الجمهورية الجزائرية الديمقراطية الشعبية

LA REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahleb Blida 1

Institut d'Aéronautique et des Études Spatiales

Département Etudes spatiales

Mémoire de fin d'études

En vue de l'obtention du diplôme de

Master en Aéronautique

***Option :* Propulsion Spatiales**

THEME

Développement d'un programme informatique basé sur python pour la méthode des caractéristiques pour la conception des tuyères supersoniques

Proposé et dirigé par :

Dr. KBAB Hakim

Co promoteur:

TCHAREK Anis

Réalisé par :

Mr. BENZIANE SALAH EDDINE,

Mr. BENSALAH DHIA ELHAK.

***Promotion :* 2022/2023**

Soutenue devant le jure composer de :

Pr. Nom Prenom Professeur Président

Dr. Nom Prenom MCA Examinateur

Dr. Nom Prenom MCB Examinateur

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF COMMITTEE APPROVAL

Dr. Nom Prenom, Chair

Institute of Aeronautics and Spaces studies

Dr. Nom Prenom

Institute of Aeronautics and spaces studies

Dr. Nom Prenom

Institute of Aeronautics and Spaces studies

Click here to enter text.

Add or Delete Committee Member

Add or Delete Department

Approved by:

Dr. FORM 9 THESIS FORM HEAD NAME HERE

Remerciements :

Nos remerciements vont premièrement à ALLAH le tout puissant de nous avoir donné la volonté, la patience qu'il nous donnés pour mener à terme ce travail. Sans que nous oublions nos très chers parents pour leurs soutien et aides très précieuses.

En premier lieu, Nous remercions notre encadreur M. KEBAB et Co-promoteur T. ANIS pour toute l'aide qu'ils nous ont apporté durant toute la durée de la réalisation de ce mémoire.

Nous tenons à témoigner toute nos reconnaissances aux nombreuses personnes qui ont contribué au succès de notre stage et qu'ils ont manifestées à notre égard.

Nous souhaitons particulièrement remercier B. OUSSAMA ingénieur à la base de maintenance Air Algérie pour sa collaboration en partageant ses connaissances, et pour son énorme aide pendant le stage pratique.

Nous remercions aussi SIDALI HAIF doctorant à l'institut d'aéronautique et des études spatiales qui n'a ménagé aucun effort tant pour ses conseils et son partage de ses connaissances sur le CFD.
Finalement, nous réservons une place singulière à toutes les personnes qui ont contribué, d'une façon directe ou indirecte à l'achèvement de ce travail.

A tous les membres de nos familles et à nos amis.

Dédicace

Je dédier ce travail avec grand amour, sincérité et fierté

À Mes chers parents pour leur patience, leur amour, leur soutien et leur encouragement.

À mes frères.

À tous les membres de la famille.

À tous mes amis, toutes mes professeures.

Et à tout qui complus ce modeste travail

Benziane Salah Eddine

Je dédie ce travail à mes chers parents qui m'ont accordé leur confiance et m'ont encouragé durant toute ma vie.

A mon frère et à mes deux sœurs.

A tous mes enseignants et professeurs.

A tous mes amis et à tous les membres de la famille.

Au club d'innovation en aérospatiale.

Bensalah Dhia Elhak

Liste des matières

Remerciements ……4

Liste des figures ……8

Nomenclature ……12

Résumé ……14

Introduction générale…… 16

I. CHAPITRE I……18

I.1. Introduction ……19
I.2. L'équation différentielle simple……19
I.3. système de deux équations différentielles partielles ……20
I.4. domaine de dépendance et champ d'influence……..22
I.5. Application pour étude d'écoulement irrotationne Supersonique plan et axisymétrique ……23
I.5.1. les équations gouvernantes…….…….... 23
I.5.1.1. L'équations de caractéristiques …….…….....23
I.5.1.2. L'équation de compatibilité ……..…….....26
I.6. Mise en œuvre numérique de la méthode des Caractéristiques……...26
I.6.1. procédure d'intégration numérique …….....26
I.6.2. la méthode directe et la méthode inverse ……..…..28
I.6.3. Les équations aux différences finis…….....29
I.6.4. Procédure de calcul pour un point interne ……..29
I.6.5. Procédure de calcule pour un point paroi direct ……..….32
I.6.6. Procédure de calcule pour un point paroi inverse ……32
I.6.7. Procédure de calcule pour un point sur l'axe de symétrie ……..….34
I.7. Application pour études de tuyères a géométries connues…….……..34
I.8. Application pour la conception de tuyères supersoniques …….….39
I.8.1. Cas d'une tuyère axisymétrique ……..…….....39
I.8.2. Cas de tuyères planes ……..…..... 42
I.9. Conclusion ……...…….…..…….43

II. CHAPITRE II ……...…….....44

II.1. Introduction ……..…..…….....45

II.2.JAVA (LANGAGE) ……...…….... 46
II.3.FORTRAN…….....….... 47

II.4. MATLAB…….. 47
II.5. LE LANGAGE C…….. 48
II.6. LE LANGAGE C++ ……..49
II.7. LE LANGAGE RUBY……..... 51
II.8. LE LANGAGE PASCAL ……...52
II.9. LE LANGAGE ADA……..... 53
II.10. LE LANGAGE DELPHI……54
II.11. SolidWorks…….....55
II.12. AUTOCAD……....56
II.13. CATIA…….....58
II.14. ADVANCED AIRCRAFT ANALYSIS (AAA) ……..59
II.15. ANSYS…….. 60

III. CHAPITRE III

III.1. Introduction…….....….. 63
III.2. Des principales caractéristiques de Python……....64
III.3. Pour quoi python …….....66
III.4. Variables……....70
III.4.1. Définition……..…… 70
III.4.2. Les types de variables…….....… 70
III.4.3. Nommage ……...…71
III.4.4. Opérations…….....…71
III.4.5. La fonction type ()…….... 72
III.4.6. Conversion de types ……....73
III.4.7. Minimum et maximum……... 73
III.5. Affichage ……...73
III.5.1. La fonction `print ()` ……73
III.5.2. BOUCLES ETcomparaisons……..……74
III.5.3. Boucles `for`……...…74
III.5.3.1. Principe……...…74
III.5.3.2. Fonction `range ()`……..…75
III.5.3.3. Nommage de la variable d'itération……....…75
III.5.3.4. Itération sur les indices ou les éléments……..…76
III.5.4. Comparaisons ……....…77
III.5.5. Boucles `while`……...…77

III.6. Tests……79
III.6.1. Définition……....…79
III.6.2. Tests à plusieurs cas…….....…79
III.6.3. Instructions `break` et `continue`…….....…..80
III.7. Modules ……...…81
III.7.1. Définition ……...……81
III.7.2. Quelques modules courants……..... 81
III.8. Fonctions…….……82
III.8.1. Passage d'argument……...…… 82
III.8.2. Renvoi de résultats…… 83
IV. CHAPITRE IV. ……..... 84

IV.1. INTRODUCTION……..……...…… 85
IV.2. Résultats obtenus en utilisant le langage python…….....……85
IV.2.1. Désigne de la tuyère……...…… 86
IV.2.2. EVOLUTION DES PARAMETRES DE L'ECOULEMENT DANS LA TUYERE OBTENUE…….....… 87
IV.3. Etude numérique (simulation) de l'écoulement dans la tuyère obtenue par python……...….92
IV.3.1. INTRODUCTION……..…92
IV.3.2. Application au cas d'étude……..…97
IV.3.3. RESULTATS ET COMMENTAIRES…….……...….102
IV.4. VALIDATION ET COMPARAISON……... 112
IV.5. Exemple d'un subroutine…….…….119
IV.5.1. Subroutine THERMO …….……..…119
IV.5.2. Subroutine THRUST…….……. 120
IV.5.3. Subroutine IVLINE …….…….121
IV.5.4. Syntax differences between Python and Fortran codes……..... 124

V. Chapitre V……..……..……..126
V.1. Conclusion…...……...… 127
V.2. Perspectives……...… 128

Liste des figures

CHAPITRE I

Figure. (I.1) : Caractéristique d'une équation différentielle de premier ordre…..19
Figure. (I.2) : Résultat de caractéristiques pour un système de deux EDP de premier ordre…….……22
Figure. (I.3) : Domaine de dépendance……..…….22
Figure. (I.4) : Champ d'influence…….…..22
Figure. (I.5) : Relation entre u, v et V,θ……..……25
Figure. (I.6) : Relation entre α et M……..……..25
Figure. (I.7) : Les caractéristiques d'un écoulement Bidimensionnel Irrotationnel et supersonique……26
Figure. (I.8) : schéma de calcul par différences finis pour La méthode des caractéristiques…….…27
Figure. (I.9) : La méthode directe……..…….29
Figure. (I.10) : La méthode inverse……..…..29
Figure. (I.11) : Point intérieur sur l'axe de symétrie……..32
Figure. (I.12) : Point paroi direct…….....32
Figure. (I.13) : Point paroi inverse…….…33
Figure. (I.14) : Point axiale…….…34
Figure. (I.15) : tuyère supersonique à divergent conique…….……..35
Figure. (I .16) : Schéma de la ligne initiale et des points préspicifiés…….…...35
Figure. (I.17) procédure de calcule d'un point interne À partir de la ligne initiale…….……...36
Figure. (I.18) : procédure de calcule d'un point axiale……..…….36
Figure. (I.19) : procédure de calcule d'un point inverse…….……37
Figure. (I.20) : procédure de calcul de la zone d'expansion initiale…….……..37
Figure. (I.21) : procédure de calcule d'un point direct…….…..37
Figure. (I.22) : Procédure de calcul de la partie divergente……..…..38
Figure. (I.23) : Croisement de deux caractéristiques pour former une onde de choque oblique……...38
Figure. (I.24) : méthode de correction de calcule dans le cas de croisement de deux caractéristiques de même famille……..….39
Figure. (I.25) : Données initiales du problème…….……..40
Figure. (I.26) : Localisation du mach de désigne sur l'axe X……..…40
Figure. (I.27) : les conditions imposées à la sortie de la tuyère……..41
Figure. (I.28) : détermination du contour d'une tuyère supersonique……...42
Figure. (I.29) : Détermination du contour d'une tuyère supersonique plane……..43

CHAPITRE III

Figure (III.1) exemple de variable ……..70
Figure (**III.2)** exemple de type de variable……71
Figure (III.3) exemple d'opération numérique……72
Figure (III.4) exemple d'opération de la puissance……72
Figure (III.5) exemple de La fonction type ()……72
Figure (III.6) exemple de conversion de types……73
Figure (III.7) exemple de minimum et maximum……73
Figure (III.8) exemple de La fonction `print()` ……74
Figure (III.9) exemple de Boucles for……74
Figure (III.10) exemple de Fonction range ()……75
Figure (III.11) exemple d'itération sur les indices ou les éléments……76
Figure (III.12) exemple d'itérations directe sur les éléments……76
Figure (III.13) exemple d'itérations avec des indices……76
Figure (III.14) exemple de la fonction enumerate()……77
Figure (III.15) exemple de la boucles while……78
Figure (III.16) exemple de stopper l'exécution……78
Figure (III.17) exemple de la fonction input ()……79
Figure (III.18) exemple de l'instruction if……79
Figure (III.19) exemple de l'instruction if et else ……80
Figure (III.20) exemple de l'instruction **`break`** ……80
Figure (III.21) exemple de l'instruction **`continue`**……81
Figure (III.22) exemple de fonction……82
Figure (III.23) exemple de passage d'arguments……83
Figure (III.24) exemple de renvoi de résultats……83
Figure (III.25) exemple de renvoie un objet de type **`tuple`**……83

CHAPITRE IV

Figure. (IV.1) Résultats de conception de la tuyère idéale plane avec la méthode des caractéristiques en utilisant le code python……86

Figure. (IV.2) Contour et courbes d'évolution du nombre de Mach pour une tuyère idéale plane calculés avec la méthode des caractéristiques programmée sur le langage Python……..……88

Figure. (IV.3) Contour et courbes d'évolution de la pression statique pour une tuyère idéale plane calculés avec la méthode des caractéristiques programmée sur le langage Python……89

Figure. (IV.4) Contour et courbes d'évolution de la température statique pour une tuyère idéale plane calculés avec la méthode des caractéristiques programmée sur le langage Python……..……..90

Figure. (IV.5) Contour et courbes d'évolution de la densité pour une tuyère idéale plane calculés avec la méthode des caractéristiques programmée sur le langage Python……..……..…..92

Figure. (IV.6) Configuration du solveur CFD ANSYS Fluent selon le cas d'étude……..……99

Figure. (IV.7) Conditions (pressions et température) imposées à l'entrée du domaine de calcul (col) …99

Figure. (IV.8) Conditions (pressions et température) imposées à la sortie de la tuyère…….….100

Figure. (IV.9) Maillage structuré à base d'éléments quadrilatères réalisé sur le package ANSYS…….101

Figure. (IV.10) Visualisation du maillage structuré de la tuyère idéal plane sur ANSYS Fluent……...102

Figure. (IV.11) Contour du nombre de Mach calculé avec le solveur CFD ANSYS Fluent……..…..103

Figure. (IV.12) Courbe d'évolution du nombre de Mach le long de la paroi de la tuyère idéale plane calculée avec le solveur CFD Fluent……..104

Figure. (IV.13) Contour du nombre de la pression statique calculé avec le solveur CFD ANSYS Fluent…….……105

Figure. (IV.14) Courbe d'évolution de la pression statique le long de la paroi de la tuyère idéale plane calculée avec le solveur CFD Fluent……..……....106

Figure. (IV.15) Contour du nombre de la température statique calculé avec le solveur CFD ANSYS Fluent…….……..…..107

Figure. (IV.16) Courbe d'évolution de la température statique le long de la paroi de la tuyère idéale plane calculée avec le solveur CFD Fluent……108

Figure. (IV.17) Contour de la densité calculé avec le solveur CFD ANSYS Fluent……...109

Figure. (IV.18) Courbe d'évolution de la densité le long de la paroi de la tuyère idéale plane calculée avec le solveur CFD Fluent……111

Figure. (IV.19) Représentation des vecteurs vitesses pour tuyère idéale plane calculés avec le solveur CFD Fluent……..……..…..112

Figure. (IV.20) Résultats de conception des contours de la tuyère idéale plane avec les codes Python et Fortran pour la méthode des caractéristiques……...113

Figure. (IV.21) Comparaison des courbes d'évolution du nombre de Mach pour une tuyère idéale plane calculées avec CFD (Fluent), méthode des caractéristiques (codes Python et Fortran)……...….115
Figure. (IV.22) Comparaison des courbes d'évolution de la pression statique pour une tuyère idéale plane calculées avec CFD (Fluent), méthode des caractéristiques (codes Python et Fortran)…….….116
Figure. (IV.23) Comparaison des courbes d'évolution de la température statique pour une tuyère idéale plane calculées avec CFD (Fluent), méthode des caractéristiques (codes Python et Fortran)……...117
Figure. (IV.24) Comparaison des courbes d'évolution de la densité pour une tuyère idéale plane calculées avec CFD (Fluent), méthode des caractéristiques (codes Python et Fortran)…….…...118

Nomenclature

Notions Latines

$\dot{m}$	Débit
A	Section de tuyère
$A*$	Section au col
$C-$	Caractéristique descendante
C	Caractéristique montante
$C*$	Vitesse effective
Cf	Le coefficient de poussée
F	Force de poussée
H	Enthalpie
Is	Impulsion spécifique
M	Nombre de mach
P	Pression statique
Pa	Pression ambiant
Pe	Pression à la sortie de la tuyère
$P0$	Pression total
R	Constante des gaz parfaits
Re	Nombre de Reynolds
T	Température statique
V	Vitesse
Ve	Vitesse à la sortie de la tuyère
$Veff$	Vitesse effective
a	Célérité de son
u	Composante axiale de vecteur vitesse
v	Composante radial de vecteur vitesse
$y*$	Hauteur de col
ye	Hauteur de sortie

Lettres grecs

α	Angle de divergence de la tuyère

$\boldsymbol{\rho}$	Masse volumique
$\boldsymbol{Cp}$	La chaleur spécifique à pression constante
$\boldsymbol{Cv}$	La chaleur spécifique à volume constant
$\boldsymbol{\gamma}$	Rapport des chaleurs spécifiques
$\boldsymbol{\mu}$	Viscosité dynamique
$\boldsymbol{\varepsilon}$	Rapport des sections
$\boldsymbol{\omega}$	Fréquence de turbulence
$\boldsymbol{\mu t}$	Viscosité turbulente
$\boldsymbol{\theta}$	Angle
$\boldsymbol{vm}$	Fonction Prandtl-Meyer

Acronyms

CFD	Computational Fluid Dynamics
GP	Gaz Parfait
HT	Haute Température
MOC	Method of characteristics

Résumé :

Ce travail porte sur l'étude des tuyères supersoniques en utilisant Python comme outil d'analyse et de simulation. La méthode des caractéristiques a été choisie comme approche principale pour la conception des tuyères. La théorie et les méthodes pertinentes ont été décrites en détail, suivies de l'implémentation d'un code de calcul basé sur la méthode des caractéristiques.

Le code de calcul a été validé en effectuant des simulations numériques de l'écoulement dans la tuyère conçue. Les résultats obtenus ont été comparés aux contours et aux graphiques représentant l'évolution des paramètres thermodynamiques. Les résultats ont montré une bonne correspondance avec les résultats expérimentaux et les résultats obtenus par d'autres méthodes de simulation.

L'utilisation de Python a offert de nombreux avantages, notamment grâce à sa polyvalence, ses bibliothèques spécialisées et sa communauté active. Python a permis de réaliser des simulations numériques efficaces, d'implémenter des modèles mathématiques complexes et d'analyser les performances des tuyères de manière précise.

En conclusion, ce travail démontre l'efficacité de Python dans l'étude des tuyères supersoniques. Les résultats obtenus ont confirmé la validité de la méthode des caractéristiques pour la conception des tuyères. Les perspectives futures comprennent l'amélioration des modèles, l'optimisation des performances, l'étude de cas spécifiques, la validation expérimentale et le développement de nouvelles méthodes. Ces perspectives ouvrent la voie à des avancées significatives dans la compréhension et l'optimisation des tuyères supersoniques dans diverses applications aérospatiales et de propulsion.

Abstract:

This work focuses on the study of supersonic nozzles using Python as an analysis and simulation tool. The method of characteristics was chosen as the main approach for nozzle design. The relevant theory and methods were described in detail, followed by the implementation of a calculation code based on the method of characteristics.

The calculation code was validated by conducting numerical simulations of the flow in the designed nozzle. The obtained results were compared to the contours and graphs depicting the evolution of thermodynamic parameters. The results showed good agreement with experimental data and results obtained by other simulation methods.

The use of Python provided numerous advantages, including its versatility, specialized libraries, and active community. Python enabled efficient numerical simulations, implementation of complex mathematical models, and precise analysis of nozzle performance.

In conclusion, this work demonstrates the effectiveness of Python in the study of supersonic nozzles. The obtained results confirmed the validity of the method of characteristics for nozzle design. Future perspectives include improving the models, optimizing performance, studying specific cases, experimental validation, and developing new methods. These perspectives pave the way for significant advancements in the understanding and optimization of supersonic nozzles in various aerospace and propulsion applications.

الملخص

تركز هذه الدراسة على دراسة فوهات فائقة الصوت باستخدام لغة البرمجة بايثون كأداة تحليل ومحاكاة. تم اختيار طريقة الخصائص كالطريقة الرئيسية لتصميم الفوهات. تم وصف النظرية والأساليب ذات الصلة بالتفصيل، تلاها تنفيذ كود حسابي يستند إلى طريقة الخصائص.

تم التحقق من صحة كود الحساب عن طريق إجراء محاكاة رقمية لتدفق الهواء في الفوهة المصممة. تمت مقارنة النتائج المتحصلة مع المنحنيات والرسوم البيانية التي تصور تطور المعلمات الحرارية. أظهرت النتائج تطابقًا جيدًا مع البيانات التجريبية والنتائج المتحصلة من طرق المحاكاة الأخرى.

استفادة بايثون من العديد من المزايا، بما في ذلك التنوع، والمكتبات المتخصصة، والمجتمع النشط. سمحت بايثون بإجراء محاكاة رقمية فعالة، وتنفيذ نماذج رياضية معقدة، وتحليل دقيق لأداء الفوهة.

في الختام، تظهر هذه الدراسة فعالية استخدام لغة البرمجة بايثون في دراسة الفوهات الفائقة الصوت. أكدت النتائج المتحصلة صحة طريقة الخصائص لتصميم الفوهات. تشمل التوجهات المستقبلية تحسين النماذج، و تحسين الأداء، ودراسة حالات محددة، والتحقق التجريبي، وتطوير طرق جديدة. تمهد هذه التوجهات الطريق لتقدمات مهمة في فهم وتحسين الفوهات الفائقة الصوت في مختلف التطبيقات الجوية والدفعية .

Introduction générale

L'étude des tuyères supersoniques joue un rôle crucial dans le domaine de l'ingénierie aérospatiale et des systèmes de propulsion. Ces tuyères sont des composants essentiels dans diverses applications, notamment les moteurs de fusées, les turboréacteurs et les scramjets. Comprendre les phénomènes d'écoulement complexes et optimiser la conception des tuyères supersoniques sont des facteurs clés pour obtenir des systèmes de propulsion efficaces et performants.

Ces dernières années, les méthodes de calcul et les simulations numériques sont devenues des outils indispensables pour analyser et concevoir des tuyères supersoniques. Ces méthodes permettent d'obtenir des informations sur les caractéristiques de l'écoulement, les propriétés thermodynamiques et les performances des tuyères. Parmi les différentes techniques de calcul disponibles, la méthode des caractéristiques (MOC) s'est imposée comme un choix populaire pour la conception et l'analyse des tuyères supersoniques.

Dans ce travail, nous nous concentrons sur l'étude des tuyères supersoniques en utilisant Python en tant que langage de programmation. Python offre une plateforme polyvalente et conviviale pour la mise en œuvre de simulations numériques, l'analyse des données et la visualisation des résultats.

D’abord, nous avons entrepris une analyse approfondie de la méthode des caractéristiques et des logiciels de conception utilisés dans le domaine des tuyères supersoniques. Nous avons décrit les fondements théoriques et les méthodologies pertinentes de cette méthode. De plus, nous avons examiné plusieurs logiciels de conception couramment utilisés dans ce domaine, en détaillant leurs caractéristiques spécifiques.

Ensuite, nous nous sommes concentrés sur l'utilisation de Python en tant qu'outil de programmation dans notre étude. Nous avons expliqué les concepts clés de Python, sa syntaxe et les bibliothèques associées. Python présente de nombreux avantages pour l'analyse et la modélisation des tuyères supersoniques, notamment sa facilité d'utilisation, sa flexibilité et sa large gamme de bibliothèques spécialisées.

Le cœur de notre travail a été consacré à la présentation détaillée de notre approche basée sur python pour l'analyse des tuyères. Nous avons développé du code de calcul qui a été validé en comparant les résultats obtenus avec des données expérimentales, la méthode des caractéristiques

(MOC) et d'autres méthodes de la dynamique des fluides computationnelle (CFD). Les simulations CFD nous ont permis de modéliser et de résoudre numériquement les équations de conservation de la masse, de la quantité de mouvement et de l'énergie, ce qui a fourni des informations détaillées sur les caractéristiques de l'écoulement et les performances des tuyères supersoniques.

Enfin, nous avons comparé les résultats obtenus à partir de notre approche basée sur Python avec ceux des autres méthodes utilisées dans le domaine. Cette comparaison nous a permis d'évaluer l'efficacité et la précision de notre approche, ainsi que de mettre en évidence les avantages spécifiques de l'utilisation de Python.

Dans l'ensemble, ce travail vise à apporter une contribution significative à l'analyse et à la conception des tuyères supersoniques en utilisant Python comme outil de calcul. Les connaissances acquises grâce à cette étude amélioreront notre compréhension de la physique de l'écoulement et de l'optimisation des performances des tuyères supersoniques. Ces avancées ouvriront la voie à de nouvelles découvertes et innovations dans le domaine des systèmes de propulsion aérospatiale.

VI.CHAPITRE I

METHODE DES CARACTERISTIQUE

VI.1. Introduction

L'interaction entre la mécanique des fluides et les mathématiques appliquées est depuis longtemps fluctueuse. Les phénomènes physiques complexes comme les écoulements transsoniques posent des problèmes mathématiques difficiles, ainsi les résultats théoriques actuels ne concernent que des exemples ou la physique est modélisée de façon très simplifiée, mais les méthodes numériques développées à partir de ces analyses mathématiques sont de plus en plus performantes et fournissent à l'ingénieur de nouveau outils d'aide à la conception. Notons que ces progrès sont intimement lié à l'emploi de calculateurs de plus en plus puissants comme les ordinateurs.

VI.2. L'équation différentielle simple

L'équation différentielle partielle de premier ordre: $af_x + bf_y + c = 0$ (I.1)

Considérons premièrement l'approche de la recherche, et réécrivant l'équation (I.1) comme suit :

$$a\left(f_x + \frac{b}{a} f_y \right) + c = 0 \qquad \text{(I.2)}$$

Si la fonction $f(x, y)$ est continue, la différence totale s'écrit ;

$$df = f_x dx + f_y dy \qquad \text{(I.3)}$$

Et peut-être encore écrite comme suit :

$$\frac{df}{dx} = \left(f_x + \frac{dy}{dx} f_y \right) \qquad \text{(I.4)}$$

La caractéristique pour l'équation (I.1) est définit comme la courbe dans la plan xy qui a une pente à chaque point donné par : $\frac{dy}{dx} = \frac{b}{a} = \lambda(x, y, f)$ (I.5)

L'équation (I.4) peut être écrite : $\frac{df}{dx} = \left(f_x + \frac{b}{a} f_y \right)$ (I.6)

Substituant l'équation (I.6) dans (I.2) on obtient l'équation (I.7)

$$adf + cdx = 0 \qquad \text{(I.7)}$$

Pour résoudre l'équation (I.1) une condition initiale est indispensable, la Fig. (I.1) , représente schématiquement la ligne de valeurs initiales $\Gamma_0(x, y)$ dans le plan (x, y) le long de laquelle $f(x, y) = f_0(x, y)$ est spécifiée.

On choisit n'importe quel point $P(x,y)$, en suite on intègre l'équation (I.5) pour pouvoir déterminer la caractéristique C passante par le point initial P et la valeur de f à n'importe quel point Q sélectionné sur la caractéristique C et déterminer en intégrant l'équation de compatibilité à partir du point P jusqu'au point Q le long de la caractéristique C

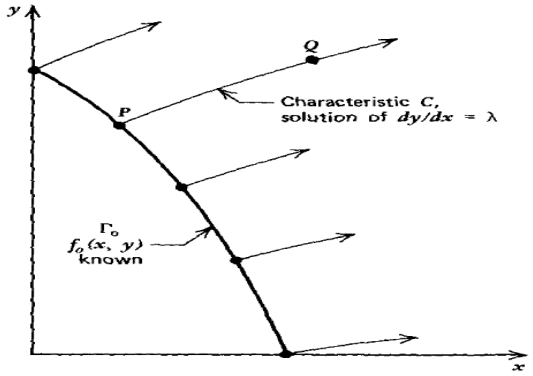

Figure. (I.1). Caractéristique d'une équation différentielle de premier ordre

En sélectionnant plusieurs points sur la ligne initiale Γ_0 comme le point P, le plant (x,y) est entièrement couvert de caractéristiques où sur chacune l'ensemble des valeurs de la fonction f sont déterminées.

Il est très important d'observer les restrictions avant d'utiliser la méthode des caractéristiques, car elle est uniquement applicable dans le cas des équations différentielles hyperbolique quasi linéaire.

Elle est développée ici pour une équation différentielle partielle de premier ordre, elles peuvent même être non homogènes et non linéaires.

VI.3. système de deux équations différentielles partielles

Considérons un système d'équations différentielles partielles notées par L_1 et L_2 comprenant les deux variables dépendantes $u(x,y)$ et $v(x,y)$:

$$\begin{aligned} L_1 &= a_{11}u_x + b_{11}u_y + a_{12}v_x + b_{12}v_y + c_1 = 0 \\ L_2 &= a_{21}u_x + b_{21}u_y + a_{22}v_x + b_{22}v_y + c_2 = 0 \end{aligned} \qquad \text{(I.8)}$$

Soit : $$L = \sigma_1 L_1 + \sigma_2 L_2 = 0 \qquad \text{(I.9)}$$

Où σ_1 et σ_2 deux coefficients à déterminer.

Remplaçons (I.8) dans (I.9) et exprimant le résultat en forme de (I.2) :

$$\begin{aligned}&\left(a_{11}\sigma_1+a_{12}\sigma_2\right)\left[u_x+\frac{\left(b_{11}\sigma_1+b_{21}\sigma_2\right)}{\left(a_{11}\sigma_1+b_{21}\sigma_2\right)}u_y\right]\\&+\left(a_{12}\sigma_1+a_{22}\sigma_2\right)\left[v_x+\frac{\left(b_{12}\sigma_1+b_{22}\sigma_2\right)}{\left(a_{12}\sigma_1+b_{22}\sigma_2\right)}v_y\right]+\left(c_1\sigma_1+c_2\sigma_2\right)=0\end{aligned}\qquad\text{(I.10)}$$

Parce que les variables dépendantes doivent être continues

$$\frac{du}{dx}=u_x+\lambda u_y\quad\text{Et}\quad\frac{dv}{dx}=v_x+\lambda v_y\quad\text{où}\quad\lambda=dy/dx\qquad\text{(I.11)}$$

Donc les caractéristiques sont définies comme les courbes ayant les pentes suivantes :

$$\lambda=\frac{\left(b_{11}\sigma_1+b_{21}\sigma_2\right)}{\left(a_{11}\sigma_1+b_{21}\sigma_2\right)}\quad\text{Et}\qquad\lambda=\frac{\left(b_{12}\sigma_1+b_{22}\sigma_2\right)}{\left(a_{12}\sigma_1+b_{22}\sigma_2\right)}\qquad\text{(I.12)}$$

Le long de la caractéristique spécifiée par (I.12) l'équation (I.10) peut s'écrire comme suit :

$$\left(a_{11}\sigma_1+a_{12}\sigma_2\right)du+\left(a_{12}\sigma_1+a_{22}\sigma_2\right)dv+\left(C_1\sigma_1+C_2\sigma_2\right)dx=0\qquad\text{(I.13)}$$

Par contre l'équation (I.12) peur être réécrite comme suit :

$$\begin{aligned}&\sigma_1\left(a_{11}\lambda-b_{11}\right)+\sigma_2\left(a_{21}\lambda-b_{21}\right)=0\\&\sigma_1\left(a_{12}\lambda-b_{12}\right)+\sigma_2\left(a_{22}\lambda-b_{22}\right)=0\end{aligned}\qquad\text{(I.14)}$$

Pour que cette équation ait une solution ; autre que la triviale $\sigma_1=\sigma_2=0$; on doit satisfaire

La condition ; $$\begin{vmatrix}\left(a_{11}\lambda-b_{11}\right)&\left(a_{21}\lambda-b_{21}\right)\\\left(a_{12}\lambda-b_{12}\right)&\left(a_{22}\lambda-b_{22}\right)\end{vmatrix}=0\qquad\text{(I.15)}$$

Ce qui nous ramène à l'équation suivante : $a\lambda^2+b\lambda+c=0$ (I.16)

Tel que les constantes a, b et s sont définies par:

$a=\left(a_{11}a_{22}-a_{12}b_{21}\right),\quad b=\left(-a_{22}b_{11}-a_{11}b_{22}+a_{12}b_{21}+a_{21}b_{12}\right)$ et $c=\left(b_{11}b_{22}-b_{12}b_{21}\right)$

Le type de l'équation différentielle partielle est déterminé selon le signe de Δ tel que $\Delta=B^2-4AC$.

Dans notre étude on se concentre sur le troisième type du faite qu'on veut étudier l'écoulement supersonique du fait qu'il est gouverné par un système d'équations hyperboliques.

Si $\Delta\succ 0$ deux caractéristiques réelles traversent chaque point et l'équation est de type hyperbolique Les deux caractéristiques sont :

$$\frac{dy}{dx}=\lambda_+\qquad\text{Et}\qquad\frac{dy}{dx}=\lambda_-\qquad\text{(I.16)}$$

La figure fig. (I.2) illustre schématiquement ce concept, la ligne initiale est notée par Γ_0 où u et v sont connues, en chaque point de cette dernière les deux caractéristiques C_+ et C_- sont extruder dans le plan (x, y) en intégrant l'équation (I.13).

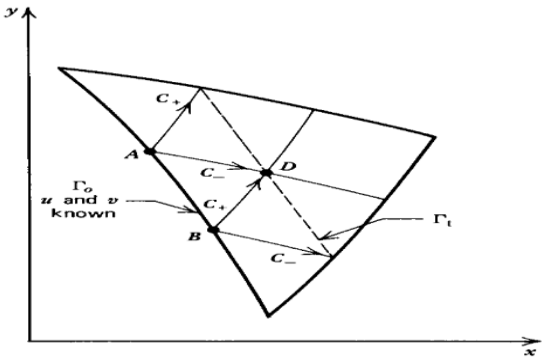

Figure. (I.2) : Résultat de caractéristiques pour un système de deux EDP de premier ordre

VI.4. domaine de dépendance et champ d'influence

La Fig.I.3. Illustre le domaine de dépendance du point P, elle est la région dans le plan (x, y) limitée par les deux caractéristiques extrudées à partir de la ligne initiale Γ_0 et qui se rencontrent au point P. c'est le domaine où la solution au problème de valeurs initiale peut être obtenue.

