

https://dspace.org/

Calculation and design of supersonic nozzles for cold gas dynamic spraying using MATLAB and ANSYS Fluent.

Mbuyamba, Jean-Baptiste Mulumba

2013-07-22
http://hdl.handle.net/10539/12865

Downloaded from a DSpace repository. https://dspace.org/

CALCULATION AND DESIGN OF SUPERSONIC NOZZLES FOR COLD GAS DYNAMIC SPRAYING USING MATLAB AND ANSYS FLUENT

Jean-Baptiste Mulumba Mbuyamba

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, May 2013

Declaration

I declare that this dissertation is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other university.

Signed this __ day of _____ 20__

_____

Jean-Baptiste Mulumba Mbuyamba.

To my parents Georges Mulumba and Symphorose Ntumba.

Acknowledgments

I want to thank a few special people who made this dissertation possible:

Dr Ionel Botef for his guidance, encouragement, and patience

My postgraduate colleagues for their support

My brother Emmanuel Tshibanda and my further's wife Rita Shimba for their constant support, encouragement, and understanding.

Abstract

One of the most daunting challenges in the Cold Gas Dynamic Spray (CGDS) process is the calculation and design of the nozzles that are used to accelerate the gas and the powder particles at supersonic speeds and so promote the deposition process. Past research into this area resulted in a wealth of knowledge but unresolved problems still exist. The actual calculations and designs of the CGDS nozzles are considered large, complex, and time consuming. Consequently, this dissertation develops a new software that focuses on the simulation of the gas and particles velocities for a large variety of CGDS process parameters. However, in order to achieve this, an unified mathematical model of various cold spray parameter was developed. Thereafter, a new software using MATLAB was developed to generate practical graphs for the CGDS process and generate the 2D recommended nozzle contour, and the Computational Fluid Dynamics (CFD) software was used to calculate and visualize the gas flow. Then, the results obtained using the two developed technologies were compared with data from the peer reviewed journal papers and it was found that the results obtain using the new MATLAB software and ANSYS Fluent were very similar with data found in the literature survey. The dissertation ends with conclusions about the new approach for the calculation and design of the CGDS nozzles, and finally highlights its theoretical and practical implications.

Contents

Declaration i

Abstract iv

Contents v

List of Figures x

List of Tables xiii

List of Abbreviations xiv

List of Symbols xv

1 INTRODUCTION 1

1.1 Background of the Research 1

1.2 Justification of the Research 1

1.3 Research Problem 2

1.4 Delimitation of Scope 2

1.5 Source of Data and Methodologies 3

1.6 Contributions 4

1.7 Outline of the Dissertation 4

1.8 Conclusion 5

2 LITERATURE SURVEY **6**

2.1 Introduction 6

2.2 Background 6

2.3 Gas Dynamics of De Laval Nozzles in Cold Spray 8

2.3.1 Assumptions for developing gas flow equations 9

2.3.2 The choice of gas 10

2.3.3 Mach Number and regimes of compressible flow 11

2.3.4 Isentropic relations 11

2.3.5 Gas conditions at the nozzle throat 12

2.3.6 Nozzle area–Mach number relation and gas conditions at the nozzle exit 14

2.3.7 Shock waves at nozzle exit 16

2.3.8 Particle velocity 18

2.3.9 Particle critical velocity 19

2.4 Gas Dynamics of MOC Nozzle 20

2.4.1 Theoretical background 20

2.4.2 Prandlt Meyer function 21

2.4.3 MOC for the steady of two dimensional supersonic flow . 23

2.4.3.1 Principe of numerical method 23

2.4.3.2 General procedure for solving the velocity potential equation 25

2.4.3.3 Minimum length nozzle 30

2.5 Optimization of the CGDS Process by Improving Nozzle Design 31

2.6 Flow Analysis using ANSYS Fluent Software 32

2.7 Conclusion 33

3 METHODOLOGY 34

3.1 Introduction 34

3.2 Parameters for the De Laval nozzle simulation 34

3.3 MOC nozzle design 37

3.4 Test diagram 41

3.5 Summary 41

4 SOFTWARE DEVELOPMENT AND ANALYSIS OF DATA 42

4.1 Introduction 42

4.2 GUI Development 42

4.3 Calculation Process and Design for De Laval Nozzle 45

4.3.1 Calculation Process 45

4.3.2 Introduction to the De Laval nozzle GUI 47

4.3.3 Program testing and results analysis 52

4.3.3.1 Test 1 52

4.3.3.2 Test 2 56
4.3.3.3 Test 3 60
4.4 MOC Nozzle Design 66
4.4.1 GUI interface 66
4.4.2 Test results and analysis 70
4.4.3 Flow analysis using CFD method with ANSYS Fluent . 72
4.4.3.1 Nozzle geometric details and mesh generation . 73
4.4.3.2 Simulation 74
4.4.3.3 Analysis of results 77
4.5 Summary 78

5 CONCLUSION AND FUTURE WORKS 79

5.1 Introduction 79
5.2 A Brief Overview of Previous Chapters 79
5.3 Conclusions about the Reseach Questions 80
5.4 Research Implications 81
5.5 Research Limitations 81
5.6 Further Research 82

REFERENCES 83

BIBLIOGRAPHY 86

APPENDICES **87**

Appendix A - Glossary 87

Appendix B - CGDS operating parameters according to MIL-STD-3021 89

Appendix C - Matlab Code for De Laval nozzle 90

Appendix C1 - Code for Construction the GUI 90

Appendix C2 - Code for Testing Plot Results 100

Appendix C21 - Code for Test 1 100

Appendix C22 - Code for Test 2–Variation of pressure . . 102

Appendix C23 - Code for Test 2–Variation of Temperature 104

Appendix D - Matlab Code for MOC Nozzle 106

Appendix D1 - Code to the curved part of the MOC nozzle . . . 106

Appendix D2 - Code GUI for MOC nozzle 113

List of Figures

2.1 Low pressure CGDS system 7

2.2 Isentropic supersonic nozzle flow 13

2.3 Variation of the gas Mach number with the nozzle expansion ratio 16

2.4 Stationary normal shock wave 17

2.5 Generalized dependency of relative particle velocity 19

2.6 Comparison of calculated v_{crit} with experimental results 20

2.7 Prandtl-Meyer Expansion 22

2.8 Geometric construction for the infinitesimal changes across a Mach wave 22

2.9 Rectangular finite-difference grid 24

2.10 Illustration of left-and right-running characteristic lines 26

2.11 Unit processes for the steady-flow 28

2.12 Approximation of characteristics by straight line 29

2.13 Flow field presentation in Minimum Length Nozzle 30

2.14 Comparison between the gas jets generated by a standard and a MOC nozzle 31

2.15 Comparison between particle velocities of a standard and MOC nozzle 32

2.16 Performance evolution of the deposition of nickel 32

3.1 Flowchart for De Laval Nozzle calculations. 35

3.2 Flowchart for MOC cozzle calculations. 38

3.3 Lines details. 40

3.4 Test diagram. 41

4.1 Property Inspector. 43

4.2 Example of M-file for GUI. 44

4.3 Main interface. 44

4.4 Main window of the tool used to simulate the De Laval nozzle. . 50

4.5 Results for Cu particles using Nitrogen 51

4.6 Effect of N_2 temperature on the velocity of particles for different sizes 53

4.7 Comparison between the particle velocities in reference and this Work for Cu powder 54

4.8 Effect of N_2 pressure on the velocity of particles 55

4.9 Comparison between particle velocities in reference and this Work for Cu powder 56

4.10 Temperature and velocity of Cu particles 57

4.11 Comparative between particle velocities in reference and this work for Cu powder 58

4.12 Temperature and velocity of Cu particles at the nozzle exit . . . 59

4.13 Comparison between the particle velocities in reference and this work for Cu powder 59

4.14 Results for Aluminium particles using Nitrogen 61

4.15 Mesh used for simulations. 62

4.16 Contours for the velocity (m/s) using Nitrogen 63

4.17 Contours for the static pressure (Pascal) using Nitrogen 64

4.18 Contours for the static temperature (Kelvin) using Nitrogen . . 64

4.19 Contours for the Mach number for De Laval nozzle 65

4.20 Main window for the MOC nozzle contour simulation. 68

4.21 Contour of the MOC nozzle 69

4.22 Plot properties vs Mach number as planned. 72

4.23 Example of mesh used for simulations. 74

4.24 Results for Cu particles using Nitrogen 75

4.25 Contours of gas velocity (m/s) for a MOC nozzle 77

List of Tables

4.1 Comparative results between the reference and this work for Cu particle velocity 53

4.2 Comparative results between the reference and this work for Cu particle velocity 55

4.3 Comparative table for Cu particle velocity at different pressures 57

4.4 Comparative table between the reference and this work for particle velocity for Cu powder 58

4.5 Mach Number comparative table for MOC nozzle design 71

4.6 Coordinates (x,y) for the MOC nozzle wall profile 73

5.1 Operating parameters 89

List of Abbreviations

CGDS	Cold Gas Dynamics Spraying
CFD	Computational Fluid Dynamic
CS	Cold Spray
GUI	Graphic User Interface
MOC	Method of Characteristic
PM	Prandtl Meyer
SI	System International

List of Symbols

A	nozzle cross-sectional area (m^2)
A_e	nozzle exit cross-sectional area (m^2)
A_p	particle projected area (m^2)
A^i	nozzle powder entry point cross–sectional area (m^2)
$A*$	nozzle throat cross-sectional area (m^2)
C_D	drag coefficient
C_p	gas heat capacity at constant pressure ($J/kg\,K$)
C_v	gas heat capacity at constant volume ($J/kg\,K$)
G_g	gas flow rate (kg/s)
G_p	powder particle flow rate (kg/s)
F_1	mechanical calibration factor
F_2	thermal calibration factor
M	Mach number
M_e	nozzle exit Mach number
N_u	Nusselt number
P	pressure (Pa)

P_a	ambiante pressure (Pa)
P_e	nozzle exit pressure (Pa)
P_g	gas pressure (Pa)
P_o	stagnation pressure (Pa)
P_s	shock pressure (Pa)
P_{shock}	pressure behind the shock waves (Pa)
P^*	nozzle throat gas pressure (Pa)
Pr	Prandtl number
R	gas constant $(J/kg\,K)$
R_e	Reynolds number
T	temperature (K)
T_p	particle temperature (K)
T_m	particle melting point (℃)
T_i	the initial particle temperature (℃)
T_g	gas temperature (K)
T^*	nozzle throat gas temperature (K)
T_e	nozzle exit particle temperature (K)
T_o	stagnation temperature (K)
V	volume (m^3)
V^*	nozzle throat gas volume (m^3)
c	speed of sound in materials (m/s)

c_p	particle heat capacity $(J/kg\,K)$
a_p	average particle acceleration (m/s^2)
c_g	gas heat capacity $(J/kg\,K)$
d	diameter (m)
d_p	powder particle diameter (m)
m	mass (kg)
$\dot{m}$	mass flow rate of the gas (kg/m^3)
$\dot{m}_v$	gas flow rate of the gas (m^3/s)
m_p	average powder particle mass (kg)
r_p	powder particle radius (m)
t	time (s)
v	velocity (m/s)
v_{crit}	critical impact velocity (m/s)
v_g	gas velocity (m/s)
v_p	particle velocity (m/s)
v^*	nozzle throat gas velocity (m/s)
x	distance (m)
σ_u	particle yield stress (Pa)
σ_{TS}	the tensile strength (Pa)
ρ	density (kg/m^3)
ρ_o	stagnation gas density (kg/m^3)

ρ_g	gas density (kg/m^3)
ρ_p	particle material density (kg/m^3)
ρ^*	nozzle throat gas density (kg/m^3)
γ	specific heat ratio (C_p/C_v)
•	**Subscripts**
c	coating
e	nozzle exit
g	gas
o	stagnation
p	particle
•	**Superscripts**
$*$	nozzle throat

1 INTRODUCTION

1.1 Background of the Research

This dissertation calculates and designs the internal profile of a supersonic nozzle for Cold Gas Dynamic Spray (CGDS) process. CGDS is a relatively new spray coating technique capable of depositing a variety of materials without extensive heating [25]. The function of the nozzle is to convert the slow moving, high pressure, high temperature gas into high velocity, lower pressure, and lower temperature of the gas [17]. Furthermore, this supersonic jet of gas is used to accelerate small and unmelted particles in size between 5 to 50 $\mu\, m$ and so achieve particle velocities of 600 to 1000 m/s [7].

Upon impact with a target surface, the solid particles deform and bond together, and rapidly build up a layer of deposited material. As a result, the inherent problems found during the traditional thermal spraying processes, such as oxidization, particle melting, grain growth, and residual tensile stress, to name only a few, can be avoided [25]. However, the coating final properties, such as micro structure, strength, and porosity are directly affected by Cold Spray process parameters such as particle properties, gas pressure, and gas temperature.

1.2 Justification of the Research

CGDS process requires a supersonic high velocity stream of gas to accelerate the powder particles at velocities exceeding particle's critical velocity [38]. The

Critical Velocity is the lowest impact velocity for a particle of a specific material to be deposited. However, many times, CGDS experiments were carried out using process parameters obtained from similar published literature, using ad hoc and untested software developed for example in Microsoft Excel, or using the try and error approach. Therefore, the optimization of the nozzle's geometry considering its influence upon powder particles became critically imperative. Consequently, the development of a new software that will allow the simulation of the gas and particles velocities for a large variety of CGDS process parameters will avoid important waste of equipment set–up time, avoid the premature degradation of the CGDS equipment, and also avoid a waste of important quantity of expensive powder.

1.3 Research Problem

The problem addressed in this research is:

How to calculate and design the internal profile of a CGDS nozzle and so effectively and successfully achieve a Cold Spray deposition?

Essentially, it is argued that the nozzle design must be calculated and verified using advanced computerized tools such as MATLAB and ANSYS, and that, in order to do this, an in depth knowledge of fluid dynamics is necessary.

1.4 Delimitation of Scope

The dissertation research proposes and develops the practical technologies for the design, testing, and analysis of the nozzles used in the CGDS process. The dissertation aims to:

- develop a MATLAB software capable to generate practical graphs for the CGDS process and generate the 2D recommended nozzle contour,

- use the Computational Fluid Dynamics (CFD) method to calculate and visualize the gas flow, and
- test the two developed technologies using data from peer reviewed journal papers.

Considering the above, it is important to indicate that this research dissertation will have its own limitations. These refer to:

- the De Laval nozzle profile is limited to a straight lines profile from the throat section to exit section,
- the design of MOC nozzle is limited to the determination of the internal shape for the divergent part of the supersonic nozzle,
- the complete nozzle design and its manufacturing is excluded, and
- the use of ANSYS Fluent software be limited to the determination of the gas velocity.

1.5 Source of Data and Methodologies

The investigative procedures to be adopted will basically be guided by the aims highlighted above. These include:

- gather CGDS process information from research publications with the main objective focused on the nozzle design,
- use the compressible flow theory to build a unified theory for the calculation of particles velocity in the CGDS process,
- use the general theory, called the *'Method of Characteristic'* (MOC) to build a model for generating of a two dimensional *'minimum – length'* nozzle for different gas – particle expansions and different exit Mach numbers,
- development a powerful MATLAB computational software that will handle all engineering calculations and desire parameters for the CGDS nozzle, and

- use the Computational Fluid Dynamics (CFD) method to simulate the gas flow in a MOC nozzle.

1.6 Contributions

In summary, the contributions of this dissertation include:

- the development of a unified mathematical model for the calculation and design of the nozzles for CGDS process,
- the development of a new MATLAB software capable to calculate the performance of the De Laval nozzles, and
- the development of a new MATLAB software capable to calculate and design the internal profile of the high performance (high gas speed with no or reduced shock waves) MOC nozzles.

1.7 Outline of the Dissertation

This dissertation is organized in five chapters which are structured, unified, and focused on solving one research problem. Each chapter has an introductory section which outlines its aim and, a concluding summary section which outlines the major themes established within it. The first chapter introduces the research problem and outlines the dissertation. Chapter 2 is a literature survey of the study. Chapter 3 reviews the methodology employed in carrying the calculations and design of nozzles for CGDS. Chapter 4 is devoted to software development and the analysis of data, and finally, chapter 5 presents the research achievements, its limitations, and some recommendations for future work.

1.8 Conclusion

This chapter has laid the foundations for the dissertation. It introduced the research problem and research issues, and also presented its aims and its limitations. Then, the methodologies were briefly described and justified, the contributions briefly highlighted and finally, the dissertation was outlined. On these foundations, the dissertation can proceed with a detailed literature survey.

2 LITERATURE SURVEY

2.1 Introduction

This chapter contains the literature survey related to the calculation and design of the internal profile of the De Laval and MOC nozzles used in the CGDS process.

2.2 Background

The design of the nozzle plays a critical role in the success of the CGDS process. For example, it was demonstrated that, if the cold–spray nozzle is designed in such a way that at each axial location the acceleration of the powder particles is maximized, a significant increase in the average velocity of the particles at the nozzle exit can be obtained [13]. In the same context, the mechanism of attachment of the particles on the substrate advocated that the speed of the powder particles at nozzle exit must be maximized. While, in general, this could be accomplished by increasing the inlet pressure of the carrier gas, for practical and economic reasons, it is desirable to maximize the particle impact velocity at a given level of the carrier gas inlet pressure by properly selecting the type of the carrier gas and its inlet temperature, and by optimizing the shape of the converging–diverging cold-spray nozzle [13]. A schematic illustrating the CGDS principle is presented in the Figure 2.1.

In order to determine the Mach number in a known section of the divergent part of the nozzle, and then, with the Mach number known, other parameters such as gas pressure, temperature, speed, and density to be determined, Dykhuizen

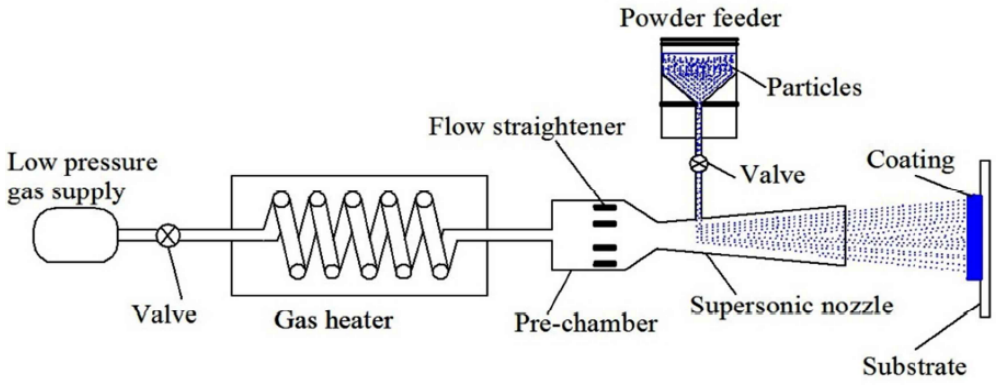

Figure 2.1: Low pressure CGDS system [23].

and Smith [7] developed a one dimensional theory they called the *Gas Dynamic Principles of Cold Spray*. Their theory provided a starting point for a more detailed experimental or numerical determination of an optimal nozzle. However, their theory did not provide a way to determine the internal profile shape of the divergent part of the nozzle.

Consequently, Al-Ajlouni [1] suggested an automatic method for the determination of the supersonic convergent-divergent nozzle profile. In his so–called MOC approach, a unit model matrix for each Mach number was initially created. Then, a Visual Basic program was developed to automatically determine the profile of the nozzle by multiplying the unit model matrix by a scale factor that is calculated according to the working requirements. However, the development provided by Al-Ajlouni was limited to Mach numbers less than or equal to 2.5.

Also, in a later study, Khine et al. [17] developed a numerical approach for the determination of the supersonic nozzle flow pattern. Their approach assumed the gas being inviscid, ideal, shock–free, and non–rotational. With these assumptions, and focusing only on the calculation of the flow properties inside the divergent section of the nozzle, they predicted the performance of the nozzle by calculating the loads induced by the aerodynamic flow. Then, in order to verify the structural integrity of the nozzle, the temperature distribution in the nozzle wall was calculated.

Furthermore, Karimi et al. [16] used Khine's method to predict the pressure on the nozzle wall, and compared these values to the available experimental data. They used the Computational Fluid Dynamic (CFD) model to simulate the gas dynamic flow field and particle trajectories within and outside of an oval–shaped supersonic cold spray nozzle, and analyzed the particles before and after the impact with the substrate.

Finally, from the above, it could be seen that, over the years, various researchers have attempted to develop various computerized tools to calculate, visualize, and better understand the CGDS process. However, there is still insufficient information on the calculation and design of CGDS nozzles in general, and also there are no commercially available computational tools to calculate and design the internal profile of the De Laval and MOC nozzles used in the CGDS process.

2.3 Gas Dynamics of De Laval Nozzles in Cold Spray

A nozzle is a device conceived to assure specific characteristics of the fluids (gas or liquid) flowing through it. During this flow, the thermal energy of the fluid is converted to kinetic energy, so the velocity of the fluid is increased. The cross sectional area of the nozzle can be circular, rectangular, square or oval. This dissertation deals only with circular section nozzle, however other sections could be calculated by approximating of their cross section area to the circular section area. The choice of a specific section could depend of specific application.

A *De Laval nozzle* is a nozzle obtained using the theory of *Quasi-one-dimensional flow*. In this context, it was demonstrated that when a fluid moves at speeds comparable to its speed of sound the density of the fluid changes become significant, and the flow is termed compressible. Such flows are difficult to obtain in liquids, however in gases, a pressure ratio of only 2/1 will likely cause a sonic flow. Thus, compressible gas flow is quite common, and this subject is

often called *Gas Dynamics* [41].

2.3.1 Assumptions for developing gas flow equations

In order to develop the flow equations that will allow the design of a CGDS nozzle, the following assumptions and simplifications are considered [14],[23],[12]:

- The gas flow is assumed to be quasi–one–dimensional. This refers to a flow where the cross sectional area $'A'$ of the nozzle, the gas pressure P_g, the velocity of gas v_g, and the gas density ρ_g are varying along one direction, say x, and a linear nozzle geometry is used.

- The model assumes an isentropic flow. This refers to an adiabatic flow (no heat transfer) which is frictionless (ideal or reversible). With the isentropic approach, the presence of the boundary layer in the region adjacent to the nozzle wall is not considered, consequently, the calculated velocity of the gas flow is slightly higher than if obtained in practice.

- The gas is treated as a perfect (ideal) gas, which is expressed by the *equation of state*:

$$P_g = \rho_g R T_g \tag{2.1}$$

  where P_g is the fluid absolute pressure,T_g is the absolute temperature, and R is the gas constant. For an ideal gas C_v and C_p are constant, so $R = C_p - C_v$ and $\gamma = \frac{C_p}{C_v}$ [42]. Therefore, considering that the gas flows from a state 1 to a state 2, the following important simplification for the isentropic flow is obtained:

$$\frac{P_2}{P_1} = \left(\frac{\rho_2}{\rho_1}\right)^{\gamma} = \left(\frac{T_2}{T_1}\right)^{\frac{\gamma}{\gamma-1}} \tag{2.2}$$

- Expansion of the gas occurs in a uniform manner, thus the flow is continuous and shock–free.

- The gas conditions are not influenced by the condition of gas–particle two–phase flow.

- The one–dimensional analysis is limited to the application of the model to regions away from the jet impingement on the substrate.

2.3.2 The choice of gas

The gas used in the CGDS process is assumed to come from a chamber with a stagnation condition. The stagnation state is defined as a state that would be attained by the fluid if it is conveyed to rest in isentropic state and without work. The properties at the stagnation state are refereed to as *stagnation properties* or *total properties*, and are designated by the subscript $'0'$ [12]. Thus, the gas condition is defined by the gas stagnation pressure (P_o), the gas stagnation temperature (T_o) and the mass flow rate of the gas ($\dot{m}$). All these parameters are set by the user.

Generally, the cost and safety of the CGDS process are affected by the choice of the gas used. Ideally, in order to transfer sufficient momentum to the powder, the gas must have a high sonic velocity and mass [38].

Typical operating gases used in CGDS process are:

1. Helium,
2. Nitrogen (N_2),
3. air, or
4. a mixture of the above.

The two main gases used in cold spray are Helium with a specific heat ratio of $\gamma = 1.66$ and Nitrogen with $\gamma = 1.4$. Both Helium and Nitrogen are inert gases. Helium has a high sonic velocity that is approximately three times that of the Nitrogen, but it is more expensive. However, this penalty can be overcome by using a gas recycling system but which also increases the price of the CGDS system. Finally, the sonic velocity of air (a diatomic gas) is slightly less than that of pure Nitrogen, but this option remains the cheapest CGDS process gas available [38].

2.3.3 Mach Number and regimes of compressible flow

The most important parameter in the analysis of the compressible flow is the *Mach Number* defined by:

$$M = \frac{v}{c} \tag{2.3}$$

where $'v'$ is the local flow velocity and $'c'$ is the local *speed of sound*.

Considering an ideal gas, the speed of sound is given by [34]:

$$c = \sqrt{\gamma RT} \tag{2.4}$$

where $'\gamma'$ is the specific heat ratio and $'T'$ is the absolute fluid temperature.

The *Mach Number* can be used to characterize the different regimes of flow [12]. These include:

- incompressible flow, where the Mach Number is very small compared to the unit (M<0.3)
- subsonic flow, where the Mach Number is less than unity, but large enough so that compressible flow properties are present (0.3<M<1).
- sonic flow, where the Mach Number is at unity ($M = 1$).
- transonic flow, the Mach Number is very close to the unity (0.8<M<1.2).
- supersonic flow, where the Mach Number is larger than the unity ($M > 1$).
- hypersonic flow, where the Mach Number is larger than five ($M > 5$).

2.3.4 Isentropic relations

Let's consider the stagnation point with v_o equal zero and P_o equal to the total pressure in the flow. At a point in the duct where the flow is undisturbed, and considering the basic fluid dynamics and thermodynamic relations for compressible flow, the energy equation is given by:

$$c_p T_g + \frac{1}{2} v_g^2 = c_p T_o \tag{2.5}$$

which implies

$$\frac{T_o}{T_g} = 1 + \frac{v_g^2}{2C_pT_g} \tag{2.6}$$

Furthermore, because $R = C_p - C_v$ and $\gamma = \frac{C_p}{C_v}$, these can be developed to get C_p as:

$$C_p = \frac{\gamma R}{\gamma - 1}$$

Therefore, by combining the two preceding equations, the following equation is achieved:

$$\frac{T_o}{T_g} = 1 + \frac{\gamma - 1}{2}\frac{{v_g}^2}{\gamma R T_g} \tag{2.7}$$

Substituting (2.3) and (2.4) into the above expression, the new equation can be expressed as function of gas local Mach Number:

$$\frac{T_o}{T_g} = 1 + \left(\frac{\gamma - 1}{2}\right) M^2 \tag{2.8}$$

Furthemore, by using the isentropic simplifications, the following relations can be deducted:

$$\frac{P_o}{P_g} = \left[1 + \left(\frac{\gamma - 1}{2}\right) M^2\right]^{\frac{\gamma}{\gamma-1}} \tag{2.9}$$

and

$$\frac{\rho_o}{\rho_g} = \left[1 + \left(\frac{\gamma - 1}{2}\right) M^2\right]^{\frac{1}{\gamma-1}} \tag{2.10}$$

Finally, using the equations above, Anderson [34] produced plots for $\frac{P}{P_o}$ and $\frac{T}{T_o}$ as a function of position along the nozzle (Figure 2.2). At the throat condition, the values of $\frac{P}{P_o} = 0.528$ and $\frac{T}{T_o} = 0.833$ where obtained by replacing M with 1.

2.3.5 Gas conditions at the nozzle throat

At the nozzle throat sonic conditions exist, so the Mach Number $M = 1$. At this point, all symbols are denoted by an asterisk, so the isentropic relations become:

$$\frac{T^*}{T_o} = \frac{2}{\gamma + 1} \tag{2.11}$$

$$\frac{P^*}{P_o} = \left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma}{\gamma-1}} \tag{2.12}$$

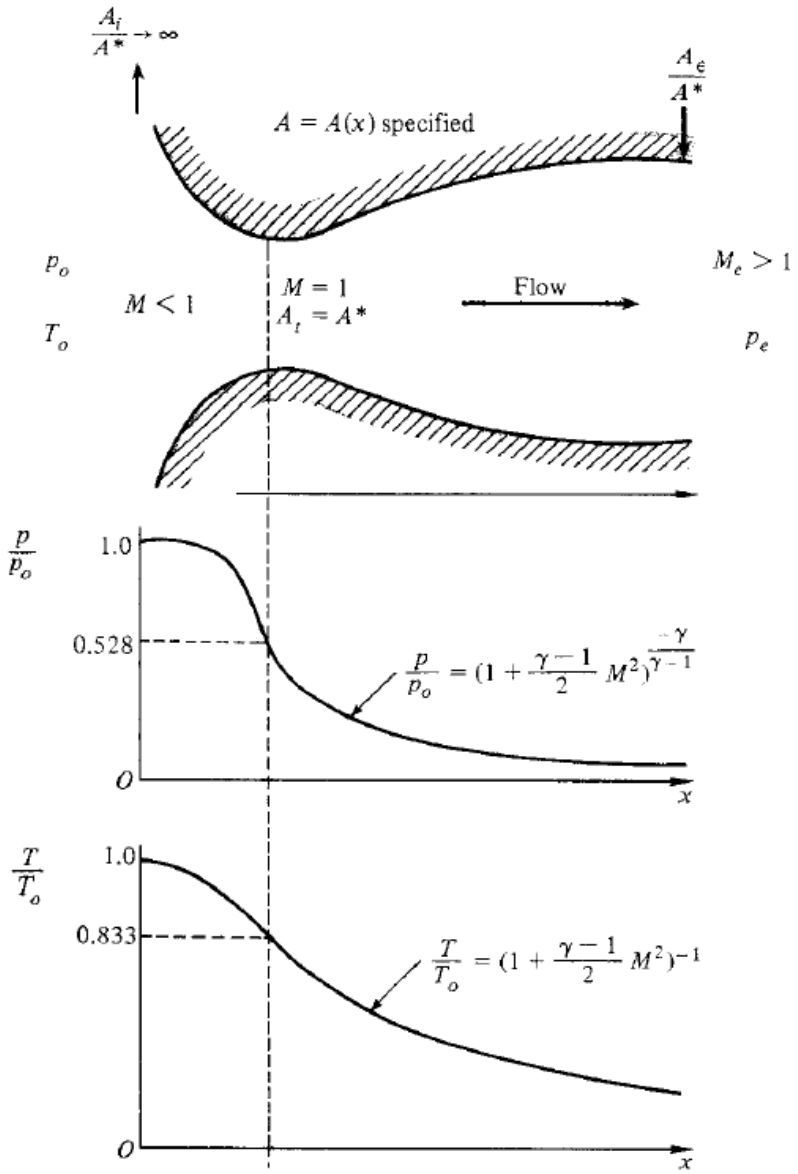

Figure 2.2: Isentropic supersonic nozzle flow [34].

$$\frac{\rho^*}{\rho_o} = \left(\frac{2}{\gamma + 1}\right)^{\frac{1}{\gamma - 1}} \tag{2.13}$$

$$v_g^* = \sqrt{\gamma R T_g^*} \tag{2.14}$$

$$\frac{c^*}{c_o} = \left(\frac{2}{\gamma + 1}\right)^{\frac{1}{2}} \tag{2.15}$$

$$\rho^* = \frac{\dot{m}}{v_g^* A^*} \tag{2.16}$$

where

- $\dot{m}$ is mass flow rate as the flux per unit throat area,
- c^* is the speed of sound,
- ρ^* is gas density (kg/m^3) at the throat of the nozzle,
- A^* is nozzle throat cross-sectional area (m^2),
- R is the gas constant.

