



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD PROFESIONAL TICOMÁN

**DESARROLLO DE CÓDIGOS PARA EL DISEÑO DE MALLAS PARA EL
ANÁLISIS DE PERFILES AERODINÁMICOS CON SUPERFICIES
HIPERSUSTENTADORAS TIPO FLAP**

TESIS
Que para obtener el grado de
INGENIERO EN AERONÁUTICA

PRESENTA
MARCO ANTONIO CARDOSO MORENO

DIRECTOR DE TESIS: M. EN C. RAFAEL MEDINA NOGUERÓN

Agradecimientos

Resumen

Indice

Indice de Figuras	IX
Indice de Tablas	XI
Nomenclatura	XIII
Introducción	XVII
Objetivo	XIX
Motivación	XXI
1. Estado del Arte	1
1.1. Historia de la dinámica de fluidos computacional	1
1.2. La dinámica de fluidos en la actualidad	2
1.3. La generación de mallas en la actualidad	2
1.4. Software disponible para DFC	3
2. Discretización Espacial	5
2.1. Método de Diferencias Finitas	5
2.1.1. Desarrollo del método	6
3. Mallas	11
3.1. Tipos de Mallas Estructuradas	13
3.1.1. Mallas tipo C	13
3.1.2. Mallas tipo O	14
3.1.3. Mallas tipo H	14
3.2. Mallas Adaptativas	16
3.3. Consideraciones preliminares para la generación de mallas	16
3.4. Técnicas de generación del mallado	16
3.5. Generación de mallas mediante métodos algebráicos	17
3.5.1. Interpolación unidireccional	17
3.5.2. Interpolación multidireccional	20
3.6. Generación de mallas mediante EDP	22
3.7. Calidad de las Mallas	35
3.7.1. Skewness	38

4. Análisis de Flujo Potencial	41
4.1. Condiciones de Frontera	43
4.1.1. Condiciones en las lejanías	43
4.1.2. Condiciones para perfiles aerodinámicos	44
4.2. Discretización para flujo subsonico estacionario	46
4.2.1. Condiciones de frontera	50
5. Desarrollo Práctico	51
5.1. Estructura de las clases	52
5.1.1. Clase Airfoil	52
5.1.2. Clase Mesh	54
5.2. Diagramas de flujo	58
5.2.1. Estructura global del programa	58
5.2.2. Funciones principales	62
6. Resultados	65
6.1. Alcance y Límites de Aplicación	65
6.1.1. Malla O. Frontera interna única	65
6.1.2. Malla O. Frontera interna múltiple	65
6.1.3. Malla C. Frontera interna única.	70
6.1.4. Malla C. Frontera interna múltiple	70
6.2. Calidad de la malla	74
6.3. Análisis de DFC	91
6.3.1. Flujo Potencial	91
6.3.2. SU2	100
7. Conclusiones	105
8. Recomendaciones y trabajos futuros	107
A. Códigos	109
A.1. Generación de Malla C con uso de librería numba	109
A.2. Generación de Malla O con uso de librería numba	119
A.3. Conversión de mallas a formato SU2	127
A.4. Archivo main. Ejemplo de ejecución	139
B. Métodos Numéricos Iterativos	145
B.1. Métodos iterativos: Jacobi, Gauss-Seidel, SOR	145
B.1.1. Método de Jacobi	145
B.1.2. Método de Gauss-Seidel	146
B.1.3. Método de Sobre-relajación SOR	146
C. Solución de Sistemas de Bloque Tridiagonal	149
Referencias	151

Indice de Figuras

2.1. Discretización por Diferencias Finitas	6
2.2. Aproximaciones mediante diferencias finitas	8
3.1. Dominios lógico y físico	12
3.2. Mallas estructuradas y no estructuradas	12
3.3. Tipología de mallas	13
3.4. Malla tipo C	14
3.5. Malla tipo O	15
3.6. Malla tipo O	15
3.7. Malla Interpolación Polinomial	19
3.8. Malla Interpolación Polinomial Acercamiento	19
3.9. Transformación de malla por P_η	21
3.10. Transformación de malla por $P_\xi P_\eta$	22
3.11. Efecto de atracción por función $P(\xi, \eta)$	25
3.12. Efecto de atracción por función $Q(\xi, \eta)$	25
3.13. Generación de malla quasi uniforme por ecuaciones parabólicas	32
3.14. Generación de malla por ecuaciones elípticas	33
3.15. Control de ortogonalidad	36
3.16. Cálculo de aspect ratio en ANSYS	37
3.17. Estructura celda cuadrilatera en Verdict	38
4.1. Condiciones de frontera de flujo potencial	44
4.2. Desarrollo de flujo estacionario sobre un perfil aerodinámico	45
4.3. Fluido en reposo relativo a un perfil alar	45
4.4. Flujo alrededor de perfil momentos después	46
4.5. Malla original y mallas escalonadas	47
5.1. Diagrama de clase airfoil y subclase NACA4	54
5.2. Diagrama de clase mesh y subclases mesh_c y mesh_o	57
5.3. Estructura global del programa	58
5.4. Diagrama de flujo para generación de mallas (preproceso)	59
5.5. Diagrama de flujo para análisis de flujo potencial(solver)	60
5.6. Diagrama de flujo para análisis de postproceso	61
5.7. Diagrama de flujo para generación de mallas mediante la ecuación de Laplace	62
5.8. Diagrama de flujo para generación de mallas mediante la ecuación de Poisson	63
5.9. Diagrama de flujo para generación de mallas mediante la ecuación de Poisson	64
6.1. Malla tipo O con perfil NACA 4415	66
6.2. Malla tipo O con perfil NACA 4415 con $\alpha = 50^\circ$ y $\alpha = -50^\circ$	67

6.3.	Malla tipo O con perfil y flap NACA 4415	68
6.4.	Malla tipo O con perfil y flap NACA 4415 a 50°	69
6.5.	Malla tipo O con perfil y flap NACA 4415 a -10°	69
6.6.	Malla tipo C con perfil NACA 4415	71
6.7.	Malla tipo C. Vista de los límites recomendados	71
6.8.	Malla tipo C con perfil y flap NACA 4415	72
6.9.	Malla tipo C con perfil y flap NACA 4415 a 20°	73
6.10.	Malla tipo C con perfil y flap NACA 4415 a -30°	73
6.11.	Aspect ratio para malla tipo O con perfil NACA 4415	75
6.12.	Aspect ratio para malla tipo O con perfil NACA 4415	76
6.13.	Skew para malla tipo O con perfil NACA 4415	77
6.14.	Skew para malla tipo O con perfil NACA 4415	78
6.15.	Aspect ratio para malla tipo O con perfil y flap NACA 4415	79
6.16.	Aspect ratio para malla tipo O con perfil y flap NACA 4415	80
6.17.	Skew para malla tipo O con perfil y flap NACA 4415	81
6.18.	Skew para malla tipo O con perfil y flap NACA 4415	82
6.19.	Aspect ratio para malla tipo C con perfil NACA 4415	83
6.20.	Aspect ratio para malla tipo C con perfil NACA 4415	84
6.21.	Skew para malla tipo C con perfil NACA 4415	85
6.22.	Skew para malla tipo C con perfil NACA 4415	86
6.23.	Aspect ratio para malla tipo C con perfil y flap NACA 4415	87
6.24.	Aspect ratio para malla tipo C con perfil y flap NACA 4415	88
6.25.	Skew para malla tipo C con perfil y flap NACA 4415	89
6.26.	Skew para malla tipo C con perfil y flap NACA 4415	90
6.27.	Malla generada para el análisis de flujo potencial. Perfil NACA 0012	92
6.28.	Líneas equipotenciales para $\alpha = 0^\circ$ del análisis de flujo potencial de perfil NACA 0012	92
6.29.	Campo de presiones alrededor de perfil NACA 0012	93
6.30.	Campo de presiones alrededor de perfil NACA 0012	94
6.31.	Líneas de corriente alrededor de perfil NACA 0012	94
6.32.	Líneas de corriente alrededor de perfil NACA 0012	95
6.33.	Coeficiente de presión a lo largo de la cuerda de perfil NACA 0012	95
6.34.	Gráfica de coeficiente de sustentación vs. ángulo de ataque para perfil NACA 0012	96
6.35.	Líneas equipotenciales para $\alpha = 0^\circ$ del análisis de flujo potencial de perfil NACA 2412	97
6.36.	Campo de presiones alrededor de perfil NACA 2412	97
6.37.	Campo de presiones alrededor de perfil NACA 2412	98
6.38.	Líneas de corriente alrededor de perfil NACA 2412	98
6.39.	Líneas de corriente alrededor de perfil NACA 2412	99
6.40.	Coeficiente de presión a lo largo de la cuerda de perfil NACA 2412	100
6.41.	Gráfica de coeficiente de sustentación vs. ángulo de ataque para perfil NACA 2412	100
6.42.	Resultados de simulaciones para malla C en SU2	102
6.43.	Resultados de simulaciones para malla tipo C en SU2	102
6.44.	Resultados de simulaciones para malla O en SU2	102
6.45.	Resultados de simulaciones para malla O en SU2	103
6.46.	Resultados de simulaciones para malla C con perfil y flap	103
6.47.	Resultados de simulaciones para malla O con perfil y flap	103
6.48.	Comparación del coeficiente de presión a $\alpha = 10^\circ$	104

Indice de Tablas

5.1. Métodos de la clase airfoil	53
5.2. Métodos de la subclase NACA4	53
5.3. Métodos de la clase mesh	55
5.4. Métodos de las clases mesh_c y mesh_o	56
6.1. Características de malla O con perfil NACA 4415	66
6.2. Características de malla O con perfil y flap NACA 4415	68
6.3. Características de malla C con perfil NACA 4415	70
6.4. Características de malla C con perfil y flap NACA 4415	72
6.5. Características de la corriente libre en el análisis de flujo potencial	91
6.6. Características del flujo para el análisis en SU2	101

Nomenclatura

DFC Dinámica de Fluidos Computacional

EDP Ecuaciones Diferenciales Parciales

i Indice i de malla estructurada

j Indice j de malla estructurada

k Indice k de malla estructurada

x Eje de las ordenadas en el dominio físico

y Eje de las abscisas en el dominio físico

ξ Eje de las ordenadas en el dominio computacional

η Eje de las abscisas en el dominio computacional

f Función de una propiedad del flujo

$\frac{\partial f}{\partial x}$ Derivada parcial de la propiedad del flujo f con respecto al eje x

$\frac{\partial^2 f}{\partial x^2}$ Segunda derivada parcial de la propiedad del flujo f con respecto al eje x

$\frac{\partial^3 f}{\partial x^3}$ Tercera derivada parcial de la propiedad del flujo f con respecto al eje x

$\frac{\partial^n f}{\partial x^n}$ N derivada parcial de la propiedad del flujo f con respecto al eje x

$\frac{\partial f}{\partial y}$ Derivada parcial de la propiedad del flujo f con respecto al eje y

$\frac{\partial^2 f}{\partial y^2}$ Segunda derivada parcial de la propiedad del flujo f con respecto al eje y

$\frac{\partial^3 f}{\partial y^3}$ Tercera derivada parcial de la propiedad del flujo f con respecto al eje y

$\frac{\partial^n f}{\partial y^n}$ N derivada parcial de la propiedad del flujo f con respecto al eje y

$\frac{\partial^2 f}{\partial x \partial y}$ Derivada parcial mixta de la propiedad del flujo f con respecto a los ejes x y y

$f_{i,j}$ Aproximación del valor de la propiedad f del flujo en un punto i,j de la malla

Δx Intervalo entre puntos en dirección del eje x

Δy Intervalo entre puntos en dirección del eje y

$O()$	Error de truncamiento de la serie de Taylor
$L(x)$	Polinomio de Lagrange en función de x
L	Base polinómica de Lagrange
$H(x)$	Polinomio de Hermite en función de x
x_0	Coordenada x de un punto en la frontera interna de la malla
y_0	Coordenada x de un punto en la frontera interna de la malla
x_1	Coordenada x de un punto en la frontera externa de la malla
y_1	Coordenada x de un punto en la frontera externa de la malla
r	Punto arbitrario de la malla expresado en coordenadas del dominio computacional ξ, η
P_ξ	Proyector del dominio computacional al dominio físico mapeando las coordenadas en el eje ξ
P_η	Proyector del dominio computacional al dominio físico mapeando las coordenadas en el eje η
ξ_{xx}	Segunda derivada parcial de ξ con respecto a x
ξ_{yy}	Segunda derivada parcial de ξ con respecto a y
η_{xx}	Segunda derivada parcial de η con respecto a x
η_{yy}	Segunda derivada parcial de η con respecto a y
$x_{\xi\xi}$	Segunda derivada parcial de x con respecto a ξ
$x_{\xi\eta}$	Derivada parcial mixta de x con respecto a ξ y η
$x_{\eta\eta}$	Segunda derivada parcial de x con respecto a η
$y_{\xi\xi}$	Segunda derivada parcial de y con respecto a ξ
$y_{\xi\eta}$	Derivada parcial mixta de y con respecto a ξ y η
$y_{\eta\eta}$	Segunda derivada parcial de y con respecto a η
P	Función de forzado en dirección del eje ξ para la solución de la ecuación de Poisson
Q	Función de forzado en dirección del eje η para la solución de la ecuación de Poisson
J	Jacobiano de la transformación
I	Inverso del Jacobiano de la transformación
X	Coordenada en el eje x del dominio computacional de un punto perteneciente a la malla
Y	Coordenada en el eje y del dominio computacional de un punto perteneciente a la malla

\vec{v}	Vector de velocidad del flujo en el dominio físico
$\vec{\nabla}$	Operador vectorial nabla
ϕ	Función de flujo potencial
H	Entalpía
H_0	Entalpía de estancamiento
ρ	Densidad
ρ_0	Densidad de estancamiento
γ	Relación de calores específicos de un gas
U	Componente de la velocidad en el dominio computacional en dirección del eje ξ
V	Componente de la velocidad en el dominio computacional en dirección del eje η
ϕ_x	Primer derivada parcial de la función de flujo potencial con respecto a x
ϕ_y	Primer derivada parcial de la función de flujo potencial con respecto a y
ϕ_ξ	Primer derivada parcial de la función de flujo potencial con respecto a ξ
ϕ_η	Primer derivada parcial de la función de flujo potencial con respecto a η
\vec{V}_∞	Vector de velocidad del flujo en corriente libre
M	Número de Mach
Γ	Circulación
α	Ángulo de ataque del perfil
δ	Ángulo de deflexión del flap

Introducción

En general, para analizar y diseñar sistemas en los cuales interviene el flujo de fluidos se cuenta con dos herramientas: la experimentación y los métodos analíticos. La experimentación implica la construcción de modelos que serán probados en túneles de viento u otras instalaciones. En el caso del cálculo, se puede realizar de manera analítica o mediante el uso de métodos numéricos, a esta última técnica se le da el nombre de dinámica de fluidos computacional (CFD, por sus siglas en inglés). El uso de la DFC (dinámica de fluidos computacional) se ha popularizado gracias al desarrollo de las capacidades computacionales, ya que las simulaciones presentan ciertas ventajas frente al enfoque experimental en términos de velocidad, seguridad y en la mayoría de casos, de costo. Gracias a esto la DFC es ampliamente usada en la actualidad en sectores de la industria como el aeroespacial, petrolero, energético, automotriz, entre otros. En el ámbito de la investigación científica también se tiene en la DFC una herramienta importante, pues permite analizar fenómenos complejos que pueden resultar difíciles de reproducir en experimentos.

Se realiza el presente trabajo con la finalidad tanto de generar en los estudiantes un interés por la dinámica de fluidos computacional, como de fomentar el desarrollo de este campo mediante la implementación de códigos. Por otro lado, se busca fomentar en el lector el uso y desarrollo de software libre.

En el capítulo 1 se realiza una breve reseña histórica, con el fin de entender los orígenes y etapas por las que ha atravesado la dinámica de fluidos computacional para así comprender la posición actual en la que se encuentra y las posibles direcciones que ésta pueda tomar. También en este primer capítulo se comenta el estado actual tanto de la DFC como tema general, además del estado actual de la generación de mallas como tema particular. Por último se hace mención a diferentes alternativas de software disponible y ampliamente reconocido, tanto programas de licencia privativa como de código abierto, para el análisis de flujos de interés y para la generación de mallas.

El capítulo 2 se enfoca en los conceptos de la discretización espacial de un espacio físico determinado, proceso necesario para poder realizar mallas que representen computacionalmente el espacio a analizar. Los métodos de discretización son aplicables también a las ecuaciones que escriben el comportamiento de los flujos, dicha discretización se hace tomando en cuenta la malla generada para el análisis. Se hace particular énfasis en el desarrollo del método de diferencias finitas, y se muestran las ecuaciones obtenidas mediante este método, las cuales sirven de aproximaciones para sustituir los términos que involucren ecuaciones diferenciales parciales.

Por su parte, el capítulo 3 centra su atención en las mallas. Se describe el concepto de malla, su razón de ser, así como la necesidad de utilizar un sistema arbitrario de coordenadas en ciertos problemas de interés práctico. También se hace referencia a la clasificación de las mallas, las diferentes

tipologías (formas) que pueden tener. De igual manera se mencionan algunas consideraciones a tomar en cuenta previo al desarrollo de la malla. Así mismo se elabora sobre las diferentes técnicas para la creación de mallas estructuradas, explicando las bases matemáticas y las ecuaciones que componen a las diversas técnicas. Por último, se cuenta con una sección dedicada la calidad de las mallas donde se contemplan algunas aspectos que son utilizados de manera amplia para la evaluación de la calidad.

En el capítulo 4 se desarrolla el análisis de flujo potencial y su adaptación para el análisis de flujos externos en los cuales se encuentra inmerso un perfil alar.

El capítulo 5 se centra en el desarrollo práctico del proyecto, es en éste donde se muestra la estructura y lógica del software desarrollado. Se presentan diagramas de clase para entender la estructura del programa, y diagramas de flujo para los métodos más representativos del mismo.

Por último, en el capítulo 6 se muestran los resultados obtenidos mediante la utilización de los códigos desarrollados. Se muestra en primera instancia una sección con los límites y alcance de las mallas generadas, continúa con resultados de calidad de malla para varias configuraciones de malla. Por último se llevaron a cabo análisis tanto de flujo potencial y de flujo turbulento y estacionario.

Objetivo

Diseñar una paquetería de software que permita la generación de mallas a través de diferentes métodos y algoritmos, que se ajusten a la geometría de un perfil aerodinámico con flaps y que resulten viables para el desarrollo de análisis de dinámica de fluidos computacional sobre dichas mallas. El desarrollo completo se realizará en lenguaje Python 3 y se publicará de acuerdo a la ideología del software libre, lo que permitirá que la comunidad sea participe en el desarrollo de nuevos módulos y métodos para los cuáles el presente trabajo pueda servir de cimiento.

El programa debe ser capaz de generar mallas mediante diferentes métodos, en este trabajo se presta principal atención a las mallas generadas por métodos algebráicos, así como por métodos basados en la solución de sistemas de ecuaciones diferenciales parciales (EDP), más específico, en EDP elípticas. Se debe generar la malla para cualquier superficie o forma deseada, en este trabajo se hace la generación de mallas alrededor de perfiles aerodinámicos, por lo que se debe desarrollar un módulo que genere la nube de puntos que describe un perfil NACA de la serie 4, pudiendo ser ajustable la densidad de puntos de la misma. También se da la opción de importar la nube de puntos de cualquier otro perfil, con la restricción de que la densidad de la malla a generar, depende de los datos de entrada que recibe el código.

Una vez generado el mallado del dominio, se creará un archivo que contenga toda la información del mismo, el cual podrá ser importado a algún SU2, software “*solver*”, para poder llevar a cabo el análisis de flujo deseado.

