzplane,nacelle_diameter/2,"nac
elle_circle",hybridShapes1,part1,hybridShapeFactory1,hybridBody1)

fill("nacelle_circle","nacelle_inlet_plane",hybridShapeFactory1,par
t1)

addcirclecenter("nacelle_center_pt",yzplane,turbine_exit_diameter/2
+nozzle_thickness,"engine_inlet_circle",hybridShapes1,part1,hybridS
hapeFactory1,hybridBody1)
    split("nacelle_inlet_plane","engine_inlet_circle",-
1,"nacelle_engine_area",hybridShapes1,hybridShapeFactory1,hybridBod
y1,part1)

extrude("nacelle_circle",x,nacelle_length,"nacelle_casing",hybridSh
apeFactory1,part1)

addpointcoord(nacelle_center+nacelle_length,0,0,"refpt","nacelle_ex
it_center_pt",hybridShapeFactory1,part1,hybridShapes1)

addcirclecenter("nacelle_exit_center_pt",yzplane,nacelle_diameter/2
,"nacelle_exit_circle",hybridShapes1,part1,hybridShapeFactory1,hybr
idBody1)

fill("nacelle_exit_circle","nacelle_exit_plane",hybridShapeFactory1
,part1)

addpointcoord(cooling_inlet_center,0,0,"refpt","cooling_inlet_cente
r_pt",hybridShapeFactory1,part1,hybridShapes1)

addcirclecenter("cooling_inlet_center_pt",yzplane,nacelle_diameter/
2,"cooling_inlet_circle",hybridShapes1,part1,hybridShapeFactory1,hy
bridBody1)

addlinedir("cooling_inlet_center_pt",z,0,500,"line3",hybridShapes1,
hybridShapeFactory1,hybridBody1,part1)

addlineangle("line3","cooling_inlet_center_pt",0,500,cooling_inlet_
deg,"line4",hybridShapes1,hybridShapeFactory1,hybridBody1,part1,yzp
lane)

join(["line3","line4"],"cooling_inlet_splitline",hybridShapes1,hybr
idShapeFactory1,hybridBody1,part1)
    split("cooling_inlet_circle","cooling_inlet_splitline",-
1,"cooling_inlet_circle_splitted",hybridShapes1,hybridShapeFactory1
,hybridBody1,part1)

extrude("cooling_inlet_circle_splitted",x,cooling_inlet_width,"cool
ing_inlet_area",hybridShapeFactory1,part1)
```

```
thicksurface("cooling_inlet_area",nozzle_thickness,nozzle_thickness
,"cooling_inlet_area_ref",hybridShapes1,part1,hybridShapeFactory1)

extract("PartBody","Nozzle","cooling_inlet_area_ref","cooling_inlet
_area_ref_extracted",part1,hybridShapeFactory1)
    split("nacelle_casing","cooling_inlet_area_ref_extracted",-
1,"nacelle_casing_splitted",hybridShapes1,hybridShapeFactory1,hybri
dBody1,part1)

extrude("cooling_inlet_circle_splitted",x,cooling_inlet_width,"cool
ing_inlet_area",hybridShapeFactory1,part1)

    #Add Ejector Section


addpointcoord(ejector_center,0,0,"refpt","ejector_center_pt",hybrid
ShapeFactory1,part1,hybridShapes1)

addcirclecenter("ejector_center_pt",yzplane,ejector_diameter/2,"eje
ctor_circle",hybridShapes1,part1,hybridShapeFactory1,hybridBody1)

extrude("ejector_circle",x,ejector_length,"ejector_casing",hybridSh
apeFactory1,part1)

addpointcoord(ejector_center+ejector_length,0,0,"refpt","ejector_ex
it_center_pt",hybridShapeFactory1,part1,hybridShapes1)

addcirclecenter("ejector_exit_center_pt",yzplane,ejector_diameter/2
,"ejector_exit_circle",hybridShapes1,part1,hybridShapeFactory1,hybr
idBody1)

fill("ejector_exit_circle","ejector_exit_plane",hybridShapeFactory1
,part1)
    split("nacelle_exit_plane","ejector_circle",-
1,"nacelle_ejector_area",hybridShapes1,hybridShapeFactory1,hybridBo
dy1,part1)
    dir_path = path.abspath(path.curdir)
    drawing_path_nacelle=solution_path+"\engine_section.igs"
    part1.Update ()
    if os.path.isfile("./engine_section.igs"):
        remove(drawing_path_nacelle)
    CATIA.ActiveDocument.ExportData(drawing_path_nacelle, "igs")
    catpartpath=dir_path+"\ nozzle.CATPart"
    CATIA.ActiveDocument.SaveAs(catpartpath)
    CATIA.ActiveDocument.Close()


    #CFD MESH WITH POINTWISE


    meshing_path=solution_path+"\\nacelle.glf"
    export_folder_path=solution_path
    mesh_status_path=solution_path+"\\mesh_status.txt"
```

```
    ############### GLYPH FILE GENERATION ##################

    if os.path.isfile("./nacelle.glf"):
        remove(meshing_path)
    os.chdir(solution_path)

    if os.path.isfile("./mesh_status.txt"):
        remove(mesh_status_path)
    os.chdir(solution_path)

    adddatabase(drawing_path_nozzle,"nacelle.glf")
    adddatabase(drawing_path_nacelle,"nacelle.glf")

    f1=["engine_casing_forward","engine_casing_forward-1"]
    splitdb("engine_casing_aft-quilt",f1,"nacelle.glf")
    splitdb("nacelle_casing_splitted-
quilt",["cooling_inlet_area"],"nacelle.glf")
    splitdb("engine_casing_aft-model",["engine_casing_aft-quilt-
split-2","nacelle_casing_splitted-quilt-split-2"],"nacelle.glf")
    unsmeshspacing(["centerbody-model","nozzle_extracted-
model"],5.0,0.95,"AdvancingFront",5.0,0,0,"nacelle.glf")
    domaincheck("nacelle.glf")
    unsmeshspacing(["engine_casing_aft-model-split-
1"],15.0,0.9,"AdvancingFront",15.0,0,0,"nacelle.glf")


deldomain(turbine_exit_center_shift,turbine_exit_center_shift,turbi
ne_exit_diameter/2,turbine_exit_diameter/2+nozzle_thickness,"nacell
e.glf")

    sec1=turbine_exit_center_shift-stage1_length
    sec1_radius=turbine_exit_diameter/2+nozzle_thickness



createconnectoratan2(sec1,sec1_radius,90,turbine_exit_center_shift,
sec1_radius,90,"nacelle.glf")

createconnectoratan2(sec1,sec1_radius,270,turbine_exit_center_shift
,sec1_radius,270,"nacelle.glf")

createdomainfromconnectors(sec1,sec1_radius,90,270,turbine_exit_cen
ter_shift,sec1_radius,90,270,"nacelle.glf")

createdomainfromconnectors(sec1,sec1_radius,269,91,turbine_exit_cen
ter_shift,sec1_radius,269,91,"nacelle.glf")

createdomainfromconnectors(cooling_inlet_center,nacelle_diameter/2,
0,360,cooling_inlet_center+cooling_inlet_width,nacelle_diameter/2,0
,360,"nacelle.glf")

connectordimension(cooling_inlet_center,nacelle_diameter/2,0,360,co
oling_inlet_center+cooling_inlet_width,nacelle_diameter/2,0,360,5,"
nacelle.glf")
```

```
connectordimension(turbine_exit_center_shift,centerbody_inlet_diame
ter/2,0,360,turbine_exit_center_shift,turbine_exit_diameter/2,0,360
,connector_dim,"nacelle.glf")

    projectdomains("nacelle.glf")
    joindomains(sec1,turbine_exit_center,sec1_radius,"nacelle.glf")
    projectdomains("nacelle.glf")

projectspesific(sec1,turbine_exit_center,sec1_radius,"engine_casing
_aft-quilt-split-2","nacelle.glf")
    resetdimension(sec1,sec1_radius,5,"nacelle.glf")
    merge(0.3,"nacelle.glf")
    joinconnectors(turbine_exit_center_shift-
1.0,turbine_exit_center_shift+1.0,sec1_radius,"nacelle.glf")
    removebreakpoints(turbine_exit_center_shift-
1.0,turbine_exit_center_shift+1.0,sec1_radius,"nacelle.glf")

createextrude(0,extrude_initial_step_size,extrude_growth_rate,extru
de_layer_number,extrude_skew_angle_limit,"nacelle.glf")

checkblockempty(0,extrude_initial_step_size,extrude_growth_rate,ext
rude_layer_number,"nacelle.glf")
    createassemblespecial("nacelle.glf")
    trexblock("blk-
2",trex_layer_number,1,trex_growth_rate,trex_skew_limit,trex_spacin
g,"nacelle.glf")

defineboundaries(cooling_inlet_center,cooling_inlet_center+cooling_
inlet_width,turbine_exit_center,turbine_exit_center_shift,turbine_e
xit_diameter/2,turbine_exit_diameter/2,ejector_center+ejector_lengt
h,ejector_center+ejector_length,"nacelle.glf")


    # Export Cae
    export_folder_path=solution_path
    mesh_file_path=export_folder_path+"\\nacelle.cas"
    exportcae(mesh_file_path,"nacelle.glf")
    mesh_status(export_folder_path.replace("\\",
"/"),"mesh_status.txt","nacelle.glf")

#    Save and Exit
    pw_save_path=export_folder_path + "\\nacelle.pw"
    save_exit(pw_save_path,"nacelle.glf")
    file2=open("pointwise_nacelle_glf.bat", "w")
    quote=' "'
    glf_path=solution_path + "\\" + "nacelle.glf"
    file2.write("Start" + quote + "Pointwise.exe" +quote + quote +
glf_path + quote+"\n")
    file2.close()

    FNULL = open(os.devnull, 'w')
    p=subprocess.call(r"pointwise_nacelle_glf.bat",stdout=FNULL,
stderr=subprocess.STDOUT)

    #CFD SOLUTION WITH FLUENT
```

```
    os.chdir(export_folder_path)
    journal_path=export_folder_path+ "\\nacelle.jou"
    if os.path.isfile("./nacelle.jou"):
        remove(journal__path)

create_journal_file(mesh_file_path,turbulence_model,turbine_exit_ma
ss_flow_rate,turbine_exit_temperature,te_swirl_deg,turbine_exit_cen
ter,cooling_inlet_gage_pressure,cooling_inlet_temperature,ejector_e
xit_gage_pressure,ejector_exit__temperature,max_iteration_number,"n
acelle.jou")

    file3=open("fluent_run.bat", "w")
    quote='"'
    file3.write( quote + "C:\Program Files\ANSYS" + ' ' +
"Inc\\v150\\"+ "fluent\\" + "ntbin\win64\\"+"fluent.exe" +quote +"
3d -t" + str(number_of_processors)+" -i nacelle.jou")
    file3.close()

    nacelle_path=export_folder_path+ "\\nacelle.cas"
    fluent_bat_path=quote + export_folder_path + "\\fluent_run.bat"
+ quote
    while not os.path.exists(mesh_status_path):
        time.sleep(10)

    p=subprocess.call(r"fluent_run.bat",stdout=FNULL,
stderr=subprocess.STDOUT)

    os.chdir(solution_path)


    mass_file_path=export_folder_path+"//cooling_inlet_mass"
    result_path=export_folder_path+"//mass_flow_rates"

    while not os.path.exists(result_path):
        time.sleep(10)
    cooling_inlet=open('cooling_inlet_mass', 'r').readlines()
    turbine_exit=open('turbine_exit_ptot', 'r').readlines()
    N=len(cooling_inlet)-1
    cooling_inlet_fr=0
    turbine_exit_ptot=0
    for i in range(N-average_iter_number+1,N):
        ci=cooling_inlet[i].split()
        te=turbine_exit[i].split()
        l1=ci[1:2]
        l2=te[1:2]
        cooling_inlet_fr=cooling_inlet_fr+np.float_(l1)
        turbine_exit_ptot=turbine_exit_ptot+np.float_(l2)
    cooling_inlet_fr_ave=cooling_inlet_fr/average_iter_number
    turbine_exit_ptot_ave=turbine_exit_ptot/average_iter_number

optimizer_ctrl=((turbine_exit_ptot_ave+ambient_pressure)/ambient_pr
essure+ turbine_exit_mass_flow_rate/cooling_inlet_fr_ave)*100
```