Also, equation (2.14) explains why Helium is a better carrier of gas than air. Helium has low molecular weight, so R is large. Helium is also monoatomic, so γ is large, therefore T^* becomes high. As a result, Helium velocity is high compared to that for air.

Finally, when the conditions at the throat are known, it is possible to determine gas conditions along the diverging section of the nozzle.

2.3.6 Nozzle area–Mach number relation and gas conditions at the nozzle exit

When the quantities change at the nozzle throat, the Mach number or the nozzle cross sectional area, must be determined along the divergent section. Therefore, the continuity relation of Fluid Mechanics is involved that gives the following relation:

$$\dot{m} = \rho v A = \rho^* v^* A^* \tag{2.17}$$

Furthermore, the perfect-gas and isentropic–flow relations are used to convert the relation above into an algebraic expression that only involves area and Mach number:

$$\frac{A}{A^*} = \frac{\rho^* v^*}{\rho v} = \frac{\rho^* c^*}{\rho v} = \frac{\rho^* \rho_o c^*}{\rho_o \rho v} = \frac{\rho^* \rho_o 1}{\rho_o \rho M^*} \tag{2.18}$$

Also, using the isentropic relations and after some algebra, the *area-Mach number relation* is obtained as follows:

$$\left(\frac{A}{A^*}\right) = \frac{1}{M^2}\left[\frac{2}{\gamma+1}\left(1+\frac{\gamma-1}{2}M^2\right)\right]^{\frac{\gamma+1}{\gamma-1}} \tag{2.19}$$

However, it must be noted that the above equations reflect the gas conditions at the nozzle exit only if a normal shock does not take place inside the nozzle. In addition, the nozzle exit condition needs to be specified in order to complete the gas dynamic calculation. Therefore, Equations (2.8), (2.9), (2.10), and (2.14) could be adapted for the nozzle exit conditions and so become:

$$\frac{P_e}{P^*} = \left(\frac{\gamma+1}{2+(\gamma-1)M^2}\right)^{\frac{\gamma}{(\gamma-1)}} \tag{2.20}$$

$$\frac{T_o}{T_e} = 1+\frac{\gamma-1}{2}M^2 \tag{2.21}$$

$$v_e = M\sqrt{\gamma R T_e} \tag{2.22}$$

$$\frac{\rho_o}{\rho_e} = \left(1+\frac{\gamma-1}{2}M^2\right)^{\frac{1}{(\gamma-1)}} \tag{2.23}$$

Furthermore, Equation (2.19) is a non–linear algebraic equation. Therefore, Grujicic at al. [14] constructed an analytical function using the approximation presented in Figure 2.3, and so calculate the inverse of *area–Mach number relation*. In this respect a *non–linear least squares procedure* is used to accommodate the value of Mach number versus area–ratio data for different values of specific heat ratio. As a result, the following relation is presented:

$$M = \left[k_1\frac{A}{A^*} + (1-k_1)\right]^{k_2} \tag{2.24}$$

where k_1 and k_2 are functions of the specific heat ratio and with values given by a *non-linear polynomial regression* analysis as

$$k_1 = 218.0629 - 243.5764\gamma + 71.7925\gamma^2 \tag{2.25}$$

$$k_2 = -0.122450 + 0.281300\gamma \tag{2.26}$$

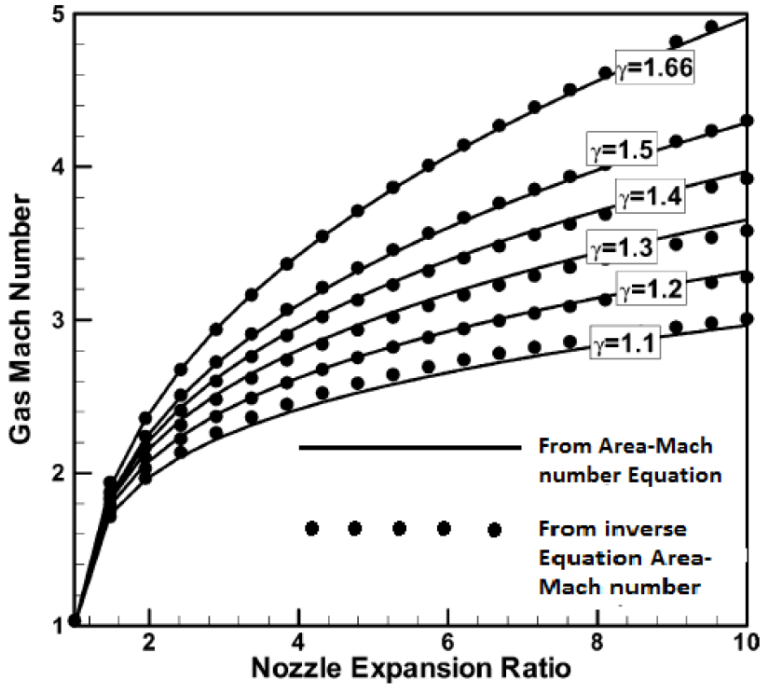

Figure 2.3: Variation of the gas Mach number with the nozzle expansion ratio and the gas specific-heat ratio γ [14].

2.3.7 Shock waves at nozzle exit

A shock wave is a thin region where the transition from supersonic velocity with low pressure state to low velocity with high pressure state occurs [40]. When the flow velocity exceeds the speed of sound, adjustments in the flow take place at these discontinuous regions. This is reflected by oscillations of v_g near the nozzle exit. In practical situations, the shock waves that occur at right angles to the flow path are termed a *normal shock*, whilst a shock wave that occurs at an angle to the flow path is termed an *oblique shock*. Figure 2.4 shows a example of normal shock wave.

To determine whether the normal shock will take place inside the nozzle, it is recommended to compare the ambient pressure with the 'shock' pressure given by Equation (2.27) [7]:

$$\frac{P_s}{P_e} = \frac{2\gamma}{\gamma+1} M_e^2 - \frac{\gamma-1}{\gamma+1} \tag{2.27}$$

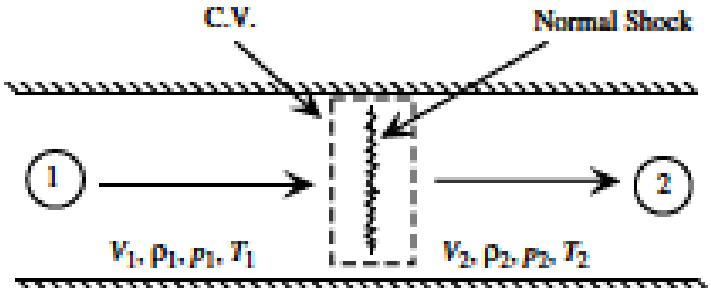

Figure 2.4: Stationary normal shock wave [12].

where P_s is the downstream shock pressure that would be obtained if a shock occurred at the nozzle exit and P_e is the exit pressure.
Note that, if the shock pressure P_s is equal to the ambient pressure, a shock occurs at the nozzle exit. If the shock pressure P_s is lower than the ambient pressure, a shock will occur inside the nozzle, so the gas flow is considered subsonic and the exit pressure is not given by Equation (2.20), but is equal to the ambient pressure.

However, for operating conditions in CGDS, the shock pressure P_s is maintained above the ambient pressure, so no shock occurs inside the nozzle and P_e is defined by Equation (2.20). Also, P_e is generally lower than the ambient pressure in order to increase the exit velocity of the gas and consequently, the average velocity of the feed powder particles.

In addition, a certain length of divergent section of nozzle could not be exceeded, otherwise a normal shock occurs inside. Furthermore, increasing the nozzle length downstream of the nozzle throat, the boundary layer thickness also increases. This leads to a decrease of the effective nozzle cross–sectional area in comparison to the geometrical cross–sectional area, and consequently, gas velocity decreases at the nozzle exit in comparison to the ideal gas flow velocity [2].

2.3.8 Particle velocity

Once the gas conditions and velocity are characterized, the particles are analyzed using a particle motion model. To calculate the particle velocity v_p, Alkimov et al. used the simple particles motion equation as follows [2]:

$$m_p v_p \frac{dv_p}{dz} = C_D \frac{\rho \left(v - v_p\right)^2}{2} S_{mid} \tag{2.28}$$

$$M_p = \frac{v - v_p}{c} \tag{2.29}$$

$$Re_p = \frac{\left(v - v_p\right) \rho d_p}{\mu} \tag{2.30}$$

where m_p is the particle mass, v_p is the particle velocity, z is the coordinate along the nozzle axis, C_D is the drag coefficient, ρ is the gas density, v is the gas velocity, S_{mid} is the cross-sectional area of the particle, M_p is the particle Mach number, c is the gas sound speed, Re_p is the particle Reynolds number, d_p is the particle diameter and μ is the viscosity. Note that the gas parameters are taken near the axis and the drag coefficient is calculated using *Henderson approximation* [2].

Furthermore, in order to solve equation (2.28) Alkimov et al. [2] determined a complex element noted $'\Omega'$, that will characterize and bind the elements of the equation, and so find the range where the relation (2.28) will be applicable (Figure 2.5) [2]:

$$\Omega = \left(\frac{d_p}{L}\right)^{0.5} \left(\frac{\rho_p v_g^2}{P_o}\right)^{0.5} \tag{2.31}$$

where d_p is the particle diameter, L is the length of divergent part, ρ_p is the density of particle material, v_g is gas velocity at the nozzle exit and P_o is the stagnation pressure. When Nitrogen is used as the process gas, v_p is determined by the equation:

$$\frac{v_p}{v_g} = \frac{1}{1 + 0.85\Omega} \tag{2.32}$$

Analyzing the correlation between the theoretical and experimental velocities of particles, the more explicit form of equation (2.32), valid for Nitrogen and Helium is given by Wu et al. as follow [30]:

$$v_p = \frac{v_g}{1 + 0.85\sqrt{\frac{D}{x}}\sqrt{\frac{\rho_p {v_g}^2}{P_o}}} \tag{2.33}$$

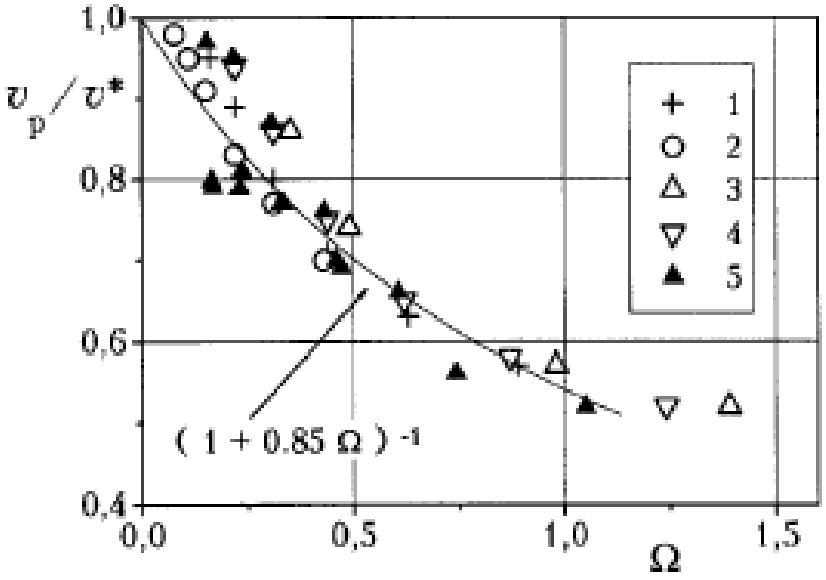

Figure 2.5: Generalized dependency of relative particle velocity at outlet of the flat supersonic nozzle [2].

where v_p is the particle velocity, P_o is the Nitrogen supply pressure measured at entrance of the entrance of the nozzle, ρ_p is the particle density, D is the particle diameter and x is the axial position.

2.3.9 Particle critical velocity

In cold spraying, *Critical Velocity* is the lowest impact velocity for a particle of a specific material to be deposited. In CGDS, the critical velocities of most metals and alloys were reported to be in range $500 - 700\,m/s$ [18]. However, preheating the particles leads to an increase ductility of the particle, and so decreases the critical velocity required for deposition.

According to Schmidt et al. [27] the critical velocity could be calculated using the formula:

$$v_{crit}^{th,mech} = \sqrt{\frac{F_1.4.\sigma_{TS}.(1 - \frac{T_i - T_R}{T_m - T_R})}{\rho} + F_2.c_p.(T_m - T_i)} \tag{2.34}$$

where $v_{crit}^{th,mech}$ is the critical velocity with mechanical and the thermal calibration factors, F_1 is the mechanical calibration factor, F_2 is the thermal

calibration factor, c_p is the specific heat, T_i is the impact temperature, T_m is the melting temperature, T_R is the reference temperature $293\,K$ and σ_{TS} the tensile strength. The SI is used for calculation. Note that considering kinetic energy and thermal dissipation effects on the impact, mechanical and the thermal calibration factors are used to correlate experimental and calculated results.

Finally, comparing critical velocities obtained by calculations and experimentations, it was found that equation (2.34) is accurate for most materials (Figure 2.6).

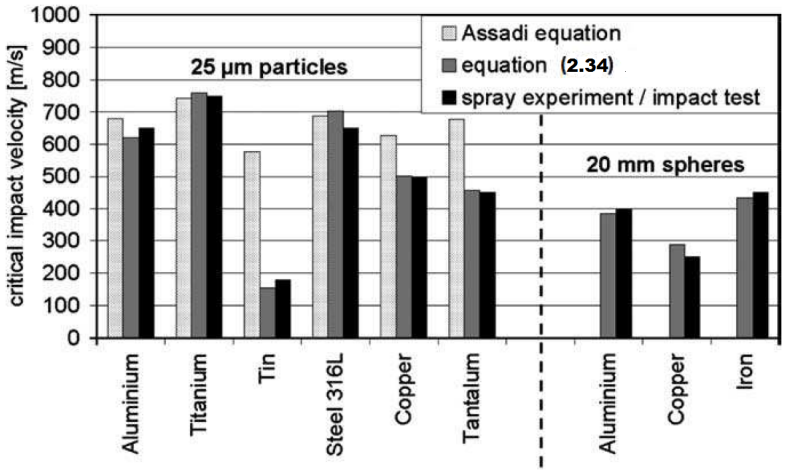

Figure 2.6: Comparison of calculated v_{crit} with experimental results of spray experiments and impact tests [27].

2.4 Gas Dynamics of MOC Nozzle

2.4.1 Theoretical background

The MOC nozzle is the nozzle obtained using *the method of characteristic*. However, in order to understand the process of designing such a nozzle, there is a need for a good understanding of two dimensional gas–dynamics theory and understanding of the flow properties inside the nozzle. The method of

characteristic is applied to a two–dimensional supersonic nozzle to compute the supersonic flow [17] and assuming that the fluid is an inviscid ideal gas, the flow is shock–free and irrotational.

One dimensional flow analysis, in many cases, gives good accuracy for predicting the flow field in the nozzle. However, for real conditions, nozzle flows are never rightfully one-dimensional. As a result, one–dimensional theory is insufficient for the analysis of real nozzle flow. Therefore, neglecting the influence of the wall, the two–dimensional flow model can be used for analyzing the flow in the nozzle. However, the wall boundary layer affects the entire area of nozzle exit. Therefore, as stated by Khine et al. [17] two-dimensional flow analysis is required to simulate the gas flow and to predict the performance characteristic of a two-dimensional nozzle.

2.4.2 Prandlt Meyer function

When a supersonic flow is turned away from itself, an expansion wave is formed and this is a antithesis of shock wave. So, referring to Figure 2.7 and 2.8, Anderson stipulated the flow aspects through an expansion wave as follows [34] (P,v in the text are referred to p and V on the figures):

- $M_2 > M_1$, the expansion corner increases the flow Mach number and the pressure, density and temperature decrease through an expansion wave.
- The expansion region as presented is composed of an infinite number of Mach waves, and each marking the Mach angle μ with the local flow direction; μ_1 for downstream flow and μ_2 upstream flow. Furthermore, because the expansion through the wave takes place across a continuous succession of Mach waves and $ds = 0$ for each Mach wave, it was concluded that the expansion is isentropic.
- The quantitative problem of Prandtl-Meyer expansion wave consists in determination of M_2, P_2 and T_2 for a given M_1, P_1, T_1 and θ_2. The starting point of analysis is considering the infinitesimal changes across a very weak wave (essentially a Mach wave) produced by an infinitesimally small flow deflection, $d\theta$ (Figure 2.8).

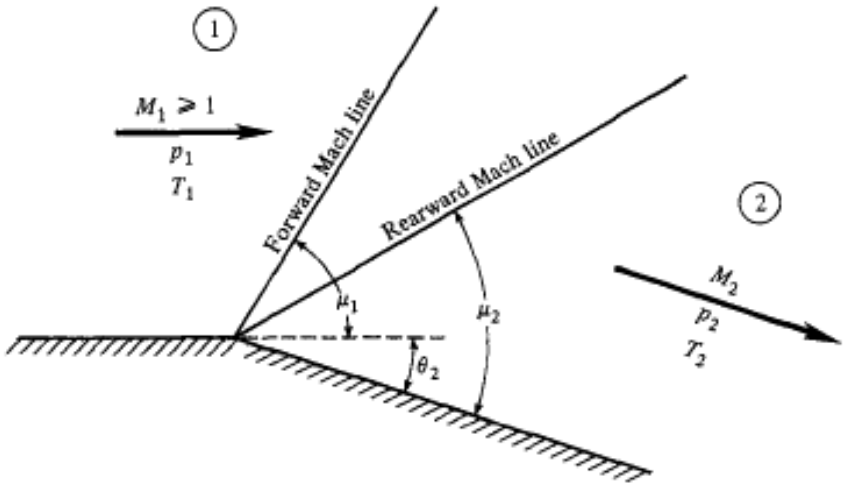

Figure 2.7: Prandtl-Meyer expansion [34].

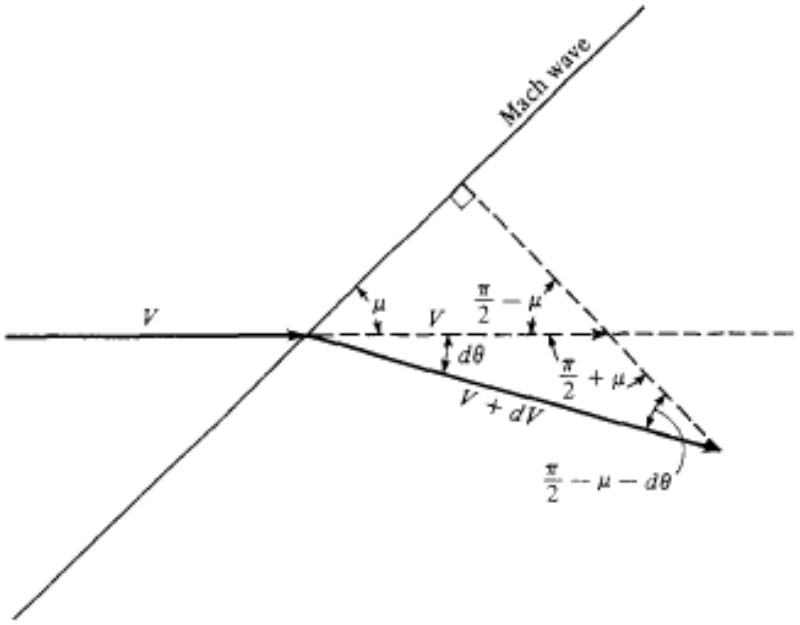

Figure 2.8: Geometric construction for the infinitesimal changes across a Mach wave; for use in the derivation of the Prandtl-Meyer function. Note that the change in velocity across the wave is normal to the wave [34].

After mathematical analysis and trigonometric development, the following equations were obtained.

$$d\theta = \frac{dv/v}{tan\mu} \tag{2.35}$$

$$\mu = sin^{-1}\frac{1}{M} \tag{2.36}$$

$$tan\mu = \frac{1}{\sqrt{M^2-1}} \tag{2.37}$$

Furthermore, considering the equations (2.35) and (2.37), the governing differential equation for Prandtl-Meyer flow is given by the Equation (2.38)

$$d\theta = \sqrt{M^2-1}\frac{dv}{v} \tag{2.38}$$

The resolution of Equation (2.38) leads to *Prandtl–Meyer function*, and represented by symbol ν.

$$\nu(M) = \sqrt{\frac{\gamma+1}{\gamma-1}}tan^{-1}\sqrt{\frac{\gamma-1}{\gamma+1}\left(M^2-1\right)} - tan^{-1}\sqrt{M^2-1} \tag{2.39}$$

The inverse of Prandtl–Meyer function is complicated to find, but the estimation in Equation 2.40 [4] gave good results for most engineering purposes.

$$M = \frac{1+Ay+By^2+Cy^3}{1+Dy+Ey^2} \tag{2.40}$$

where $y = \left(\frac{\nu}{\nu_\infty}\right)^{2/3}$ and $\nu_\infty = \frac{\pi}{2}\left(\sqrt{6}-1\right)$, the maximum turning angle. For Nitrogen with $\gamma = 1.4$, the constants are $A = 1.3604$, $B = 0.0962$, $C = -0.5127$, $D = -0.6722$, $E = -0.3278$.

2.4.3 MOC for the steady of two dimensional supersonic flow

In this section, the numerical Method of Characteristics is investigated and the general procedure is summarized.

2.4.3.1 Principe of numerical method

The principle of numerical method can be summarized as follow:
Consider the calculation of the supersonic, irrotational, incompressible and

stable flow field properties at discrete points in the space as shown in Figure 2.9.

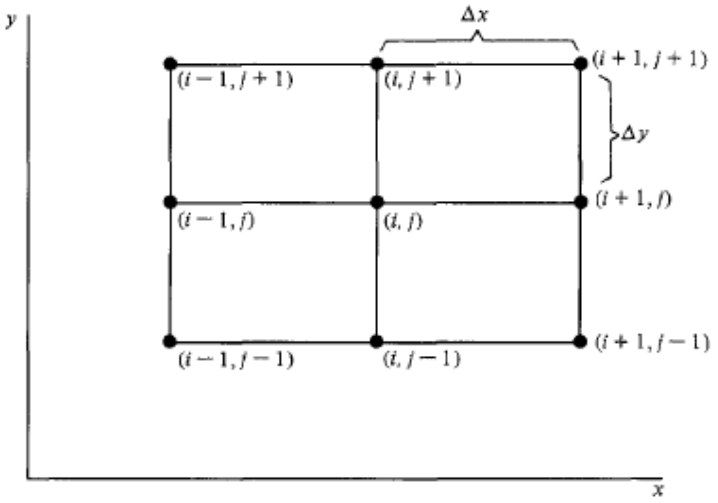

Figure 2.9: Rectangular finite-difference grid [34].

If $v_{i,j}$ is the velocity at the point (i, j) (where i denotes the x component of velocity), then the velocity $v_{i+1,j}$ at point $(i + 1, j)$ can be found using a Taylor's series as follow:

$$v_{i+1,j} = v_{i,j} + \left(\frac{\partial v}{\partial x}\right)_{i,j} \Delta x + \left(\frac{\partial^2 v}{\partial x^2}\right)_{i,j} \frac{(\Delta x)^2}{2} + ... \tag{2.41}$$

An optimum value $(\Delta x)_{opt}$, at which maximum accuracy is obtained considering all the numerical errors, exists.

The second term can be neglected; and in the remaining expression, $\frac{\partial v}{\partial x}$ must be determined to find $v_{i+1,j}$. And considering a vector v in the space, scalar $\Phi(x, y, z)$ can be determined, such that

$$v \equiv \nabla\Phi$$

where ϕ is called the *velocity potential*. Furthermore, irrotational flow means $\nabla \times v = 0$ (∇ is the vector derivative operator).

For a two–dimensional and steady flow, the continuity equation

$$\nabla.(\rho v) = 0 \tag{2.42}$$

after vectorial and derivative mathematical development, becomes

$$\left[1 - \frac{1}{c^2}\left(\frac{\partial\phi}{\partial x}\right)^2\right]\frac{\partial^2\phi}{\partial x^2} + \left[1 - \frac{1}{c^2}\left(\frac{\partial\phi}{\partial y}\right)^2\right]\frac{\partial^2\phi}{\partial y^2} - \frac{2}{c^2}\frac{\partial\phi}{\partial x}\frac{\partial\phi}{\partial y}\frac{\partial^2\phi}{\partial x\partial y} = 0 \tag{2.43}$$

called the *velocity potential equation*, where c is the speed of sound and can be determined as follows:

$$c^2 = {a_0}^2 - \frac{\gamma - 1}{2}\left(\left(\frac{\partial\phi}{\partial x}\right)^2 + \left(\frac{\partial\phi}{\partial y}\right)^2 + \left(\frac{\partial\phi}{\partial z}\right)^2\right) \tag{2.44}$$

a_0 is a known constant of the flow.

The solution to the *velocity potential equation*, can be approached either by *exact numerical solutions*, by *transformation of variables*, or by *linearized solutions*. However, modern CFD numerical techniques allow complicated applications to be solved.

2.4.3.2 General procedure for solving the velocity potential equation

The general procedure to solve the two dimensional velocity potential equation flow using the MOC method, can be summarized in three steps as follows [34]:

1. find the characteristic lines,
2. find the compatibility equations; these are ordinary differential equations along the characteristic lines, that are obtained from a combination of partial differential equation, and
3. solve the compatibility equations step by step along the characteristic lines; a starting point can be where initial condition are given.

A system of three equations, (i) Equation (2.43), (ii) the differential of v_x (dv_x) and (iii) the differential of v_y (dv_y) is formed, and using Cramer's rule, the solution of $\frac{\partial^2\phi}{\partial x\partial y}$ can be found as follows:

$$\frac{\partial^2\phi}{\partial x\partial y} = \frac{\begin{vmatrix} 1-\frac{v_x^2}{c^2} & 0 & 1-\frac{v_y^2}{c^2} \\ dx & dv_x & 0 \\ 0 & dv_y & dy \end{vmatrix}}{\begin{vmatrix} 1-\frac{v_x^2}{c^2} & \frac{-2v_xv_y}{c^2} & 1-\frac{v_y^2}{c^2} \\ dx & dy & 0 \\ 0 & dx & dy \end{vmatrix}} = \frac{N}{D} \tag{2.45}$$

where N is the numerator and D the denominator.

Considering the relation (2.45), when $D = 0$ characteristic lines can be developed after algebraic trigonometric manipulation; in fact *the Mach line* is the line that makes a *Mach angle* with respect to the streamline direction at a given point. This line is also the line along which the derivative of v_x is indeterminate and across which can be discontinuous. Moreover the derivatives of the other flow variables, such as P, ρ, T, v_y, etc., are also indeterminate along this line. So, Anderson determined the slope of the characteristic lines as follows:

$$\left(\frac{dy}{dx}\right)_{char} = tan\,(\theta \pm \ \mu) \tag{2.46}$$

where $\left(\frac{dy}{dx}\right)_{char}$ is the slope. Figure 2.10 gives a graphical interpretation of this equation. Equation (2.46) shows that a characteristic line called C_+ at point

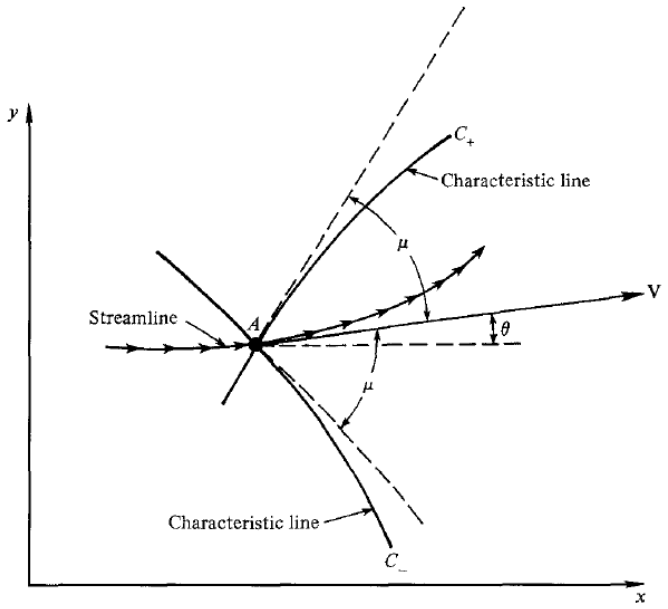

Figure 2.10: Illustration of left-and right-running characteristic lines [34].

A is inclined above the streamline direction by the angle μ. Furthermore, the characteristic lines through point A are the left and right-running Mach waves through point A. As seen, the characteristic lines are Mach lines. The left-running Mach wave is called C_-.

Furthermore, from the relation (2.45), when $N = 0$ compatibility equations[i] can be simplified to:

$$d\theta = \pm\sqrt{M^2 - 1}\frac{dv}{v} \tag{2.47}$$

After integration and considering Prandtl-Meyer flow, the Equation (2.47) can be developed to form:

$$\theta + \nu(M) = constant = K_- \tag{2.48}$$

and

$$\theta - \nu(M) = constant = K_+ \tag{2.49}$$

with K_- along the C_- characteristic and K_+ along the C_+ characteristic respectively.

Finally, *the unit process* is a series of specific computations to solve compatibility equations point by point along the characteristic lines. These points can be internal to the flowfield or on the free boundary. The process is simplified as follows [34]:

Considering the internal steady flow (Figure 2.11), the knowledge of flowfield conditions of two points ('1' and '2') can help to determine conditions at the third point ('3') located by intersection of characteristic lines passing by the two points.

Consider θ_i, ν_i, μ_i, $(K_-)_i$ and $(K_+)_i$ flowfield conditions related to point i.

From equations (2.48) and (2.49), it can be stated that

$$\theta_1 + \nu_1 = (K_-)_1$$

$$\theta_2 - \nu_2 = (K_+)_2$$

$$\theta_3 + \nu_3 = (K_-)_3 = (K_-)_1$$

$$\theta_3 - \nu_3 = (K_+)_3 = (K_+)_2$$

[i]Note that the compatibility equations are the equation that describes the variation of flow properties along the characteristic lines.

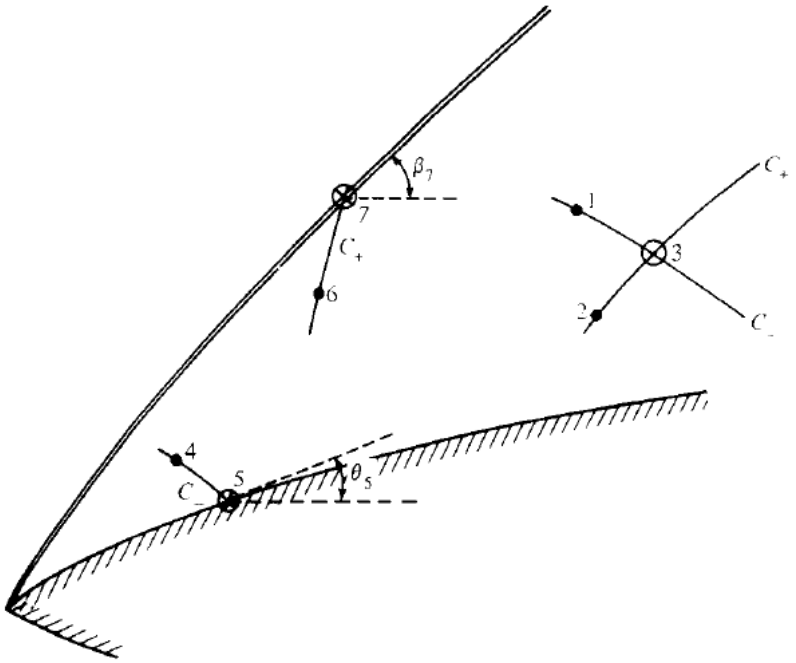

Figure 2.11: Unit processes for the steady-flow, two-dimensional irrotational method of characteristic [34].