La Fig.I.4. Illustre le champ d'influence du point Q situé sur la ligne initiale Γ_0 c'est la région dans le plan x, y qui contient tous les points influencés par la valeur initiale au point Q, le champ d'influence est composé de tous les points incluant le point Q dans leur domaines de dépendances, autrement dit c'est la région limitée par les deux caractéristiques extrudées à partir du point Q.

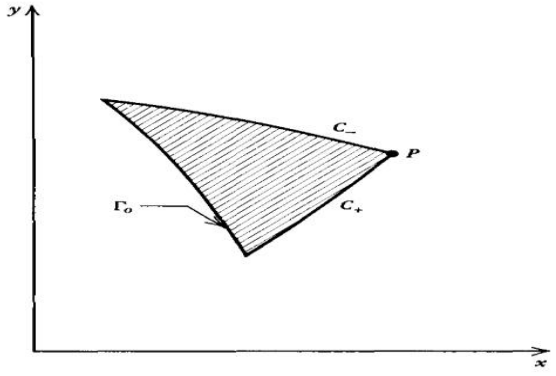

Figure. (I.3) : Domaine de dépendance

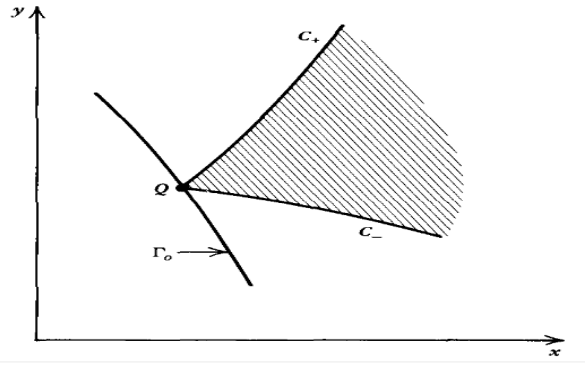

Figure. (I.4) : Champ d'influence

VI.5. Application pour étude d'écoulement irrotationne Supersonique plan et axisymétrique

VI.5.1. les équations gouvernantes

Les équations sont rassemblées ci-dessous.

- L'équation de dynamique des gaz (I.77) :

$$\left(u^2+a^2\right)u_x+\left(v^2-a^2\right)v_y+2uvu_y-\delta\frac{a^2v}{y}=0$$

- La condition de l'irrotationalité (I.79) : $u_y+u_x=0$
- Et la relation de la vitesse de son (I.17) : $a=a(V)=a(u,v)$

VI.5.1.1. L'équations de caractéristiques

La procédure citée avant dans la partie (I.3) pour un système d'équation comprenant les deux équations (I.77) et (I.79). $\sigma_1\left(équation(3.31)\right)+\sigma_2\left(équation(3.33)\right)=0$

Remplaçons les deux équations (II.77) et (II.79) dans l'équation précédente et faisant ressortir σ_1 et σ_2 en facteur, on obtient l'équation (I.18);

$$\begin{aligned}&\sigma_1\left(u^2-a^2\right)\left[u_x+\frac{\sigma_1(2uv)+\sigma_2}{\sigma_1\left(u^2-a^2\right)}u_y\right]\\&+\left(-\sigma_2\right)\left[v_x+\frac{\sigma_1\left(v^2-a^2\right)}{-\sigma_2}v_y\right]-\frac{\sigma_1\delta a^2v}{y}=0\end{aligned}\qquad\text{(I.18)}$$

Si on vérifie le critère de continuité des variables $u(x,y)$ et $v(x,y)$ alors l'équation est valable, et en accord avec l'équation (I.12) on obtient automatiquement les pentes des caractéristiques correspondant à la relation (I.18) qui sont égaux aux coefficients de u_y et de v_y :

$$\lambda=\frac{\sigma_1(2uv)+\sigma_2}{\sigma_1\left(u^2-a^2\right)}\qquad\text{Et}\qquad\lambda=\frac{\sigma_1\left(v^2-a^2\right)}{-\sigma_2}\qquad\text{(I.19)}$$

Substituant l'équation (I.12) dans (I.19), on obtient :

$$\sigma_1\left(u^2-a^2\right)du-\sigma_2dv-\left(\frac{\sigma_1\delta a^2v}{y}\right)dx=0\qquad\text{(I.20)}$$

Il reste maintenant à dériver les expressions par rapport à λ et éliminer les inconnues σ_1 et σ_2 de l'équation (I.20), l'équation (I.19) peut être écrite par :

$$\begin{aligned}\sigma_1\left[\left(u^2-a^2\right)\lambda-2uv\right]+\sigma_2(-1)=0\\ \sigma_1\left(v^2-a^2\right)+\sigma_2(\lambda)=0\end{aligned} \qquad \text{(I.21)}$$

il faut bien que :

$$\begin{vmatrix}\left[\left(u^2-a^2\right)\lambda-2uv\right] & -1\\ \left(v^2-a^2\right) & \lambda\end{vmatrix}=0 \qquad \text{(I.22)}$$

Donc : $$\left(u^2-a^2\right)\lambda^2-2uv\lambda+\left(v^2-a^2\right)=0 \qquad \text{(I.23)}$$

L'équation (I.23) est analogue à l'équation (I.16).

Résolvant l'équation (I.23) pour la valeur de λ pour que l'équation (I.20) soit valable, on obtient l'équation (I.24):

$$\lambda_{\pm}=\left(\frac{dy}{dx}\right)_{\pm}=\frac{uv\pm a^2\sqrt{M^2-1}}{u^2-a^2} \qquad \text{(I.24)}$$

L'équation (I.24) produit deux résultats différents pour λ notés en + et en – correspondant respectivement aux signes positif et négatif précédant la racine carrée.

Ces deux équations différentielles ordinaires représentent deux courbes dans le plan (x,y) qui sont bien entendu les deux caractéristiques de ce modèle de calcule.

Elles ne sont réels que lorsque le nombre de mach $M>1$.

A partir des géométries présentées sur les figures (I.5) et (I.6) , une autre forme alternative de l'équation (I.24) peut être obtenue en exprimant u et v en fonction du module de vitesse V et de l'angle de lignes de courants ou de l'écoulement, et on déduit aussi une expression de M en fonction de l'angle de Mach α.

$$u=V\cos\theta \quad ; \quad v=V\sin\theta \quad ; \quad \theta=\tan^{-1}\left(\frac{v}{u}\right) \qquad \text{(I.25)}$$

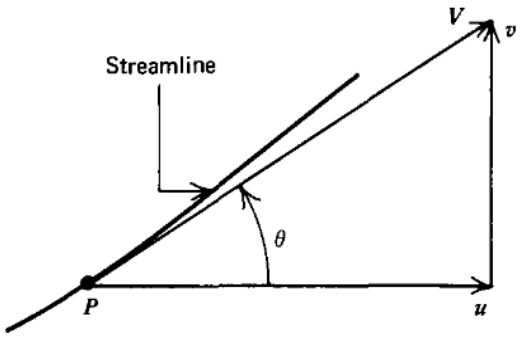

Figure. (I.5) : Relation entre u, v et V,θ

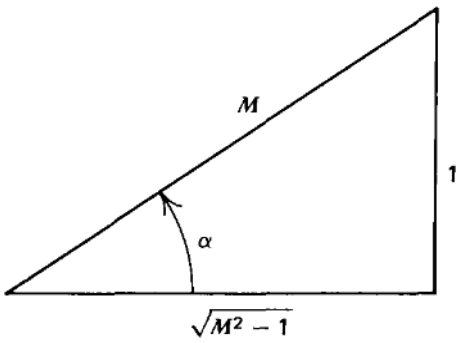

Figure. (I.6) : Relation entre α et M

A partir de la figure (I.6) on peut obtenir les relations suivantes:

$$\alpha = \sin^{-1}\left(\frac{1}{M}\right) \quad ; \; M = \frac{V}{a} = \frac{1}{\sin\alpha} \quad ; \quad \sqrt{M^2 - 1} = \cot\alpha \qquad \text{(I.26)}$$

Le remplacement des équations (I.45) et (I.46) dans l'équation (I.24) permet d'aboutir à l'équation (I.27):

$$\left(\frac{dy}{dx}\right)_{\pm} = \lambda_{\pm} = \tan\left(\theta \pm \alpha\right) \qquad \text{(I.27)}$$

La figure (I.7) illustre la géométrie des caractéristiques C_+ et C_- dans le plan (x,y) elle montre clairement aussi que les deux caractéristiques, son bien symétriques par rapport à la ligne de courant, faisant respectivement les angles α_+ et α_- avec l'angle θ de la ligne de courant, qui est l'angle formé par la tangente a la ligne de courant au point P avec l'axe x.

La ligne caractéristique est la courbe, le long où les informations dans un écoulement sont propagées d'un point amant vers un point avale.

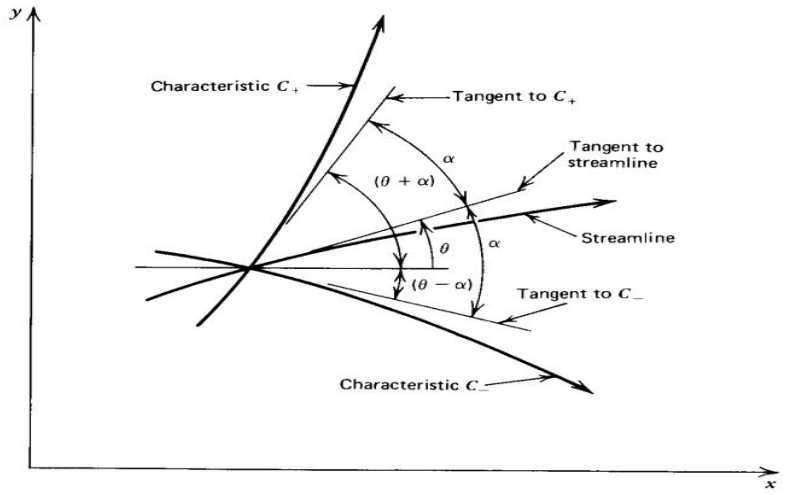

Figure. (I.7) : Les caractéristiques d'un écoulement Bidimensionnel Irrotationnel et supersonique

VI.5.1.2. L'équation de compatibilité

L'équation (I.20) est l'équation de compatibilité pour un écoulement supersonique. Elle exprime la relation entre u et v le long de la caractéristique (ligne de mach).

Pour appliquer l'équation (I.20), les paramètres σ_1 et σ_2 doivent être éliminés, ce qui est fait en résolvant l'équation (I.21) pour σ_2 en fonction de σ_1.

Donc :

$$\sigma_2 = \sigma_1\left[\left(u^2 - a^2\right)\lambda - 2uv\right]$$

$$\sigma_2 = -\sigma_1 \frac{v^2 - a^2}{\lambda} \qquad \text{(I.28)}$$

En remplaçant l'équation (I.28) dans l'équation (I.20) et divisons par σ_1, on obtient finalement l'équation de compatibilité pour ce type d'écoulement :

$$\left(u^2 - a^2\right)du_{\pm} + \left[2uv - \left(u^2 - a^2\right)\lambda_{\pm}\right]dv_{\pm} - \left(\frac{\delta a^2 v}{y}\right)dx_{\pm} = 0 \qquad \text{(I.29)}$$

En remplaçant l'équation (I.27) dans l'équation (I.29) on obtient une autre forme pour l'équation de compatibilité présenté ci-dessous.

$$du_{\pm} + \lambda_{\mp}dv_{\pm} - \delta\left(\frac{a^2 v}{y}\right)dx_{\pm} = 0 \qquad \text{(I.30)}$$

VI.6. Mise en œuvre numérique de la méthode des Caractéristiques

VI.6.1. procédure d'intégration numérique

La figure (I.8) illustre comment la caractéristique passante par le point 1 et celle passante par le point 2 s'intercepte au point 4, cette procédure concerne une point interne, une petite modification et le processus sera applicable sur un point paroi et un point sur l'axe.

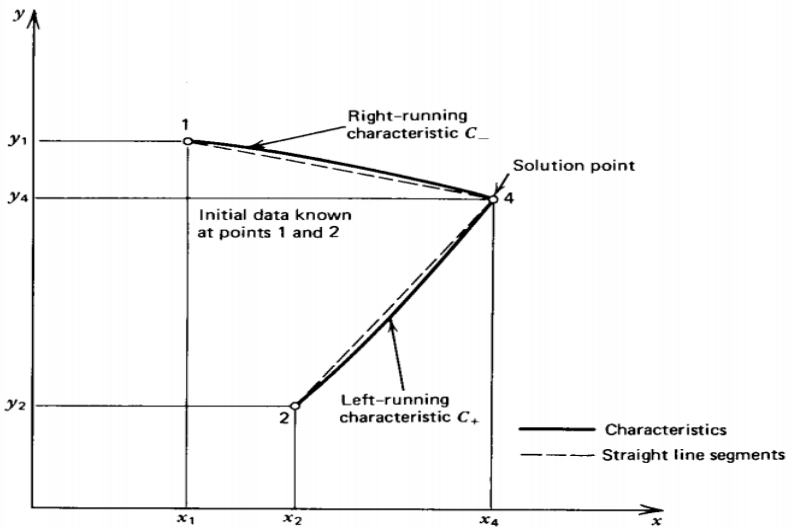

Figure. (I.8) : schéma de calcul par différences finis pour La méthode des caractéristiques

Les étapes de bases de cette méthode sont présentées ci-dessous.

Considérons l'équation différentielle ordinaire suivante :

$$\frac{dy}{dx} = f\left(x, y\right) \qquad \text{(I.31)}$$

Qui peut être écrite sous la forme ; $dy = f\left(x, y\right)dx$ (I.32)

Le problème est d'intégrer l'équation (I.32) numériquement à partir d'un point de départ connu désigné comme $\left(x_i, y_i\right)$ où $y_i = y\left(x_i\right)$.

La valeur prédite de la solution à $x_{i+1} = x_i + \Delta x$, notée par $y^0\left(x_i + \Delta x\right) = y^0{}_{i+1}$ est obtenue à partir de l'algorithme de prédiction d'Euler suivant ;

$$y^0{}_{i+1} = y_i + f\left(x_i, y_i\right)\Delta x \qquad \text{(I.33)}$$

L'exactitude de la solution par la méthode d'Euler est obtenue en employant y_i et $y^0{}_{i+1}$ pour estimer la valeur de $y_{i+1/2} = y\left(x_i + \Delta x/2\right)$, et en remplaçant $f\left(x_i, y_i\right)$ dans l'équation (I.33) par $f\left(x, y\right)$ déterminé au point central de l'intervalle, donc;

$$y^{1}_{i+1} = y_i + f\left(x_i + \frac{\Delta x}{2}, \frac{y_i + y^{0}_{i+1}}{2}\right)\Delta x \qquad \text{(I.34)}$$

Où $y^{1}_{i+1} = y^{1}\left(x_i + \Delta x\right)$ est la valeur correcte de la solution au point x_{i+1}. L'équation (I.34) est l'algorithme de correction. La méthode de prédiction correction peut être plus exacte en remplaçant la valeur de y^{0}_{i+1} par la valeur de y^{1}_{i+1} dans l'équation (I.34) pour obtenir le résultat le plus exact y^{2}_{i+1}, ou le signe 2 représente la deuxième application du correcteur, et la formule suivante représente le cas générale du correcteur :

$$y^{n}_{i+1} = y_i + f\left(x_i + \frac{\Delta x}{2}, \frac{y_i + y^{n-1}_{i+1}}{2}\right)\Delta x \qquad \text{(I.35)}$$

Où y^{n}_{i+1} est la valeur de y après n applications du correcteur l'équation (I.35) est appelé l'algorithme de prédiction correction d'Euler avec itérations.

Généralement pour la stabilité de cette méthode numérique on limite toujours la valeur de Δx par une borne supérieur, mais dans notre cas d'étude d'un écoulement isentropique cette dernière et tellement grande que la stabilité n'est pas affecté.

VI.6.2. la méthode directe et la méthode inverse

Il existe deux solutions possibles pour intégrer les équations de caractéristiques et celles de compatibilité ; la *méthode direct* et la *méthode inverse*.

La figure(Fig.I.9) illustre schématiquement la méthode directe et la figure(I.10) montre la méthode inverse.

L'avantage de la méthode directe c'est que le point 1 et 2 et leurs propriétés sont connus sans procéder à une interpolation. Quand a la méthode inverse, elle a un petit avantage dans la logique de calcule, toutefois du fait que le nombre et la position des point solutions doivent être déterminés à l'avance. En générale la méthode directe est la plus précise des deux méthodes.

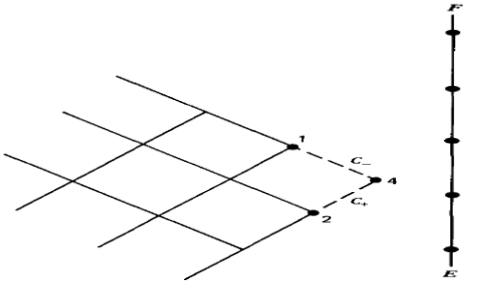

Figure. (I.9) : La méthode directe

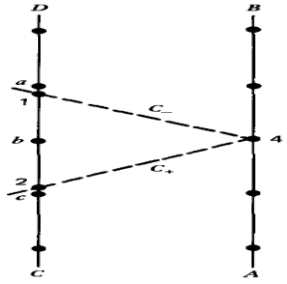

Figure. (I.10) : La méthode inverse

VI.6.3. Les équations aux différences finis

La figure (I.8) illustre schématiquement le maillage pour différences finis pour la détermination des propriétés de l'écoulement dans un point interne.

$$\Delta y_{\pm} = \lambda_{\pm} \Delta x_{\pm} \qquad \text{(III.36)}$$

$$Q_{\pm} \Delta u_{\pm} + R_{\pm} \Delta v_{\pm} - S_{\pm} \Delta x_{\pm} = 0 \qquad \text{(III.37)}$$

$$\lambda_{\pm} = \tan(\theta \pm \alpha) \qquad \text{(III.38)}$$

$$Q = u^2 - a^2 \qquad \text{(III.39)}$$

$$R = 2uv - (u^2 - a^2)\lambda \qquad \text{(III.40)}$$

$$S = \delta \frac{a^2 v}{y} \qquad \text{(III.41)}$$

Les relations ci-dessus présente les équations en différences finis des équations de caractéristiques et de compatibilités où le signe + et – représentent respectivement les caractéristiques montantes et les caractéristiques descendantes.

VI.6.4. Procédure de calcul pour un point interne

En utilisant les équations (I.36) dans leur formes éclatées on trouve ce qui suit ;

$$y_4 - \lambda_+ x_4 = y_2 - \lambda_+ x_2 \qquad \text{(I.42)}$$

$$y_4 - \lambda_- x_4 = y_1 - \lambda_- x_1 \qquad \text{(I.43)}$$

Les équations (I.42) et (I.43) forment un système à deux équations d'où on peut tirer facilement les inconnues x_4 et y_4, ou les pentes sont présentées ci-dessous:

$$\lambda_+ = \tan(\theta_+ + \alpha_+) \qquad \text{(I.44)}$$

$$\lambda_- = \tan(\theta_- + \alpha_-) \qquad \text{(I.45)}$$

Où ; $$\theta_{\pm} = \tan^{-1}\left(\frac{v_{\pm}}{u_{\pm}}\right) \qquad \text{(I.46)}$$

$$V_{\pm} = \sqrt{u_{\pm}^2 + v_{\pm}^2} \qquad \text{(I.47)}$$

$$a_{\pm} = a\left(V_{\pm}\right) \qquad \text{(I.48)}$$

$$M_{\pm} = \frac{V_{\pm}}{a_{\pm}} \qquad \text{(I.49)}$$

$$\alpha_{\pm} = \sin^{-1}\left(\frac{1}{M_{\pm}}\right) \qquad \text{(I.50)}$$

Par conséquent les valeurs de $\theta_{\pm}$, $\alpha_{\pm}$ et de $\lambda_{\pm}$ peuvent être déterminées en spécifiant $u_{\pm}$ et $v_{\pm}$.

Les équations de comptabilités (I.37) peuvent aussi être écrites comme suit :

$$Q_{+}u_4 + R_{+}v_4 = T_{+} \qquad \text{(I.51)}$$

$$Q_{-}u_4 + R_{-}v_4 = T_{-} \qquad \text{(I.52)}$$

Tel que ;

$$T_{+} = S_{+}\left(x_4 - x_2\right) + Q_{+}u_2 + R_{+}v_2 \qquad \text{(I.53)}$$

$$T_{-} = S_{-}\left(x_4 - x_1\right) + Q_{-}u_1 + R_{-}v_1 \qquad \text{(I.54)}$$

De même manière (I.51) et (I.52) forment un système à deux inconnues qu'on peut résoudre pour u_4 et v_4, où les coefficients sont listés ci-dessous ;

$$Q_{+} = \left(u_{+}^2 - a_{+}^2\right) \qquad \text{(I.55)}$$

$$R_{+} = \left(2u_{+}v_{+} - Q_{+}\lambda_{+}\right) \qquad \text{(I.56)}$$

$$S_{+} = \delta\frac{a_{+}^2 v_{+}}{y_{+}} \qquad \text{(I.57)}$$

$$Q_{-} = \left(u_{-}^2 - a_{+}^2\right) \qquad \text{(I.58)}$$

$$R_{-} = \left(2u_{-}v_{-} - Q_{-}\lambda_{-}\right) \qquad \text{(I.59)}$$

$$S_{-} = \delta\frac{a_{-}^2 v_{-}}{y_{-}} \qquad \text{(I.60)}$$

Par conséquent les coefficients $Q_{\pm}$, $R_{\pm}$ et $T_{\pm}$ peuvent être déterminées en spécifiant uniquement $u_{\pm}$, $v_{\pm}$ et $y_{\pm}$.

Pour l'algorithme de prédiction d'Euler, les valeurs de $u_{\pm}, v_{\pm}$ et $y_{\pm}$ sont données par ;

$$u_{+} = u_2 \qquad v_{+} = v_2 \qquad \mathrm{y}_{+} = y_2 \tag{I.61}$$

$$u_{-} = u_1 \qquad v_{-} = v_1 \qquad \mathrm{y}_{-} = y_1 \tag{I.62}$$

Et pour l'algorithme de correction d'Euler, les valeurs de $u_{\pm}, v_{\pm}$ et $y_{\pm}$ sont données par ;

$$u_{+} = \frac{u_2 + u_4}{2} \qquad v_{+} = \frac{v_2 + v_4}{2} \qquad \mathrm{y}_{+} = \frac{y_2 + y_4}{2} \tag{I.63}$$

$$u_{-} = \frac{u_1 + u_4}{2} \qquad v_{-} = \frac{v_1 + v_4}{2} \qquad \mathrm{y}_{-} = \frac{y_1 + y_4}{2} \tag{I.64}$$

Les valeurs moyennes de $\theta_{\pm}, V_{\pm}, a_{\pm}, M_{\pm}$ et de $\alpha_{\pm}$ sont obtenues en injectant les valeurs ci-dessus dans les équations (I.55) à (I.60). en résolvant les même équations qu'avant pour une deuxième fois on obtient les solutions corrigées suivantes $x_4^1, y_4^1\ u_4^1$ et v_4^1.

On peut répéter la même démarche prescrite ci-dessus. Et ça en utilisant l'algorithme de prédiction correction d'Euler avec itération (I.35). On arrête le processus de calcule à n itérations, lorsque on atteint la convergence décrite par la relation logique suivante :

$$\left| P^{n} - P^{n-1} \right| \leq (\text{tolérence spécifié}) \tag{I.65}$$

Où P représente x_4, y_4, u_4 et v_4.

Dans ce cas le rapport v_2/y_2 est approximé au rapport v_1/y_1 pour le prédicteur, par contre le correcteur S est basé sur la valeur moyenne de y_{+} et de v_{+} qui ne sont pas nulles.

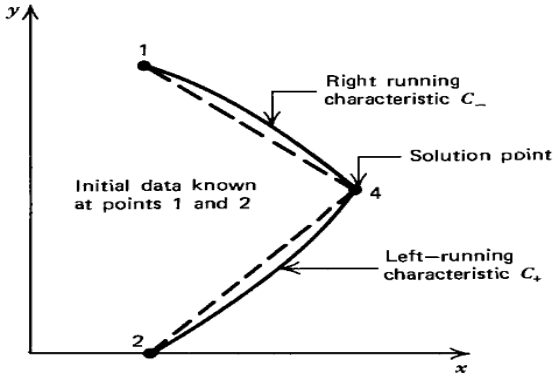

Figure. (I.11) : Point intérieur sur l'axe de symétrie

VI.6.5. Procédure de calcule pour un point paroi direct

Sur la paroi, la direction du vecteur vitesse d'écoulement doit être exactement égale à la pente de la paroi certaines modifications doivent être portées sur la procédure de calcul applicable sur un point interne.

Par conséquent seulement une équation de caractéristique et une équation de compatibilité sont valables pour déterminer la position et les propriétés du point paroi direct , alors

$y = y(x)$ Spécifié (I.66) $\frac{dy}{dx} = \tan\theta = \frac{v}{u}$ Spécifié (I.67)

La figure (I.12) montre le cas où la paroi est sur la moitié supérieure du plan (x, y).si le point paroi direct est déterminé, les équations (I.66) et (I.67) doivent être résolues simultanément avec les équations (I.42) et (I.51) qui sont valables le long de la caractéristique C_+.

Si la paroi est sur la demi-partie inférieure du plan (x, y), là on utilise les équations (I.43) et (I.52).

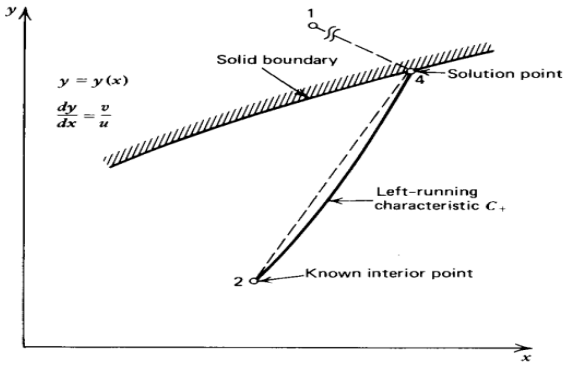

Figure. (I.12) : Point paroi direct

VI.6.6. Procédure de calcule pour un point paroi inverse

La figure (I.13), illustre schématiquement la situation où les valeurs initiales au point 1 et 3 sont connues d'après des calculs précédents. Le point 4 est le point solution pré-spécifié, le point 2 est le lieu d'intersection de la caractéristique C_+ passante par le point 4 et la caractéristique 13. Du fait que la position du point 4 (x_4, y_4) est pré spécifié. Ainsi les équations de compatibilités (I.51) et (I.67) peuvent être utilisées pour calculer u_4 et v_4 si la position et les propriétés au point 2 sont connues.

Le point 2 est localisé en déterminant la position de l'intersection de la caractéristique 24 (I.42) avec la caractéristique 13 (I.43), les propriétés au point 2 sont déterminés par une interpolation linéaire entre les points 1 et 3.

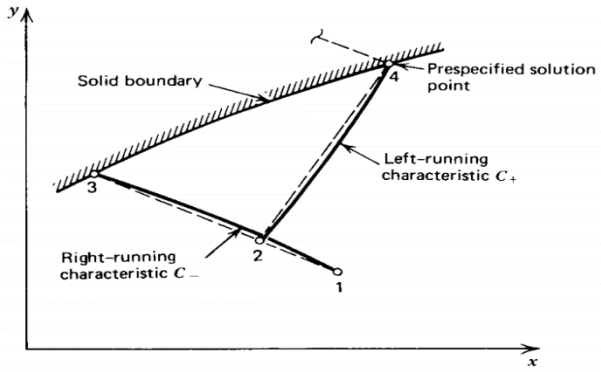

Figure. (I.13) : Point paroi inverse

Comme la pente de la caractéristique 24 dépend des paramètres inconnus des points 2 et 4, l'algorithme de prédiction correction d'Euler est utilisé pour localiser le point 2.

L'étape de prédiction primaire est pour déterminer les propriétés de l'écoulement au point 4, la seconde méthode de prédiction correction est appliquée pour déterminer la position du point 2.

Premièrement les propriétés au point 2 sont supposée égales à celles du point 3 et la pente λ_+ de la caractéristique 24 est déterminée.

Deuxièmement en utilisant cette valeur de la pente λ_+, la position du point 2 sur la caractéristique 24 qui passe à travers le point 4 est déterminée.

Troisièmement les propriétés au point 2 sont obtenues à l'aide d'une interpolation linéaire entre les points 1 et 3. L'étape de correction est après aborder en utilisant la valeur de u_4 et v_4 à partir de la troisième étape, et le point 2 est relocalisé. La partie correction est répétée itérativement jusqu'à ce que la position du point 2 converge à une tolérance désirée.

Apres avoir déterminé la position du point 2, une étape de prédiction primaire est faite pour déterminer les propriétés du point 4, l'étape de correction primaire est utilisé justement avec le point 2 repositionné avec l'étape de prédiction correction secondaire discutée ci-dessus. Maintenant la pente de la caractéristique 24 est basée sur la valeur de u et v entre les points 2 et 4, des itérations peuvent être appliquées pour l'étape de correction primaire.

VI.6.7. Procédure de calcule pour un point sur l'axe de symétrie

Pour un écoulement bidimensionnel axisymétrique, l'axe x est l'axe de symétrie. La figure (I.14) présente schématiquement le point axial (point 4). Si le point 1 est un point sur la caractéristique C_- passante sur le point 4, alors comme illustré sur la figure (I.14), le point 2 en dessous de l'axe de symétrie peut être définit comme une image du point 1. Le point 4 est alors analogue a un point interne, par conséquent la procédure développée pour un point interne peut être appliquée pour calculer les propriétés d'un point sur l'axe. Toutefois, dans ce cas $y_4 = v_4 = \theta_4 = 0$, ce qui simplifie la procédure de calcul. Pour ce qui suit, seule la caractéristique 14 est employée et les équations (I.43) et (I.52) sont résolues simultanément avec les données $y_4 = v_4 = \theta_4 = 0$.

Pour un écoulement axisymétrique, le coefficient S est donné par l'équation (I.41) est indéterminé sur l'axe de symétrie, mais on ne trouve pas de problèmes durant le calcul du prédicteur, parce que le prédicteur calcule S au point 1, de plus le correcteur calcul S en se basant sur les valeurs moyennes de y_+ et de v_+ qui sont différents de zéro.

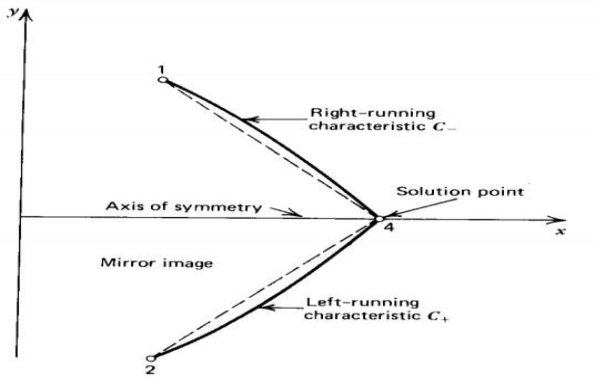

Figure. (I.14) : Point axiale

VI.7. Application pour études de tuyères a géométries connues

La figure (I.15) illustre schématiquement la configuration d'une tuyère convergente-divergente à géométrie connue.

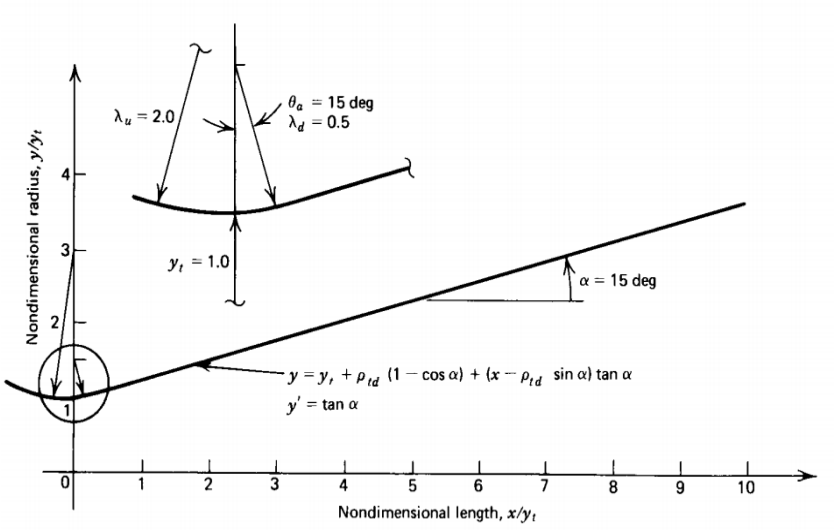

Figure. (I.15) : tuyère supersonique a divergent conique

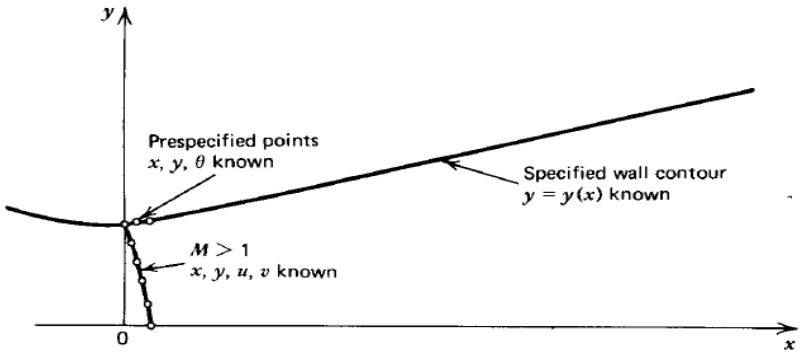

Figure. (I .16) : Schéma de la ligne initiale et des points préspicifiés

Dans la partie divergente initiale amant au col appelé zone d'expansion, le gradient de propriétés thermodynamiques est très large, alors le maillage dans cette région doit être raffiné d'avantage, et pour le faire on opte pour la méthode inverse qui nous offre l'opportunité d'un raffinage personnalisé. Avant d'arriver à la paroi il faut d'abord calculer les point internes commençant à partir du point initiale situé sur l'axe noté 2 représenté sur la figure (I.17).