```
    print(" ", func_eval,"     ",lobed_number,"
",nozzle_lobed_tangent_radius,"         ",nozzle_exit_diameter, \
          "       ", centerbody_exit_diameter,"
",round(cooling_inlet_fr_ave[0],2),"
",round(turbine_exit_ptot_ave[0],1) ,\
          "          ",round(optimizer_ctrl[0],1),)
    os.chdir(original_path)
    return
[cooling_inlet_fr_ave,turbine_exit_ptot_ave,optimizer_ctrl]
```

CAD Functions

```
import win32com.client


#Define Functions

def
addpointcoord(x,y,z,refpt,ptname,hybridShapeFactory1,part1,hybridSh
apes1):
    hybridShapePointCoord1 =
hybridShapeFactory1.AddNewPointCoord(x,y,z)
    hybridShapePointCoord1.name=ptname
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Item("Nozzle")
    part1.InWorkObject = hybridShapePointCoord1
    hybridShapePointCoord2 = hybridShapes1.Item(refpt)
    reference1 =
part1.CreateReferenceFromObject(hybridShapePointCoord2)
    hybridShapePointCoord1.PtRef = reference1
    hybridBody1.AppendHybridShape(hybridShapePointCoord1)
    part1.Update()

def addplane(refpoint,
refplane,itemname,hybridShapeFactory1,part1):
    reference1=refplane
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Item("Nozzle")
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapePointCoord1 = hybridShapes1.Item(refpoint)
    reference2 =
part1.CreateReferenceFromObject(hybridShapePointCoord1)
    hybridShapePlaneOffsetPt1 =
hybridShapeFactory1.AddNewPlaneOffsetPt(reference1, reference2)
    hybridShapePlaneOffsetPt1.name=itemname
    hybridBody1.AppendHybridShape(hybridShapePlaneOffsetPt1)
    part1.InWorkObject = hybridShapePlaneOffsetPt1
    part1.Update
```

```
def
addpointintersect(item1,item2,intname,hybridShapes1,part1,hybridSha
peFactory1,hybridBody1):
    hybridShapeLinePtDir1 = hybridShapes1.Item(item1)
    ref1 = part1.CreateReferenceFromObject(hybridShapeLinePtDir1)
    hybridShapeLinePtDir1 = hybridShapes1.Item(item2)
    ref2 = part1.CreateReferenceFromObject(hybridShapeLinePtDir1)
    hybridShapeIntersection1 =
hybridShapeFactory1.AddNewIntersection(ref1, ref2)
    hybridShapeIntersection1.PointType = 1
    hybridShapeIntersection1.name=intname
    hybridBody1.AppendHybridShape(hybridShapeIntersection1)
    part1.InWorkObject = hybridShapeIntersection1
    hybridShapeFactory1.GSMVisibility(ref1, 0)
    hybridShapeFactory1.GSMVisibility(ref2, 0)
    part1.Update ()

def
addcirclecenter(centerpt,plane,r,circlename,hybridShapes1,part1,hyb
ridShapeFactory1,hybridBody1):
    centerpoint = hybridShapes1.Item(centerpt)
    ref1 = part1.CreateReferenceFromObject(centerpoint)
    hybridShapeCircleCtrRad1 =
hybridShapeFactory1.AddNewCircleCtrRad(ref1, plane,False, r)
    hybridShapeCircleCtrRad1.name=circlename
    hybridBody1.AppendHybridShape(hybridShapeCircleCtrRad1)
    part1.InWorkObject = hybridShapeCircleCtrRad1
    hybridShapeFactory1.GSMVisibility(ref1, 0)
    part1.Update()

def
addcirclebitangent(item1,item2,r,plane,planept,tangentname,hybridSh
apes1,part1,hybridShapeFactory1,hybridBody1):
    hybridShapeCircleCtrRad1 = hybridShapes1.Item(item1)
    ref1 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad1)
    hybridShapeCircleCtrRad2 = hybridShapes1.Item(item2)
    ref2 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad2)
    ref3=plane
    refpoint=hybridShapes1.Item(planept)
    ref4=part1.CreateReferenceFromObject(refpoint)
    supportplane= hybridShapeFactory1.AddNewPlaneOffsetPt(ref3,
ref4)
    hybridShapeCircleBitangentRadius1 =
hybridShapeFactory1.AddNewCircleBitangentRadius(ref1, ref2,
supportplane, r, -1, -1)
    hybridShapeCircleBitangentRadius1.DiscriminationIndex = 1
    hybridShapeCircleBitangentRadius1.BeginOfCircle = 1
    hybridShapeCircleBitangentRadius1.TangentOrientation1 = -1
    hybridShapeCircleBitangentRadius1.TangentOrientation2 = -1
    hybridShapeCircleBitangentRadius1.TrimMode = 0
    hybridShapeCircleBitangentRadius1.name=tangentname

hybridBody1.AppendHybridShape(hybridShapeCircleBitangentRadius1)
```

```
    part1.InWorkObject = hybridShapeCircleBitangentRadius1
    hybridShapeFactory1.GSMVisibility(ref1, 0)
    hybridShapeFactory1.GSMVisibility(ref2, 0)
    hybridShapeFactory1.GSMVisibility(ref3, 0)
    hybridShapeFactory1.GSMVisibility(ref4, 0)
    part1.Update()

def
thicksurface(refsurface,thickness1,thickness2,itemname,hybridShapes
1,part1,hybridShapeFactory1):
    bodies1 = part1.Bodies
    body1 = bodies1.Item("PartBody")
    part1.InWorkObject = body1
    shapeFactory1 = part1.ShapeFactory
    reference1 = part1.CreateReferenceFromName("")
    thickSurface1 = shapeFactory1.AddNewThickSurface(reference1, 0,
thickness1, 0.000000)
    hybridShapeBlend1 = hybridShapes1.Item(refsurface)
    reference2 = part1.CreateReferenceFromObject(hybridShapeBlend1)
    thickSurface1.Surface = reference2
    length1=thickSurface1.BotOffset
    length1.Value=thickness2
    thickSurface1.name=itemname
    hybridShapeFactory1.GSMVisibility(reference2, 0)
    part1.update()

def splitbody(splititem,itemname,part1,hybridShapeFactory1):
    bodies1 = part1.Bodies
    body1 = bodies1.Item("PartBody")
    part1.InWorkObject = body1
    shapeFactory1 = part1.ShapeFactory
    reference1 = part1.CreateReferenceFromName("")
    split1 = shapeFactory1.AddNewSplit(reference1, 0)
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Item("Nozzle")
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapePlaneOffsetPt1 = hybridShapes1.Item(splititem)
    reference2 =
part1.CreateReferenceFromObject(hybridShapePlaneOffsetPt1)
    split1.Surface = reference2
    split1.name=itemname
    hybridShapeFactory1.GSMVisibility(reference2, 0)
    part1.Update()


def
split(item1,item2,orientation,splitname,hybridShapes1,hybridShapeFa
ctory1,hybridBody1,part1):
    hybridShapeCircleCtrRad1 = hybridShapes1.Item(item1)
    ref1 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad1)
    hybridShapeCircleBitangentRadius1 = hybridShapes1.Item(item2)
    ref2 =
part1.CreateReferenceFromObject(hybridShapeCircleBitangentRadius1)
    hybridShapeSplit1 = hybridShapeFactory1.AddNewHybridSplit(ref1,
```

```
ref2,orientation)
    hybridShapeSplit1.name=splitname
    hybridShapeFactory1.GSMVisibility(ref1, 0)
    hybridShapeFactory1.GSMVisibility(ref2, 0)
    hybridBody1.AppendHybridShape(hybridShapeSplit1)
    part1.InWorkObject = hybridShapeSplit1
    hybridShapeFactory1.GSMVisibility(ref1, 0)
    hybridShapeFactory1.GSMVisibility(ref2, 0)
    part1.Update()

def
addlinedir(centerpt,drct,start,end,lnname,hybridShapes1,hybridShape
Factory1,hybridBody1,part1):
    centerpoint = hybridShapes1.Item(centerpt)
    ref1 = part1.CreateReferenceFromObject(centerpoint)
    hybridShapeLinePtDir1 =
hybridShapeFactory1.AddNewLinePtDir(ref1, drct, start,end, False)
    hybridShapeLinePtDir1.name=lnname
    hybridBody1.AppendHybridShape(hybridShapeLinePtDir1)
    part1.InWorkObject = hybridShapeLinePtDir1
    hybridShapeFactory1.GSMVisibility(ref1, 0)
    part1.Update()

def
addspline(point1,point2,tngt1,tngt2,spname,hybridShapeFactory1,part
1):
    hybridShapeSpline1 = hybridShapeFactory1.AddNewSpline()
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Item("Nozzle")
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapePointCoord1 = hybridShapes1.Item(point1)
    reference1 =
part1.CreateReferenceFromObject(hybridShapePointCoord1)
    hybridShapeLineExplicit1 = hybridShapes1.Item(tngt1)
    reference2 =
part1.CreateReferenceFromObject(hybridShapeLineExplicit1)

hybridShapeSpline1.AddPointWithConstraintFromCurve(reference1,refer
ence2, 1.000000, -1, 1)
    hybridShapePointCoord2 = hybridShapes1.Item(point2)
    reference3 =
part1.CreateReferenceFromObject(hybridShapePointCoord2)
    hybridShapeLineExplicit2 = hybridShapes1.Item(tngt2)
    reference4 =
part1.CreateReferenceFromObject(hybridShapeLineExplicit2)
    hybridShapeSpline1.AddPointWithConstraintFromCurve(reference3,
reference4, 1.000000, -1, 1)
    hybridShapeSpline1.name=spname
    hybridBody1.AppendHybridShape(hybridShapeSpline1)
    part1.InWorkObject = hybridShapeSpline1
    hybridShapeFactory1.GSMVisibility(reference1, 0)
    hybridShapeFactory1.GSMVisibility(reference3, 0)
    part1.Update()

def addlineangle(centerln,centerpt,start,end,angle,lnname,
```

```
    hybridShapes1,hybridShapeFactory1,hybridBody1,part1,yzplane):
    hybridShapeLinePtDir1 = hybridShapes1.Item(centerln)
    ref1 = part1.CreateReferenceFromObject(hybridShapeLinePtDir1)
    ref2 = yzplane
    centerpoint = hybridShapes1.Item(centerpt)
    ref3 = part1.CreateReferenceFromObject(centerpoint)
    hybridShapeLineAngle1 = hybridShapeFactory1.AddNewLineAngle \
    (ref1, ref2, ref3, False,start,end,angle,False)
    hybridShapeLineAngle1.name=lnname
    hybridBody1.AppendHybridShape(hybridShapeLineAngle1)
    part1.InWorkObject = hybridShapeLineAngle1
    hybridShapeFactory1.GSMVisibility(ref1, 0)
    hybridShapeFactory1.GSMVisibility(ref2, 0)
    hybridShapeFactory1.GSMVisibility(ref3, 0)
    part1.Update()

def
join(f,joinname,hybridShapes1,hybridShapeFactory1,hybridBody1,part1
):
    element1 = hybridShapes1.Item(f[0])
    ref1 = part1.CreateReferenceFromObject(element1)
    element2 = hybridShapes1.Item(f[1])
    ref2 = part1.CreateReferenceFromObject(element2)
    hybridShapeAssemble1 = hybridShapeFactory1.AddNewJoin(ref1,
ref2)
    for i in range(2,len(f)):
        element3 = hybridShapes1.Item(f[i])
        ref3 = part1.CreateReferenceFromObject(element3)
        hybridShapeAssemble1.AddElement(ref3)
        hybridShapeFactory1.GSMVisibility(ref3, 0)
    hybridShapeAssemble1.SetConnex(0)
    hybridShapeFactory1.GSMVisibility(ref1, 0)
    hybridShapeFactory1.GSMVisibility(ref2, 0)
    hybridShapeAssemble1.SetManifold(0)
    hybridBody1.AppendHybridShape(hybridShapeAssemble1)
    part1.InWorkObject = hybridShapeAssemble1
    hybridShapeAssemble1.name=joinname
    part1.Update()

def
rotate(item,instance,angle,rotationaxis,rtname,part1,hybridBody1):
    shapeFactory1 = part1.ShapeFactory
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapeitem1 = hybridShapes1.Item(item)
    reference1 = part1.CreateReferenceFromName("")
    reference2 = part1.CreateReferenceFromName("")
    circPattern1 =
shapeFactory1.AddNewSurfacicCircPattern(hybridShapeitem1, 1, 2,
20.000000, 45.000000, 1, 1, reference1, reference2, True, 0.000000,
True, False)
    circPattern1.SetRotationAxis(rotationaxis)
    circPattern1.name=rtname
    angularRepartition1 = circPattern1.AngularRepartition
    intParam1 = angularRepartition1.InstancesCount
    intParam1.Value = instance
```