Solving the last two equations, θ_3 and ν_3 are expressed as:

$$\theta_3 = \frac{1}{2}\left[(K_-)_1 + (K_+)_2\right]$$

$$\nu_3 = \frac{1}{2}\left[(K_-)_1 - (K_+)_2\right]$$

So, the flow conditions at point 3 are determined, and knowing θ_3 and ν_3, all other flow properties can be determine as follows:

1. Knowing ν_3, use the Prandtl-Meyer function to obtain the associate M_3
2. Knowing M_3 and the initial conditions of pressure and temperature, determine P_3 and T_3
3. Knowing T_3, the speed of sound can be computed: $c_3 = \sqrt{\gamma R T_3}$. And, $v_3 = M_3 c_3$.

To determine the exact location of point 3, an approximate but sufficiently accurate procedure is used. This involves the determination of the slopes of C_- and C_+, and assuming that characteristic the lines are straight-line segments

between the grid points.

Thus the slope of C_- can be computed as

$$\left[\frac{1}{2}(\theta_1 + \theta_3) - \frac{1}{2}(\mu_1 + \mu_3)\right]$$

and the slope of C_+ can be computed as

$$\left[\frac{1}{2}(\theta_2 + \theta_3) + \frac{1}{2}(\mu_2 + \mu_3)\right]$$

The result is illustrated in Figure 2.12.

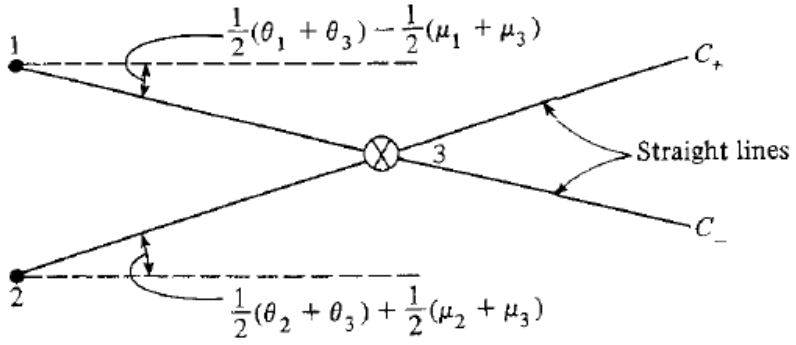

Figure 2.12: Approximation of characteristics by straight line [34].

If the conditions at a point near the wall are known and using Figure 2.11 (with point 4 near the wall and point 5 on the wall) the flow variables at the wall can be determined as follows:

$$(K_-)_4 = \theta_4 + \nu_4$$

and considering that the point 4 and the point 5 are on the same line,

$$(K_-)_4 = (K_-)_5 = \theta_5 + \nu_5$$

As the shape is known, the flow is tangent to the wall, and consequently, θ_5 is known. Thus ν_5 can be determined by:

$$\nu_5 = \nu_4 + \theta_4 - \theta_5$$

Finally, for this study, the starting point for nozzle calculation is taken on the sonic line that is assumed to be a straight line.

2.4.3.3 Minimum length nozzle

The present study focuses on the *'minimum-length' nozzle* (Figure 2.13) in which the expansion section is shrunk to a point, and thereafter, the expansion takes place through a centered Prandtl-Meyer wave emanating from a sharp-corner throat with an angle θ called *wall maximum*.

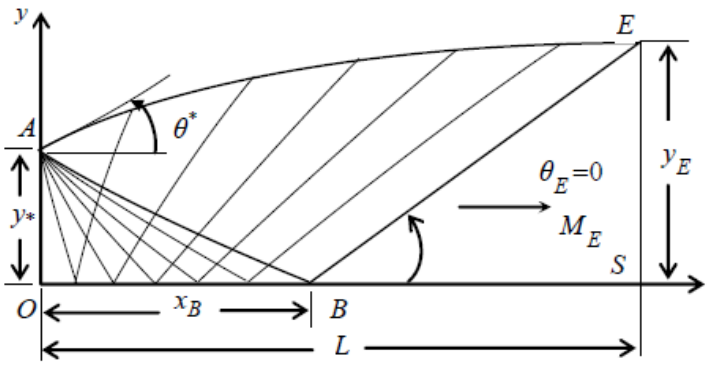

Figure 2.13: Flow field presentation in Minimum Length Nozzle [31]

.

The regions in Figure 2.13 can be broken into 3 regions:

1. region of Kernel (Area OAB) – this is a non–simple (crossed by 2 types of line) waves region,
2. transition Region (Area ABE) – this is simple (crossed by 1 type of line) waves regions, and
3. area BSE – in this region the flow is uniform and the Mach number is M_E.

Finally, the equations of gas motion assuming the 'minimum length' nozzle can be solved graphically and step by step.

2.5 Optimization of the CGDS Process by Improving Nozzle Design

There have been many efforts in the direction of improving the quality of the CGDS deposition process. However, in this respect, the development of the nozzle design has offered the best results [20], where especially the method of characteristics (MOC), was used to develop new nozzle designs that provided a significantly more homogeneous particle acceleration than that of the standard nozzle [11].

Figure 2.14 illustrates the comparison between the flow fields of the free gas jets of a *standard* type nozzle and one designed using the MOC method. Furthermore, Figure 2.15 shows the impact velocities of a $20\,\mu m$ copper particle as function of gas inlet temperature for the trumpet–shaped standard nozzle and the bell-shaped MOC–designed nozzle, using Nitrogen process gas and a pressure of $30\,bar$. The arrows indicate the increase of particle velocity when using a MOC nozzle.

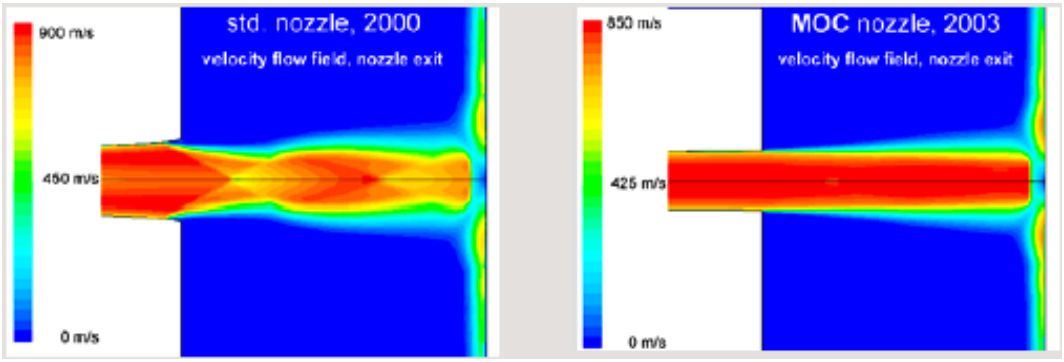

Figure 2.14: Comparison between the gas jets generated by a standard and a MOC nozzle [19].

Finally, as Francois [8] indicated, the rate of deposition in CGDS process were better for MOC nozzles compared to other types of nozzles (Figure 2.16).

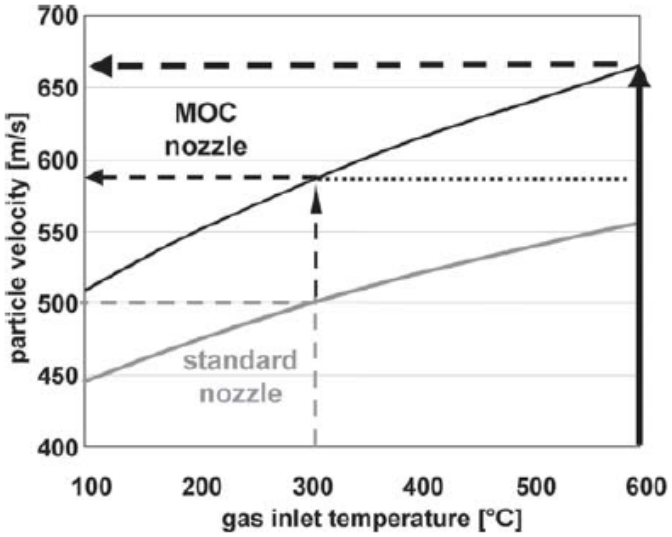

Figure 2.15: Comparison between particle velocities of a standard and MOC nozzle [11].

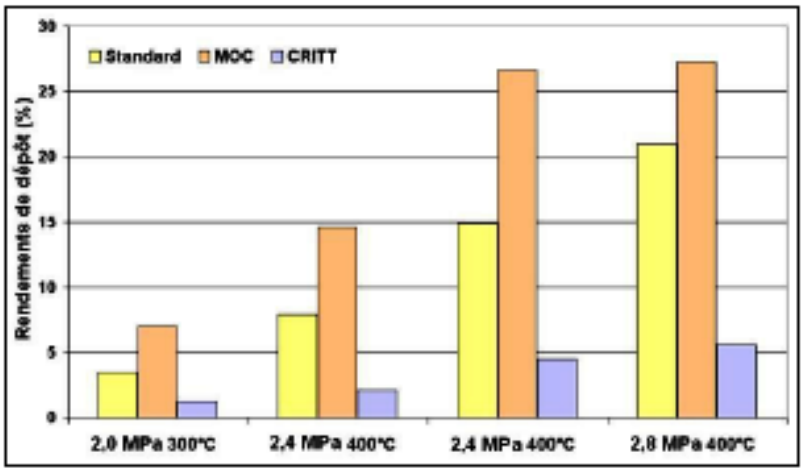

Figure 2.16: Performance evolution of the deposition of nickel according to nozzle used [8].

2.6 Flow Analysis using ANSYS Fluent Software

In order to optimize the cold spray parameters, Tabbara et al.[25] adopted the Computational Fluid Dynamics (CFD) technique to examine the effects

of changing the nozzle cross–section shape, the particle size, and process gas type on the gas flow characteristics through the nozzle. Also, they used the CFD technique to assess the powder particle velocity at the nozzle exit, assess the spray distribution, and to compare all the CFD results with the practical experiments. In addition, the CFD was used to model the turbulence and the multi-phase flows [22].

Furthermore, ANSYS Fluent is a CFD software which operates after the flow field has been divided into a few hundred thousand finite volume cells. Then, the flow is evaluated using the Navier-Stokes equations and other scalar equations for each cell and taking into account the flow heat conduction, the turbulence, and the frictional losses [29]. The advantages of using CFD computational method are the detailed information obtained about the gas temperature and velocity fields, and the details about the trajectories, temperatures, and velocities of the particle throughout the nozzle and the free jets [29]. Consequently, it was concluded that CFD Software could become an important tool for the CGDS research [11].

2.7 Conclusion

This chapter reviewed the relevant literatures related to the calculation and design of supersonic nozzles for CGDS using Matlab and Ansys Fluent. Also, the gas dynamics theories involved in the De Laval nozzle design and MOC nozzle design have been analyzed, and so, it was provided the background knowledge for the calculation and design of the nozzles that will be carried out Chapter 4.

3 METHODOLOGY

3.1 Introduction

Chapter 2 reviewed the relevant literature related to the calculation and design of supersonic nozzles for CGDS process. This chapter describes the methodologies used to answer the main research question presented in chapter 1. Chapter 3 is also the starting point of the development of a new software using MATLAB high–level language.

3.2 Parameters for the De Laval nozzle simulation

De Laval nozzle is the typical nozzle used in the CGDS process. Therefore, it is critically important to know the performance of a specific De Laval nozzle that uses specific input parameters.

Consequently, this dissertation will develop a new GUI software in MATLAB that will have the possibility to take as input the critical CGDS parameters and compare the powder particle speed v_p achieved by a specific nozzle with the critical speed required by the specific powder material in order to be deposited.

Thus, in order to have a clear methodology to follow the development of the new software, the flowchart presented in Figure 3.1 was developed. The steps calculation are as follows:

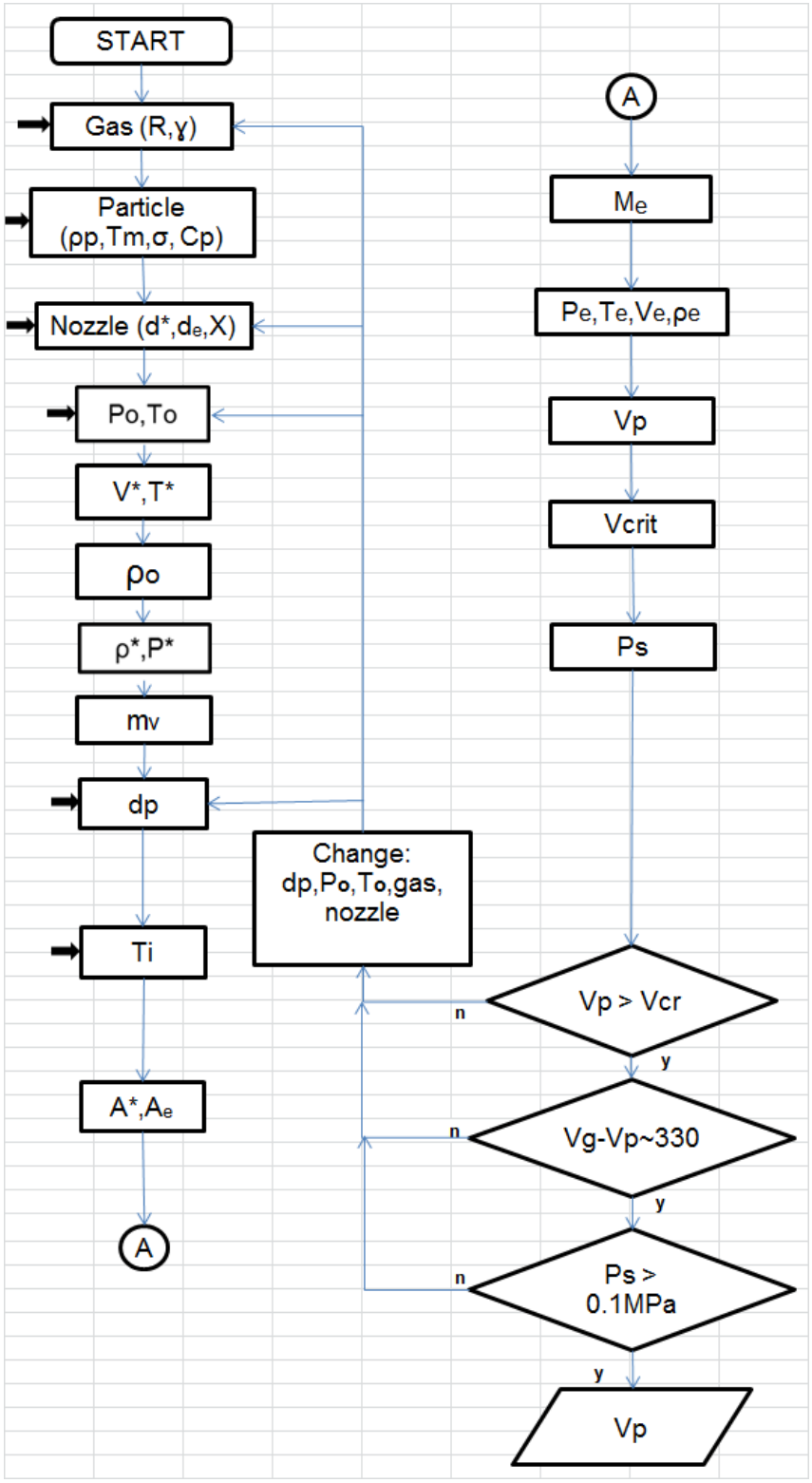

Figure 3.1: Flowchart for De Laval Nozzle calculations.

Step 1: select the working gas; the gas constant R and the gas specific heat ratio γ should be automatically provided by the program.

Step 2: select the material of particle by to be deposed; material properties such as tensile strength σ, density of material particle ρ_p, specific heat c_p, melting temperature T_m, mechanical calibration factor, and thermal calibration factor should be automatically provided by the program.

Step 3: enter data for the nozzle d^*, d_e and x.

Step 4: enter the stagnation temperature T_o and the stagnation pressure P_o.

Step 5: compute gas throat temperature T^*, and gas throat velocity v^*.

Step 6: compute stagnation density ρ_o.

Step 7: compute gas throat density ρ^*, then determine the throat pressure P^*.

Step 8: compute the gas flow rate $\dot{m}_v$.

Step 9: enter the powder particle diameter d_p.

Step 10: enter the impact temperature T_i.

Step 11: compute nozzle's section A^* and nozzle's section A_e.

Step 12: compute the Mach number of the gas at the nozzle exit M_e.

Step 13: compute the exit pressure of the gas P_e, then determine gas exit

temperature T_e, the exit gas velocity v_e, and finally gas exit density ρ_e.

Step 14: compute the particle velocity v_p.

Step 15: compute the critical velocity v_{crit}.

Step 16: compute the shock pressure P_s.

Step 17: verify if v_p is greater than v_{crit}; if v_p is not greater then v_{crit}, the user have to *increase/decrease* one of the input parameters; for cost considerations recommended order to change the input parameters is P_o, T_o, d_p, gas, and finally the nozzle; if v_p is greater then v_{crit}, then go to the next stage.

Step 18: verify if the difference (v_e - v_p) equals about the *speed of sound*[i].

Step 19: verify that a shock wave is not present inside the nozzle; if $P_s < P_a$ the user must *increase/decrease* one or more then one of the input parameters as in the previous step; if $P_s > P_a$, the calculated v_p could be considered as a value that meets the CGDS deposition requirements.

3.3 MOC nozzle design

As presented in chapter 2, the MOC nozzle is obtained using the method of characteristic that is applied to a two–dimensional supersonic nozzle. A MOC nozzle will provide or increased particle speed for the same input of CGDS parameters. Also, the minimum length of the nozzle's internal curved profile

[i]Calculations have shown that a relative velocity between the gas and the particle for Mach number equals to the square of 2, allows to be achieved a density and velocity that maximizes the acceleration of the particles. Experiments have shown that the gas–particle relative velocity must be maintained at about the speed of sound and that this value corresponds to a Mach number equal to 1[8].

is followed by a straight barrel section where the speed of the particle is increased due to larger contact time between the gas and particles. As a result, the quality of the CGDS deposition will increase.

Consequently, this dissertation will develop a new GUI software using MATLAB that will have the possibility to take as input a planned Mach number, the gas specific heat, the nozzle's throat diameter, and plot the internal profile of the MOC nozzle. Also, in order to improve the flow of the gas, the software will verify the shockwave at the exit of the nozzle.

Thus, in order to have a clear methodology to follow for the development of a new software, the flowchart in Figure 3.2 was developed.

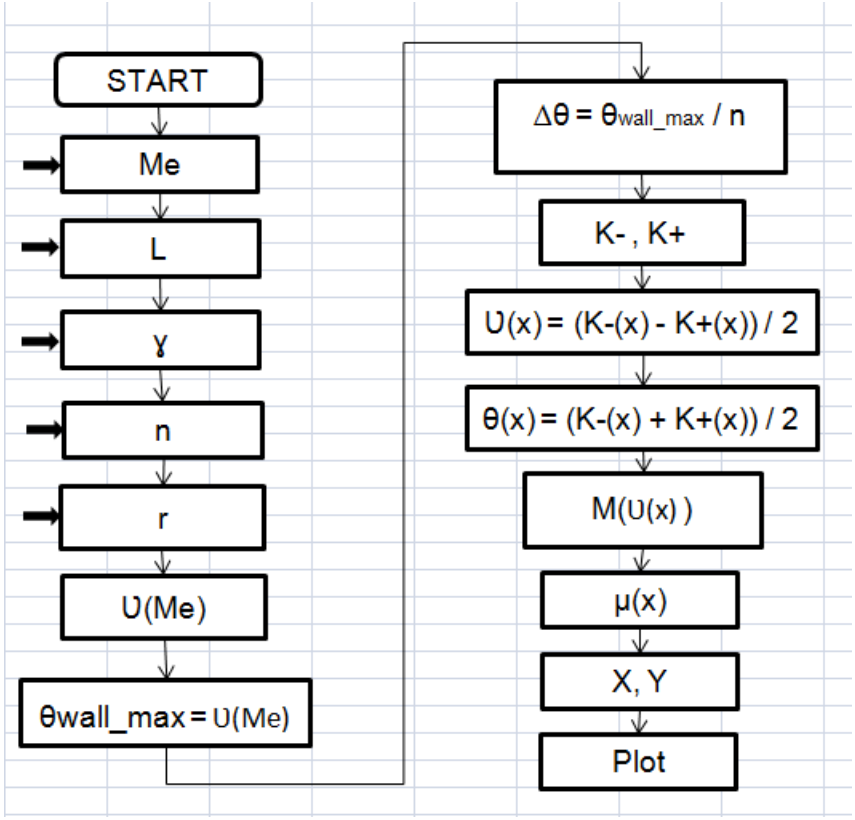

Figure 3.2: Flowchart for MOC cozzle calculations.

The calculations steps are as follows:

Step 1: input the Mach number needed at the nozzle exit.

Step 2: input the length of divergent part of the nozzle.

Step 3: input the ratio of specific heat.

Step 4: input the number of characteristic lines.

Step 5: input the radius of the nozzle's throat.

Step 6: calculate of the Prandtl Meyer function for Mach number given in the first step using the inverse of Prandtl Meyer function.

Step 7: calculate the 'max angle' of the duct wall $\theta_{wall\,max}$ with respect to the *x direction*. Note that the x direction represents the flow direction.
The *total corner angle* $\theta_{wall\,max}$ at the throat can be determined as followed

$$\theta_{wall\,max} = \frac{\nu(M)}{2} \tag{3.1}$$

Note that $\nu(M)$ is the Prandtl-Meyer function corresponding to the designed exit Mach number. The expansion fan is replaced by the finite number of right running characteristics starting from point 1, in such a way that the flow, as it crosses these n characteristic lines, turns from 0 to $\theta_{wall\,max}$.

Step 8: calculate $\Delta\theta$.
Each characteristic line turns the flow direction by

$$\Delta\theta = \left(\frac{\theta_{wall\,max}}{n}\right) \tag{3.2}$$

Step 9: calculate the Prandtl Meyer constants.
As the starting point for the gas is at sonic conditions, each right running characteristic line has a $'\nu'$ value equal the value of θ. Then K_+ and K_- are computed.

Step 10: calculate of Prandtl Meyer angles ν on a oblique line x using the Prandtl Meyer constants.

Step 11: calculate the angle of duct wall θ on a oblique line x using the Prandtl Meyer constants.

Step 12: calculate the Mach number M as a function of Prandtl Meyer angles on a oblique line x.

Step 13: calculate the local Mach angle μ on a oblique line x.

Step 14: determine, using the geometric principle of intersection of two straight lines, the coordinates of the points on the contour with respect to the x axis and y axis. The points will be determined by intersection of the shape-line and the oblique-line. See Figure 3.3 for clarification.

Step 15: plot the points and connect them.

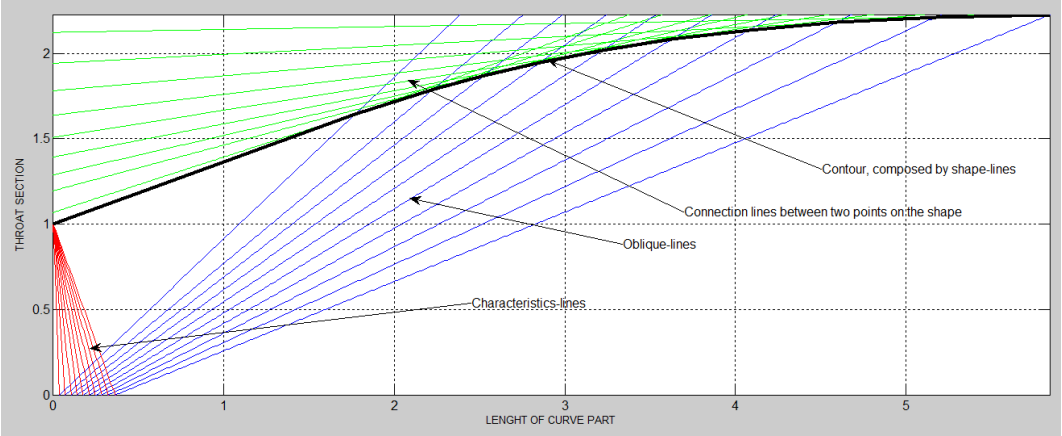

Figure 3.3: Lines details.

3.4 Test diagram

The test diagram is presented in Figure 3.4

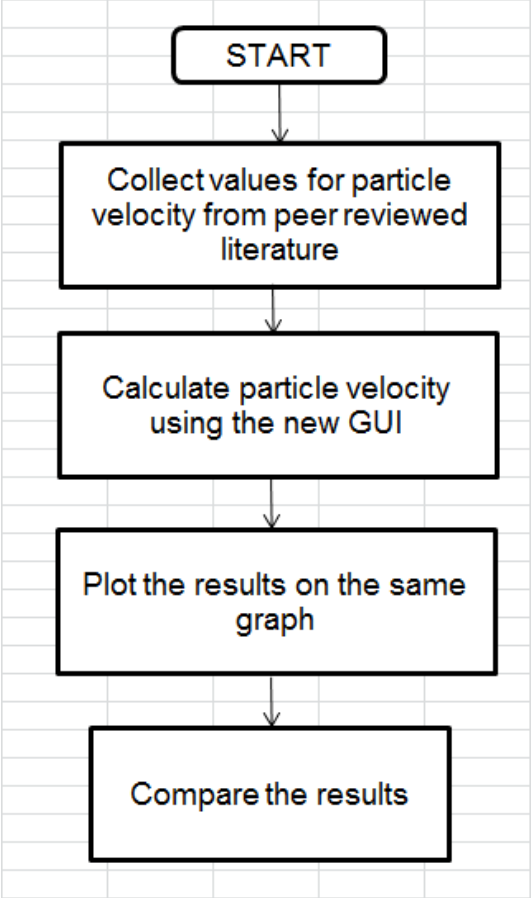

Figure 3.4: Test diagram.

3.5 Summary

Chapter 3 focused on the methodologies that need to be followed for the development of the new software using MATLAB. These methodologies included all the necessary steps for the calculation and design of the De Laval and the MOC nozzles, and also the necessary steps for testing the results. In addition, chapter 3 represented the starting point in the development of the new MATLAB software. Consequently, the next chapter will focus on the new GUI construction and its testing.

4 SOFTWARE DEVELOPMENT AND ANALYSIS OF DATA

4.1 Introduction

This section presents the software development for the calculation and design of the internal profile of the nozzle. Then, the new software will be tested and the results compared with the data found in the published literature.

The new software is developed in MATLAB. The De Laval nozzle one dimensional approach calculations are compared with those achieved using the ANSYS Fluent software. Furthermore, the results for the two dimensional approach used for the MOC nozzle design that is also developed in MATLAB will be compared with the CFD Ansys Fluent results.

4.2 GUI Development

MATLAB GUI development is very important because it contains all the input values of the user, all important calculated results, and all the resulted designs of the internal profile of the nozzle.

The GUI was developed using Graphic User Interface DEveloper (GUIDE) in MATLAB [39], and each component included in GUIDE was connected with one or more user defined routines known as callbacks. When a user pushes a

button or selects a menu item, the execution of a specific callback developed by the author of this report is performed. Also, using a tool called 'Property Inspector', each component in GUIDE is identified by a tag (name) and by a set of characteristics set by the programmer (Figure 4.1).

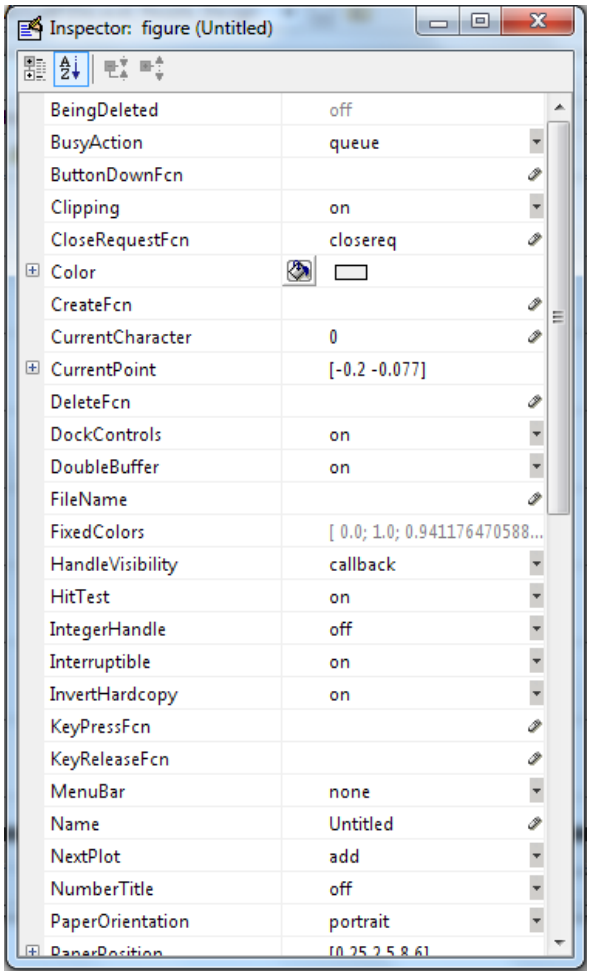

Figure 4.1: Property Inspector.

When the Editor is saved, two files with the same name but different extensions, are automatically created. These two files are: **'.fig'**, used to enter the inputs into the program, and **'.m'** file used to call the callback structure. An example of a 'm-file' is shown in Figure 4.2.

Also, Figure 4.3 shows the main interface of the developed GUI with its two button: 'Simulation Parameters *De Laval nozzle*' and 'Contour Design *MOC nozzle*'.

Finally, when the GUI was developed a number of other software literature

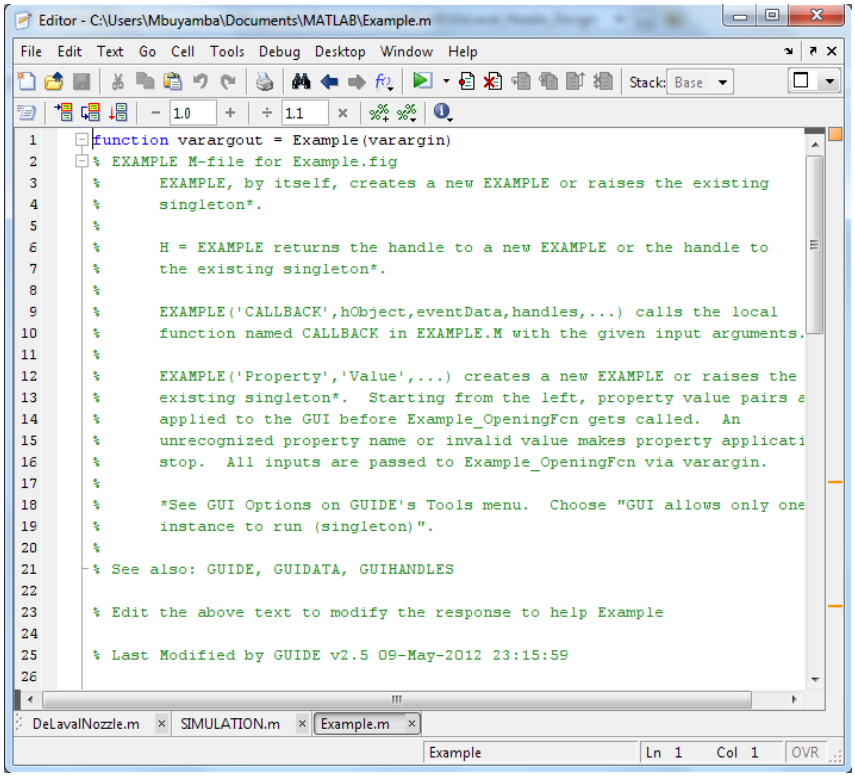

Figure 4.2: Example of M-file for GUI.