Los objetivos particulares son:

- Realizar un módulo que genere la nube de puntos de perfiles NACA de la serie de 4 dígitos. Así mismo, proporcionar la funcionalidad adecuada para que el usuario pueda importar la nube de puntos que describa cualquier otro perfil deseado.
- Realizar un módulo que modifique la nube de puntos característica del perfil, agregándole la información correspondiente al flap utilizado. Se consideran diferentes tipos de flaps.
- Desarrollar módulos para generar las mallas alrededor de la nube de puntos proporcionada. Se contemplan variadas tipologías, como lo son las mallas tipo “O” y las mallas tipo “C”, analizando sus ventajas y desventajas, así como una comparación entre las tipologías. Las mallas serán generadas mediante varios enfoques, como pueden ser la generación por ecuaciones algebráicas y por ecuaciones diferenciales parciales.
- Llevar a cabo diferentes análisis, ya sea en el software de código libre SU2 o mediante el desarrollo de un módulo que realice el análisis correspondiente mediante flujo potencial. Este

último punto con el propósito de probar la aplicabilidad y confiabilidad de las mallas y darles una aplicación de interés real.

Motivación

Existe en la actualidad, una vasta cantidad de software enfocado a la dinámica de fluidos computacional, tanto software privativo (o de licencia) como software libre. Para el caso del software privativo, el usuario paga por la licencia de uso del programa con lo cual puede usarlo, pero no tiene forma de saber la forma en la que el programa funciona ni la implementación del mismo. En cuanto al software libre, el usuario tiene la posibilidad de acceder al código fuente con el cual corre el programa, para analizarlo, comprenderlo e incluso, de ser necesario, modificarlo ya sea para mejorar una característica ya implementada, o bien, para agregar una nueva funcionalidad al programa. Tanto el uso de software privativo, como la poca implementación de códigos y documentación accesible en México, han conducido a un punto en el que la comunidad académica en el país se limite únicamente a usar los programas sin un entendimiento claro de su funcionamiento, excluyéndose a sí misma del desarrollo de software.

Se desarrolla entonces este trabajo, con la intención de proporcionar información sobre la implementación de códigos de generación de mallas, para que el lector pueda a su vez, continuar contribuyendo al desarrollo de software libre para la dinámica de fluidos computacional a nivel nacional.

Los códigos aquí presentados quedan abiertos para que cualquier persona, ya sea alumno, profesor, investigador o simplemente entusiasta de la dinámica de fluidos computacional, pueda revisarlos, utilizarlos en beneficio propio, modificarlos, e incluso construir software nuevo a partir de éste.

Capítulo 1

Estado del Arte

1.1. Historia de la dinámica de fluidos computacional

La DFC tiene sus orígenes en el desarrollo de dos métodos numéricos que son las herramientas base de esta disciplina, el método de diferencias finitas y el método de elementos finitos. En 1910, Richardson presentó en la “Royal Society of London” un artículo con una solución mediante el método de diferencias finitas para un análisis de la distribución de esfuerzos de una presa de mampostería. Por otro lado, el primer trabajo mediante el método de elementos finitos, fue publicado en 1956 en el Aeronautical Science Journal por Turner, Clough, Martin y Topp, el cual trata aplicaciones del método para el análisis de esfuerzos en aeronaves. [1]

Durante la segunda guerra mundial, así como en los años siguientes a la misma, el profesor John Von Neumann desarrolló un método para determinar la estabilidad numérica para la resolución de problemas dependientes del tiempo. Su trabajo fue publicado por O’brien en 1950. El método de Von Neumann es ampliamente aplicado hoy en día para determinar la estabilidad numérica. [2]

En la década de los años 60s se dan los primeros esfuerzos en el desarrollo de técnicas para la generación de mallas. [3]

Al inicio de la década de 1970 se da un auge en la DFC gracias al incremento de las capacidades de procesamiento de las computadoras, incluso hoy en día el avance de la DFC va de la mano con el desarrollo de las capacidades computacionales. [4] Es en ésta década cuando se desarrollan, gracias a un grupo del “Imperial College” de Londres, algoritmos para flujos incompresibles de baja velocidad, así como el algoritmo SIMPLE (Semi-Implicit Method for Pressure-Linked Equations), lo cual sirvió como base para el desarrollo de esquemas de solución para las ecuaciones de Navier-Stokes para flujo incompresible. [2]

Para la década de 1980 se comienza a dar solución a las ecuaciones de Euler tanto en dos dimensiones como en tres dimensiones. Para mediados de esta década, gracias de nuevo, al incremento en la capacidad de cálculo de los ordenadores, se comienza a hacer análisis de flujos viscosos, los cuales son descritos por las ecuaciones de Navier-Stokes. [4] Es también en el año 1980 que Steger y Chausee proponen un esquema para la generación de mallas estructuradas mediante el uso de ecuaciones diferenciales parciales hiperbólicas. A lo largo de toda esta década son explorados diferentes algoritmos para la generación de mallas mediante ecuaciones hiperbólicas.

A finales de la década de 1980, comenzó una nueva etapa el desarrollo de técnicas para la generación de mallas. Dicha etapa se caracteriza por la implementación de códigos de generación de mallas comprehensivas, multi propósito, tridimensionales. [3]

1.2. La dinámica de fluidos en la actualidad

La DFC tiene hoy en día, un rol tan importante, que puede considerarse una tercera rama de la dinámica de fluidos, siendo las dos primeras las ramas experimental y la teoría pura. [5]

Actualmente, la DFC se aplica en diversos sectores, como el aeroespacial, diseño de turbomecánica, carros y navíos. Además de campos como la meteorología, oceanografía, astrofísica e incluso arquitectura. Algunas técnicas desarrolladas para la DFC ahora se usan también en la solución de las ecuaciones de Maxwell, lo que demuestra la creciente importancia que tiene la DFC como herramienta en ingeniería. [4]

Uno de los ámbitos en los que la DFC ha tenido mayor impacto es en el diseño de aeronaves, gracias al rápido decremento en los costos de computación, ocasionado por una mejora en las capacidades de cálculo y tecnológicas de los ordenadores, así como de el fácil acceso a los mismos, aunado a un incremento en el costo de experimentos en túneles de viento. Esto ha dado como resultado que el cálculo de las características aerodinámicas de una aeronave resulte más barato mediante DFC que mediante la experimentación en túnel de viento. De modo que en la industria el diseño preliminar de aeronaves se realiza mediante cálculos en computadoras, mientras que los detalles del diseño final se analizan experimentalmente. [5]

En la actualidad, la DFC es capaz de analizar flujos laminares sin mayor complicación, sin embargo, aún no es capaz de analizar flujos turbulentos, los cuales se presentan en la mayoría de casos prácticos, por lo que se debe de recurrir a modelos de turbulencia, los cuales en general suelen dar buenos resultados. Cabe recalcar que ningún modelo es universal, por lo cual se debe prestar atención al problema que se analizará para así escoger de manera correcta el modelo a emplear. [6]

1.3. La generación de mallas en la actualidad

En la actualidad se ha desarrollado un gran número de métodos avanzados para la generación de mallas, entre los que destacan los métodos algebráicos, mediante la solución de ecuaciones diferenciales parciales elípticas, hiperbólicas, parabólicas, mallas variables, etc. La creación y mejora de todas estas técnicas nos ha llevado a un punto en el que se pueden realizar análisis en dominios de geometrías complejas.

Debido al desarrollo exitoso de esta área, el campo de generación numérica de mallas puede ser considerado una nueva disciplina matemática, con su propia metodología, tecnología y su propio enfoque.

La generación numérica de mallas, al ser una herramienta de amplia utilización tanto en la industria de la ingeniería, como en campos de computación científica, es ahora reconocida como una asignatura en algunas universidades. [3]

1.4. Software disponible para DFC

OpenFOAM

OpenFOAM (Open Source Field Operation and Manipulation) es un entorno de trabajo para el desarrollo de ejecutables de aplicaciones que usa las funciones contenidas en aproximadamente 100 librerías escritas en C++. Contiene solvers específicos para cada tipo de problema en mecánica de fluidos, así como de mecánica del medio continuo. [7]

OpenFOAM se distribuye bajo la licencia GPL (General Public License), la cual garantiza a los usuarios finales la libertad de usar, estudiar, compartir y modificar el código fuente del software, lo cual es basicamente, la filosofía del software libre.

Xfoil

Es un programa interactivo para el diseño y análisis de perfiles operando en regímenes subsónicos. Dadas las coordenadas de la nube de puntos que describen la forma de un perfil, un número de Reynolds y el número de Mach, Xfoil puede calcular la distribución de presión a lo largo del perfil, y con esto, las fuerzas de levantamiento y arrastre. Consiste en una colección de funciones como:

- Análisis Viscoso de perfiles alares existentes.
- Diseño y rediseño de perfiles mediante la modificación de la distribución de velocidad superficial
- Rediseño de perfiles mediante la modificación de sus parámetros geométricos. [8]

Xfoil es desarrollado por el MIT (Massachusetts Institute of Technology), escrito en lenguaje FORTRAN y distribuido bajo la licencia GPL.

XFLR5

XFLR5 es una herramienta desarrollada para el análisis de perfiles alares, alas y aviones volando a bajos Números de Reynolds. Fue creado con dos propósitos principales, dar una interfaz amigable al programa “Xfoil”, y pasar el código fuente original de FORTRAN a lenguaje C / C++, es decir que los algoritmos con los que se desarrolló Xfoil son exactamente los mismos en XFLR5. [9] Este programa no ha sido desarrollado con fines profesionales, sino para uso meramente personal y se distribuye bajo la licencia GPL.

XFlow

Es un software privativo desarrollado por la empresa Dassault Systèmes, el cual ofrece soluciones mediante el método de Lattice-Boltzman basado en partículas. El usuario puede analizar casos de aerodinámica transitoria, flujos multifase complejos, acústica aérea, geometrías en movimiento e interacciones fluido-estructura. Destaca por sus capacidades de renderizado. [10]

ANSYS Fluent

Ansys Fluent es un software privativo creado por la empresa ANSYS, Inc. dedicado a la simulación de DFC, implementado para dar soluciones mediante el método de volúmenes finitos.

SU2

SU2 es una colección de software “*open source*” desarrollado en los lenguajes Python y C++ para el análisis de ecuaciones diferenciales parciales mediante métodos numéricos del actual estado del arte de la DFC. Enfocado principalmente a su aplicación en la industria aeronáutica, así como la industria automotriz, naval y de energías renovables, entre otras. [11]

Helyx

Paquete “*open source*” para análisis de dinámica de fluidos computacional, desarrollado a partir de OpenFOAM. Es compatible con la mayoría de sistemas Linux y Windows. Además del sistema base para los análisis, presenta diversas herramientas enfocadas en facilitar el despliegue del software. Proporciona una interfaz gráfica, a diferencia de OpenFOAM y SU2 con los cuales la interacción se interacciona con el programa mediante la línea de comandos. También proporciona servicios en la nube. A pesar de ser software de código abierto, es de paga. [12]

Capítulo 2

Discretización Espacial

Las soluciones analíticas modeladas mediante ecuaciones diferenciales parciales de las ecuaciones que describen el flujo de fluidos dan resultados “continuos” a lo largo del dominio que se está estudiando, por otro lado, las soluciones numéricas dan resultados en puntos “discretos” del dominio llamados nodos, éstos, unidos mediante líneas, conforman la malla del dominio en el que se está trabajando el problema. Para el caso de un análisis en dos dimensiones, la malla se forma por triángulos o cuadriláteros, mientras que en un caso de tridimensional la malla es conformada por tetraedros o hexaedros.

En las aplicaciones de DFC se busca que la distribución de los nodos sea uniforme, dado que esto simplifica los algoritmos de solución a implementar, y que se traduce en un ahorro de tiempo en la ejecución de los programas, además de un uso considerablemente menor de memoria.

Existen básicamente dos tipos de mallas:

- Estructuradas: los nodos de la malla están alineados entre sí y son identificados mediante índices i, j, k . Una malla estructurada bidimensional está formada por cuadriláteros, mientras que una malla tridimensional se forma por hexahédros. Figura 3.2a.
- No estructuradas: los nodos no tienen un orden particular, esto implica que no se puedan identificar mediante índices, por lo que se debe llevar a cabo una forma diferente de identificación. Figura 3.2b

2.1. Método de Diferencias Finitas

El método de diferencias finitas fue de los primeros métodos numéricos que se emplearon para la resolución de ecuaciones diferenciales parciales, y ha sido hasta la fecha, uno de los más utilizados en aplicaciones de DFC. Este método está basado en las series de expansión de Taylor.

“El uso del método de diferencias finitas en la DFC tiene como propósito el reemplazar las derivadas parciales que aparecen en las ecuaciones que gobiernan el flujo, por coeficientes diferenciales que dan como resultado un sistema de ecuaciones algebraicas, el cual se resuelve para obtener las propiedades del campo de flujo en puntos discretos, es decir, en los nodos de la malla.” [5]

2.1.1. Desarrollo del método

Supongamos que se tiene el valor de una propiedad de flujo de $f_{i,j}$ en un punto (i, j) y se pretende calcular el valor de dicha propiedad en un punto $(i+1, j)$ (Figura 2.1) es decir, se quiere conocer el valor de $f_{i+1,j}$, dicho valor se puede expresar mediante una serie de expansión de Taylor, quedando:

$$f_{i+1,j} = f_{i,j} + \left(\frac{\partial f}{\partial x} \right)_{i,j} \Delta x + \left(\frac{\partial^2 f}{\partial x^2} \right)_{i,j} \frac{(\Delta x)^2}{2!} + \left(\frac{\partial^3 f}{\partial x^3} \right)_{i,j} \frac{(\Delta x)^3}{3!} + \cdots + \left(\frac{\partial^n f}{\partial x^n} \right)_{i,j} \frac{(\Delta x)^n}{n!} \quad (2.1)$$

Matemáticamente, esta serie es una solución exacta bajo una de dos condiciones:

- La serie está formada por un número infinito de términos, haciendo que la serie converja
- El valor del intervalo entre puntos tiende a cero ($\Delta x \rightarrow 0$)

Para un análisis computacional resulta impráctico el trabajar con un número infinito de términos, por lo que se trunca la ecuación y se busca refinar la malla, haciendo el intervalo Δx lo más pequeño posible, incrementando de esta manera la exactitud de la solución.

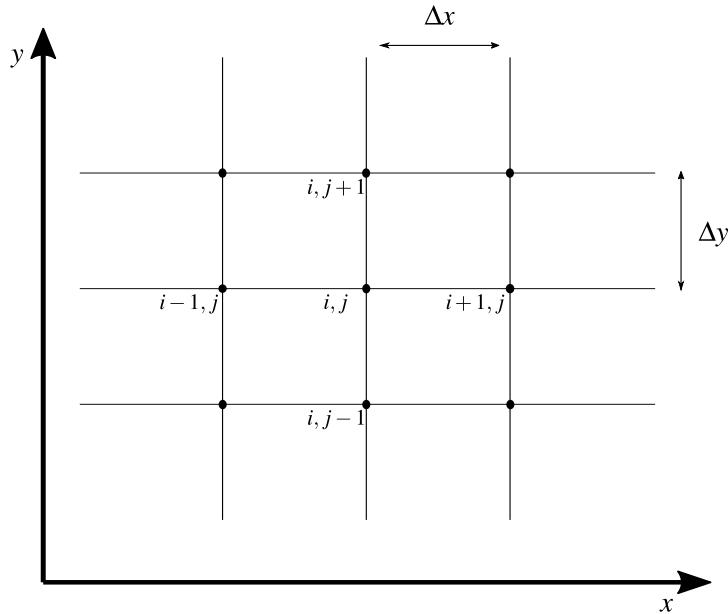


Figura 2.1: Discretización espacial (malla) por el método de diferencias finitas [5]

Todos los términos de la serie truncados se agrupan en un sólo término conocido como error de truncamiento y es representado por la simbología $O()$. Volviendo a la ecuación (2.1) y resolviendo para $\left(\frac{\partial f}{\partial x} \right)_{i,j}$ tenemos:

$$\left(\frac{\partial f}{\partial x} \right)_{i,j} = \frac{f_{i+1,j} - f_{i,j}}{\Delta x} - \left(\frac{\partial^2 f}{\partial x^2} \right)_{i,j} \frac{\Delta x}{2} - \left(\frac{\partial^3 f}{\partial x^3} \right)_{i,j} \frac{(\Delta x)^2}{6} - \dots \quad (2.2)$$

o bien:

$$\left(\frac{\partial f}{\partial x} \right)_{i,j} = \frac{f_{i+1,j} - f_{i,j}}{\Delta x} + O(\Delta x)^2$$

que para simplificar, queda:

$$\left(\frac{\partial f}{\partial x} \right)_{i,j} = \frac{f_{i+1,j} - f_{i,j}}{\Delta x} \quad (2.3)$$

El símbolo $O(\Delta x)$ representa el error de truncamiento de la serie de expansión, y para este caso particular se dice que tiene una aproximación de primer orden, ya que se están despreciando los términos de orden superior.

La ecuación 2.3 se le conoce como una aproximación tipo “forward” de primer orden y representa el valor aproximado del valor de la derivada parcial $\left(\frac{\partial f}{\partial x} \right)_{i,j}$ calculada mediante la relación lineal del punto (i, j) con el punto $(i + 1, j)$. (Figura 2.2a)

Para obtener una aproximación tipo “backward” se debe generar una serie de Taylor para el punto $(i - 1, j)$ expandiendo a partir del punto (i, j)

$$f_{i-1,j} = f_{i,j} + \left(\frac{\partial f}{\partial x} \right)_{i,j} (-\Delta x) + \left(\frac{\partial^2 f}{\partial x^2} \right)_{i,j} \frac{(-\Delta x)^2}{2!} + \left(\frac{\partial^3 f}{\partial x^3} \right)_{i,j} \frac{(-\Delta x)^3}{3!} + \cdots + \left(\frac{\partial^n f}{\partial x^n} \right)_{i,j} \frac{(-\Delta x)^n}{n!} \quad (2.4)$$

Una vez generada la serie, se sigue el mismo procedimiento empleado en la derivación de la aproximación “forward” para obtener:

$$\left(\frac{\partial f}{\partial x} \right)_{i,j} = \frac{f_{i,j} - f_{i-1,j}}{\Delta x} \quad (2.5)$$

que es la ecuación de una aproximación “backward”. (Figura 2.2b)

Para diferentes aplicaciones de DFC, no basta con tener aproximaciones con precisión de primer orden, por lo cual se opta por generar una ecuación que tenga una precisión de segundo orden, a esta se le conoce como “aproximación central de segundo orden”. La deducción de ésta, parte de la resta de la ecuación 2.1 y la ecuación 2.4, que da como resultado:

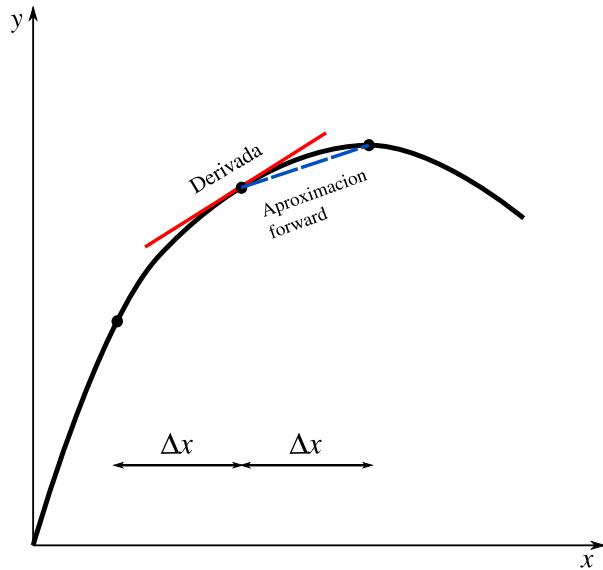
$$f_{i+1,j} - f_{i-1,j} = 2 \left(\frac{\partial f}{\partial x} \right)_{i,j} \Delta x + 2 \left(\frac{\partial^3 f}{\partial x^3} \right)_{i,j} \frac{(\Delta x)^3}{3!} + \cdots + 2 \left(\frac{\partial^n f}{\partial x^n} \right)_{i,j} \frac{(\Delta x)^n}{n!} \quad (2.6)$$

que se puede simplificar a:

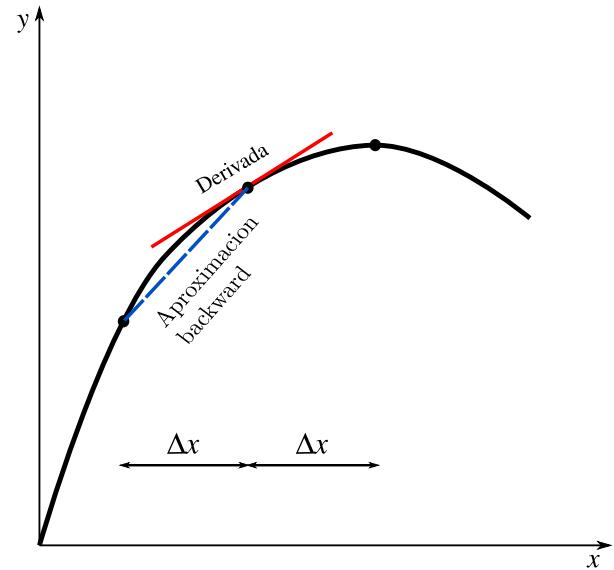
$$\left(\frac{\partial f}{\partial x} \right)_{i,j} = \frac{f_{i+1,j} - f_{i-1,j}}{2\Delta x} \quad (2.7)$$

se observa en la ecuación 2.7 que para obtener un coeficiente en el punto (i, j) se está utilizando información proveniente de los nodos $(i - 1, j)$ e $(i + 1, j)$, adyacentes a dicho punto. (Figura 2.2c)

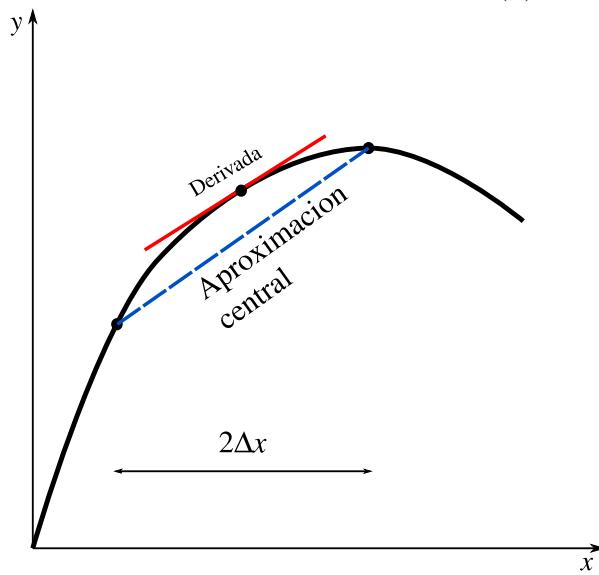
Para el análisis de flujos viscosos, además de los coeficientes que sustituyen a las derivadas parciales de primer orden se necesitan desarrollar aproximaciones para las derivadas parciales de segundo orden, dado que, en las ecuaciones que describen el flujo de fluidos viscosos, las ecuaciones de Navier-Stokes, los términos de mayor orden son derivadas parciales de segundo orden.



(a) Aproximación forward



(b) Aproximación backward



(c) Aproximación central

Figura 2.2: Aproximaciones mediante diferencias finitas

Si se suman las ecuaciones 2.1 y 2.4, se obtiene un coeficiente para $\frac{\partial^2 f}{\partial x^2}$, como se muestra a continuación:

$$f_{i+1,j} + f_{i-1,j} = 2f_{i,j} + \left(\frac{\partial^2 f}{\partial x^2} \right)_{i,j} \frac{(\Delta x)^2}{2!} + \left(\frac{\partial^4 f}{\partial x^4} \right)_{i,j} \frac{(\Delta x)^4}{4!} + \cdots + \left(\frac{\partial^n f}{\partial x^n} \right)_{i,j} \frac{(\Delta x)^n}{n!}$$

despejando la segunda derivada parcial y despreciando el error de truncamiento:

$$\left(\frac{\partial^2 f}{\partial x^2} \right) = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2} \quad (2.8)$$

a la ecuación 2.8 se le conoce como aproximación central para la derivada de segundo orden.

Ahora se desea obtener una aproximación por diferencias finitas para una derivada parcial mixta, llámese $\frac{\partial^2 f}{\partial x \partial y}$, donde f es una función dependiente de la posición de la partícula de fluido a lo largo de dos ejes, x y y . Dado que:

$$\left(\frac{\partial^2 f}{\partial x \partial y} \right) = \frac{\partial}{\partial y} \left(\frac{\partial f}{\partial x} \right) \quad (2.9)$$

podemos desarrollar una aproximación “central” de la derivada parcial de y en función de x , es decir:

$$\left(\frac{\partial^2 f}{\partial x \partial y} \right)_{i,j} = \frac{\left(\frac{\partial f}{\partial y} \right)_{i+1,j} - \left(\frac{\partial f}{\partial y} \right)_{i-1,j}}{2\Delta x} \quad (2.10)$$

y desarrollando una diferencia central para la derivada parcial de f con respecto a y tenemos:

$$\left(\frac{\partial^2 f}{\partial x \partial y} \right)_{i,j} = \frac{1}{2\Delta x} \left[\frac{f_{i+1,j+1} - f_{i+1,j-1}}{2\Delta y} - \frac{f_{i-1,j+1} - f_{i-1,j-1}}{2\Delta y} \right] \quad (2.11)$$

$$\left(\frac{\partial^2 f}{\partial x \partial y} \right)_{i,j} = \frac{1}{4\Delta x \Delta y} (f_{i+1,j+1} + f_{i-1,j-1} - f_{i+1,j-1} - f_{i-1,j+1}) \quad (2.12)$$

La misma metodología se aplica para obtener las aproximaciones por el método de diferencias finitas de una función respecto a un eje cualquiera, llámese eje y , ξ o η .

La mayor ventaja que presenta el método de diferencias finitas es su simplicidad de implementación, aunque presenta una limitante, el método únicamente es aplicable a mallas estructuradas, además de tampoco poderse aplicar a cuerpos de geometría curvilíneas, por lo que es necesario hacer una transformación de la malla, de un dominio físico a un dominio lógico, como se analiza en el capítulo 3.

Es este último punto, la piedra angular en la importancia del desarrollo de métodos de generación de mallas, ya que se busca trabajar con mallas en las cuales, en sus sistema de coordenadas computacional, pueda ser aplicado el método de diferencias finitas para la solución de problemas.

Capítulo 3

Mallas

Las mallas proveen soporte matemático para llevar a cabo una solución numérica de las ecuaciones que gobiernan el campo de análisis como medio continuo. La solución numérica se obtiene superponiendo la malla sobre el medio continuo, discretizando las ecuaciones respecto a la malla y por último, aplicando un algoritmo numérico de solución a la discretización de las ecuaciones. Este proceso resulta ser una evaluación de la solución en los nodos de la malla.

La mecánica de fluidos trabaja con ecuaciones no lineales, las cuales describen los flujos de fluidos, dichas ecuaciones en la mayoría de los casos prácticos de análisis no pueden ser resueltas de una manera analítica, por lo que se ha optado por la solución de dichas ecuaciones mediante métodos de aproximaciones entre los cuales encontramos los métodos de expansión y perturbación, método de diferencias finitas, método de volúmenes finitos e incluso el método de elementos finitos. En general los de mayor aplicación práctica son los tres últimos, pero para poder usarlos es necesario discretizar el campo de análisis mediante el uso de una malla. [13]

La utilización del método de diferencias finitas es directa si el problema se puede expresar en un sistema de coordenadas cartesianas, aunque el mismo puede aplicarse a sistemas de coordenadas polares, cilíndricas o esféricas, dando como resultado una malla rectangular estructurada, con espaciamiento uniforme entre los nodos en la dirección de los ejes del sistema. La mayoría de casos prácticos de interés en la DFC, tratan con geometrías complejas, lo que hace que sea muy difícil, si no imposible generar una malla en alguno de los sistemas de coordenadas antes mencionados, que ajuste en su frontera interna de manera exacta a la forma de la geometría a analizar. Esto presenta una limitante que se debe atender, se debe llevar a cabo una transfofrmación de sistemas de coordenadas, llevando el dominio físico del problema a un dominio computacional. (Figura 3.1) El dominio computacional es una abstracción matemática, mientras que el dominio físico es el dominio continuo para el cual se desea una solución numérica.

Una malla es un conjunto de puntos distribuidos a lo largo de un campo dentro del cuál se llevarán a cabo los cálculos para obtener la solución de ecuaciones diferenciales parciales. Existen dos tipos principales de mallas: estructuradas y no estructuradas. (Figura 3.2) Las mallas estructuradas se forman mediante la intersección de coordenadas curvilíneas, formando celdas cuadriláteras en dos dimensiones, o hexahedros en tres dimensiones. En contraste, las mallas no estructuradas no tienen relación alguna con las direcciones coordenadas del sistema, éstas consisten generalmente de triángulos y tetrahédros en dos y tres dimendiones respectivamente, aunque en realidad se puede hacer uso de cualquier forma geométrica. Para este trabajo se considerará el desarrollo de mallas

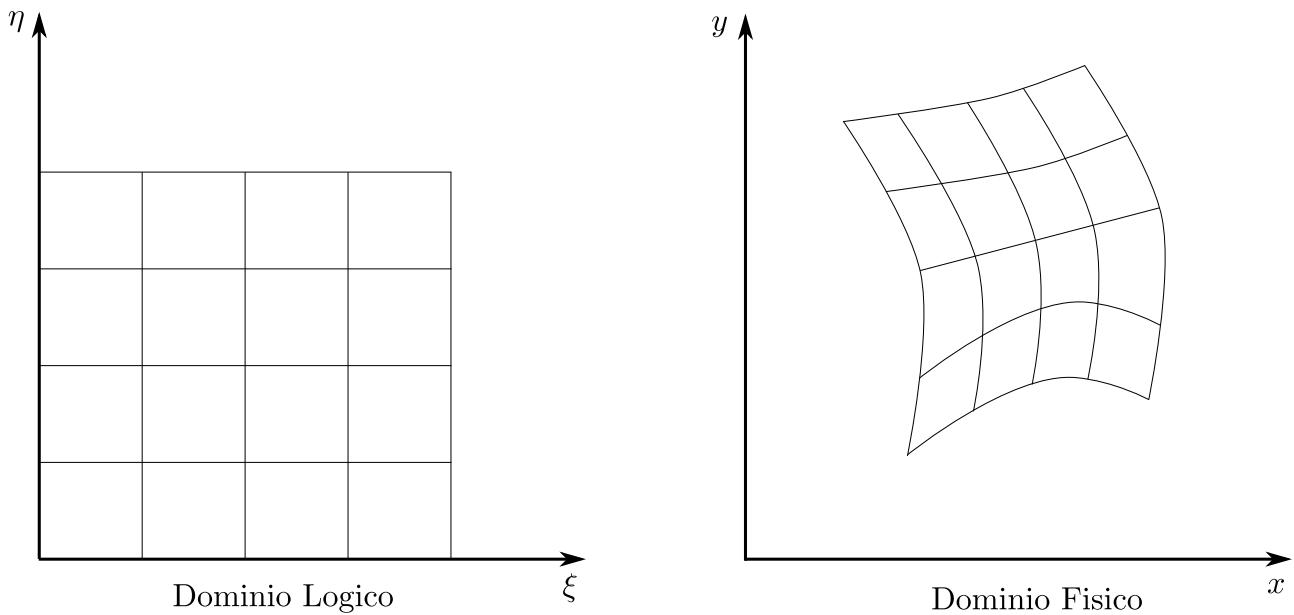


Figura 3.1: Dominios lógico (izquierda) y físico (derecha). [14]

estructuradas principalmente, que en principio tendrán una forma curvilínea ajustada a la forma del perfil alar con el que se esté trabajando como frontera interna del dominio.

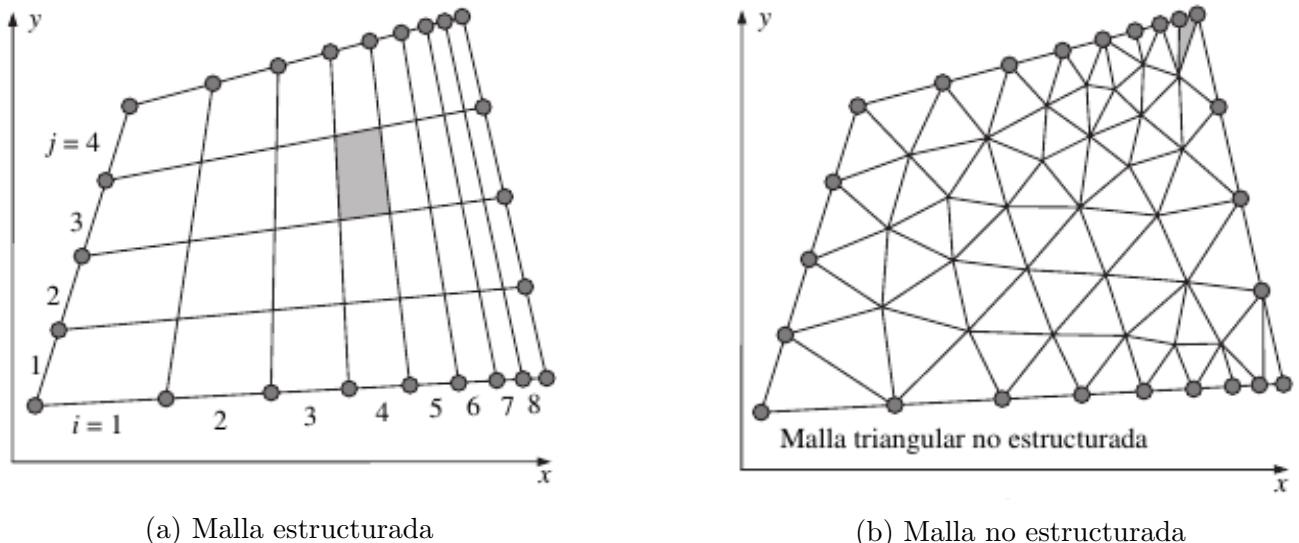


Figura 3.2: Mallas estructuradas y no estructuradas. [6]

La malla debe de generarse bajos ciertas restricciones las cuales suelen ser difíciles de satisfacer por completo. En la actualidad el tiempo que toma generar una malla llega a ser mayor en órdenes de magnitud que el tiempo requerido para la construcción y análisis de la solución del flujo sobre la malla. En especial ahora que hay mayor disponibilidad de software para la solución de flujos. [13]

3.1. Tipos de Mallas Estructuradas

Existen tres tipos base de mallas conocidos como mallas C, H u O respectivamente. El nombre que dichos tipos de mallado reciben se debe a que su estructura asemeja a dichas letras (en mayúsculas) desde una vista de planta.

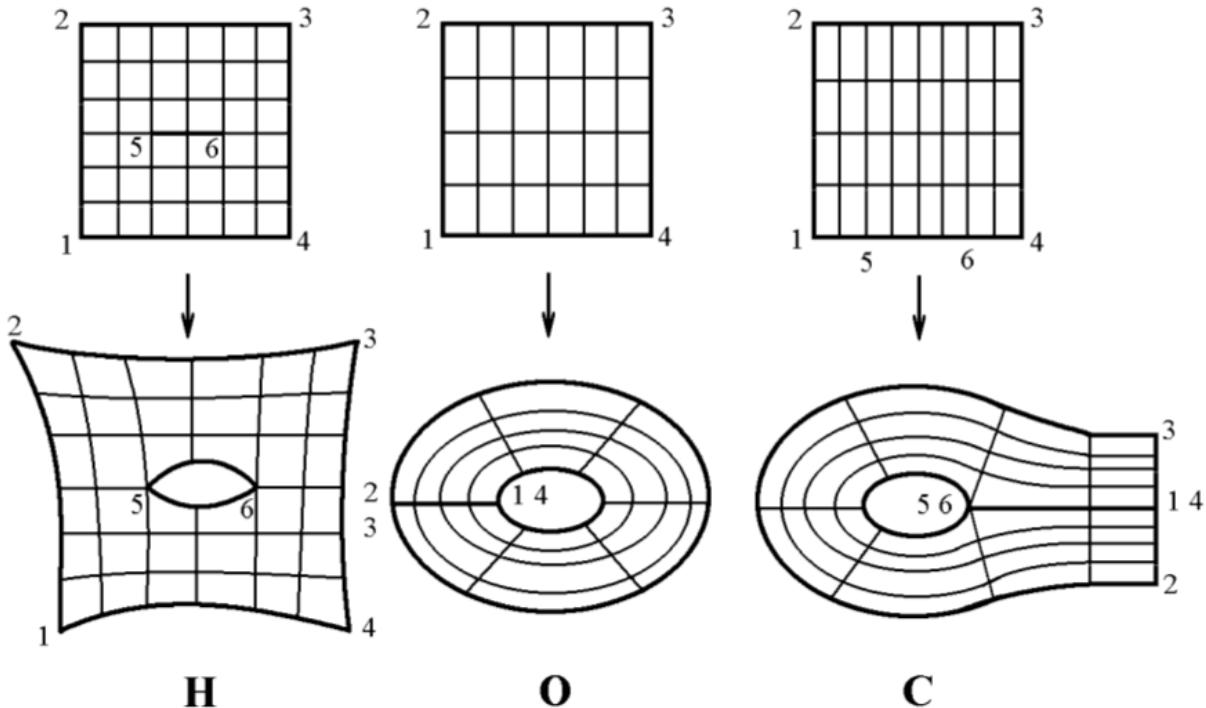


Figura 3.3: Tipología de malla y sus respectivos espacios computacionales.
H(izquierda), O(centro), C(derecha). [15]

3.1.1. Mallas tipo C

Las mallas tipos C están compuestas por una sección semicircular y otra rectangular en la frontera exterior, mientras que la frontera interior es de la misma forma del objeto que se analiza.

Suelen utilizarse para el análisis de flujos alrededor de perfiles alares, pues ofrecen como principal ventaja un buen análisis del flujo en la zona de deflexión de la estela, en especial si se compara contra las mallas tipo O. [16]

La transformación de esta malla de un espacio físico a un espacio computacional da como resultado una malla conformada por rectángulos. Durante la transformación se debe identificar un segmento, también conocido como corte, ya que este proceso implica conceptualmente, una separación en dicho corte, para posteriormente deformar el espacio físico llevándolo así a convertirse en una malla estructurada uniforme. En la figura 3.4 se observa que en el dominio físico, el segmento (ab) representa la sección del mallado donde se realiza el corte y que también se le llama ($a'b'$), y que en el dominio computacional esta sección representa dos segmentos diferentes.