```
    angularRepartition2 = circPattern1.AngularRepartition
    angle1 = angularRepartition2.AngularSpacing
    angle1.Value = angle

def
blend(item1,item2,closingpt1,closingpt2,blendname,hybridBody1,part1
):
    hybridShapeFactory1 = part1.HybridShapeFactory
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapeCircleCtrRad1 = hybridShapes1.Item(item1)
    part1.Update
    hybridShapeBlend1 = hybridShapeFactory1.AddNewBlend()
    hybridShapeBlend1.Coupling = 1
    reference2 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad1)
    hybridShapeBlend1.SetCurve(1, reference2)
    hybridShapeBlend1.SetOrientation( 1, 1)
    hybridShapeIntersection1 = hybridShapes1.Item(closingpt1)
    reference3 =
part1.CreateReferenceFromObject(hybridShapeIntersection1)
    hybridShapeBlend1.SetClosingPoint(1, reference3)
    hybridShapeAssemble1 = hybridShapes1.Item(item2)
    reference4 =
part1.CreateReferenceFromObject(hybridShapeAssemble1)
    hybridShapeBlend1.SetCurve (2, reference4)
    hybridShapeBlend1.SetOrientation(2, 1)
    hybridShapeIntersection2 = hybridShapes1.Item(closingpt2)
    reference5 =
part1.CreateReferenceFromObject(hybridShapeIntersection2)
    hybridShapeBlend1.SetClosingPoint(2, reference5)
    hybridShapeBlend1.SmoothAngleThresholdActivity = False
    hybridShapeBlend1.SmoothDeviationActivity = False
    hybridShapeBlend1.RuledDevelopableSurface = False
    hybridShapeFactory1.GSMVisibility(reference2, 0)
    hybridShapeFactory1.GSMVisibility(reference3, 0)
    hybridShapeFactory1.GSMVisibility(reference4, 0)
    hybridShapeFactory1.GSMVisibility(reference5, 0)
    hybridBody1.AppendHybridShape(hybridShapeBlend1)
    hybridShapeBlend1.name=blendname

def multisectionssurface(item1,item2,itemname,part1):
    hybridShapeFactory1 = part1.HybridShapeFactory
    hybridShapeDirection1 =
hybridShapeFactory1.AddNewDirectionByCoord(1.000000, 2.000000,
3.000000)
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Item("Nozzle")
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapeCircleCtrRad1 = hybridShapes1.Item(item1)
    reference1 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad1)
    hybridShapeExtremum1 =
hybridShapeFactory1.AddNewExtremum(reference1,
hybridShapeDirection1, 1)
    hybridBody1.AppendHybridShape(hybridShapeExtremum1)
```

```
    part1.InWorkObject = hybridShapeExtremum1
    part1.Update
    hybridShapeDirection2 =
hybridShapeFactory1.AddNewDirectionByCoord(1.000000, 2.000000,
3.000000)
    hybridShapeCircleCtrRad2 = hybridShapes1.Item(item2)
    reference2 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad2)
    hybridShapeExtremum2 =
hybridShapeFactory1.AddNewExtremum(reference2,
hybridShapeDirection2, 1)
    hybridBody1.AppendHybridShape(hybridShapeExtremum2)
    part1.InWorkObject = hybridShapeExtremum2
    part1.Update
    hybridShapeLoft1 = hybridShapeFactory1.AddNewLoft()
    hybridShapeLoft1.SectionCoupling = 1
    hybridShapeLoft1.Relimitation = 1
    hybridShapeLoft1.CanonicalDetection = 2
    reference3 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad1)
    reference4 =
part1.CreateReferenceFromObject(hybridShapeExtremum1)
    hybridShapeLoft1.AddSectionToLoft(reference3, 1, reference4)
    reference5 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad2)
    reference6 =
part1.CreateReferenceFromObject(hybridShapeExtremum2)
    hybridShapeLoft1.AddSectionToLoft(reference5, 1, reference6)
    originElements1 = part1.OriginElements
    hybridShapePlaneExplicit1 = originElements1.PlaneYZ
    reference7 =
part1.CreateReferenceFromObject(hybridShapePlaneExplicit1)
    hybridShapeLoft1.SetStartSectionTangent(reference7)
    hybridBody1.AppendHybridShape(hybridShapeLoft1)
    part1.InWorkObject = hybridShapeLoft1
    hybridShapeLoft1.name=itemname
    hybridShapeFactory1.GSMVisibility(reference1, 0)
    hybridShapeFactory1.GSMVisibility(reference2, 0)
    hybridShapeFactory1.GSMVisibility(reference3, 0)
    hybridShapeFactory1.GSMVisibility(reference4, 0)
    hybridShapeFactory1.GSMVisibility(reference5, 0)
    hybridShapeFactory1.GSMVisibility(reference6, 0)

def fill(ref1,itemname,hybridShapeFactory1,part1):
    hybridShapeFill1 = hybridShapeFactory1.AddNewFill()
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Item("Nozzle")
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapeCircleCtrRad1 = hybridShapes1.Item(ref1)
    reference1 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad1)
    hybridShapeFill1.AddBound(reference1)
    hybridShapeFill1.Continuity = 1
    hybridBody1.AppendHybridShape(hybridShapeFill1)
    part1.InWorkObject = hybridShapeFill1
```

```
    hybridShapeFill1.name=itemname
    hybridShapeFactory1.GSMVisibility(reference1, 0)
    part1.Update

def
extract(mainbody,geo_set,item,itemname,part1,hybridShapeFactory1):
    bodies1 = part1.Bodies
    body1 = bodies1.Item(mainbody)
    shapes1 = body1.Shapes
    reference1 = shapes1.Item(item)
    hybridShapeExtract1 =
hybridShapeFactory1.AddNewExtract(reference1)
    hybridShapeExtract1.PropagationType = 3
    hybridShapeExtract1.ComplementaryExtract = False
    hybridShapeExtract1.IsFederated = False
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Item(geo_set)
    hybridBody1.AppendHybridShape(hybridShapeExtract1)
    part1.InWorkObject = hybridShapeExtract1
    hybridShapeExtract1.name=itemname
    hybridShapeFactory1.GSMVisibility(reference1, 0)

def
extrude(item,direction,ext_length,itemname,hybridShapeFactory1,part
1):
    hybridShapeDirection1=direction
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Item("Nozzle")
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapeCircleCtrRad1 = hybridShapes1.Item(item)
    reference1 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad1)
    hybridShapeExtrude1 =
hybridShapeFactory1.AddNewExtrude(reference1, ext_length, 0.000000,
hybridShapeDirection1)
    hybridShapeExtrude1.SymmetricalExtension = 0
    hybridShapeExtrude1.name=itemname
    hybridBody1.AppendHybridShape(hybridShapeExtrude1)
    part1.InWorkObject = hybridShapeExtrude1
    hybridShapeFactory1.GSMVisibility(reference1, 0)

def revolve(revelement,revaxis,revname,CATIA):
    partDocument1 = CATIA.ActiveDocument
    part1 = partDocument1.Part
    hybridShapeFactory1 = part1.HybridShapeFactory
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Item("Nozzle")
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapeAssemble1 = hybridShapes1.Item(revelement)
    reference1 =
part1.CreateReferenceFromObject(hybridShapeAssemble1)
    hybridShapeLinePtDir1 = hybridShapes1.Item(revaxis)
    reference2 =
part1.CreateReferenceFromObject(hybridShapeLinePtDir1)
    hybridShapeRevol1 = hybridShapeFactory1.AddNewRevol(reference1,
```

```
360.000000, 0.000000, reference2)
    hybridShapeRevol1.name=revname
    hybridBody1.AppendHybridShape(hybridShapeRevol1)
    part1.InWorkObject = hybridShapeRevol1
    hybridShapeFactory1.GSMVisibility(reference1, 0)
    hybridShapeFactory1.GSMVisibility(reference2, 0)

def hide(itemname,CATIA):
    partDocument1 = CATIA.ActiveDocument
    selection1 = partDocument1.Selection
    visPropertySet1 = selection1.VisProperties
    part1 = partDocument1.Part
    hybridBodies1 = part1.HybridBodies
    hybridBody1 = hybridBodies1.Item("Nozzle")
    hybridShapes1 = hybridBody1.HybridShapes
    for i in range(0,len(itemname)):
        hybridShapeExtract1 = hybridShapes1.Item(itemname[i])
        hybridShapes1 = hybridShapeExtract1.Parent
        selection1.Add(hybridShapeExtract1)
    visPropertySet1 = visPropertySet1.Parent
    visPropertySet1.SetShow(1)
    selection1.Clear

def disassemble(item):
    sel.clear
    sel.Add(hybridShapes1.Item(item))
    wshshell=win32com.client.Dispatch("WScript.Shell")
    wshshell.AppActivate("CATIA V5")
    CATIA.StartCommand("Disassemble")
    sel.clear
    sel.Add(hybridShapes1.Item(item))
    wshshell.SendKeys('{ENTER}'), True

def fillet():
    reference1 =
part1.CreateReferenceFromBRepName("REdge:(Edge:(Face:(Brp:(GSMFill.
5;1);None:();Cf11:()); "\

"Face:(Brp:(GSMExtrude.4;0:(Brp:(GSMCircle.13)));None:();Cf11:());N
one:(Limits1:();Limits2:());Cf11:());WithTemporaryBody;" \

"WithoutBuildError;WithSelectingFeatureSupport;MFBRepVersion_CXR15)
", hybridShapeAssemble1)

def
extrapolation(boundaryname,surfacename,extrapolname,hybridShapes1,h
ybridShapeFactory1,hybridBody1,part1):
    hybridShapes1 = hybridBody1.HybridShapes
    hybridShapeCircleCtrRad1 = hybridShapes1.Item(boundaryname)
    reference1 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad1)
    hybridShapeBlend1 = hybridShapes1.Item(surfacename)
    reference2 = part1.CreateReferenceFromObject(hybridShapeBlend1)
    hybridShapeExtrapol1 =
hybridShapeFactory1.AddNewExtrapolLength(reference1,
```

```
reference2,1.000000)
    hybridShapeExtrapol1.BorderType = 1
    hybridShapeExtrapol1.SetAssemble(True)
    hybridShapeExtrapol1.PropagationMode = 0
    hybridShapeExtrapol1.ExtendEdgesMode = False
    hybridShapeExtrapol1.ConstantLengthMode = False
    reference3 =
part1.CreateReferenceFromObject(hybridShapeCircleCtrRad1)
    hybridShapeExtrapol1.SetBoundary(1, reference3)
    hybridShapeExtrapol1.SetLimitType(1, 0)
    hybridShapeExtrapol1.SetLength(1,0)
    hybridShapeExtrapol1.name=extrapolname
```

Grid Generator Functions

```
######################################################################
#                          AKAY BAYAT                                #
#              METU Aerospace Engineering MS Student                 #
#                    Grid Generator Functions                        #
######################################################################

global file1
quotes='"'
def openfile(filename):
    global file1
    file1=open(filename,"a")
    return

def adddatabase(drawing_path,filename):
    openfile(filename)
    lines_of_file=["package require PWI_Glyph 2.4\n", \
    "pw::Application setUndoMaximumLevels 5\n",\
    "set var1 [pw::Application begin DatabaseImport]\n", \
    "$var1 initialize -type Automatic" + " {" + \
    drawing_path + "}\n","  $var1 setAttribute  \
    ModelAssembleTolerance 1.0\n"," $var1 setAttribute \
    QuiltAssembleAngle 1.0\n" \
    " $var1 read\n"," $var1 convert\n","$var1 end\n"]
    file1.writelines(lines_of_file)
    file1.close()
    return

def splitdb(item,splititemslist,filename):
    openfile(filename)
    file1.write("set _DB(1) [pw::DatabaseEntity getByName "
+quotes+item +quotes + "]\n")
    for i in range(0,len(splititemslist)):
        file1.write("set db" + str(i) +  " [pw::DatabaseEntity
getByName " +quotes + str(splititemslist[i]) +quotes+ "]\n")
```