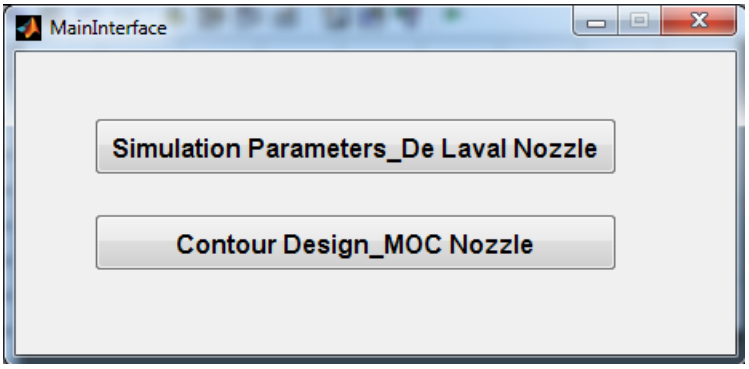

Figure 4.3: Main interface.

recommendations were applied. These include aspects such as: the reason for creating a GUI, the consideration related to the user (his mental capacity), a simple *user–friendly interface* [32], and the possibility to easily add new functionalities to the software.

4.3 Calculation Process and Design for De Laval Nozzle

4.3.1 Calculation Process

Chapter 3 presented a flowchart for the calculation of the De Laval nozzle. The present section uses the flowchart algorithm and practically demonstrates its use. Consequently, the reader could find below a numerical example based on the experimental work given in Stoltenhoff et al. [29].

Conditions of the experiment

• *Working gas*
– `Nitrogen`
– `Gas constant` $R\ (J/kg\,K) = 296.8$
– `Specific heat ratio` $\gamma = 1.4$

• *Stagnation conditions*
– `Stagnation temperature` $T_o\,(K) = 593$
– `Stagnation pressure` $P_o\,(MPa) = 2.5$

• *Nozzle geometry*
– `Throat diameter` $d^*\ (mm) = 2.7$
– `Exit diameter` $d_e\ (mm) = 8.1$
– `Divergent length nozzle` $x\ (mm) = 90$

• *Powder Particles*
– `Copper`
– `Diameter of particle` $d_p\ (\mu m) = 15$
– `Tensile strength` $\sigma_{TS}\ (MPa) = 210$
– `Particle material density` $\rho_p\ (kg/m^3) = 7870$
– `Particle heat capacity` $c_p\ (J/kg\,K) = 390$

– `Melting temperature` $T_m\ (K) = 1535$
– `Mechanical calibration factor` $F_1 = 1.2$
– `Thermal calibration factor` $F_2 = 0.3$

Calculated Data

- *Throat Temperature*

$$T^* = \frac{593}{1 + \frac{1.4-1}{2}} = 494.3\,K$$

- *Throat Velocity*

$$v^* = \sqrt{1.4 * 296.8 * 494.3} = 453.2\,m/s$$

- *Stagnation Density*

$$\rho_o = \frac{2.5 * 10^6}{296.8 * 593} = 14.2\,kg/m^3$$

The density and pressure at the throat can be determined as follows:

$$\rho^* = 14.2 * \left(\frac{2}{1.4+1}\right)^{\frac{1}{1.4-1}} = 9\,kg/m^3$$

$$P^* = 9 * 296.8 * 494.3 = 1.32\,MPa$$

The gas flow rate can be determines as follows:

$$\dot{m}_v = \left(\frac{2}{\gamma+1}\right)^{\frac{1}{\gamma-1}} * V^* * A^* * 3600$$

$$\dot{m}_v = \left(\frac{2}{1.4+1}\right)^{\frac{1}{1.4-1}} * 453.2 * \left(\frac{3.14 * 0.0027^2}{4}\right) * 3600 = 6\,m^3/hour$$

- *Mach number*

The Mach number at the exit of the nozzle can be calculated using the constants $k1$ (Equation 2.25) and $k2$ (Equation 2.26) as follows:

$$k1 = 218.0629 - 243.5764 * 1.4 + 71.7925 * 1.4^2 = 17.77$$

$$k2 = -0.122450 + 0.281300 * 1.4 = 0.27137$$

$$M_e = (17.77 * 9 + (1 - 17.77))^{0.27137} = 3.8461$$

- *Exit Pressure*

$$P_e = 1.32 * \left(\frac{1.4+1}{2+(1.4-1)*3.8461^2} \right)^{\left(\frac{1.4}{1.4-1}\right)} = 0.02025\,MPa$$

- *Exit Temperature*

$$T_e = \frac{593}{1+\frac{1.4-1}{2}*3.8461^2} = 149.804\,K$$

- *Exit Gas Velocity*

$$v_e = 3.8461 * \sqrt{1.4*296.8*149.804} = 959.57\,m/s$$

- *Exit Gas Density*

$$\rho_e = \frac{9*\left(\frac{1.4+1}{2}\right)^{\frac{1}{1.4-1}}}{\left(1+\frac{1.4-1}{2}*3.8461^2\right)^{\frac{1}{1.4-1}}} = 0.455\,kg/m^3$$

- *Particle Velocity*

$$v_p = \frac{959.57}{1+0.85*\sqrt{\frac{15*10^{-6}}{90*10^{-3}}}*\sqrt{\frac{7870*959.57^2}{2.5*10^6}}} = 603.2\,m/s$$

- *Critical Velocity*

$$v_{crit}^{th,mech} = \sqrt{\frac{1.2*4*210*10^6*\left(1-\frac{293.15-293.15}{1535-293.15}\right)}{7870} + 0.3*390*(1535-293.15)}$$
$$= 522.855\,m/s$$

Shock Pressure

$$P_s = 0.02025 * \left(\frac{2*1.4}{1.4+1}*3.8461^2 - \frac{1.4-1}{1.4+1} \right) = 0.346\,MPa$$

Following the verification algorithm presented above, the GUI for the new software was possible to be developed.

4.3.2 Introduction to the De Laval nozzle GUI

The developed GUI simulation window is presented in Figure 4.4 and its associated code could be found in Appendix C1.

After the user clicks on the button 'Simulation Parameters De Laval Nozzle' in the *MainInterface* (Figure 4.3), the MATLAB file 'DeLavalNozzle.m' is opened. Then, by selecting the '*Run*' button, the *Interface* shown in Figure 4.4 is displayed and the following areas could be identified:

'Specifications' button
For a better understanding of the use of GUI, the user should click the '*Specifications*' button that contains the following data:

- The 'm-files' for Helium.m and Nitrogen.m gases; and Copper.m, Aluminum.m, Nickel.m, Titanium.m and Steel316L.m files for powder particles. Note: if required, additional 'm-files' could be added using the 'MainDeLavalNozzleSimulation' file and the *Propriety Inspector* window.
- The mass flow to be used that varies from $0\,m^3/hour$ to $100\,m^3/hour$. For a stagnation temperature of $273.15\,K$, this limits the maximum pressure for Helium at 1.5 MPa and at 4.6 MPa for Nitrogen. Note: the limitations of the actual CGDS system should be considered.
- All input data must be entered using the units shown on the GUI *interface*.

'Compute' button – this button will start the simulation.

'Exit' button – this button is used to exit the interface.

Input Nozzle – this section contains the geometrical parameters of the nozzle. These are:
– throat diameter: `Throat Dia., mm`
– exit diameter: `Exit Dia., mm`
– length of divergent part: `x, mm`
– area ratio: `AreaRatio` (as calculated)

Input Gas – this section contains the carrier-gas parameters. These are:
– type of gas: `Select gas` (selected from the popup menu)

– stagnation temperature: `Input Temperature, K`
– stagnation pressure: `Input Pressure, MPa`

Input Particle – this section contains input data for the particle. These are:
– the particle material: `Select Particle` (selected from the popup menu)
– the particle diameter: `Particle Diameter, micron`
– the impact temperature: `Impact Temp.,(293.15K)`
– the impact temperature: `Impact Temp.,(373.15K)`

Output gas properties – this section contains the calculated conditions at the nozzle's throat and at the nozzle's exit. These are:
– the pressure: `Pressure,MPa`
– the temperature: `Temperature,K`
– the density: `Density,` $kg/m3$
– the gas velocity: `Gas Velocity,m/s`

Output Velocities – this section contains the calculated velocities and the gas exit Mach number. These are:
– the Mach number: `Mach number`
– the critical velocity: `Critical Velocity,m/s` (2 values)
– the particle velocity: `Particle Velocity,m/s`

Shock Pressure – this section contains the calculated shock pressure: `Shock Pressure,MPa`
Atm Pressure, MPa – this is the atmospheric pressure in `MPa`
Atm Temperature, K – this is the atmospheric temperature in `K`
Gas Flow Rate, $m^3/hour$ – this is the gas flow rate in $m^3/hour$

Plots – this section displays two plots:
– the 'Variation of *Particle Velocity*' in m/s function of *distance* x in mm from the nozzle's throat, and
– the variation of the *'Temperature'* in $Kelvin$ and the *'Pressure'* in MPa function of gas *Mach number*.

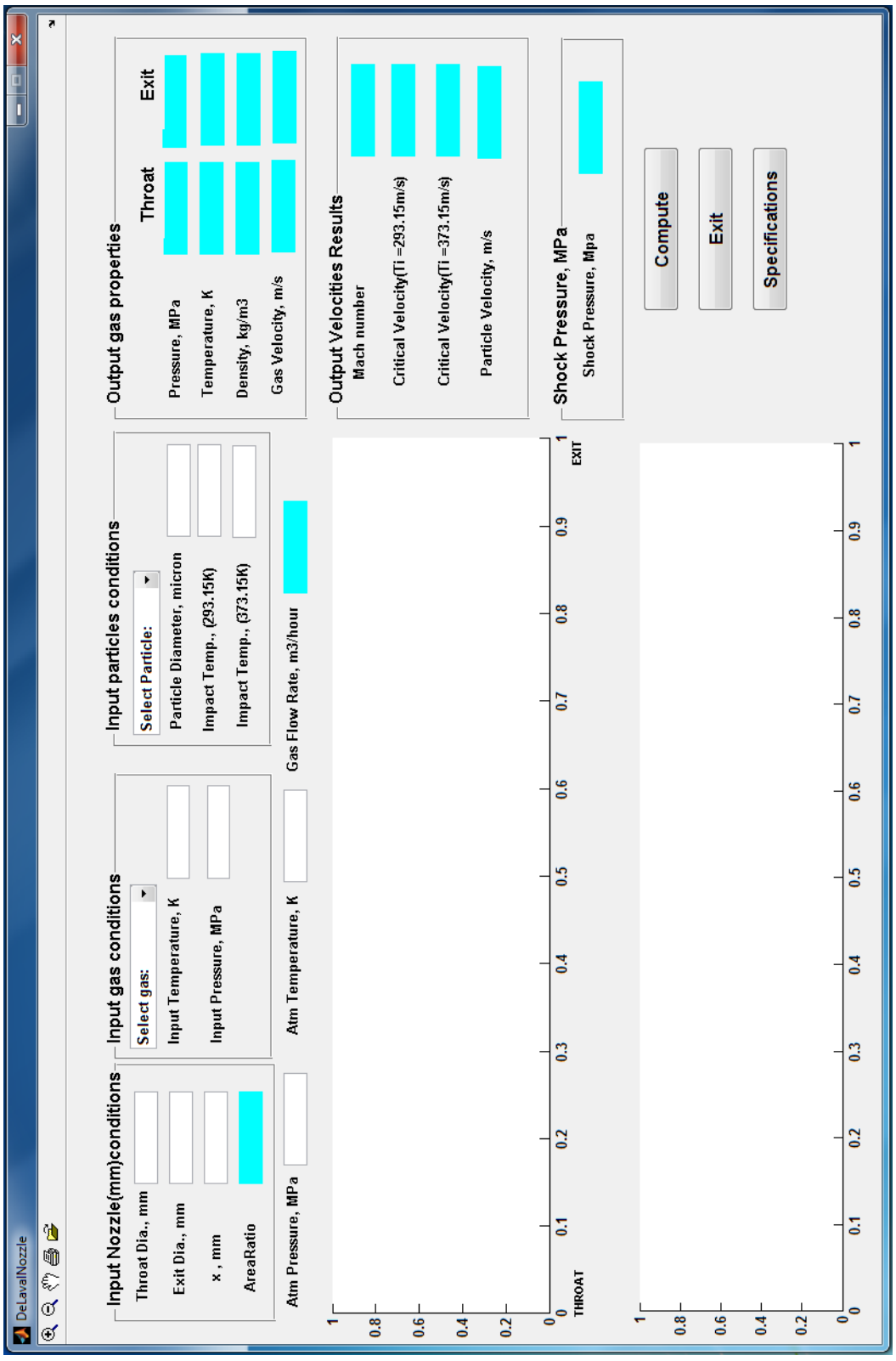

Figure 4.4: Main window of the tool used to simulate the De Laval nozzle.

To test the new GUI Software, the same parameters used in section *'Calculation Process'* are entered and the same result are achieved (Figure 4.5).

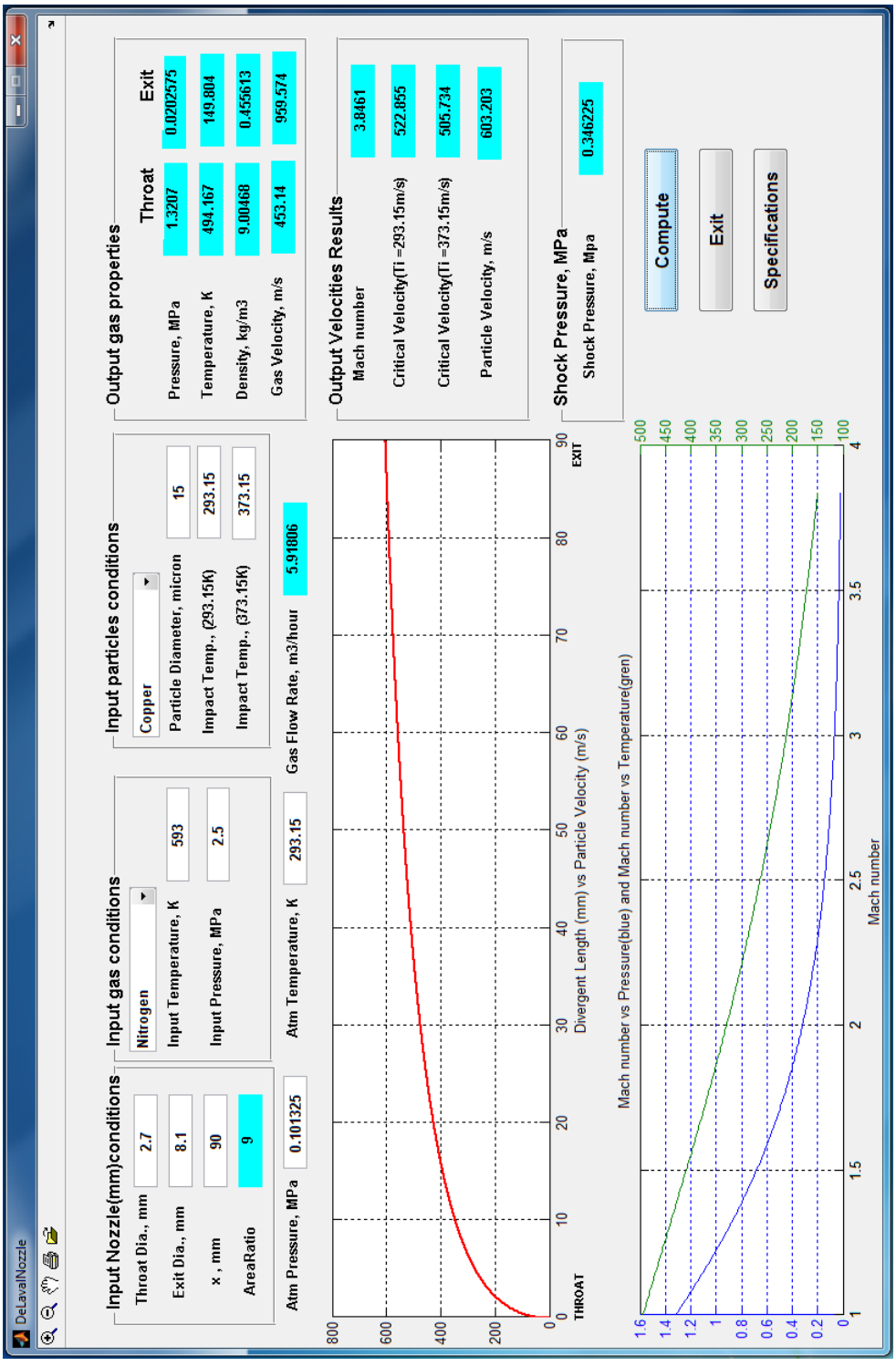

Figure 4.5: Results for Cu particles using N_2, $d_p = 15\,\mu m$, $T_o = 593\,K$ and $P_o = 2.5$ MPa, Area ratio = 9,$d^* = 2.7\,mm$.

4.3.3 Program testing and results analysis

Previews research have demonstrated that the CFD results are quite accurate when compared with the experimental results. Therefore, in order to test the MATLAB and GUI calculations, a number of tests were conducted. Furthermore, the calculated results were compared with data from the published literature. MATLAB calculations were also compared with those obtained using CFD simulation. Finally, for an easy analysis, all the results were presented in a table and statistical errors for each case were performed using the *'Variance of Interpolation Error'* and the *'Maximum Difference'*.

4.3.3.1 Test 1

This test was conducted using data from Li and Li [21]. The characteristics of the De Laval nozzle are as follows: *throat diameter* equals $2\,mm$; *exit diameter* equals $5\,mm$; and the *divergent length* equals $40\,mm$. Other parameters include: Nitrogen at a pressure of 2 MPa; Copper particles d_p of $5, 10, 20, 30, 40$ and $50\,\mu\,m$; and gas temperatures of $300.15, 423.15, 573.15, 723.15$ and $873.15\,K$.

Then, using the same operating parameters, the results from Li and Li [21] were compared with those obtained using the present developed software. The results of this comparison is shown in Table 4.1 where particle velocity from Li and Li [21] were estimated from Figure 4.6. Furthermore, the comparative results from Table 4.1 are plotted in Figure 4.7. The MATLAB code used to perform this test could be found in Appendix C21.

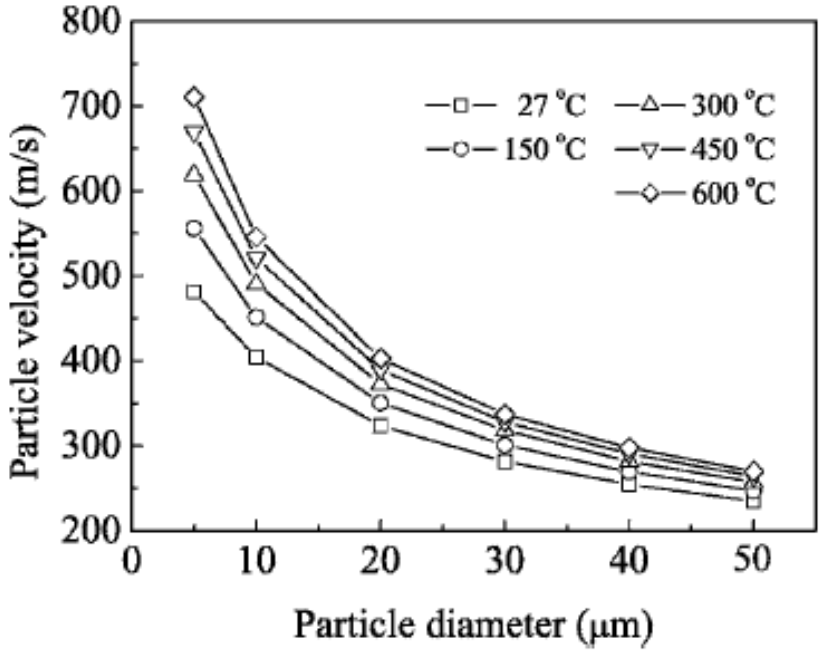

Figure 4.6: Effect of N_2 temperature on the velocity of particles for different particle diameters, and a pressure of 2 MPa [21].

Temp.	Particle Velocity for Cu powder using Nitrogen at 2 Mpa							Error	Max. Diff.
	Set of Diameters	5	10	20	30	40	50		
300.15 K	Set Vp_Reference	481	405	325	290	260	240	0.23	55.13
	Set Vp_This Works	475	425	370	337	313	295	0.39	
423.15 K	Set Vp_Reference	555	450	350	300	270	245	3.63	71.89
	Set Vp_This Works	535	473	406	366	338	317	0.84	
573.15 K	Set Vp_Reference	620	490	375	320	280	250	4.37	85.91
	Set Vp_This Works	592	516	438	392	360	336	1.81	
723.15 K	Set Vp_Reference	670	525	395	330	290	260	9.21	89.83
	Set Vp_This Works	637	550	462	411	376	350	2.45	
873.15 K	Set Vp_Reference	720	550	405	340	300	265	11.61	95.81
	Set Vp_This Works	675	578	481	426	389	361	3.13	

Table 4.1: Comparative results between the reference and this work for Cu particle velocity, and N_2 at $2\,MPa$ at different temperatures.

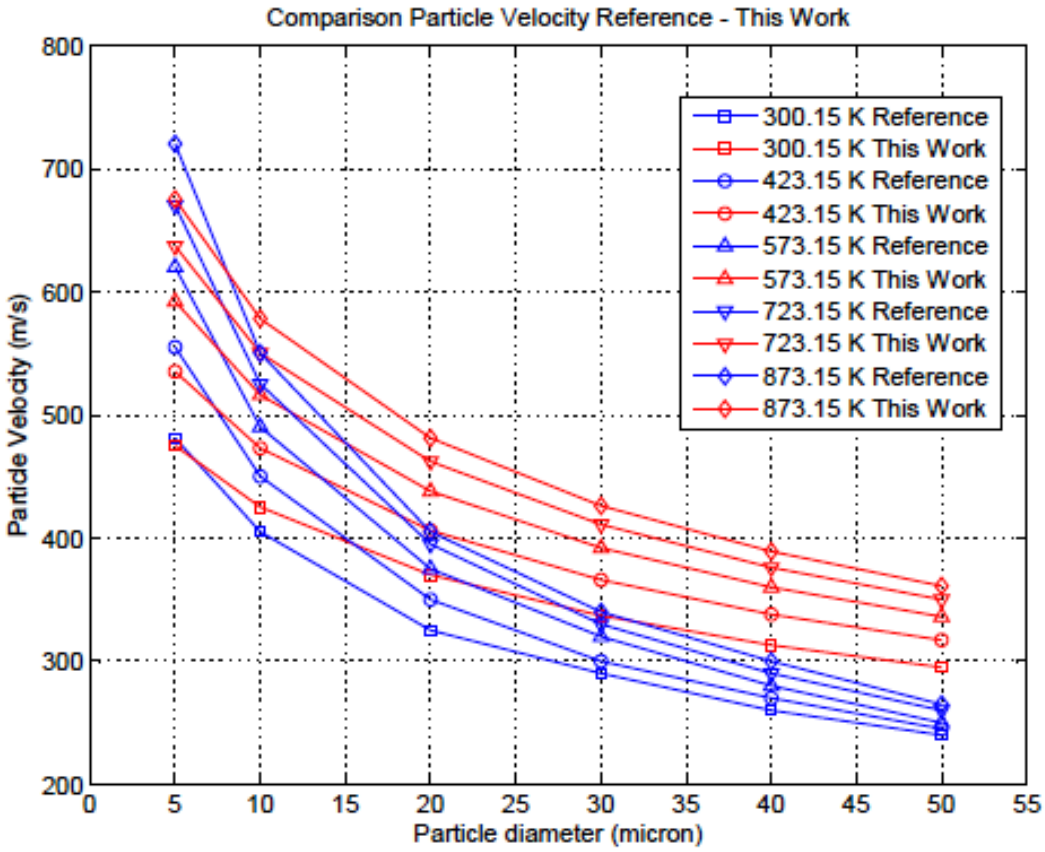

Figure 4.7: Comparison between the particle velocities in reference and this work for Cu powder, using N_2 at a pressure of 2 MPa.

In addition, a comparison between the calculated particles velocities and the reference was performed for different particle diameters d_p $(5, 10, 20, 30, 40$ and $50\ \mu m)$, and Nitrogen at a temperature of $573.15\,K$, but different pressures $(1, 2$ and 3 MPa). The estimated curves in Figure 4.8 are recorded in Table 4.2 together with the results obtained using the developed software. Figure 4.9 represents the plot of results in Table 4.2. The Matlab code used to perform this test could be found in Appendix C21.

Finally, by analyzing data from Figures 4.7 and 4.9 the following conclusions could be highlighted :

- there is a similarity between the analytical results and the CFD results referring to the pace of curves,
- v_p determined by CFD method is lower than the one determined using the developed software but this corroborates well with the finding results

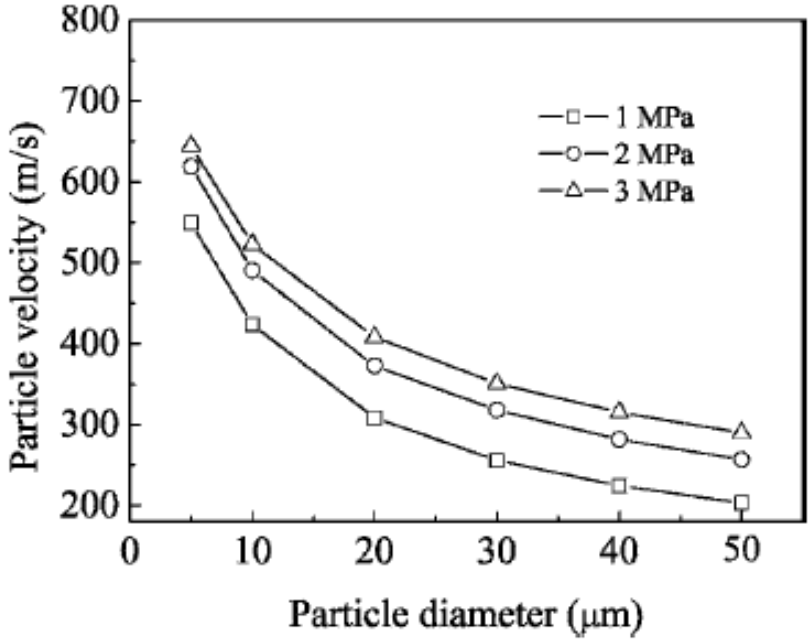

Figure 4.8: Effect of N_2 pressure on the velocity of particles with different sizes at a temperature of 300 [21].

Pressure	DATA Particle Velocity for Cu powder using Nitrogen at 573.15 K							Error	Max. Diff.
	Set of Diameters	5	10	20	30	40	50		
1 MPa	Set Vp_Reference	550	425	308	260	230	200	1.72	66.00
	Set Vp_This Works	516	438	360	317	288	266	1.68	
2 Mpa	Set Vp_Reference	623	495	375	316	283	259	5.99	77.65
	Set Vp_This Works	592	516	438	392	360	336	1.81	
3 Mpa	Set Vp_Reference	645	525	408	350	318	295	4.26	87.44
	Set Vp_This Works	633	561	484	437	405	380	1.71	

Table 4.2: Comparative results between the reference and this work for Cu particle velocity, and N_2 at $573.15\,K$ at different pressures.

in [6]. Note: the analytical method will give a gas velocity greater than the real conditions; the CFD results are closer to the real conditions because simulation conditions are made to be closer to real conditions; the maximum difference between each set of two curves in the two analyzed cases is less than $96\,m/s$ and this figure agrees with the previous experimentation presented by Champagne et al. [5].

- for better simulation results the particle diameter should be $8\,\mu m$ or higher,
- the maximum difference between the curves from Ansys Fluent and from

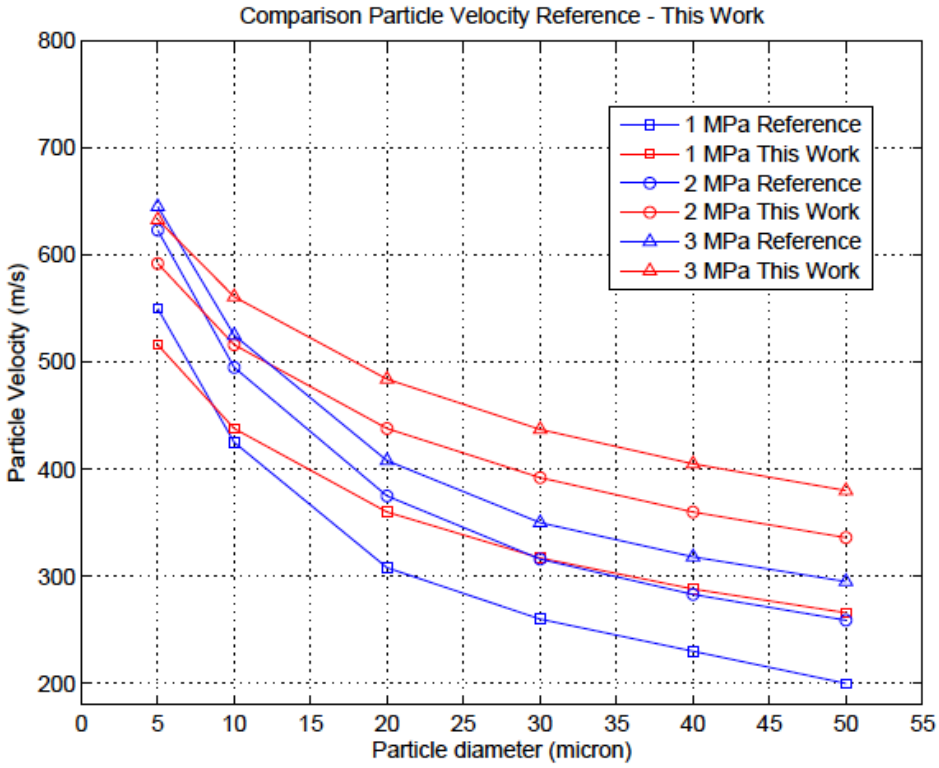

Figure 4.9: Comparison between particle velocities in reference and this work for Cu powder, using N_2 at a temperature of $573.15\,K$.

the new software increases with the increase of the temperature for a fixed pressure, and the maximum difference between the curves from Ansys Fluent and the new software increases with the increase of the pressure and a fixed temperature; this difference should be kept low when using analytical method; a small particle diameter gives a small difference between 2 curves plotted with the same conditions.