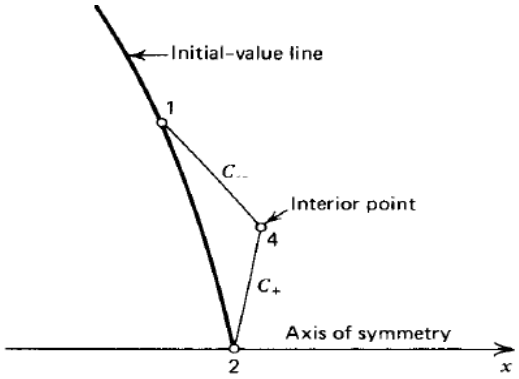

Figure. (I.17) procédure de calcule d'un point interne
À partir de la ligne initiale

Les propriétés de l'écoulement sur le point solution noté 4 sur la même figure sont alors déterminées. On procède ensuite au calcul du point sur l'axe noté 4 sur la figure (I.18) à partir du point 1 qui est lui-même le point interne calculé avant.

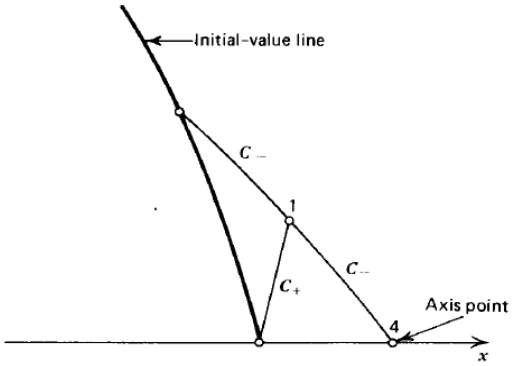

Figure. (I.18) : procédure de calcule d'un point axiale

Maintenant que la ligne initiale et son voisinage est raffiné, il nous reste donc qu'à déterminer les propriétés de l'écoulement dans toute la région d'expansion initiale en extrudant toutes les caractéristiques descendantes provenant des points préspicifiés en utilisant le processus du point interne représenté sur la figure (I.19).

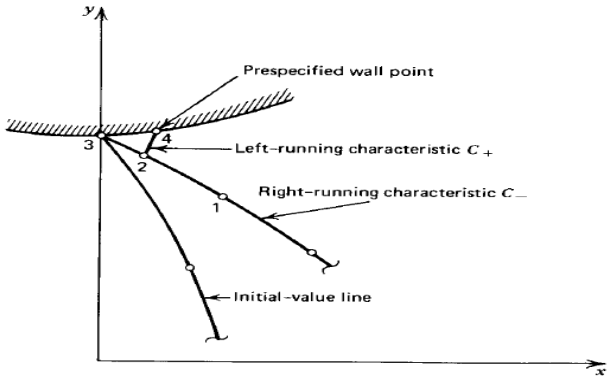

Figure. (I.19) : procédure de calcule d'un point inverse

La figure (I.20) montre le résultat de maillage de la zone d'expansion initiale.

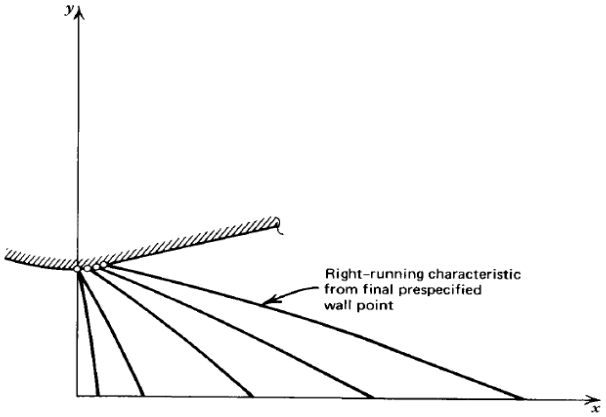

Figure. (I.20) : procédure de calcul de la zone d'expansion initiale

Pour cela on peut sans problèmes utiliser la méthode directe montrée sur la figure (I.21) et expliquée dans la partie (I.5.6) pour trouver la distribution des propriétés le long de cette zone.

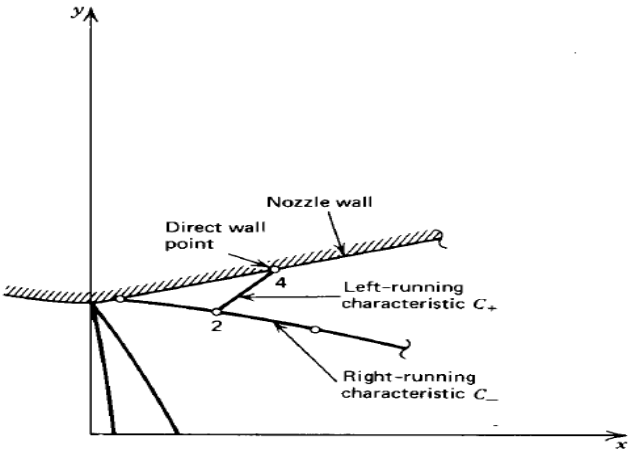

Figure. (I.21) : procédure de calcule d'un point direct

On applique la même logique de calcul pour toute la partie divergente, les résultats d'application de cette procédure sont montrés sur la figure (I.22).

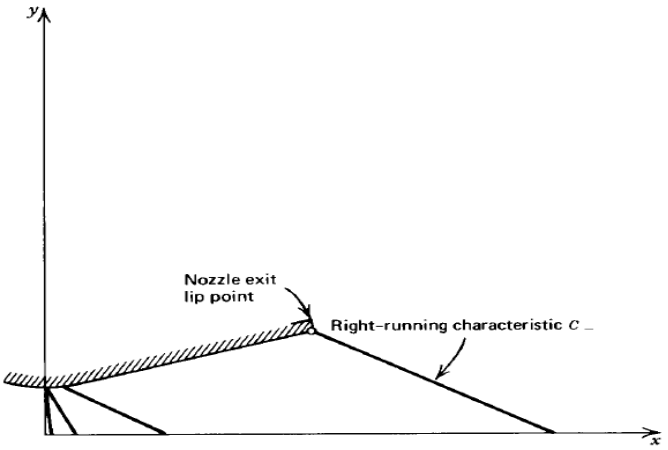

Figure. (I.22) : Procédure de calcul de la partie divergente

Au niveau du point d'attache ou le rayon de courbure avale au col se joint tangentiellement a la partie divergente. À ce point le contour du profil et sa pente sont continués, mais la courbure de la paroi est discontinue. Cette discontinuité généralement génère des ondes à faible compression. Quand deux caractéristiques de même famille se croisent une onde de choc oblique est formée comme le montre la figure (I.23)

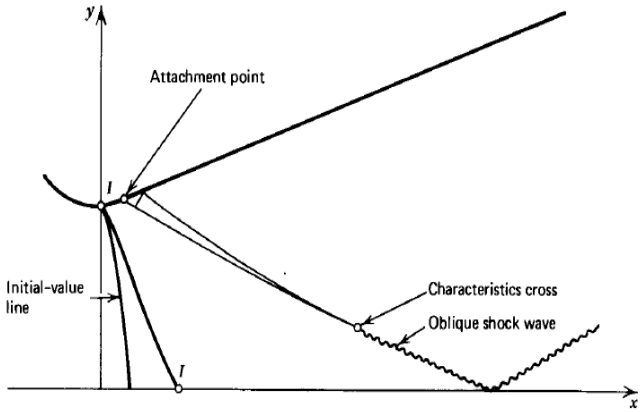

Figure. (I.23) : Croisement de deux caractéristiques pour former une onde de choque oblique

La négligence de celle-ci pendant les calculs peut engendrer de sérieux erreurs, Plusieurs méthodes ont étés proposées, parmi ces méthodes on trouve celles présentée sur la figure (I.24)

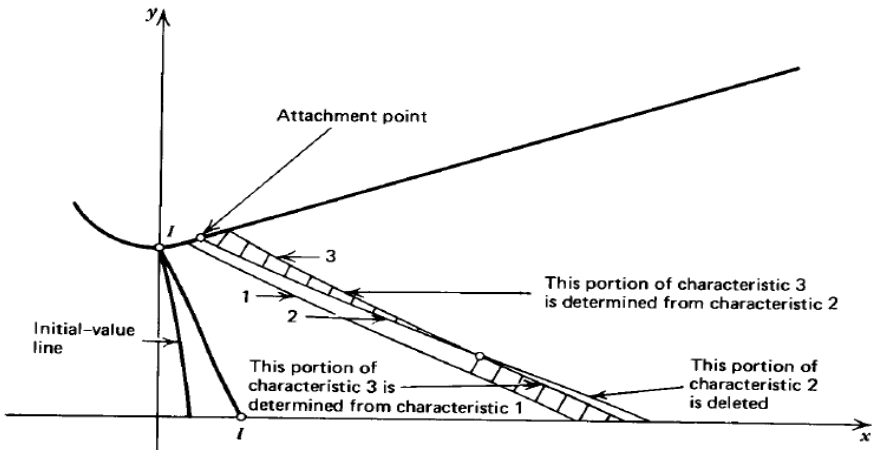

Figure. (I.24) : méthode de correction de calcule dans le cas de croisement de Deux caractéristiques de même famille

Cette procédure est très simple elle est divisée en trois étapes :

- ✓ La première consiste à localier le lieu d'intersection des deux caractéristiques par exemple 2 et 3 prestées sur la figure (4.24)
- ✓ la deuxième étape est de supprimer la partie qui vient après l'intersection de la caractéristique 2
- ✓ en fin la dernière étape est de calculer la partie inferieur après l'intersection de la caractéristique 3 à partir de la caractéristique 1.

VI.8. Application pour la conception de tuyères supersoniques

VI.8.1. Cas d'une tuyère axisymétrique

La figure (I.25) montre les données initiales nécessaires pour le démarrage de calculs dans le but de dimensionner la partie divergente de cette tuyère.

La courbe AB est le contour d'admission, BC est la courbe qui représente le contour de la zone d'expansion initiale, par contre BD est la ligne initiale supersonique.

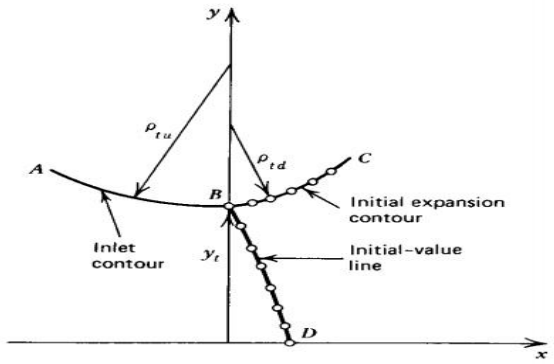

Figure. (I.25) : Données initiales du problème

L'étape qui suit c'est bien de prendre en considération les données initiales sur paroi et sur la ligne initiale pour commencer le calcul des paramètres de l'écoulement dans la région d'expansion initiale représenté sur la figure (I.26) en utilisant le même algorithme de calcul que celui présenté dans la partie (I.6).

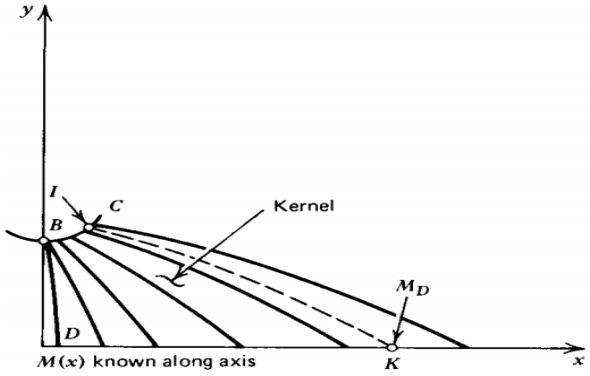

Figure. (I.26) : Localisation du mach de désigne sur l'axe X

Les paramètres de l'écoulement dans la zone d'expansion initiale son entièrement déterminés à partir des données de la ligne initiale est les conditions posées sur la paroi de cette zone, elle est appelée *zone de karnel.*

La distribution du nombre de mach sur l'axe X, $M_{CL} = M(x)$ est déterminée à partir de cet écoulement, comme représenté sur la figure (I.26), le mach de design M_D est ensuite localisé sur l'axe X., les paramètres de l'écoulement sont ensuite déterminés sur la caractéristique sortante à partir de M_D noté IK, en faisant une simple interpolation entre les paramètres des deux caractéristiques sortantes des points amant et aval au point K sur l'axe de symétrie.

Dans le cas où on exige que l'écoulement à la sortie de la tuyère soit uniforme et parallèle a $M = M_D$ et $\theta = 0$, le cas par exemple des tuyères destinées à usage en soufflerie, alors dans cette région, toute les caractéristiques doivent êtres des lignes droites formant un angle de $\alpha_D = \sin^{-1}\left(1/M_D\right)$ comme le montre la figure (I.27).

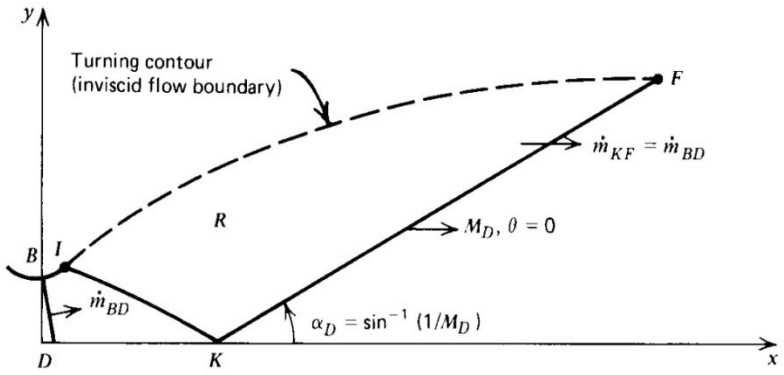

Figure. (I.27) : les conditions imposées à la sortie de la tuyère

Ainsi la caractéristique droite faisant l'angle α_D par rapport à l'axe de symétrie est extrudée à partir du point k jusqu'au point F. le débit massique qui traverse la ligne KF, doit être égal au débit calculé qui traverse la ligne initiale BD sur la figure (I.27) .

Maintenant le problème est de déterminer les paramètres de l'écoulement à l'intérieur de la région R et au même temps le contour idéal de la paroi du divergente notée IF qui peut bien évidement nous délivrer ce que on a exigé à la sortie de la tuyère.

La solution à ce problème n'est pas vraiment loin de ce qu'on a fait pour l'étude de tuyère à contour donné. Ce qu' il faut appliquer c'est bien la procédure de calcul d'un point interne et puis la généraliser sur tout le domaine de la région R en commençant par la ligne KF comme montré sur la figure (I.28).

en faisant monter la caractéristique à partir du point situé sur la ligne KF on calcule au même temps à chaque point interne de coordonnées x, y le débit massique qui traverse la tuyère entre l'axe de symétrie et le point au quel on est abouti en utilisant la relation suivante.

$$d\dot{m} = \rho V.dA = \rho u \pi y dy \qquad \text{(I.67)}$$

On sait bien que le débit au point sur l'axe est nulle, alors l'intégrale de l'équation(I.67) est donnée par la relation suivante;

$$\dot{m} = 2\pi \int_0^{y_L} \rho u y dy \qquad \text{(I.68)}$$

Pendant la procédure de calcul du débit le long de la caractéristique en faisant une interpolation entre les deux derniers points calculés on trouve la position du point paroi.

Cette procédure est répétée pour chaque point sur la ligne KF jusqu'à ce qu'on trouve une caractéristique qui comporte un seul point qui définit le point F, là on arrête la calcule et on dit que le profil est définit.

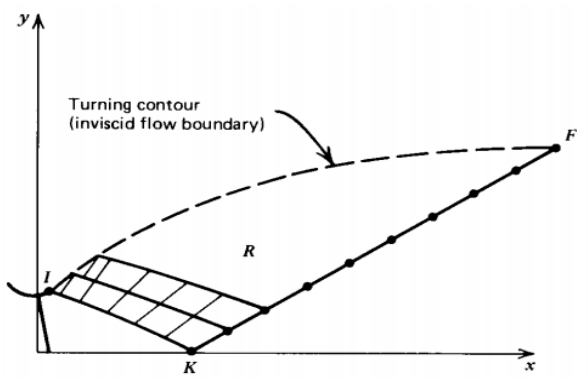

Figure. (I.28) : détermination du contour d'une tuyère supersonique

VI.8.2. Cas de tuyères planes

La figure (I.29) montre schématiquement les caractéristiques droites sortant d'un point sélectionné sur la ligne IK, la limite de chaque ligne caractéristique est déterminer par le calcul du débit massique comme décrit dans la partie (I.7.1) et le contour de la tuyère est déterminé sans intégration numérique.

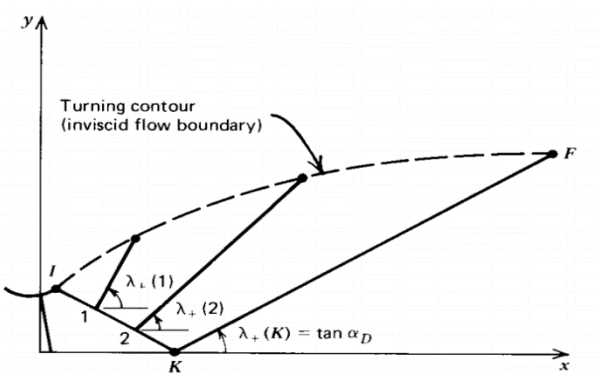

Figure. (I.29) : Détermination du contour d'une tuyère supersonique plane

VI.9. Conclusion

La méthode des caractéristiques présente son importance dans le dimensionnement des tuyères supersoniques. En spécifiant le nombre de mach à la sortie ou le report de pression ou encore le poussée désirée, et les paramètres thermodynamiques d'arrêt du fluide en question et les dimensions du col, le profil de la tuyère est facilement obtenue. Pour l'étude d'écoulement interne, on a choisir entre deux méthodes, celle des caractéristiques et celle des éléments finis.

.

VII. CHAPITRE II

LOGICIEL DE PROGRAMMATION ET DE SIMULATION

VII.1. Introduction

Dans ce chapitre, nous explorerons le domaine des logiciels multi physiques qui jouent un rôle essentiel dans la simulation et l'analyse de problèmes complexes impliquant plusieurs phénomènes physiques. Parmi ces logiciels, nous trouverons des outils puissants tels qu'OpenFOAM, COMSOL, Gambit, Nastran, PATRAN, SimScale, Converge CFD, Sim center, et SUE CFD. Chacun de ces logiciels offre une approche unique pour modéliser et résoudre des systèmes multidisciplinaires, permettant ainsi aux ingénieurs et aux chercheurs d'étudier des problèmes impliquant des interactions entre différentes forces et énergies. En parallèle, nous aborderons également les langages de programmation couramment utilisés dans ce domaine, tels que Java, Fortran, Matlab, C, C++, Ruby, Pascal, Ada et Delphi. Ces langages offrent des fonctionnalités avancées pour le développement de modèles numériques et d'algorithmes spécifiques aux simulations multi physiques.

Dans l'ensemble, ces logiciels sont largement utilisés dans les domaines technologique, industriel et de la recherche scientifique. Ces logiciels permettent aux ingénieurs, aux chercheurs et aux scientifiques de modéliser, simuler et analyser des systèmes complexes impliquant plusieurs phénomènes physiques.

Dans le domaine technologique, ces logiciels sont utilisés pour la conception et l'optimisation de produits et de systèmes, tels que les avions, les voitures, les turbines, les équipements électroniques, les dispositifs médicaux, etc. Ils permettent de simuler et d'évaluer le comportement des produits sous diverses conditions de fonctionnement, ce qui permet d'identifier les problèmes potentiels, d'optimiser les performances et de réduire les coûts de développement.

Dans l'industrie, ces logiciels sont utilisés pour améliorer les processus de fabrication, la gestion de la chaîne d'approvisionnement, la sécurité et l'efficacité énergétique. Par exemple, ils peuvent être utilisés pour simuler la circulation des fluides dans une usine, optimiser la distribution de la chaleur dans un système de refroidissement ou analyser les contraintes mécaniques sur les équipements industriels.

En recherche scientifique, ces logiciels jouent un rôle clé dans la compréhension des phénomènes physiques et dans la modélisation de systèmes complexes. Ils sont utilisés dans des domaines tels que la physique des matériaux, la bio-ingénierie, la climatologie, l'astrophysique, la géologie, etc. Ces logiciels permettent aux chercheurs d'étudier et de prédire le comportement de systèmes réels, d'explorer de nouvelles idées et de valider leurs hypothèses grâce à des simulations numériques.

VII.2. JAVA (LANGAGE)

Java est un langage de programmation orienté objet et une plateforme informatique développée par Sun Microsystems (maintenant Oracle Corporation). Java est un langage de programmation polyvalent qui a été conçu pour être portable, c'est-à-dire qu'il peut fonctionner sur différentes plateformes, telles que Windows, macOS, Linux, etc. Il est basé sur le concept de "Write Once, Run Anywhere" (WORA), ce qui signifie que le code Java peut être écrit une fois et exécuté sur n'importe quelle plateforme compatible Java. Java est utilisé dans divers domaines, notamment le développement d'applications de bureau, le développement d'applications web, les applications mobiles, les systèmes embarqués, les jeux vidéo, l'intelligence artificielle, l'Internet des objets (IoT), etc. Il est largement utilisé dans l'industrie et est considéré comme l'un des langages les plus populaires et les plus utilisés.

Avantages :

- Portabilité : L'un des principaux avantages de Java est sa portabilité. Le code Java peut être exécuté sur différentes plateformes sans nécessiter de modifications importantes, ce qui réduit les coûts de développement et facilite le déploiement.
- Sécurité : Java offre un environnement sécurisé grâce à son modèle de sécurité robuste qui protège contre les vulnérabilités telles que les débordements de tampon, les accès non autorisés à la mémoire, etc. De plus, il intègre une gestion automatique de la mémoire (garbage collection) pour prévenir les fuites de mémoire.
- Large écosystème : Java dispose d'une vaste bibliothèque standard (Java API) qui fournit des fonctionnalités prêtes à l'emploi pour le développement d'applications. De plus, il existe une grande communauté de développeurs, de forums et de ressources en ligne, ce qui facilite l'apprentissage et le support.

Inconvénients :

- Performances : Par rapport à certains langages de programmation compilés tels que C++, Java peut être légèrement moins performant en raison de l'exécution de code sur une machine virtuelle (Java Virtual Machine - JVM). Cependant, les performances se sont améliorées au fil des versions.
- Consommation de mémoire : L'exécution d'applications Java peut nécessiter une quantité relativement élevée de mémoire en raison de la surcharge de la JVM et de la gestion automatique de la mémoire. Cela peut être un inconvénient pour les applications nécessitant une utilisation efficace de la mémoire.

VII.3. FORTRAN

Fortran (Formula Translation) est un langage de programmation utilisé principalement dans le domaine scientifique et de l'ingénierie pour le calcul scientifique et les simulations numériques. Fortran est l'un des plus anciens langages de programmation encore utilisés aujourd'hui. Il a été développé dans les années 1950 pour faciliter la programmation des calculs scientifiques et des simulations numériques. Fortran a évolué au fil des années avec plusieurs versions, la plus récente étant Fortran 2018. Fortran est couramment utilisé dans le domaine scientifique et de l'ingénierie pour des applications nécessitant des calculs scientifiques et numériques intensifs. Il est utilisé dans des domaines tels que l'aéronautique, l'astrophysique, la météorologie, la modélisation des matériaux, la physique des particules, l'ingénierie structurelle, etc.

Avantages :

- Performances : Fortran est réputé pour ses performances élevées dans les calculs scientifiques. Il offre des fonctionnalités optimisées pour les opérations numériques et peut tirer parti des architectures matérielles avancées.
- Syntaxe simple : Fortran utilise une syntaxe relativement simple et claire, ce qui facilite la lecture et la compréhension du code. Il est souvent considéré comme plus facile à apprendre que certains autres langages de programmation.
- Support des tableaux multidimensionnels : Fortran prend en charge les tableaux multidimensionnels, ce qui en fait un choix populaire pour les calculs scientifiques nécessitant des opérations sur des matrices et des tableaux de données.

Inconvénients :

- Abstraction limitée : Comparé à certains langages de programmation modernes, Fortran offre une abstraction limitée et des fonctionnalités de programmation orientée objet moins avancées. Il est davantage axé sur les calculs numériques et les opérations mathématiques.
- Écosystème restreint : Le nombre de développeurs utilisant Fortran est relativement réduit par rapport à d'autres langages de programmation plus récents. Cela peut entraîner une disponibilité plus limitée de ressources, de bibliothèques et de supports communautaires.
- Syntaxe obsolète : Bien que Fortran ait évolué au fil des années, certaines parties du langage conservent une syntaxe obsolète et peuvent sembler moins modernes que d'autres langages de programmation récents.

VII.4. MATLAB

MATLAB est un environnement de programmation et un langage de programmation développés par MathWorks. Il est largement utilisé dans les domaines scientifiques et techniques pour le calcul numérique, l'analyse de données, la modélisation, la simulation et la visualisation

MATLAB, acronyme de "MATrix LABoratory", est un environnement de développement intégré (IDE) qui fournit un langage de programmation de haut niveau pour le calcul numérique et l'analyse des données. Il est basé sur des matrices et offre des fonctionnalités puissantes pour le traitement des signaux, l'optimisation, l'algèbre linéaire, la statistique, la modélisation, etc. MATLAB utilise une syntaxe simple et expressive qui facilite le développement rapide d'applications et de prototypes.

MATLAB est utilisé dans divers domaines, notamment les mathématiques, la physique, l'ingénierie, les sciences de la vie, la finance, l'économie, l'informatique, etc. Il est couramment utilisé pour effectuer des calculs numériques, résoudre des équations mathématiques, simuler des systèmes dynamiques, traiter des signaux et des images, effectuer des analyses statistiques et créer des visualisations graphiques.

Avantages :

- Facilité d'utilisation : MATLAB offre une syntaxe intuitive et un grand nombre de fonctions prédéfinies qui facilitent le développement rapide de code. Il est apprécié pour sa simplicité et sa convivialité, même pour les personnes sans formation en programmation.
- Vaste bibliothèque : MATLAB dispose d'une vaste bibliothèque de fonctions et de boîtes à outils spécialisées pour des domaines spécifiques, tels que le traitement du signal, l'optimisation, la statistique, la modélisation des systèmes dynamiques, etc. Cela permet d'accéder à des fonctionnalités avancées sans avoir à les développer à partir de zéro.
- Visualisation graphique : MATLAB propose des outils puissants pour créer des visualisations graphiques, des tracés et des animations. Cela facilite la compréhension et la communication des résultats et des données.

Inconvénients :

- Coût : MATLAB est un logiciel commercial et son coût peut être élevé, en particulier pour les utilisateurs individuels ou les petites entreprises. Cependant, il existe des options d'octroi de licences académiques et des alternatives open source comme Octave, qui est compatible avec MATLAB.
- Performance limitée : Dans certaines situations, MATLAB peut être moins performant que des langages de programmation compilés tels que C++ ou Fortran, en particulier pour les calculs intensifs ou les boucles de grande taille. Cependant, MATLAB propose des fonctionnalités d'optimisation et peut être utilisé en conjonction avec des langages de programmation plus rapides pour améliorer les performances.

VII.5. LE LANGAGE C

Le langage C est un langage de programmation de haut niveau créé à la fin des années 1960 par Dennis Ritchie chez Bell Labs. Il est largement utilisé dans le développement logiciel, en particulier pour les systèmes d'exploitation, les pilotes de périphériques, les applications embarquées et les logiciels de bas niveau.

Le langage C est un langage de programmation impératif et procédural. Il est connu pour sa simplicité, sa puissance et son efficacité. C offre un contrôle précis sur le matériel et permet aux développeurs d'accéder directement à la mémoire et aux ressources système.

Le langage C est utilisé dans divers domaines, notamment :

- Développement de systèmes d'exploitation : De nombreux systèmes d'exploitation, tels que Unix, Linux et Windows, sont largement écrits en C.
- Programmation système et embarquée : C est utilisé pour développer des pilotes de périphériques, des microcontrôleurs, des logiciels embarqués et d'autres systèmes de bas niveau.
- Développement d'applications : C peut être utilisé pour créer des applications de bureau, des applications scientifiques, des jeux, des logiciels de traitement d'images, etc.

Avantages :

- Efficacité et performance : Le langage C est reconnu pour sa rapidité d'exécution et son efficacité en termes d'utilisation des ressources système. Il permet un contrôle précis du matériel et une optimisation fine du code.
- Portabilité : Les programmes écrits en C peuvent être portables sur différentes plates-formes avec peu ou pas de modifications. Cela est dû à la disponibilité d'implémentations de compilateurs C pour de nombreuses architectures matérielles.
- Accès à la mémoire : C offre un accès direct à la mémoire et prend en charge les opérations de bas niveau, ce qui est essentiel pour certaines applications nécessitant une manipulation précise des données.

Inconvénients :

- Complexité : Le langage C peut être considéré comme plus complexe que certains langages de programmation de plus haut niveau. Il requiert une bonne compréhension des concepts de base de la programmation, ainsi que des connaissances sur la gestion manuelle de la mémoire.
- Vulnérabilités : En raison de l'accès direct à la mémoire, les programmes en C sont plus sujets à des erreurs de sécurité telles que les débordements de tampon et les problèmes de sécurité liés à la gestion de la mémoire.
- Développement plus lent : Étant donné que C nécessite un niveau plus bas de détails d'implémentation, le développement en C peut prendre plus de temps et nécessiter une plus grande quantité de code par rapport à certains langages de plus haut niveau.

VII.6. LE LANGAGE C++

Le langage C++ est une extension du langage de programmation C. Il a été développé dans les années 1980 par Bjarne Stroustrup et offre des fonctionnalités supplémentaires par rapport au langage C.

Le langage C++ est un langage de programmation orienté objet, ce qui signifie qu'il permet la programmation basée sur des objets et des classes. Il est compatible avec le langage C

et offre de nombreuses fonctionnalités supplémentaires, notamment l'encapsulation, l'héritage, le polymorphisme, les gabarits (templates), etc. C++ est souvent considéré comme une extension du langage C, car il conserve sa syntaxe et ses fonctionnalités, tout en ajoutant des fonctionnalités orientées objet.

C++ est utilisé dans de nombreux domaines, notamment :

- Développement de logiciels : C++ est utilisé pour le développement d'applications de bureau, de logiciels système, de jeux vidéo, de moteurs de rendu graphique, etc.
- Systèmes embarqués : C++ est largement utilisé dans le développement de logiciels pour les systèmes embarqués, tels que les dispositifs mobiles, les appareils électroniques, les systèmes de contrôle, etc.
- Haute performance : C++ est utilisé pour les applications nécessitant des performances élevées, telles que le calcul scientifique, les simulations, le traitement d'images, etc.

Avantages :

- Programmation orientée objet : C++ offre la possibilité de programmer de manière orientée objet, ce qui permet une meilleure organisation et structure du code, ainsi qu'une réutilisabilité accrue.
- Performances élevées : C++ permet un contrôle précis du matériel et une optimisation bas niveau, ce qui en fait un choix populaire pour les applications nécessitant des performances élevées.
- Compatibilité avec C : Étant une extension du langage C, C++ est compatible avec le code C existant, ce qui facilite l'intégration de code C dans les projets C++.

Inconvénients :

- Complexité accrue : Comparé au langage C, C++ peut être considéré comme plus complexe en raison de ses fonctionnalités orientées objet et de sa syntaxe plus étendue. Cela peut rendre l'apprentissage et la maîtrise de C++ plus difficiles pour les débutants.
- Gestion de la mémoire : Bien que C++ offre une gestion de la mémoire plus flexible que C grâce à des fonctionnalités telles que le ramasse-miettes (garbage collector) ou les pointeurs intelligents, il nécessite toujours une attention particulière pour éviter les fuites de mémoire et les problèmes de gestion de la mémoire.
- Temps de développement : Étant donné que C++ offre plus de fonctionnalités et de flexibilité, le développement en C++ peut prendre plus de temps par rapport à d'autres langages de programmation plus haut niveau.

Le langage C++ est largement utilisé pour ses performances élevées, sa flexibilité et sa compatibilité avec le code C. Cependant, il nécessite une bonne compréhension des concepts de programmation orientée objet et une attention particulière à la gestion de la mémoire

VII.7. LE LANGAGE RUBY

Le langage Ruby est un langage de programmation dynamique et interprété qui a été créé au Japon par Yukihiro Matsumoto dans les années 1990. Il est réputé pour sa simplicité syntaxique, sa lisibilité et sa flexibilité.