```
    file1.write("set var1 [pw::Application begin Modify [list
$_DB(1)]]\n")
    file1.write("$_DB(1) split -reassemble [list ")
    for i in range(0,len(splititemslist)):
        file1.write("$db"+str(i) +" ")
    file1.write("]\n")
    file1.write("$var1 end\n")
    file1.write("unset var1\n")
    file1.write("pw::Application markUndoLevel {Split Quilt}\n")
    file1.close()

def unsmeshspacing(itemlist,dimensionspacing,boundarydecay, /
    algorithmtype,maxedgelength,minedgelength,maxangle,filename):
    openfile(filename)
    file1.write("pw::Connector setCalculateDimensionMethod
Spacing\n")
    if dimensionspacing!=0:
        file1.write("pw::Connector setCalculateDimensionSpacing " +
str(dimensionspacing)+"\n")
    if boundarydecay!=0:
        file1.write("pw::DomainUnstructured setDefault
BoundaryDecay "+ str(boundarydecay)+"\n")
    file1.write("pw::DomainUnstructured setDefault Algorithm " +
algorithmtype+"\n")
    if maxedgelength!=0:
        file1.write("pw::DomainUnstructured setDefault
EdgeMaximumLength " + str(maxedgelength) +"\n")
    if minedgelength!=0:
        file1.write("pw::DomainUnstructured setDefault
EdgeMinimumLength " + str(minedgelength)+"\n")
    if maxangle!=0:
        file1.write("pw::DomainUnstructured setDefault
NormalMaximumDeviation " + str(maxangle)+"\n")
    itemlistname=""
    for i in range(0,len(itemlist)):
        file1.write("set db" + str(i) + " [pw::DatabaseEntity
getByName " +quotes + itemlist[i] +quotes+"]\n")
        itemlistname=itemlistname + " $db"+str(i)
    file1.write("set var1 [pw::DomainUnstructured createOnDatabase
-parametricConnectors Aligned -merge 0 -reject _TMP(unused) [list"
+ itemlistname + "]]\n")
    file1.write("unset _TMP(unused)\n")
    file1.write("unset var1\n")
    file1.write("pw::Application markUndoLevel {Domains On DB
Entities}\n")
    file1.close()

def deldomain(xlower,xupper,rlower,rupper,filename):
    openfile(filename)
    xlower=str(xlower-1)
    xupper=str(xupper+1)
    rlower=str(rlower-0.1)
    rupper=str(rupper+0.25)
    lines_of_file=["set doms [pw::Grid getAll -type
pw::Domain]\n","set i 0\n","foreach dmns $doms {\n",\
```

```
    "   set xyz [[lindex $doms $i] getXYZ -grid 1]\n","   set x1
[lindex $xyz 0]\n","   set y1 [lindex $xyz 1]\n",\
    "   set z1 [lindex $xyz 2]\n","   set r [expr
{($y1**2+$z1**2)**0.5}]\n","   if {$x1<" + xupper + " & $x1>" + \
    xlower +" & $r<" + rupper + " & $r>"+ rlower + "} {\n","
pw::Entity delete [list [lindex $doms $i]]\n",\
    "   break}\n","   set i [expr {$i+1}]}\n"]
    file1.writelines(lines_of_file)
    file1.close()

def
createconnector(x1low,x1up,z1low,z1up,x2low,x2up,z2low,z2up,filenam
e):
    openfile(filename)
    x1low=str(x1low-1)
    x1up=str(x1up+1)
    z1low=str(z1low-0.1)
    z1up=str(z1up+0.1)
    x2low=str(x2low-1)
    x2up=str(x2up+1)
    z2low=str(z2low-0.1)
    z2up=str(z2up+0.1)
    lines_of_file=["set cons [pw::Grid getAll -type
pw::Connector]\n","set i 0\n","foreach con $cons {\n", \
    "   set xyz [[lindex $cons $i] getXYZ -grid 1]\n","   set x1
[lindex $xyz 0]\n","   set z1 [lindex $xyz 2]\n",\
    "   set r [expr {($y1**2+$z1**2)**0.5}]\n","   if {$x1<" + x1up
+ " & $x1>" + x1low +" & $z1<" + z1up + \
    " & $z1>"+ z1low + "} {\n","   set con1 $con}\n","   if {$x1<"
+ x2up + " & $x1>" + x2low +" & $z1<" + z2up +\
    " & $z1>"+ z2low + "} {\n","   set con2 $con}\n","   set i
[expr {$i+1}]}\n"," set _TMP(PW_5) [pw::SegmentSpline create]\n",\
    " $_TMP(PW_5) addPoint [$con1 getPosition -arc 1]\n","
$_TMP(PW_5) addPoint [$con2 getPosition -arc 1]\n", \
    " set _TMP(con_3) [pw::Connector create]\n"," $_TMP(con_3)
addSegment $_TMP(PW_5)\n"," unset _TMP(PW_5)\n",\
    " unset con1\n"," unset con2\n"]
    file1.writelines(lines_of_file)
    file1.close()

def createconnectoratan2(x1,r1,deg1,x2,r2,deg2,filename):
    openfile(filename)
    x1low=str(x1-1)
    x1up=str(x1+1)
    r1low=str(r1-0.1)
    r1up=str(r1+0.25)
    deg1low=str(deg1-1)
    deg1up=str(deg1+1)
    x2low=str(x2-1)
    x2up=str(x2+1)
    r2low=str(r2-0.1)
    r2up=str(r2+0.25)
    deg2low=str(deg2-1)
    deg2up=str(deg2+1)
    lines_of_file=["set cons [pw::Grid getAll -type
```

```
pw::Connector]\n","set i 0\n","foreach con $cons {\n", \
    "   set xyz [[lindex $cons $i] getXYZ -grid 1]\n","   set x1
[lindex $xyz 0]\n","   set y1 [lindex $xyz 1]\n", \
    "   set z1 [lindex $xyz 2]\n","   set r [expr
{($y1**2+$z1**2)**0.5}]\n", "if {$x1<"+ x1up +"& $x1>" + x1low + \
    " & $r<" +r1up +" & $r>"+r1low+"} {\n","       set deg [expr
(atan2($y1,$z1)*180/3.14) ]\n",\
    "if {$deg<0} {set deg [expr ($deg+360)] }\n","if {$deg>" +
deg1low+" & $deg<" + deg1up + "} {set con1 $con }}\n",\
    "if {$x1<"+ x2up +"& $x1>" + x2low + " & $r<" +r2up +" &
$r>"+r2low+"} {\n","      set deg [expr (atan2($y1,$z1)*180/3.14)
]\n",\
    "if {$deg<0} {set deg [expr ($deg+360)] }\n","if {$deg>" +
deg2low+" & $deg<" + deg2up + "} {set con2 $con }}\n", \
    "set i [expr {$i+1}]\n","}\n"," set connector
[pw::SegmentSpline create]\n"," $connector addPoint [$con1
getPosition" +\
    " -arc 0]\n"," $connector addPoint [$con2 getPosition -arc
0]\n"," set _TMP(con_3) [pw::Connector create]\n"," $_TMP(con_3)"+\
    " addSegment $connector\n"," unset connector\n"," $_TMP(con_3)
calculateDimension\n"," unset con1\n"," unset con2\n"]
    file1.writelines(lines_of_file)
    file1.close()
def
createdomainfromconnectors(x1,r1,deg1start,deg1final,x2,r2,deg2star
t,deg2final,filename):
    openfile(filename)
    x1low=str(x1-1)
    x1up=str(x1+1)
    r1low=str(r1-0.1)
    r1up=str(r1+0.25)
    deg1low=str(deg1start)
    deg1up=str(deg1final)
    x2low=str(x2-1)
    x2up=str(x2+1)
    r2low=str(r2-0.1)
    r2up=str(r2+0.25)
    deg2low=str(deg2start)
    deg2up=str(deg2final)
    xmlow=str((x1+x2)/2-10)
    xmup=str((x1+x2)/2+10)
    if deg1start<deg1final:
        connect1=str(" & ")
    else:
        connect1=str(" || ")
    if deg1start<deg1final:
        connect2=str(" & ")
    else:
        connect2=str(" || ")
    lines_of_file=["set cons [pw::Grid getAll -type
pw::Connector]\n","set i 0\n","set j 0\n","set conlist [list]\n", \
    "foreach con $cons {\n","   set xyz [[lindex $cons $i] getXYZ -
grid 1]\n","   set x1 [lindex $xyz 0]\n", \
    "   set y1 [lindex $xyz 1]\n","   set z1 [lindex $xyz 2]\n","
set xyzmid [[lindex $cons $i] getXYZ -parameter [list 0.5]]\n"
```

```
    "   set r [expr {($y1**2+$z1**2)**0.5}]\n","   set xm [lindex
$xyzmid 0]\n","   if {$x1<"+ x1up+" & $x1>" + x1low+" & $r<" + r1up
+ \
    " & $r>" + r1low +"} {\n", "   set deg [expr
(atan2($y1,$z1)*180/3.14) ]\n","   if {$deg<0} {set deg [expr
($deg+360)] }\n", \
    "   if {$deg>" + deg1low + connect1 + "$deg<" + deg1up +" }
{\n","   lappend conlist $con}}\n","   if {$x1<" + x2up +" & $x1>"
+ x2low + "& $r<" + r2up + " & $r>" + r2low + " } {\n", \
    "   set deg [expr (atan2($y1,$z1)*180/3.14) ]\n", "   if
{$deg<0} {set deg [expr ($deg+360)] }\n","   if {$deg>" + deg2low +
connect2 +"$deg<" + deg2up + "} {\n","    lappend conlist
$con}}\n", \
    "   if {$xm>" + xmlow +" & $xm<" + xmup +"} {\n","  lappend
conlist $con}\n","   set i [expr {$i+1}]\n","   }\n", \
    "set unstDoms [pw::DomainUnstructured createFromConnectors -
reject rejectCons $conlist]\n"]
    file1.writelines(lines_of_file)
    file1.close()
def projectdomains(filename):
    openfile(filename)
    lines_of_file=["pw::Application markUndoLevel {Domains On DB
Entities}\n","set doms [pw::Grid getAll -type pw::Domain]\n", \
    "set i 0\n","set domlist [list]\n","foreach dmns $doms {\n","
lappend domlist $dmns\n","   set i [expr {$i+1}]}\n", \
    "pw::Entity project -type ClosestPoint -interior $domlist\n"]
    file1.writelines(lines_of_file)
    file1.close()
def
createextrude(flipdirection,initialsize,growthfactor,stepnumber,ske
wangle,filename):
    openfile(filename)
    flipdirection=str(flipdirection)
    initialsize=str(initialsize)
    growthfactor=str(growthfactor)
    stepnumber=str(stepnumber)
    skewangle=str(skewangle)
    lines_of_file=["pw::Application markUndoLevel {Domains On DB
Entities}\n","set doms [pw::Grid getAll -type pw::Domain]\n", \
    "set i 0\n","set domlist [list]\n","foreach dmns $doms {\n","
lappend domlist $dmns\n","    set i [expr {$i+1}]}\n", \
    "set mode1 [pw::Application begin Create]\n","  set blk
[pw::FaceUnstructured createFromDomains $domlist]\n", \
     "set face [lindex $blk 0]\n"," set extblock [pw::BlockExtruded
create]\n","    $extblock addFace $face\n","$mode1 end\n", \
    "unset mode1\n","set mode2 [pw::Application begin
ExtrusionSolver [list $extblock]]\n","  $mode2 setKeepFailingStep
true\n", \
    "set bl1 [pw::GridEntity getByName "+ '"' + "blk-1"+ '"'
+"]\n","  $bl1 setExtrusionSolverAttribute DirectionFlipped " +
flipdirection + "\n", \
    "$bl1 setExtrusionSolverAttribute NormalInitialStepSize " +
initialsize +"\n","$bl1 setExtrusionSolverAttribute
SpacingGrowthFactor " + growthfactor + \
    "\n", "$bl1 setExtrusionSolverAttribute StopAtSkewAngle
```