4.3.3.2 Test 2

This test was conducted using data from Stoltenhoff et al. in [29]. The characteristics of the De Laval nozzle used for tests are as follows: *throat diameter* equals $2.7\,mm$; *exit diameter* equals $8.1\,mm$; and the *divergent length* equals $90\,mm$. Other parameters include Nitrogen at a temperature of $593\,K$; Copper particles d_p of $15\,\mu m$; and gas pressure of $1.5, 2, 2.5, 3\,and\,3.5\,MPa$.

Then, using the same operating parameters, the results from Stoltenhoff et al. in [29] were compared with those obtained using the present developed software. The results of this comparison is shown in Table 4.3 where the particle velocity from Stoltenhoff et al. in [29] were estimated from Figure 4.10. Furthermore, the comparative results from Table 4.3 are plotted in Figure 4.11. The MATLAB code used to perform this test could be found in Appendix C22.

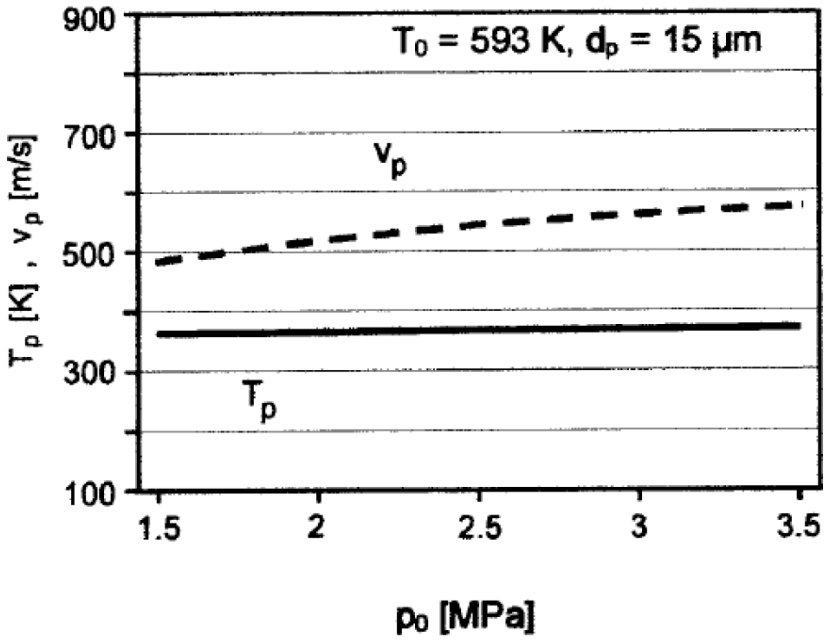

Figure 4.10: Temperature and velocity of Cu particles at the nozzle exit as function of the gas inlet pressure P_o [29].

Temp.	DATA Particle Velocity for Cu powder 15 micron using Nitrogen						Error	Mean. Diff.
	Set of Pressure	1.5	2	2.5	3	3.5		
593 K	Set Vp_Reference	480	520	555	570	580	0.00	56.80
	Set Vp_This Works	544	578	603	624	640	0.00	

Table 4.3: Comparative table between reference and this work for particle velocity for Cu powder of $15\,\mu m$, using N_2 at a temperature of $593\,K$ and at different pressures.

In addition, a comparison between particles velocities in the reference was performed, for different temperatures $(293, 393, 493, 593, 693\,and\,793\,K)$, using Nitrogen at a pressure of $2.5\,MPa$ and Copper particles with d_p equal to $15\,\mu m$. The curves from Figure 4.12 were estimated and their coordinates recorded in Table 4.4 together with the results obtained using the developed software. Figure 4.13 represents the results in Table 4.4. The Matlab code

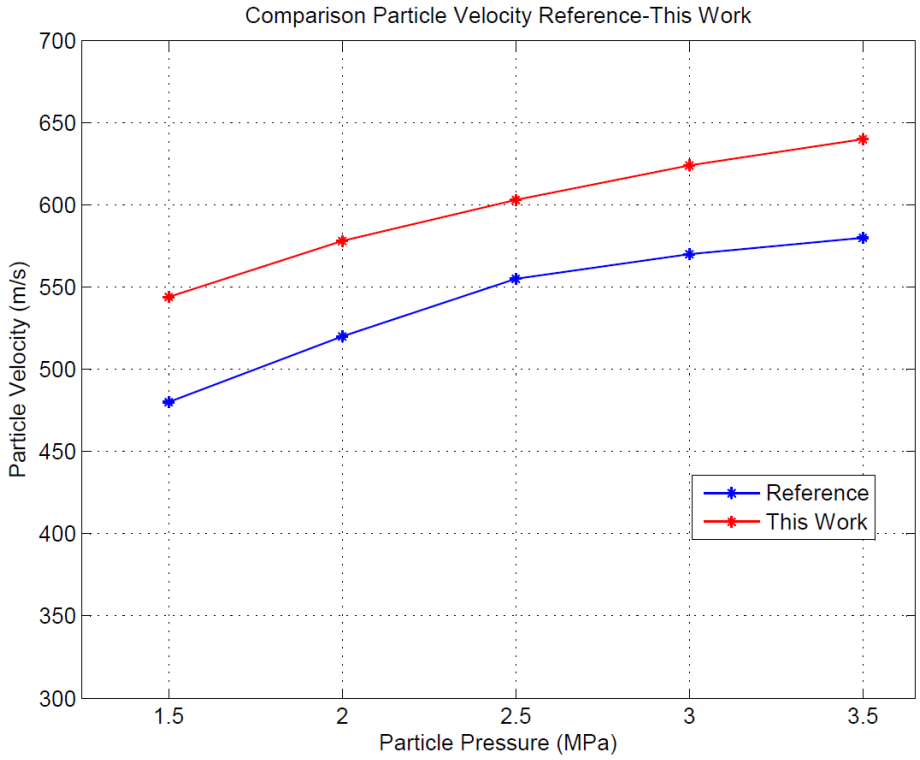

Figure 4.11: Comparative between particle velocities in reference and this work for Cu powder of 15 μm, using N_2 at a temperature of 593 K and at different pressures.

used to perform this test could be found in Appendix C23.

Finally, by analyzing the Figures 4.11 and 4.13, similar conclusions with the conclusions in test 1 could be drawn. However, it is important to remark that, the results for the test 2 give a difference between the curves of less than 60 m/s.

Pressure	DATA Particle Velocity for Cu powder 15 micron using Nitrogen							Error	Mean. Diff.
	Set of Temperature	293	393	493	593	693	793		
2.5 Mpa	Set Vp_Reference	450	480	510	535	565	585	2.43	57.17
	Set Vp_This Works	477	527	569	603	633	659	1.19	

Table 4.4: Comparative table between the reference and this work for particle velocity for Cu powder of 15 μm, using N_2 at a pressure of 2.5 MPa and at different temperatures.

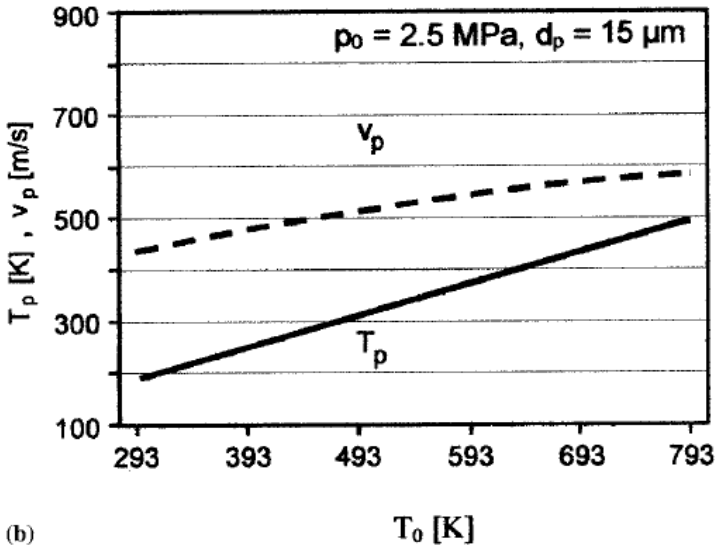

Figure 4.12: Temperature and velocity of Cu particles at the nozzle exit as function of the gas inlet temperature T_o [29].

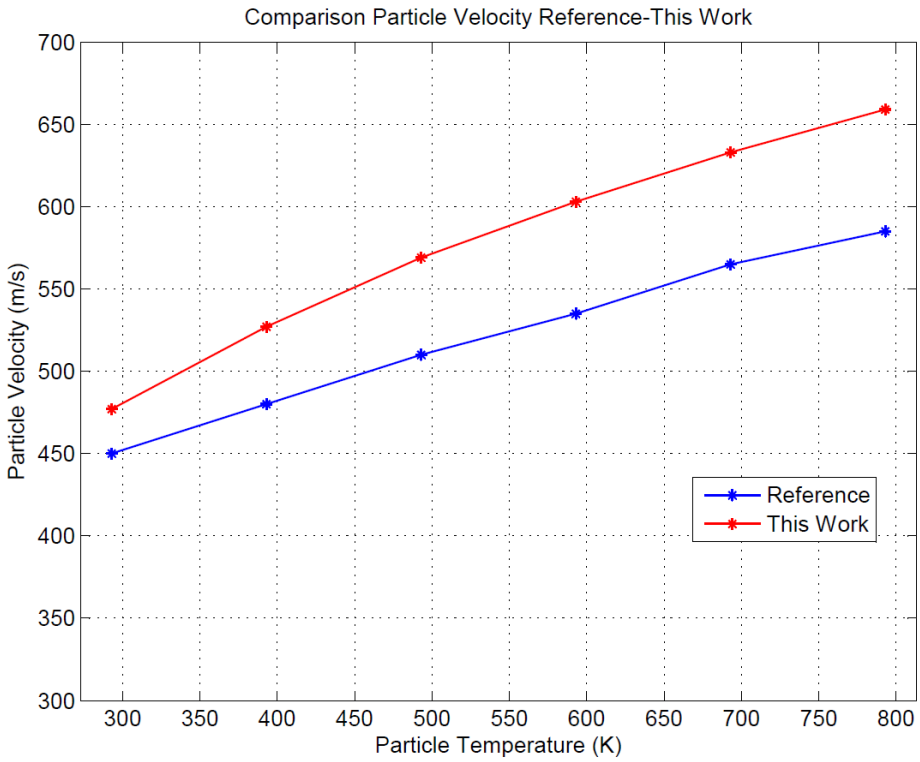

Figure 4.13: Comparison between the particle velocities in reference and this work for Cu powder of 15 μm, and using N_2 at a Pressure of 2.5 MPa.

4.3.3.3 Test 3

This test was conducted considering a De Laval nozzle used in the 'Integrated Supersonic Spray Technology Laboratory' at Wits University. The characteristics of this nozzle are as follows: *throat diameter* equals $2\,mm$; *exit diameter* equals $6\,mm$; *divergent length* equals $136.8\,mm$, *input diameter* equals $9.7\,mm$; *convergent length* equals $3.8\,mm$, and the length of the barrel at input equals $27\,mm$. Nitrogen was selected as the carrier gas at 1.48296 MPa, the powder was Aluminum particles with d_p equals $27\,\mu m$, and the selected working temperature was $550\,K$.

Particle velocity v_p in m/s was determined using the developed software. The results of these calculations are presented in the GUI format in Figure 4.14. Also, v_p was determined with v_g obtained using ANSYS Fluent software and then, the two results were compared and discussed.

Note: The following data represents the parameters for v_g calculation using ANSYS Fluent software.

Mesh generation

In order to analyze the flow in the nozzle, a mesh was created automatically using the *'Quadrilaterals'* method in the Ansys software. Figure 4.15 presents the resultant mesh with the following details:
97172 nodes, binary.
173 nodes, binary.
3357 2D wall faces, zone 1, binary.
189396 2D interior faces, zone 2, binary.
126 2D pressure-inlet faces, zone 6, binary.
45 2D pressure-outlet faces, zone 7, binary.
95580 quadrilateral cells, zone 3, binary.

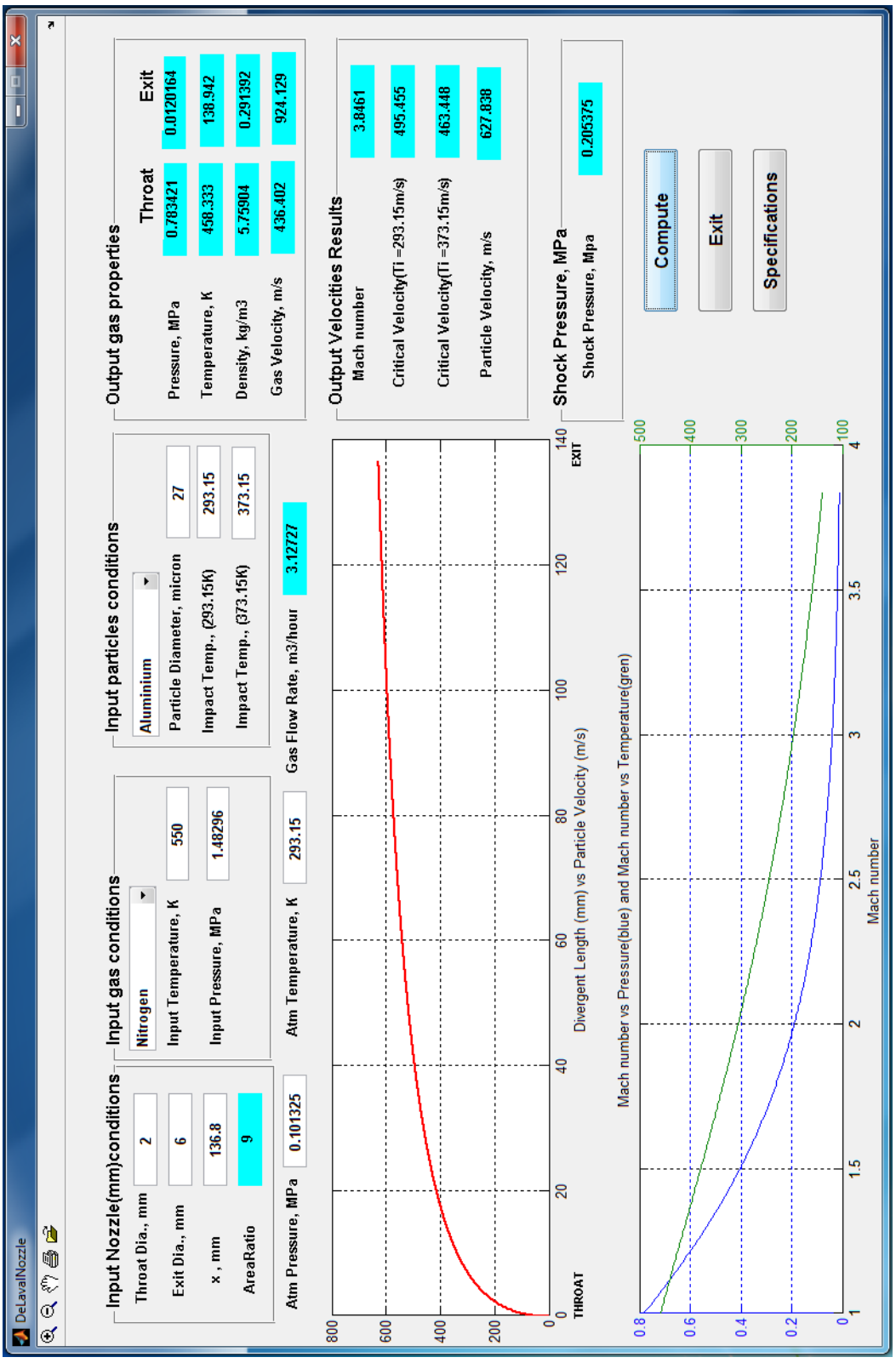

Figure 4.14: Results for Aluminium particles using N_2, $d_p = 27\,\mu m$, $T_o = 550\,K$ and $P_o = 1.48296$ MPa, Area ratio = 9,$d^* = 2\,mm$.

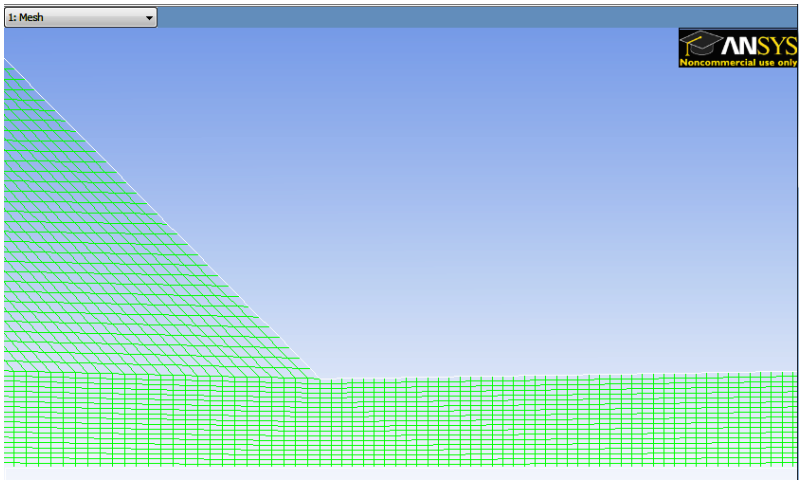

Figure 4.15: Mesh used for simulations.

Simulation

The simulation progress was conducted following the steps below:

- select **'Solver'**; chose *Type* as *Pressure-Based*, *Velocity Formulation* as *Absolute*, *Time* as *Steady* and *2D Space* as *Planar*.
- for the **'Model'**, make sure that *Energy Equation* is selected in the *Energy* window, and the standard $K - \epsilon$ turbulence model was chosen for the simulation.
- for the **'Materials'**, air was selected.
- the *'Operating Pressure'* in the *'Operating Conditions'* window was set to 0.
- for **'Boundary Conditions'**, in the window *'Pressure Inlet'*, the *'Gauge Total Pressure' (pascal)* was entered equal to 1482960 and the *'Supersonic/Initial Gauge Pressure' (pascal)* was entered equal to 783422 and the *'Total Temperature' (Kelvin)* was entered equal to 550; in the window *'Pressure Outlet'*, the *'Gauge Pressure' (pascal)* was selected equal to 12016.4 and the *'Backflow Total Temperature' (Kelvin)* equals to 138.942; the *wall* was set to *wall boundary* type.
- for the **'Solution Methods'**, the flow was kept *Default* as proposed by the software.

- for the **'Solution Initialization'**, *Relative to Cell Zone* was selected as *Reference Frame*.
- for **'Convergence Criteria'**, the solution was iterated until the residual for the equations falls bellow $1e-6$, and
- the **'Number of Iterations'** was fixed to 4500.

Based on the above settings, Figure 4.16 gives the simulation result for the velocity (m/s); Figure 4.17 gives the simulation result for the static pressure (Pascal), Figure 4.18 gives the simulation result for static temperature (Pascal), and Figure 4.19 gives the simulation result for the Mach number.

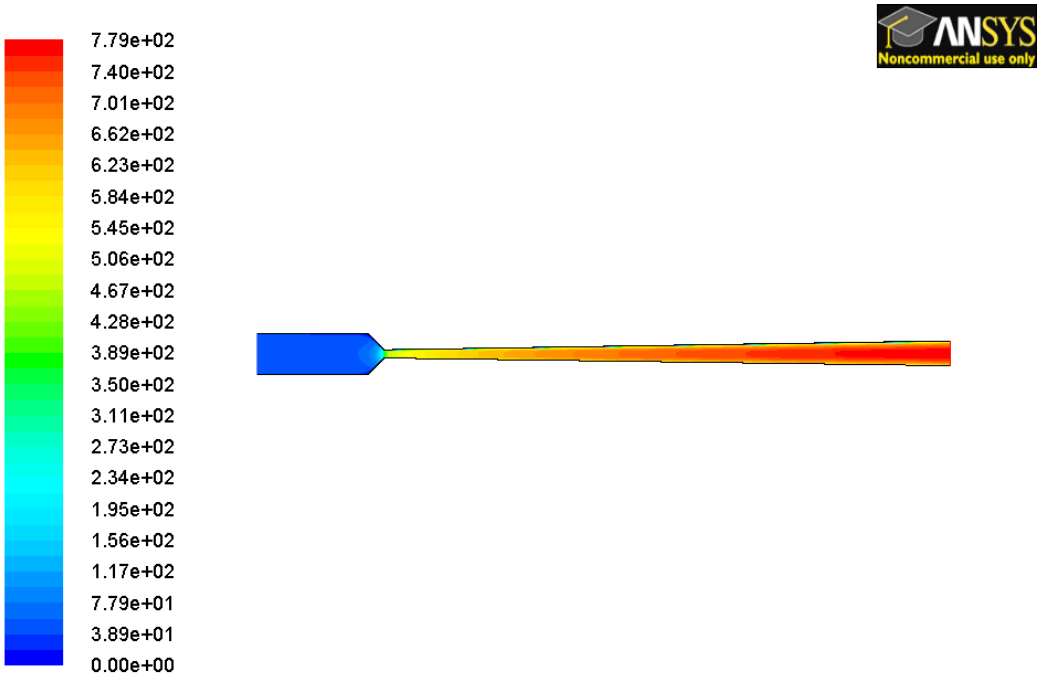

Figure 4.16: Contours for the velocity (m/s) using N_2 with $T_o = 550\,K$ and $P_o = 1.48296$ MPa for De Laval Nozzle, Area ratio = 9,$d^* = 2\,mm$.

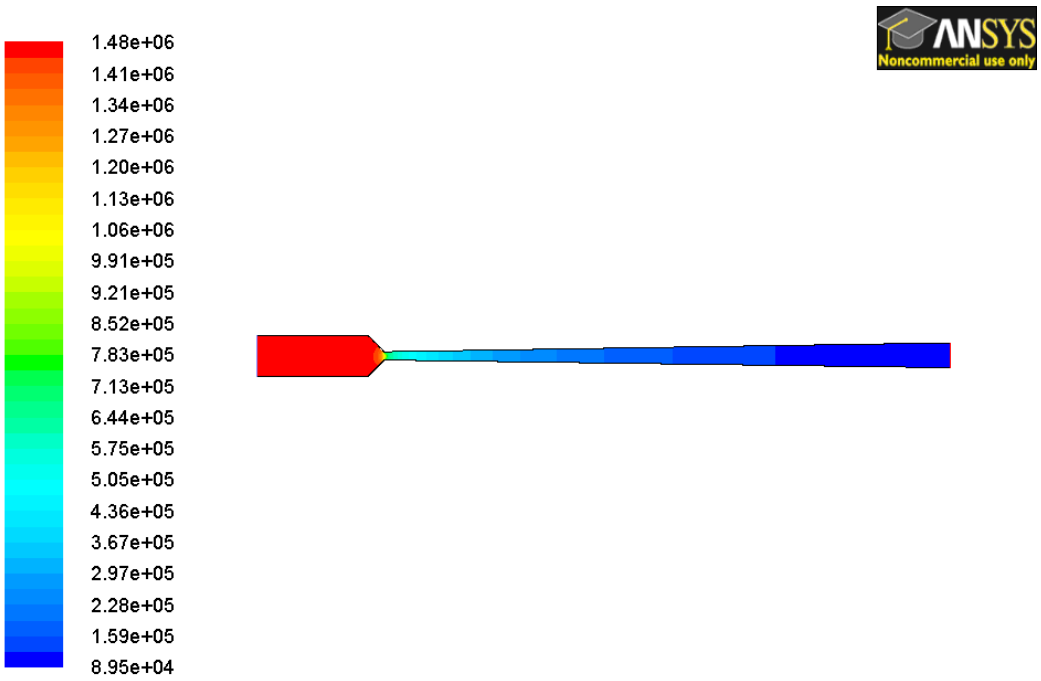

Figure 4.17: Contours for the static pressure (Pascal) using N_2 with $T_o = 550\,K$ and $P_o = 1.48296$ MPa for De Laval nozzle, Area ratio = 9,$d^* = 2\,mm$.

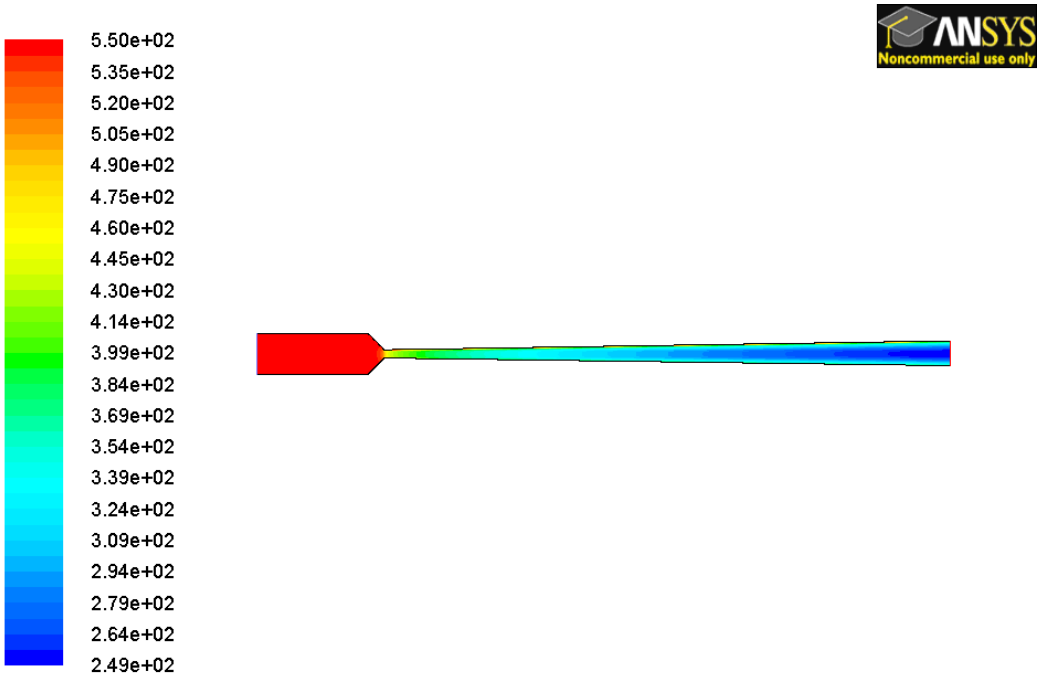

Figure 4.18: Contours for the static temperature (Kelvin) using N_2 with $T_o = 550\,K$ and $P_o = 1.48296$ MPa for De Laval nozzle, Area ratio = 9,$d^* = 2\,mm$.

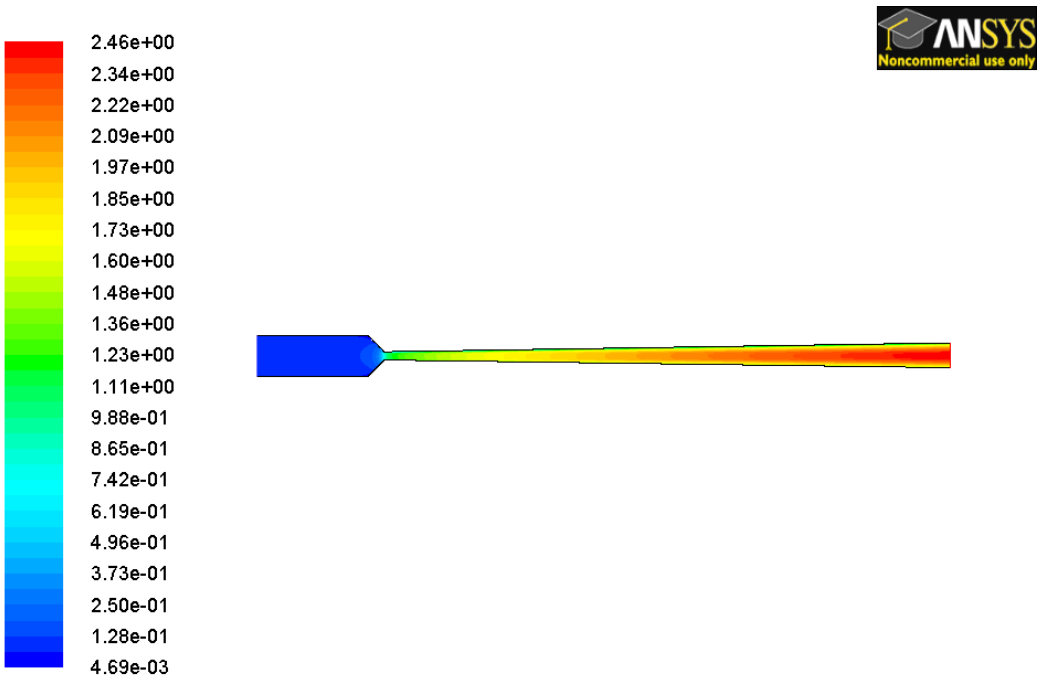

Figure 4.19: Contours for the Mach number for De Laval nozzle, Area ratio = 9,$d^* = 2\,mm$, using N_2 with $T_o = 550\,K$ and $P_o = 1.48296$ MPa.

In addition, using data from Figure 4.16 and Alkimov's Equation 2.33, v_p was determined as follows:

$$v_p = \frac{779}{1 + 0.85 * \sqrt{\frac{27*10^{-6}}{136.8*10^{-3}}} * \sqrt{\frac{2712*779^2}{1.482960*10^6}}} = 557\,m/s$$

Finally, comparing the results for the v_p obtained using the developed software and v_p from v_g determined using the ANSYS software, it is concluded that:

- the difference of $71\,m/s$ between the v_p determined using the new GUI ($v_p = 628\,m/s$) and the v_p determined using v_g from ANSYS Fluent ($v_p = 557\,m/s$) could be due to the fact that analytical method used selected parameters while the CFD method used more realistic conditions. (the CFD method considers the boundary layer condition)

- the value of 2.46 for the exit Mach number found in Figure 4.19 is less than the one found in the GUI tool. This could also be explained by the fact that the CFD method considers more realistic conditions; in fact, the CFD method considers the boundary layer condition. Furthermore, the

physical properties of fluid flow are governed by the mass conservation equation, the momentum conservation equation and the energy conservation equation, written in the form of partial differential equations, and solved by numerical method; this approach allowed an analysis more close to real situation.

Finally, it should be noted that, for all the above tests, the difference between $v_{crit}^{th,mech}$ as estimated by the developed software and the results presented by Schmidt et al. [27], could be justified by the very specific condition in Schmidt's experiment which includes aspects such as the tensile strength (σ_{TS}) and heat capacity (c_p) of the particle material, as well as the difference between the powder particle size used in calculation and the actual particle used in the experiment.

4.4 MOC Nozzle Design

In order to increase the particle impact velocities and to remove or significantly reduce the possibility of shockwaves generation when using De Laval nozzle, a new nozzle was designed using the Method of Characteristic.

4.4.1 GUI interface

The new GUI was developed using MATLAB. Figure 4.20 shows the input parameters, output calculations, curved section of the MOC nozzle, and the whole divergent section of the MOC nozzle. Also, Figure 4.21 shows a practical use of the GUI interface. The associated GUI code can be found in Appendix D2.

In order to operate this new interface, the user has to select the button *'Contour Design Moc Nozzle'* in the *MainInterface* (Figure 4.3) That will open the *'MocNozzle.m'* file. Then, the user must run the file that will display the *GUI Interface* as shown Figure 4.20 and with the following areas:

Input Parameters

– `Mach Number` as planned

– `Specific Heat`

– `Constant of Perfect Gas, R` (J/KgK)

– `Diameter Throat` (mm)

– `Divergent Length` (mm)

Output Results

– `Max angle duct wall` (radians)

– `Length Curve Section` (mm)

– `Exit Diameter` (mm)

– `Area Ratio`

– `Exit Mach Number` (at Curve part)

Curve Section MOC Nozzle – this displays the curve section of the nozzle and the coordinates of the points in the plot can be read under *X,mm* and *Y,mm*.