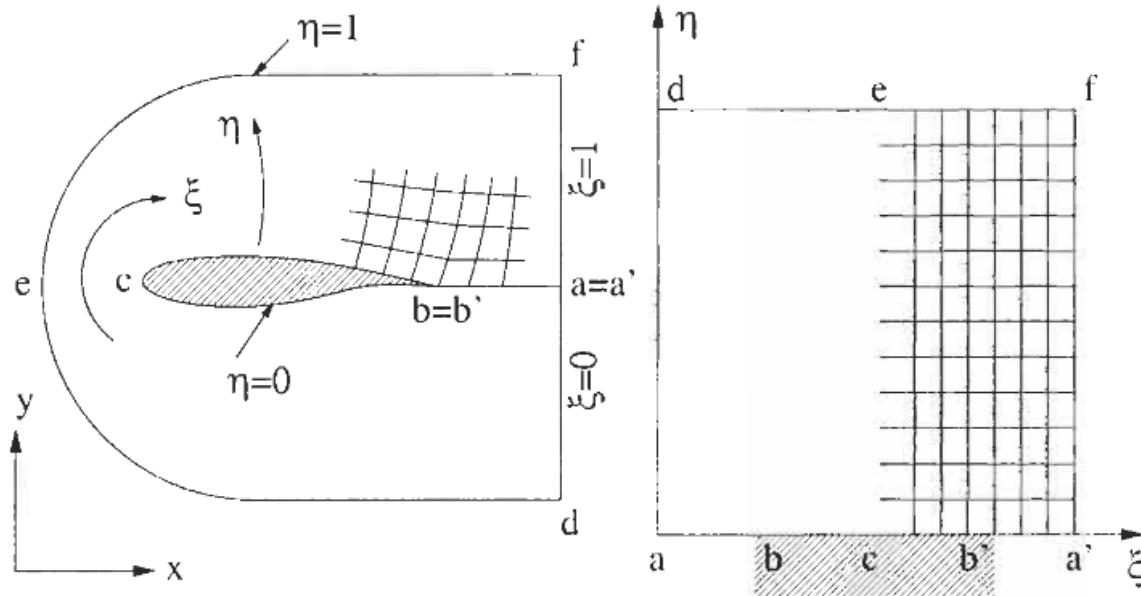


Figura 3.4: Malla tipo C y su transformación del espacio físico (izquierdo) al espacio lógico (derecha). [4]

3.1.2. Mallas tipo O

Este tipo de mallas son principalmente de sección circular, de ahí su nombre. Suelen ser utilizadas para el análisis del flujo alrededor de algunos componentes de aeronaves, como puede ser el fuselaje o las góndolas, entre otros. [15] También pueden llegar a utilizarse en el análisis de perfiles alares, pero presentan la dificultad de ofrecer una baja calidad de mallado en la sección cercana al borde de salida. [4] [16]

En este caso, al igual que sucede con las mallas tipo C, el resultado en el dominio lógico es un cuadrado. El proceso de transformación de este tipo de malla se puede conceptualizar como el desdoble del espacio físico a partir de un corte, en la figura 3.5 se representa como (ac) o como ($a'c'$), para después deformarlo hasta alcanzar la forma cuadrada esperada.

3.1.3. Mallas tipo H

Este tipo de mallas son utilizadas principalmente en los análisis de turbomaquinaria, específicamente en la zona de los alabes. [4] [16] También suelen utilizarse en el análisis de alas de aeronaves. [15]

La transformación en este tipo de malla, también da como resultado un dominio computacional cuadrado, que puede tener o no, un corte al interior el cual corresponde a la frontera interior del dominio físico. (Figura 3.3 (H)) La frontera externa del dominio computacional corresponde a la frontera externa del dominio físico.

La figura 3.6 muestra el uso de una malla tipo H para el análisis del flujo a través de los álabes de una turbina, y su transformación del dominio físico al computacional. En este caso no existe un

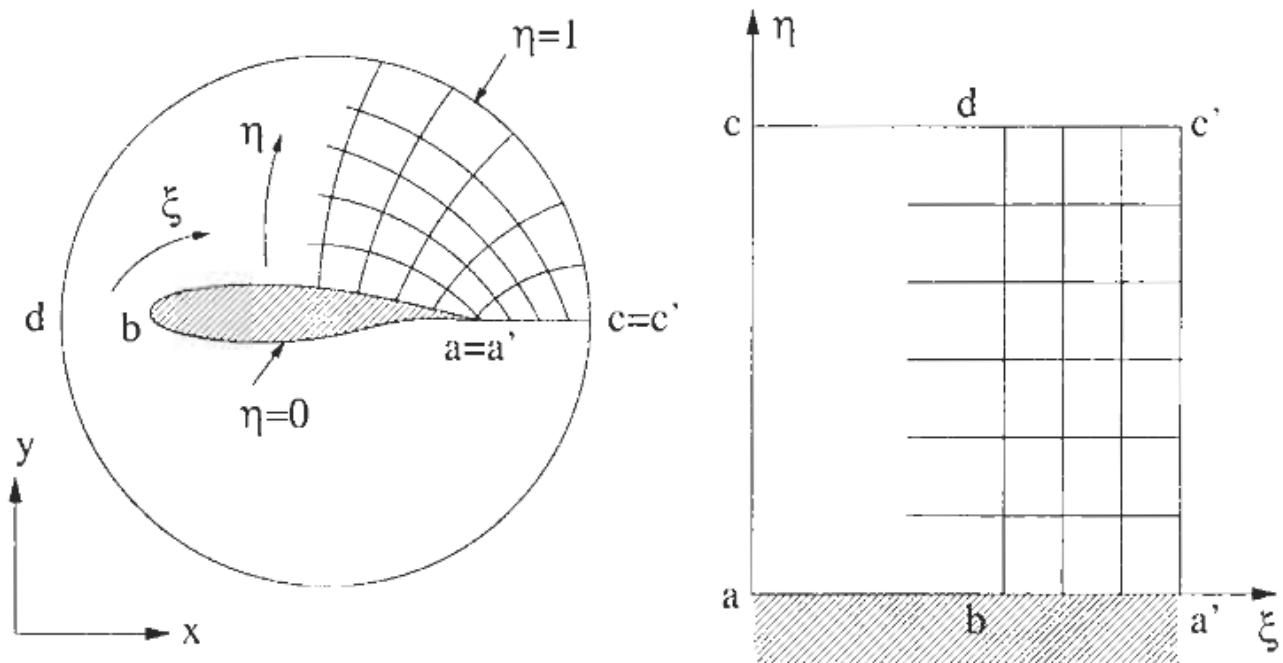


Figura 3.5: Malla tipo O y su transformación del espacio físico (izquierdo) al espacio lógico (derecha). [4]

cuerpo en el centro del mallado, por lo que no hay corte alguno en el dominio lógico.

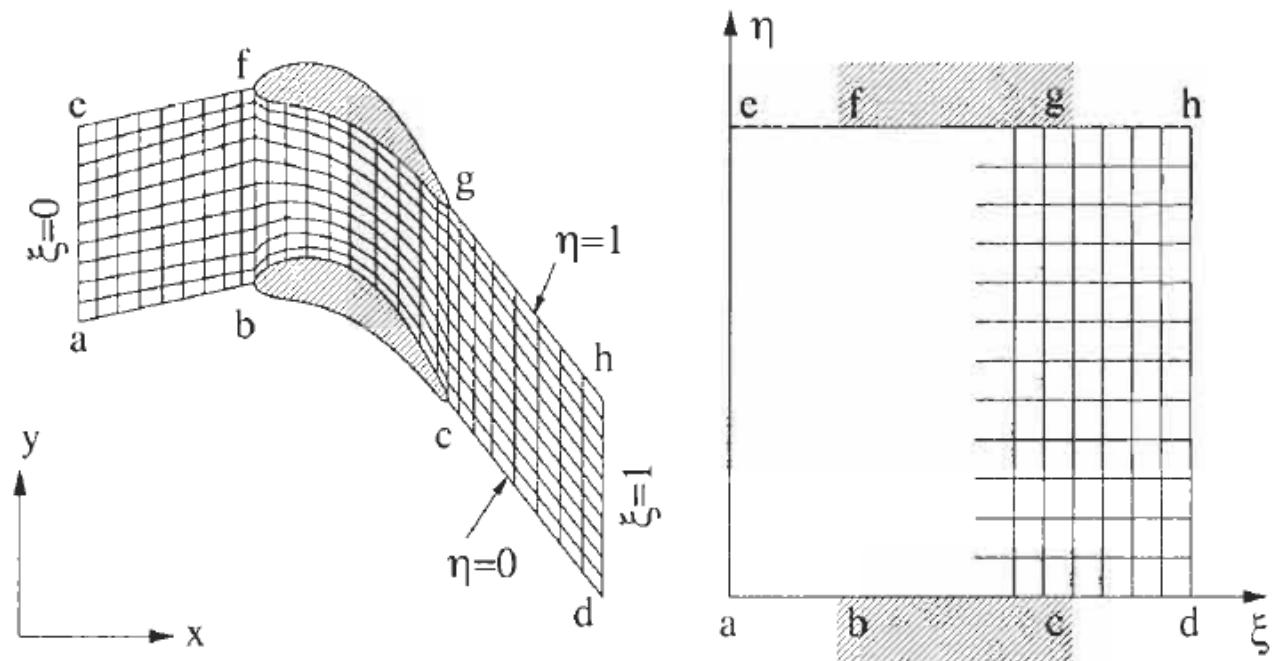


Figura 3.6: Malla tipo H y su transformación del espacio físico (izquierdo) al espacio lógico (derecha). [4]

En el caso de que se quiera ocupar una malla H con un cuerpo a analizar en el interior del dominio, como el caso representado en la figura 3.3 (H) se puede realizar un mallado uniforme, quizás mediante un método de interpolación algebráica, sobre el dominio físico quedando así coincidente con el dominio computacional. Los puntos 5 y 6 requieren de un trato especial al momento de llevar a cabo la solución, un ejemplo de esto puede ser que todos los puntos del mallado que caen dentro del objeto de análisis no sean utilizados durante la solución.

3.2. Mallas Adaptativas

A pesar de que existen varios modelos algebráicos y diferenciales para la generación de mallas, aún hay problemas para los cuales estos métodos no generen una malla adecuada, ya sea que la malla se aleje mucho de la ortogonalidad en ciertos puntos, que esté coprimida, e incluso no se llegue a la convergencia. Desde la década de 1960, época en la que la generación de mallas comienza a ser estudiada con mayor seriedad, se han analizado enfoques variables para solucionar los problemas presentados en los enfoques diferenciales.

3.3. Consideraciones preliminares para la generación de mallas

La densidad de la malla es el primer aspecto que se debe considerar, se debe generar una malla solo suficientemente densa para que la aproximación numérica sea precisa, sin embargo, se debe tener cuidado en no hacer la malla demasiado densa, tanto que resulte impráctico llevar a cabo un análisis y llegar a una eventual solución.

Otro importante aspecto a tener en cuenta durante este proceso, es la eficiencia computacional. Por un lado el uso de memoria podría ser tan grande que resulte impráctico llevar a cabo este proceso, por lo que una organización adecuada de la información es de vital importancia. Por otro lado, la generación de mallas debe llevarse a cabo mediante la implementación de códigos lo menos complejos posible.

3.4. Técnicas de generación del mallado

La generación de mallas *per se*, es un proceso libre durante su desarrollo, es decir, no se lleva a cabo mediante alguna fórmula o procedimiento estricto, por lo tanto, cualquier desarrollo se puede considerar apto para este propósito siempre y cuando dé como resultado una mallado adecuado para el análisis que se pretende realizar. Sin embargo, si hay ciertas consideraciones a tomar en cuenta, tales como la capacidad de manejar problemas con múltiples variables, las cuales podrían variar en varios órdenes de magnitud. Del mismo modo, se debe considerar la posibilidad de presentar factores de compresión en ciertas áreas de la malla, en contraste con una malla uniforme.

Podemos señalar principalmente dos tipos de técnicas para la generación de mallas estructuradas, de entre el amplio número de técnicas existentes al día de hoy, las cuales son:

- Métodos algebráicos: es de todos, el método más sencillo de implementar, una de sus ventajas es la rapidez con la que se genera la malla. Se utiliza un método de interpolación para determinar los nodos del mallado, partiendo de las coordenadas de la frontera externa e interna. Se realiza la transformación del dominio físico al lógico mediante una ecuación algebráica.
- Metódos mediante ecuaciones diferenciales parciales (EDP): se resuelve una EDP para obtener la distribución de los nodos, la resolución de las mismas se lleva a cabo mediante aproximaciones por el método de diferencias finitas. La distribución de la malla puede ser uniforme o concentrada dependiendo del tipo de ecuación que se resuelva.

3.5. Generación de mallas mediante métodos algebráicos

Como ya se mencionó previamente, los métodos más eficientes para la generación de mallas son los métodos algebráicos. Estos métodos basados en interpolación son de gran utilidad en la industria, debido a su sencilla implementación, además de la capacidad de control sobre la densidad de la malla respecto a un nodo dentro del dominio. Un aspecto en contra de la generación de mallas mediante estos métodos es que no tienen la capacidad de suavizar las curvas, si no que tienden a mantener la forma de las fronteras, por lo que si se llegara a presentar algún tipo de discontinuidad en la frontera interna (superficie de análisis), ésta seguirá presente en los nodos internos de la malla. Es por esto que estos métodos son utilizados de manera general, como una primera aproximación, la cual sirve como condición inicial en la generación de mallas mediante la solución de sistemas de ecuaciones diferenciales parciales. [17]

La idea fundamental sobre la cual se desarrollan todos los métodos algebráicos es el uso de funciones de interpolación matemática para interpolar entre algunos puntos conocidos o previamente asignados (fronteras) para así obtener la distribución de puntos que se ubican entre los ya mencionados. El método de interpolación puede variar dependiendo del método algebráico que se esté empleando, pero la idea base permanece. [18]

Los métodos algebráicos relacionan un dominio computacional, descrito por un cuadrado en 2D y un cubo en 3D, con un dominio físico de geometría arbitraria.

3.5.1. Interpolación unidireccional

La interpolación unidireccional se lleva a cabo interpolando una línea a la vez, en una dirección del espacio computacional, ya sea ξ o η . Es decir, para interpolar en la dirección ξ (desarrollo que se presenta en este trabajo), se deben seleccionar puntos en la frontera tanto interna como externa, que correspondan a la misma línea $\eta = \text{constante}$ y a lo largo de la cual se hará la interpolación, haciendo la variación en la coordenada ξ , interpolando así, línea por línea hasta terminarse de generar la malla. El mismo procedimiento se puede llevar a cabo interpolando en la dirección de η .

Interpolación Polinomial

En este método se ocupa el polinomio de Lagrange como ecuación de interpolación para generar la malla.

$$L_i(x) = \frac{(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0)(x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} \quad i = 0, 1, \dots, n \quad (3.1)$$

El polinomio resultante será de grado n , donde el numerador omite el término $(x - x_i)$

El caso más sencillo con el que se puede trabajar el polinomio de Lagrange es generando un polinomio de grado 1, que resulta ser una línea recta que pasa a través de dos puntos (x_0, y_0) y (x_1, y_1) , los cuales pertenecen a la frontera interna y externa respectivamente. Para este caso, se tienen los siguientes términos:

$$L_0 = \frac{(x - x_1)}{(x_0 - x_1)} \quad L_1 = \frac{(x - x_0)}{(x_1 - x_0)}$$

dando como resultado la ecuación que describe a una recta

$$y = y_0 \frac{(x - x_1)}{(x_0 - x_1)} + y_1 \frac{(x - x_0)}{(x_1 - x_0)} = y_0(1 - \xi) + y_1\xi \quad (3.2)$$

de donde se sabe que

$$\xi = \frac{x - x_0}{x_1 - x_0} \quad (3.3)$$

y se observa que $\xi = 0, 1$ cuando $x = x_0, x_1$ respectivamente.

Despejando x de la ecuación 3.3 se obtiene:

$$x = (x_1 - x_0)\xi - x_0 = x_0(1 - \xi) + x_1\xi \quad (3.4)$$

por lo que la ecuación constitutiva de éste método, para una interpolación en dirección del eje ξ del espacio computacional puede escribirse de la siguiente manera:

$$r(\xi, \eta_j) = (1 - \xi)r(0, \eta_j) + \xi r(1, \eta_j) \quad (3.5)$$

Se puede hacer uso de la generación de un polinomio de mayor grado, para lo cual se deben proporcionar más puntos previamente definidos por los cuales tiene que pasar la curva descrita del polinomio. Para generar una ecuación de grado n se deben proporcionar como datos de inicio, al menos $n+1$, es decir, si se quisiera hacer la interpolación mediante un polinomio de grado 2, se debe proporcionar la información de al menos 3 puntos. Siempre cabe la posibilidad de que los puntos pertenezcan a una linea recta, en cuyo caso los términos de mayor orden se verán eliminados.

Las figuras 3.7 y 3.8 presentan una malla y su vista de detalle generada mediante este método alrededor de un perfil aerodinámico.

Interpolación mediante polinomios de Hermite

La interpolación mediante el polinomio de Lagrange se realiza mediante el uso de valores conocidos de ciertos puntos, sin embargo, es posible generar una malla partiendo tanto de valores conocidos de una función como de los valores de su primer derivada para ciertos puntos dados. El polinomio que describe este método se escribe como:

$$p(x) = \sum_{i=0}^n y_i H_i(x) + \sum_{i=0}^n y'_i \tilde{H}_i(x) \quad (3.6)$$

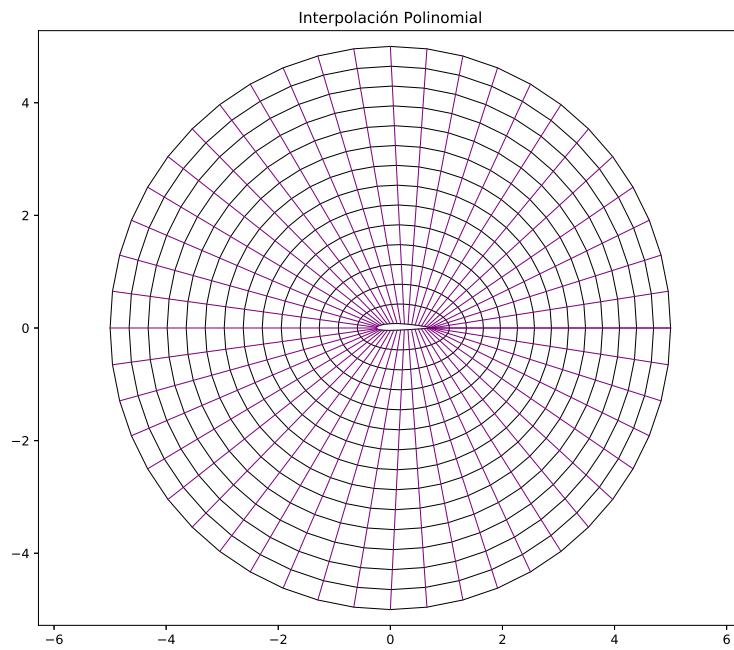


Figura 3.7: Malla tipo O generada mediante el método de interpolación polinomial alrededor de un perfil NACA 2412. Densidad de malla 49×15

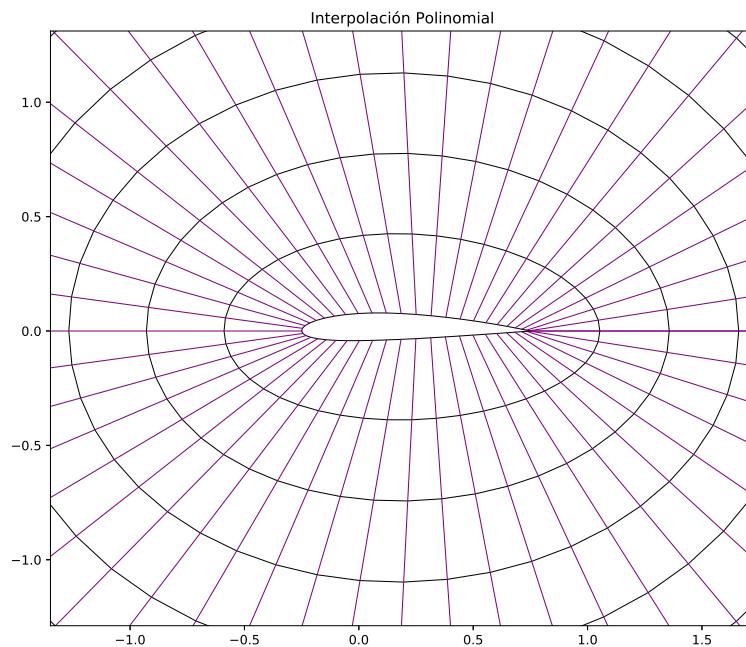


Figura 3.8: Vista de acercamiento a la malla generada de la figura 3.7

Los siguientes polinomios definen los polinomios de Hermite $H_i(x)$ y $\tilde{H}_i(x)$ en función de polinomios de Lagrange:

$$H_i(x) = \{1 - 2L\ell_i(x_i)(x - x_i)\} [L_i]^2 \quad (3.7)$$

$$\tilde{H}_i(x) = (x - x_i) [L_i]^2 \quad (3.8)$$

Se suele usar polinomios de Hermite cubicos, para los cuales se requiere el uso de polinomios de Lagrange de primer grado, es decir polinomios lineales. La ecuación constitutiva de la interpolación en dirección del eje ξ , por polinomios cúbicos de Hermite queda expresada de la siguiente manera:

$$r(\xi) = \sum_{i=0}^n r_i H_i(\xi) + \sum_{i=0}^n r' i \tilde{H}_i(\xi) \quad (3.9)$$

que a su vez, desarrollando las operaciones matemáticas, se expresa como:

$$\begin{aligned} r(\xi, \eta_j) = & r(0, \eta_j)(2\xi^3 - 3\xi^2 + 1) + r(1, \eta_j)(3\xi^2 - 2\xi^3) \\ & + r'(0, \eta_j)(\xi^3 - 2\xi^2 + \xi) + r'(1, \eta_j)(\xi^3 - \xi^2) \end{aligned} \quad (3.10)$$

3.5.2. Interpolación multidireccional

La interpolación multidireccional consiste en la obtención de una ecuación algebraica que permita la interpolación simultánea en 2 o más direcciones. Para este caso (2D), se generan ecuaciones que permitan la interpolación simultánea, en el espacio computaciones, en ambas direcciones ξ y η .