```
Off\n","$bl1 setExtrusionSolverAttribute StopAtSkewAngle " +
skewangle +"\n","  $mode2 run " + stepnumber + "\n",  "  $mode2 run
-1\n","$mode2 end\n"]
    file1.writelines(lines_of_file)
    file1.close()
def merge(tolerance,filename):
    openfile(filename)
    tolerance=str(tolerance)
    lines_of_file=["set mrg [pw::Application begin Merge]\n","
$mrg mergeConnectors -visibleOnly -exclude None -tolerance " +
tolerance + "\n", \
    "$mrg end \n", "unset mrg\n"]
    file1.writelines(lines_of_file)
    file1.close()
def createassemblespecial(filename):
    openfile(filename)
    lines_of_file=["set doms [pw::Grid getAll -type
pw::Domain]\n","set i 0\n","set total_dom_number 0\n","set domlist
[list]\n", \
                   "set variable [list]\n","set face
[pw::FaceUnstructured create]\n","foreach dmns $doms {\n", \
                   "set $variable [lindex [split $dmns " + ' "'
+"_"+'"'+"] 1]\n","set total_dom_number [expr
{$total_dom_number+1}]}\n",\
                   "set dom_extrude [expr $total_dom_number/2 + 2
]\n","set begin [pw::Application begin Create]\n", "set blk
[pw::BlockUnstructured create]\n","foreach dmns $doms {\n", \
                   "   set dom_number [lindex [split $dmns " + '"'
+"_" +'"' +"] 1]\n","        if {$dom_number>$dom_extrude} {\n", \
                   "$face addDomain $dmns   }}\n"," $blk addFace
$face\n","$begin end\n","unset begin\n","unset face\n","unset
blk\n", \
                   "pw::Application markUndoLevel {Assemble
Block}\n", "set bl2 [pw::GridEntity getByName " + '"' +"blk-2" +'"'
+"]\n", \
                   "set initialize [pw::Application begin
UnstructuredSolver [list $bl2]]\n","  $initialize run
Initialize\n", \
                   "$initialize end\n","unset initialize\n"]
    file1.writelines(lines_of_file)
    file1.close()
def joindomains(xmin,xmax,r,filename):
    openfile(filename)
    xmin=str(xmin)
    xmax=str(xmax)
    rmin=str(r-0.1)
    rmax=str(r+0.25)
    lines_of_file=["set doms [pw::Grid getAll -type
pw::Domain]\n","set i 0\n","set domlist [list]\n","foreach dmns
$doms {\n", \
                   "   set cellnumber [$dmns getCellCount]\n","
set midcell [expr {$cellnumber/2}]\n", \
                   "   set midcell [expr {round($midcell)}]\n","
set xyz [[lindex $doms $i] getXYZ -grid $midcell]\n",\
                   "   set x1 [lindex $xyz 0]\n","   set y1 [lindex
```

```
$xyz 1]\n","   set z1 [lindex $xyz 2]\n",\
                    "   set r [expr {($y1**2+$z1**2)**0.5}]\n","
if {$x1<" + xmax + " & $x1>" + xmin + " & $r<" + rmax + " & $r>" +
rmin + "} {\n", \
                    "    lappend domlist $dmns}\n","   set i [expr
{$i+1}]}\n","set dmnjoin [pw::DomainUnstructured join -reject
unjoinDoms $domlist]\n"]
    file1.writelines(lines_of_file)
    file1.close()
def projectspesific(xmin,xmax,r,projectdb,filename):
    openfile(filename)
    xmin=str(xmin)
    xmax=str(xmax)
    rmin=str(r-0.1)
    rmax=str(r+0.25)
    lines_of_file=["set doms [pw::Grid getAll -type
pw::Domain]\n","set i 0\n","set domlist [list]\n","foreach dmns
$doms {\n", \
                    "    set cellnumber [$dmns getCellCount]\n","
set midcell [expr {$cellnumber/2}]\n","   set midcell [expr
{round($midcell)}]\n", \
                    "    set xyz [[lindex $doms $i] getXYZ -grid
$midcell]\n","    set x1 [lindex $xyz 0]\n","   set y1 [lindex $xyz
1]\n", \
                    "    set z1 [lindex $xyz 2]\n","    set r [expr
{($y1**2+$z1**2)**0.5}]\n", \
                    "    if {$x1<" +xmax + " & $x1>" +xmin+" & $r<"
+rmax+" & $r>"+rmin+"} {\n","    lappend domlist $dmns}\n", \
                    "    set i [expr {$i+1}]}\n","set projectdb
[pw::DatabaseEntity getByName " + '"' +projectdb +'"'+"]\n", \
                    "set projectdomain [subst {[list
$domlist]}]\n","set project [pw::Application begin Modify
$projectdb]\n", \
                    "set projectdatabase [list $projectdb]\n","
pw::Entity project -type ClosestPoint -interior $projectdomain ", \
                    "$projectdatabase\n","$project end\n"]
    file1.writelines(lines_of_file)
    file1.close()
def resetdimension(x,r,dim,filename):
    openfile(filename)
    xmin=str(x-1)
    xmax=str(x+1)
    rmin=str(r-0.1)
    rmax=str(r+0.25)
    dim=str(dim)
    lines_of_file=["set cons [pw::Grid getAll -type
pw::Connector]\n","set i 0\n","set conlist [list]\n","foreach con
$cons {\n", \
                    "    set xyz [[lindex $cons $i] getXYZ -grid
1]\n","   set x1 [lindex $xyz 0]\n","   set y1 [lindex $xyz 1]\n",
\
                    "    set z1 [lindex $xyz 2]\n","   set r [expr
{($y1**2+$z1**2)**0.5}]\n", \
                    "    if {$x1<" + xmax + " & $x1>" + xmin + " &
$r<" + rmax +" & $r>" +rmin +"} {\n", "   lappend conlist $con}\n",
```

```
\
                    "    set i [expr {$i+1}]}\n","set resetdim
[pw::Application begin Dimension]\n","  set conlistres
[pw::Collection create]\n", \
                    "    $conlistres set $conlist\n","  $conlistres
do resetGeneralDistributions\n","  $conlistres do
setDimensionFromSpacing " + dim +"\n", \
                    "    $conlistres delete\n","  $resetdim balance
-resetGeneralDistributions\n","$resetdim end\n"]
    file1.writelines(lines_of_file)
    file1.close()
def
trexdomain(x,xave,rmax,rmin,conx,conr,maxlayer,fulllayer,growthrate
,spacing,maxlength,filename):
    openfile(filename)
    conxmin=str(conx-1)
    conxmax=str(conx+1)
    xavemin=str(xave-1)
    xavemax=str(xave+1)
    rmin=str(rmin-0.1)
    rmax=str(rmax+0.25)
    conrmin=str(conr-0.1)
    conrmax=str(conr+0.25)
    maxlayer=str(maxlayer)
    fulllayer=str(fulllayer)
    growthrate=str(growthrate)
    spacing=str(spacing)
    maxlength=str(maxlength)
    lines_of_file=["set doms [pw::Grid getAll -type
pw::Domain]\n","set cons [pw::Grid getAll -type pw::Connector]\n",
\
                    "set i 0\n","set j 0\n","set k 0\n","set domlist
[list]\n","set conlist [list]\n","foreach dmns $doms {\n", \
                    "   set cellnumber [$dmns getPointCount]\n","
set xlist [list]\n","   set rlist [list]\n", \
                    "for {set i 1 } {$i <= $cellnumber} {incr i}
{\n","   set xyz [$dmns getXYZ -grid $i]\n","   set x1 [lindex $xyz
0]\n", \
                    "   set y1 [lindex $xyz 1]\n","   set z1 [lindex
$xyz 2]\n","   set r [expr {($y1**2+$z1**2)**0.5}]\n","   lappend
xlist $x1\n","   lappend rlist $r}\n", \
                    "   set xmin [tcl::mathfunc::min {*}$xlist]\n","
set xmax [tcl::mathfunc::max {*}$xlist]\n", \
                    "   set xave [expr {($xmin + $xmax)/2}]\n","
set xminindex [lsearch $xlist $xmin]\n","   set rxmin [lindex
$rlist $xminindex]\n", \
                    "   if {$xave<" + xavemax +" & $xave>" +xavemin
+ " & $rxmin<" + rmax +" & $rxmin>" + rmin +"} {\n","   set
trexdomain $dmns}}\n", \
                    "set moderef [pw::Application begin
UnstructuredSolver [list $trexdomain]]\n", \
                    "foreach con $cons {\n","   set xyz [[lindex
$con $j] getXYZ -grid 1]\n","   set x1 [lindex $xyz 0]\n","   set
y1 [lindex $xyz 1]\n", \
                    "   set z1 [lindex $xyz 2]\n","   set r [expr
```

```
{($y1**2+$z1**2)**0.5}]\n","   if {$x1<" + conxmax+" & $x1>" +
conxmin + \
                    " & $r<" + conrmax+" & $r>" + conrmin + "}
{\n","   lappend conlist $con}}\n","set trexfunc [pw::Collection
create]\n", \
                    "$trexfunc set [list
$trexdomain]\n","$trexdomain setUnstructuredSolverAttribute
TRexMaximumLayers " + maxlayer +"\n", \
                    "$trexdomain setUnstructuredSolverAttribute
TRexFullLayers " + fulllayer + "\n", \
                    "$trexdomain setUnstructuredSolverAttribute
TRexGrowthRate " + growthrate + "\n", \
                    "$trexdomain setUnstructuredSolverAttribute
EdgeMaximumLength " + maxlength + "\n", \
                    "set leBC [pw::TRexCondition create]\n","$leBC
setType Wall\n","$leBC setSpacing " + spacing + "\n", "foreach cons
$conlist {\n", \
                    "$leBC apply [list [list $trexdomain [lindex
$conlist $k]]]\n","incr k}\n","$moderef run Decimate\n","$moderef
run Refine\n","$moderef end\n", \
                     "unset moderef\n"]
    file1.writelines(lines_of_file)
    file1.close()
def
connectordimension(x1,r1,deg1start,deg1final,x2,r2,deg2start,deg2fi
nal,condim,filename):
    openfile(filename)
    x1low=str(x1-1)
    x1up=str(x1+1)
    r1low=str(r1-0.1)
    r1up=str(r1+0.1)
    deg1low=str(deg1start)
    deg1up=str(deg1final)
    x2low=str(x2-1)
    x2up=str(x2+1)
    r2low=str(r2-0.1)
    r2up=str(r2+0.25)
    deg2low=str(deg2start)
    deg2up=str(deg2final)
    xmlow=str((x1+x2)/2-10)
    xmup=str((x1+x2)/2+10)
    condim=str(condim)
    if deg1start<deg1final:
        connect1=str(" & ")
    else:
        connect1=str(" || ")
    if deg1start<deg1final:
        connect2=str(" & ")
    else:
        connect2=str(" || ")
    lines_of_file=["set cons [pw::Grid getAll -type
pw::Connector]\n","set i 0\n","set j 0\n","set conlist [list]\n", \
    "foreach con $cons {\n","   set xyz [[lindex $cons $i] getXYZ -
grid 1]\n","   set x1 [lindex $xyz 0]\n", \
    "   set y1 [lindex $xyz 1]\n","   set z1 [lindex $xyz 2]\n","
```