Ruby est un langage de programmation orienté objet qui met l'accent sur la simplicité et la productivité du développeur. Il offre une syntaxe élégante et expressive qui facilite la lecture et l'écriture du code. Ruby est interprété, ce qui signifie que le code source est exécuté directement sans nécessiter de compilation préalable.

Ruby est utilisé dans divers domaines, notamment :

- Développement web : Ruby on Rails, un framework web basé sur Ruby, est très populaire pour le développement d'applications web dynamiques et interactives.
- Scripting : Ruby est souvent utilisé comme langage de scripting pour automatiser des tâches, créer des scripts système et gérer des flux de travail.
- Développement d'applications : Ruby est utilisé pour développer une grande variété d'applications, des outils en ligne de commande aux applications de bureau.

Avantages :

- Simplicité syntaxique : Ruby a été conçu pour être lisible et facile à comprendre, ce qui permet aux développeurs d'écrire du code concis et expressif.
- Productivité élevée : Grâce à sa syntaxe concise et à sa grande bibliothèque standard, Ruby permet de développer rapidement des applications avec moins de code.
- Flexibilité : Ruby est un langage très flexible qui permet aux développeurs d'adopter différentes approches de programmation et de s'adapter aux besoins changeants d'un projet.

Inconvénients :

- Performance : Comparé à certains langages compilés tels que C++ ou Java, Ruby peut être plus lent en termes d'exécution. Cependant, cela dépend du contexte d'utilisation et des optimisations apportées au code.
- Écosystème plus restreint : Bien que Ruby dispose d'une communauté active et d'une bibliothèque standard étendue, son écosystème peut être moins vaste que celui d'autres langages plus populaires. Cela peut entraîner une disponibilité limitée de certaines bibliothèques spécialisées.
- Moins adapté aux applications intensives en calcul : En raison de ses performances relatives, Ruby peut être moins adapté aux applications nécessitant des calculs intensifs ou des traitements de données massifs.

Ruby est apprécié pour sa simplicité, sa lisibilité et sa productivité élevée. Il est particulièrement populaire dans le développement web grâce à Ruby on Rails, qui offre un cadre de développement web complet et puissant. Cependant, il est important de prendre en compte les

performances et les exigences spécifiques du projet avant de choisir Ruby comme langage de programmation.

VII.8. LE LANGAGE PASCAL

Le langage Pascal est un langage de programmation impératif et structuré qui a été créé dans les années 1970 par Niklaus Wirth. Il est conçu pour favoriser la lisibilité du code, la fiabilité et la facilité de maintenance.

Le langage Pascal est un langage de programmation qui met l'accent sur la clarté et la lisibilité du code. Il utilise une syntaxe structurée avec des blocs, des instructions conditionnelles et des boucles. Le langage Pascal est statiquement typé, ce qui signifie que les types des variables doivent être

Le langage Pascal a été initialement développé pour l'enseignement de la programmation et la conception de logiciels pédagogiques. Il a également été utilisé pour développer des applications dans divers domaines, notamment :

- Éducation : Pascal est souvent utilisé pour enseigner les concepts de programmation et les structures de contrôle aux débutants en programmation.
- Développement d'applications : Bien que Pascal ne soit pas aussi largement utilisé que d'autres langages de programmation, il a été utilisé pour développer des applications de bureau, des outils scientifiques, des systèmes embarqués et des logiciels éducatifs.

Avantages :

- Lisibilité et clarté du code : La syntaxe du Pascal est conçue pour être facilement lisible et compréhensible, ce qui facilite la maintenance et la collaboration sur les projets.
- Gestion des erreurs : Le langage Pascal met l'accent sur la détection précoce des erreurs et la gestion rigoureuse des exceptions, ce qui favorise la fiabilité du code.
- Structure et organisation : Pascal encourage la programmation structurée et fournit des mécanismes pour organiser le code en modules et en unités, ce qui facilite la gestion des projets de grande envergure.

Inconvénients :

- Écosystème limité : Comparé à d'autres langages plus populaires, l'écosystème et les bibliothèques disponibles pour Pascal peuvent être plus restreints, ce qui peut limiter les options de développement et l'adoption de nouvelles technologies.
- Moins adapté aux applications modernes : En raison de sa popularité relativement faible dans l'industrie, Pascal peut être moins adapté au développement d'applications modernes nécessitant des fonctionnalités avancées et une intégration avec d'autres technologies.
- Limitations de performances : En raison de sa conception orientée vers la lisibilité et la sécurité, le langage Pascal peut offrir des performances légèrement inférieures à celles de certains langages de programmation plus optimisés pour les calculs intensifs.

Le langage Pascal reste pertinent pour l'enseignement de la programmation et pour des projets spécifiques où la lisibilité du code et la fiabilité sont des priorités. Cependant, dans des contextes plus larges, d'autres langages de programmation peuvent offrir une plus grande polyvalence et des fonctionnalités plus avancées.

VII.9. LE LANGAGE ADA

Le langage Ada est un langage de programmation structuré et orienté objet qui a été développé dans les années 1970 et 1980 par le département de la Défense des États-Unis. Il a été conçu pour des applications critiques nécessitant une fiabilité, une sécurité et une maintenabilité élevées.

Le langage Ada est un langage de programmation à usage général qui met l'accent sur la fiabilité, la sécurité et la maintenabilité du code. Il prend en charge la programmation structurée, la programmation orientée objet et la programmation concurrente. Ada est un langage fortement typé, ce qui signifie que les types doivent être définis explicitement et respectés strictement.

Ada est couramment utilisé dans les applications critiques nécessitant une haute fiabilité et une sécurité accrue, notamment :

- Domaine militaire et aérospatial : Ada est largement utilisé pour le développement de systèmes embarqués, de logiciels de contrôle de vol, de simulateurs de vol, de systèmes de commande et de systèmes de défense.
- Secteur médical : Ada est utilisé dans le développement de logiciels médicaux, tels que les dispositifs médicaux, les systèmes d'imagerie et les logiciels de traitement médical.
- Transport et infrastructure : Ada est utilisé pour le développement de systèmes de contrôle du trafic aérien, de systèmes de gestion du trafic routier et de systèmes ferroviaires.

Avantages :

- Fiabilité et sécurité : Ada est conçu pour garantir la fiabilité et la sécurité du code, en utilisant des mécanismes de vérification statique et des contrôles rigoureux de type. Cela en fait un choix privilégié pour les applications critiques où les erreurs peuvent avoir des conséquences graves.
- Maintenabilité et évolutivité : Le langage Ada encourage une programmation modulaire et offre des fonctionnalités de gestion des dépendances et de réutilisabilité du code, ce qui facilite la maintenabilité et l'évolutivité des projets.
- Support pour la programmation concurrente : Ada offre des mécanismes intégrés pour la programmation concurrente, permettant la gestion des tâches parallèles et des communications entre celles-ci.

Inconvénients :

- Courbe d'apprentissage : Ada peut nécessiter un apprentissage plus poussé en raison de sa syntaxe et de ses concepts spécifiques. Les développeurs doivent comprendre les directives strictes du langage pour respecter les règles de typage et de structuration.
- Écosystème limité : Comparé à d'autres langages de programmation plus populaires, l'écosystème d'Ada peut être plus restreint en termes de bibliothèques, d'outils et de ressources disponibles.
- Moins adapté pour certains domaines : En raison de sa spécialisation dans les applications critiques, Ada peut être moins adapté pour les projets plus généraux ou pour les applications nécessitant des fonctionnalités spécifiques ou des intégrations avec d'autres technologies.

Le langage Ada est particulièrement adapté aux applications critiques nécessitant une fiabilité et une sécurité élevées, notamment dans le domaine

VII.10. LE LANGAGE DELPHI

Le langage Delphi est un langage de programmation basé sur le Pascal qui est principalement utilisé pour le développement d'applications logicielles sous l'environnement de développement intégré (IDE) Delphi. Il a été créé par Borland au début des années 1990 et est souvent utilisé pour la programmation d'applications Windows.

Le langage Delphi est un langage de programmation orienté objet qui est basé sur le Pascal. Il offre une syntaxe claire et lisible qui facilite le développement d'applications logicielles. Delphi est associé à l'IDE Delphi, qui fournit des outils de développement puissants, une interface de conception visuelle et une bibliothèque de composants pour créer rapidement des interfaces utilisateur graphiques (GUI).

Delphi est principalement utilisé pour le développement d'applications logicielles sous Windows. Il est couramment utilisé dans les domaines suivants :

- Développement d'applications de bureau : Delphi est utilisé pour créer des applications de bureau Windows avec une interface utilisateur graphique riche et des fonctionnalités avancées.
- Développement d'applications d'entreprise : Delphi est utilisé pour créer des applications d'entreprise, telles que des systèmes de gestion de bases de données, des applications de gestion de la chaîne d'approvisionnement, des logiciels de comptabilité, etc.
- Développement d'applications client-serveur : Delphi permet de créer des applications client-serveur en utilisant les protocoles et technologies de communication courants.

Avantages :

- Rapidité de développement : Delphi, avec son IDE puissant et sa bibliothèque de composants, permet un développement rapide d'applications avec une interface utilisateur graphique attrayante.
- Productivité élevée : Delphi offre une syntaxe claire et lisible, ce qui facilite la compréhension et la maintenance du code. Il fournit également des fonctionnalités avancées de refactoring et de débogage pour améliorer la productivité des développeurs.
- Bonne performance : Les applications développées avec Delphi ont généralement de bonnes performances, notamment en termes de vitesse d'exécution et d'utilisation efficace des ressources système.

Inconvénients :

- Plateforme limitée : Delphi est principalement destiné au développement d'applications pour Windows. Bien qu'il y ait des alternatives pour d'autres plateformes (comme FireMonkey pour la création d'applications multiplateformes), l'écosystème de Delphi est principalement orienté vers Windows.
- Écosystème plus restreint : Comparé à d'autres langages de programmation plus populaires, l'écosystème de Delphi peut être plus limité en termes de bibliothèques tierces et de ressources communautaires disponibles.
- Moins adapté pour les applications Web : Bien que Delphi offre des fonctionnalités pour le développement d'applications Web, il est moins utilisé dans ce domaine par rapport à d'autres langages et frameworks plus largement adoptés.

Le langage Delphi est apprécié pour son IDE puissant, sa rapidité de développement et sa facilité de création d'applications Windows avec une interface utilisateur graphique riche. Il est principalement utilisé pour les applications

VII.11. SolidWorks

SolidWorks est un logiciel de conception assistée par ordinateur (CAO) en 3D, largement utilisé dans le domaine de l'ingénierie et de la conception de produits. Il permet de créer, de simuler et de visualiser des modèles 3D de pièces et d'assemblages, ainsi que de générer des dessins techniques détaillés.

SolidWorks est un logiciel de CAO développé par Dassault Systèmes, utilisé pour la conception de produits et de pièces en 3D. Il offre une interface conviviale et des fonctionnalités avancées pour modéliser des pièces, créer des assemblages, effectuer des simulations et générer des dessins techniques.

SolidWorks est utilisé dans de nombreux domaines de l'ingénierie et de la conception, notamment :

- Conception mécanique : SolidWorks est couramment utilisé pour concevoir des pièces mécaniques, des systèmes de machines, des équipements industriels, des outils, etc.
- Conception de produits : Il est utilisé pour la conception de produits grand public, tels que les appareils électroniques, les dispositifs médicaux, les biens de consommation, les meubles, etc.
- Ingénierie de fabrication : SolidWorks est utilisé pour l'ingénierie de fabrication, y compris la conception de moules, de matrices et d'outillages.
- Architecture et conception de bâtiments : SolidWorks propose également des fonctionnalités pour la conception architecturale et la modélisation de bâtiments.

Avantages :

- Interface conviviale : SolidWorks offre une interface utilisateur intuitive et conviviale, facilitant la prise en main et l'utilisation du logiciel.
- Modélisation 3D puissante : SolidWorks propose des outils avancés pour créer des modèles 3D précis et complexes, permettant aux ingénieurs et aux concepteurs de visualiser et de valider leurs conceptions.
- Intégration des fonctionnalités de simulation : SolidWorks intègre des fonctionnalités de simulation, ce qui permet aux utilisateurs de tester et de vérifier les performances et le comportement de leurs modèles avant la fabrication.
- Génération de dessins techniques : Le logiciel offre des fonctionnalités avancées pour générer des dessins techniques détaillés avec des vues en coupe, des cotations, des annotations, etc.

Inconvénients:

- Coût élevé : SolidWorks est un logiciel commercial et son coût peut être élevé, en particulier pour les petites entreprises ou les utilisateurs individuels.
- Apprentissage initial : En raison de sa richesse fonctionnelle, l'apprentissage de SolidWorks peut demander du temps et des efforts, en particulier pour les débutants en CAO.
- Limitations des fonctionnalités avancées : Bien que SolidWorks propose de nombreuses fonctionnalités avancées, certaines tâches très spécifiques ou complexes peuvent nécessiter l'utilisation de modules complémentaires ou d'autres logiciels spécialisés.

VII.12. AUTOCAD

AutoCAD est un logiciel de conception assistée par ordinateur (CAO) utilisé pour la création de dessins techniques en 2D et en 3D. Il est développé par Autodesk et est largement utilisé dans les domaines de l'architecture, de l'ingénierie, de la construction et de la fabrication.

AutoCAD est un logiciel de CAO qui permet aux utilisateurs de créer, de modifier et de partager des dessins techniques précis. Il offre une interface graphique conviviale et une grande variété d'outils pour la création de dessins en 2D et en 3D. AutoCAD prend en charge de nombreux formats de fichiers et offre des fonctionnalités avancées telles que la

modélisation solide, la gestion des calques, la gestion des annotations et la création de rendus.

AutoCAD est utilisé dans divers domaines de l'architecture, de l'ingénierie, de la construction et de la fabrication, notamment :

- Architecture : AutoCAD est utilisé pour la conception de bâtiments, la création de plans d'étage, la modélisation de structures, la création de rendus architecturaux, etc.
- Ingénierie : Il est utilisé pour la conception de produits, la création de dessins techniques, la modélisation de pièces mécaniques, la simulation d'assemblages, etc.
- Construction : AutoCAD est utilisé pour la création de plans de construction, la coordination des projets, la gestion des données de construction, etc.
- Fabrication : Il est utilisé pour la conception de moules, la création de dessins pour l'usinage, la modélisation de produits, la documentation de fabrication, etc.

Avantages :

- Polyvalence : AutoCAD est utilisé dans de nombreux domaines et offre une large gamme d'outils et de fonctionnalités pour répondre aux besoins de diverses industries.
- Précision : AutoCAD permet de créer des dessins techniques précis avec des mesures et des dimensions précises, ce qui est essentiel dans les domaines de l'architecture et de l'ingénierie.
- Collaboration : AutoCAD facilite la collaboration entre les différents acteurs d'un projet en permettant le partage facile des dessins, la coordination des modifications et la gestion des versions.
- Personnalisation : AutoCAD offre des fonctionnalités de personnalisation avancées, permettant aux utilisateurs de créer des scripts, des macros et des menus personnalisés pour automatiser des tâches récurrentes et améliorer l'efficacité.

Inconvénients :

- Courbe d'apprentissage : AutoCAD est un logiciel complexe, ce qui peut entraîner une courbe d'apprentissage raide, en particulier pour les utilisateurs débutants.
- Coût : AutoCAD est un logiciel commercial, et son coût peut être élevé, en particulier pour les petites entreprises ou les utilisateurs individuels.
- Dépendance aux licences : L'utilisation d'AutoCAD nécessite l'acquisition de licences, ce qui peut limiter son accessibilité pour certains utilisateurs.

Dans l'ensemble, AutoCAD est un logiciel de CAO largement utilisé dans les industries de l'architecture, de l'ingénierie, de la construction et de la fabrication. Il offre des fonctionnalités avancées pour la création de dessins techniques précis et la modélisation en 2D et en 3D.

VII.13. CATIA

CATIA, développé par Dassault Systèmes, est un logiciel de conception assistée par ordinateur (CAO) utilisé dans plusieurs industries pour la conception de produits complexes en 3D.

CATIA (Computer-Aided Three-Dimensional Interactive Application) est un logiciel de CAO haut de gamme qui offre des fonctionnalités complètes pour la conception, la modélisation, la simulation et la documentation de produits en 3D. Il est utilisé dans des secteurs tels que l'automobile, l'aérospatiale, la construction navale, l'industrie manufacturière et d'autres domaines nécessitant des conceptions complexes.

CATIA est utilisé dans de nombreux domaines de l'ingénierie et de la conception, notamment :

- Automobile : CATIA est largement utilisé dans l'industrie automobile pour la conception de véhicules, la modélisation de pièces, la simulation des performances, la gestion des assemblages, etc.
- Aérospatiale : Il est utilisé pour la conception d'aéronefs, la modélisation de structures, la simulation des charges aérodynamiques, la gestion des systèmes, etc.
- Construction navale : CATIA est utilisé pour la conception de navires, la modélisation de coques, la simulation des mouvements maritimes, la gestion des systèmes à bord, etc.
- Industrie manufacturière : Il est utilisé pour la conception de machines, la modélisation de pièces mécaniques, la simulation des processus de fabrication, la gestion des assemblages, etc.

Avantages :

- Fonctionnalités complètes : CATIA offre une gamme complète d'outils de conception, de modélisation, de simulation et de documentation, permettant aux utilisateurs de réaliser des conceptions complexes et de gérer tout le processus de développement du produit.
- Collaboration et gestion des données : CATIA facilite la collaboration entre les équipes grâce à des fonctionnalités avancées de partage de données, de gestion des modifications et de communication.
- Intégration du cycle de vie des produits (PLM) : CATIA intègre des fonctionnalités de gestion du cycle de vie des produits, permettant de gérer efficacement toutes les étapes du développement de produits, de la conception à la fabrication et à la maintenance.
- Adaptabilité : CATIA est conçu pour s'adapter aux besoins spécifiques de chaque industrie, offrant des fonctionnalités spécialisées et des modules complémentaires pour répondre aux exigences particulières.

Inconvénients :

- Complexité : CATIA est un logiciel complexe avec une courbe d'apprentissage raide, ce qui peut nécessiter une formation approfondie pour maîtriser toutes ses fonctionnalités.

- Coût : CATIA est un logiciel commercial, et son coût peut être élevé, en particulier pour les petites entreprises ou les utilisateurs individuels.
- Exigences matérielles élevées : En raison de sa complexité, CATIA nécessite souvent des ordinateurs puissants pour une utilisation fluide et efficace.

VII.14. ADVANCED AIRCRAFT ANALYSIS (AAA)

Advanced Aircraft Analysis (AAA) est un outil logiciel développé par DARcorporation pour la conception et l'analyse des avions. Il est largement utilisé dans l'industrie aérospatiale par les ingénieurs, les chercheurs et les concepteurs impliqués dans le développement de nouveaux avions ou la modification d'avions existants.

Advanced Aircraft Analysis (AAA) est un logiciel complet qui offre différentes fonctionnalités d'analyse pour la conception d'avions, l'évaluation des performances, l'analyse de la stabilité et du contrôle, ainsi que le dimensionnement préliminaire. Il intègre des analyses aérodynamiques, de structures, de propulsion et de systèmes pour soutenir le processus de conception et évaluer les caractéristiques de performance de l'avion.

AAA est principalement utilisé dans l'industrie aérospatiale, notamment pour :

- La conception d'avions : Il aide dans les phases conceptuelles et préliminaires de conception, en aidant les ingénieurs à optimiser la configuration de l'avion et à déterminer ses capacités de performance.
- L'évaluation des performances : AAA permet l'évaluation des paramètres de performance des avions tels que la portée, l'endurance, les performances en montée, les performances au décollage et à l'atterrissage.
- L'analyse de stabilité et de contrôle : AAA permet l'analyse de la stabilité longitudinale, latérale et directionnelle de l'avion, ainsi que l'évaluation du contrôle de l'avion dans différentes conditions de vol.

Avantages :

- Fonctionnalités complètes : AAA offre une gamme complète de fonctionnalités d'analyse pour soutenir la conception et l'évaluation des performances des avions.
- Précision : AAA utilise des modèles et des méthodes d'analyse avancés pour fournir des résultats précis et fiables.
- Gain de temps : AAA permet d'accélérer le processus de conception en automatisant certaines tâches d'analyse et en fournissant des outils efficaces pour l'évaluation des performances.
- Visualisation des résultats : AAA offre des fonctionnalités de visualisation graphique pour aider les utilisateurs à interpréter les résultats des analyses.

Inconvénients :

- Courbe d'apprentissage : AAA est un logiciel complexe, ce qui peut nécessiter une courbe d'apprentissage pour les nouveaux utilisateurs.

- Coût : AAA est un logiciel commercial et son coût peut être élevé, en particulier pour les petites entreprises ou les utilisateurs individuels.
- Dépendance aux données : AAA dépend de la disponibilité et de la qualité des données aérodynamiques, structurales et de propulsion pour fournir des résultats précis.

VII.15. ANSYS

ANSYS est un logiciel de simulation par éléments finis (FEA) utilisé pour l'analyse et la modélisation de divers phénomènes physiques. Il est développé par ANSYS, Inc. et est largement utilisé dans de nombreux domaines de l'ingénierie.

ANSYS est une suite logicielle complète qui propose des outils pour la simulation numérique, l'analyse de structures, la dynamique des fluides, l'électromagnétisme, l'optimisation de conception, la simulation des systèmes embarqués, etc. Il utilise la méthode des éléments finis pour résoudre des équations mathématiques complexes et modéliser le comportement physique des systèmes.

ANSYS est utilisé dans divers domaines de l'ingénierie, notamment :

- Mécanique : Pour l'analyse structurelle, la dynamique, la fatigue, la thermique, etc.
- Fluides : Pour la dynamique des fluides, l'écoulement des fluides, la combustion, etc.
- Électromagnétisme : Pour la simulation des champs électromagnétiques, les antennes, les circuits électroniques, etc.
- Systèmes embarqués : Pour la simulation des circuits électriques, la compatibilité électromagnétique, la conception de systèmes électroniques, etc.

Avantages :

- Large gamme de fonctionnalités : ANSYS offre une grande variété d'outils pour différents domaines de l'ingénierie, ce qui permet de résoudre des problèmes complexes et d'effectuer des analyses détaillées.
- Précision : ANSYS utilise des techniques de modélisation avancées et des méthodes de résolution numérique pour obtenir des résultats précis et fiables.
- Adaptabilité : ANSYS peut être utilisé dans divers secteurs de l'ingénierie, ce qui le rend polyvalent et adaptable à différents besoins et applications.
- Intégration : ANSYS peut être intégré à d'autres logiciels et outils de conception, permettant une collaboration efficace et un flux de travail simplifié.

Inconvénients :

- Courbe d'apprentissage : L'utilisation d'ANSYS nécessite une certaine expertise et une formation pour tirer pleinement parti de ses fonctionnalités avancées.

- Coût : ANSYS est un logiciel commercial et son coût peut être élevé, en particulier pour les petites entreprises ou les utilisateurs individuels.
- Complexité : En raison de sa puissance et de sa diversité de fonctionnalités, ANSYS peut être complexe à utiliser pour les débutants et nécessite une bonne compréhension des principes de base de la simulation numérique.

VIII. CHAPITRE III

TUTORIAL SUCCINT DE PYTHON

VIII.1. Introduction

Ce chapitre est dédié à une introduction succincte au langage de programmation Python. Que vous soyez novice en programmation ou que vous souhaitiez simplement vous familiariser avec Python, ce tutoriel vous fournira les bases essentielles pour commencer à écrire du code Python de manière efficace. Nous aborderons les concepts fondamentaux du langage, la syntaxe de base et les structures de contrôle, ainsi que certaines fonctionnalités avancées. Que vous ayez des projets en tête ou que vous souhaitiez simplement acquérir une nouvelle compétence, ce tutorial vous aidera à démarrer rapidement avec Python

Python est un langage de programmation informatique incroyablement polyvalent et puissant. Il a été créé en 1989 par Guido van Rossum, aux Pays-Bas, et depuis lors, il est devenu l'un des langages de programmation les plus populaires au monde.

Ce qui rend Python si attrayant, c'est sa simplicité et sa lisibilité. Sa syntaxe est conçue pour être facile à lire et à comprendre, ce qui en fait un excellent choix pour les débutants en programmation. Même si vous n'avez aucune expérience préalable en programmation, Python vous permettra de vous lancer rapidement et de commencer à écrire du code fonctionnel.

Python est utilisé dans une grande variété de domaines. Pour commencer, il est largement utilisé dans le développement web. De nombreux frameworks populaires, tels que Django et Flask, sont basés sur Python, ce qui facilite la création d'applications web robustes et évolutives.

De plus, Python est également très populaire dans le domaine de l'apprentissage automatique (machine learning) et de l'apprentissage profond (deep learning). Sa bibliothèque principale, appelée "NumPy", fournit des fonctionnalités avancées pour la manipulation de données et les calculs scientifiques. Des bibliothèques supplémentaires telles que "TensorFlow" et "PyTorch" permettent aux développeurs de créer des modèles d'apprentissage automatique complexes et de résoudre des problèmes de classification, de régression et de reconnaissance d'images, entre autres.

En outre, Python est utilisé dans des logiciels d'imagerie 2D et d'animation 3D de pointe tels que Inkscape, Blender et Autodesk. Il offre une flexibilité et une facilité d'utilisation qui en font un choix privilégié pour de nombreux artistes et créateurs.

Grosso-modo, Python est un langage de programmation polyvalent, adapté aux débutants et aux professionnels, utilisé dans de nombreux domaines, tels que le développement web,

l'apprentissage automatique, l'apprentissage profond et les applications scientifiques. Si vous cherchez à acquérir une compétence précieuse et à élargir vos horizons en matière de programmation, Python est un excellent choix à considérer.

VIII.2. Des principales caractéristiques de Python :

Les principales caractéristiques de Python :

- **Codage lisible** :

La philosophie de conception de Python met fortement l'accent sur la lisibilité du code. La syntaxe simple et claire de Python permet aux développeurs d'écrire du code facile à comprendre, ce qui facilite la maintenance et la collaboration entre les membres d'une équipe de développement. Le principe du "zen de Python" encourage l'écriture de code élégant et lisible, ce qui rend le langage très apprécié par les développeurs.

- **Modules de support étendus** :

Python dispose d'une vaste bibliothèque standard qui comprend de nombreux modules prêts à l'emploi pour accomplir différentes tâches. Ces modules couvrent un large éventail de domaines, tels que le traitement des chaînes de caractères, la manipulation de fichiers, les opérations réseau, l'analyse de données, la génération de graphiques, etc. Cette richesse de modules facilite le développement rapide d'applications en utilisant des fonctionnalités préconstruites, ce qui permet aux développeurs de gagner du temps et d'éviter de réinventer la roue.

- **Développement communautaire :**

Python bénéficie d'une communauté de développeurs active et engagée. Cette communauté contribue à l'évolution continue du langage en proposant de nouvelles fonctionnalités, en identifiant et en résolvant des problèmes, et en partageant des ressources et des connaissances. Les forums de discussion, les listes de diffusion, les sites web et les événements dédiés à Python permettent aux développeurs d'interagir et de collaborer, favorisant ainsi un environnement d'apprentissage et de croissance.

- **Intégration facile de services web :**

Python facilite l'intégration de services web grâce à des bibliothèques et des frameworks robustes tels que Flask et Django. Ces outils permettent aux développeurs de créer rapidement des applications web et d'interagir avec des API, des bases de données et d'autres services web. Python offre également des bibliothèques pour les protocoles de communication tels que HTTP,

FTP, JSON, etc., ce qui facilite la création d'applications interconnectées et d'interfaces utilisateur dynamiques.

- **Structure de données facile à utiliser** :

Python fournit une variété de structures de données intégrées, telles que les listes, les tuples, les dictionnaires et les ensembles. Ces structures de données sont faciles à utiliser et offrent des fonctionnalités avancées pour la manipulation et la gestion des données. Par exemple, les listes permettent d'ajouter, de supprimer et de modifier des éléments facilement, tandis que les dictionnaires permettent de stocker des données sous forme de paires clé-valeur, offrant un accès rapide aux valeurs.

- **Typé dynamiquement :**

Python est un langage typé dynamiquement, ce qui signifie que les types des variables sont déterminés automatiquement lors de l'exécution du programme. Cela offre une flexibilité aux développeurs, car ils n'ont pas besoin de déclarer explicitement le type des variables. Cependant, Python prend également en charge l'annotation de types, ce qui permet aux développeurs de spécifier des types pour les variables, les arguments de fonction, etc., améliorant ainsi la lisibilité et la maintenance du code.

- **Orienté objet :**

Python est un langage de programmation orienté objet, ce qui signifie qu'il prend en charge les concepts de la programmation orientée objet tels que les classes, les objets, l'héritage, etc. La programmation orientée objet permet de structurer le code de manière modulaire et de réutiliser du code existant, favorisant ainsi la gestion de projets complexes et la facilité de maintenance.

- **Indépendant de la plateforme :**

Python est un langage indépendant de la plateforme, ce qui signifie que les programmes Python peuvent être exécutés sur différentes plateformes, telles que Windows, macOS, Linux, etc., sans nécessiter de modifications majeures. Cela rend Python très portable et facilite le déploiement d'applications sur différentes machines et systèmes d'exploitation.

- **Applications de bureau basées sur une interface graphique :**

Python offre plusieurs bibliothèques et frameworks, tels que Tkinter, PyQt et wxPython, qui permettent de créer des applications de bureau avec une interface graphique utilisateur (GUI).

Ces outils offrent des fonctionnalités pour créer des fenêtres, des boutons, des boîtes de dialogue et d'autres éléments d'interface utilisateur, facilitant ainsi le développement d'applications conviviales et interactives.

En combinant toutes ces caractéristiques, Python devient un choix populaire pour de nombreux développeurs, qu'ils soient débutants ou expérimentés, dans divers domaines de programmation.

VIII.3. Pour quoi python

- **Plate-forme croisée**

Python est devenu un langage de programmation unique pour les développeurs du monde entier. Sa polyvalence intégrée le rend idéal pour les solutions logicielles telles que les applications mobiles, les applications de bureau, le développement web et la programmation hardware. Quel que soit l'environnement numérique ou la plateforme, Python peut fournir des solutions logicielles rapides et robustes sans sacrifier les performances.

- **Multitâches**

Python est particulièrement bien adapté pour offrir un support multitâche aux entreprises de toutes tailles. Étant l'un des langages les plus fiables qui soit, Python peut rester concentré sur le maintien de la sécurité des données critiques tout en fournissant simultanément des applications web de haute performance.

- **Développement rapide**

En permettant le prototypage et les itérations rapides, les développeurs web choisissent Python parce qu'il combine une productivité élevée avec une facilité d'utilisation et devient un langage idéal pour les startups et petites PME. Ceci est particulièrement important lorsque la mise sur le marché d'un produit web offre un avantage concurrentiel.

- **Évolutivité**

Une autre préoccupation majeure de Python est qu'il permet aux développeurs de faire évoluer facilement et rapidement leurs projets web. Les bibliothèques de code prédéfinies de Python sont adaptables, offrant aux développeurs une flexibilité incroyable lorsque les projets web évoluent dans des directions différentes.

- **Intégration du langage**

Il est très courant pour les développeurs web de devoir s'appuyer sur des applications web déjà existantes. Python est facile d'intégration à toute une série d'autres langages de programmation largement utilisés, tels que Ruby, C (en utilisant CPython), Java et PHP. Cela offre une flexibilité de premier ordre pour les projets web de complexité variable.

- **Récupération de données sur le web**

Ce langage de programmation est parfait pour le « web scraping », la collecte de données à partir de sites web et la manipulation facilitée de grands ensembles de données. Une fois les données collectées, les développeurs web peuvent utiliser les bibliothèques Python pour passer au crible la plupart des données au lieu de devoir écrire du code manuellement pour leur traitement.

- **Bibliothèques et Framework**

Les développeurs Python peuvent profiter de la disponibilité gratuite de centaines de bibliothèques et de Framework, conçus pour répondre à toutes les exigences de programmation et, en fin de compte, pour faire gagner du temps et des ressources aux entreprises de toutes tailles. Parmi les bibliothèques Python les plus populaires auprès des développeurs de logiciels figurent :

- **NumPy** : NumPy est une bibliothèque fondamentale pour le calcul scientifique en Python. Elle fournit des structures de données avancées, telles que les tableaux multidimensionnels, ainsi que des fonctions mathématiques et des opérations de manipulation de données efficaces.
- SciPy : SciPy est une bibliothèque qui s'appuie sur NumPy et offre des fonctionnalités supplémentaires pour le calcul scientifique. Elle inclut des modules pour l'optimisation, l'algèbre linéaire, le traitement du signal, l'analyse statistique, la simulation, et bien plus encore.
- **Django** : Django est un framework web puissant et complet, utilisé pour le développement rapide d'applications web robustes et sécurisées. Il intègre un ORM (Object-Relational Mapping) pour faciliter l'interaction avec les bases de données, et fournit des fonctionnalités telles que l'authentification utilisateur, la gestion des formulaires, la génération de vues, etc.
- **Theano** : Theano est une bibliothèque spécialisée dans le calcul numérique et le machine learning. Elle permet d'exprimer et d'optimiser des calculs mathématiques impliquant des

tableaux multidimensionnels, ce qui la rend très utilisée pour la création de modèles d'apprentissage automatique.