Interpolación Transfinita - TFI

El método de TFI es un método de interpolación multivariable. Cuando se aplica el TFI para la generación de mallas, se restringe la malla para que coincida con la fronteras especificadas. Este método consiste en la suma de las interpolaciones de una sola variable para cada una de las coordenadas computacionales. Existen diversos tipos de interpolación a lo largo de una coordenada, por lo que existen prácticamente un número ilimitado de variantes de TFI creadas a partir de la combinación de diversas interpolaciones de una sola variable. [13]

Una de las principales ventajas del método de interpolación transfinita es que asegura una concordancia en las fronteras en ambas direcciones. Su esencia es la de la interpolación en cada una de las coordenadas computacionales, formando así los productos tensores de las interpolaciones, y finalmente, se lleva a cabo una suma “booleana”. Las interpolaciones unidireccionales son una combinación lineal de datos proporcionados por los usuarios del dominio físico para valores dados de las coordenadas del dominio lógico.

A partir de la existencia de una transformación $r = r(\xi, \eta)$ ($x = x(\xi, \eta)$, $y = y(\xi, \eta)$) la cual hace un mapeo de un cuadrado de dimensiones $0 < \xi < 1$, $0 < \eta < 1$ en el dominio computacional, con la región del dominio físico, delimitada por una frontera externa y una interna, que se pretende analizar. De igual modo, se puede escribir otra transformación P_ξ , la cual recibe el nombre de proyector, y que relaciona puntos en el espacio lógico con puntos en el espacio físico, definida como:

$$P_\xi(\xi, \eta) = (1 - \xi)r(0, \eta) + \xi r(1, \eta) \quad (3.11)$$

Como ya se explicó en la sección 3.5.1, esta transformación hace una transformación con la cual las fronteras en la dirección de ξ son mapeadas, mientras que las fronteras en dirección de η son reemplazadas por líneas rectas. De manera análoga se puede definir el proyector:

$$P_\eta(\xi, \eta) = (1 - \eta)r(\xi, 0) + \eta r(\xi, 1) \quad (3.12)$$

con el cual se preservan las fronteras en dirección η y se reemplazan las fronteras en ξ por líneas rectas.

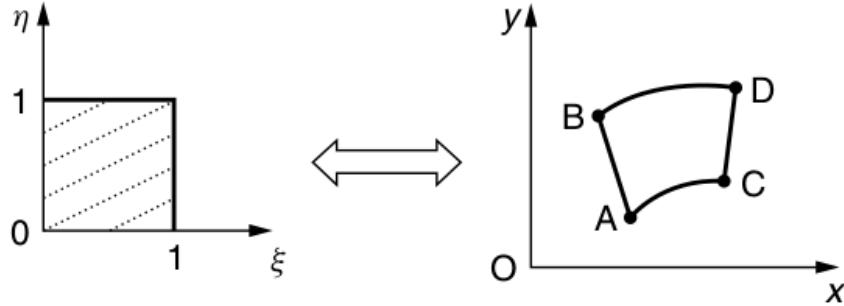


Figura 3.9: Transformación de una malla a través del proyector P_η [17]

La figura 3.9 representa gráficamente la transformación de la malla del dominio lógico al dominio computacional mediante la aplicación del proyector P_η . Se observa como las fronteras conformadas por los segmentos AC y BD se conservan durante las transformación, mientras que las fronteras AB y CD son reemplazadas por líneas rectas.

A partir de estas ecuaciones, se puede definir un mapeo compuesto $P_\xi P_\eta$, de tal manera que:

$$\begin{aligned} P_\xi(P_\eta(\xi, \eta)) &= P_\xi((1 - \eta)r(\xi, 0) + \eta r(\xi, 1)) \\ &= (1 - \xi)[(1 - \eta)r(0, 0) + \eta r(0, 1)] + \xi[(1 - \eta)r(1, 0) + \eta r(1, 1)] \quad (3.13) \\ &= (1 - \xi)(1 - \eta)r(0, 0) + (1 - \xi)\eta r(0, 1) + \xi(1 - \eta)r(1, 0) + \xi\eta r(1, 1) \end{aligned}$$

Con esta transformación compuesta, se logra preservar los 4 vértices en los que se interseccionan las fronteras, sin embargo, las cuatro fronteras son reemplazadas por líneas rectas. Es decir, las líneas $\xi = \text{constante}$ y $\eta = \text{constante}$ en el dominio computacional son transformadas como líneas rectas en el dominio físico, como se muestra en la figura 3.10.

Si consideramos los diferentes mapeos de un lado, bajo cualquiera de los proyectores previamente desarrollados, por ejemplo el lado $\eta = 0$, bajo el proyector P_ξ es mapeado con una línea recta AC , si por otro lado se lleva a cabo la transformación mediante el proyector P_η el mapeo resultante es uno a uno, es decir que la recta $\eta = 0$ se mapea con la curva de la frontera AC . Por último transformando con el proyector compuesto $P_\xi P_\eta$ también se hace un mapeo con la línea recta AC .

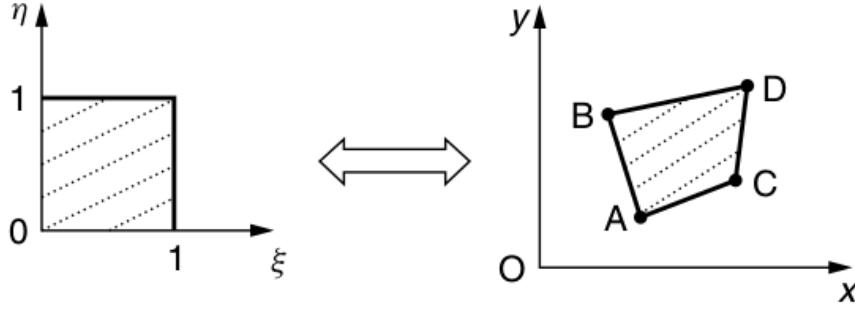


Figura 3.10: Transformación de una malla a través del proyector compuesto $P_\xi P_\eta$ [17]

Desarrollando la misma lógica y consideraciones a las 4 fronteras de la frontera, se concluye que el mapeo compuesto ($P_\xi + P_\eta - P_\xi P_\eta$) lleva a cabo una transformación con un mapeo uno a uno de las fronteras del cuadrado de longitud uitaria (dominio computacional) con la curvas de frontera del dominio físico. A este mapeo se le conoce como la suma booleana de las transformaciones P_ξ y P_η , y se escribe $P_\xi \oplus P_\eta$, por lo tanto:

$$P_\xi \oplus P_\eta = P_\xi + P_\eta - P_\xi P_\eta \quad (3.14)$$

con lo que la fórmula final queda:

$$\begin{aligned} (P_\xi \oplus P_\eta)(\xi, \eta) &= P_\xi(\xi, \eta) + P_\eta(\xi, \eta) - P_\xi P_\eta(\xi, \eta) \\ &= (1 - \xi)r(0, \eta) + \xi r(1, \eta) + (1 - \eta)r(\xi, 0) + \eta r(\xi, 1) \\ &\quad - (1 - \xi)(1 - \eta)r(0, 0) - (1 - \xi)\eta r(0, 1) - (1 - \eta)\xi r(1, 0) \\ &\quad - \xi\eta r(1, 1) \end{aligned} \quad (3.15)$$

Esta ecuación es la base del método de TFI para dos dimensiones. La malla se genera tomando valores discretos ξ_i, η_j para los ejes ξ y η .

3.6. Generación de mallas mediante EDP

La generación de mallas mediante estos métodos se ha esparcido gracias a la versatilidad que poseen y la relativa facilidad con la que pueden ser aplicados. La idea sobre la cual se desarrolla el método, es la de obtener la solución numérica de una ecuación diferencial parcial, esta solución representa las coordenadas de la malla, la cual debe coincidir en su frontera interna, con la forma geométrica del cuerpo que se pretende analizar. [18]

Existen tres esquemas predominantes de generación de mallas mediante la solución de sistemas de ecuaciones diferenciales parciales:

- Ecuaciones Elípticas
- Ecuaciones Hiperbólicas
- Ecuaciones Parabólicas

Generación de mallas mediante EDP elípticas

Para la solución de un sistema de EDP elípticas es necesario definir las condiciones de frontera a lo largo de todos los puntos pertenecientes a la misma, es decir, que se tengan condiciones de frontera de Dirichlet. La solución se obtiene mediante diferentes métodos numéricos iterativos entre los que destacan el método de Jacobi, el método de Gauss-Seidel y métodos de sobre relajación. Información sobre el procedimiento y desarrollo de estos métodos está disponible en el apéndice B.

La idea es calcular las coordenadas (ξ, η) , pertenecientes al dominio computacional, y que la solución del sistema de ecuaciones genere las correspondientes coordenadas (x, y) del dominio físico, con un mapeo uno a uno, es decir, que exista una relación adecuada entre el dominio computacional y el dominio físico.

Considerando la ecuación de Laplace:

$$\xi_{xx} + \xi_{yy} = 0 \quad (3.16a)$$

$$\eta_{xx} + \eta_{yy} = 0 \quad (3.16b)$$

donde, como ya se ha mencionado, ξ y η representan las coordenadas del dominio computacional y además, son variables dependientes de x y de y . Si se quiere trabajar con la forma inversa, es decir, que ahora x y y sean variables dependientes de ξ y η , la ecuación de Laplace se expresa como:

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = 0 \quad (3.17a)$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = 0 \quad (3.17b)$$

donde:

$$\alpha = x_\eta^2 + y_\eta^2$$

$$\beta = x_\xi x_\eta + y_\xi y_\eta$$

$$\gamma = x_\xi^2 + y_\xi^2$$

La distribución de los nodos, obtenidos mediante la solución de la ecuación de Laplace (ecuación 3.16), tiende a ser uniforme debido al efecto suavizante de dicha ecuación. Para poder manipular esta distribución, es necesario agregar funciones de forzado a la ecuación 3.16, dando como resultado la ecuación de Poisson:

$$\xi_{xx} + \xi_{yy} = P(\xi, \eta) \quad (3.18a)$$

$$\eta_{xx} + \eta_{yy} = Q(\xi, \eta) \quad (3.18b)$$

si de nuevo, esta ecuación es transformada, volviendo las variable x y y las variables dependientes, queda expresada como

$$\alpha x_{\xi\xi} + \beta x_{\xi\eta} + \gamma x_{\eta\eta} = -I^2[P(\xi, \eta)x_\xi + Q(\xi, \eta)x_\eta] \quad (3.19a)$$

$$\alpha y_{\xi\xi} + \beta y_{\xi\eta} + \gamma y_{\eta\eta} = -I^2[P(\xi, \eta)y_\xi + Q(\xi, \eta)y_\eta] \quad (3.19b)$$

Las funciones P y Q se seleccionan dependiendo de la necesidad específica del problema que se analiza. Dichas necesidades pueden ser, por ejemplo, el agrupamiento de nodos alrededor de un punto específico, o quizás sea la búsqueda de un sistema ortogonal en la superficie.

El uso de las funciones P y Q para manipular la densidad de puntos alrededor de ciertas zonas se logra atrayendo las líneas vecinas a una línea seleccionada. La misma funciona para los nodos, haciendo la atracción de los nodos colindantes hacia el nodo seleccionado. O bien, se puede llevar a cabo una combinación de ambos efectos.

Las ecuaciones para P y Q que llevan a cabo este proceso son:

$$P(\xi, \eta) = - \sum_{m=1}^M a_m \frac{\xi - \xi_m}{|\xi - \xi_m|} \exp(-c_m |\xi - \xi_m|) \\ - \sum_{n=1}^N b_n \frac{\xi - \xi_n}{|\xi - \xi_n|} \exp\{-d_n [(\xi - \xi_n)^2 + (\eta - \eta_n)^2]^{\frac{1}{2}}\} \quad (3.20)$$

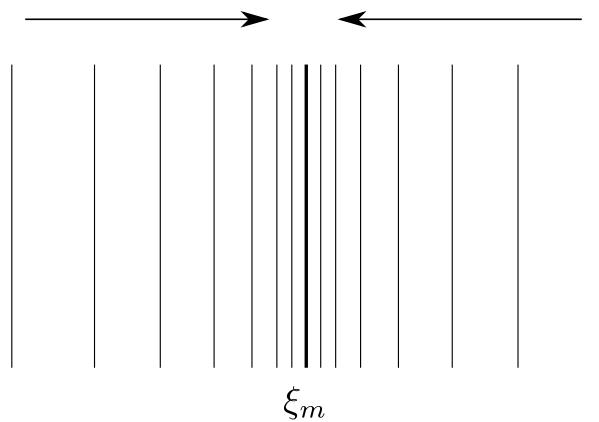
$$Q(\xi, \eta) = - \sum_{m=1}^M a_m \frac{\eta - \eta_m}{|\eta - \eta_m|} \exp(-c_m |\eta - \eta_m|) \\ - \sum_{n=1}^N b_n \frac{\eta - \eta_n}{|\eta - \eta_n|} \exp\{-d_n [(\xi - \xi_n)^2 + (\eta - \eta_n)^2]^{\frac{1}{2}}\} \quad (3.21)$$

estas ecuaciones fueron propuestas en 1974 por Thompson, Thames y Mastin [19], y son ampliamente referidas en diversos textos tanto de dinámica de fluidos computacional como de generación numérica de mallas.

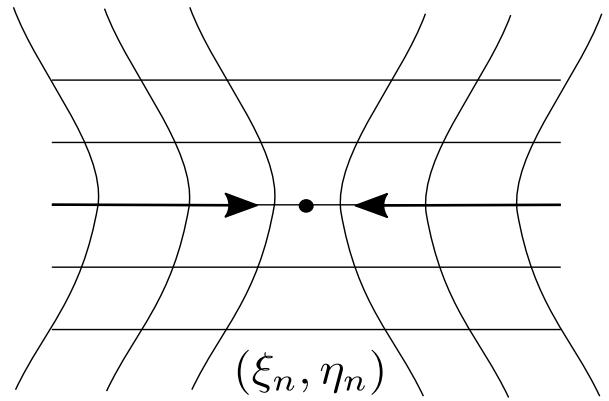
En las ecuaciones 3.20 y 3.21 el valor de M es el número de líneas existentes en la malla, tanto líneas coordenadas $\xi = \xi_m$ como líneas $\eta = \eta_m$ y N representa el número de nodos ($\xi = \xi_n, \eta = \eta_n, 0 \leq \xi_n, \eta_n \leq 1$) a los que la malla se verá atraída. Los factores a_m y b_n son factores de amplificación, por su parte los factores c_m y d_n son factores de decaimiento. Estos cuatro parámetros son datos de entrada para el código, asignados por el usuario, y los cuatro deben ser valores positivos.

El primer término en la ecuación para $P(\xi, \eta)$ tiene como efecto la atracción de las líneas ξ (líneas a lo largo de las cuales la coordenada en ξ permanece constante) hacia la línea $\xi = \xi_m$ en el dominio físico con una amplitud a_m , mientras que el segundo término atrae las líneas ξ hacia un punto determinado con una amplitud b_n . Estos parámetros tienen el mismo efecto en la ecuación de $Q(\xi, \eta)$, pero la atracción se ve reflejada en las líneas donde la coordenada η se mantiene constante.

Las funciones $(\xi - \xi_m)/|\xi - \xi_m|$ y $(\eta - \eta_m)/|\eta - \eta_m|$ son funciones que solo pueden dar como resultado valores ± 1 y su propósito dentro de la fórmula es garantizar que la atracción se dé, en caso de las líneas ξ y η por ambos lados de las mismas, y en todos los nodos vecinos para el caso de la atracción hacia un punto (ξ_n, η_n) . Asignar un valor negativo a los factores de amplitud da como resultado un efecto contrario, es decir, se crea un efecto de repulsión.

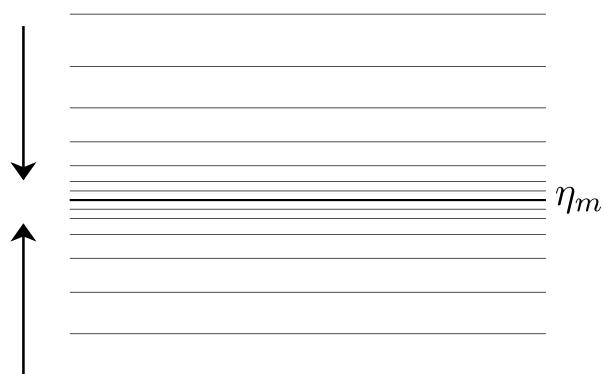


(a) Efecto de atracción a la línea $\xi = \xi_m$

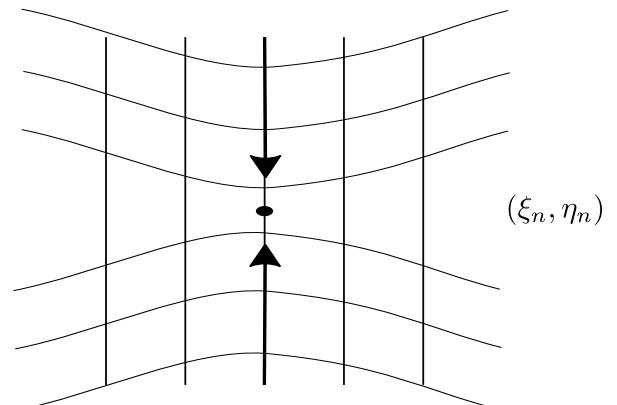


(b) Efecto de atracción por función al punto (ξ_n, η_n)

Figura 3.11: Efecto de atracción en el eje ξ por la función $P(\xi, \eta)$



(a) Efecto de atracción a la línea $\eta = \eta_m$



(b) Efecto de atracción al punto (ξ_n, η_n)

Figura 3.12: Efecto de atracción en el eje η por la función $Q(\xi, \eta)$

Las ecuaciones discretizadas para la generación de mallas mediante las ecuaciones de Laplace y Poisson quedan

$$X_{i,j} = \frac{1}{2(\alpha + \gamma)} \left[\alpha(X_{i+1,j} + X_{i-1,j}) + \gamma(X_{i,j+1} - X_{i,j-1}) \right] \quad (3.22)$$

$$- \frac{\beta}{2} (X_{i+1,j+1} + X_{i-1,j-1} - X_{i-1,j+1} - X_{i+1,j-1}) \quad (3.23)$$

$$+ \frac{I^2}{2} (P(X_{i+1,j} - X_{i-1,j}) + Q(X_{i,j+1} - X_{i,j-1})) \quad (3.24)$$

$$Y_{i,j} = \frac{1}{2(\alpha + \gamma)} \left[\alpha(Y_{i+1,j} + Y_{i-1,j}) + \gamma(Y_{i,j+1} - Y_{i,j-1}) \right] \quad (3.25)$$

$$- \frac{\beta}{2} (Y_{i+1,j+1} + Y_{i-1,j-1} - Y_{i-1,j+1} - Y_{i+1,j-1}) \quad (3.26)$$

$$+ \frac{I^2}{2} (P(Y_{i+1,j} - Y_{i-1,j}) + Q(Y_{i,j+1} - Y_{i,j-1})) \quad (3.27)$$

donde $P = 0$ y $Q = 0$ si se desea dar solución a la ecuación de Laplace.