Divergent Section MOC Nozzle – this displays the entire internal profile of the nozzle.

Computer and Plot – it runs the calculations and displays the plotting results.

Exit – to exit the program.

Help buttom. – to get some help with the data.

Note: In order to enter a new set of data, the interface must be closed.

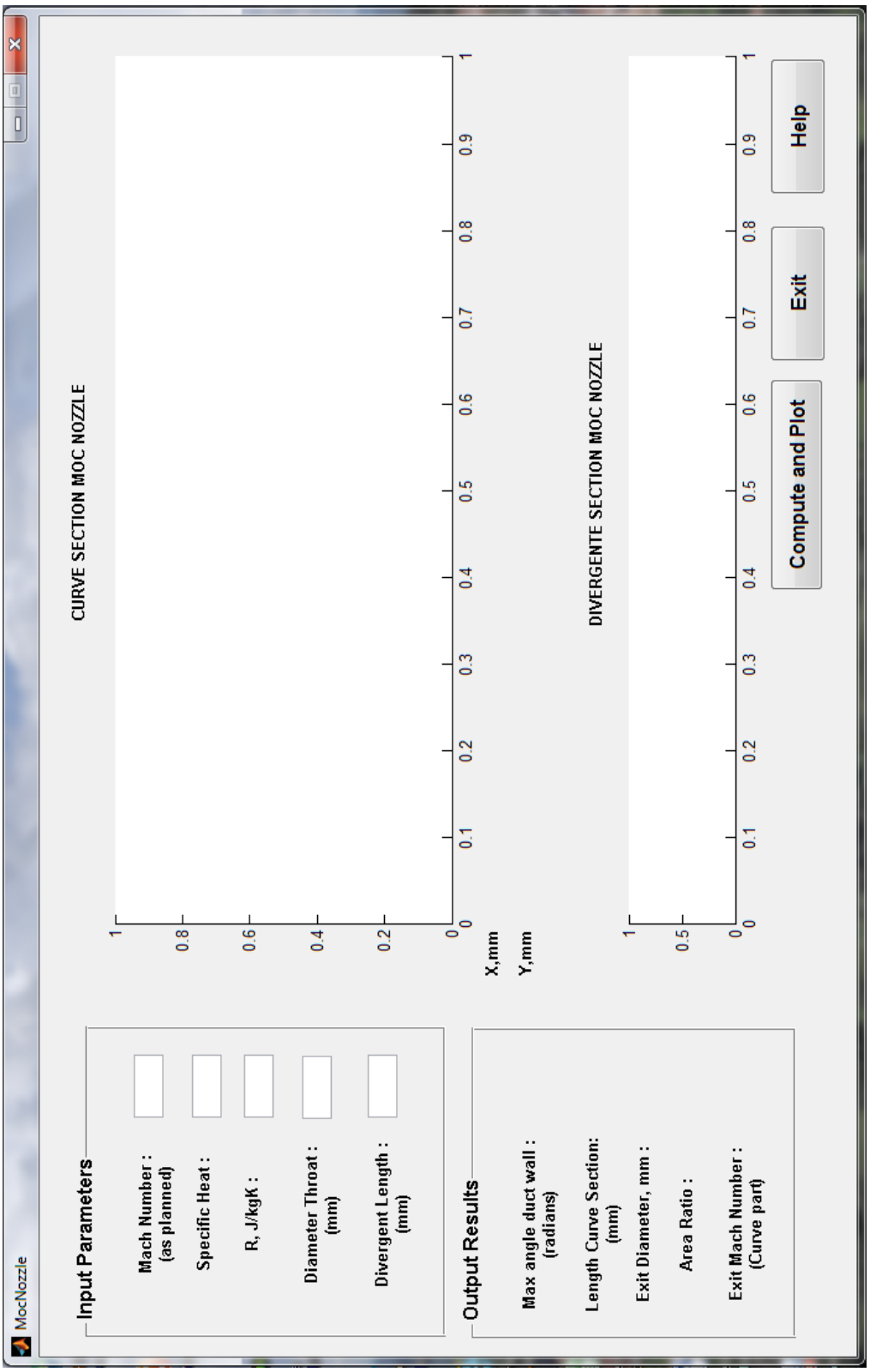

Figure 4.20: Main window for the MOC nozzle contour simulation.

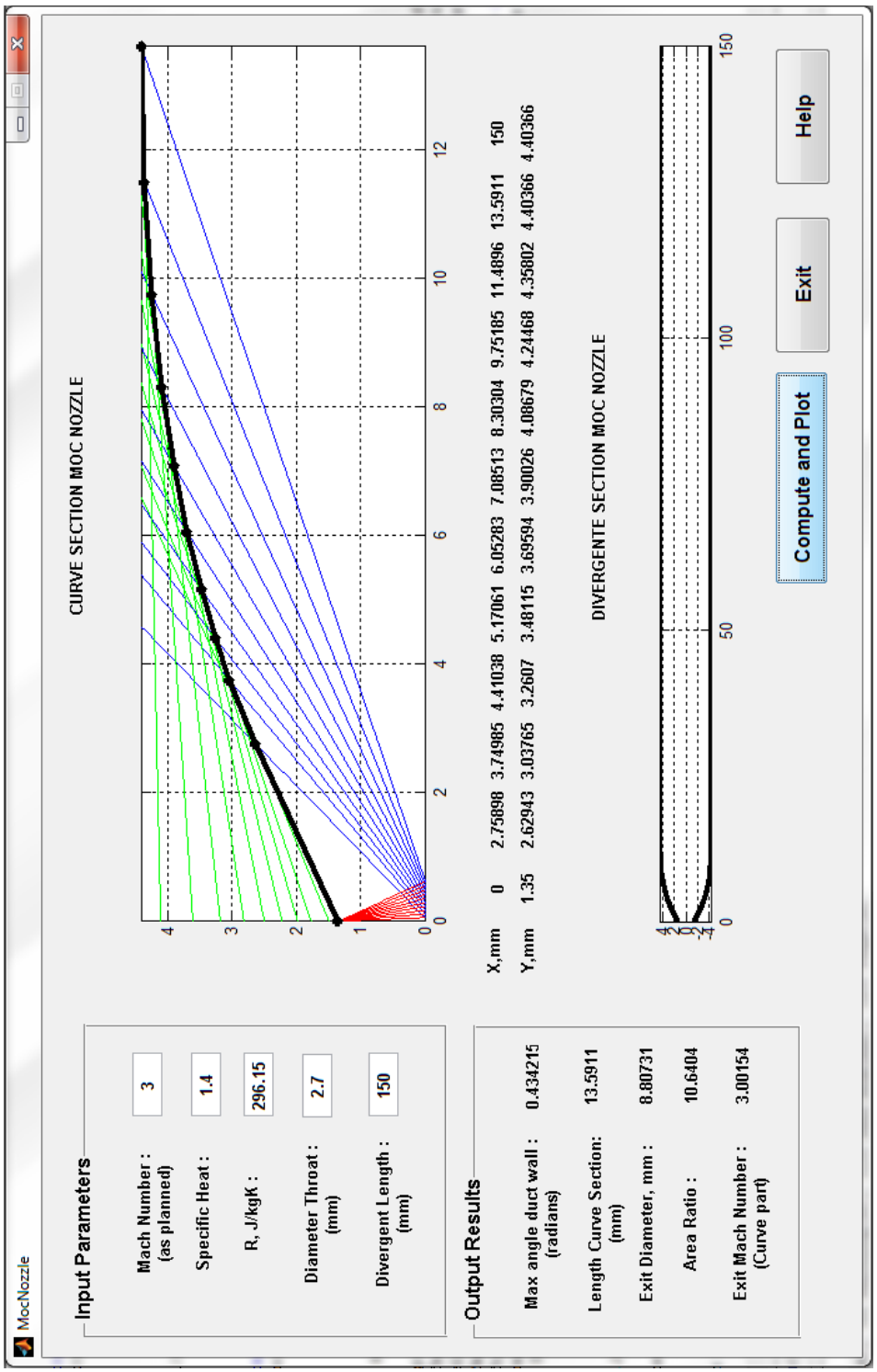

Figure 4.21: Contour of the MOC nozzle for $\gamma = 1.4$, $d_{throat} = 2.7\,mm$, $M = 3$, $Length = 150\,mm$.

4.4.2 Test results and analysis

In order to conduct the test, the input parameters must be specified.
For this test, the *Mach number* varies from 1.4 to 3 with a step equals to 0.2 and with a radius equal to the unit. Nitrogen (specific heat = 1.4) was used as the carrier gas and the nozzle throat diameter was fixed to $2\,mm$. Table 4.5 presents the experimental results for the following parameters:

- Mach number as planned – this is defined at the beginning of the program, and should be found at the exit of the curved section
- Mach number at the exit of curve section,
- exit diameter – diameter of the nozzle at the exit,
- area ratio – the ratio between the exit section area and the throat section area,
- length of curved section – the *minimum length* nozzle as defined previously, and
- the maximum angle of duct wall – *wall maximum* as previously defined in the literature survey (see section 2.4.3.3)

The plots of the results from Table 4.5 are presented in Figure 4.22. Also, analyzing Table 4.5 and Figure 4.22, the following could be highlighted:

- the maximum difference between the planned Mach number and the Mach number obtained at the exit of the curved part of the nozzle is equal to 0.00154
- the minimum difference between the planned Mach number and the Mach number at the exit of curved part of nozzle is equal to 0.00016
- this difference increases with the Mach number, however the maximum achieved difference of 0.00154 is enough accurate. This accuracy shows that the number of 10 characteristic lines retained to build the program are acceptable for the design of the MOC nozzle with a maximum Mach number equals to 3 (see Appendix B).

Mach number (as Planned)	1.4	1.6	1.8	2	2.2	2.4	2.6	2.8	3
Exit Mach number(curve part)	1.40016	1.6003	1.80046	2.00064	2.20083	2.40102	2.6012	2.80138	3.00154
Exit Diameter (mm)	2.19808	2.40153	2.67546	3.02775	3.46945	4.01687	4.6901	5.51493	6.52394
Area Ratio	1.20788	1.44184	1.78948	2.29148	3.00926	4.03382	5.49926	7.6036	10.6404
Length Curve Section (mm)	1.46019	1.90937	2.45456	3.12645	3.96205	5.00748	6.32089	7.97615	10.0675
Max Angle duct wall (radians)	0.07843	0.12968	0.18086	0.23021	0.27692	0.32067	0.36141	0.39921	0.43422

Table 4.5: Mach Number comparative table for MOC nozzle design, with radius at the throat equal to 1, using N_2 ($\gamma = 1, R = 296.15K$).

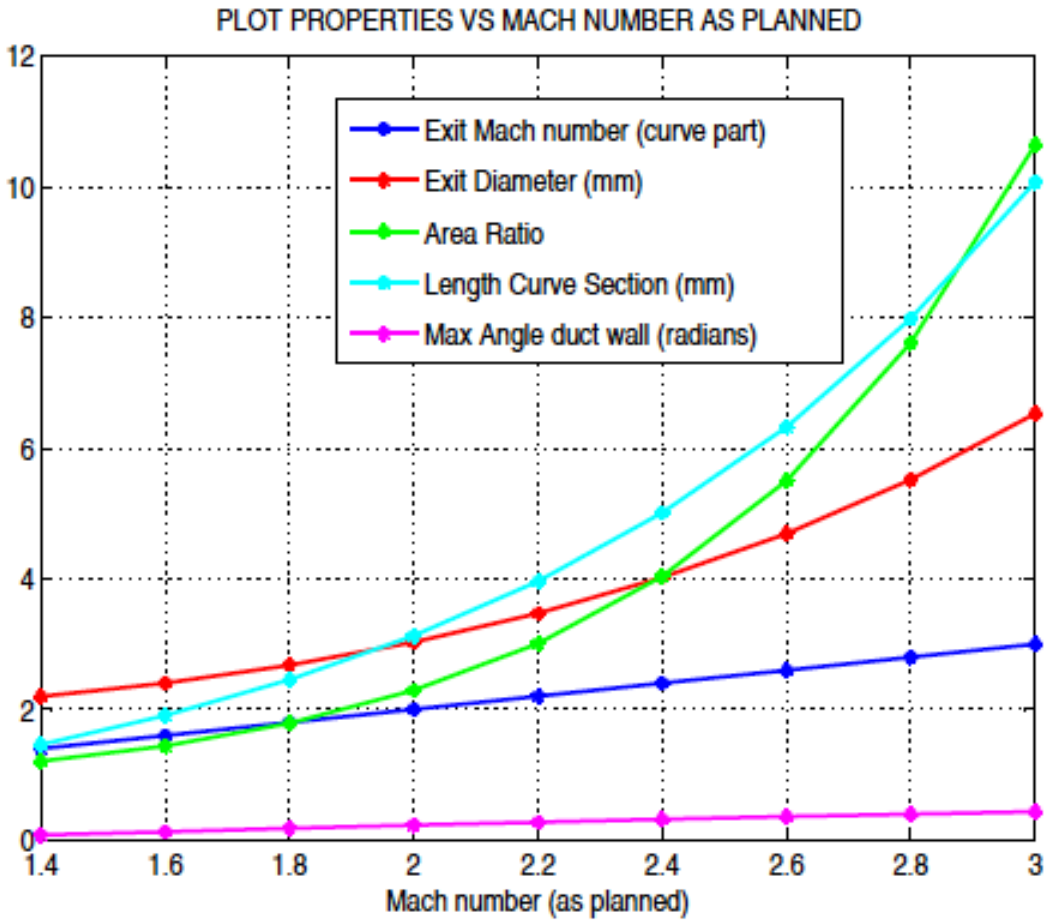

Figure 4.22: Plot properties vs Mach number as planned.

Finally, the exit diameter, and consequently the area ratio, the length of curve part, and the maximum angle of the duct wall increase with the planned Mach number. Furthermore, it could be concluded that for the same conditions the area ratio of a MOC nozzle is higher that the area ratio of a De Laval nozzle.

4.4.3 Flow analysis using CFD method with ANSYS Fluent

The CFD simulation can be used for a proper characterization of the gas flow in the CGDS process. The CFD procedure uses a number of equations, simplifications and assumptions, and a meshing tool to transform the physical domain into a computational domain. The CFD package used in this work is Ansys Fluent 12.0.16. The calculation was limited only to the gas jets of the

MOC nozzle but Equation 2.33 could be used to link v_g and v_p. In addition, it should be mentioned that the CFD investigation presented in this section is limited to the simulation of the gas flow in the divergent part of the MOC nozzle, the evaluation of the gas velocity and its Mach number, pressure, and temperature at the exit of the nozzle.

4.4.3.1 Nozzle geometric details and mesh generation

In this section, the geometry of the nozzle in Figure 4.21 is used. Therefore, $Mach\,number = 3$, $\gamma = 1.4$, $R = 296.15 J/Kg\,K$, $Throat\,diameter = 2.7\,mm$, and $Divergent\,Length = 150\,mm$. Note that the $Throat\,diameter = 2.7\,mm$ was selected as smallest cross section for the acceleration of Copper powder according with the previous study [11].

Using the above parameters, the (x,y) coordinates that will define the MOC nozzle wall profile are presented in Table 4.5. Note that these values could be found in Figure 4.21.

x–coordinate	y–coordinate
0	1.35
2.75898	2.62943
3.74985	3.03765
4.41038	3.2607
5.17061	3.48115
6.05283	3.69594
7.08513	3.90026
8.30304	4.08679
9.75185	4.24468
11.4896	4.35805
13.5911	4.40366
150	4.40366

Table 4.6: Coordinates (x,y) for the MOC Nozzle wall profile with the following properties: $Mach\,number = 3$, $\gamma = 1.4$, $R = 296.15 J/Kg\,K$, $Throat\,diameter = 2.7\,mm$ and $Divergent\,Length = 150\,mm$.

Then, in order to generate the nozzle shape in ANSYS, the coordinates from Table 4.6 were entered manually in the *'Ansys Design Modeler'* tool (DM). With this data, the mesh was created automatically using the *'Mapped Face Meshing'* and the *'Quadrilaterals'* method. Figure 4.23 presents the meshing result on a portion of the nozzle length with that has the followed values:

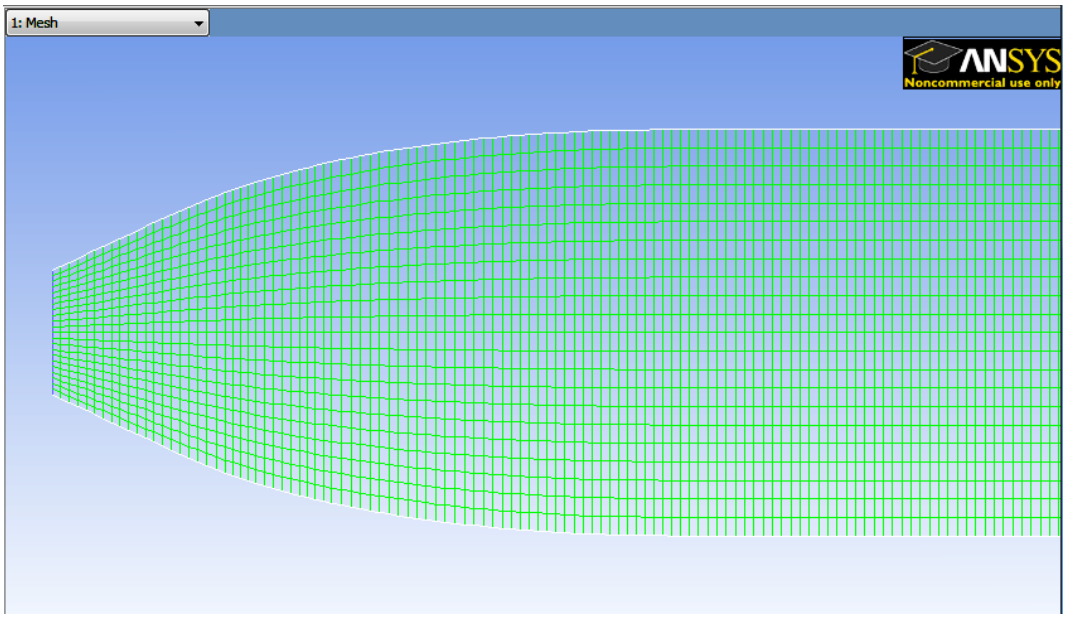

Figure 4.23: Example of mesh used for simulations.

- 17296 nodes, binary.
- 46 nodes, binary.
- 1506 2D wall faces, zone 1, binary.
- 32357 2D interior faces, zone 2, binary.
- 22 2D pressure-inlet faces, zone 6, binary.
- 22 2D pressure-outlet faces, zone 7, binary.
- 16566 quadrilateral cells, zone 3, binary.

4.4.3.2 Simulation

In order to simulate the gas flow within the MOC nozzle, the parameters from Figure 4.24 were used. These are as follows:

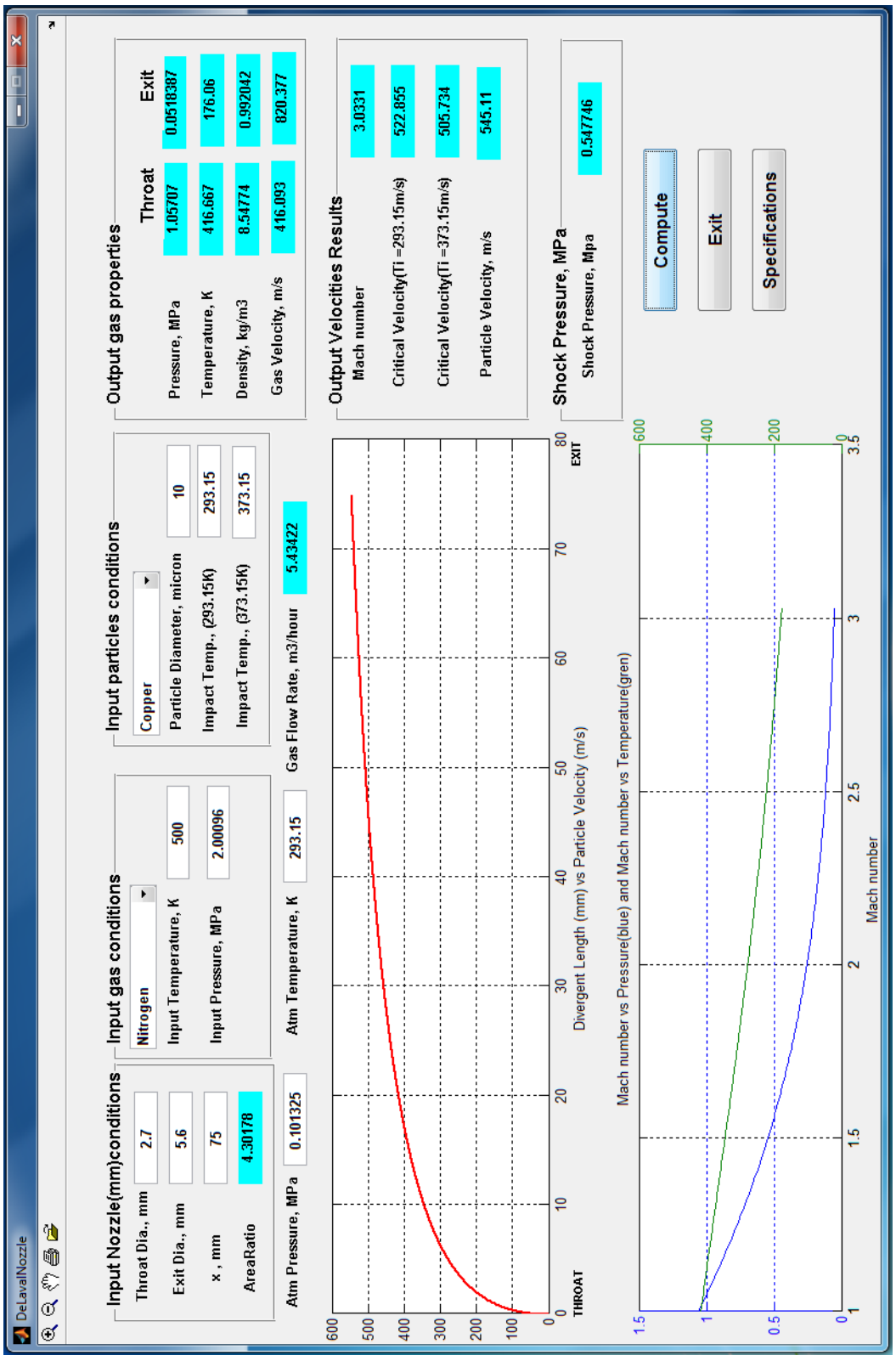

Figure 4.24: Results for Cu particles using N_2, $d_p = 10\,\mu m$, $T_o = 500\,K$ and $P_o = 2$ MPa, Area ratio $= 4.3$,$d^* = 2.7\,mm$.

- for the **'Solver'** selected, chose *'Type'* as *'Density–Based'*, *'Velocity Formulation'* as *'Absolute'*, *'Time'* as *'Steady'* and *'2D Space'* as *'Planar'*.
- for the **'Model'**, make sure that *'Energy Equation'* is clicked in the *'Energy'* window, and the standard $K - \epsilon$ turbulence model was chosen for the simulation.
- for the **'Materials'**, air was selected with properties of Ideal–Gas.
- the *'Operating Pressure'* in the *'Operating Conditions'* window was set to 0.
- for the **'Boundary Conditions'**, in the window *'Pressure Inlet'*, the *'Gauge Total Pressure (pascal)'* equals 2000960 and the *'Supersonic/Initial Gauge Pressure (pascal)'* equals 1057070, the *'Total Temperature (Kelvin)'* equals 416.667, the *'Gauge Pressure (Pascal)'* equals 51838.8 and the *'Backflow Total Temperature (Kelvin)'* equals 176.06. The *'wall'*was set to *'wall boundary'* type.
- for the **'Solution Methods'** the flow was selected to *'Second Order Upwind'*.
- for the **'Solution Initialization'**, as *'Reference Frame'*, *'Relative to Cell Zone'* was selected. And as *'Initial Values'*, *'Gauge Pressure (Pascal)'* was selected to 1057070, *'Axial Velocity (m/s)'* was selected to 416.093, Radial Velocity (m/s) was selected to 0 and the Temperature (Kelvin) was selected to 416.667.
- the **'Convergence Criteria'**, the solution was iterated until the residual for the equations falls bellow $1e - 6$.
- the **'Number of Iterations'** was fixed to 4500.

Based on the above settings, Figure 4.25 gives the simulation result of the contours of velocity magnitude (m/s).

Also, using the Alkimov's equation (Equation 2.33) v_p can be calculated as follows:

$$v_p = \frac{741}{1 + 0.85 * \sqrt{\frac{15*10^{-6}}{150*10^{-3}}} * \sqrt{\frac{7870*741^2}{2*10^6}}} = 560\, m/s$$

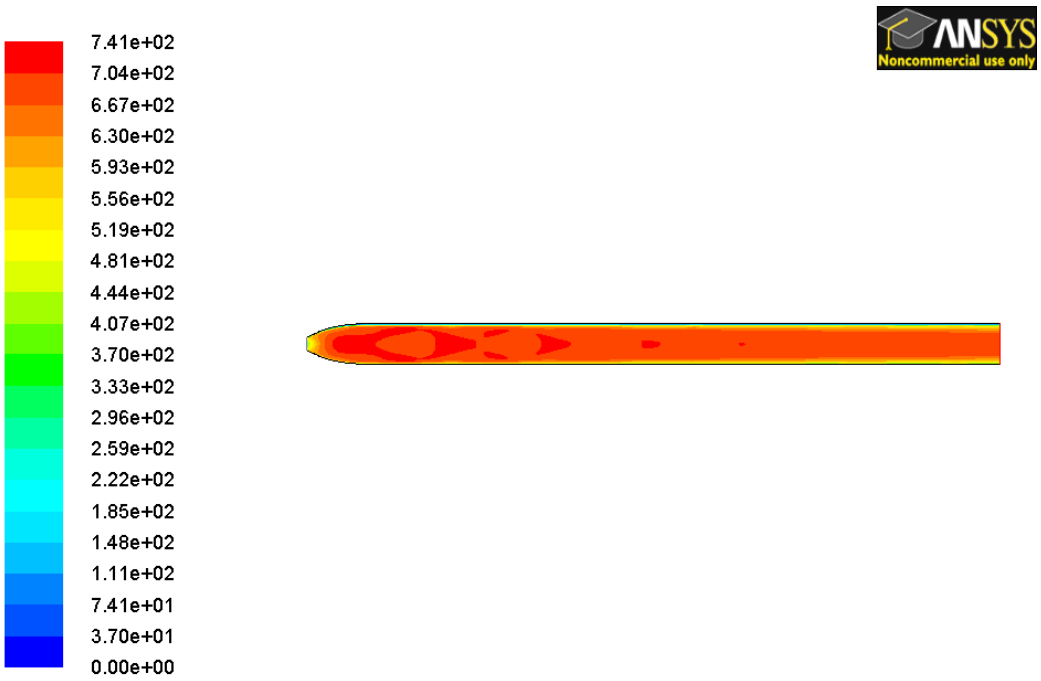

Figure 4.25: Contours of gas velocity (m/s) for a MOC nozzle with $Mach\,number = 3$, $\gamma = 1.4$, $R = 296.15 J/Kg\,K$, $Throat\,diameter = 2.7\,mm$ and $Divergent\,Length = 150\,mm$.

4.4.3.3 Analysis of results

Analyzing the results, it could be concluded that:

- the flow of the gas is disturbed at the exit of the curved part of the nozzle and stabilized in its barrel section. The maximum difference of velocities between these areas is evaluated at $37\,m/s$ which is an acceptable value for the range of velocities used in CGDS. However, even the difference in particle velocities above is not significant, an improvement can be envisaged due to the straight and concentrated gas flow at the exit section of the nozzle.
- the difference between the v_p using the De Laval nozzle and the v_p using the MOC nozzle is $15\,m/s$ that is justified by the method selected for simulation and boundary conditions that reduce the simulated velocity of the gas compare with the values achieved using the new software.
- previous studies demonstrated that the presence of substrate had little influence on particle acceleration; therefore, it could be concluded that,

the straight gas flow at the exit of the MOC nozzle and the reduced or eliminated shock waves inside of the MOC nozzle, will improve the quality of the CGDS deposition process (Figure 2.14).

4.5 Summary

There is a need to predict the correct particle deposition parameters used in the CGDS process before any practical experimentation is performed. Therefore, this chapter presented the development of a new computational tool using MATLAB that is capable to calculate the performance of an existing De Laval nozzle, and also is capable to calculate and design the internal profile of high performance (high gas speed with no or reduced shock waves) MOC nozzles. In addition, in order to test the new developed software, a number of tests were conducted. The results of the tests showed that the new software developed in MATLAB are very similar to those found in the peer reviewed and published literature.

5 CONCLUSION AND FUTURE WORKS

5.1 Introduction

This dissertation has been organised into five chapters which were structured, unified, and focused on solving one research problem. The first chapter set the scene by introducing the core research problem and outlined the path that the reader will travel towards its conclusion. Chapter 2 identified from the existing body of knowledge issues related to the calculation and design of the nozzles used in CGDS process, and unified this knowledge into a new mathematical model. Then, Chapter 3 presented the methodologies used to answer the research question and Chapter 4 used these methods to develop a new software for nozzle calculations and design. Finally, Chapter 5 briefly summarised the previous chapters, and then, prior to making conclusions about the research, it explains how the new and the old pieces fit to make the whole picture clear.

5.2 A Brief Overview of Previous Chapters

The problem addressed in this research was:

How to calculate and design the internal profile of a CGDS nozzle and so effectively and successfully achieve a Cold Spray deposition?

In order to find the answer to the research problem, the literature survey focused on the issues of the De Laval nozzle calculation and design, but also considered the complex and multifaceted issues of the MOC nozzle calculation and design.

As a result of this research, it has been found that the CGDS process requires a supersonic high velocity stream of gas to accelerate the powder particles at velocities exceeding particle's critical velocity. However, many times, CGDS experiments were carried out using inadequate process parameters and so wasting unnecessary time and money. Therefore, the optimization of the nozzle's geometry and the simulation of the CGDS process parameters were considered critically imperative.

Consequently, this dissertation developed a new software that allows the simulation of the gas and particles velocities for a large variety of CGDS process parameters. Software development used a number of established techniques that included the MATLAB software that was used to generate practical graphs for the CGDS process and generate the 2D recommended nozzle contour and the Computational Fluid Dynamics (CFD) software that was used to calculate and visualize the gas flow. Finally, the results obtained using the two developed technologies were compared with data from the peer reviewed journal papers and it was found that the results obtain using the new MATLAB software and ANSYS Fluent were very similar with data found in the literature survey.

5.3 Conclusions about the Reseach Questions

Based on the dissertation's qualitative findings, the research problem presented above can now receive a concise answer. Essentially, it is argued that for an efficient and successful cold gas dynamic spray deposition there is a need for:

- an in–depth understanding of the cold spraying process and its parameters,
- an unified mathematical model of various cold spray theories, and

- new nozzle designs with improved performance. However, because the design of high performance nozzles is a complex and time consuming process, there is a need for:
- new software as the one developed in this dissertation, that
- promote the simplification of the nozzle design process and also
- supports the human which should be an integrated part of the CGDS system.

5.4 Research Implications

The research performed in this dissertation has theoretical and practical implications.