- **Pandas** : Pandas est une bibliothèque populaire pour la manipulation et l'analyse des données. Elle fournit des structures de données performantes, telles que les DataFrames, qui permettent de manipuler et d'analyser facilement des ensembles de données tabulaires.
- **Keras** : Keras est une bibliothèque haut niveau pour le développement d'applications d'apprentissage automatique et d'apprentissage profond. Elle fournit une interface conviviale et abstraite pour la construction de modèles de réseaux de neurones, simplifiant ainsi le processus de création et d'entraînement des modèles.
- **PyTorch** : PyTorch est un framework d'apprentissage automatique open source largement utilisé. Il offre une grande flexibilité et une facilité d'utilisation pour la création et l'entraînement de modèles d'apprentissage profond. PyTorch est également apprécié pour sa capacité à exécuter des calculs sur des GPU.
- **TensorFlow** : TensorFlow est un autre framework d'apprentissage automatique et d'apprentissage profond très populaire. Il permet de construire et de déployer des modèles d'apprentissage automatique à grande échelle, en utilisant des graphiques de calcul et des calculs distribués.

➢ **Science des données**

Python est largement utilisé dans les applications de science des données (autrement appelée Data Science) au sein des communautés scientifiques et de recherche. Sa facilité d'utilisation et sa syntaxe simple font de ce langage un choix idéal et facile pour les personnes qui n'ont pas bénéficié de formation d'ingénieur.

➢ **Tâches automatisées**

Vous souhaitez automatiser plusieurs tâches industrielles nécessitant de nombreux outils et modules complexes ? La solution est simple : Python. Pour automatiser des applications web à des fins de test ou pour effectuer des processus de scraping web, les développeurs peuvent utiliser la bibliothèque Sélénium par exemple.

➢ **Apprentissage automatique**

Du développement informatique au déploiement et à la maintenance des applications, Python apporte un accès à d'excellentes bibliothèques et outils, ce qui fait de lui un choix de

premier ordre pour les développeurs travaillant sur des projets d'apprentissage automatique et d'IA (intelligence artificielle). En tant que langage polyvalent, Python effectue des tâches complexes d'apprentissage automatique et crée des prototypes afin de tester votre produit.

- **Parrainage d'entreprises**

Tout langage de programmation informatique soutenu par des entreprises renommées aura tendance à prospérer et à se développer plus efficacement que les autres langages. Comme Python est largement adopté et soutenu par des géants de la technologie tels que Google, Facebook et Amazon, la liste des outils de soutien et des bibliothèques utiles disponibles pour les développeurs ne fait que s'allonger.

- **Soutien communautaire**

Nous ne nous contentons pas d'explorer l'un des langages de programmation les plus populaires au monde, mais aussi l'un des plus anciens et des mieux établis du secteur. La courbe d'apprentissage de Python est moins rapide que celle d'autres langages, tout simplement parce que sa communauté de développeurs offre une abondance de ressources d'apprentissage et de possibilités d'orientation, en passant par des séminaires et tutoriels aux livres et forums de développeurs.

- **Facilité d'apprentissage**

Les nouveaux venus dans le secteur du développement de logiciels apprécieront certainement la facilité et la rentabilité des débuts de la programmation en Python. Avec une syntaxe simplifiée et un accent mis sur le langage naturel, il s'agit de l'un des langages de programmation les plus accessibles actuellement disponibles. Sa force réside essentiellement dans l'écriture de code et son exécution à des vitesses beaucoup plus rapides que les autres langages.

- **Python Academia**

En combinant tous les avantages majeurs de Python ci-dessus, il est clair que Python est considéré comme un langage de programmation de premier ordre. Il est principalement enseigné dans les écoles et les universités du monde entier pour cette raison. Avec son large éventail d'applications et sa facilité d'utilisation, le langage Python offre un excellent premier portail vers

le monde étonnant de la programmation informatique, quelle que soit l'orientation professionnelle de l'étudiant.

VIII.4. Variables

VIII.4.1. Définition

Une variable est une zone de la mémoire de l'ordinateur dans laquelle une valeur est stockée. En Python, la déclaration d'une variable et son initialisation (c'est-à-dire la première valeur que l'on va stocker dedans) se font en même temps.

Ligne 1. Dans cet exemple, nous avons déclaré, puis initialisé la variable x avec la valeur 2. Notez bien qu'en réalité, il s'est passé plusieurs choses :

- Python a « deviné » que la variable était un entier. On dit que Python est un langage au typage dynamique.
- Python a alloué (réservé) l'espace en mémoire pour y accueillir un entier. Chaque type de variable prend plus ou moins d'espace en mémoire. Python a aussi fait en sorte qu'on puisse retrouver la variable sous le nom x.
- Enfin, Python a assigné la valeur 2 à la variable x.

```
1  >>> x = 2
2  >>> x
3  2
```

Figure (III.1) exemple de variable

VIII.4.2. Les types de variables

Le type d'une variable correspond à la nature de celle-ci. Les trois principaux types dont nous aurons besoin dans un premier temps sont les entiers, les nombres décimaux que nous appellerons *floats* et les chaînes de caractères .Bien sûr, il existe de nombreux autres types (par exemple, les booléens, les nombres complexes, etc.). Dans l'exemple précédent, nous avons

stocké un nombre entier dans la variable x, mais il est tout à fait possible de stocker des *floats*, des chaînes de caractères ou de nombreux autres types de variable que nous verrons par la suite

```
>>> y = 3.14
>>> y
3.14
>>> a = "bonjour"
>>> a
'bonjour'
>>> b = 'salut'
>>> b
'salut'
>>> c = """girafe"""
>>> c
'girafe'
>>> d = '''lion'''
>>> d
'lion'
```

Figure (III.2) exemple de type de variable

VIII.4.3. Nommage

Le nom des variables en Python peut être constitué de lettres minuscules, de lettres majuscules, de nombres ou du caractère souligné (_). Vous ne pouvez pas utiliser d'espace dans un nom de variable.

Par ailleurs, un nom de variable ne doit pas débuter par un chiffre et il n'est pas recommandé de le faire débuter par le caractère (_). De plus, il faut absolument éviter d'utiliser un mot « réservé » par Python comme nom de variable (par exemple : print, range, for, from, etc.).

Enfin, Python est sensible à la casse, ce qui signifie que les variables Test, test ou TEST sont différentes.

VIII.4.4. Opérations

Les quatre opérations arithmétiques de base se font de manière simple sur les types numériques L'opérateur / effectue une division. Contrairement aux opérateurs +, - et *,

```
>>> x = 45
>>> x + 2
47
>>> x - 2
43
>>> x * 3
135
>>> y = 2.5
>>> x - y
42.5
>>> (x * 10) + y
452.5
```

Figure (III.3) exemple d'opération numérique

L'opérateur puissance utilise les symboles **

```
>>> 2**3
8
>>> 2**4
16
```

Figure (III.4) exemple d'opération de la puissance

VIII.4.5. La fonction type ()

Si vous ne vous souvenez plus du type d'une variable, utilisez la fonction type () qui vous le rappellera.

```
>>> x = 2
>>> type(x)
<class 'int'>
>>> y = 2.0
>>> type(y)
<class 'float'>
>>> z = '2'
>>> type(z)
<class 'str'>
```

Figure (III.5) exemple de La fonction type ()

VIII.4.6. Conversion de types

En programmation, on est souvent amené à convertir les types, c'est-à-dire passer d'un type numérique à une chaîne de caractères ou vice-versa. En Python, rien de plus simple avec les fonctions Int (), float () et str (). Pour vous en convaincre,

```
>>> i = 3
>>> str(i)
'3'
>>> i = '456'
>>> int(i)
456
>>> float(i)
456.0
>>> i = '3.1416'
>>> float(i)
3.1416
```

Figure (III.6) exemple de conversion de types

VIII.4.7. Minimum et maximum

Python propose les fonctions min () et max () qui renvoient respectivement le minimum et le maximum de plusieurs entiers et / ou *floats* :

```
>>> min(1, -2, 4)
-2
>>> pi = 3.14
>>> e = 2.71
>>> max(e, pi)
3.14
>>> max(1, 2.4, -6)
2.4
```

Figure (III.7) exemple de minimum et maximum

VIII.5. Affichage

VIII.5.1. La fonction `print ()`

La fonction print () affiche l'argument qu'on lui passe entre parenthèses **et** un retour à ligne. Ce retour à ligne supplémentaire est ajouté par défaut. Si toutefois, on ne veut pas afficher ce retour à la ligne, on peut utiliser l'argument par « mot-clé » end :

```
>>> print("Hello world!")
Hello world!
>>> print("Hello world!", end="")
Hello world!>>>
```

Figure (III.8) exemple de La fonction `print()`

VIII.6. BOUCLES ET COMPARAISONS

VIII.6.1. Boucles `for`

VIII.6.1.1. Principe

En programmation, on est souvent amené à répéter plusieurs fois une instruction. Incontournables à tout langage de programmation, les boucles vont nous aider à réaliser cette tâche de manière compacte et efficace.

```
>>> animaux = ["girafe", "tigre", "singe", "souris"]
>>> for animal in animaux:
...     print(animal)
...
girafe
tigre
singe
souris
```

Figure (III.9) exemple de Boucles **`for`**

La variable `animal` est appelée variable d'itération, elle prend successivement les différentes valeurs de la liste `animaux` à chaque itération de la boucle. Celle-ci est créée par Python la première fois que la ligne contenant le **`for`** est exécutée. Une fois la boucle terminée, cette variable d'itération **`animal`** ne sera pas détruite et contiendra ainsi la dernière valeur de la liste `animaux`.

Notez bien les types des variables utilisées ici : `animaux` est une liste sur laquelle on itère, et `animal` est une chaîne de caractères car chaque élément de la liste est une chaîne de caractères. Nous verrons plus loin que la variable d'itération peut être de n'importe quel type selon la liste parcourue. En Python, une boucle itère toujours sur un objet dit **séquentiel** tel qu'une liste. Nous verrons aussi plus tard d'autres objets séquentiels sur lesquels on peut itérer dans une boucle.

D'ores et déjà, prêtez attention au caractère **deux-points** « : » à la fin de la ligne débutant par `for`. Cela signifie que la boucle `for` attend un **bloc d'instructions**, en l'occurrence toutes les instructions que Python répétera à chaque itération de la boucle. On appelle ce bloc d'instructions le **corps de la boucle**. Comment indique-t-on à Python où ce bloc commence et se termine ? Cela est signalé uniquement par l'**indentation**, c'est-à-dire le décalage vers la droite de la (ou des) ligne(s) du bloc d'instructions.

VIII.6.1.2. Fonction `range ()`

Python possède la fonction `range ()` sur les *Listes* et qui est aussi bien commode pour faire une boucle sur une liste d'entiers de manière automatique :

```
>>> for i in range(4):
...     print(i)
...
0
1
2
3
```

Figure (III.10) exemple de Fonction range ()

VIII.6.1.3. Nommage de la variable d'itération

Dans l'exemple précédent, nous avons choisi le nom `i` pour la variable d'itération. Ceci est une habitude en informatique et indique en général qu'il s'agit d'un entier (le nom `i` vient sans doute du mot indice ou *index* en anglais). Nous vous conseillons de suivre cette convention afin d'éviter les confusions, si vous itérez sur les indices vous pouvez appeler la variable d'itération `i` (par exemple dans `for i in range(4) :`).

Si, par contre, vous itérez sur une liste comportant des chaînes de caractères, mettez un nom explicite pour la variable d'itération. Par exemple :

```
For prénom in ["Joe", "Bill", and «John"]:
```

VIII.6.1.4. Itération sur les indices ou les éléments

Revenons à notre liste animaux. Nous allons maintenant parcourir cette liste :

```
>>> animaux = ["girafe", "tigre", "singe", "souris"]
>>> for i in range(4):
...     print(animaux[i])
...
girafe
tigre
singe
souris
```

Figure (III.11) exemple d'itération sur les indices ou les éléments

La variable i prendra les valeurs successives 0, 1, 2 et 3 et on accèdera à chaque élément de la liste animaux par son indice (*i.e.* animaux[i]). Notez à nouveau le nom i de la variable d'itération car on itère sur les indices.

Quand utiliser l'une ou l'autre des 2 méthodes ? La plus efficace est celle qui réalise **les itérations directement sur les éléments** :

```
>>> animaux = ["girafe", "tigre", "singe", "souris"]
>>> for animal in animaux:
...     print(animal)
...
girafe
tigre
singe
souris
```

Figure (III.12) exemple d'itérations directe sur les éléments

Toutefois, il se peut qu'au cours d'une boucle vous ayez besoin des indices, auquel cas vous devrez itérer sur les indices :

```
>>> animaux = ["girafe", "tigre", "singe", "souris"]
>>> for i in range(len(animaux)):
...     print(f"L'animal {i} est un(e) {animaux[i]}")
...
L'animal 0 est un(e) girafe
L'animal 1 est un(e) tigre
L'animal 2 est un(e) singe
L'animal 3 est un(e) souris
```

Figure (III.13) exemple d'itérations avec des indices

Python possède toutefois la fonction enumerate () qui vous permet d'itérer sur les indices et les éléments eux-mêmes.

```
>>> animaux = ["girafe", "tigre", "singe", "souris"]
>>> for i, animal in enumerate(animaux):
...     print(f"L'animal {i} est un(e) {animal}")
...
L'animal 0 est un(e) girafe
L'animal 1 est un(e) tigre
L'animal 2 est un(e) singe
L'animal 3 est un(e) souris
```

Figure (III.14) exemple de la fonction enumerate ()

VIII.6.2. Comparaisons :

Python est capable d'effectuer toute une série de comparaisons entre le contenu de deux variables, telles que :

Syntaxe Python	Signification
==	égal à
!=	différent de
>	supérieur à
>=	supérieur ou égal à
<	inférieur à
<=	inférieur ou égal à

VIII.6.3. Boucles `while`

Une autre alternative à l'instruction for couramment utilisée en informatique est la boucle while. Le principe est simple. Une série d'instructions est exécutée tant qu'une condition est vraie. Par exemple:

```
1  >>> i = 1
2  >>> while i <= 4:
3  ...     print(i)
4  ...     i = i + 1
5  ...
6  1
7  2
8  3
9  4
```

Figure (III.15) exemple de la boucles while

Remarquez qu'il est encore une fois nécessaire d'indenter le bloc d'instructions correspondant au corps de la boucle (ici, les instructions lignes 3 et 4).

Une boucle while nécessite généralement trois éléments pour fonctionner correctement :

- Initialisation de la variable d'itération avant la boucle (ligne 1).
- Test de la variable d'itération associée à l'instruction while (ligne 2).
- Mise à jour de la variable d'itération dans le corps de la boucle (ligne 4).

Vous pouvez néanmoins toujours stopper l'exécution d'un script Python à l'aide de la combinaison de touches *Ctrl-C* . Par exemple :

```
1  i = 0
2  while i < 10:
3      print("Le python c'est cool !")
```

Figure (III.16) exemple de stopper l'exécution

Ici, nous avons omis de mettre à jour la variable `i` dans le corps de la boucle. Par conséquent, la boucle ne s'arrêtera jamais (sauf en pressant *Ctrl-C*) puisque la condition `i < 10` sera toujours vraie.

La boucle `while` combinée à la fonction `input()` peut s'avérer commode lorsqu'on souhaite demander à l'utilisateur une valeur numérique. Par exemple :

```
>>> i = 0
>>> while i < 10:
...     reponse = input("Entrez un entier supérieur à 10 : ")
...     i = int(reponse)
...
Entrez un entier supérieur à 10 : 4
Entrez un entier supérieur à 10 : -3
Entrez un entier supérieur à 10 : 15
>>> i
15
```

Figure (III.17) exemple de la fonction input ()

La fonction input () prend en argument un message (sous la forme d'une chaîne de caractères), demande à l'utilisateur d'entrer une valeur et renvoie celle-ci sous forme d'une chaîne de caractères. Il faut ensuite convertir cette dernière en entier (avec la fonction Int()).

VIII.7. Tests

VIII.7.1. Définition

Les **tests** sont un élément essentiel à tout langage informatique si on veut lui donner un peu de complexité car ils permettent à l'ordinateur de prendre des décisions. Pour cela, Python utilise l'instruction if . Voici un exemple :

```
>>> x = 2
>>> if x == 2:
...     print("Le test est vrai !")
...
Le test est vrai !
```

Figure (III.18) exemple de l'instruction if

VIII.7.2. Tests à plusieurs cas

Parfois, il est pratique de tester si la condition est vraie ou si elle est fausse dans une même instruction **if**. Plutôt que d'utiliser deux instructions **if**, on peut se servir des instructions **if** et **else** :

```
>>> x = 2
>>> if x == 2:
...     print("Le test est vrai !")
... else:
...     print("Le test est faux !")
...
Le test est vrai !
>>> x = 3
>>> if x == 2:
...     print("Le test est vrai !")
... else:
...     print("Le test est faux !")
...
Le test est faux !
```

Figure (III.19) exemple de l'instruction if et else

On peut utiliser une série de tests dans la même instruction **if**, notamment pour tester plusieurs valeurs d'une même variable.

VIII.7.3. Instructions `break` et `continue`

Ces deux instructions permettent de modifier le comportement d'une boucle (**for** ou **while**) avec un test. L'instruction **break** stoppe la boucle.

```
>>> for i in range(5):
...     if i > 2:
...         break
...     print(i)
...
0
1
2
```

Figure (III.20) exemple de l'instruction **`break`**

L'instruction **continue** saute à l'itération suivante, sans exécuter la suite du bloc d'instructions de la boucle.

```
>>> for i in range(5):
...     if i == 2:
...         continue
...     print(i)
...
0
1
3
4
```

Figure (III.21) exemple de l'instruction **`continue`**

VIII.8. Modules

VIII.8.1. Définition

Les modules sont des programmes Python qui contiennent des fonctions que l'on est amené à réutiliser souvent (on les appelle aussi bibliothèques ou *libraries*). Ce sont des « boîtes à outils » qui vont vous être très utiles.

Les développeurs de Python ont mis au point de nombreux modules qui effectuent une quantité phénoménale de tâches. Pour cette raison, prenez toujours le réflexe de vérifier si une partie du code que vous souhaitez écrire n'existe pas déjà sous forme de module.

La plupart de ces modules sont déjà installés dans les versions standards de Python.

VIII.8.2. Quelques modules courants

Il existe une série de modules que vous serez probablement amenés à utiliser si vous programmez en Python. voici une liste non exhaustive.

- *math* : fonctions et constantes mathématiques de base (sin, cos, exp, pi...).
- *Sys* : interaction avec l'interpréteur Python, passage d'arguments (cf. plus bas).
- *os* : dialogue avec le système d'exploitation.
- *random* : génération de nombres aléatoires.
- *time* : accès à l'heure de l'ordinateur et aux fonctions gérant le temps.
- *url lib* : récupération de données sur internet depuis Python.
- *Tkinter* : interface python avec Tk. Création d'objets graphiques
- *re* : gestion des expressions régulières

Enfin, notez qu'il existe de nombreux autres modules externes qui ne sont pas installés de base dans Python mais qui sont très utilisés en bio-informatique et en analyse de données. Citons-en quelques-uns : *NumPy* (manipulations de vecteurs et de matrices, algèbre linéaire), *Bio python* (recherche dans les banques de données biologiques, manipulation de séquences ou de

structures de biomolécules), *matplotlib* (représentations graphiques : courbes, nuages de points, diagrammes en bâtons...), *pandas* (analyse de données)...

VIII.9. Fonctions

En programmation, les fonctions sont très utiles pour réaliser plusieurs fois la même opération au sein d'un programme. Elles rendent également le code plus lisible et plus clair en le fractionnant en blocs logiques.

Pour définir une fonction, Python utilise le mot-clé **def**. Si on souhaite que la fonction renvoie quelque chose, il faut utiliser le mot-clé **return**. Par exemple :

```
1  >>> def carre(x):
2  ...     return x**2
3  ...
4  >>> print(carre(2))
5  4
```

Figure (III.22) exemple de fonction

Notez que la syntaxe de **def** utilise les deux points comme les boucles **for** et **while** ainsi que les tests **if**, un bloc d'instructions est donc attendu. De même que pour les boucles et les tests, l'indentation de ce bloc d'instructions est obligatoire.

VIII.9.1. Passage d'arguments

Le nombre d'arguments que l'on peut passer à une fonction est variable. Nous avons vu ci-dessus des fonctions auxquelles on passait 0 ou 1 argument. Souvenez-vous par exemple de **range** (1, 10) ou encore **range** (1, 10, 2). Le nombre d'argument est donc laissé libre à l'initiative du programmeur qui développe une nouvelle fonction.

Une particularité des fonctions en Python est que vous n'êtes pas obligé de préciser le type des arguments que vous lui passez, dès lors que les opérations que vous effectuez avec ces arguments sont valides. Python est en effet connu comme étant un langage au « typage dynamique », c'est-à-dire qu'il reconnaît pour vous le type des variables au moment de l'exécution. Par exemple :

```
>>> def fois(x, y):
...     return x*y
...
>>> fois(2, 3)
6
>>> fois(3.1415, 5.23)
16.430045000000003
>>> fois("to", 2)
'toto'
>>> fois([1,3], 2)
[1, 3, 1, 3]
```

Figure (III.23) exemple de passage d'arguments

L'opérateur * reconnaît plusieurs types (entiers, *floats*, chaînes de caractères, listes). Notre fonction **fois** () est donc capable d'effectuer des tâches différentes ! Même si Python autorise cela. En général, il est plus judicieux que chaque argument ait un type précis (entiers, *floats*, chaînes de caractères, etc.) et pas l'un ou l'autre.

VIII.9.2. Renvoi de résultats

Un énorme avantage en Python est que les fonctions sont capables de renvoyer plusieurs objets à la fois, comme dans cette fraction de code :

```
>>> def carre_cube(x):
...     return x**2, x**3
...
>>> carre_cube(2)
(4, 8)
```

Figure (III.24) exemple de renvoi de résultats

En réalité Python ne renvoie qu'un seul objet, mais celui-ci peut être séquentiel, c'est-à-dire contenir lui-même d'autres objets. Dans notre exemple Python renvoie un objet de type **`tuple`**, Notre fonction pourrait tout autant renvoyer une liste

:

```
>>> def carre_cube2(x):
...     return [x**2, x**3]
...
>>> carre_cube2(3)
[9, 27]
```

Figure (III.25) exemple de renvoie un objet de type **`tuple`**

IX. CHAPITRE IV

Résultats et discussion

IX.1. INTRODUCTION

Dans ce chapitre consacré aux résultats, nous allons explorer le design d'une tuyère TIC (truncated ideal contour) en utilisant la méthode des caractéristiques (MOC), et ce en utilisant le langage de programmation Python. Une fois que nous aurons obtenu le design souhaité, nous procéderons à l'étude de l'évolution des paramètres thermodynamiques de l'écoulement à travers la tuyère, tels que la pression, le nombre de Mach et la température. Les résultats obtenus seront ensuite comparés à ceux obtenus à l'aide du langage Fortran et des simulations réalisées avec l'outil ANSYS. Cette comparaison nous permettra de valider nos résultats et de mieux comprendre les performances de notre approche basée sur Python.

Les programmes informatiques (Fortran et Python MOC) nécessitent la spécification de certains paramètres géométriques et thermodynamiques. Les paramètres géométriques comprennent le rayon de la gorge (Rth) fixé à 0,01 m, le rayon de courbure amont (Rtu) égal à 0,03 m et le rayon de courbure aval (Rtd) également égal à 0,03 m.

En ce qui concerne les paramètres thermodynamiques, les valeurs sont les suivantes : la pression de stagnation (pression totale) (représentant la pression totale de la chambre de combustion) est fixée à P0 = 30 bars, et la température de stagnation (température totale) (représentant la température totale de la chambre de combustion) est T0 = 243 K.

Le fluide considéré est l'air, avec un rapport de chaleur spécifique (gamma) de 1,4 et une capacité thermique spécifique (Cp) de 1006,43 J/kg·K. Ces paramètres thermodynamiques jouent un rôle crucial dans la détermination du comportement et des caractéristiques de l'écoulement d'air à l'intérieur de la tuyère TIC.

En spécifiant avec précision ces paramètres géométriques et thermodynamiques, les programmes informatiques seront en mesure d'effectuer les calculs et simulations nécessaires pour l'analyse et la conception de la tuyère TIC.

IX.2. Résultats obtenus en utilisant le langage python

Lorsqu'il s'agit de réaliser une étude de tuyère en utilisant Python, la performance dépendra principalement des techniques de programmation utilisées, de l'efficacité des algorithmes et de la qualité de l'implémentation.

Cette étude présente une approche basée sur Python pour le design et l'analyse des performances d'une tuyère de moteur d'avion. Ont utilisé des coordonnées spécifiques le long de

l'axe x et y de la tuyère pour modéliser les variations de température, pression, nombre de Mach, vitesse et densité. Ont développé un code Python pour calculer ces paramètres et évaluer l'efficacité de la tuyère.

IX.2.1. Désigne de la tuyère

La fig. 1 illustre le profil de la tuyère obtenu par la méthode des caractéristiques programmé en utilisant le langage Python.

Ces résultats sont obtenus en utilisant les paramètres ci-dessous

1. Paramètres géométriques :
    - Rayon au col (Rth) = 0,01 m
    - Rayon de courbure amont (Rtu) = 0,03 m
    - Rayon de courbure aval (Rtd) = 0,03 m
2. Paramètres thermodynamiques :
    - Pression de stagnation (pression totale) (Pression totale de la chambre de combustion) P0 = 30 bars
    - Température de stagnation (température totale) (Température totale de la chambre de combustion) T0 = 243 K
    - Fluide = Air avec un rapport de chaleur spécifique (gamma) égal à 1,4 et une capacité thermique spécifique Cp = 1006,43 J/kg·K

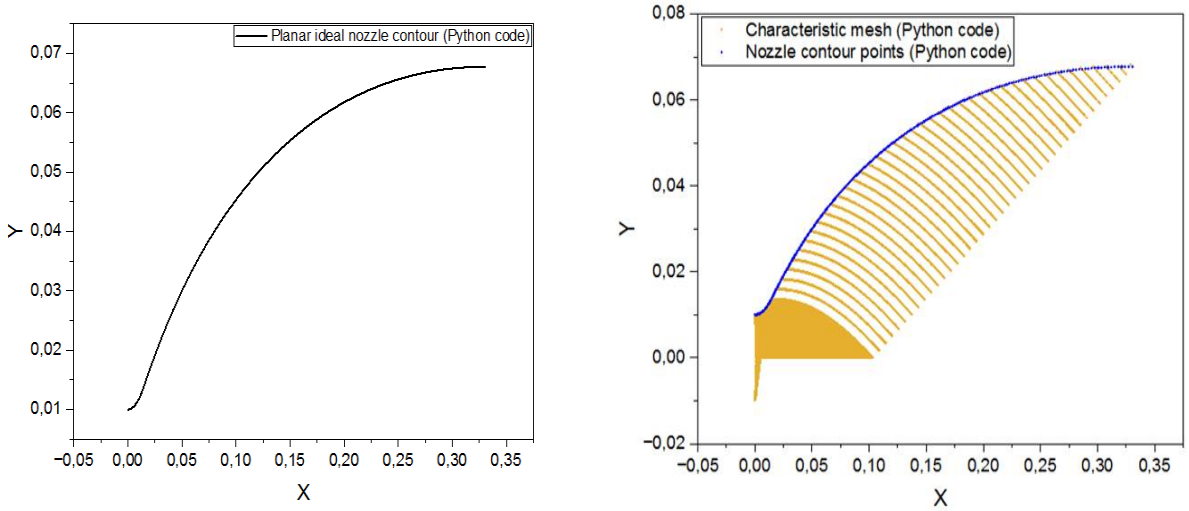

Figure. (IV.1) Résultats de conception de la tuyère idéale plane avec la méthode des caractéristiques en utilisant le code Python.

Les caractéristiques géométriques de la tuyère idéale de contour plan obtenues à l'aide du programme basé sur la méthode des caractéristiques en Python sont les suivantes :

- Longueur de la tuyère : 0,330439 m
- Rayon de sortie de la tuyère : 0,067733 m

Ces mesures fournissent des informations essentielles sur les dimensions de la tuyère idéale de contour plan conçue à l'aide du programme Python. Elles déterminent la taille et la forme de la tuyère, ce qui influence directement les caractéristiques et les performances de l'écoulement à travers celle-ci. Ces résultats sont le fruit de l'application précise et rigoureuse de la méthode des caractéristiques dans le programme Python.

IX.2.2. EVOLUTION DES PARAMETRES DE L'ECOULEMENT DANS LA TUYERE OBTENUE

- **Le nombre de mach**

La Figure 2 présente de manière détaillée le contour de la tuyère TIC ainsi que l'évolution du nombre de Mach le long de sa paroi. Il est intéressant de noter que le nombre de Mach augmente progressivement à mesure que l'on se déplace le long de la tuyère. De plus, une observation importante est que cette augmentation est plus rapide sur la paroi de la tuyère que le long de son axe central. Cette observation est cohérente avec les principes bien établis de la forme du profil de vitesse dans une tuyère, qui est généralement de nature parabolique. Cette forme parabolique est le résultat de la distribution optimale des vitesses du flux de gaz à l'intérieur de la tuyère, ce qui contribue à maximiser l'efficacité de la conversion de l'énergie thermique en poussée propulsive. La compréhension de ces caractéristiques et de leur évolution le long de la tuyère est cruciale pour une conception précise et efficace des tuyères dans les systèmes de propulsion des fusées.

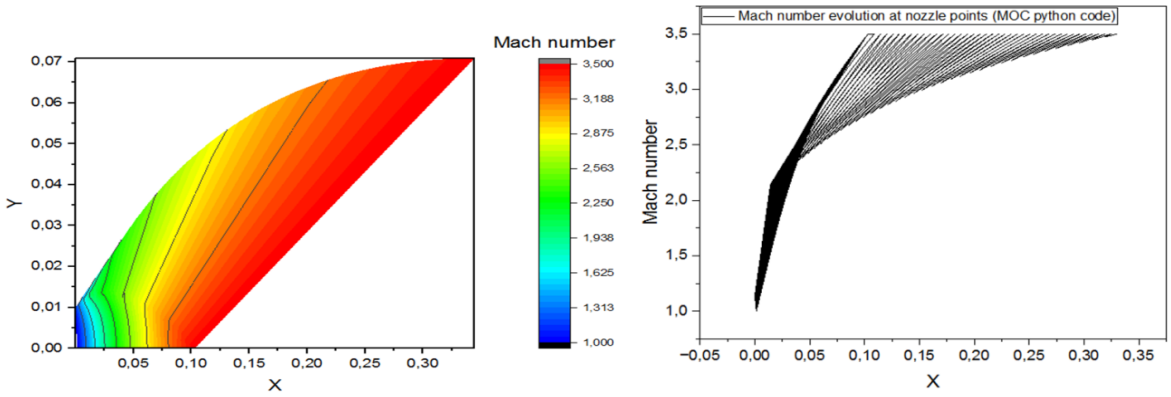

Figure. (IV.2) Contour et courbes d'évolution du nombre de Mach pour une tuyère idéale plane calculés avec la méthode des caractéristiques programmée sur le langage Python.

Une observation supplémentaire importante est que l'augmentation du nombre de Mach le long de la paroi de la tuyère, au niveau du col, se produit de manière abrupte. Cela s'explique par le concept de conception des tuyères profilées, où l'objectif est d'obtenir des gradients élevés dans la zone d'expansion initiale près du col, puis de ramener progressivement le flux de gaz vers la sortie de la tuyère dans une direction axiale.

À la sortie de la tuyère, il est également remarquable que le nombre de Mach sur l'axe central est supérieur à celui le long de la paroi. Cette différence est principalement due aux pertes engendrées par les frottements entre le fluide et la paroi de la tuyère. Ces pertes de rendement peuvent résulter de la viscosité du fluide ainsi que des effets de turbulence et de la rugosité de la surface interne de la tuyère.

Il est essentiel de prendre en compte ces aspects lors de la conception des tuyères, car ils ont un impact direct sur les performances et l'efficacité globale du système de propulsion. Des efforts sont donc déployés pour minimiser les pertes de rendement, améliorer la transition du profil de vitesse et optimiser la forme de la tuyère afin d'obtenir une conversion maximale de l'énergie thermique en poussée propulsive.

- **La pression**

La Fig.3 montre le contour l'évolution de la pression le long de la paroi de cette tuyère, on constate que la pression diminue sur la paroi et l'axe, du col jusqu'à la section de sortie où elle se rapproche de la valeur de la pression atmosphérique, ce qui est très normale dans notre cas puisque la tuyère est du type adaptée**.**

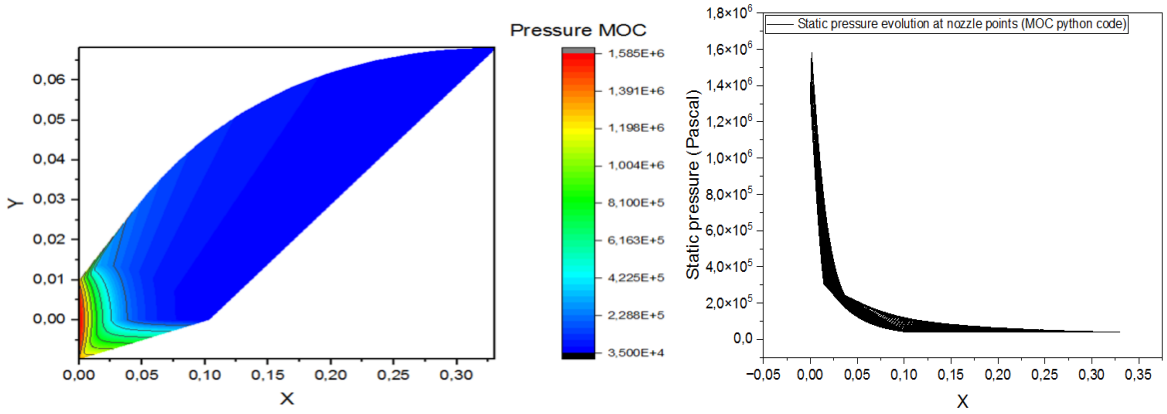

Figure (IV.3) Contour et courbes d'évolution de la pression statique pour une tuyère idéale plane calculés avec la méthode des caractéristiques programmée sur le langage Python.