Generación de mallas mediante EDP hiperbólicas

La generación de mallas mediante EDP elípticas puede ser demasiado costosa en términos de tiempo de cómputo, así como en el uso de memoria en el ordenador. Esto se debe a que dicho esquema intenta hacer coincidir un sistema de coordenadas curvilíneo a cuatro curvas de frontera, seis en mallas tridimensionales. De aquí surge un nuevo enfoque, en el cual solo se requiere iniciar la solución con una sola frontera e ir avanzando hacia afuera en el dominio físico usando EDP hiperbólicas. [17]

Con este propósito se toma en cuenta el trabajo publicado por Steger y Chausse [20] para la generación de mallas bidimensionales ortogonales y con control del área de celda. Se inicia el proceso con una frontera única en la que se considera que $\eta = 0$. Se propone también un sistema de ecuaciones de primer orden para las coordenadas x, y como funciones de ξ, η :

$$g_{12} = g_1 \cdot g_2 = x_\xi x_\eta + y_\xi y_\eta = 0 \quad (3.28a)$$

$$|g_1 \times g_2| = x_\xi y_\eta - x_\eta y_\xi = F \quad (3.28b)$$

La primera ecuación dota a la malla de ortogonalidad, mientras que la segunda tiene el término F como una medida del área de la celda (si el producto $\delta\xi\delta\eta$ es el mismo para cada celda). Se puede trabajar utilizando F como una función de ξ, η , lo que permitiría incrementar la densidad de la malla cerca de la frontera donde $\eta = 0$ haciendo que F tenga un valor pequeño en dicha zona. De las ecuaciones 3.28:

$$x_\eta = -\frac{F}{x_\xi^2 + y_\xi^2} y_\xi \quad (3.29a)$$

$$y_\eta = \frac{F}{x_\xi^2 + y_\xi^2} x_\xi \quad (3.29b)$$

Antes de proceder a la solución del sistema de las ecuaciones 3.28 se deben tomar en cuenta algunas consideraciones. En primera instancia debe tenerse en cuenta que el sistema es un sistema no lineal por lo que un proceso de linearización debe llevarse a cabo, una segunda consideración es que al ser un sistema de ecuaciones hiperbólicas se debe llevar a cabo el proceso de solución mediante un proceso de marcha, en este esquema la marcha se realiza en la dirección del eje η . Tercera, para un sistema de ecuaciones hiperbólicas se debe especificar una condición inicial así como una condición de frontera y por último, con el objetivo de evitar la presencia de oscilaciones, se puede agregar un término de “damping”, o viscosidad artificial, al lado derecho de las ecuaciones.

Para el proceso de linearización, se utiliza el esquema iterativo de Newton, el cual especifica que un término no lineal es aproximado mediante

$$AB = A^{k+1}B^k + B^{k+1}A^k - A^kB^k \quad (3.30)$$

donde el superíndice k representa el estado previo conocido. A partir de este punto el superíndice $k + 1$, el cual representa el nivel actual a obtener, será eliminado de las ecuaciones, y se entiende que todos los términos sin superíndice pertenecen al nivel actual $k + 1$. Por lo tanto, el sistema de ecuaciones hiperbólicas es expresado en su forma lineal como:

$$x_\xi x_\eta^k + x_\xi^k x_\eta - x_\xi^k x_\eta^k + y_\xi y_\eta^k + y_\xi^k y_\eta - y_\xi^k y_\eta^k = 0 \quad (3.31a)$$

$$x_\xi y_\eta^k + x_\xi^k y_\eta - x_\xi^k y_\eta^k - x_\eta y_\xi^k - x_\eta^k y_\xi + x_\eta^k y_\xi^k = F \quad (3.31b)$$

Aplicando el concepto de las ecuaciones 3.28 a las ecuaciones 3.31 podemos simplificar las últimas, quedando como

$$x_\xi x_\eta^k + x_\xi^k x_\eta + y_\xi y_\eta^k + y_\xi^k y_\eta = 0 \quad (3.32a)$$

$$x_\xi y_\eta^k + x_\xi^k y_\eta - x_\eta y_\xi^k - x_\eta^k y_\xi = F + F^k \quad (3.32b)$$

Este sistema de ecuaciones puede escribirse de forma compacta como

$$[A] R_\xi + [B] R_\eta = H \quad (3.33)$$

donde:

$$R = \begin{bmatrix} x \\ y \end{bmatrix} \quad A = \begin{bmatrix} x_\eta^k & y_\eta^k \\ y_\eta^k & -x_\eta^k \end{bmatrix} \quad B = \begin{bmatrix} x_\xi^k & y_\xi^k \\ -y_\xi^k & x_\xi^k \end{bmatrix} \quad H = \begin{bmatrix} 0 \\ F + F^k \end{bmatrix}$$

Por definición, el sistema de ecuaciones descrito por la ecuación 3.33 es hiperbólico si los eigenvalores de $[B]^{-1}[A]$ son reales. Se observa que

$$[B]^{-1} = \frac{1}{DN} \begin{bmatrix} x_\xi^k & -y_\xi^k \\ y_\xi^k & x_\xi^k \end{bmatrix}$$

por lo tanto

$$[C] = [B]^{-1}[A] = \begin{bmatrix} x_\xi^k x_\eta^k - y_\xi^k y_\eta^k & x_\xi^k y_\eta^k + x_\eta^k y_\xi^k \\ x_\xi^k y_\eta^k + x_\eta^k y_\xi^k & -(x_\xi^k x_\eta^k - y_\xi^k y_\eta^k) \end{bmatrix}$$

donde

$$DN = (x_\xi^k)^2 + (y_\xi^k)^2$$

Los eigenvalores de $[C]$ son

$$\lambda_{1,2} = \pm \sqrt{\frac{(x_\eta^k)^2 + (y_\eta^k)^2}{DN}}$$

los cuales cumplen con la condición de ser siempre reales (para un sistema hiperbólico) siempre que

$$DN = (x_\xi^k)^2 + (y_\xi^k)^2 \neq 0$$

Para obtener un sistema de ecuaciones algebráico, debemos aplicar el método de diferencias finitas. Para la sustitución de las derivadas parciales respecto al eje ξ se utiliza una aproximación central de segundo orden y para el caso de las derivadas parciales respecto al eje η se hará uso de aproximaciones de primer orden tipo “backwards”, resultando la ecuación en

$$[A] \frac{R_{i+1,j} - R_{i-1,j}}{2\Delta\xi} + [B] \frac{R_{i,j} - R_{i,j-1}}{\Delta\eta} = [H]_{i,j} \quad (3.34)$$

que al ser multiplicada por la matriz $[B]^{-1}$ y reacomodada queda cómo

$$\frac{R_{i,j} - R_{i,j-1}}{\Delta\eta} + [B]^{-1}[A] \frac{R_{i+1,j} - R_{i-1,j}}{2\Delta\xi} = [B]^{-1} H_{i,j} \quad (3.35)$$

en donde las matrices A y B son evaluadas en la línea $(j-1)$. Esta ecuación se reacomoda y se le asignan nuevos términos, con el propósito de compactar a la misma, dando como resultado

$$[AA] R_{i-1,j} + [BB] R_{i,j} + [CC] R_{i+1,j} = [DD]_{i,j} \quad (3.36)$$

donde

$$\begin{aligned}[AA] &= -\frac{1}{2\Delta\xi} [C]_{i,j-1} \\ [BB] &= \frac{1}{\Delta\eta} [I] \\ [CC] &= \frac{1}{2\Delta\xi} [C]_{i,j-1} \\ [DD] &= [B]_{i,j-1}^{-1} H_{i,j} + \frac{R_{i,j-1}}{\Delta\eta}\end{aligned}$$

una vez que se ha desarrollado la ecuación 3.36 para todas las instancias existentes a lo largo de i para un mismo nivel j , se obtiene un sistema de bloque tridiagonal

$$\left[\begin{array}{ccc} [BB]_2 & [CC]_2 & \\ [AA]_3 & [BB]_3 & [CC]_3 \\ \ddots & \ddots & \ddots \\ & \ddots & \ddots \\ & & \ddots \\ [AA]_{m-2} & [BB]_{m-2} & [CC]_{m-2} \\ [AA]_{m-1} & [BB]_{m-1} & \end{array} \right] \left[\begin{array}{c} R_2 \\ R_3 \\ \vdots \\ R_{m-2} \\ R_{m-1} \end{array} \right] = \left[\begin{array}{c} [DD]_2 \\ [DD]_3 \\ \vdots \\ [DD]_{m-2} \\ [DD]_{m-1} \end{array} \right] \quad (3.38)$$

los términos $[DD]_2$ y $[DD]_{m-1}$ son afectados por las condiciones de frontera de la siguiente manera

$$\begin{aligned}[DD]_2 &= [DD]_2 - [AA]_2 R_1 \\ [DD]_{m-1} &= [DD]_{m-1} - [CC]_{m-1} R_m\end{aligned}$$

Este sistema de ecuaciones se resuelve avanzando en dirección del eje η , siempre y cuando la distribución de punto en la superficie y en las fronteras sea proporcionada.

Los puntos en las fronteras deben ser libres de flotar, es decir, deben calcularse al final de cada iteración. Esto se logra aplicando la condición de ortogonalidad a las fronteras $i = 1$ e $i = m$ para actualizar dichos puntos.

Un algoritmo para la solución de sistemas de ecuaciones de bloque tridiagonal es proporcionado en el apéndice C

Generación de mallas mediante EDP parabólicas

El esquema elíptico desarrollado por Thompson et. al. [19] ha sido de los más utilizados en la generación de mallas estructuradas, sin embargo presenta ciertas desventajas que han hecho que se desarrolle otros enfoques y esquemas que sean más rápidos para obtener una solución y que consuman menos recursos computacionales. De este razonamiento surge el método propuesto por Steger y Chausse [20] el cual utiliza ecuaciones hiperbólicas para este propósito, entre las ventajas que dicho método presenta con respecto al esquema elíptico destaca el uso de un procedimiento de marcha lo que se traduce en un menor tiempo de ejecución del código dado que el esquema elíptico

utiliza métodos iterativos y el tiempo de ejecución de un método de marcha es del orden que toma a un método iterativo llevar a cabo una iteración. Por otro lado, el usar ecuaciones hiperbólicas tiene a su vez desventajas como son la propagación de discontinuidades que puedan presentarse en la frontera interna conforme la solución marcha de dentro hacia afuera, otro punto importante de desventaja es que a menudo se necesita introducir términos que representen una “viscosidad artificial” en la solución ya que las soluciones de este tipo de ecuaciones tienden a ser inestables dadas sus características matemáticas. Por último, no es posible definir una frontera externa.

En este trabajo se presenta el esquema desarrollado por Nakamura [21] y que fue retomado por Siladic [18], el cual propone un método de generación de mallas a través de la solución de ecuaciones diferenciales parciales parabólicas. El uso de ecuaciones parabólicas presenta ciertas ventajas, entre ellas destaca el uso de métodos de solución de marcha, similares a los utilizados en la solución de ecuaciones hiperbólicas, dado que es una solución de problemas con condiciones iniciales, esto se traduce de igual manera en un tiempo de cómputo bajo. Por otro lado, las ecuaciones parabólicas comparten mucho del comportamiento matemático que está presente en las ecuaciones elípticas, que para el caso particular de la generación de mallas se traduce en el hecho de poseer un efecto difusivo que permite suavizar cualquier discontinuidad presente en la frontera interna conforme el procedimiento marcha de dentro hacia afuera. Por último, se posee la capacidad de definir las condiciones de frontera impuestas en la frontera externa.

En resumen, lo explicado arriba, es decir, el desarrollo de un algoritmo de generación de mallas mediante ecuaciones parabólicas tiene dos puntos de vital importancia, el primero de ellos es el bajo tiempo de ejecución computacional que es requerido para lograr la solución del problema, que representa una pequeña fracción del tiempo demandado por el esquema elíptico. El segundo es la baja demanda de memoria para llevar a cabo los cálculos requerido, y que además también es sustancialmente menor que el requerido por un esquema elíptico.

Se propone el siguiente conjunto de ecuaciones

$$a(\xi, \eta)x_\eta = b(\xi, \eta)x_{\xi\xi} + c(\xi, \eta)V_x(\xi, \eta) + d(\xi, \eta) \quad (3.40a)$$

$$a(\xi, \eta)y_\eta = b(\xi, \eta)y_{\xi\xi} + c(\xi, \eta)V_y(\xi, \eta) + d(\xi, \eta) \quad (3.40b)$$

donde a, b y c pueden ser constantes o alguna función de (ξ, η) , los valores (x, y) representan las coordenadas en el dominio físico mientras que los valores (ξ, η) representan las coordenadas en el plano computacional, por último V_x y V_y son considerados términos fuente. En este esquema, la frontera externa se impone como una restricción sobre el comportamiento de la solución en la dirección j mientras que la frontera interna funge como condición inicial del problema. Las ecuaciones 3.40 son discretizadas mediante el método de diferencias finitas, las derivadas parciales con respecto al eje η son sustituidas por aproximaciones tipo “backwards” mientras que las derivadas con respecto al eje ξ son reemplazadas por aproximaciones centrales de segundo orden. Dadas las condiciones iniciales para x y y en $\eta = 0$, es decir, una vez proporcionada la nube de puntos que definen la frontera interna, se obtiene un sistema de bloque tridiagonal, el cual debe ser resuelto para cada valor de η contemplado.

Los valores iniciales son especificados como:

$$x(\xi, 0) = x_0(\xi) \quad (3.41a)$$

$$y(\xi, 0) = y_0(\xi) \quad (3.41b)$$

donde $x_0(\xi)$ y $y_0(\xi)$ las coordenadas de la superficie de la frontera, es decir, del perfil aerodinámico a analizar. Para entender el efecto de los términos fuente, se asigna un valor cero a las constantes b y d , a su vez $a = c = 1$, expresando las ecuaciones resultantes en términos de incrementos en lugar de términos diferenciales:

$$\Delta x = V_x(\xi, \eta) \Delta \eta \quad (3.42a)$$

$$\Delta y = V_y(\xi, \eta) \Delta \eta \quad (3.42b)$$

en las ecuaciones 3.42 se observa que conforme η aumenta, el cambio tanto en x como en y está determinado por V_x y V_y respectivamente, lo que implica que ambos términos fuente deben especificarse de tal modo que x y y aumenten en la dirección y magnitud deseados. Por otro lado, las segundas derivadas parciales $x_{\xi\xi}$ y $y_{\xi\xi}$ en las ecuaciones 3.40 tienen un efecto disipador para los intervalos en dirección ξ . Los valores de los términos fuente V_x y V_y pueden ser determinados mediante interpolación, ya sea polinomial o lineal, entre las fronteras interna y externa. Si se desea tener un control sobre la ortogonalidad del mallado, es posible la introducción de una frontera falsa para la frontera externa que permita lograr dicha característica, sumado a esto, es posible variar la distancia entre nodos en ambas direcciones ξ y η al discretizar las ecuaciones 3.40 sobre una malla no uniforme en el plano computacional.

En la mayoría de casos, y en general en la literatura encontrada en la generación de mallas, se asume una malla uniforme y en algunos casos se sugiere que el espacio entre nodos sea igual a la unidad, esto último con el objetivo de simplificar las ecuaciones. Sin embargo, estas consideraciones no son estrictamente necesarias sino que representan simplificaciones del problema, por lo que la generación de una malla cuasi uniforme en el plano computacional, y que después se utilice en el análisis de flujos como si de una malla uniforme se tratase, es también un enfoque válido.

En la figura 3.13 se muestra un ejemplo de malla no uniforme, donde ΔL y ΔR representan incrementos en la dirección i , a su vez ΔU y ΔD representan incrementos en términos de j para un punto (i, j) dado. Retomando lo expuesto en el capítulo 2, se pueden usar series de Taylor para aproximar los términos que incluyen derivadas en las ecuaciones gobernantes en el esquema parabólico, quedando de la siguiente manera:

$$f_x = \frac{f_{i+1,j} - f_{i,j}}{\Delta R} \quad y \quad f_x = \frac{f_{i,j} - f_{i-1,j}}{\Delta L}$$

$$f_y = \frac{f_{i,j+1} - f_{i,j}}{\Delta U} \quad y \quad f_y = \frac{f_{i,j} - f_{i,j-1}}{\Delta D}$$

$$f_{xx} = \frac{2}{\Delta R + \Delta L} \left(\frac{f_{i+1,j} - f_{i,j}}{\Delta R} - \frac{f_{i,j} - f_{i-1,j}}{\Delta L} \right)$$

$$f_{yy} = \frac{2}{\Delta D + \Delta U} \left(\frac{f_{i,j+1} - f_{i,j}}{\Delta U} - \frac{f_{i,j} - f_{i,j-1}}{\Delta D} \right)$$

$$f_{xy} = \frac{f_{i+1,j+1} - f_{i+1,j-1} - f_{i-1,j+1} + f_{i-1,j-1}}{(\Delta R + \Delta L)(\Delta D + \Delta U)}$$

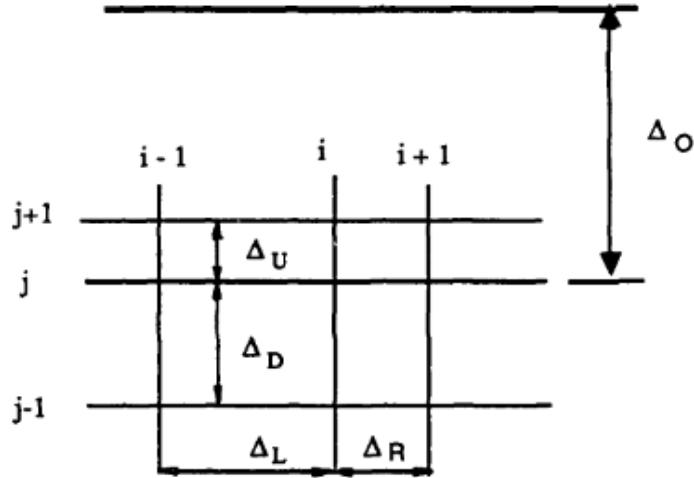


Figura 3.13: Generación de malla estructurada cuasi uniforme desarrollada mediante ecuaciones parabólicas [18]

con esto, las ecuaciones gobernantes 3.40 se discretizan de la siguiente manera

$$\frac{a(x_{i,j} - x_{i,j-1})}{\Delta D} = \frac{2b}{\Delta L + \Delta R} \left[\frac{x_{i+1,j} - x_{i,j}}{\Delta R} - \frac{x_{i,j} - x_{i-1,j}}{\Delta L} \right] + c \frac{XB_{i,J_{\max}} - x_{i,j}}{\Delta O} + d \quad (3.43a)$$

$$\frac{a(y_{i,j} - y_{i,j-1})}{\Delta D} = \frac{2b}{\Delta L + \Delta R} \left[\frac{y_{i+1,j} - y_{i,j}}{\Delta R} - \frac{y_{i,j} - y_{i-1,j}}{\Delta L} \right] + c \frac{YB_{i,J_{\max}} - y_{i,j}}{\Delta O} + d \quad (3.43b)$$

Para poder determinar los valores de los coeficientes a , b y c y para entender el método presente para el control de la densidad de mallado, se deben relacionar las ecuaciones gobernantes con un conjunto de ecuaciones conocidas en el plano computacional. Con este propósito es que se seleccionan las ecuaciones de Laplace en el modelo de generación de mallas mediante ecuaciones elípticas (ecuaciones 3.17), intentando obtener un método equivalente al obtenido en el esquema elíptico, en el que cualquier línea m en la dirección η sea forzada a moverse hacia una línea $m - 1$. En la figura 3.14 se muestra un representación de lo descrito, siendo para un punto (i, j) , F_{i-1} y F_i incrementos en la dirección ξ y a su vez g_{j-1} y g_j incrementos en la dirección η .