```
set xyzmid [[lindex $cons $i] getXYZ -parameter [list 0.5]]\n"
    "   set r [expr {($y1**2+$z1**2)**0.5}]\n","   set xm [lindex
$xyzmid 0]\n","   if {$x1<"+ x1up+" & $x1>" + x1low+" & $r<" + r1up
+ \
    " & $r>" + r1low +"} {\n", "   set deg [expr
(atan2($y1,$z1)*180/3.14) ]\n","   if {$deg<0} {set deg [expr
($deg+360)] }\n", \
    "   if {$deg>" + deg1low + connect1 + "$deg<" + deg1up +" }
{\n","   lappend conlist $con}}\n","   if {$x1<" + x2up +" & $x1>"
+ x2low + "& $r<" + r2up + " & $r>" + r2low + " } {\n", \
    "   set deg [expr (atan2($y1,$z1)*180/3.14) ]\n", "   if
{$deg<0} {set deg [expr ($deg+360)] }\n","   if {$deg>" + deg2low +
connect2 +"$deg<" + deg2up + "} {\n","    lappend conlist
$con}}\n", \
    "   if {$xm>" + xmlow +" & $xm<" + xmup +"} {\n","  lappend
conlist $con}\n","   set i [expr {$i+1}]\n","   }\n", \
    "set condimmode [pw::Application begin Dimension]\n","set
condim [pw::Collection create]\n","$condim set $conlist\n","$condim
do resetGeneralDistributions\n", \
    "$condim do setDimensionFromSpacing "+ condim + "\n","$condim
delete\n","unset condim\n","$condimmode balance -
resetGeneralDistributions\n", \
    "$condimmode end\n","unset condimmode\n"]
    file1.writelines(lines_of_file)
    file1.close()
def
trexblock(blockname,maxlayer,fulllayer,growthrate,skewness,spacing,
filename):
    openfile(filename)
    maxlayer=str(maxlayer)
    fulllayer=str(fulllayer)
    growthrate=str(growthrate)
    skewness=str(skewness)
    spacing=str(spacing)
    lines_of_file=["set doms [pw::Grid getAll -type
pw::Domain]\n","set i 0\n","set j 0\n","set total_dom_number 0\n",
\
                   "set domlist [list]\n","set variable
[list]\n","set face [pw::FaceUnstructured create]\n", \
                   "foreach dmns $doms {\n", "set $variable
[lindex [split $dmns "+ quotes +"_" +quotes +"] 1]\n","set
total_dom_number [expr {$total_dom_number+1}]}\n", \
                   "set dom_extrude [expr $total_dom_number/2 + 2
]\n", "foreach dmns $doms {\n","  set dom_number [lindex [split
$dmns " + quotes +"_" +quotes +"] 1]\n", \
                   "       if {$dom_number>$dom_extrude} {\n",
"lappend domlist $dmns  }}\n", "set trexfunc [pw::Collection
create]\n", \
                   "set blk [pw::GridEntity getByName " +quotes +
blockname + quotes + "]\n","$trexfunc set [list $blk]\n", \
                   "$blk setUnstructuredSolverAttribute
TRexMaximumLayers " + maxlayer + "\n", "$blk
setUnstructuredSolverAttribute TRexFullLayers " +fulllayer +"\n", \
                   "$blk setUnstructuredSolverAttribute
TRexGrowthRate " + growthrate + "\n","$blk
```

```
setUnstructuredSolverAttribute TRexSkewCriteriaEquivolume " +
skewness +"\n", "$trexfunc delete\n","unset trexfunc\n", \
                    "set leBC [pw::TRexCondition create]\n","$leBC
setType Wall\n","$leBC setSpacing "+ spacing +"\n","foreach dmns
$domlist {\n", \
                    "$leBC apply [list [list $blk [lindex $domlist
$j]]]\n","incr j}\n","$blk initialize\n"]
    file1.writelines(lines_of_file)
    file1.close()
def
defineboundaries(cimin,cimax,temin,temax,termin,termax,eomin,eomax,
filename):
    openfile(filename)
    cimin=str(cimin-10)
    cimax=str(cimax+10)
    temin=str(temin-10)
    temax=str(temax+10)
    termin=str(termin-0.1)
    termax=str(termax+0.2)
    eomin=str(eomin-10)
    eomax=str(eomax+10)
    lines_of_file=["pw::Application setCAESolver {ANSYS Fluent}
3\n","set doms [pw::Grid getAll -type pw::Domain]\n","set i
0\n","set j 0\n", \
                    "set k 1\n","set total_dom_number 0\n","set
domlist [list]\n","set variable [list]\n","foreach dmns $doms
{\n","set $variable [lindex [split $dmns " + quotes +"_" +quotes
+"] 1]\n", \
                    "set total_dom_number [expr
{$total_dom_number+1}]}\n","set dom_extrude [expr
$total_dom_number/2 +3 ]\n","foreach dmns $doms {\n", \
                    "set dom_number [lindex [split $dmns " + quotes
+"_" +quotes +"] 1]\n","     if {$dom_number<$dom_extrude}
{\n","lappend domlist $dmns   }}\n", \
                    "set wallbc [pw::BoundaryCondition
create]\n","pw::Application markUndoLevel {Create BC}\n","set blk
[pw::GridEntity getByName " +quotes +"blk-1"+ quotes +"]\n", \
                    "set bc1 [pw::BoundaryCondition getByName "
+quotes +"bc-2" + quotes + "]\n","foreach dmns $domlist {\n","$bc1
apply [list [list $blk [lindex $domlist $j]]]\n", \
                    "incr j}\n","$wallbc setName "+ quotes +" wall"+
quotes + "\n","foreach dmns $domlist {\n","   set xyz [$dmns getXYZ
-grid $k]\n", \
                    "set y1 [lindex $xyz 1]\n","   set z1 [lindex
$xyz 2]\n","   set r [expr {($y1**2+$z1**2)**0.5}]\n","   set x1
[lindex $xyz 0]\n", \
                    "if {$x1<" + cimax +"& $x1>" +cimin +"} {\n","
set ci_dom $dmns}\n","   if {$x1<"+ temax +" & $x1>"+ temin +" &
$r<" + termax +" & $r>"+ termin+"} {\n", \
                    "set te_dom $dmns}\n","   if {$x1<"+ eomax+" &
$x1>" +eomin+"} {\n","   set eo_dom $dmns}\n","   incr j}\n","set
cibc [pw::BoundaryCondition create]\n", \
                    "pw::Application markUndoLevel {Create
BC}\n","set bc2 [pw::BoundaryCondition getByName "+ quotes  +"bc-
3"+ quotes +"]\n", \
```

```
                    "$bc2 apply [list [list $blk $ci_dom]]\n","$bc2
setName "+quotes + "cooling-inlet" + quotes +"\n","set tebc
[pw::BoundaryCondition create]\n", \
                    "pw::Application markUndoLevel {Create
BC}\n","set bc3 [pw::BoundaryCondition getByName " + quotes +"bc-
4"+ quotes +"]\n", \
                    "$bc3 apply [list [list $blk $te_dom]]\n","$bc3
setName " +quotes +"turbine-exit"+ quotes +"\n","set eobc
[pw::BoundaryCondition create]\n", \
                    "pw::Application markUndoLevel {Create
BC}\n","set bc4 [pw::BoundaryCondition getByName " +quotes + "bc-5"
+quotes +"]\n", \
                    "$bc4 apply [list [list $blk $eo_dom]]\n","$bc4
setName "+ quotes +"ejector-out"+ quotes +"\n"]
    file1.writelines(lines_of_file)
    file1.close()

def exportcae(output_path,filename):
    openfile(filename)
    lines_of_file=["set doms [pw::Grid getAll -type
pw::Domain]\n","set blks [pw::Grid getAll -type pw::Block]\n","set
caelist [list]\n", \
                    "foreach blocks $blks {lappend caelist
$blocks}\n","foreach dmns $doms {lappend caelist $dmns}\n", \
                    "set exportcae [pw::Application begin CaeExport
[pw::Entity sort  $caelist]]\n", \
                    "  $exportcae initialize -type CAE {"+
output_path +"}\n","  $exportcae setAttribute
CellCombineAnisotropic true\n","  if {![$exportcae verify]} {\n", \
                    "error" + quotes + "Data verification failed." +
quotes + "\n", "  }\n","  $exportcae write\n","$exportcae
end\n","unset exportcae\n"]
    file1.writelines(lines_of_file)
    file1.close()
def mesh_status(mesh_path,mesh_status_file,filename):
    openfile(filename)
    lines_of_file=["cd " + quotes + mesh_path + quotes + "\n","set
data "+ quotes +"mesh finished" + quotes + "\n","set file_name "
+quotes +mesh_status_file + quotes +"\n", \
                    "set file_obj [open $file_name w]\n","puts -
nonewline $file_obj $data\n","close $file_obj\n"]
    file1.writelines(lines_of_file)
    file1.close()

def save_exit(output_path,filename)     :
    openfile(filename)
    lines_of_file=["pw::Application save {" + output_path+ "}\n",
"pw::Application exit\n"]
    file1.writelines(lines_of_file)
    file1.close()

def domaincheck(filename):
    openfile(filename)
    lines_of_file=["set doms [pw::Grid getAll -type
pw::Domain]\n","set i 0\n","set total_dom_number 0\n","set domlist
```

```
[list]\n","set variable [list]\n", \
                    "set face [pw::FaceUnstructured
create]\n","foreach dmns $doms {\n","set $variable [lindex [split
$dmns  "+ "_" +"] 1]\n", \
                    "set total_dom_number [expr
{$total_dom_number+1}]}\n"
                    "if {$total_dom_number<7} {\n","set _DB(1)
[pw::DatabaseEntity getByName " +quotes +"nozzle_extracted-quilt-3"
+quotes+"]\n", \
                    "set _TMP(PW_23) [pw::DomainUnstructured
createOnDatabase -parametricConnectors Aligned -merge 0 -reject
_TMP(unused) [list $_DB(1)]]\n", \
                    "unset _TMP(unused)\n","unset
_TMP(PW_23)\n","pw::Application markUndoLevel {Domains On DB
Entities}\n", \
                    "set _DM(1) [pw::GridEntity getByName " +quotes
+"dom-7" + quotes +"]\n","set _TMP(mode_5) [pw::Application begin
UnstructuredSolver [list $_DM(1)]]\n", \
                    "$_TMP(mode_5) run Refine\n","$_TMP(mode_5)
end\n","unset _TMP(mode_5)\n","}\n"]
    file1.writelines(lines_of_file)
    file1.close()

def joinconnectors(xmin,xmax,r,filename):
    openfile(filename)
    xmin=str(xmin)
    xmax=str(xmax)
    rmin=str(r-0.1)
    rmax=str(r+0.25)
    lines_of_file=["set i 0\n","set conlist1 [list]\n","foreach con
$cons {\n","set xyz [[lindex $cons $i] getXYZ -grid 1]\n","set x1
[lindex $xyz 0]\n", \
                    "set y1 [lindex $xyz 1]\n","set z1 [lindex $xyz
2]\n","set r [expr {($y1**2+$z1**2)**0.5}]\n","if {$x1<" +xmax + "
& $x1>" + xmin +" & $r<" +rmax+" & $r>" + rmin +" } {\n", \
                    "lappend conlist1 $con}\n","set i [expr
{$i+1}]}\n","set _TMP(PW_1) [pw::Connector join -reject
_TMP(ignored) -keepDistribution $conlist1]\n"]
    file1.writelines(lines_of_file)
    file1.close()

def removebreakpoints(xmin,xmax,r,filename):
    openfile(filename)
    xmin=str(xmin)
    xmax=str(xmax)
    rmin=str(r-0.1)
    rmax=str(r+0.25)
    lines_of_file=["set i 0\n","set conlist2 [list]\n","foreach con
$cons {\n","set xyz [[lindex $cons $i] getXYZ -grid 1]\n","set x1
[lindex $xyz 0]\n", \
                    "set y1 [lindex $xyz 1]\n","set z1 [lindex $xyz
2]\n","set r [expr {($y1**2+$z1**2)**0.5}]\n","if {$x1<" +xmax + "
& $x1>" + xmin +" & $r<" +rmax+" & $r>" + rmin +" } {\n", \
                    "lappend conlist2 $con}\n","set i [expr
{$i+1}]}\n","set _TMP(mode_1) [pw::Application begin Modify
```

```
$conlist2]\n","foreach con2 $conlist2 {\n", \
                    "$con2 removeAllBreakPoints}\n","$_TMP(mode_1)
end\n"]
    file1.writelines(lines_of_file)
    file1.close()

def maxedgelength(maxedge,filename):
    openfile(filename)
    maxedge=str(maxedge)
    lines_of_file=["pw::DomainUnstructured setDefault
EdgeMaximumLength " + maxedge + "\n"]
    file1.writelines(lines_of_file)
    file1.close()


def
checkblockempty(flipdirection,initialsize,growthfactor,stepnumber,f
ilename):
    openfile(filename)
    flipdirection=str(flipdirection)
    initialsize=str(initialsize)
    growthfactor=str(growthfactor)
    stepnumber=str(stepnumber)
    lines_of_file=["    if {[$bl1 getCellCount]==0} {\n"," $bl1
delete\n", \
    "pw::Application markUndoLevel {Domains On DB Entities}\n","set
doms [pw::Grid getAll -type pw::Domain]\n", \
    "set i 0\n","set domlist [list]\n","foreach dmns $doms {\n","
lappend domlist $dmns\n","    set i [expr {$i+1}]}\n", \
    "set mode1 [pw::Application begin Create]\n","  set blk
[pw::FaceUnstructured createFromDomains $domlist]\n", \
     "set face [lindex $blk 0]\n"," set extblock [pw::BlockExtruded
create]\n","    $extblock addFace $face\n","$mode1 end\n", \
    "unset mode1\n","set mode2 [pw::Application begin
ExtrusionSolver [list $extblock]]\n","  $mode2 setKeepFailingStep
true\n", \
    "set bl1 [pw::GridEntity getByName "+ '"' + "blk-1"+ '"'
+"]\n","  $bl1 setExtrusionSolverAttribute DirectionFlipped " +
flipdirection + "\n", \
    "$bl1 setExtrusionSolverAttribute NormalInitialStepSize " +
initialsize +"\n","$bl1 setExtrusionSolverAttribute
SpacingGrowthFactor " + growthfactor + \
    "\n","  $mode2 run " + stepnumber + "\n",  "  $mode2 run -
1\n","$mode2 end\n", "}\n"]
    file1.writelines(lines_of_file)
    file1.close()
```

Solver Functions