L'étude de la pression exercée sur la paroi de la tuyère revêt une importance capitale, car elle permet de calculer les charges latérales qui agissent sur les parois de la tuyère tout au long de son évolution en altitude. La connaissance de ces charges est essentielle pour assurer la résistance et l'intégrité structurelle de la tuyère.

Au fur et à mesure que la fusée s'élève en altitude, la pression atmosphérique diminue. Cette diminution de pression a un impact direct sur la pression exercée sur la paroi de la tuyère. En effet, la différence de pression entre l'intérieur de la tuyère et l'extérieur crée une force latérale qui est transmise aux parois de la tuyère.

La distribution de cette force latérale le long de la tuyère dépend de plusieurs facteurs, tels que la forme de la tuyère, la géométrie du flux de gaz et les caractéristiques de l'écoulement. Des études approfondies sont menées pour modéliser et prédire ces charges latérales, en tenant compte des variations de pression tout au long de l'évolution en altitude.

La compréhension précise de ces charges latérales permet aux ingénieurs de concevoir des tuyères robustes et résistantes capables de supporter ces forces. Cela implique l'utilisation de matériaux appropriés, des conceptions structurelles adéquates et des marges de sécurité suffisantes pour faire face aux variations de pression et aux contraintes mécaniques induites.

- **La température**

L'analyse de l'évolution de la température le long de la tuyère revêt une importance cruciale, car elle est un paramètre essentiel pour le calcul des performances et le choix des matériaux de construction.

Pour un même nombre de Mach, la température varie tout au long de la tuyère en raison des processus de combustion et de l'expansion des gaz de propulsion. Au point d'entrée de la tuyère, la température est généralement très élevée en raison de la combustion intense qui se produit dans la chambre de combustion. Au fur et à mesure que les gaz de combustion s'écoulent le long de la tuyère, ils se détendent et leur température diminue progressivement.

La connaissance de l'évolution de la température est essentielle pour le calcul précis des performances du système de propulsion. La température influence directement les propriétés du flux de gaz, notamment sa vitesse, sa densité et sa pression. Ces propriétés sont essentielles pour évaluer la poussée générée par la tuyère et prédire les performances globales du moteur.

De plus, la température joue un rôle crucial dans la sélection des matériaux de construction de la tuyère. Les températures élevées peuvent entraîner des contraintes thermiques importantes sur les matériaux, pouvant éventuellement provoquer leur déformation ou leur dégradation. Par conséquent, il est essentiel de choisir des matériaux résistants à la chaleur et capables de supporter les températures extrêmes rencontrées dans la tuyère.

Les données sur l'évolution de la température le long de la tuyère sont obtenues grâce à des modèles de combustion et des simulations numériques avancées. Ces informations sont utilisées pour optimiser la conception de la tuyère, choisir les matériaux appropriés et garantir des performances optimales du système de propulsion.

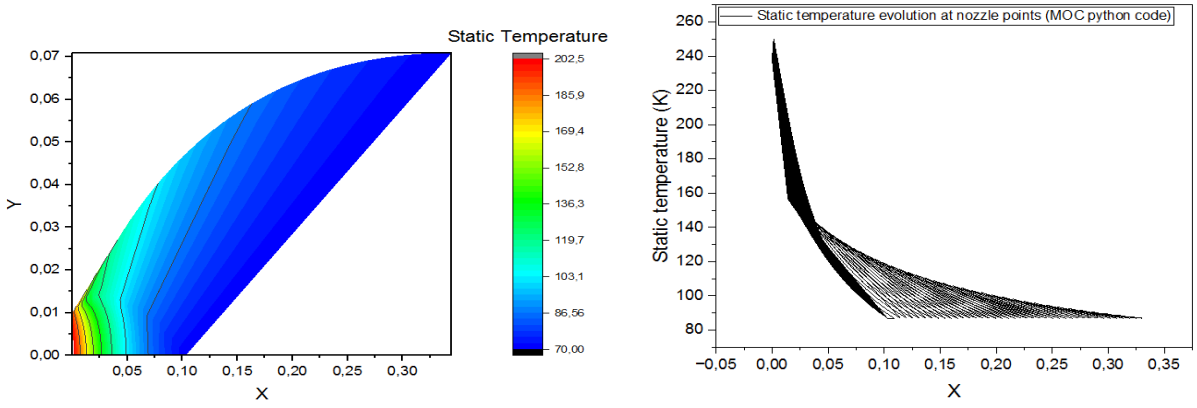

Figure. (IV.4) Contour et courbes d'évolution de la température statique pour une tuyère idéale plane calculés avec la méthode des caractéristiques programmée sur le langage Python.

Il est effectivement observé que la température statique diminue le long de la tuyère en raison de l'augmentation de la vitesse du flux de gaz. Cette corrélation entre la température et la vitesse est illustrée par nos résultats présentés dans la Figure 4.

Lorsque les gaz de combustion s'écoulent à travers la tuyère, ils subissent une expansion progressive, ce qui entraîne une augmentation de leur vitesse. Selon le principe de la conservation de l'énergie, cette augmentation de la vitesse s'accompagne d'une diminution de la température statique du flux de gaz. Cette diminution de température est due à la conversion de l'énergie thermique en énergie cinétique, conforme aux principes fondamentaux de la thermodynamique.

Les résultats affichés dans la Figure 4 illustrent cette variation de température le long de la tuyère. Ils mettent en évidence la diminution progressive de la température statique à mesure que le flux de gaz s'accélère. Cette observation est cruciale pour comprendre les phénomènes thermodynamiques qui se produisent à l'intérieur de la tuyère et pour évaluer les performances du système de propulsion.

En étudiant cette évolution de la température statique, les ingénieurs et les chercheurs peuvent affiner les modèles de conception de la tuyère, optimiser les performances du moteur et prendre des décisions éclairées concernant le choix des matériaux. En comprenant comment la température évolue en fonction de la vitesse du flux de gaz, il est possible de garantir une conception thermiquement stable, de prévenir les problèmes de surchauffe ou de dégradation des matériaux, et de maximiser l'efficacité globale du système de propulsion.

- **La densité**

La Figure 5 offre une représentation visuelle de l'évolution de la masse volumique des gaz émanant de la chambre de combustion. Cette courbe est essentielle pour comprendre la composition et les propriétés du flux de gaz tout au long de son passage dans la tuyère.

La masse volumique des gaz est une mesure de la densité du fluide, indiquant la quantité de matière présente par unité de volume. Dans le contexte des systèmes de propulsion, la masse volumique des gaz peut varier en fonction de plusieurs facteurs, tels que la composition des gaz de combustion, la pression et la température.

Dans la chambre de combustion, où se produit la réaction chimique de combustion, la masse volumique des gaz est relativement élevée en raison de la présence de produits de combustion et de la pression élevée. Cependant, à mesure que les gaz s'écoulent dans la tuyère, ils subissent une expansion et une détente, ce qui entraîne une diminution de la masse volumique.

La Figure 5 met en évidence cette diminution progressive de la masse volumique des gaz à mesure qu'ils traversent la tuyère. Cette évolution est influencée par les variations de pression et

de température tout au long du trajet du flux de gaz. La courbe obtenue dans la Figure 5 permet de visualiser ces variations et de déterminer comment la densité du flux de gaz évolue.

La connaissance de l'évolution de la masse volumique des gaz est d'une grande importance pour de nombreux aspects de la conception et de l'analyse des systèmes de propulsion. Elle est utilisée pour calculer la poussée générée par la tuyère, évaluer les performances du moteur et prévoir le comportement du flux de gaz dans différentes conditions de fonctionnement.

De plus, la variation de la masse volumique a des implications directes sur le choix des matériaux de construction de la tuyère. Les matériaux doivent être capables de résister aux variations de densité et de supporter les contraintes mécaniques induites par ces variations.

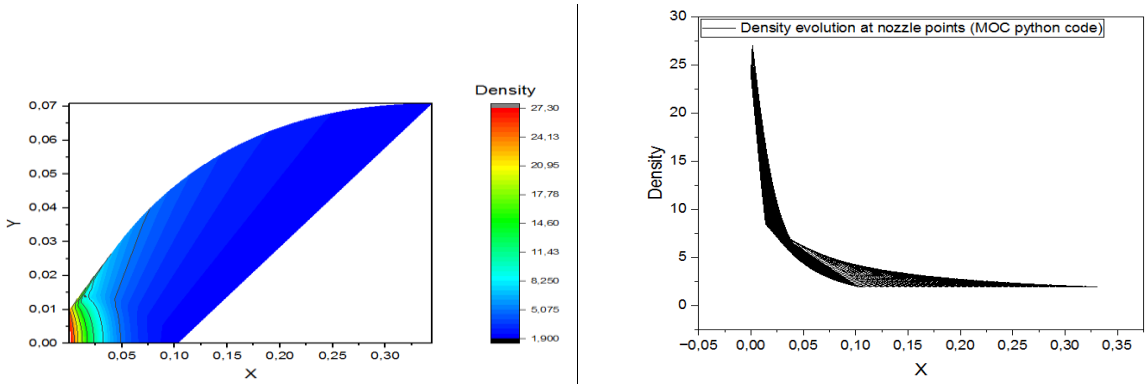

Figure. (IV.5) Contour et courbes d'évolution de la densité pour une tuyère idéale plane calculés avec la méthode des caractéristiques programmée sur le langage Python.

IX.3. Etude numérique (simulation) de l'écoulement dans la tuyère obtenue par python

IX.3.1. INTRODUCTION

La simulation numérique de la dynamique des fluides est utilisée de plus en plus comme outil d'analyse en recherche et en industrie. Cependant, la génération de maillage demeure un défi majeur, en particulier pour la simulation d'écoulements turbulents, car il peut être très difficile et coûteux en temps de générer un maillage qui produira des résultats précis. La modélisation et la simulation interviennent pour :

- La compréhension de la structure et des interactions à l'intérieur d'un système (déterminer le rendement, la performance…) ;
- L'étude du comportement du système par rapport à son environnement extérieur (consommation énergétique/cout…) ;
- La prédiction du comportement d'un système pour des situations nouvelles ou extrêmes;

- La conception de nouveaux dispositifs/ composants, étude de système avant la création de prototype et mise en œuvre de nouveaux procédés (stratégies et algorithmes de contrôle);
- L'optimisation des solutions lors de la conception.

Les développements et les progrès réalisés au cours des deux dernières décennies ont conduit à l'apparition d'une méthodologie qui est devenue standard. Comme pour tout système complexe, la clef de la maîtrise réside dans l'identification et la modularisation des tâches. Actuellement, la méthodologie standard découpe le processus de simulation en quatre tâches distinctes, qui sont :

- Modélisation Géométrique ;
- Maillage ;
- Résolution ;
- Analyse et Visualisation.

La simulation ne supplante pas l'expérimentation mais la complète. Elle permet d'effectuer l'analyse du problème dans des conditions réalistes (reproduire des tests que l'on fait en expérimentation pour mieux les comprendre et à moindre cout) ou au contraire dans des conditions d'essais extrêmes/marginales (climats extrêmes, défauts d'installations…).

A travers la simulation, le système étudié devient plus flexible. On peut facilement faire des études paramétriques. L'utilisateur peut aussi faire varier l'échelle de temps pour une étude donnée, ce qui est impossible par ailleurs. La simulation se présente presque toujours sous forme d'un programme ou des outils informatiques. Ces derniers sont couramment appelés des environnements de simulation.

Dans ce chapitre, nous présenterons des cas tests servant à valider nos modèles de calculs qui seront utilisés par la suite, dans le Chapitre 5, pour optimiser le profil d'une tuyère double galbe. Nous allons utiliser l'environnement ANSYS-ICEM pour générer les maillages et l'environnement ANSYS-Fluent pour résoudre les équations de nos modèles. La validation sera basée sur des bases de données expérimentales et numériques issues de notre revue de littérature.

1. GEOMETRIE

Plus les détails géométriques d'une application cible peuvent être déterminés, plus les résultats obtenues de la simulation précisent le champ d'écoulement ; Cela ne veut pas dire que toutes les composantes géométriques doivent être modelés. Pour bien résoudre les détails

géométriques d'une configuration, plus de points de maillage seront exigés, et comme résultat, de plus longs délais d'exécution seront exigés. Le niveau de précision auquel la géométrie est modelée dépend du type des résultats exigés et du temps d'exécution acceptable

Le fait d'avoir le profil de la tuyère (pour tous nos cas tests) sous forme une liste de coordonnées, toutes nos géométries ont été générées dans l'environnement de travail ANSYS-ICEM. Notons que cela a l'avantage d'éviter toute un travail supplémentaire de nettoyage de la géométrie. D'un autre côté, toute discontinuité susceptible lors de la conversion de format de fichiers (i.e. CATIA→Igs ou SOLIDWORKS →Igs) est évitée.

2. MAILLAGE

La génération du maillage est primordiale quant à la réussite des calculs et à la précision des résultats. Il diffère fortement selon le problème et les calculs que nous allons vouloir effectuer. Il sert à la représentation discrète de variables continues. Toutefois, un bon maillage est un compromis entre la précision recherchée et le temps de calcul.

Dans cette partie, nous avons pour objectif de créer un maillage qui, sous la base de nos connaissances en mécanique des fluides et en méthodes numériques, serait approprié à des calculs à grand nombre de Mach, e.g. affiner le maillage dans les zones où nous nous attendons à de forts gradients. Notons que nos maillages ont été générés dans l'environnement ANSYS ICEM CFD selon la procédure suivante :

- Dessiner ou importer une géométrie ;
- Définir un domaine de calcul;
- Créer un blocking dans le cas d'un maillage structuré ;
- Générer le pré-maillage puis le maillage;
- Convertir le maillage structuré en un maillage non structuré;
- Proposer les conditions aux limites, i.e. segmenter le domaine de calcul à des zones qu'on puisse attribuer des conditions au limites;
- Lorsque le maillage est terminé et que les conditions aux limites sont correctement imposées, il ne reste qu'à exporter ce maillage sous un format lisible par le solveur, Fluent en l'occurrence. Ceci se fait par la commande File/Export/Mesh qui permet d'écrire un fichier de maillage avec l'extension ".msh".

3. CONDITIONS AUX LIMITES

À toutes les frontières du volume de contrôle qui sont des interfaces entre les régions dans lesquelles l'écoulement sera simulé et les régions en dehors du domaine de calcul, les propriétés du fluide et de l'écoulement doivent être spécifiées. Souvent, simplement la spécification du type de la paroi, e.g. paroi adiabatique, paroi de non glissement … etc., est suffisante, et dans d'autre cas, des informations complémentaires sont exigées, e.g. la température de la paroi

Les écoulements étudiés dans ce travail s'effectuent dans des domaines confines limités par la paroi de la tuyère. Nous avons, donc, à préciser pour chaque cas d'écoulement, quatre types de conditions aux limites à savoir :

- d'écoulement à l'entrée du domaine.
- Conditions d'écoulement à la sortie du domaine.
- Conditions de symétrie.
- Conditions aux parois solides.

Le nombre de conditions à l'entrée ou à la sortie dépend de la nature locale d'écoulement, i.e. supersonique ou subsonique.

Conditions d'entrée

Si l'écoulement à l'entrée est subsonique, trois conditions sont à imposer :

- Pression totale ou de stagnation ; P0.
- Température totale ou de stagnation ; T0.
- Vitesse transversale nulle ; u2 = 0.

Pour un écoulement supersonique à l'entrée, le nombre de Mach doit être spécifié et par conséquent, quatre conditions sont nécessaires :

- Pression totale ; P0.
- Température totale ; T0.
- Vitesse transversale nulle ; u2 = 0.
- Vitesse axiale u1 correspondante au nombre de Mach spécifié.

La pression totale P0 n'est pas une variable indépendante. Cependant, la valeur P0 désirée est assurée par l'imposition de la pression statique Ps correspondante. Ainsi, la pression statique sera imposée de manière non linéaire. Elle est alors actualisée tout le long de calcul de façon à garantir la valeur de la pression totale désirée, [115].

Conditions de sortie

En général, une seule condition à la sortie est imposée pour les écoulements internes. Cette condition correspond à la pression statique à la sortie.

Condition de symétrie

Les écoulements étudiés sont des écoulements bidimensionnels à symétrie plane. Seule la moitié du domaine fluide est alors considérée et la condition de symétrie est assurée en imposant une vitesse (donc la quantité de mouvement) transversale nulle le long du plan de symétrie.

Conditions aux parois solides

Les conditions aux parois solides sont différentes selon que l'écoulement est visqueux ou non.

Écoulement non-visqueux : Dans un écoulement non visqueux, les particules fluides doivent glisser au contact d'une paroi solide. Cette condition de glissement est assurée grâce à la condition d'imperméabilité de la paroi qui se traduit par une vitesse normale du fluide nulle le long de cette paroi.

Écoulement visqueux Si l'écoulement est visqueux, au contact d'une paroi solide, la viscosité impose que les particules fluides restent attachées à cette paroi. La vitesse du fluide est alors nulle sur la paroi

$$u = 0 \quad \text{et} \quad u = 0$$

En ce qui concerne la condition d'une paroi adiabatique, nous imposons dans les deux cas d'écoulements, l'une des deux possibilités suivantes:

- Soit en imposant une répartition uniforme de la température, égale à la température de l'entourage (système extérieur), sur la paroi d'où, une paroi isotherme.
- Soit en isolant la paroi de l'extérieur en considérant que le flux de chaleur normale à la paroi soit nul. La paroi est alors dite adiabatique.

4. RESOLUTION

L'étape de la résolution est effectuée avec le solveur Fluent. Ce dernier fourni, en générale, de bons résultats en simulation des écoulements internes, [158]. L'un des intérêts de ce

logiciel de simulation, est qu'il dispose d'un nombre relativement important de modèles, pouvant faire face à divers aspects de la mécanique des fluides : écoulements diphasiques (miscible, non miscible, cavitation, solidification), turbulence (LES, k-ε, k-ω, S-A, Reynolds Stress…), combustion (pré-mélangé et non pré-mélangé), transport de particules, écoulements en milieux poreux, maillages mobiles et dynamiques avec reconstruction du maillage, entre autres. Les schémas numériques temporels et spatiaux peuvent être modifiés pour améliorer la convergence. Fluent est parallélisé et il permet de tirer parti des systèmes multiprocesseurs aussi bien au sein d'une seule machine qu'en réseau. Les équations gouvernantes utilisées dans ce logiciel sont formulées en utilisant l'approche de volume fini.

Les possibilités de visualisation sont nombreuses, on peut par exemple tracer les valeurs du coefficient de frottement pariétal afin de détecter un éventuel décollement, visualiser des lignes de courant ou d'autres paramètres de l'écoulement et de la turbulence. Il est également possible de tracer les contours de différentes variables qui décrivent l'écoulement : pression, vitesse, variables turbulentes, en utilisant la fonction display/contours.

IX.3.2. Application au cas d'étude

La mécanique des fluides numérique ou *computational fluid dynamics* (CFD) est devenu indispensable pour comprendre les différents phénomènes physiques intervenants dans les écoulements de fluides. Ces écoulements sont régis par des équations de conservation. Elles forment un système d'équations de conservation différentielles aux dérivés partielles (EDP) non linéaire qu'on doit résoudre. Cela est difficile à résoudre pour la majorité des problèmes rencontrés dans le domaine de l'ingénierie. Cependant il est possible d'obtenir une approximation de la solution de ces systèmes à l'aide des méthodes numériques en utilisant les moyens informatiques. Modèle de calcule ANSYS 14.5 est un outil de conception assistée par ordinateur (CAO) qui permet la conception et la génération des géométries en 3D/2D et appliquer des simulations. Il permet de construire des surfaces et des volumes à partir d'une série de points définissant la géométrie de base. Une fois la géométrie construite, elle peut être exportée en différents formats vers le générateur de maillage et les solveurs pour faire les analyses ou les simulations. ANSYS 19 est un logiciel de génie mécanique qui collabore avec différents types de logiciels utilisés dans différents domaines comme : Vibrations, mécanique des fluides, aérodynamique, transfert de chaleur…etc. Pour notre étude, deux systèmes de

composants d'ANSYS 14.5 sont utilisés pour effectuer le calcul aérothermique ; GEOMETRY, ICEM CFD et un système d'analyse FLUENT. FLUENT est un solveur qui utilise un maillage non structuré 2D ou 3D (avec la méthode de volume fini). Ces maillages sont : soit des maillages triangulaires (tétraédriques en 3D), soit des maillages structurés interprétés en format non structuré comme des rectangles (hexaèdres), pour une simulation de tous les écoulements compressibles ou incompressibles, impliquant des phénomènes physiques complexes tels que la turbulence, le transfert thermique, les réactions chimiques, les écoulements multiphasiques… sur les géométries industrielles. L'analyse en 2D ou 3D montre la capacité de FLUENT à simuler les caractéristiques de l'écoulement autour des tuyères supersonique. Ce problème est résolu en utilisant les équations d'Euler.

Le cas d'étude concerne une tuyère obtenue à l'aide de la méthode des caractéristiques en utilisant du code Python. Les caractéristiques géométriques de cette tuyère, à savoir sa longueur et sa section de sortie, sont données respectivement par 0.330439 m et 0.067733 m, avec un Mach de sortie imposé et égale à 3.5.

Les conditions aux limites pour le calcul de la dynamique des fluides computationnelle (CFD) sont les suivantes :

- Solveur CFD : Fluent
- Type de solveur : Basé sur la densité (plus précis pour les simulations de flux compressibles)
- Modèle spatial : Plan (2D)
- Temps : Stationnaire

Entrée (col)

- Type de limite : Entrée de pression
- Pression totale : 30 bars
- Pression statique : 15,85 bars
- Température totale : 243 K

Sortie (extrémité de la tuyère)

- Type de limite : Sortie de pression

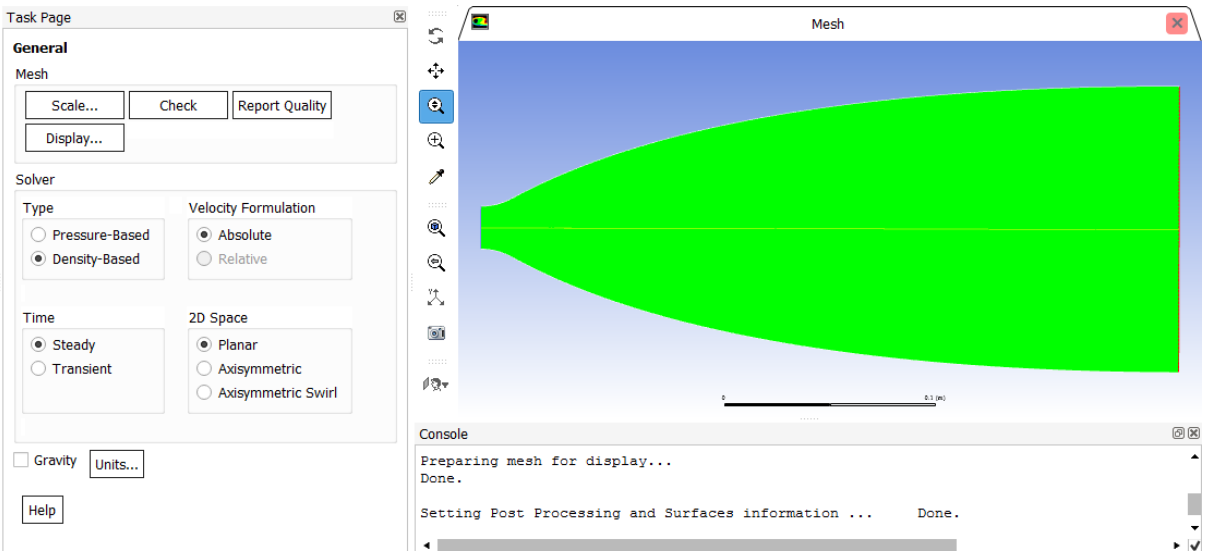

Figure. (IV.6) Configuration du solveur CFD ANSYS Fluent selon le cas d'étude.

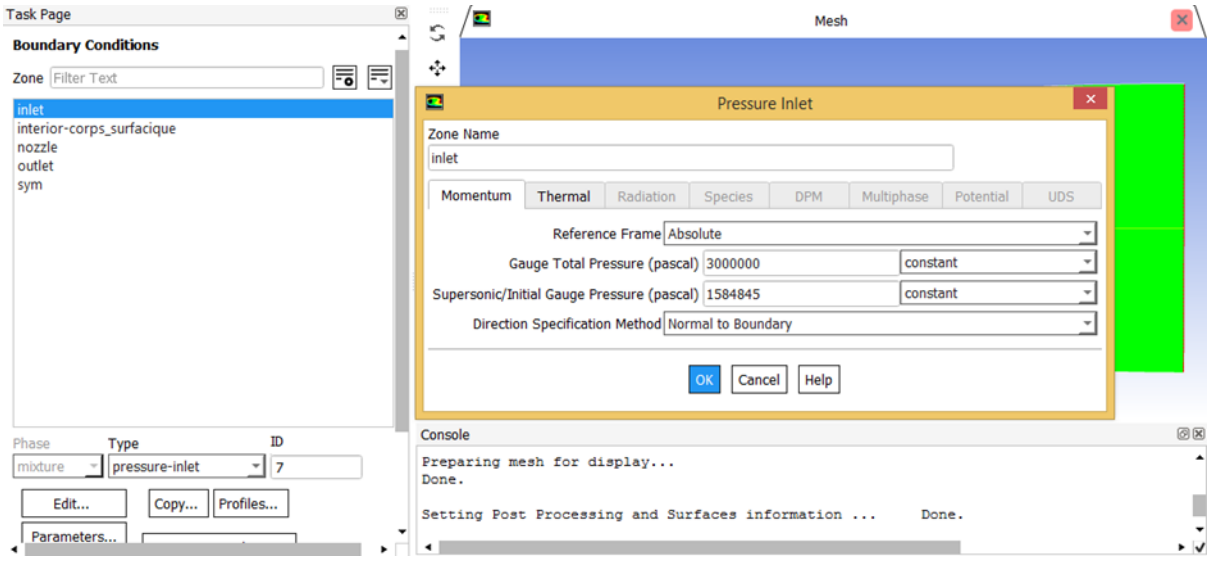

Figure. (IV.7) Conditions (pressions et température) imposées à l'entrée du domaine de calcul (col).

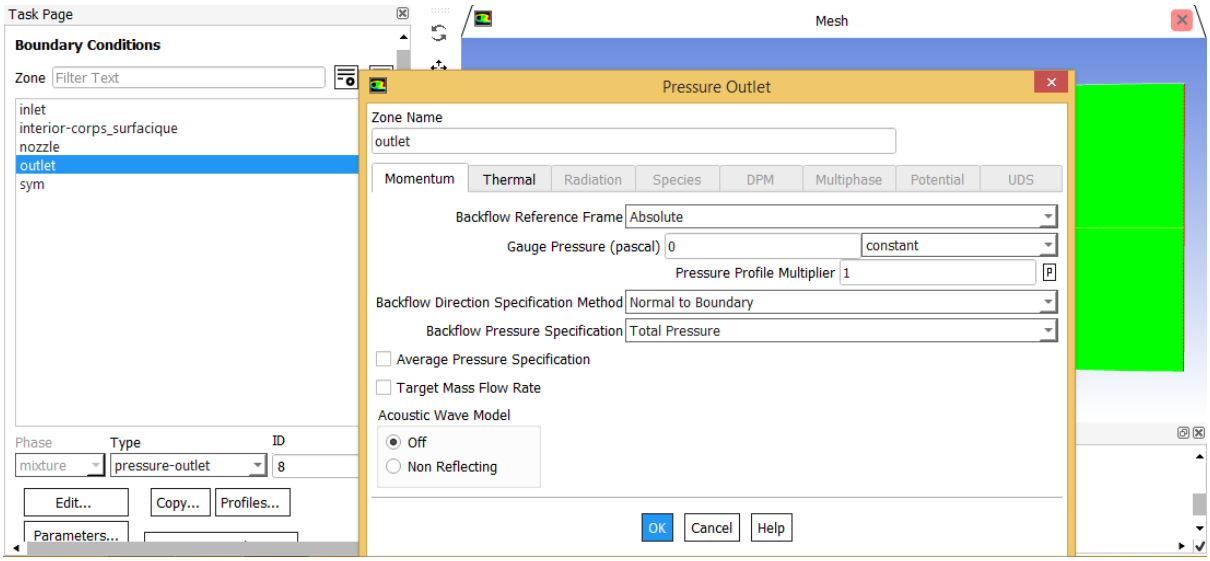

Figure. (IV.8) Conditions (pressions et température) imposées à la sortie de la tuyère.

Le maillage utilisé dans cette étude est de type quadrilatéral (maillage structuré), ce qui signifie que les éléments du maillage sont des quadrilatères. Les statistiques du maillage sont les suivantes :

- Nombre d'éléments : 25 460
- Nombre de nœuds : 26 169

Le choix de ce type de maillage a été justifié par

- **Sa Régularité** : Les maillages structurés sont constitués d'éléments géométriquement réguliers, tels que des quadrilatères ou des hexaèdres. Cela permet une meilleure représentation des géométries complexes avec une distribution régulière des nœuds, facilitant ainsi la résolution numérique.
- **Sa Connectivité simple** : La connectivité des nœuds dans un maillage structuré est généralement plus simple que dans un maillage non structuré. Chaque nœud a des relations de voisinage bien définies, ce qui facilite le calcul des flux et des gradients à travers le maillage.
- **Sa Précision** : Les maillages structurés peuvent fournir une meilleure précision dans certains cas, en particulier lorsque la géométrie du domaine est régulière et que les conditions de bord sont uniformes. La distribution régulière des nœuds permet une interpolation plus précise des variables physiques à travers le maillage.

- **Ses Performances** : Les maillages structurés peuvent être plus efficaces en termes de temps de calcul et de consommation de mémoire par rapport aux maillages non structurés, car ils permettent une utilisation plus efficace des ressources informatiques.
- **Son Adaptabilité** : Les maillages structurés sont plus faciles à adapter et à raffiner localement. Il est possible d'ajuster la résolution du maillage en fonction des régions d'intérêt sans perturber la structure globale du maillage.

Cependant, il convient de noter que les maillages structurés peuvent être plus complexes à générer pour des géométries irrégulières ou complexes, et qu'ils peuvent limiter la flexibilité de modélisation par rapport aux maillages non structurés. Le choix entre un maillage structuré et un maillage non structuré dépend des spécificités de l'application et des contraintes du problème.

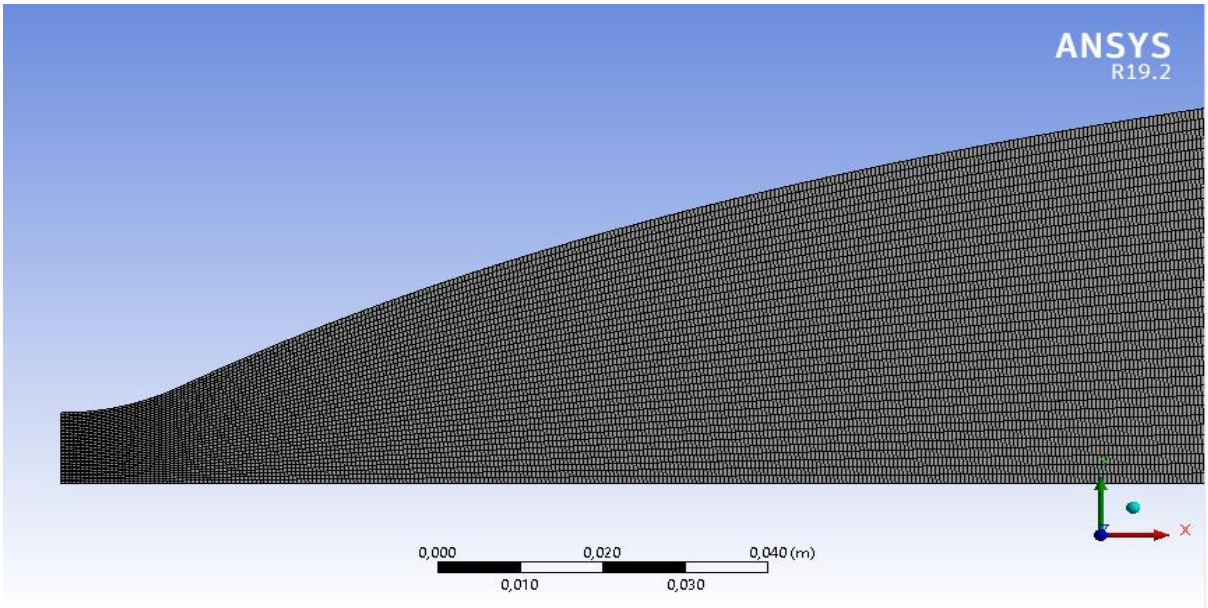

Figure. (IV.9) Maillage structuré à base d'éléments quadrilatères réalisé sur le package ANSYS.