Las ecuaciones del modelo de generación de mallas elíptico (ecuaciones 3.17), bajo este nuevo esquema basado en la generación de una malla cuasi uniforme se discretizan en (ξ, η) dando como resultado

$$\begin{aligned} & \frac{2\alpha}{F_i + F_{i-1}} \left[\frac{s_{i-1,j} - s_{i,j}}{F_{i-1}} + \frac{s_{i+1,j} - s_{i,j}}{F_i} \right] + \beta \left[\frac{s_{i+1,j+1} - s_{i+1,j-1} - s_{i-1,j+1} + s_{i-1,j-1}}{(F_{i-1} + F_i)(g_{j-1} + g_j)} \right] \\ & + \frac{2\gamma}{g_j + g_{j-1}} \left[\frac{s_{i,j-1} - s_{i,j}}{g_{j-1}} + \frac{s_{i,j+1} - s_{i,j}}{g_j} \right] = 0 \end{aligned} \quad (3.44)$$

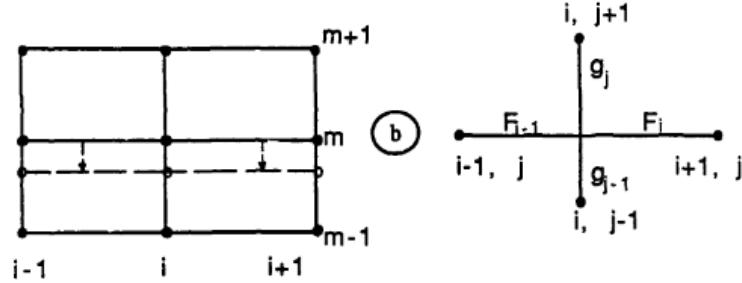


Figura 3.14: Generación de malla estructurada no uniforme desarrollada mediante ecuaciones elípticas [18]

A partir de la ecuación 3.44 se propone un procedimiento de marcha, reemplazando en dicha ecuación los valores $x_{i,j+1}$ y $y_{i,j+1}$ por valores conocidos $XO_{i,j+1}$ y $YO_{i,j+1}$ los cuales son obtenidos mediante interpolación lineal entre los valores de la frontera interna (perfil aerodinámico) y la frontera externa (tipología de la malla). La distancia g_j puede cambiarse gradualmente a lo largo de la malla para una mayor versatilidad en la generación de la ésta.

En el caso más sencillo, aquel en el que no interesa forzar la ortogonalidad en las fronteras, a las variables XO y YO se les asignan los valores $(XB_{i,J_{\max}}, YB_{i,J_{\max}})$ en la frontera externa. Reordenando los términos, y donde s representa cualquier coordenada x o y

$$\begin{aligned} & \frac{2\alpha}{F_i + F_{i-1}} \left[\frac{s_{i-1,j} - s_{i,j}}{F_{i-1}} + \frac{s_{i+1,j} - s_{i,j}}{F_i} \right] + \frac{2\gamma}{g_{j-1} + G_j} \left[\frac{-s_{i,j}}{g_{j-1}} - \frac{s_{i,j}}{G_j} \right] \\ &= -\beta \left[\frac{SO_{i+1,j+1} - SO_{i-1,j-1} - s_{i+1,j-1} + s_{i-1,j-1}}{(F_{i-1} + F_i)(g_{j-1} + G_j)} \right] - \frac{2\gamma}{g_{j-1} + G_j} \left[\frac{s_{i,j-1}}{g_{j-1}} + \frac{SO_{i,j+1}}{G_j} \right] \end{aligned} \quad (3.45)$$

donde

$$\begin{aligned} G_j &= g_j + g_{j+1} + \dots + g_{J_{\max}-1} = \eta_{\max} - \eta_j \\ g_{j-1} &= \eta_j - \eta_{j-1}; \quad F_i = \xi_{i+1} - \xi_i; \quad F_{i-1} = \xi_i - \xi_{i-1} \end{aligned}$$

G_j representa la distancia entre las líneas (i, j) e (i, JO) en el plano computacional, por lo tanto $SO_{i,j} = XO_{i,j}$ o $YO_{i,j}$. Si comparamos ambos modelos discutidos en esta sección, es decir el modelo parabólico (ecuaciones 3.43) y el modelo elíptico (ecuaciones 3.45) se puede establecer una clara relación, al colocar de un lado las incógnitas de cada ecuación, y del otro todos los valores conocidos, es decir

$$\begin{aligned} & - \left(\frac{b}{\Delta L} + \frac{b}{\Delta R} \right) \frac{2s_{i,j}}{\Delta L + \Delta R} - \left(\frac{a}{\Delta D} + \frac{c}{\Delta O} \right) s_{i,j} + \frac{b}{\Delta R} \frac{2s_{i-1,j}}{\Delta R + \Delta L} + \frac{b}{\Delta R} \frac{2s_{i+1,j}}{\Delta R + \Delta L} \\ &= -c \frac{SB_{i,J_{\max}}}{\Delta O} - a \frac{s_{i,j-1}}{\Delta D} - d \end{aligned} \quad (3.46)$$

$$\begin{aligned}
& - \left(\frac{\alpha}{F_{i-1}} + \frac{\alpha}{F_i} \right) \frac{2s_{i,j}}{F_i + F_{i-1}} - \left(\frac{\gamma}{g_{j-1}} + \frac{\gamma}{G_j} \right) \frac{2s_{i,j}}{G_j + g_{j-1}} + \frac{\alpha}{F_{i-1}} \frac{2s_{i-1,j}}{F_i + F_{i-1}} + \frac{\alpha}{F_i} \frac{2s_{i,j}}{F_i + F_{i-1}} \\
& = - \frac{2\gamma}{g_{j-1} + G_j} \left[\frac{SO_{i,j+1}}{G_j} + \frac{s_{i,j-1}}{g_{j-1}} \right] - \beta \left[\frac{SO_{i+1,j+1} - SO_{i-1,j+1} - s_{i+1,j-1} + s_{i-1,j-1}}{(F_i + F_{i-1})(g_{j-1} + G_j)} \right]
\end{aligned} \tag{3.47}$$

de este reacomodo se puede observar que ambas ecuaciones tienen la misma forma, por lo que es posible establecer una relación entre sus coeficientes

$$b = \alpha$$

$$a = c = \frac{2\gamma}{G_j + g_{j-1}}$$

$$d = \beta \left[\frac{SO_{i+1,j+1} - SO_{i-1,j+1} - s_{i+1,j-1} + s_{i-1,j-1}}{(F_i + F_{i-1})(G_j + g_{j-1})} \right]$$

donde α , β y γ son los coeficientes de las ecuaciones 3.17. El sistema representado por la ecuación 3.45 representa un sistema tridiagonal de matrices 2x2 el cual se desarrolla del siguiente modo

$$A = \frac{2\alpha}{F_{i-1}(F_i + F_{i-1})} ; \quad C = \frac{2\alpha}{F_i(F_i + F_{i-1})}$$

$$B = \frac{-2\alpha}{F_i + F_{i-1}} \left(\frac{1}{F_i} + \frac{1}{F_{i-1}} \right) - \frac{2\gamma}{G_j + g_{j-1}} \left(\frac{1}{G_j} + \frac{1}{g_{j-1}} \right)$$

$$D_x = -\beta \frac{XO_{i+1,j+1} - XO_{i-1,j+1} - x_{i+1,j-1} + x_{i-1,j-1}}{(F_i + F_{i-1})(G_j + g_{j-1})} - \frac{2\gamma}{G_j + g_{j-1}} \left[\frac{x_{i,j-1}}{g_{j-1}} + \frac{XO_{i,j+1}}{G_j} \right]$$

$$D_y = -\beta \frac{YO_{i+1,j+1} - YO_{i-1,j+1} - y_{i+1,j-1} + y_{i-1,j-1}}{(F_i + F_{i-1})(G_j + g_{j-1})} - \frac{2\gamma}{G_j + g_{j-1}} \left[\frac{y_{i,j-1}}{g_{j-1}} + \frac{YO_{i,j+1}}{G_j} \right]$$

finalmente, las ecuaciones gobernantes del sistema parabólico pueden expresarse como:

$$Ax_{i-1,j} + Bx_{i,j} + Cx_{i+1,j} = D_x \tag{3.49a}$$

$$Ay_{i-1,j} + By_{i,j} + Cy_{i+1,j} = D_y \tag{3.49b}$$

Las ecuaciones 3.49 se resuelven simultáneamente para todos los puntos de la malla para el mismo nivel η (línea j) resolviendo el sistema tridiagonal para cada $x_{i,j}$ y $y_{i,j}$. La solución comienza en $j = 1$ (justo después de la frontera interna) y finaliza en $j = J_{\max} - 2$ (un nivel antes de la frontera externa). Los coeficientes α , β y γ se calculan usando las coordenadas de puntos adyacentes que ya se han generado, dichos valores son sustituidos en las aproximaciones por diferencias finitas utilizadas para reemplazar las primeras derivadas parciales, las cuales para este sistema quedan expresadas como

$$x_\xi = \frac{x_{i+1,j-1} - x_{i-1,j-1}}{F_i + F_{i-1}} \quad (3.50a)$$

$$y_\xi = \frac{y_{i+1,j-1} - y_{i-1,j-1}}{F_i + F_{i-1}} \quad (3.50b)$$

$$x_\eta = \frac{XO_{i,j+1} - x_{i,j-1}}{g_{j-1} + G_j} \quad (3.50c)$$

$$y_\eta = \frac{YO_{i,j+1} - y_{i,j-1}}{g_{j-1} + G_j} \quad (3.50d)$$

Los términos no uniformes en la separación de la malla descritos arriba tienen un efecto similar al control de la densidad de mallado mediante funciones exponenciales presentado por Thompson, Thames y Mastin [19] para la generación de mallas mediante el esquema elíptico.

Ortogonalidad de la Malla

“Al analizar y resolver flujos viscosos o capas límite, la ortogonalidad de las líneas de la malla en las cercanías a las superficies de cuerpos es deseable para representar todas las derivadas normales de manera simple y precisa” [18]. Si se desea lograr esto, se debe tener la capacidad de controlar la orientación de las líneas, lo cual puede lograrse introduciendo una frontera externa modificada lo que permite determinar los valores de los términos fuente. Es decir, si la frontera externa modificada se posiciona de manera que sus puntos se localizan perpendicularmente a sus correspondientes sobre la superficie del cuerpo, las líneas se alejarán del cuerpo de manera ortogonal. Esto es mejor ilustrado en la figura 3.15.

Si nos ubicamos en un punto (i, j) con sus correspondientes coordenadas $(XO_{i,j+1}, YO_{i,j+1})$, se observa de la figura 3.15 una recta AA que es perpendicular a la superficie de la frontera interna la cual nace del punto $(i, 0)$, punto que corresponde a la misma línea $\xi = cte$ del punto en el que nos ubicamos. También podemos observar en la figura un arco de circunferencia CC , dicho arco tiene su centro en el punto $(i, 0)$ y pasa por las coordenadas $(XB_{i,J_{\max}}, YB_{i,J_{\max}})$ de la frontera externa no modificada. La intersección de la línea AA y del arco CC representa los valores dados para las coordenadas $(XO_{i,j+1}, YO_{i,j+1})$ con el uso de una frontera modificada.

3.7. Calidad de las Mallas

La calidad de la malla es un tema de vital importancia en la generación de las mismas, ya que representa un indicativo de la confiabilidad y la usabilidad de la malla. Por lo tanto es necesario desarrollar métodos que permitan analizar dicha característica en las mallas generadas con el fin de poder detectar y en su caso, modificar, las anomalías y errores que se presenten.

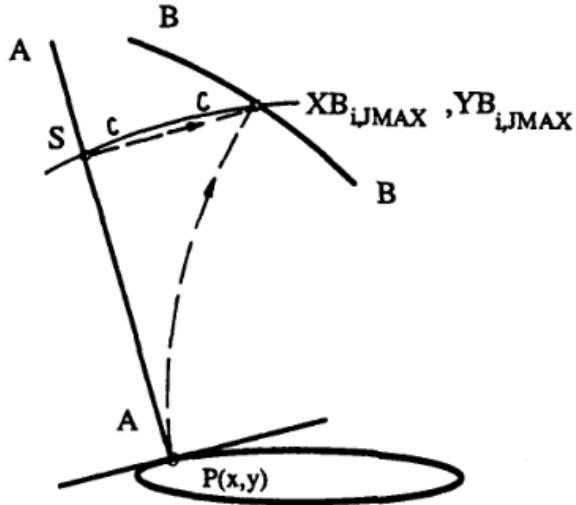


Figura 3.15: Control de ortogonalidad en la generación de mallas por ecuaciones parabólicas [18]

Esto conlleva a la necesidad de determinar las características correctas con las cuales se evaluará la calidad, y entender como los resultados obtenidos se relacionan con los errores que puedan presentarse durante las simulaciones de flujos.

Las características que más suelen evaluarse son:

- Skewness
- Alargamiento
- Torsión
- Aspect Ratio
- Volúmen de Celda
- Densidad de Nodos

En el caso de mallas estructuradas, dados los conceptos con los que se lleva a cabo su generación, se pueden definir muchas de las características de la malla tomando la transformación de coordenadas como propiedades de las curvas, superficies y volúmenes. Estos valores se determinan mediante los elemetos del tensor de la métrica de la transformación y las derivadas de éste. [15]

Aspect Ratio

El aspect ratio es un valor característico de cada celda, mide que tan alejada está la forma de la celda de una figura que sirve de base para la comparación. Para mallas bidimensionales, las celdas pueden ser triangulares o cuadrilaterales, en el caso de celdas triangulares la figura base suele ser un triángulo equilátero, mientras que para celdas de cuatro lados, algunas fuentes toman como figura base el rombo, mientras que otras fuentes toman como base el cuadrado.

Por ejemplo, Liseikin [15], presenta un método que toma como referencia el rombo, dicho método viene dado por la ecuación

$$Q_{as}^l = \frac{g_{ii}}{g_{jj}} + \frac{g_{jj}}{g_{ii}} = \frac{(g_{ii} + g_{jj})^2}{g_{ii}g_{jj}} - 2 \quad (3.51)$$

esta ecuación presenta la desigualdad $Q_{as}^l \geq 2$ la cual se vuelve una igualdad si, y solo si, $g_{ii} = g_{jj}$, lo cual se traduce a que el paralelogramo es un rombo. Por lo tanto, esta ecuación indica qué tan alejada está la forma de la celda de un rombo.

Por su parte ANSYS, uno de los programas más reconocidos en el ámbito de la ingeniería asistida por computadora (CAE por sus siglas en inglés), considera el aspect ratio como una medida del alargamiento de la celda, y lo calcula como la relación entre el valor máximo y el valor mínimo de alguno de diferentes valores como pueden ser: distancia entre el centroide de la celda a los nodos, distancia entre el centroide de la celda y los centroides de las caras que la forman. La figura 3.16 es una representación gráfica de este método.

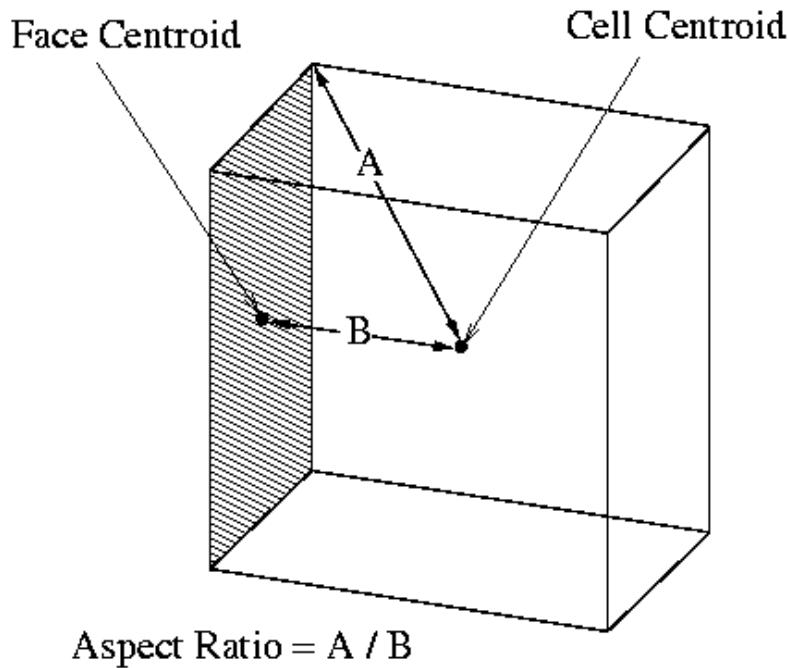


Figura 3.16: Cálculo de aspect ratio para una celda tridimensional en ANSYS

Otro método que se debe resaltar es el empleado en la librería “The Verdict Quality Library”, esta librería contiene rutinas para evaluar la calidad geométrica de diferentes tipos de celdas como pueden ser triangulos, cuadrilateros, tetraedros y hexaedros. Para el cálculo del aspect ratio de celdas formadas por cuadrilateros utiliza la siguiente fórmula

$$q = \frac{L_{\max} (L_0 + L_1 + L_2 + L_3)}{4A} \quad (3.52)$$

donde

$$L_{\max} = \max (L_0, L_1, L_2, L_3)$$

$$A = \frac{1}{2} |\vec{L}_0 \times \vec{L}_1| + \frac{1}{2} |\vec{L}_2 \times \vec{L}_3|$$

la figura 3.17 muestra graficamente los elementos de una celda formada por un cuadrilatero, con los cuales se calculan las métricas propuestas por esta librería.

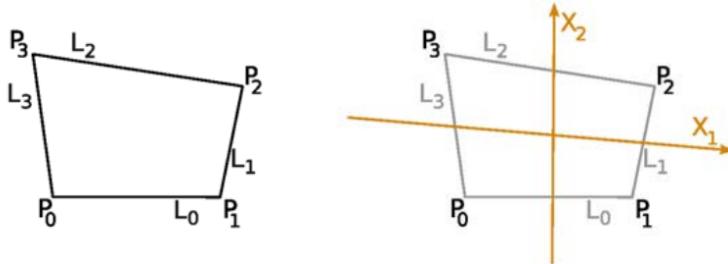


Figura 3.17: Estructura de celda cuadrilátera en Verdict. Del lado izquierdo se muestran los nodos y los elementos que la forman. Del lado derecho se muestran los ejes principales de la misma [22]

Esta librería y sus métodos son utilizados por diversos programas, entre ellos destaca el programa SALOME, el cual es un software de código abierto desarrollado para las etapas de preproceso y postproceso en análisis computacionales de ingeniería.

3.7.1. Skewness

El “skew” puede interpretarse geométricamente como el sesgo de la celda, es decir, se interpreta como la desviación de la simetría, dicha simetría depende del tipo de celda que contenga la malla, por ejemplo para celdas triangulares la simetría corresponde a una celda equilátera la cual por lo tanto se considera no sesgada. En el caso en el que la malla está formada por celdas cuadriláterales, se considera una celda no sesgada a una celda cuadrada o rectangular.