```
##########################################################################
#                              AKAY BAYAT                                #
```

```
#               METU Aerospace Engineering MS Student                #
#                     Grid Generator Functions                       #
########################################################################

import math
global file2
quotes='""'

def openfile(filename):
    global file1
    file1=open(filename,"a")
    return

def create_journal_file(case_file_path,solution_method, \
    te_wrate,te_temp,te_swirl_deg,sw_ctr,ci_gage, \
    ci_temp,eo_gage,eo_temp,iter_num,filename):

    openfile(filename)
    te_wrate=str(te_wrate) #Turbine exit mass flow rate
    te_temp=str(te_temp)   #Turbine exit temperature
    ci_gage=str(ci_gage)   #Cooling inlet gage pressure
    ci_temp=str(ci_temp)   #Cooling inlet temperature
    eo_temp=str(ci_temp)   #Ejector out temperature
    eo_gage=str(ci_gage)   #Ejector out gage pressure
    sw_ctr=str(sw_ctr)     #Swirl center location
    te_swirl=str(round(math.tan(te_swirl_deg* \
    math.pi/180),2))       #Turbine exit swirl degree
    iter_num=str(iter_num) #Maximum iteration number

    lines_of_file=["/file read-case " + quotes +  \
    case_file_path +quotes+ "\n", \
    "/file/set-batch-options yes yes no\n", \
    "/mesh/scale 0.001 0.001 0.001\n","/define/models/" +
    "/define/models/viscous/"+ solution_method + \
    "solver/pressure-based yes\n","/define/models/" +
    "energy yes yes yes\n", \
    "/mesh/repair-improve/improve-quality\n", \
    " yes\n","/define/materials/change-create air air" +
    " yes ideal-gas no no no no no no\n", \
                    "/define/boundary-conditions/modify-
zones/zone-type turbine-exit mass-flow-inlet\n", \
                    "/define/boundary-conditions/mass-flow-inlet
turbine-exit yes yes no " + te_wrate +" no " + te_temp+ " no 0
yes no no yes no 0 no " + te_swirl +" no 1 1 0 0 " + swirl_center
+" 0 0 0 no no yes 5 10\n", \
                    "/define/boundary-conditions/modify-
zones/zone-type cooling-inlet pressure-inlet\n", \
                    "/define/boundary-conditions/pressure-inlet
cooling-inlet yes no " + ci_gage +" no 0 no " + ci_temp +" no yes
no no yes 5 10\n", \
                    "/define/boundary-conditions/modify-
zones/zone-type ejector-out pressure-outlet\n", \
                    "/define/boundary-conditions/pressure-outlet
ejector-out no " + eo_gage +" no " + eo_temp+" no yes no no yes 5
10 no yes no\n", \
```