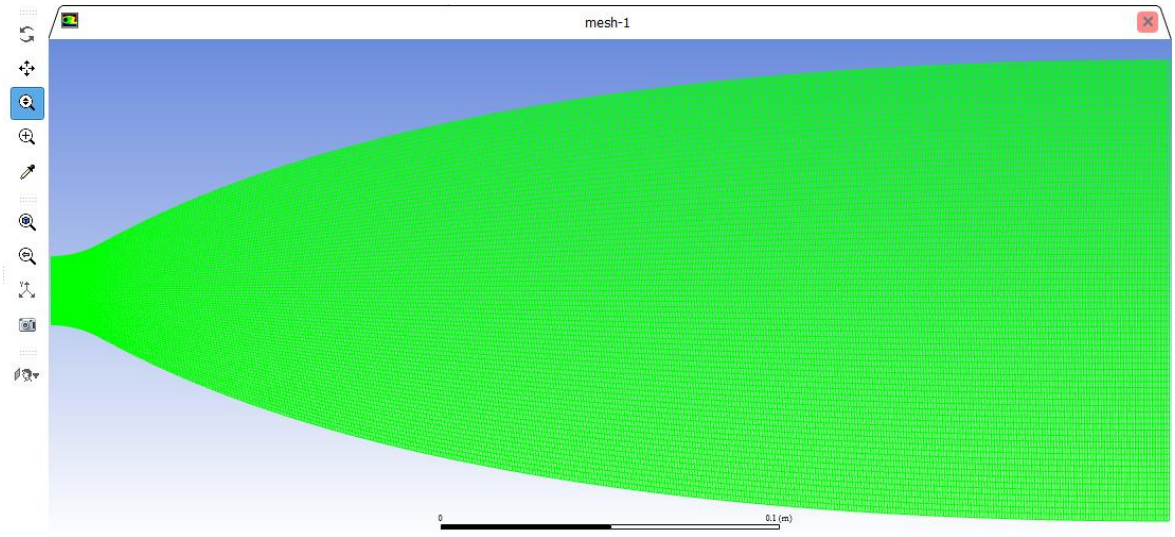

Figure. (IV.10) Visualisation du maillage structuré de la tuyère idéal plane sur ANSYS Fluent.

IX.3.3. RESULTATS ET COMMENTAIRES

- **Contours de Mach**

Les Figures 11 et 12 présentent respectivement le contour et l'évolution de la pression obtenus en utilisant le design spécifique de la tuyère, avec un nombre de Mach imposé de 3.5. Ces résultats ont été obtenus à l'aide d'un code de calcul Python dans l'environnement Fluent 19, qui permet une analyse détaillée du comportement de la tuyère.

La Figure 11 représente le contour de la pression le long de la tuyère. Elle offre une visualisation graphique de la distribution spatiale de la pression à travers la géométrie de la tuyère. Cette information est cruciale pour comprendre comment la pression évolue et se propage tout au long du système de propulsion. Les variations de pression peuvent être observées le long de la paroi de la tuyère, ainsi que dans les différentes sections et zones d'expansion.

D'autre part, la Figure 12 illustre l'évolution de la pression en fonction de la position le long de la tuyère. Cette courbe permet d'analyser comment la pression varie au fur et à mesure que les gaz de combustion se déplacent dans la tuyère. Elle fournit des informations précieuses sur les variations de pression qui se produisent à différents stades de l'écoulement, notamment dans les sections d'entrée et de sortie de la tuyère.

L'utilisation du code de calcul Python dans l'environnement Fluent 19 offre une approche puissante pour modéliser et simuler le comportement de la tuyère. Il permet d'obtenir des résultats précis et détaillés concernant les caractéristiques de la pression, ce qui facilite l'analyse et la compréhension du système de propulsion.

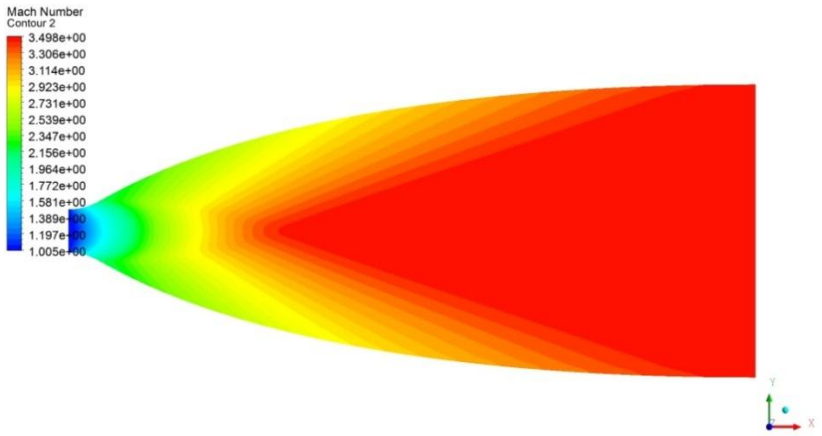

Figure. (IV.11) Contour du nombre de Mach calculé avec le solveur CFD ANSYS Fluent.

Une observation importante est que l'écoulement est uniforme à la sortie de la tuyère, atteignant une valeur très proche du design attendu, avec un nombre de Mach de 3.498, ce qui confirme le succès de notre code de calcul.

En examinant le contour du nombre de Mach le long de la tuyère, on constate une augmentation régulière de sa valeur. À l'entrée de la tuyère, le nombre de Mach est supersonique, atteignant M=1.005, tandis qu'à la sortie, il atteint une valeur de Mach de 3.498, se rapprochant ainsi du nombre de Mach de design de 3.5. Cette évolution du nombre de Mach est cohérente avec les principes de la Méthode des Caractéristiques (MOC), qui est utilisée pour la conception de la tuyère.

En ce qui concerne la région proche de la paroi, il n'y a pas de changements significatifs par rapport aux régions éloignées de la paroi. Cela est dû au fait que l'écoulement dans cette région est considéré comme non visqueux, ce qui signifie qu'il y a une condition de glissement à la paroi. En conséquence, les effets de la viscosité sur l'écoulement sont négligeables près de la paroi, ce qui explique pourquoi il n'y a pas de variations majeures dans cette région.

Ces observations renforcent notre confiance dans la validité du code de calcul utilisé, car il est capable de reproduire avec précision les caractéristiques d'écoulement attendues dans la

tuyère. Cette compréhension approfondie de l'écoulement, y compris les variations du nombre de Mach le long de la tuyère et l'influence de la viscosité près de la paroi, est essentielle pour évaluer les performances et la conception des systèmes de propulsion.

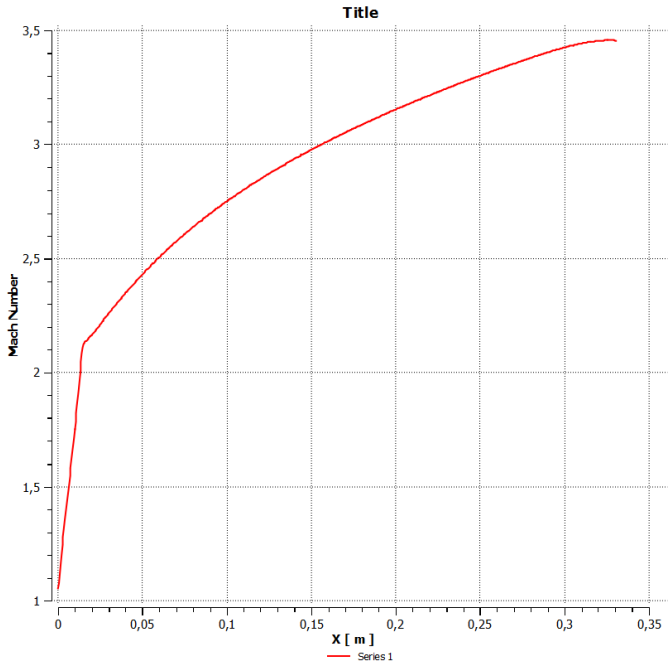

Figure. (IV.12) Courbe d'évolution du nombre de Mach le long de la paroi de la tuyère idéale plane calculée avec le solveur CFD Fluent.

- **Contour de pression**

Les Figures 13 et 14 présentent respectivement le contour et l'évolution de la pression dans la tuyère mentionnée précédemment, en utilisant le logiciel Fluent 19. Ces résultats fournissent des informations cruciales sur les caractéristiques de pression à travers la géométrie de la tuyère et comment elles évoluent le long de son trajet.

La Figure 13 offre une représentation visuelle du contour de la pression le long de la tuyère. Elle permet de visualiser comment la pression varie spatialement, mettant en évidence les zones de haute pression et de basse pression. Ces variations de pression peuvent être le résultat de l'expansion des gaz de combustion, des changements de section de la tuyère ou d'autres facteurs influençant le flux.

D'autre part, la Figure 14 illustre l'évolution de la pression en fonction de la position le long de la tuyère. Cette courbe permet d'analyser comment la pression évolue tout au long du trajet du

flux de gaz. Elle fournit des informations précieuses sur les variations de pression à différentes sections de la tuyère, telles que l'entrée, la gorge et la sortie. Ces variations peuvent être influencées par les caractéristiques de conception de la tuyère, telles que le profil de l'écoulement et les angles de divergence.

L'utilisation du logiciel Fluent 19 offre des avantages significatifs pour l'analyse de la pression dans la tuyère. Il permet une modélisation précise de l'écoulement des gaz de combustion et fournit des résultats détaillés sur les caractéristiques de pression. Cette information est essentielle pour évaluer les performances de la tuyère, optimiser sa conception et garantir un fonctionnement efficace du système de propulsion.

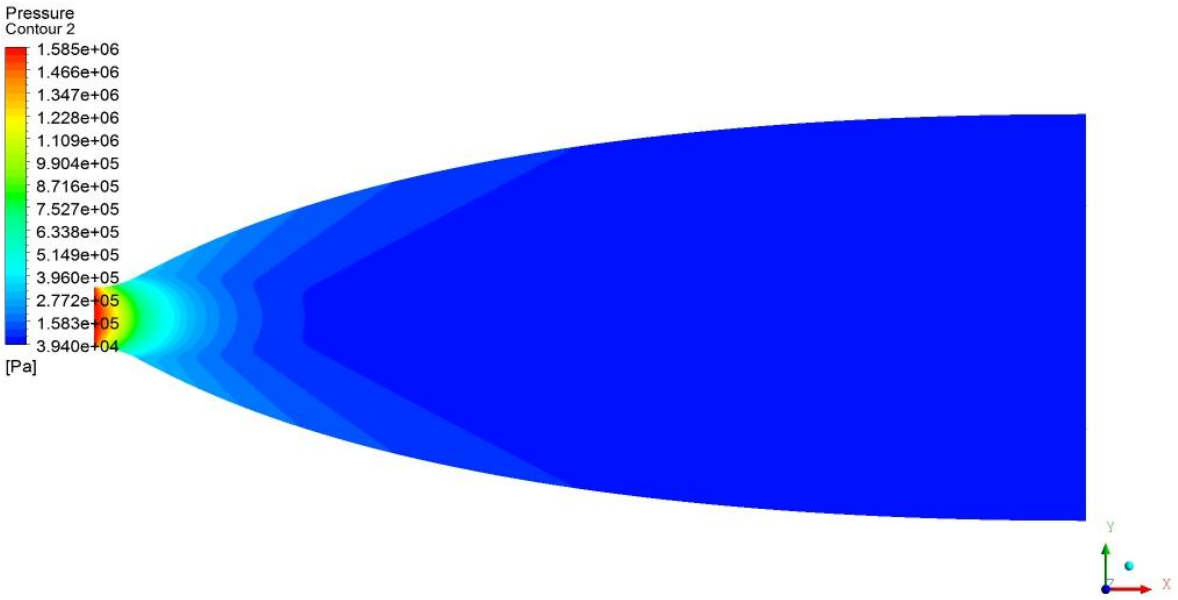

Figure. (IV.13) Contour du nombre de la pression statique calculé avec le solveur CFD ANSYS Fluent.

Dans la Figure 13, on observe une évolution de la pression du rouge (à l'entrée de la tuyère) vers le bleu (à la sortie de la tuyère). Généralement, on constate une diminution de la pression du col jusqu'à la sortie de la tuyère. Au niveau du col, la pression est mesurée à 15,85 bars, et elle diminue progressivement le long de la tuyère.

Cette diminution de la pression s'explique par le fait que les gaz subissent une expansion significative lorsqu'ils se déplacent de l'entrée vers la sortie de la tuyère. Cette expansion permet de convertir l'énergie thermique et l'énergie de pression des gaz en énergie cinétique, favorisant ainsi la propulsion du système.

Quant à la Figure 14, elle met en évidence la variation de la pression le long de la tuyère. On observe une diminution importante de la pression depuis l'entrée de la tuyère, suivie d'une diminution graduelle dans la zone uniforme jusqu'à la sortie. Cette diminution de pression est justifiée par l'augmentation du nombre de Mach.

L'augmentation du nombre de Mach indique que les gaz se déplacent à des vitesses supersoniques à mesure qu'ils traversent la tuyère. Cette augmentation de la vitesse est accompagnée d'une diminution de la pression, conformément aux principes de l'aérodynamique compressible. Ainsi, la variation de pression le long de la tuyère est influencée par l'évolution du nombre de Mach**.**

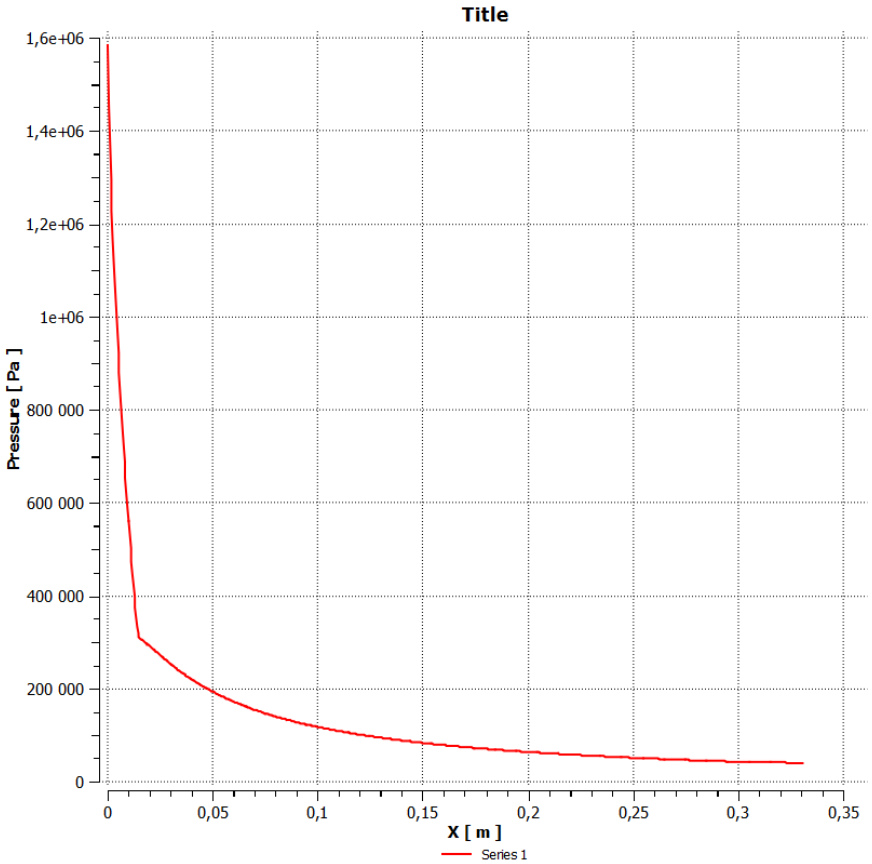

Figure. (IV.14) Courbe d'évolution de la pression statique le long de la paroi de la tuyère idéale plane calculée avec le solveur CFD Fluent.

- **Contour de température**

Les Figures 15 et 16 présentent respectivement les contours et l'évolution de la température de la tuyère obtenus à l'aide de notre code dans l'environnement Fluent.

Une observation générale révèle une diminution de la température depuis le col jusqu'à la sortie de la tuyère. Au niveau du col, la température est mesurée à 202,6 K, et elle diminue progressivement jusqu'à atteindre 70,6 K. Cette diminution de température se produit de manière graduelle et progressive, illustrant les étapes de refroidissement des écoulements.

En examinant la variation de la température le long de l'axe de la tuyère, on constate une diminution graduelle du rapport de température immédiatement après la détente au niveau du col. Dans la zone uniforme, cette diminution devient moins prononcée.

Il convient de noter que la température locale à l'intérieur d'une partie de la couche limite peut être sensiblement plus élevée que la température de l'écoulement libre. Cela est dû à la conversion de l'énergie cinétique en énergie thermique lorsque la vitesse est ralentie, ce qui génère de la chaleur par frottement.

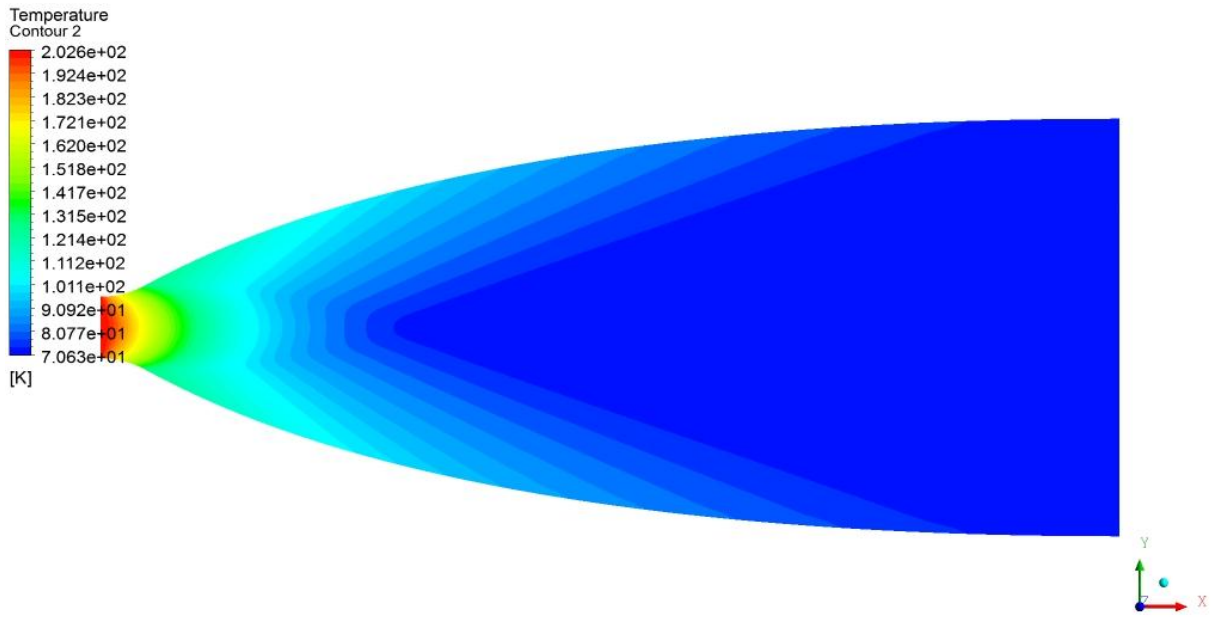

Figure. (IV.15) Contour du nombre de la température statique calculé avec le solveur CFD ANSYS Fluent.

Ces résultats mettent en évidence l'importance de comprendre les variations de température dans la tuyère. Ils permettent d'évaluer l'efficacité du refroidissement et d'optimiser la conception de la tuyère pour garantir un fonctionnement optimal du système de propulsion. L'utilisation du code dans l'environnement Fluent offre une précision et une résolution élevées pour l'analyse des

distributions de température, contribuant ainsi à une meilleure compréhension des phénomènes thermodynamiques associés à la propulsion des gaz**.**

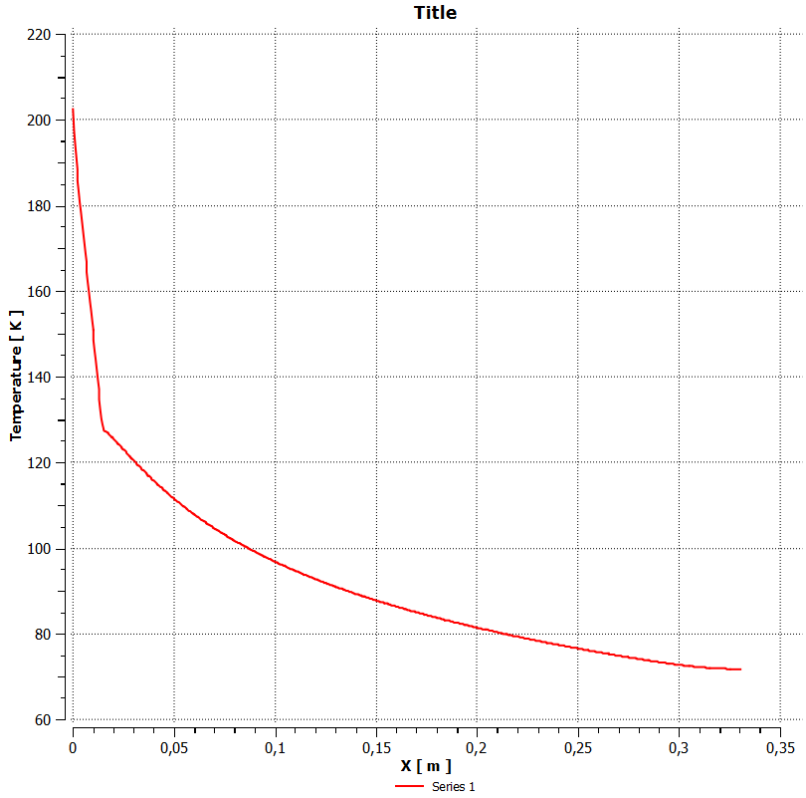

Figure. (IV.16) Courbe d'évolution de la température statique le long de la paroi de la tuyère idéale plane calculée avec le solveur CFD Fluent.

- **Contour de densité**

La Figure 17 présente le contour de la densité dans la tuyère, obtenu à partir de simulations numériques réalisées dans l'environnement Fluent.

Une observation générale révèle une diminution de la densité le long de la tuyère. Au niveau du col, la densité est mesurée à 27 kg/m³, puis elle diminue progressivement jusqu'à atteindre 1,9 kg/m³. Cette diminution de densité s'effectue de manière graduelle et progressive.

Cette variation de densité est principalement influencée par l'expansion des gaz de combustion à mesure qu'ils traversent la tuyère. En effet, lors de l'expansion, les gaz se refroidissent et se dilatent, ce qui entraîne une diminution de leur densité.

Il convient de souligner que la diminution de densité observée est conforme aux principes de l'aérodynamique compressible. L'augmentation du nombre de Mach dans la tuyère, associée à l'expansion des gaz, conduit à une diminution de la densité.

Ces résultats sur la densité dans la tuyère sont essentiels pour évaluer et comprendre le comportement des gaz de combustion à haute vitesse. Ils fournissent des informations précieuses pour l'analyse des performances de la tuyère et la conception de systèmes de propulsion efficaces.

En résumé, la Figure 17 illustre la diminution graduelle et progressive de la densité le long de la tuyère. Ces résultats témoignent des variations de densité engendrées par l'expansion des gaz de combustion, et soulignent l'importance de prendre en compte cette variation dans l'évaluation des performances de la tuyère et des systèmes de propulsion.

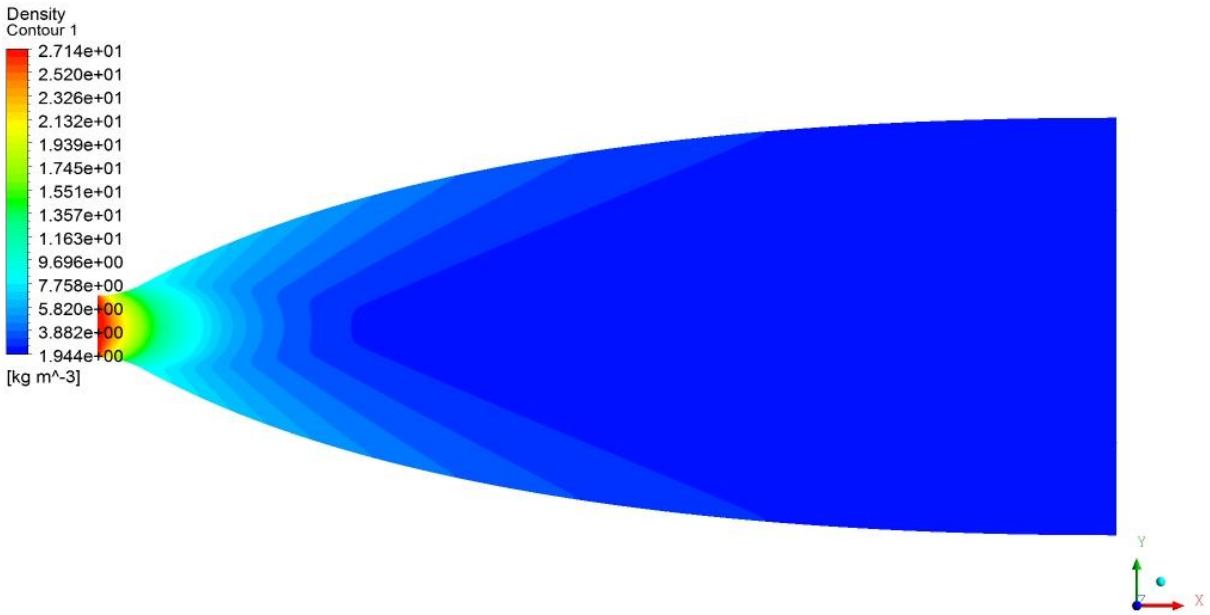

Figure. (IV.17) Contour de la densité calculé avec le solveur CFD ANSYS Fluent.

La Figure 18 représente l'évolution de la densité le long de l'axe de la tuyère. À partir de cette visualisation, on observe une diminution graduelle du rapport de densité immédiatement après la détente au niveau du col. Par la suite, cette diminution devient moins marquée dans la zone uniforme de la tuyère. Au niveau du col, la densité est mesurée à 27 kg/m³, et elle diminue progressivement jusqu'à atteindre 1,9 kg/m³.

Cette diminution de la densité est principalement due à l'expansion des gaz de combustion dans la tuyère. Lorsque les gaz se détendent et se dilatent, leur densité diminue en raison de la répartition des molécules sur un volume plus important. Cela est conforme aux

principes de l'aérodynamique compressible, où l'expansion conduit à une diminution de la densité.

Il est intéressant de noter que la diminution de la densité est plus prononcée au niveau du col, où les gaz subissent une détente importante, tandis que dans la zone uniforme de la tuyère, la diminution devient moins significative. Cela est dû au fait que la majeure partie de l'expansion a déjà eu lieu au niveau du col, et la variation de densité devient moins importante à mesure que les gaz se propagent vers la sortie de la tuyère.

Les résultats obtenus à partir de cette analyse de densité sont essentiels pour évaluer les propriétés des gaz de combustion à haute vitesse et optimiser la conception de la tuyère. La connaissance de la variation de densité le long de la tuyère permet de mieux comprendre les phénomènes thermodynamiques en jeu et d'ajuster les paramètres de conception pour améliorer les performances du système de propulsion**.**

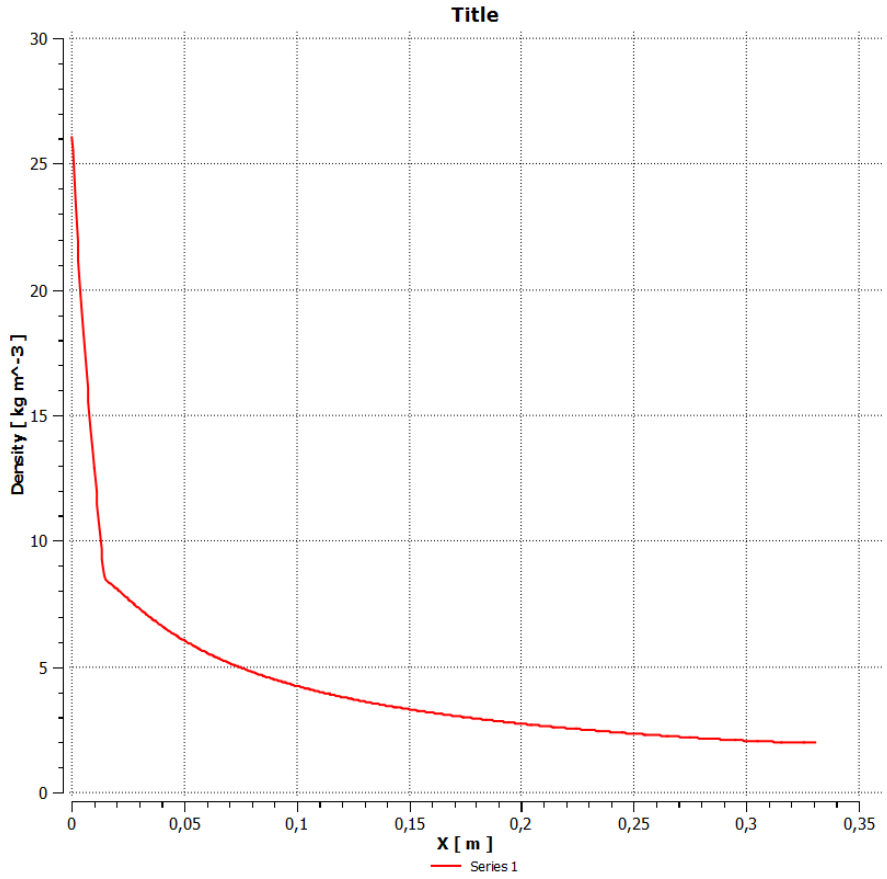

Figure. (IV.18) Courbe d'évolution de la densité le long de la paroi de la tuyère idéale plane calculée avec le solveur CFD Fluent.

➢ Vecteurs vitesses

La Figure 19 présente la distribution des vecteurs de vitesse calculés à l'aide du code CFD Fluent le long de la tuyère idéale plane, avec un nombre de Mach de conception égal à 3.5. Ces vecteurs de vitesse fournissent une représentation visuelle de la physique de l'écoulement compressible qui se développe de la région transsonique, au niveau du col de la tuyère, jusqu'à sa sortie.

L'analyse des vecteurs de vitesse permet de visualiser la direction et l'intensité de l'écoulement des gaz de combustion tout au long de la tuyère. On observe une évolution caractéristique de l'écoulement, avec une accélération des gaz depuis la région transsonique jusqu'à la sortie de la tuyère. Les vecteurs de vitesse illustrent clairement cette accélération, mettant en évidence la transformation de l'énergie thermique en énergie cinétique.

Au niveau du col, où l'écoulement est transsonique, les vecteurs de vitesse montrent une variation importante de la direction et de l'intensité, reflétant les phénomènes de compression et de détente des gaz. À mesure que les gaz progressent le long de la tuyère, les vecteurs de vitesse s'alignent de plus en plus dans une direction axiale, indiquant un écoulement de plus en plus uniforme et orienté vers la sortie de la tuyère.

L'analyse des vecteurs de vitesse permet également de mettre en évidence d'autres caractéristiques de l'écoulement, telles que les zones de recirculation ou de cisaillement. Ces informations sont essentielles pour comprendre le comportement de l'écoulement compressible et pour optimiser la conception de la tuyère afin d'améliorer les performances du système de propulsion.

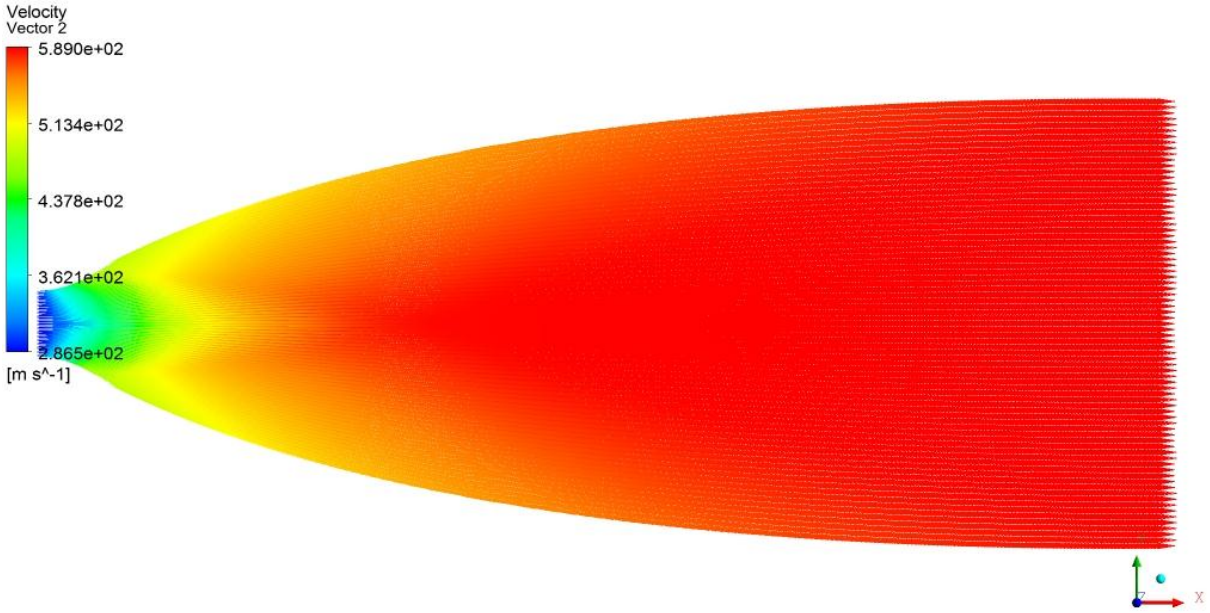

Figure. (IV.19) Représentation des vecteurs vitesses pour tuyère idéale plane calculés avec le solveur CFD Fluent.