The implication for theory refers to the development of a unified cold gas dynamic spray process mathematical model focused on the nozzle design that was used to develop the MATLAB software and the ANSYS tests.

The implications for practice include benefits, which combined could improve not only the productivity, consistency, clarity, accuracy, and quality of the cold gas dynamic spray process itself, but also improve various related activities such as production lead-time, production scheduling, and capacity utilization.

5.5 Research Limitations

The dissertation has aimed to:

- develop a MATLAB software capable to generate practical graphs for the CGDS process and generate the 2D recommended nozzle contour,
- use the Computational Fluid Dynamics (CFD) method to calculate and visualize the gas flow, and

- test the two developed technologies using data from peer reviewed journal papers.

The research was focused and its aims achieved. However, the dissertation had a number of limitations acknowledged by the author. These limitations, that do not detract the significance of the dissertation' findings, refer to: the De Laval nozzle profile that was limited to a straight lines profile from the throat section to the nozzle exit section; the design of the MOC nozzle was limited to the determination of the internal shape for the divergent part of the supersonic nozzle; the external profile of the nozzle design and its manufacturing were excluded; and the use of ANSYS Fluent software was limited only to the determination of the gas velocity.

5.6 Further Research

This final section is written to help students and other researchers in selection and design of future research directions that could be foreseen.

Further research suggestions include but are not limited to: the design and construction of an improved GUI for the MATLAB software; the development of a modular code inside the MATLAB software; and an enhanced ANSYS approach for the cold spray development.

Finally, the research literature suggested that there is a need for a software for CGDS nozzle calculations. This dissertation showed that it is both theoretically and practically possible to design, build, and test such software and also set a foundation for further research.

REFERENCES

[1] Al-Ajlouni M., *An Automatic Method for Creating the Profile of Supersonic Convergent-Divergent Nozzle*, Journal of Mechanical and Industrial Engineering, vol 4 No. 3, 2010, pp. 404–411

[2] Alkhimov A.P., Kosarev V.F. and Klinkov S.V., *The Features of Cold Spray Nozzle Design*, Institute of Theoretical and Applied Mechanics SB RAS, Journal of Thermal Spray Technology Volume 10(2), ASM International,Novosibirsk, 2001, pp 375

[3] Amardip Ghosh,*Supersonic Nozzle Design Using 2D method of characteristics*, Department of Aerospace Engineering, University of Maryland.

[4] Carmichael Ralph, *How do you compute the inverse of the Prandtl-Meyer Function?*, Public Domain Aeronautical Software, Santa Cruz, 2007.

[5] Champagne V. K. et al., *Comparison of Empirical and Theoretical Computations of Velocity for a Cold Spray Nozzle*

[6] Champagne V. K. et al., *Theoretical and Experimental Particle Velocity in Cold Spray*, Journal of Thermal Spray Technology, vol 20 No. 3, 2011, pp. 425–431

[7] Dykhuizen R.C., Smith M.F., *Gas Dynamic Principles of Cold Spray*, ASM International, JTTEE5 7, 1998, pp. 205–212.

[8] Francois R., *Contribution to the development of a Cold Gas Dynamic Spray System (C.G.D.S.) for the realization of nickel coatings*, Sciences des procedes de Ceramiques et Traitements de Surface, Universite de Limoges, These No. 45, 2005

[9] Jodoin B., *Cold Spray Nozzle Mach Number Limitation*, Journal of Thermal Spray Technology, vol 11, No. 4, 2002, pp. 496–507

[10] Gartner F, Schmidt T., Stoltenhoff T. and Kreye H.,*Recent Developments and Potential Applications of Cold Spraying*, Advanced Engineering Materials 8, No. 7, 2006

[11] Gartner F., Stoltenhoff T., Schmidt T. and Kreye H.,*The Cold Spray Process and Its Potential for Industrial Applications*, Journal of Thermal Spray Technology, Vol 15, No. 2, 2006, pp. 223–232

[12] , Ghajar Afshin J., *Engineering Handbook CRC Chap 36, Compressible flow*, Oklahoma State University, CRC Press LLC, 2005.

[13] Grujicic M., Tong C., DeRosset W. S. and Helfritch D., *Flow Analysis and Nozzle-Shape Optimization for the Cold-Gas Dynamic-Spray Process*, Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, Vol 217, 2003, pp. 1603–1613

[14] Grujicic M., Zhao C. L., Tong C., DeRosset W. S. and Helfritch D.,*Analysis of the impact velocity of powder particles in the cold-gas dynamic-spray process*; Materials Science and Engineering A368, Elsevier B.V., 2004,pp. 222–230

[15] Helfritch D. and Champagne V., *A Model Study of Powder Particle Size Effects in Cold Spray Deposition*, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, 2008.

[16] Karimi M. et al., *Numerical Simulation of the Cold Gas Dynamic Spray Process*, Journal of Thermal Spray Technology, ASM International, Vol 15 (4), 2006

[17] Khine T. Naung, Nyunt Soe, *Numerical Approach for Determination of Supersonic Nozzle Flow Pattern*, GMSARN International Conference on Sustainable Development: Issues and Prospects for the GMS, 2008, pp. 12–14

[18] Kim Hyung-Jun, Leeb Chang-Hee , Hwanga Soon-Young, *Superhard nano WC12% Co coating by cold spray deposition*, Materials Science and Engineering A 391, 2005, pp. 243-248

[19] Klassen T. and Kreye H., *The Cold Spray Process and its Optimisation*, Institute of Materials Technology, Helmut-Schmidt-University, 2006.

[20] Klassen Thomas,

http://www.hsu-hh.de/werkstoffkunde/index_P4OTiQPRdTHq9uNv.html

[21] Li Wen-Ya and Li Chang-Jiu, *Optimal Design of Spray Gun Nozzle at a Limited Space*, Journal of Thermal Spray Technology, vol 14 No. 3, 2005, pp. 391–396

[22] Lupoi R. and ONeill W., *An Investigation on Powder Stream in Cold Gas Spray (CGS) Nozzles*, V European Conference on Computational Fluid Dynamics, Lisbon, 2010

[23] Maev, R. and Leshchynsky V., *Introduction to Low Pressure Gas Dynamic Spray*, Physics and Technology, Wiley-VCH, 2008

[24] Pasquale D., Harinck J., Guardone A. and Rebay S., *Geometry Optimization For Quasi-Uniform Flows From Supersonic Nozzles*, European Conference on Computational Fluid Dynamics, Lisbon, 2010

[25] Tabbara et al., *Study on Process Optimization of Cold Gas Spraying*, ASM International, 2010

[26] Tu Jiyuan et al., *Computational Fluid Dynamics. A Pratical Approach* 1st ed., Butterworth-Heinemann, Amsterdam, 2008

[27] Schmidt T., Gartner F., Assadi H., Kreye H., *Development of a Generalized Parameter Window for Cold Spray Deposition*, Acta Materialia 54, Elsevier, 2006, pp. 729-742

[28] Schmidt T., Gartner F., Kreye H., *New Developments in Cold Spray Based on Higher Gas and Particle Temperatures*, Journal of Thermal Spray Technology, vol 15 No. 4, 2006, pp. 488–494

[29] Stoltenhoff T., Kreye H. and Richter H.J., *An Analysis of the Cold Spray Process and Its Coatings*, Journal of Thermal Spray Technology, vol 11, No. 4, 2002, pp. 542–550

[30] Wu J. et al., *Measurement of Particle Velocity and Characterization of Deposition in Aluminium Alloy Kinetic Spraying Process*, Applied Surface Science, vol 252, 2005, pp. 1368–1377

[31] Zebbiche T. and Youbi Z.,*Supersonic Two-Dimentional Minimum Length Nozzle Design at High Temperature*, Emirates Journal for Engineering Research 11 (1), 2006, pp.91–102

BIBLIOGRAPHY

[32] Anatolii P. et al., *Cold Spray Technology*, Technology and Engineering, Elsevier, 2007

[33] Anderson, J. D. Jr., *Computational Fluid Dynamics, The Basics with Applications*, McGraw-Hill, 1995

[34] Anderson, J. D. Jr, *Modern Compressible Flow, With Historical Perspective* 3rd ed., McGraw-Hill, 2003

[35] Ansys, Inc., *Ansys Fluent 12.0 Theory Guide*, Lebanon, 2009

[36] Ansys, Inc., *Ansys Fluent 12.0 Tutorial Guide*, Lebanon, 2009

[37] Brain H. and Daniel T., *Essential Matlab for Engineers and Scientists*, Elsevier, 4th ed., London, 2010

[38] Champagne V. K., *The Cold Spray Materials Deposition Process. Fundamentals and Applications*, Woodhear Publishing in Materials CRC Press LLC, Abington Cambridge, 2007

[39] MathWorks, Inc., *Matlab 7, Creating Graphical User Interfaces*, United States, 2007

[40] Oosthuizen H. Patrick and Carscallen E. William, *Compressible Fluid Flow* , McGraw-Hill, 1997

[41] Patric F. Dunn, *Chap 9: Compressible Flow*

[42] Shapiro H. and Moran M., *Fundamentals of Engineering Thermodynamics*, John Wiley and Sons Ltd, England, 2006

APPENDICES

Appendix A - Glossary

- Nozzle – a device designed to control the characteristics of a fluid flow as its exit(or enters) an enclosed chamber or pipe.
- Supersonic Nozzle – a convergent–divergent nozzle designed to increase the velocity of gas to values higher than the speed of sound.
- De Laval nozzle – a nozzle that as the longitudinal section of the divergent part represented by a straight line that connects the throat and the exit areas.
- MOC nozzle – a nozzle that has its divergent section calculated using the Method of Characteristic.
- Cold Gas Dynamics Spray – a high-rate material deposition process in which small, unmelted powder particles ranging from $1\,to\,50\mu$m in diameter are accelerated to velocities on the order of $600\,to\,1000\,m/s$ in a supersonic jet of compressed gas. Upon impact with a substrate surface, the solid particles deform and bond together, building up a layer of deposited material realizing a recharge or a coating.
- Critical Velocity – the lowest impact velocity for a particle of a specific material to be deposited.
- Compressible flow – a fluid in which the fluid density varies.
- Isentropic flow – an adiabatic flow (no heat transfer) which is frictionless (ideal or reversible), and the entropy is constant.
- Shockwave – a fully developed compression wave of large amplitude, across which density, pressure, temperature, and particle velocity change

drastically. A shock wave is, in general, curved. However, many shock waves that occur in practical situations are straight, being either at right angles to the flow path (termed a normal shock) or at an angle to the flow path (termed an oblique shock).

- Mach number – a dimensionless measure of compressibility, defined as the ratio of the local flow velocity and the local speed of sound.
- Inviscid gas – the flow of an ideal fluid that is assumed to have no viscosity.
- Ideal gas – theoretical gas composed of a set of randomly moving, non-interacting point particles and that obey to the ideal gas law.
- non – rotational gas flow – the flow of a fluid in which the curl of the fluid velocity is equal to zero.
- gas inert – a gas which does not undergo chemical reactions under a set of given conditions.
- non–linear least squares procedure – the form of least squares analysis used to fit a set of observations with a model that is non–linear.
- non-linear polynomial regression – form of linear regression in which the relationship between the independent variable x and the dependent variable y is modeled as a 'n' polynomial function.

Appendix B – CGDS operating parameters according to MIL-STD-3021

	High Pressure Powder Injection System	Low Pressure Powder Injection System
Working gas	N_2, He, air	N_2, He, air
Gas pressure, MPa	0.25 – 5.0	0.5 – 1.7
Gas preheat, °C	20 - 1000	20 - 600
Gas flow rate, m^3/hour	2 - 200	15 - 78
Maximum Gas Mach #	1 - 3	1 - 3
Powder flow rate, g/s	0.1 – 2.0	0.1 – 1.0
Particle size, µm	5 - 100	5 - 50

Table 5.1: Operating parameters

Appendix C - Matlab Code for De Laval nozzle

Appendix C1 - Code for GUI construction

```
function varargout = DeLavalNozzle(varargin)
% DELAVALNOZZLE M-file for DeLavalNozzle.fig
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @DeLavalNozzle_OpeningFcn, ...
                   'gui_OutputFcn',  @DeLavalNozzle_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before DeLavalNozzle is made visible.
function DeLavalNozzle_OpeningFcn(hObject, eventdata, handles, varargin)

%Initilizing of particle properties
handles.SigmaTS = 0; %data structure for storing tensible strength
handles.rhop = 0; %data structure for storing particle velocity
handles.Cpp = 0; %data structure for storing specific heat of particle
handles.Tm = 0; %data structure for storing particle melting temperature
handles.F1 = 0; %data structure for storing the mechanical calibration
                %factor
handles.F2 = 0; %data structure for storing the thermal calibration factor

%Initializing gas properties
```

```
handles.rhog = 0; %data structure for storing gas density
handles.R = 0; %data structure for storing gas constant
handles.gam = 0; %data structure for storing specific heat ratio

% Choose default command line output for DeLavalNozzle
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
function varargout = DeLavalNozzle_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;
function T0_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function T0_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in Compute.
function Compute_Callback(hObject, eventdata, handles)
%-------------------------INPUT DATA-----------------------------------
%-------------------------Data Gas from m.files------------------------
%gam, specific heat ratio
%R, universal gas constant [J/kg K]
%-------------------------Data Particles from m.files------------------
%SigmaTS, storing tensible strength [Pa]
%rhop, particle velocity [kg/s]
%vp, particle velocity [kg/m^3]
%Cpp, specific heat of particle [J/kg K]
%Tm, particle melting temperature [K]
%F1, mechanical calibration factor
%F2, thermal calibration factor
%-------------------------Atmospheric Conditions-----------------------
AtmP = str2num(get(handles.AtmP,'String')); %atmospheric pressure [MPa]
```

```
AtmT = str2num(get(handles.AtmT,'String')); % atmosheric temperature [K]
%------------------------Data nozzle form interface---------------------
dstar = str2num(get(handles.dstar,'String')); %throat diameter
de = str2num(get(handles.de,'String')); %exit diameter
x = str2num(get(handles.x,'String')); %length divergent part
%------------------------Data gas from interface-----------------------
T0 = str2num(get(handles.T0,'String')); %stagnation temperature [K]
P0 = str2num(get(handles.P0,'String')); %stagnation pressure [MPa]
%------------------------Data Particles form interface-----------------
Dp = str2num(get(handles.Dp,'String')); %particle diameter [Micron]
Ti = str2num(get(handles.Ti,'String')); %Impact temperature [293.15K]
Tih = str2num(get(handles.Tih,'String')); %Impact temperature [373.15K]
%-------------------------CALCULATIONS---------------------------------
Astar = (3.14*dstar^2)/4; %throat area [mm^2];
Ae = (3.14*de^2)/4; %exit area [mm^2];
Tstar = T0 / (1+(handles.gam-1)/2); %temperature at throat [K]
Vstar = (handles.gam*handles.R*Tstar)^(1/2); %velocity at throat [m/s]
rho0 = (P0*10^6)/(handles.R*T0); %stagnation density [kg/m^3]
rhostar = rho0*(2/(handles.gam+1))^(1/(handles.gam-1));...
    %sonic gas density [kg/m^3]
Pstar = (rhostar * handles.R * Tstar)/10^6; %pressure at throat [MPa]
mvdot =(2/(handles.gam+1))^(1/(handles.gam-1))*Vstar*Astar*10^(-6)*3600;...
    %gas flow rate [m^3/hour]
AreaRatio = Ae / Astar; %area ratio
k1 = 218.0629 - 243.5764*handles.gam + 71.7925*handles.gam^2; %constant
k2 = -0.122450 + 0.281300*handles.gam; %constant
Me = (k1*AreaRatio + (1-k1))^k2; %Mach number from Area ratio
Pe = Pstar*((handles.gam+1)/(2+(handles.gam-1)*Me^2))^(handles.gam/...
     (handles.gam-1)); %pressure at exit [MPa]
Te = T0 / (1+((handles.gam-1)/2)*Me^2); %temperature at exit [K]
Ve = Me * (handles.gam*handles.R*Te)^0.5; %velocity of gas at exit [m/s]
rhoe = (rhostar*((handles.gam+1)/2)^(1/(handles.gam-1))) /...
     ((1+((handles.gam-1)/2)*Me^2)^(1/(handles.gam-1)));...
     %exit gas density [kg/m^3]
Vp = Ve / (1+0.85* sqrt(Dp/(x*10^3)) * sqrt(handles.rhop*Ve^2/(P0*10^6)));
TR = AtmT; %reference temperature [K]
```

```
VcritL =sqrt((handles.F1*4*handles.SigmaTS*(1-(Ti-TR)/(handles.Tm-TR)))...
        /handles.rhop + handles.F2*handles.Cpp*(handles.Tm - Ti));...
        %critical velocity [m/s]
VcritH =sqrt((handles.F1*4*handles.SigmaTS*(1-(Tih-TR)/(handles.Tm-TR)))...
        /handles.rhop + handles.F2*handles.Cpp*(handles.Tm - Tih));...
        %critical velocity [m/s]
Ps = Pe*((((2*handles.gam)/(handles.gam+1))*(Me^2))-((handles.gam-1)/...
    (handles.gam+1))); %shock pressure [MPa]
%---------------------------OUTPUT RESULTS-------------------------------
set(handles.mvdot,'String',mvdot);
set(handles.Tstar,'String',Tstar);
set(handles.Vstar,'String',Vstar);
set(handles.rhostar,'String',rhostar);
set(handles.Pstar,'String',Pstar);
set(handles.AreaRatio,'String',AreaRatio);
set(handles.Me,'String',Me);
set(handles.Pe,'String',Pe);
set(handles.Te,'String',Te);
set(handles.Ve,'String',Ve);
set(handles.rhoe,'String',rhoe);
set(handles.Ps,'String',Ps);
set(handles.Vp,'String',Vp);
set(handles.VcritL,'String',VcritL);
set(handles.VcritH,'String',VcritH);

%---------------------Plot Particle Velocity----------------------------
%var(), mean variation of()
Varx = 0:.01:x;
%VarVp = Ve .* ((3.*1.*rhoe.*Varx.*10.^(-3))./...
%(2.*Dp.*10^(-6).*handles.rhop)).^0.5;
VarVp = Ve ./ (1.+0.85.*(Dp./(Varx.*10.^3)).^0.5.*(handles.rhop.*Ve.^2./...
        (P0.*10.^6)).^0.5);
axes(handles.PlotAxe1);
plot(Varx,VarVp,'r','LineWidth',2);
xlabel('Divergent Length (mm) vs Particle Velocity (m/s)');
grid
```

```
%---------------Plot Pressure and Temperature superimposed---------------
VarMe = [1:.01:Me];
VarPe = Pstar.*((handles.gam+1)./(2.+(handles.gam-1).*VarMe.^2)).^...
        (handles.gam ./(handles.gam-1));
VarTe = T0 ./ (1.+((handles.gam-1)/2).*VarMe.^2);
axes(handles.PlotAxe2);
plotyy(VarMe,VarPe,VarMe,VarTe),...
title('Mach number vs Pressure(blue) and Mach number vs Temperature(gren)')
xlabel('Mach number'),
grid
function dstar_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function dstar_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function de_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function de_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function x_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function x_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end
function Dp_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function Dp_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function Ti_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function Ti_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on selection change in GasData.
function GasData_Callback(hObject, eventdata, handles)
val=get(hObject,'Value');
str=get(hObject,'String');
switch str{val}
    case 'Helium'
        Helium_Properties;
        handles.R = R;
        handles.gam = gam;

    case 'Nitrogen'
        Nitrogen_Properties;
        handles.R = R;
        handles.gam = gam;
end
guidata(hObject,handles);
% --- Executes during object creation, after setting all properties.
```

```
function GasData_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on selection change in PartData.
function PartData_Callback(hObject, eventdata, handles)
val=get(hObject,'Value');
str=get(hObject,'String');
switch str{val}
    case 'Aluminium'
        Aluminium_Properties;
        handles.SigmaTS = SigmaTS;
        handles.rhop = rhop;
        handles.Cpp = Cpp;
        handles.Tm = Tm;
        handles.F1 = F1;
        handles.F2 = F2;

     case 'Copper'
        Copper_Properties;
        handles.SigmaTS = SigmaTS;
        handles.rhop = rhop;
        handles.Cpp = Cpp;
        handles.Tm = Tm;
        handles.F1 = F1;
        handles.F2 = F2;

 case 'Nickel'
        Nickel_Properties;
       handles.SigmaTS = SigmaTS;
        handles.rhop = rhop;
        handles.Cpp = Cpp;
        handles.Tm = Tm;
        handles.F1 = F1;
```

```
        handles.F2 = F2;

    case 'Steel316L'
        Steel316L_Properties;
        handles.SigmaTS = SigmaTS;
        handles.rhop = rhop;
        handles.Cpp = Cpp;
        handles.Tm = Tm;
        handles.F1 = F1;
        handles.F2 = F2;

    case 'Titanium'
        Titanium_Properties;
        handles.SigmaTS = SigmaTS;
        handles.rhop = rhop;
        handles.Cpp = Cpp;
        handles.Tm = Tm;
        handles.F1 = F1;
        handles.F2 = F2;
end
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function PartData_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes during object creation, after setting all properties.
function Compute_CreateFcn(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function PlotAxe1_CreateFcn(hObject, eventdata, handles)

% --- Executes on button press in Exit.
```

```
function Exit_Callback(hObject, eventdata, handles)
% hObject    handle to Exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.figure1)
% --- Executes during object creation, after setting all properties.
function Exit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes on button press in Specification.
function Specifications_Callback(hObject, eventdata, handles)
% hObject    handle to Specification (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('Specifications.pdf')

function P0_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function P0_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function AtmP_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function AtmP_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end


function AtmT_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function AtmT_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function Tih_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function Tih_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Appendix C2 - Code for the conducted tests

Appendix C21 - Code for Test 1

```
%------------------------ CODE for TEST 1 -------------------------
clc; % Clean screen
clear all; % Clean variables in work space
% 'Ref' refers to data from reference used
% DiaPart, set of particle diameters
% PaVeRef, set of particle velocities
% FitRef, Interpolation polynome
% ValFitRef, valeurs of Interpolation Polynome
% ErrorIntRef, error between interpolation and original curve
% 'Gui' refers to data from this work
% PaVeGui, set of particle velocities
% FitGui, Interpolation polynome
% ValFitGui, valeurs of Interpolation Polynome
% ErrorIntGui, error between interpolation and original curve
% MaxDif, maximum difference of Velocity between Reference and This Work

DiaPart = input('DiaPart =');
PaVeRef = input('PaVeRef =');
PaVeGui = input('PaVeGui =');
plot (DiaPart,PaVeRef,'b-*','LineWidth',1);
hold on
plot (DiaPart,PaVeGui,'r-*','LineWidth',1);
grid on
axis([0, 55, 200, 800])
title('Comparison Particle Velocity');
xlabel('Particle diameter (micron)');
ylabel('Particle Velocity (m/s)');

disp('REFERENCE Interpolation Polynome_Velocity:')
FitRef = polyfit(DiaPart,PaVeRef,4)
ValFitRef = polyval(FitRef,DiaPart);
ErrorIntRef =PaVeRef-ValFitRef;
disp('REFERENCE Variance of Interpolation error_Velocity:')
```

```
num2str(std(ErrorIntRef).^2)

disp('THIS WORK Interpolation polynome_Velocity:')
FitGui = polyfit(DiaPart,PaVeGui,4)
ValFitGui = polyval(FitGui,DiaPart);
ErrorIntGui = PaVeGui - ValFitGui;
disp('THIS WORK Variance of Interpolation error_Velocity is:')
num2str(std(ErrorIntGui).^2)
disp('Maximum difference_Velocity REFERENCE-THIS WORK')
MaxDif = max(ValFitGui - ValFitRef)
```

Appendix C22 - Code for Test 2–Variation of pressure

```
%------------------------ CODE for TEST 2 Pressure -----------------
clc; % Clean screen
clear all; % Clean variables in work space
% 'Ref' refers to data from reference used
% PrePart, set of particle pressure
% PaVeRef, set of particle velocities
% FitRef, Interpolation polynome
% ValFitRef, valeurs of Interpolation Polynome
% ErrorIntRef, error between interpolation and original curve
% 'Gui' refers to data from this work
% PaVeGui, set of particle velocities
% FitGui, Interpolation polynome
% ValFitGui, valeurs of Interpolation Polynome
% ErrorIntGui, error between interpolation and original curve
%Dif, the difference vector between Reference and This Work
% MeanDif, the average of difference of Velocity between Reference
% and This Work

PrePart = input('PrePart =');
PaVeRef = input('PaVeRef =');
PaVeGui = input('PaVeGui =');
plot (PrePart,PaVeRef,'b-*','LineWidth',1);
hold on
plot (PrePart,PaVeGui,'r-*','LineWidth',1);
grid on
axis([1.25, 3.65, 300, 700])
title('Comparison Particle Velocity Reference-This Work');
xlabel('Particle Pressure (MPa)');
ylabel('Particle Velocity (m/s)');

disp('REFERENCE Interpolation Polynome_Velocity:')
FitRef = polyfit(PrePart,PaVeRef,4)
ValFitRef = polyval(FitRef,PrePart);
ErrorIntRef =PaVeRef-ValFitRef;
```

```
disp('REFERENCE Variance of Interpolation error_Velocity:')
num2str(std(ErrorIntRef).^2)

disp('THIS WORK Interpolation polynome_Velocity:')
FitGui = polyfit(PrePart,PaVeGui,4)
ValFitGui = polyval(FitGui,PrePart);
ErrorIntGui = PaVeGui - ValFitGui;
disp('THIS WORK Variance of Interpolation error_Velocity is:')
num2str(std(ErrorIntGui).^2)
Dif = (ValFitGui - ValFitRef)
disp('Mean of difference_Velocity REFERENCE-THIS WORK')
MeanDif = mean(Dif)
```

Appendix C23 - Code for Test 2–Variation of the temperature

```
%------------------------- CODE for TEST 2 Temperature --------------
clc; % Clean screen
clear all; % Clean variables in work space
% 'Ref' refers to data from reference used
% TePart, set of particle temperature
% PaVeRef, set of particle velocities
% FitRef, Interpolation polynome
% ValFitRef, valeurs of Interpolation Polynome
% ErrorIntRef, error between interpolation and original curve
% 'Gui' refers to data from this work
% PaVeGui, set of particle velocities
% FitGui, Interpolation polynome
% ValFitGui, valeurs of Interpolation Polynome
% ErrorIntGui, error between interpolation and original curve
%Dif, the difference vector between Reference and This Work
% MeanDif, the average of difference of Velocity between Reference
% and This Work

TePart = input('TePart =');
PaVeRef = input('PaVeRef =');
PaVeGui = input('PaVeGui =');
plot (TePart,PaVeRef,'b-*','LineWidth',1);
hold on
plot (TePart,PaVeGui,'r-*','LineWidth',1);
grid on
axis([273, 813, 300, 700])
title('Comparison Particle Velocity Reference-This Work');
xlabel('Particle Temperature (K)');
ylabel('Particle Velocity (m/s)');

disp('REFERENCE Interpolation Polynome_Velocity:')
FitRef = polyfit(TePart,PaVeRef,2)
ValFitRef = polyval(FitRef,TePart);
ErrorIntRef =PaVeRef-ValFitRef;
```

```
disp('REFERENCE Variance of Interpolation error_Velocity:')
num2str(std(ErrorIntRef).^2)

disp('THIS WORK Interpolation polynome_Velocity:')
FitGui = polyfit(TePart,PaVeGui,2)
ValFitGui = polyval(FitGui,TePart);
ErrorIntGui = PaVeGui - ValFitGui;
disp('THIS WORK Variance of Interpolation error_Velocity is:')
num2str(std(ErrorIntGui).^2)
Dif = (ValFitGui - ValFitRef)
disp('Mean of difference_Velocity REFERENCE-THIS WORK')
MeanDif = mean(Dif)
```

Appendix D - Matlab Code for MOC Nozzle

Appendix D1 - Code to the curved part of the MOC nozzle

```
%PLOTING OF CONTOUR FOR SUPERSONIC FLOW
%Me the Mach number at exit
%gam ratio of gas specific heats
%n number of characteristic lines, denotes the precision of the contour
%form
%r the radius at the throat
%nuMe PRANDTL MEYER function corresponding to the design exit Mach number
%thethaWmax the max angle of the duct wall with respect to the x direction
%varthetha angle between two characteristic lines
%knegx represents K(-) and kposx represents K(+), PRANDTL MEYER constants
%on oblique line x
%nux PRANDTL MEYER angles on oblique line x
%thethax the angle of duct wall on oblique line x
%Mx Mach Number on oblique line x
%mux the local Mach angle on oblique line x
%xi coordinates of the points on the contour with respect to the x axis
%yi coordinates of the points on the contour with respect to the y axis
%shapelinex a line on the contour starting from the point x
%charlinex characteristic line passing through point x
%sloblinex the slope of oblique line from characteristic line x
%oblinex oblique line from characteristic line x

%INPUT CONSTANTS
Me = input('Me = ');
gam = input('gam = ');
n = input('n = ');
r = input('r = ');
nuMe = (sqrt((gam + 1) / (gam - 1))) * ((atan(sqrt(((gam - 1)./...
       (gam + 1)) * (Me^2 - 1))))) - atan(sqrt(Me^2 - 1));
thethaWmax = nuMe / 2
varthetha = thethaWmax / n

%DETERMINATION OF PRANDTL MEYER CONSTANTS
```

```
kneg1 = [varthetha*2:varthetha*2:nuMe];
kpos1 = zeros(1,n);
kneg2 = [kneg1(2):varthetha*2:nuMe];
kpos2 = - kneg1(2).*ones(1,n-1);
kneg3 = [kneg1(3):varthetha * 2:nuMe];
kpos3 = - kneg1(3) .* ones(1,n-2);
kneg4 = [kneg1(4):varthetha * 2:nuMe];
kpos4 = - kneg1(4) .* ones(1,n-3);
kneg5 = [kneg1(5):varthetha * 2:nuMe];
kpos5 = - kneg1(5) .* ones(1,n-4);
kneg6 = [kneg1(6):varthetha * 2:nuMe];
kpos6 = - kneg1(6) .* ones(1,n-5);
kneg7 = [kneg1(7):varthetha * 2:nuMe];
kpos7 = - kneg1(7) .* ones(1,n-6);
kneg8 = [kneg1(8):varthetha * 2:nuMe];
kpos8 = - kneg1(8) .* ones(1,n-7);
kneg9 = [kneg1(9):varthetha * 2:nuMe];
kpos9 = - kneg1(9) .* ones(1,n-8);
kneg10 = [kneg1(10):varthetha * 2:nuMe];
kpos10 = - kneg1(10) .* ones(1,n-9);


%DETERMINATION OF PRANDTL MEYER FUNCTION
nu1 = (kneg1 - kpos1)/2;
nu2 =(kneg2 - kpos2)/2;
nu3 = (kneg3 - kpos3)/2;
nu4 = (kneg4 - kpos4)/2;
nu5 = (kneg5 - kpos5)/2;
nu6 = (kneg6 - kpos6)/2;
nu7 = (kneg7 - kpos7)/2;
nu8 = (kneg8 - kpos8)/2;
nu9 = (kneg9 - kpos9)/2;
nu10 = (kneg10 - kpos10)/2;