```
                    "/solve/monitors/surface/set-monitor
tot_mfr_residual "+quotes +"Mass Flow Rate" + quotes +" turbine-
exit cooling-inlet ejector-out () yes 2 yes yes total_mass 1\n",
\
                    "/solve/monitors/surface/set-monitor
ci_mfr_residual " +quotes +"Mass Flow Rate" + quotes +" cooling-
inlet () yes 3 yes yes cooling_inlet_mass 1\n", \
                    "/solve/monitors/surface/set-monitor
turbine_exit_ptot "+quotes +"Area-Weighted Average"+quotes +"
total-pressure turbine-exit () yes 4 yes yes turbine_exit_ptot
1\n", \
                    "/solve/initialize/hyb-
initialization\n","/solve/iterate " + iteration_number + "\n", \
                    "/report/surface-integrals/area-weighted-avg
turbine-exit () total-pressure yes " + quotes +"total-pressure"
+quotes +"\n", \
                    "/report/fluxes/mass-flow yes yes "+ quotes
+"mass_flow_rates" +quotes + "\n", \
                    "exit ok"]
    file1.writelines(lines_of_file)
    file1.close()
```

C. ANALYSIS RESULTS

Results with 3 Parameters and Base Objective Function

n_{iter}	n_l	D_8 [mm]	D_7 [mm]	$\dot{m}_{te}$ [kg/s]	P_{te} [Pa]	F_{obj}
1	7	133.4	333.9	1.53	6636	407.5
2	12	142.9	358.8	1.16	6431	502.1
3	10	152.4	356.9	1.23	6190	479.0
4	7	118.2	362.7	1.59	6387	395.6
5	11	142.4	343.2	1.27	5919	469.7
6	8	126.3	368.3	1.38	6312	440.5
7	5	175.6	348.2	1.44	9377	429.5
8	9	177.7	358	1.28	6191	467.1
9	6	149.3	335.5	1.55	7498	404.4
10	10	103.7	340.8	1.13	7233	513.7
11	8	159	346.2	1.50	6354	413.3
12	7	120.5	342.2	1.47	6931	420.3
13	9	134.4	366.6	1.26	6169	472.9
14	8	129.3	364.3	1.42	6250	430.8
15	7	131.9	352.7	1.69	6338	378.2
16	5	161.1	349.6	1.54	9550	408.5
17	9	171.9	368	1.23	6403	479.2
18	8	158.2	366.1	1.35	6333	448.1
19	6	151	344.5	1.59	7428	396.7
20	6	112.8	331.6	1.48	7973	419.2
21	7	147.3	363.2	1.55	6263	403.1
22	12	109.6	351.4	1.21	6052	485.1
23	9	163.2	336.3	1.45	6012	423.7
24	5	178.7	332.7	1.51	9362	414.4
25	11	106.9	335.1	1.34	5768	450.1
26	11	164.5	339.1	1.13	7233	513.7
27	11	145.6	344	1.26	5893	472.6
28	12	108.5	354	1.16	6010	501.7
29	11	170.7	361	1.18	6360	495.3
30	10	154.5	338.6	1.40	5966	435.3
31	10	173.6	365.1	1.18	6387	495.3

n_{iter}	n_l	D_8 [mm]	D_7 [mm]	$\dot{m}_{te}$ [kg/s]	P_{te} [Pa]	F_{obj}
32	8	166.8	340.2	1.49	6277	415.3
33	6	102.4	347.5	1.59	8134	397.5
34	9	136.7	354.6	1.40	6223	435.5
35	6	116.3	359.7	1.68	7382	380.9
36	10	137.9	351	1.34	6117	450.5
37	10	128.2	361.1	1.22	6109	482.0
38	8	122	330.5	1.44	6140	426.1
39	9	110.9	370	1.27	6425	470.2
40	6	140.3	356.6	1.77	7013	367.9
41	12	168.1	355.3	1.17	6277	498.5
42	7	114.3	337.7	1.58	6787	397.9
43	11	101.6	348.9	1.26	5969	472.7
44	6	123.9	345.8	1.64	7647	388.0
45	8	155.7	332.1	1.47	6066	419.4
46	7	131.1	357.8	1.62	6359	390.2
47	11	160	358.8	1.20	6348	488.7
48	10	112	365.3	1.20	6403	488.7
49	7	178.8	362.7	1.51	6216	411.1
50	11	155.6	343.2	1.13	7233	513.7
51	11	148.6	351.3	1.23	6078	478.9
52	6	172.9	352.3	1.68	6982	380.5
53	6	104.9	348.6	1.64	7973	388.3
54	6	151.2	349	1.64	7305	387.6
55	10	156.8	334.4	1.38	5840	440.0
56	11	121.7	346.2	1.28	5987	466.9
57	7	174.2	363	1.53	6240	407.0
58	9	123.7	342.9	1.45	6207	423.9
59	6	120.9	368.4	1.73	7134	372.6
60	5	136.2	366.6	1.83	9213	361.5
61	12	125.6	369.8	1.12	6822	516.9
62	5	113.5	351.2	1.90	10856	353.7
63	6	101.5	355.8	1.66	7811	384.7
64	11	121.9	364.3	1.18	6334	495.2
65	6	133.1	337.7	1.13	7233	513.7
66	6	131.6	355.2	1.67	7339	382.5
67	6	116	340.9	1.59	7926	397.2
68	10	130.7	368.7	1.17	6485	498.8

n_{iter}	n_l	D_8 [mm]	D_7 [mm]	$\dot{m}_{te}$ [kg/s]	P_{te} [Pa]	F_{obj}
69	5	178.7	366.5	1.70	8393	378.7
70	11	106.9	355.1	1.21	6056	485.1
71	10	152.3	339.1	1.40	5983	435.3
72	8	111.5	347.9	1.55	6352	403.2
73	5	141.9	354	1.67	9688	385.0
74	11	124.5	346.9	1.27	6004	469.8
75	6	108.3	349.3	1.58	8029	399.2
76	5	116.9	344.2	1.87	10833	357.6
77	8	166.8	354.2	1.51	6145	411.0
78	7	161.2	345.9	1.55	6531	403.4
79	9	158	341.1	1.13	7233	513.7
80	5	128.7	346.9	1.73	10407	376.0
81	10	143.5	365.2	1.20	6290	488.6
82	10	157.5	356.5	1.23	6180	479.0
83	6	120	350.5	1.64	7675	388.0
84	11	101.8	350.1	1.24	5981	475.7
85	9	165.5	359.7	1.24	6067	475.8
86	7	101.3	356.2	1.72	6503	373.5
87	9	114.3	337.7	1.47	6094	419.4
88	10	101.6	330.7	1.40	5669	435.0
89	7	122.5	351.6	1.71	6429	375.0
90	6	155.7	336.4	1.62	7331	391.2

Results with 3 Parameters and Pressure Dominated Function

n_{iter}	n_l	D_8 [mm]	D_7 [mm]	$\dot{m}_{te}$ [kg/s]	P_{te} [Pa]	F_{obj}
1	5	132.1	361.6	1.82	9357	692.3
2	12	131.5	350.3	1.18	5967	813.5
3	7	146.2	333.8	1.48	6463	737.9
4	8	146.6	335.7	1.47	6058	738.3
5	7	138.4	347.6	1.50	6613	734.3
6	10	161.4	330.9	1.35	5505	764.5

n_{iter}	n_l	D_8 [mm]	D_7 [mm]	$\dot{m}_{te}$ [kg/s]	P_{te} [Pa]	F_{obj}
7	5	130.2	342.6	1.69	10200	714.1
8	10	166.5	367.8	1.11	7089	843.1
9	6	111.5	341	1.51	7897	737.5
10	5	179.2	354	1.53	8774	737.1
11	10	136.9	346.3	1.34	5961	769.0
12	9	113.8	338.1	1.45	5974	742.4
13	11	109.8	348.2	1.11	7089	843.1
14	7	127.1	369.2	1.52	6270	728.7
15	7	125.2	344.5	1.48	6725	738.9
16	7	169.3	351.5	1.57	6273	718.9
17	9	141.4	358.6	1.26	6034	791.7
18	7	116.4	362.8	1.56	6310	721.0
19	11	148.9	333.5	1.28	5491	783.5
20	8	169.5	366.1	1.30	6229	780.9
21	9	121.9	357.8	1.29	6010	782.8
22	11	176.7	345.7	1.22	5782	799.8
23	8	144.1	355.4	1.11	7089	843.1
24	12	124.4	364.8	1.13	6461	833.1
25	11	172.1	332.2	1.26	5421	789.1
26	11	155.8	351.3	1.20	5963	806.9
27	11	151.4	339.2	1.26	5691	790.2
28	10	106.6	331.2	1.37	5585	759.7
29	8	152.8	337.7	1.47	6073	738.4
30	9	139.7	340.6	1.42	6035	749.5
31	11	102.4	352.3	1.22	5926	800.4
32	9	103.6	349.5	1.43	6119	747.5
33	6	154.7	359.5	1.68	6444	700.1
34	8	100.3	335.2	1.47	6278	739.2
35	9	163	344	1.40	6076	754.4
36	8	159	367.2	1.31	6208	778.1
37	10	134	356.6	1.22	5912	800.3
38	5	108.4	368.4	2.04	10058	667.4
39	9	174.5	353.8	1.34	6072	769.5
40	10	119.7	360.5	1.19	6046	810.5
41	12	119.5	363.7	1.12	6330	836.2
42	6	175.7	337	1.56	7062	724.1
43	6	158.1	356.8	1.80	6567	683.5

n_{iter}	n_l	D_8 [mm]	D_7 [mm]	$\dot{m}_{te}$ [kg/s]	P_{te} [Pa]	F_{obj}
44	6	165.2	364.5	1.58	6629	718.5
45	6	114.7	341.8	1.53	7820	733.2
46	6	117.9	368.4	1.70	6766	698.2
47	12	171.6	365.8	1.11	6600	841.1
48	6	117	361.8	1.67	7108	704.5
49	8	129.5	361.2	1.40	5988	754.0
50	5	160.8	347.6	1.49	9452	748.2
51	6	111.9	348	1.61	7715	717.5
52	5	132.8	336.1	1.73	9958	706.7
53	10	157.3	349.4	1.30	5999	780.0
54	6	162.3	357.7	1.68	6298	699.5
55	5	142.5	359.2	1.74	9083	701.5
56	10	128.2	335	1.11	7089	843.1
57	8	114.4	338.1	1.47	6233	739.1
58	8	149.7	357.6	1.46	6012	740.3
59	7	158	345.6	1.50	6470	733.7
60	6	125.2	363	1.65	7013	707.5
61	5	169.3	349.7	1.49	9188	747.1
62	11	176.1	360.5	1.11	7089	843.1
63	11	152.6	362.8	1.15	6217	824.9
64	8	148.9	358.4	1.41	5882	751.2
65	8	106.2	358	1.50	5952	731.5
66	6	121.9	345.3	1.53	7668	732.5
67	11	145.1	351.2	1.21	5948	803.6
68	5	175.4	355.4	1.60	8679	723.3
69	8	153.4	364.9	1.34	6163	769.8
70	8	150.3	362.1	1.33	6109	772.3
71	11	122.9	362.8	1.11	7089	843.1
72	6	151.4	341.8	1.55	7319	727.1
73	10	139.5	357.8	1.20	6026	807.2
74	11	127.3	332.3	1.30	5494	777.9
75	5	118	360.4	1.94	9836	678.3
76	11	133.3	338.8	1.28	5699	784.4
77	6	129.7	364.3	1.70	6824	698.4
78	6	154.7	340.8	1.55	7264	726.9
79	8	114.4	335.2	1.49	6110	734.3
80	8	154.8	344	1.48	6191	736.7

n_{iter}	n_l	D_8 [mm]	D_7 [mm]	$\dot{m}_{te}$ [kg/s]	P_{te} [Pa]	F_{obj}
81	5	159	354.8	1.58	9125	728.9
82	8	119.4	368.5	1.35	6136	767.1
83	8	132.1	349.1	1.54	6090	723.9
84	9	104.2	351.6	1.44	6079	745.1
85	9	120.7	360.5	1.27	6086	788.9
86	7	140.5	363.8	1.53	6216	726.5
87	6	175.7	337	1.56	7062	724.1
88	10	123.8	356.8	1.22	5993	800.7
89	9	146.2	364.5	1.11	7089	843.1
90	12	114.7	359.3	1.14	6224	828.5

Results with 4 Parameters and Base Objective Function

n_{iter}	n_l	D_8 [mm]	D_7 [mm]	D_6 [mm]	$\dot{m}_{te}$ [kg/s]	P_{te} [Pa]	F_{obj}
1	5	143.9	368.2	148.4	1.86	9192	357.3
2	7	100.7	364.7	118.8	1.61	6468	392.1
3	10	156.2	354.4	143.8	1.24	6217	476.0
4	10	117.8	361.2	85.6	1.11	7089	521.0
5	7	110.9	349.1	128.4	1.50	7012	414.0
6	12	166.5	347.3	98.3	1.18	5841	494.7
7	7	178.3	359.2	110.6	1.54	5732	404.5
8	10	160.8	357.3	123.4	1.22	6150	482.1
9	7	129.7	341.9	94.8	1.47	6659	420.0
10	11	157.5	346.2	126.1	1.22	5892	481.8
11	11	170.5	333.9	108.4	1.28	5545	466.4
12	9	137.4	357.6	149	1.36	6139	445.3
13	5	177.2	356.3	90.2	1.66	8418	385.3
14	7	127.1	345.4	115.2	1.51	6770	411.7
15	6	161.7	349.4	112.6	1.51	7298	412.2
16	5	164.2	361.4	123	1.70	8769	379.1
17	8	104.3	332.2	110	1.44	6220	426.2
18	8	121.5	366.8	91.7	1.35	6141	447.9

n_{iter}	n_l	D_8 [mm]	D_7 [mm]	D_6 [mm]	$\dot{m}_{te}$ [kg/s]	P_{te} [Pa]	F_{obj}
19	10	119.7	352.3	141.1	1.32	6275	456.0
20	11	150.3	363.8	93	1.13	6265	512.7
21	9	125.1	366.2	127.4	1.26	6153	472.9
22	8	167.1	343.3	84	1.45	6122	423.8
23	12	128.5	334.4	107.6	1.20	5332	487.6
24	12	151.2	330.9	130.7	1.11	7089	521.0
25	5	110.7	336.9	81.7	1.80	10654	367.2
26	10	126	342.1	86.2	1.36	5885	445.1
27	10	133.9	335.9	120.5	1.36	5855	445.1
28	6	102.9	368.8	88.7	1.65	7357	385.9
29	8	154.4	362.8	114.3	1.37	6155	442.8
30	9	137.1	338.4	87.8	1.11	7089	521.0
31	12	112.6	344.8	138.6	1.20	5879	488.2
32	8	175.6	339	83	1.44	6003	426.0
33	8	159.8	358.3	135.1	1.41	6036	433.0
34	11	115.6	364.8	99.7	1.16	6239	501.9
35	6	135.5	333.2	80.8	1.53	7316	408.2
36	8	142.4	336.6	137.4	1.48	6259	417.4
37	11	114	360.1	117.3	1.19	6222	491.8
38	9	168.2	347.5	133.7	1.39	6226	438.0
39	11	116.6	351.3	125.2	1.22	6076	482.0
40	9	163.2	334.8	117.4	1.43	5874	428.1
41	11	132.1	330.5	90.9	1.11	7089	521.0
42	9	149.2	343.5	102.4	1.11	7089	521.0
43	6	178.8	348.6	145.8	1.53	7454	408.3
44	9	105.8	341.1	101.9	1.44	6102	426.1
45	5	174.5	355.5	147	1.72	9100	376.2
46	10	108.6	367.4	137.1	1.18	6446	495.3
47	6	171.6	369.9	145.3	1.63	6819	388.9
48	6	173.1	350.6	140.2	1.50	7445	414.4
49	9	131	339.9	129.1	1.43	6146	428.4
50	10	144.3	331.9	100.8	1.35	5600	447.4
51	8	147.1	359.5	112.7	1.40	5910	435.2
52	7	155.2	340.2	132.7	1.50	6622	413.6
53	11	108	365.7	121.5	1.16	6298	502.0
54	6	102.5	354.9	132.1	1.51	8174	413.1

n_{iter}	n_l	D_8 [mm]	D_7 [mm]	D_6 [mm]	$\dot{m}_{te}$ [kg/s]	P_{te} [Pa]	F_{obj}
55	11	140	362.1	106.7	1.15	6199	505.4
56	7	146.3	337.8	97.1	1.52	6468	409.3
57	7	120.4	353.4	103.6	1.68	6262	379.7
58	10	140.1	353	105.5	1.28	6104	467.0
59	6	152.2	344.4	96.1	1.56	7219	402.2
60	7	123.3	351.5	142.3	1.66	6638	383.5
61	7	149.8	356.3	126	1.62	6035	389.8
62	12	173	352.9	118.8	1.19	6162	491.7
63	9	148.7	362.3	81.5	1.22	5875	481.8
64	7	144.7	362	122.8	1.57	6209	399.2
65	9	110.9	349.1	141.2	1.46	6368	421.9
66	9	128.7	354.3	116.8	1.39	6132	437.9
67	8	178.3	359.8	110.6	1.37	5926	442.6
68	6	160.8	365.9	134.7	1.63	6785	388.8
69	7	129.7	341.9	122.7	1.50	6788	413.7
70	9	103.5	338.2	128.5	1.44	6128	426.1
71	5	121.9	367.4	108.4	1.90	9576	352.4
72	9	113.2	357.6	96.9	1.32	6136	455.9
73	5	177.2	362.6	104.6	1.61	8365	394.0
74	6	120	348.9	100.4	1.54	7697	406.6
75	10	161.7	333.4	120.1	1.35	5689	447.4
76	9	121.2	357.9	101	1.30	6012	461.2
77	5	129.4	352.5	110	1.74	10078	374.1
78	11	134.5	355.3	88.7	1.16	5927	501.6
79	6	141.1	360.8	107.6	1.65	6840	385.4
80	10	136.5	362.9	93	1.19	6034	491.6
81	12	108.9	366.2	127.4	1.13	6460	512.9
82	9	113.1	343.3	84	1.42	6076	430.7
83	6	146.8	368.3	107.6	1.68	6662	380.2
84	6	168.3	354.3	108.2	1.62	6904	390.7
85	12	110.3	336.9	115.1	1.21	5428	484.5
86	10	126	358.8	140.2	1.19	6232	491.8
87	8	137	353.8	94.3	1.52	6003	408.8
88	10	102.9	352.3	88.7	1.29	5957	464.0
89	7	150.8	362.8	119.7	1.55	6098	402.9
90	8	137.1	338.4	102.9	1.11	7089	521.0

n_{iter}	n_l	D_8 [mm]	D_7 [mm]	D_6 [mm]	$\dot{m}_{te}$ [kg/s]	P_{te} [Pa]	F_{obj}
91	9	112.6	367.2	96.1	1.27	6086	469.9
92	5	143.9	352.8	83	1.61	9349	395.1
93	12	128.2	358.3	104.8	1.15	6297	505.5
94	5	107.4	348.7	109.1	1.96	11102	346.4
95	7	135.5	358.1	104.1	1.59	6297	395.5
96	6	105.5	354.9	126.5	1.11	7089	521.0
97	8	114	341.7	117.3	1.49	6369	415.4
98	8	120.7	357.5	113.3	1.49	5936	415.0
99	8	116.6	345.9	93.3	1.53	6187	407.0
100	10	126.8	350	101.2	1.33	6035	453.1
101	11	143	339.1	145.9	1.27	5900	469.7
102	9	116.8	360.8	97.5	1.28	5970	466.9
103	6	178.8	349.5	89.2	1.11	7089	521.0
104	9	155.1	348.6	101.9	1.39	6080	437.8
105	11	159.6	338	127.2	1.27	5734	469.5
106	11	116.2	346	128.8	1.27	6011	469.8
107	7	120	339.8	149.4	1.52	7000	409.9
108	9	123.2	350.6	114.6	1.46	6036	421.5
109	7	131	337.9	145.5	1.52	6807	409.6
110	6	144.3	343.4	99.4	1.52	7428	410.3
111	5	107.7	342.4	120.1	2.11	10958	329.2
112	6	165.7	345.9	132.7	1.50	7481	414.5
113	8	106	359.5	99.6	1.44	6002	426.0
114	5	102.5	354.9	143.9	2.41	11228	302.7
115	6	140	369.2	120.5	1.66	6875	383.7
116	9	146.3	361	113.5	1.11	7089	521.0
117	8	120	353.4	125.8	1.51	6303	411.2
118	11	115.3	353.4	136.3	1.22	6182	482.1
119	6	128.8	344.4	149	1.46	8103	423.7
120	8	123.3	341.7	133.1	1.48	6446	417.6