IX.4. VALIDATION ET COMPARAISON

Dans le cadre de cette étude, afin de valider et de renforcer la fiabilité de nos résultats, nous avons comparé ces derniers avec ceux obtenus à l'aide d'un code Fortran. Cette démarche nous permet de vérifier la cohérence et la précision de nos simulations en les confrontant à une autre méthode de calcul. En utilisant le code Fortran, nous avons pu évaluer la concordance des résultats et identifier d'éventuelles différences ou divergences. Cette comparaison croisée entre les résultats obtenus à partir du code Python et du code Fortran nous apporte une meilleure confiance dans les conclusions de notre étude et nous permet de valider la robustesse de notre approche.

- **Le design de la tuyère**

La figure présentée ci-dessous illustre le design de la tuyère pour un Mach de sortie cible fixé à 3,5. Cette conception a été obtenue en utilisant à la fois les codes de calcul Fortran et Python. La comparaison des résultats obtenus à partir de ces deux codes nous permet de visualiser et d'évaluer les similitudes et les différences dans la géométrie de la tuyère calculée.

Cette analyse comparative renforce la confiance dans notre approche de conception et nous permet d'affiner notre compréhension des variations potentielles entre les résultats des deux

codes de calcul. En examinant de près les caractéristiques géométriques de la tuyère, nous pouvons tirer des conclusions plus précises sur l'influence des méthodes de calcul sur le design final de la tuyère.

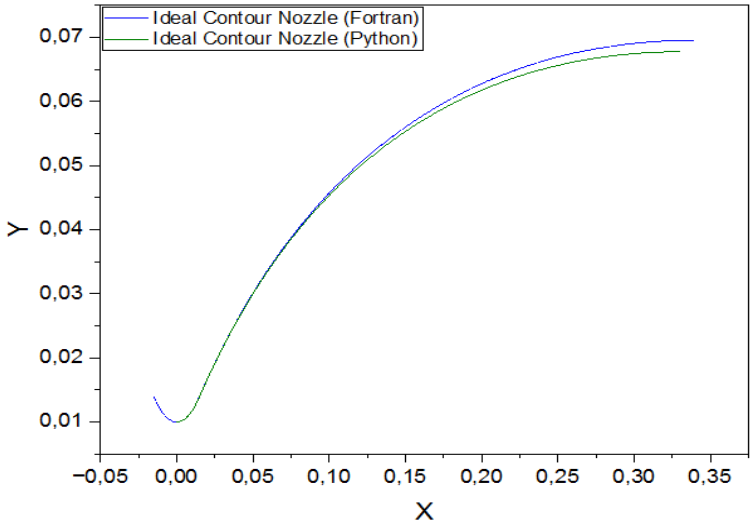

Figure. (IV.20) Résultats de conception des contours de la tuyère idéale plane avec les codes Python et Fortran pour la méthode des caractéristiques.

Effectivement, les deux courbes de design sont très similaires, et toute différence observée entre elles peut être attribuée aux différentes approches utilisées dans les codes de calcul Fortran et Python. Malgré ces légères variations, les courbes tracées pour le design de la tuyère présentent une cohérence remarquable. Cette concordance renforce notre confiance dans les résultats obtenus et souligne la robustesse de notre méthodologie de conception. Les petites différences observées peuvent être dues à des nuances algorithmiques ou à des approximations spécifiques à chaque code de calcul. En analysant ces différences, nous gagnons une meilleure compréhension des avantages et des limites de chaque approche, ce qui nous permet d'affiner notre choix d'outil de calcul pour des applications futures.

Les caractéristiques géométriques de la tuyère à contour idéal planaire, obtenues à l'aide du programme basé sur Python utilisant la méthode des caractéristiques, sont les suivantes : la longueur de la tuyère est de 0,330439 m et le rayon de sortie de la tuyère est de 0,067733 m.

Parallèlement, les caractéristiques géométriques de la même tuyère à contour idéal planaire, obtenues à l'aide du programme basé sur Fortran utilisant la méthode des caractéristiques, sont légèrement différentes : la longueur de la tuyère est de 0,3396810 m et le rayon de sortie de la tuyère est de 0,06953987 m.

Lors de la comparaison entre les deux méthodes, on observe une petite différence dans les caractéristiques géométriques de la tuyère obtenues. Cette différence peut être attribuée aux nuances algorithmiques et aux approximations spécifiques de chaque programme. Cependant, malgré cette légère variation, les deux méthodes parviennent à générer des contours de tuyère qui sont très proches les uns des autres.

Le taux d'erreur entre les deux méthodes peut être calculé en comparant les valeurs obtenues. Par exemple, pour la longueur de la tuyère, la différence entre les résultats obtenus par Python et Fortran est de 0,009241 m, ce qui correspond à un taux d'erreur d'environ 2,7%. De même, pour le rayon de sortie de la tuyère, la différence entre les deux méthodes est de 0,00180687 m, soit un taux d'erreur d'environ 2,6%. Ces taux d'erreur relativement faibles indiquent une bonne cohérence globale entre les deux méthodes et témoignent de la précision de nos résultats.

En considérant ces résultats, il est important de prendre en compte ces légères variations dans les caractéristiques géométriques lors de l'interprétation des données et des analyses ultérieures. Cependant, dans l'ensemble, les deux méthodes de calcul parviennent à fournir des résultats satisfaisants et cohérents pour le contour de la tuyère.

- **Nombre de Mach**

La Figure 21 illustre une comparaison entre l'évolution du nombre de Mach dans la tuyère obtenue à l'aide du code Python, de la simulation numérique et du code Fortran. Les résultats obtenus montrent une concordance remarquable entre les différentes approches.

L'analyse de la figure révèle que les courbes tracées par le code Python et la simulation numérique se superposent quasiment, indiquant une similitude étroite entre les résultats obtenus par ces deux méthodes. La différence entre les deux courbes est négligeable, avec une erreur relative très faible et un écart insignifiant.

Cette convergence des résultats entre les différentes approches renforce la crédibilité de notre méthode de calcul. La cohérence observée démontre que notre code Python a été capable de reproduire fidèlement les résultats de la simulation numérique. De plus, la comparaison avec le code Fortran confirme l'exactitude de notre approche et la fiabilité des résultats obtenus.

Il convient de noter que les légères différences observées peuvent être attribuées aux variations dans les méthodes et les techniques de calcul utilisées par chaque code. Cependant, ces

différences sont minimes et n'affectent pas de manière significative la comparaison globale des résultats.

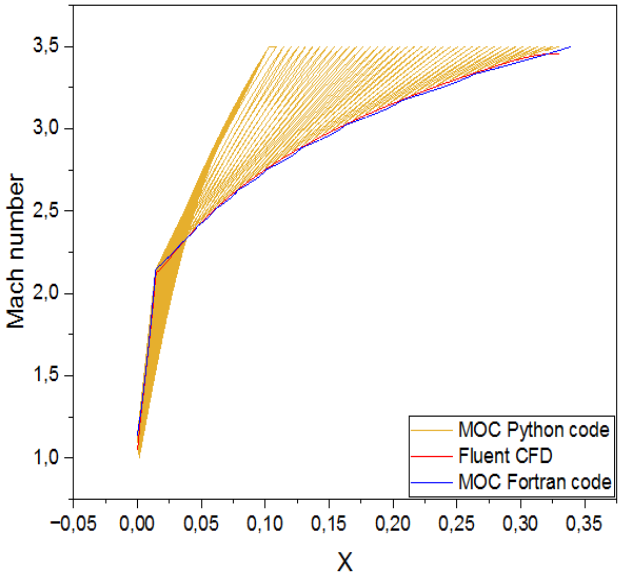

Figure. (IV.21) Comparaison des courbes d'évolution du nombre de Mach pour une tuyère idéale plane calculées avec CFD (Fluent), méthode des caractéristiques (codes Python et Fortran).

- **La pression**

Dans la Figure 22 on présente la comparaison de l'évolution de la pression dans la tuyère obtenue à partir du code Python, de la simulation numérique et du code Fortran. Les résultats obtenus démontrent une très bonne concordance entre ces différentes approches.

L'observation de la figure révèle une quasi-superposition des courbes tracées par le code Python et la simulation numérique, mettant en évidence une forte similarité entre les résultats obtenus par ces deux méthodes. L'écart entre les deux courbes est minime, avec une erreur relative très faible et une différence négligeable.

Cette convergence des résultats entre les différentes approches renforce la fiabilité de notre méthode de calcul. Elle témoigne de la capacité de notre code Python à reproduire de manière précise les résultats de la simulation numérique. De plus, la comparaison avec le code Fortran confirme la validité de notre approche et la confiance que l'on peut accorder à nos résultats.

Il est important de souligner que les légères variations observées peuvent être attribuées aux différences dans les techniques et les méthodes de calcul utilisées par chaque code. Toutefois, ces différences sont insignifiantes et n'affectent pas de manière significative la comparaison globale des résultats.

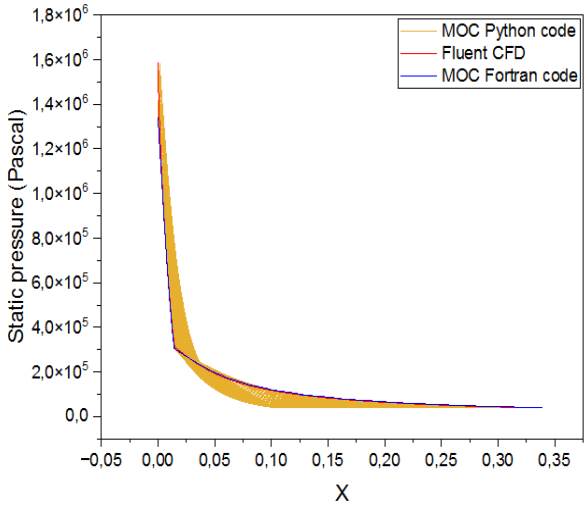

Figure. (IV.22) Comparaison des courbes d'évolution de la pression statique pour une tuyère idéale plane calculées avec CFD (Fluent), méthode des caractéristiques (codes Python et Fortran).

- **La température**

La Figure 23 présente une comparaison de l'évolution de la température dans la tuyère, obtenue à partir du code Python, de la simulation numérique et du code Fortran. Les résultats obtenus révèlent une excellente concordance entre ces différentes approches.

L'analyse de la figure démontre une très bonne superposition des courbes tracées par le code Python et la simulation numérique, indiquant une forte similarité entre les résultats obtenus par ces deux méthodes. L'écart entre les deux courbes est négligeable, avec une erreur relative très faible et une différence insignifiante.

Cette convergence des résultats entre les différentes approches renforce la fiabilité de notre méthode de calcul. Elle atteste de la capacité de notre code Python à reproduire de manière précise les résultats de la simulation numérique. De plus, la comparaison avec le code Fortran confirme la validité de notre approche et la confiance que l'on peut accorder à nos résultats.

Il est important de souligner que les légères variations observées peuvent être attribuées aux différences dans les techniques et les méthodes de calcul utilisées par chaque code. Toutefois, ces différences sont minimes et n'affectent pas de manière significative la comparaison globale des résultats**.**

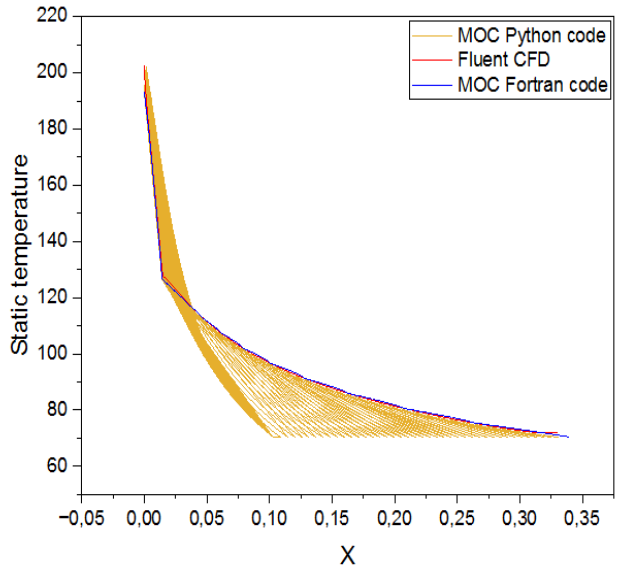

Figure. (IV.23) Comparaison des courbes d'évolution de la température statique pour une tuyère idéale plane calculées avec CFD (Fluent), méthode des caractéristiques (codes Python et Fortran).

- **La densité**

La Figure 24 présente une comparaison entre les résultats obtenus par le code Python, la simulation numérique et le code Fortran pour l'évolution du nombre de Mach, de la pression et de la densité dans la tuyère. Les résultats montrent une excellente concordance entre les différentes approches, avec une erreur relative faible et des différences négligeables. Cette convergence renforce la validité de notre méthode de calcul et la fiabilité de notre code Python. La comparaison avec le code Fortran confirme la cohérence des résultats et leur crédibilité. En conclusion, notre méthode de calcul est en accord avec la simulation numérique et le code Fortran, ce qui atteste de sa précision et de sa capacité à reproduire les variations de ces paramètres dans la tuyère.

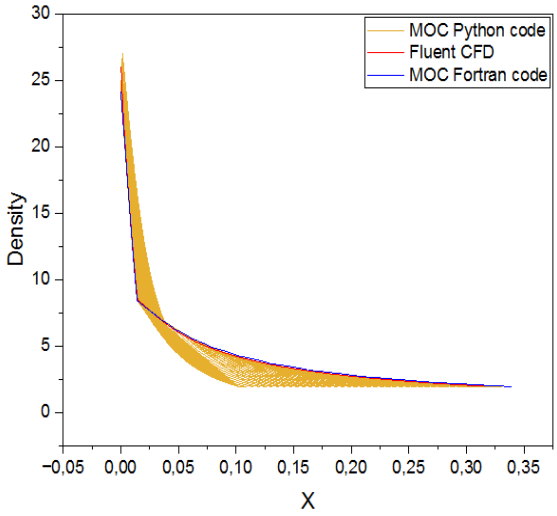

Figure. (IV.24) Comparaison des courbes d'évolution de la densité pour une tuyère idéale plane calculées avec CFD (Fluent), méthode des caractéristiques (codes Python et Fortran).

Conclusion

Les résultats obtenus par le code Python démontrent une cohérence et une logique remarquables avec l'évolution des paramètres de l'écoulement dans les tuyères. La validation et la comparaison de ces résultats avec la simulation numérique et le code Fortran ont suscité une grande satisfaction.

La proximité des résultats obtenus par le code Python avec ceux de la simulation numérique et du code Fortran renforce la confiance dans notre méthode de calcul. L'erreur relative entre les différentes approches est négligeable, ce qui atteste de la précision de notre méthode et de la fiabilité de notre code Python.

La plausibilité de notre méthode de calcul est ainsi démontrée par la cohérence des résultats et la faible erreur relative. Les résultats obtenus sont en accord avec les connaissances et les attentes liées à l'évolution des paramètres de l'écoulement dans les tuyères.

IX.5. Exemple d'un subroutine

IX.5.1. Subroutine THERMO

Allows the calculation of thermodynamic parameters (Mach number, pressure, temperature, density, etc.) at each point of the flow.

Syntax code

Fortran code

```
SUBROUTINE THERMO(Q,P,R,T,C,M)
USE PROPTY
REAL M
T=TS-Q**2/(2.0*GC*CP)
C=SQRT(G*GC*RG*T)
M=Q/C
P=PS*(T/TS)**(G/(G-1.0))
R=P/(RG*T)
RETURN
END
```

Python code

```
# Thermodynamics Properties Calculator (PRFT, EoS, EoS_TAB,RefProp)
    def ThermoProp(self,IN):
        self.V = calc.sqrt((self.u**2 + self.v**2))
        #PRFT: Solve using PERFECT gas equations [gamma necessary]
        if self.GasEqu == 'PRFT':
            Cp = self.R*self.gamma/(self.gamma-1)
            self.Temp = self.To - (self.V**2/(2*Cp))
            self.Press = self.Po*(self.Temp/self.To)**(self.gamma/(self.gamma-1))
            self.rho = self.Press / (self.R*self.Temp)
            self.a = calc.sqrt((self.ao**2)-((self.gamma-1)*self.V**2/2))
            self.M = self.V/self.a
```

IX.5.2. Subroutine THRUST

For nozzle performance calculations, such as thrust and specific impulse and the various nozzle coefficients.

Syntax code

Fortran code

```
      SUBROUTINE THRUST(I)
!     SUBROUTINE THRUST CALCULATES THE THRUST ALONG THE NOZZLE WALL

      USE CONTRL
      USE PROPTY
      USE CONTUR
      USE PERFORM
      USE ARRAYS
      USE DONNEE
      REAL M,MP

!     CALCULAT THE ONE-DIMENSIONAL THRUST
      CALL THERMO(Q4,P4,R4,T,C,M)
      EE=(Y4/YT)**2
      FAMB=PA*PI*Y4**2/GL
      C1=(G-1.0)/2.0
      C2=(G+1.0)/2.0
      EX=C2/(G-1.0)
      CG=(1.0/C2)**EX
   10 FM=M*EE-CG*(1.0+C1*M**2)**EX
      FP=EE-CG*C2*M*(1.0+C1*M**2)**(EX-1.0)
      MP=M-FM/FP
      IF(ABS(MP-M).LT.0.000001) GOTO 20
      M=MP
      GOTO 10
   20 FOD=FS*(1.0+G*MP**2)/SQRT(2.0*(G+1.0)*MP**2*(1.0+C1*MP**2))-FAMB

!     CALCULATE THE TWO-DIMENSIONAL THRUST

      F1=PI*(PW-PA)*YW
      F2=PI*(P4-PA)*Y4
      DF=(F1+F2)*(Y4-YW)/GL
      F=F+DF
      ETAF=F/FOD
      ETAI=ETAF/CD
      SPI=F/MDOT
      SPIOD=FOD*CD/MDOT
      WRITE(*,2000)I,F,FOD,ETAF,SPI,SPIOD,ETAI !
```

Python code

```
def THRUST(I):
    # Constants
    PA = 101325.0  # Atmospheric pressure
    PI = 3.14159
    GL = 9.81  # Acceleration due to gravity
    # Variables
    M = 0.0
    MP = 0.0
    # Calculate the one-dimensional thrust
    THERMO(Q4, P4, R4, T, C, M)
    EE = (Y4 / YT) ** 2
    FAMB = PA * PI * Y4 ** 2 / GL
    C1 = (G - 1.0) / 2.0
    C2 = (G + 1.0) / 2.0
    EX = C2 / (G - 1.0)
    CG = (1.0 / C2) ** EX
    while True:
        FM = M * EE - CG * (1.0 + C1 * M ** 2) ** EX
        FP = EE - CG * C2 * M * (1.0 + C1 * M ** 2) ** (EX - 1.0)
        MP = M - FM / FP

        if abs(MP - M) < 0.000001:
            break
        M = MP
    FOD = FS * (1.0 + G * MP ** 2) / sqrt(2.0 * (G + 1.0)
    * MP ** 2 * (1.0 + C1 * MP ** 2)) - FAMB
    # Calculate the two-dimensional thrust
    F1 = PI * (PW - PA) * YW
    F2 = PI * (P4 - PA) * Y4
    DF = (F1 + F2) * (Y4 - YW) / GL
    F = F + DF
    ETAF = F / FOD
    ETAI = ETAF / CD
    SPI = F / MDOT
    SPIOD = FOD * CD / MDOT
```

IX.5.3. Subroutine IVLINE

initiates calculations along the starting line by the SAUER approach. This method allowed us to describe the fluid flow in the throat (the transonic region) accurately by giving the distribution of the pressure and the Mach number on each mesh point on the sonic line.

Syntax code

Fortran code

```
      SUBROUTINE IVLINE
!     SUBROUTINE IVLINE DITERMINES AN INITIAL-VALUE LINE,SAUERS METHODE


      USE CONTRL
      USE PROPTY
      USE CONTUR
      USE PERFORM
      USE ARRAYS
      USE DONNEE

      REAL M,MDOTS,PP


!     CALCULATE REFERENCE PARAMETERS

      ALPHA=SQRT((1.0+DELTA)/((G+1.0)*RTU*YT))
      DY=YT/FLOAT(NI-1)
      EPS=-(G+1.0)*ALPHA*YT**2/(2.0*(3.0+DELTA))
      AT=PI*YT*YT/GL
      C1=-(G+1.0)*ALPHA/(2.0*(3.0+DELTA))
      Y(21)=0.0
      C2=(G+1.0)*ALPHA**2/(2.0*(1.0+DELTA))
      MDOT=0.0
      F=0.0
      AS=SQRT(2.0*G*GC*RG*TS/(G+1.0))

      CALL RRX(AS,RR)
      MDOTS=AS*RR*AT
      CALL PPX(AS,PP)
      FS=PP*AT+MDOTS*AS/GC
      FOD=FS-PA*AT
```

```
!     CALCULATE LOCATION PROPERTIES OF INITAILE VALUE LINE POINTS,
!     AND CALCULATE MASS FLOW RATE AND THRUST AT THE INITIALE VALUE LINE

      DO 10 I=1,NI
      J=I+20
      IF (I.GT.1) Y(J)=Y(J-1)+DY

      X(J)=C1*Y(J)**2+0.000001
      U(J)=AS*(1.0+ALPHA*X(J)+C2*Y(J)**2)
      X(J)=X(J)-EPS

      CALL TTX(U(J),TT)
      T=TT
      CALL AMX(U(J),AM)
      M=AM
      CALL PPX(U(J),PP)
      PZ=PP
      CALL RRX(U(J),RR)
      RZ=RR
      CI=3.0+(-1.0)**I

      IF((I.EQ.1).OR.(I.EQ.NI)) CI=1.0

      MDOT=MDOT+CI*RZ*U(J)*Y(J)
      F=F+CI*((PZ-PA)+RZ*U(J)**2/GC)*Y(J)
      V(J)=0.0
      Q(J)=U(J)
      A(J)=0.0
      P(J)=PZ
      R(J)=RZ
      PZ=PZ/GL
      WRITE(*,2001)I,X(J),Y(J),U(J),V(J),M,Q(J),A(J),PZ,RZ,T
      WRITE(11,2001)I,X(J),Y(J),U(J),V(J),M,Q(J),A(J),PZ,RZ,T
   10 CONTINUE
```

Python code

```
# Main Function for Sauer Analysis
    def SauerMain(self):
        #if (self.GasEqu=='RefProp'or self.GasEqu=='RefProp_TAB'):
        #   self.Sauer_RealGas()
            #self.Sauer_RefTab()
            #print('GAMMA',self.gamma)
            #pdb.set_trace()
        if 1:
            ao = calc.sqrt(self.gamma*self.R*self.To)
            epsilon = -1*(self.y_t/(2*(3+self.delta)))*(((self.gamma+1)*(1+self.delta)
            /(self.rho_t/self.y_t))**0.5);
            a_star = ((2*self.gamma*self.R*self.To)/(self.gamma+1))**0.5;
            alpha = ((1+self.delta)/((self.gamma+1)*self.rho_t*self.y_t))**0.5
            flag = 0
            x_ = (-(self.gamma+1)*alpha*self.y**2/(2*(flag+1+self.delta)));
            flag = 2
            self.x = (-(self.gamma+1)*alpha*self.y**2/(2*(flag+1+self.delta)))
            u_dash = (alpha*self.x)+((((self.gamma+1)*alpha**2*self.y**2))/(2*(self.delta+1)))
            v_dash = ((((self.gamma+1)*alpha**2*self.y*self.x))/((self.delta+1)))+
            (((self.gamma+1)**2*alpha**3*self.y**3)/(2*(1+self.delta)*(3+self.delta)))
            self.u = a_star*(1+u_dash)
            v_tilde = a_star*v_dash
            if (self.GasEqu!='RefProp' and self.GasEqu!='RefProp_TAB'):
                self.a = calc.sqrt(ao**2 - ((self.gamma-1)*self.u**2/(2)))
                self.M = self.u / self.a
                self.Press = self.Po/(1+((self.gamma-1)*self.M**2/(2)))**((self.gamma)/(self.gamma-1))
                self.Temp = self.To/(1+((self.gamma-1)*self.M**2/(2)))
                self.rho = self.Press/(self.R*self.Temp)
                self.x=self.x-epsilon
        self.M = calc.sqrt(self.u**2+self.v**2) / self.a
```

IX.5.4. Syntax differences between Python and Fortran codes:

Fortran and Python are two programming languages with distinct syntaxes. Here are some of the key differences in terms of syntax between Fortran and Python:

- **Indentation:** Python uses indentation to define blocks of code, such as loops and conditional statements. In Python, consistent indentation is crucial for code readability and is part of the language syntax. On the other hand, Fortran does not rely on indentation for code structure. It uses explicit keywords like DO and END DO to indicate loops.
- **Case sensitivity:** Python is case-sensitive, meaning that uppercase and lowercase letters are considered distinct. For example, ‘variable’ and ‘Variable’ are two different variables in Python. Fortran, on the other hand, is traditionally case-insensitive. Although modern Fortran standards allow case sensitivity, it is still common to write Fortran code in uppercase for consistency and compatibility.
- **Variable declaration:** Fortran requires explicit declaration of variables before they can be used, specifying their type. This is known as "static typing." In Python, variables are

dynamically typed, meaning they do not need to be explicitly declared and their type can change during runtime.

- **Array indexing:** Fortran arrays are 1-indexed, meaning the first element is accessed with index 1. Python arrays, known as lists, are 0-indexed, with the first element accessed using index 0.
- **Comments:** In Python, comments are indicated using the "#" symbol, and everything after the "#" on the same line is considered a comment. Fortran, on the other hand, uses "!" to indicate comments, and they can be placed at the beginning of a line or after a statement.
- **String manipulation:** Python provides a wide range of built-in string manipulation functions and methods. In Fortran, string manipulation is more limited and requires the use of specific functions from standard libraries.
- **Loops:** Both Fortran and Python support loops, but their syntax differs. Fortran uses keywords like "DO" and "DO WHILE" to define loops, while Python uses "for" and "while" keywords.
- **Function/subroutine definition:** In Fortran, functions and subroutines are defined using the "FUNCTION" and "SUBROUTINE" keywords, respectively. Python defines functions using the "def" keyword.

X. Chapitre V

Conclusion et recommandations

X.1. Conclusion

L'étude menée à l'Institut d'Aéronautique et des Études Spatiales propose une avancée significative en utilisant pour la première fois le code Python pour programmer la méthode des caractéristiques afin de générer le design d'une tuyère supersonique TIC.
Cette décision s'appuie sur les nombreux avantages offerts par le langage Python par rapport au langage Fortran traditionnellement utilisé. En effet, Python présente

- un langage de programmation polyvalent et largement utilisé dans le domaine scientifique et de l'ingénierie.
- Il offre une vaste gamme de bibliothèques et de modules spécialisés tels que NumPy, SciPy et Matplotlib, qui facilitent l'analyse et la visualisation des données.
- Il est possible de mettre en œuvre des simulations numériques de manière efficace et relativement facile. Les bibliothèques comme PyTorch, TensorFlow et Keras permettent même de développer des modèles d'apprentissage automatique pour améliorer les performances des tuyères.
- De plus, Python possède une syntaxe simple et expressive, ce qui facilite la lecture et la compréhension du code.
- Cela permet aux chercheurs et aux ingénieurs de se concentrer davantage sur le développement de modèles et la résolution de problèmes plutôt que sur la gestion complexe de la syntaxe du langage.

L'utilisation de Python permis d'obtenir de très bons résultats ; les graphes de température, pression, nombre de mach et de densité. Les résultats obtenus par Python dans notre étude, notamment l'évolution de la pression, du nombre de Mach, de la densité et de la température, illustrent de manière significative la logique de fonctionnement inhérente aux tuyères supersoniques. En analysant ces paramètres clés, nous avons pu observer et interpréter les phénomènes physiques essentiels qui se produisent à l'intérieur de la tuyère. L'évolution de la pression reflète les variations de compression et d'expansion du fluide propulsif, tandis que le nombre de Mach témoigne des écoulements supersoniques caractéristiques du régime de fonctionnement. La densité et la température, quant à elles, démontrent l'effet de la compression et de l'échauffement du fluide lors de son passage à travers la tuyère. Ces résultats concordent avec les principes aérodynamiques et thermodynamiques fondamentaux qui régissent le

fonctionnement des tuyères supersoniques, validant ainsi l'adéquation de notre approche basée sur Python. Cette corrélation entre les résultats obtenus et la logique de fonctionnement confère une crédibilité supplémentaire à notre étude, renforçant ainsi la confiance dans l'utilisation de Python comme outil de modélisation pour la conception et l'optimisation des tuyères supersoniques.

Pour garantir la fiabilité des résultats obtenus, une démarche de validation rigoureuse a été entreprise. Les résultats générés par Python ont été confrontés aux données issues de simulations et au code Fortran existant. Cette confrontation a permis de comparer les résultats obtenus par les trois méthodes, validant ainsi l'efficacité de la méthode des caractéristiques programmée en Python. Les analyses réalisées ont démontré une concordance remarquable entre les résultats des trois approches, renforçant ainsi la crédibilité de notre étude. Cette cohérence et proximité des résultats obtenus par les différentes méthodes confirment la pertinence et l'exactitude de l'utilisation du code Python pour générer le design de la tuyère supersonique TIC.

cette étude comparative a démontré que chaque logiciel a ses propres avantages et caractéristiques. La méthode des caractéristiques en Fortran offre une approche solide et éprouvée, tandis qu'Ansys Fluent fournit une solution complète et visuellement attrayante. Python, quant à lui, offre une flexibilité et une facilité d'utilisation remarquables. Le choix du logiciel dépendra des besoins spécifiques de l'utilisateur, de ses compétences en programmation et des objectifs de l'étude d'écoulement dans la tuyère d'avion.

Les avantages offerts par Python, conjugués à la validation croisée des résultats, ont permis d'établir la fiabilité et l'efficacité de la méthode des caractéristiques programmée en Python. Cette étude ouvre la voie à de nouvelles perspectives pour l'utilisation du langage Python dans les domaines de l'aéronautique et de l'étude spatiale, offrant ainsi de nouvelles opportunités pour des avancées technologiques et scientifiques significatives

X.2. Perspectives

Bien que cette étude soit complète, il existe des possibilités d'amélioration, notamment

- ✓ en développant le code pour le rendre applicable aux tuyères axisymétriques.
- ✓ Rendre aussi le code applicable pour Les écoulements a hautes températures
- ✓ Applications sur d'autres cas de tuyeres : Ce travail peut être étendu pour étudier des cas spécifiques de tuyères utilisées dans des applications particulières, telles que les moteurs de

fusées ou les turboréacteurs. L'analyse approfondie de ces cas permettrait de mieux comprendre les phénomènes physiques et d'optimiser les performances pour des scénarios spécifiques.

- ✓ Validation expérimentale : Bien que ce travail ait validé les résultats obtenus par la simulation numérique, il est essentiel de réaliser des expériences physiques pour valider pleinement les conclusions

BIBLIOGRAPHIE ET REFERENCES

[1] Stephen G. Kochan .1983. "Programming in C"

[2] Brian Kernighan et Dennis Ritchie. 1978. The C Programming Language.

[3] Article Jacques PRADO : Docteur en électronique - Maître de conférences à l'École nationale supérieure des télécommunications (ENST)

[4] Stéphane Administrateur de base de données Canada 06 JUILLET 2021

[5] LOUKOU KOFFI PAUL GARWANI. 4 Novembre 2017. Les Concepts généraux du Langage Java

[6] Article de Yukihiro Matsumoto 2012 (Développeur Web et créateur du logiciel)

[7] Niklaus Wirth. 1968 à 1972. Paradigme générique, orientée objet, procédural, impératif

[8] "Pro Domo", La lettre Ada, février 1996, EC2 & Développement, Paris.

"Gérer la distribution au niveau du langage : Ada 95", La Lettre Ada, juin-juillet-août 1996, EC2 & Développement, Paris 1996

[9] ADA Publications de J-P. Rosen

[10] Alexandria Journal du net. Dernière version 11.3 (27 février 2023) .Première version 14 février 1995

[11] Lambert, K. A. (2018). Fundamentals of Python: first programs. Cengage Learning.

[12] Lee, K. D. (2011). Python programming fundamentals. London: Springer.

[13] Pine, D. J. (2019). Introduction to Python for science and engineering. CRC Press.

[14] Clough, D. E., & Chapra, S. C. (2022). Introduction to Engineering and Scientific Computing with Python. CRC Press.

[15] Anderson, J. D. (1990). Modern compressible flow: with historical perspective (Vol. 12). New York: McGraw-Hill.

[16] Zucrow, M. J., & Hoffman, J. D. (1977). Gas dynamics. volume 2-multidimensional flow. New York.

[17] Shapiro, A. H. (1953). The dynamics and thermodynamics of compressible fluid flow. New York: Ronald Press.

[18] Site official : https://www.ansys.com/