%DETERMINATION OF PRANDTL MEYER ANGLE
thetha1 =(kneg1 + kpos1)/2;
```

```
thetha2 =(kneg2 + kpos2)/2;
thetha3 = (kneg3 + kpos3)/2;
thetha4 = (kneg4 + kpos4)/2;
thetha5 = (kneg5 + kpos5)/2;
thetha6 = (kneg6 + kpos6)/2;
thetha7 = (kneg7 + kpos7)/2;
thetha8 = (kneg8 + kpos8)/2;
thetha9 = (kneg9 + kpos9)/2;
thetha10 = (kneg10 + kpos10)/2;

%DETERMINATION OF MACH NUMBER
M1 = (1 + 0.7863*nu1.^0.66667 + 0.03214*nu1.^1.33333 - 0.099*nu1.^2)./...
     (1 - 0.38853*nu1.^0.66667 - 0.10951*nu1.^1.33333);
M2 = (1 + 0.7863*nu2.^0.66667 + 0.03214*nu2.^1.33333 - 0.099*nu2.^2)./...
     (1 - 0.38853*nu2.^0.66667 - 0.10951*nu2.^1.33333);
M3 = (1 + 0.7863*nu3.^0.66667 + 0.03214*nu3.^1.33333 - 0.099*nu3.^2)./...
     (1 - 0.38853*nu3.^0.66667 - 0.10951*nu3.^1.33333);
M4 = (1 + 0.7863*nu4.^0.66667 + 0.03214*nu4.^1.33333 - 0.099*nu4.^2)./...
     (1 - 0.38853*nu4.^0.66667 - 0.10951*nu4.^1.33333);
M5 = (1 + 0.7863*nu5.^0.66667 + 0.03214*nu5.^1.33333 - 0.099*nu5.^2)./...
     (1 - 0.38853*nu5.^0.66667 - 0.10951*nu5.^1.33333);
M6 = (1 + 0.7863*nu6.^0.66667 + 0.03214*nu6.^1.33333 - 0.099*nu6.^2)./...
     (1 - 0.38853*nu6.^0.66667 - 0.10951*nu6.^1.33333);
M7 = (1 + 0.7863*nu7.^0.66667 + 0.03214*nu7.^1.33333 - 0.099*nu7.^2)./...
     (1 - 0.38853*nu7.^0.66667 - 0.10951*nu7.^1.33333);
M8 = (1 + 0.7863*nu8.^0.66667 + 0.03214*nu8.^1.33333 - 0.099*nu8.^2)./...
     (1 - 0.38853*nu8.^0.66667 - 0.10951*nu8.^1.33333);
M9 = (1 + 0.7863*nu9.^0.66667 + 0.03214*nu9.^1.33333 - 0.099*nu9.^2)./...
     (1 - 0.38853*nu9.^0.66667 - 0.10951*nu9.^1.33333);
M10=(1 + 0.7863*nu10.^0.66667 + 0.03214*nu10.^1.33333 - 0.099*nu10.^2)./...
      (1 - 0.38853*nu10.^0.66667 - 0.10951*nu10.^1.33333);

%DETERMINATION OF MACH ANGLE
mu1 = asin(1 ./ M1);
mu2 = asin(1 ./ M2);
mu3 = asin(1 ./ M3);
```

```
mu4 = asin(1 ./ M4);
mu5 = asin(1 ./ M5);
mu6 = asin(1 ./ M6);
mu7 = asin(1 ./ M7);
mu8 = asin(1 ./ M8);
mu9 = asin(1 ./ M9);
mu10 = asin(1 ./ M10);

x = [0:0.01:100];

%START POINT
x0 = 0;
y0 = r;

%DETERMINATION OF COORDINATES OF POINT 1 ON THE CONTOUR
shapeline1 = x * tan(thethaWmax) + y0;
charline1 = - cot(thetha1(1)) * x + y0;
slobline1 = thetha1(10) + mu1(10);
obline1 = slobline1 * x - r * slobline1 / cot(thetha1(1));
x1 = ((-r * slobline1 / cot(thetha1(1))) - y0) /...
    (tan(thethaWmax) - slobline1);
y1 = slobline1 * x1 - r * slobline1 / cot(thetha1(1));

%DETERMINATION OF COORDINATES OF POINT 2 ON THE CONTOUR
shapeline2 = x * tan((thetha1(10)+thetha1(8))/2) - x1 * ...
             tan((thetha1(10)+thetha1(8))/2) + y1;
charline2 = - cot(thetha1(2)) * x + r;
slobline2 = thetha2(9) + mu2(9);
obline2 = slobline2 * x - r * slobline2 / cot(thetha1(2));
x2 = ((-r * slobline2 / cot(thetha1(2))) - (- x1 * tan((thetha1(10)+...
    thetha1(8))/2) + y1))/((tan((thetha1(10)+thetha1(8))/2)) - slobline2);
y2 = slobline2 * x2 - r * slobline2 / cot(thetha1(2));

%DETERMINATION OF COORDINATES OF POINT 3 ON THE CONTOUR
shapeline3 = x * tan((thetha1(8)+thetha1(7))/2) - x2 * ...
             tan((thetha1(8)+thetha1(7))/2) + y2;
```

```
charline3 = - cot(thetha1(3)) * x + r;
slobline3 = thetha3(8) + mu3(8);
obline3 = slobline3 * x - r * slobline3 / cot(thetha1(3));
x3 = ((-r * slobline3 / cot(thetha1(3))) - ( - x2 * tan((thetha1(8)+...
    thetha1(7))/2) + y2))/((tan((thetha1(8)+thetha1(7))/2)) - slobline3);
y3 = slobline3 * x3 - r * slobline3 / cot(thetha1(3));

%DETERMINATION OF COORDINATES OF POINT 4 ON THE CONTOUR
shapeline4 = x * tan((thetha1(7)+thetha1(6))/2) - x3 * ...
             tan((thetha1(7)+thetha1(6))/2) + y3;
charline4 = - cot(thetha1(4)) * x + r;
slobline4 = thetha4(7) + mu4(7);
obline4 = slobline4 * x - r * slobline4 / cot(thetha1(4));
x4 = ((-r * slobline4 / cot(thetha1(4))) - ( - x3 * tan((thetha1(7)+...
    thetha1(6))/2) + y3))/((tan((thetha1(7)+thetha1(6))/2)) - slobline4);
y4 = slobline4 * x4 - r * slobline4 / cot(thetha1(4));

%DETERMINATION OF COORDINATES OF POINT 5 ON THE CONTOUR
shapeline5 = x * tan((thetha1(6)+thetha1(5))/2) - x4 * ...
             tan((thetha1(6)+thetha1(5))/2) + y4;
charline5 = - cot(thetha1(5)) * x + r;
slobline5 = thetha5(6) + mu5(6);
obline5 = slobline5 * x - r * slobline5 / cot(thetha1(5));
x5 = ((-r * slobline5 / cot(thetha1(5))) - ( - x4 * tan((thetha1(6)+...
    thetha1(5))/2) + y4))/((tan((thetha1(6)+thetha1(5))/2)) - slobline5);
y5 = slobline5 * x5 - r * slobline5 / cot(thetha1(5));

%DETERMINATION OF COORDINATES OF POINT 6 ON THE CONTOUR
shapeline6 = x * tan((thetha1(5)+thetha1(4))/2) - x5 * ...
             tan((thetha1(5)+thetha1(4))/2) + y5;
charline6 = - cot(thetha1(6)) * x + r;
slobline6 = thetha6(5) + mu6(5);
obline6 = slobline6 * x - r * slobline6 / cot(thetha1(6));
x6 = ((-r * slobline6 / cot(thetha1(6))) - ( - x5 * tan((thetha1(5)+...
    thetha1(4))/2) + y5))/((tan((thetha1(5)+thetha1(4))/2)) - slobline6);
y6 = slobline6 * x6 - r * slobline6 / cot(thetha1(6));
```

```
%DETERMINATION OF COORDINATES OF POINT 7 ON THE CONTOUR
shapeline7 = x * tan((thetha1(4)+thetha1(3))/2) - x6 * ...
             tan((thetha1(4)+thetha1(3))/2) + y6;
charline7 = - cot(thetha1(7)) * x + r;
slobline7 = thetha7(4) + mu7(4);
obline7 = slobline7 * x - r * slobline7 / cot(thetha1(7));
x7 = ((-r * slobline7 / cot(thetha1(7))) - ( - x6 * tan((thetha1(4)+...
    thetha1(3))/2) + y6))/((tan((thetha1(4)+thetha1(3))/2)) - slobline7);
y7 = slobline7 * x7 - r * slobline7 / cot(thetha1(7));

%DETERMINATION OF COORDINATES OF POINT 8 ON THE CONTOUR
shapeline8 = x * tan((thetha1(3)+thetha1(2))/2) - x7 * ...
             tan((thetha1(3)+thetha1(2))/2) + y7;
charline8 = - cot(thetha1(8)) * x + r;
slobline8 = thetha8(3) + mu8(3);
obline8 = slobline8 * x - r * slobline8 / cot(thetha1(8));
x8 = ((-r * slobline8 / cot(thetha1(8))) - ( - x7 * tan((thetha1(3)+...
    thetha1(2))/2) + y7))/((tan((thetha1(3)+thetha1(2))/2)) - slobline8);
y8 = slobline8 * x8 - r * slobline8 / cot(thetha1(8));

%DETERMINATION OF COORDINATES OF POINT 9 ON THE CONTOUR
shapeline9 = x * tan((thetha1(2)+thetha1(1))/2) - x8 * ...
             tan((thetha1(2)+thetha1(1))/2) + y8;
charline9 = - cot(thetha1(9)) * x + r;
slobline9 = thetha9(2) + mu9(2);
obline9 = slobline9 * x - r * slobline9 / cot(thetha1(9));
x9 = ((-r * slobline9 / cot(thetha1(9))) - ( - x8 * tan((thetha1(2)+...
    thetha1(1))/2) + y8))/((tan((thetha1(2)+thetha1(1))/2)) - slobline9);
y9 = slobline9 * x9 - r * slobline9 / cot(thetha1(9));

%DETERMINATION OF COORDINATES OF POINT 10 ON THE CONTOUR
shapeline10 = x * tan(thetha1(1)/2) - x9 * tan(thetha1(1)/2) + y9;
charline10 = - cot(thetha1(10)) * x + r;
slobline10 = thetha10(1) + mu10(1);
obline10 = slobline10 * x - r * slobline10 / cot(thetha1(10));
```

```
x10 = ((-r * slobline10 / cot(thetha1(10))) - (- x9 * tan(thetha1(1)/2)+...
    y9))/((tan(thetha1(1)/2)) - slobline10);
y10 = slobline10 * x10 - r * slobline10 / cot(thetha1(10));

%PLOT
plot(x,shapeline1,'g-',x,charline1,'r-',x,obline1,'b-',...
x,shapeline2,'g-',x,charline2,'r-',x,obline2,'b-',...
x,shapeline3,'g-',x,charline3,'r-',x,obline3,'b-',...
x,shapeline4,'g-',x,charline4,'r-',x,obline4,'b-',...
x,shapeline5,'g-',x,charline5,'r-',x,obline5,'b-',...
x,shapeline6,'g-',x,charline6,'r-',x,obline6,'b-',...
x,shapeline7,'g-',x,charline7,'r-',x,obline7,'b-',...
x,shapeline8,'g-',x,charline8,'r-',x,obline8,'b-',...
x,shapeline9,'g-',x,charline9,'r-',x,obline9,'b-',...
x,shapeline10,'g-',x,charline10,'r-',x,obline10,'b-')

disp(['     K(-)         K(+)  THETHA      NU        M         MU'])
disp([kneg1',kpos1',thetha1',nu1',M1',mu1'])
disp([kneg2',kpos2',thetha2',nu2',M2',mu2'])
disp([kneg3',kpos3',thetha3',nu3',M3',mu3'])
disp([kneg4',kpos4',thetha4',nu4',M4',mu4'])
disp([kneg5',kpos5',thetha5',nu5',M5',mu5'])
disp([kneg6',kpos6',thetha6',nu6',M6',mu6'])
disp([kneg7',kpos7',thetha7',nu7',M7',mu7'])
disp([kneg8',kpos8',thetha8',nu8',M8',mu8'])
disp([kneg9',kpos9',thetha9',nu9',M9',mu9'])
disp([kneg10',kpos10',thetha10',nu10',M10',mu10'])

xp = [x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 x10];
yp = [y0 y1 y2 y3 y4 y5 y6 y7 y8 y9 y10];
hold on
plot(xp,yp,'k','LineWidth',3)
grid on
axis equal
axis([0 x10 0 y10])
```

Appendix D2 - Code GUI for MOC nozzle

```
function varargout = MocNozzle(varargin)
% MOCNOZZLE M-file for MocNozzle.fig
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MocNozzle_OpeningFcn, ...
                   'gui_OutputFcn',  @MocNozzle_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before MocNozzle is made visible.
function MocNozzle_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = MocNozzle_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

function MachNumber_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes during object creation, after setting all properties.
function MachNumber_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function SpecificHeat_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function SpecificHeat_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function DiameterThroat_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function DiameterThroat_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function DivLength_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function DivLength_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

function edit9_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit10_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit11_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit12_Callback(hObject, eventdata, handles)
```

```
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit13_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit14_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit15_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
```

```
function edit15_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit16_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit17_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit18_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
```

```
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit19_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit20_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit21_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end

function edit22_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit23_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit24_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit24_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit25_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function edit25_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in PlotMocNozzle.
function PlotMocNozzle_Callback(hObject, eventdata, handles)
% hObject    handle to PlotMocNozzle (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%-----------------------------PLOT CODE ----------------------------------

%GENERATION OF SHAPE FOR SUPERSONIC FLOW IN COLD GAS DYNAMIC SPRAYING
%Me the Mach number at exit
%gam ratio of gas specific heats
%n number of characteristic lines, denotes the precision of the contour
%form
%r the radius at the throat
%nuMe PRANDTL MEYER function corresponding to the design exit Mach number
%thethaWmax the max angle of the duct wall with respect to the x direction
%varthetha angle between two characteristic lines
%knegx represents K(-) and kposx represents K(+), PRANDTL MEYER constants
%on oblique line x
%nux PRANDTL MEYER angles on oblique line x
%thethax the angle of duct wall on oblique line x
%Mx Mach Number on oblique line x
%mux the local Mach angle on oblique line x
%xi coordinates of the points on the contour with respect to the x axis
%yi coordinates of the points on the contour with respect to the y axis
```

```
%shapelinex a line on the contour starting from the point x
%charlinex characteristic line passing through point x
%sloblinex the slope of oblique line from characteristic line x
%oblinex oblique line from characteristic line x
%L length of divergent part

%INPUT CONSTANTS
n = 10;
Me = str2num(get(handles.MachNumber,'String'));
gam = str2num(get(handles.SpecificHeat,'String'));
DiaThroat = str2num(get(handles.DiameterThroat,'String'));
r = DiaThroat / 2;
L = str2num(get(handles.DivLength,'String'));
R = str2num(get(handles.R,'String'));
nuMe = (sqrt((gam + 1) / (gam - 1))) * ((atan(sqrt(((gam - 1)./...
       (gam + 1)) * (Me^2 - 1))))) - atan(sqrt(Me^2 - 1));
thethaWmax = nuMe / 2
set(handles.thethaWmax,'String',thethaWmax);
varthetha = thethaWmax / n;

%DETERMINATION OF PRANDTL MEYER CONSTANTS
kneg1 = [varthetha*2:varthetha*2:nuMe];
kpos1 = zeros(1,n);
kneg2 = [kneg1(2):varthetha*2:nuMe];
kpos2 = - kneg1(2).*ones(1,n-1);
kneg3 = [kneg1(3):varthetha * 2:nuMe];
kpos3 = - kneg1(3) .* ones(1,n-2);
kneg4 = [kneg1(4):varthetha * 2:nuMe];
kpos4 = - kneg1(4) .* ones(1,n-3);
kneg5 = [kneg1(5):varthetha * 2:nuMe];
kpos5 = - kneg1(5) .* ones(1,n-4);
kneg6 = [kneg1(6):varthetha * 2:nuMe];
kpos6 = - kneg1(6) .* ones(1,n-5);
kneg7 = [kneg1(7):varthetha * 2:nuMe];
kpos7 = - kneg1(7) .* ones(1,n-6);
kneg8 = [kneg1(8):varthetha * 2:nuMe];
```

```
kpos8 = - kneg1(8) .* ones(1,n-7);
kneg9 = [kneg1(9):varthetha * 2:nuMe];
kpos9 = - kneg1(9) .* ones(1,n-8);
kneg10 = [kneg1(10):varthetha * 2:nuMe];
kpos10 = - kneg1(10) .* ones(1,n-9);


%DETERMINATION OF PRANDTL MEYER FUNCTION
nu1 = (kneg1 - kpos1)/2;
nu2 =(kneg2 - kpos2)/2;
nu3 = (kneg3 - kpos3)/2;
nu4 = (kneg4 - kpos4)/2;
nu5 = (kneg5 - kpos5)/2;
nu6 = (kneg6 - kpos6)/2;
nu7 = (kneg7 - kpos7)/2;
nu8 = (kneg8 - kpos8)/2;
nu9 = (kneg9 - kpos9)/2;
nu10 = (kneg10 - kpos10)/2;

%DETERMINATION OF PRANDTL MEYER ANGLE
thetha1 =(kneg1 + kpos1)/2;
thetha2 =(kneg2 + kpos2)/2;
thetha3 = (kneg3 + kpos3)/2;
thetha4 = (kneg4 + kpos4)/2;
thetha5 = (kneg5 + kpos5)/2;
thetha6 = (kneg6 + kpos6)/2;
thetha7 = (kneg7 + kpos7)/2;
thetha8 = (kneg8 + kpos8)/2;
thetha9 = (kneg9 + kpos9)/2;
thetha10 = (kneg10 + kpos10)/2;

%DETERMINATION OF MACH NUMBER
M1 = (1 + 0.7863*nu1.^0.66667 + 0.03214*nu1.^1.33333 - 0.099*nu1.^2)./...
     (1 - 0.38853*nu1.^0.66667 - 0.10951*nu1.^1.33333);
M2 = (1 + 0.7863*nu2.^0.66667 + 0.03214*nu2.^1.33333 - 0.099*nu2.^2)./...
     (1 - 0.38853*nu2.^0.66667 - 0.10951*nu2.^1.33333);
```

```
M3 = (1 + 0.7863*nu3.^0.66667 + 0.03214*nu3.^1.33333 - 0.099*nu3.^2)./...
     (1 - 0.38853*nu3.^0.66667 - 0.10951*nu3.^1.33333);
M4 = (1 + 0.7863*nu4.^0.66667 + 0.03214*nu4.^1.33333 - 0.099*nu4.^2)./...
     (1 - 0.38853*nu4.^0.66667 - 0.10951*nu4.^1.33333);
M5 = (1 + 0.7863*nu5.^0.66667 + 0.03214*nu5.^1.33333 - 0.099*nu5.^2)./...
     (1 - 0.38853*nu5.^0.66667 - 0.10951*nu5.^1.33333);
M6 = (1 + 0.7863*nu6.^0.66667 + 0.03214*nu6.^1.33333 - 0.099*nu6.^2)./...
     (1 - 0.38853*nu6.^0.66667 - 0.10951*nu6.^1.33333);
M7 = (1 + 0.7863*nu7.^0.66667 + 0.03214*nu7.^1.33333 - 0.099*nu7.^2)./...
     (1 - 0.38853*nu7.^0.66667 - 0.10951*nu7.^1.33333);
M8 = (1 + 0.7863*nu8.^0.66667 + 0.03214*nu8.^1.33333 - 0.099*nu8.^2)./...
     (1 - 0.38853*nu8.^0.66667 - 0.10951*nu8.^1.33333);
M9 = (1 + 0.7863*nu9.^0.66667 + 0.03214*nu9.^1.33333 - 0.099*nu9.^2)./...
     (1 - 0.38853*nu9.^0.66667 - 0.10951*nu9.^1.33333);
M10=(1 + 0.7863*nu10.^0.66667 + 0.03214*nu10.^1.33333 - 0.099*nu10.^2)./...
      (1 - 0.38853*nu10.^0.66667 - 0.10951*nu10.^1.33333);

%DETERMINATION OF MACH ANGLES
mu1 = asin(1 ./ M1);
mu2 = asin(1 ./ M2);
mu3 = asin(1 ./ M3);
mu4 = asin(1 ./ M4);
mu5 = asin(1 ./ M5);
mu6 = asin(1 ./ M6);
mu7 = asin(1 ./ M7);
mu8 = asin(1 ./ M8);
mu9 = asin(1 ./ M9);
mu10 = asin(1 ./ M10);

%START POINT
x = [0:0.01:100];
x0 = 0;
y0 = r;

%DETERMINATION OF COORDINATES OF POINT 1 ON THE CONTOUR
shapeline1 = x * tan(thethaWmax) + y0;
```

```
charline1 = -(cot(thetha1(1))) * x + y0;
slobline1 = thetha1(10) + mu1(10);
obline1 = slobline1 * x - r * slobline1 / cot(thetha1(1));
x1 = ((-r * slobline1 / cot(thetha1(1))) - y0) /...
    (tan(thethaWmax) - slobline1);
y1 = slobline1 * x1 - r * slobline1 / cot(thetha1(1));

%DETERMINATION OF COORDINATES OF POINT 2 ON THE CONTOUR
shapeline2 = x * tan((thetha1(10)+thetha1(8))/2) - x1 * ...
             tan((thetha1(10)+thetha1(8))/2) + y1;
charline2 = - cot(thetha1(2)) * x + r;
slobline2 = thetha2(9) + mu2(9);
obline2 = slobline2 * x - r * slobline2 / cot(thetha1(2));
x2 = ((-r * slobline2 / cot(thetha1(2))) - (- x1 * tan((thetha1(10)+...
    thetha1(8))/2) + y1))/((tan((thetha1(10)+thetha1(8))/2)) - slobline2);
y2 = slobline2 * x2 - r * slobline2 / cot(thetha1(2));

%DETERMINATION OF COORDINATES OF POINT 3 ON THE CONTOUR
shapeline3 = x * tan((thetha1(8)+thetha1(7))/2) - x2 * ...
             tan((thetha1(8)+thetha1(7))/2) + y2;
charline3 = - cot(thetha1(3)) * x + r;
slobline3 = thetha3(8) + mu3(8);
obline3 = slobline3 * x - r * slobline3 / cot(thetha1(3));
x3 = ((-r * slobline3 / cot(thetha1(3))) - ( - x2 * tan((thetha1(8)+...
    thetha1(7))/2) + y2))/((tan((thetha1(8)+thetha1(7))/2)) - slobline3);
y3 = slobline3 * x3 - r * slobline3 / cot(thetha1(3));

%DETERMINATION OF COORDINATES OF POINT 4 ON THE CONTOUR
shapeline4 = x * tan((thetha1(7)+thetha1(6))/2) - x3 * ...
             tan((thetha1(7)+thetha1(6))/2) + y3;
charline4 = - cot(thetha1(4)) * x + r;
slobline4 = thetha4(7) + mu4(7);
obline4 = slobline4 * x - r * slobline4 / cot(thetha1(4));
x4 = ((-r * slobline4 / cot(thetha1(4))) - ( - x3 * tan((thetha1(7)+...
    thetha1(6))/2) + y3))/((tan((thetha1(7)+thetha1(6))/2)) - slobline4);
y4 = slobline4 * x4 - r * slobline4 / cot(thetha1(4));
```

```
%DETERMINATION OF COORDINATES OF POINT 5 ON THE CONTOUR
shapeline5 = x * tan((thetha1(6)+thetha1(5))/2) - x4 * ...
             tan((thetha1(6)+thetha1(5))/2) + y4;
charline5 = - cot(thetha1(5)) * x + r;
slobline5 = thetha5(6) + mu5(6);
obline5 = slobline5 * x - r * slobline5 / cot(thetha1(5));
x5 = ((-r * slobline5 / cot(thetha1(5))) - ( - x4 * tan((thetha1(6)+...
    thetha1(5))/2) + y4))/((tan((thetha1(6)+thetha1(5))/2)) - slobline5);
y5 = slobline5 * x5 - r * slobline5 / cot(thetha1(5));

%DETERMINATION OF COORDINATES OF POINT 6 ON THE CONTOUR
shapeline6 = x * tan((thetha1(5)+thetha1(4))/2) - x5 * ...
             tan((thetha1(5)+thetha1(4))/2) + y5;
charline6 = - cot(thetha1(6)) * x + r;
slobline6 = thetha6(5) + mu6(5);
obline6 = slobline6 * x - r * slobline6 / cot(thetha1(6));
x6 = ((-r * slobline6 / cot(thetha1(6))) - ( - x5 * tan((thetha1(5)+...
    thetha1(4))/2) + y5))/((tan((thetha1(5)+thetha1(4))/2)) - slobline6);
y6 = slobline6 * x6 - r * slobline6 / cot(thetha1(6));

%DETERMINATION OF COORDINATES OF POINT 7 ON THE CONTOUR
shapeline7 = x * tan((thetha1(4)+thetha1(3))/2) - x6 * ...
             tan((thetha1(4)+thetha1(3))/2) + y6;
charline7 = - cot(thetha1(7)) * x + r;
slobline7 = thetha7(4) + mu7(4);
obline7 = slobline7 * x - r * slobline7 / cot(thetha1(7));
x7 = ((-r * slobline7 / cot(thetha1(7))) - ( - x6 * tan((thetha1(4)+...
    thetha1(3))/2) + y6))/((tan((thetha1(4)+thetha1(3))/2)) - slobline7);
y7 = slobline7 * x7 - r * slobline7 / cot(thetha1(7));

%DETERMINATION OF COORDINATES OF POINT 8 ON THE CONTOUR
shapeline8 = x * tan((thetha1(3)+thetha1(2))/2) - x7 * ...
             tan((thetha1(3)+thetha1(2))/2) + y7;
charline8 = - cot(thetha1(8)) * x + r;
slobline8 = thetha8(3) + mu8(3);
```

```
obline8 = slobline8 * x - r * slobline8 / cot(thetha1(8));
x8 = ((-r * slobline8 / cot(thetha1(8))) - ( - x7 * tan((thetha1(3)+...
    thetha1(2))/2) + y7))/((tan((thetha1(3)+thetha1(2))/2)) - slobline8);
y8 = slobline8 * x8 - r * slobline8 / cot(thetha1(8));

%DETERMINATION OF COORDINATES OF POINT 9 ON THE CONTOUR
shapeline9 = x * tan((thetha1(2)+thetha1(1))/2) - x8 * ...
             tan((thetha1(2)+thetha1(1))/2) + y8;
charline9 = - cot(thetha1(9)) * x + r;
slobline9 = thetha9(2) + mu9(2);
obline9 = slobline9 * x - r * slobline9 / cot(thetha1(9));
x9 = ((-r * slobline9 / cot(thetha1(9))) - ( - x8 * tan((thetha1(2)+...
    thetha1(1))/2) + y8))/((tan((thetha1(2)+thetha1(1))/2)) - slobline9);
y9 = slobline9 * x9 - r * slobline9 / cot(thetha1(9));

%DETERMINATION OF COORDINATES OF POINT 10 ON THE CONTOUR
shapeline10 = x * tan(thetha1(1)/2) - x9 * tan(thetha1(1)/2) + y9;
charline10 = - cot(thetha1(10)) * x + r;
slobline10 = thetha10(1) + mu10(1);
obline10 = slobline10 * x - r * slobline10 / cot(thetha1(10));
x10 = ((-r * slobline10 / cot(thetha1(10))) - (- x9 * tan(thetha1(1)/2)+...
    y9))/((tan(thetha1(1)/2)) - slobline10);
y10 = slobline10 * x10 - r * slobline10 / cot(thetha1(10));

%DISPLAY CONTOUR POINTS
set(handles.x0,'String',x0);
set(handles.x1,'String',x1);
set(handles.x2,'String',x2);
set(handles.x3,'String',x3);
set(handles.x4,'String',x4);
set(handles.x5,'String',x5);
set(handles.x6,'String',x6);
set(handles.x7,'String',x7);
set(handles.x8,'String',x8);
set(handles.x9,'String',x9);
set(handles.x10,'String',x10);
```

```
set(handles.L,'String',L);

set(handles.y0,'String',y0);
set(handles.y1,'String',y1);
set(handles.y2,'String',y2);
set(handles.y3,'String',y3);
set(handles.y4,'String',y4);
set(handles.y5,'String',y5);
set(handles.y6,'String',y6);
set(handles.y7,'String',y7);
set(handles.y8,'String',y8);
set(handles.y9,'String',y9);
set(handles.y10,'String',y10);
set(handles.y11,'String',y10);

% PROPERTIES CALCULATION
ExitMachNumberMoc = M10; %Exit Mach Number from curve part
AreaRatioMoc = y10^2 / r^2;

% DISPLAY VALUES
set(handles.ExitMachNumberMoc,'String',ExitMachNumberMoc);
set(handles.AreaRatioMoc,'String',AreaRatioMoc);

%PLOT CURVES CONTOUR
axes(handles.PlotCurve);
plot(x,shapeline1,'g-',x,charline1,'r-',x,obline1,'b-',...
x,shapeline2,'g-',x,charline2,'r-',x,obline2,'b-',...
x,shapeline3,'g-',x,charline3,'r-',x,obline3,'b-',...
x,shapeline4,'g-',x,charline4,'r-',x,obline4,'b-',...
x,shapeline5,'g-',x,charline5,'r-',x,obline5,'b-',...
x,shapeline6,'g-',x,charline6,'r-',x,obline6,'b-',...
x,shapeline7,'g-',x,charline7,'r-',x,obline7,'b-',...
x,shapeline8,'g-',x,charline8,'r-',x,obline8,'b-',...
x,shapeline9,'g-',x,charline9,'r-',x,obline9,'b-',...
x,shapeline10,'g-',x,charline10,'r-',x,obline10,'b-')
hold on
```

```
xc = [x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 x10];
yc = [y0 y1 y2 y3 y4 y5 y6 y7 y8 y9 y10];
plot(xc,yc,'k-*','LineWidth',3)
grid on
axis equal
axis([0 x10 0 y10])

%PLOT ENTIRE NOZZLE CONTOUR
xp = [x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 L];
yp = [y0 y1 y2 y3 y4 y5 y6 y7 y8 y9 y10 y10];
ypn = -1*yp;
axes(handles.PlotShape);
plot(xp,yp,'k',xp,ypn,'k','LineWidth',3)
grid
axis equal
axis([0 L -y10 y10])

% DISPLAY PLOT RESULTS
set(handles.LengthCurve,'String',x10);
ExitDiameter = 2*y10;
set(handles.ExitDia,'String',ExitDiameter);

% --- Executes on button press in Exit.
function Exit_Callback(hObject, eventdata, handles)
% hObject    handle to Exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.figure1)

function P0_Callback(hObject, eventdata, handles)
% hObject    handle to P0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of P0 as text
% str2double(get(hObject,'String')) returns contents of P0 as a double
```

```
% --- Executes during object creation, after setting all properties.
function P0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to P0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function T0_Callback(hObject, eventdata, handles)
% hObject    handle to T0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of T0 as text
% str2double(get(hObject,'String')) returns contents of T0 as a double


% --- Executes during object creation, after setting all properties.
function T0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to T0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function R_Callback(hObject, eventdata, handles)
% hObject    handle to R (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of R as text
% str2double(get(hObject,'String')) returns contents of R as a double


% --- Executes during object creation, after setting all properties.
function R_CreateFcn(hObject, eventdata, handles)
% hObject    handle to R (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in HelpMoc.
function HelpMoc_Callback(hObject, eventdata, handles)
% hObject    handle to HelpMoc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('SpecificationsMoc.pdf')
```