Cengel [6] presenta una fórmula para calcular el “skew” de la celda mediante la siguiente formula

$$Q_{EAS} = MAX \left(\frac{\theta_{max} - \theta_{igual}}{180^\circ - \theta_{igual}}, \frac{\theta_{igual} - \theta_{min}}{\theta_{igual}} \right) \quad (3.54)$$

donde θ_{min} y θ_{max} son los ángulos mínimo y máximo entre los lados de las celdas, mientras que θ_{igual} es el ángulo entre los lados de una celda ideal, es decir no sesgada, el cual para celdas cuadriláteras es $\theta_{igual} = 90^\circ$ y para celdas triangulares corresponde a $\theta_{igual} = 60^\circ$.

Los resultados de esta fórmula arrojan valores entre 0 y 1, donde 0 representa una celda sin sesgo, es decir que la celda es un cuadrado o un rectángulo si la malla se compone de cuadriláteros. Celdas muy distorsionadas o alejadas de dicha simetría tienden a tener valores cercanos a la unidad.

Por su parte, “The Verdict Quality Library” presenta su propio método para evaluar el “skew” de una celda. Este método evalúa el ángulo existente entre los ejes principales de la celda y de manera más precisa, evalúa el valor del coseno de dicho ángulo. La figura 3.17 muestra gráficamente los elementos que conforman una celda.

El método usado por la librería es el siguiente. Primero se deben normalizar los ejes principales \vec{X}_1 y \vec{X}_2 , los cuales están definidos por

$$\vec{X}_1 = (\vec{P}_1 - \vec{P}_0) + (\vec{P}_2 - \vec{P}_3) \quad (3.55a)$$

$$\vec{X}_2 = (\vec{P}_2 - \vec{P}_1) + (\vec{P}_3 - \vec{P}_0) \quad (3.55b)$$

y normalizados se definen como

$$\hat{X}_1 = \frac{\vec{X}_1}{\|\vec{X}_1\|} \quad (3.56a)$$

$$\hat{X}_2 = \frac{\vec{X}_2}{\|\vec{X}_2\|} \quad (3.56b)$$

Con esto, el “skew” está dado por

$$q = |\hat{X}_1 \cdot \hat{X}_2| \quad (3.57)$$

En este método el rango de valores también se encuentra entre 0 y 1, sin embargo a diferencia del método propuesto por Cengel, en éste una celda sin cesgo presenta un valor $q = 1$, mientras que los valores de q para celdas muy sesgadas tienden a 0.

Capítulo 4

Análisis de Flujo Potencial

El modelo de flujo potencial es la más simple consideración de flujo no viscoso en tomar en cuenta los efectos de compresibilidad.

Para el caso de flujos externos, en este caso el flujo de una corriente de aire alrededor de un perfil aerodinámico, existen ciertas consideraciones que permiten tratar el flujo como no viscoso e irrotacional.

Se sabe que la condición de irrotacionalidad requiere la ausencia de viscosidad, así como de gradientes de velocidad en la dirección normal al flujo. Teóricamente estas condiciones en el flujo son imposibles de conseguir, sin embargo en flujos de fluidos con valores relativamente pequeños de viscosidad (como el agua y el aire) los efectos de la capa límite suceden en una sección en teoría demasiado pequeña, con lo que dicha sección puede ser despreciada durante el análisis. En cuanto al resto del flujo, la condición de irrotacionalidad, así como la no consideración de los efectos viscosos, son simplificaciones válidas.

Como ya se mencionó, la condición principal que debe tener el flujo para poder ser considerado potencial y no viscoso es la irrotacionalidad. De acuerdo al teorema de Kelvin, si el flujo es irrotacional en un momento, continuara siendo irrotacional y por lo tanto también será isentrópico.

Estas consideraciones teóricas llevan consigo una serie de simplificaciones a las ecuaciones de Navier Stokes que facilitan el análisis de diversos flujos. En primera instancia, al decirse que el flujo es no viscoso se debe despreciar los efectos de los esfuerzos cortantes, dicha simplificación nos permite trabajar con las ecuaciones de Euler. Si además de esto, se toma como irrotacional el flujo, se obtienen las ecuaciones de Bernoulli.

Para flujos no viscosos, la función de potencial puede definirse como:

$$\vec{v} = \vec{\nabla}\phi \quad (4.1)$$

La forma conservativa de la ecuación del potencial puede obtenerse a partir de la ecuación de la continuidad.

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \vec{\nabla} \phi) = 0 \quad (4.2)$$

La ecuación de momento y energía pueden reducirse a la siguiente ecuación para la entalpía de estancamiento:

$$\frac{\partial \phi}{\partial t} + H = H_0 \quad (4.3)$$

donde H_0 representa la entalpía de estancamiento y es un valor constante para todo el fluido.

La densidad es una función tanto de $\vec{\nabla}\phi$ como de $\frac{\partial \phi}{\partial t}$, que para el caso de un gas perfecto (dado que el flujo potencial se considera isentrópico) con valores definidos para la densidad y entalpía de estancamiento ρ_0 y H_0 se escribe como:

$$\frac{\rho}{\rho_0} = \left[1 - \frac{(\vec{\nabla}\phi)^2}{2H_0} - \frac{\frac{\partial \phi}{\partial t}}{H_0} \right]^{\frac{1}{\gamma-1}} \quad (4.4)$$

Para el caso de un flujo estacionario la ecuación del flujo potencial resulta:

$$\vec{\nabla} \cdot (\rho \vec{\nabla}\phi) = 0 \quad (4.5)$$

mientras que la ecuación de la energía, a través de las relaciones para gas isentrópico, queda como

$$H \equiv h + \frac{\vec{v}^2}{2} = H_0 \quad (4.6)$$

Dada una transformación de sistemas coordenados como la descrita en capítulos anteriores:

$$\begin{aligned} \xi &= \xi(x, y) \\ \eta &= \eta(x, y) \end{aligned}$$

la ecuación de flujo potencial puede ser escrita como

$$\frac{\partial}{\partial t} \left(\frac{\rho}{J} \right) + \frac{\partial}{\partial \xi} \left(\rho \frac{U}{J} \right) + \frac{\partial}{\partial \eta} \left(\rho \frac{V}{J} \right) = 0 \quad (4.7)$$

Las componentes de la velocidad en el dominio computacional U y V pueden definirse en función de las componentes del sistema cartesiano u y v como:

$$U = \xi_x \phi_x + \xi_y \phi_y = \xi_x u + \xi_y v \quad (4.8a)$$

$$V = \eta_x \phi_x + \eta_y \phi_y = \eta_x u + \eta_y v \quad (4.8b)$$

A partir de las ecuaciones 4.1 y 4.5 se puede obtener una ecuación para flujo potencial estacionario

$$\vec{\nabla} \cdot (\rho \vec{v}) = \frac{\partial}{\partial \xi} \left[(g^{11} \phi_\xi + g^{12} \phi_\eta) \frac{\rho}{J} \right] + \frac{\partial}{\partial \eta} \left[(g^{21} \phi_\xi + g^{22} \phi_\eta) \frac{\rho}{J} \right] = 0 \quad (4.9)$$

ya que también se tiene:

$$U = g^{11} \phi_\xi + g^{12} \phi_\eta \quad (4.10)$$

$$V = g^{21} \phi_\xi + g^{22} \phi_\eta \quad (4.11)$$

A su vez la matriz del tensor de la transformación $[g]$ tiene las siguientes componentes

$$g^{11} = \xi_x^2 + \xi_y^2 \quad (4.12a)$$

$$g^{12} = g^{21} = \xi_x \eta_x + \xi_y \eta_y \quad (4.12b)$$

$$g^{22} = \eta_x^2 + \eta_y^2 \quad (4.12c)$$

En la mayoría de los flujos de análisis práctico, es generalmente necesario trabajar con la transformación inversa:

$$x = x(\xi, \eta) \quad (4.13a)$$

$$y = y(\xi, \eta) \quad (4.13b)$$

la cual es obtenida a través de las transformaciones

$$\xi_x = J y_\eta \quad \xi_y = -J x_\eta \quad (4.14)$$

$$\eta_x = -J y_\xi \quad \eta_y = J x_\xi \quad (4.15)$$

con el Jacobiano

$$J = \frac{1}{x_\xi y_\eta - x_\eta y_\xi} \quad (4.16)$$

4.1. Condiciones de Frontera

Para que una simulación se lleve a cabo es necesario establecer un dominio computacional, el cual contará con una o más fronteras Γ , sobre dichas fronteras se deben especificar las correspondientes condiciones de frontera que dotarán de sentido físico a la simulación, es decir, para este caso se deben especificar condiciones de frontera que coincidan con la teoría de flujo potencial.

4.1.1. Condiciones en las lejanías

En la frontera externa el campo de fluido se asume como conocido. Para casos de flujos externos, como lo es el caso de un perfil inmerso en un flujo uniforme con velocidad \vec{V}_∞ , el flujo potencial está definido mediante:

$$\phi = \vec{V}_\infty \cdot \vec{x} + \phi_0 \quad (4.17)$$

donde ϕ_0 es una constante arbitraria y \vec{x} representa la distancia a un punto de la frontera con respecto a una referencia establecida previamente.

Para cuerpos que producen sustentación con una circulación Γ_B , se debe tomar en cuenta la contribución de dicha circulación al flujo potencial en una distancia lejana. En el caso de perfil bidimensionales esto se logra mediante la adición de un vórtice, corregido por efectos de compresibilidad:

$$\phi_\infty = \vec{V}_\infty \cdot \vec{x} + \frac{\Gamma_B}{2\pi} \tan^{-1} \left[\sqrt{1 - M_\infty^2} \tan(\theta - \alpha_\infty) \right] + \phi_0 \quad (4.18)$$

donde θ es la posición angular de un punto en la lejanía, Γ_B es la circulación y M_∞ es el número de Mach correspondiente a la velocidad de la corriente libre \vec{V}_∞ con un ángulo de incidencia α_∞ .

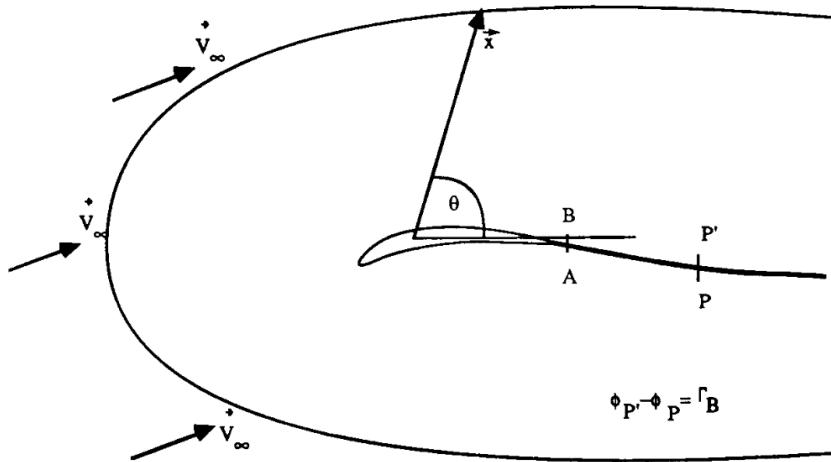


Figura 4.1: Condiciones de frontera de flujo potencial para perfil aerodinámico bidimensional [24]

4.1.2. Condiciones para perfiles aerodinámicos

Circulación y condición de Kutta

La condición de Kutta implica que en un flujo no viscoso e incompresible alrededor de un perfil alar, el fluido sale del borde de salida de manera suave.

Supongase el flujo sobre un perfil aerodinámico, en el momento en que el flujo ha comenzado así como en los primeros instantes, el fluido intentará rodear el borde de salida, desde el lado inferior al superior, esto implica la existencia de un punto de estancamiento en el extrados del perfil. Sin embargo, este comportamiento no es sostenible en un flujo real por mucho tiempo. En realidad conforme el flujo termina de desarrollarse hasta llegar a un estado estacionario, dicho punto de estancamiento va recorriendose hacia la parte posterior hasta llegar al borde de salida, a partir de que se logra esta condición se cumple la condición de Kutta, es decir el fluido deja el borde de salida de manera suave.

La figura 4.2 muestra este comportamiento, en la subfigura 4.2.1 se ve el instante inicial y se evidencia la presencia de un punto de estancamiento en el extrados del perfil. Conforme se avanza en los cuadros se observa como dicho punto de estancamiento se desplaza hacia la sección posterior del perfil hasta llegar al cuadro 4.2.6 donde se ha alcanzado el flujo estacionario. Es en este último cuadro donde se observa el cumplimiento de la condición de Kutta.

En resumen lo que la condición de Kutta establece es que la circulación alrededor de un perfil es la adecuada justo para asegurar que el flujo deja de manera suave el borde de salida. [25]

Condiciones de frontera

Los perfiles aerodinámicos requieren la presencia de una circulación cuya intensidad es definida por la condición de Kutta. El origen de dicha circulación se da a partir del inicio del flujo alrededor del perfil, y tiene explicación gracias al teorema de Kelvin de la circulación, el cual dice que la razón de cambio de la circulación con respecto al tiempo alrededor de una curva cerrada conformada por

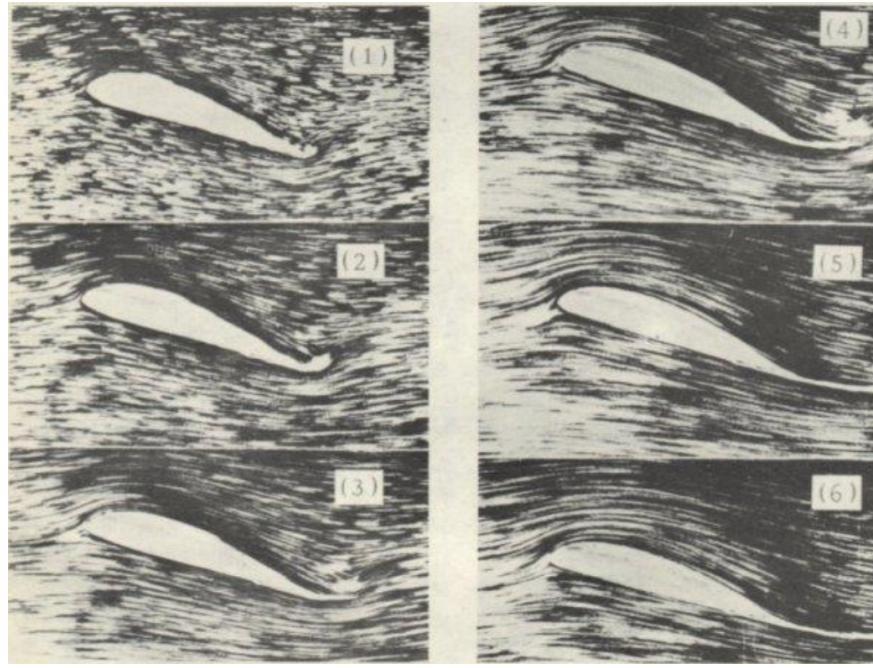


Figura 4.2: Desarrollo de flujo estacionario sobre un perfil aerodinámico, desde el reposo hasta una velocidad estacionaria.

el mismo fluido es igual a cero

$$\frac{D\Gamma}{Dt} = 0 \quad (4.19)$$

En un instante inicial, cuando el flujo alrededor del perfil está en reposo y su velocidad es igual a cero, la circulación alrededor de una curva arbitraria que rodee al perfil también será cero, tal como se muestra en la figura 4.3.

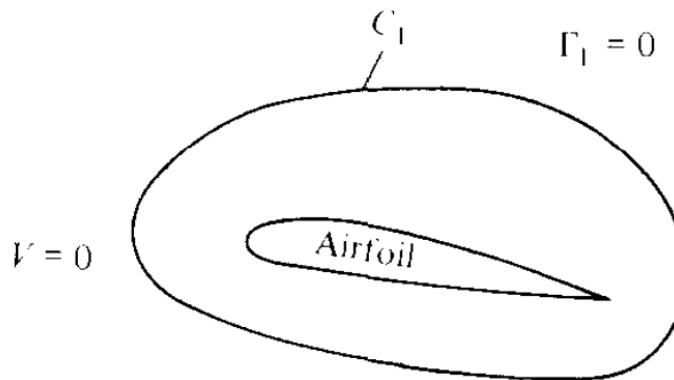


Figura 4.3: Fluido en reposo relativo a un perfil alar [25]

Al pasar los primeros instantes el fluido intentará rodear el borde de salida del perfil con lo cual se formará una región con grandes gradientes de velocidad y vorticidad en el borde de salida. Esta región es característica de las partículas que la forman con lo cual se comienza a desplazar

corriente abajo hasta que la condición de Kutta se cumple, durante este proceso esta región adopta la forma de un vórtice al cual se le conoce como vórtice de inicio.

Una vez lograda la condición de Kutta, el vórtice de inicio ya no está presente en el perfil sino que continuará alejándose corriente abajo conforme el tiempo avanza. Sin embargo, tal como se muestra en la figura 4.4, la curva que conformaba en el reposo al fluido alrededor del perfil se ha deformado, y ahora contiene además al vórtice de inicio. Se sabe gracias al teorema de Kelvin que la circulación alrededor de la curva C_2 debe ser cero, pero si esta curva la dividimos en C_3 y C_4 conteniendo al vórtice y al perfil respectivamente, observamos que la circulación alrededor de C_3 tendrá un valor finito debido al vórtice y por lo tanto alrededor de la curva C_4 debe existir una circulación de igual magnitud en sentido contrario, con lo que se logra que alrededor de la curva C_2 la circulación sea cero.

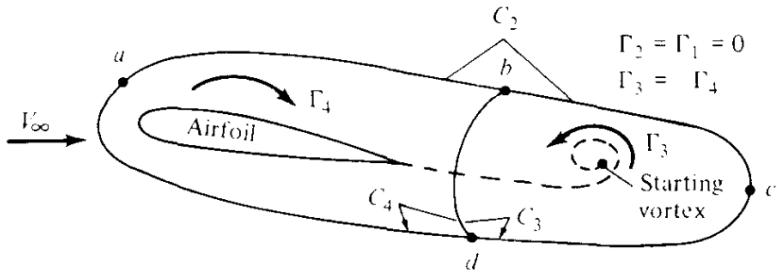


Figura 4.4: Flujo alrededor de perfil momentos después [25]

En simulaciones computacionales un corte debe definirse, a lo largo del cual existirá una discontinuidad del potencial dado por

$$\phi_{P'} - \phi_P = \Gamma_B = \phi_B - \phi_A \quad (4.20)$$

con lo cual se logra tomar en cuenta los efectos del vórtice de inicio en los resultados.

El valor de la circulación debe ser actualizado en cada iteración del proceso de solución, esto se logra imponiendo velocidades o presiones iguales en ambos lados del borde de salida.

4.2. Discretización para flujo subsonico estacionario

La ecuación de flujo potencial en su variable estacionaria para flujo subsonico es una ecuación diferencial parcial elíptica. Esto presenta la facilidad de utilizar diferencias finitas como método de discretización del problema, sin embargo es importante recalcar que este no es el único método válido y que también es posible llevar a cabo la discretización mediante los métodos de volúmenes y elementos finitos.

Ya que los flujos subsónicos presentan generalmente un comportamiento suave es posible realizar las simulaciones con mallas que tengan una baja densidad de celdas, a excepción ciertas zonas donde se generan grandes gradientes como lo son esquinas, y bordes de ataque y de salida de perfiles alares, entre otros.

Sin embargo la discretización de la ecuación de flujo potencial supone un esfuerzo mayor, debido a que a partir de la malla generada previamente se deben calcular otras dos mallas escalonadas en los puntos medios. Para una malla con puntos (i, j) donde $i = 1, \dots, M$ y $j = 1, \dots, N$ se tendrán puntos escalonados $(i \pm \frac{1}{2}, j \pm \frac{1}{2})$. En los puntos pertenecientes a la malla original se calcula la función de potencial, mientras que en las mallas intercaladas se calculan tanto la densidad como la velocidad.

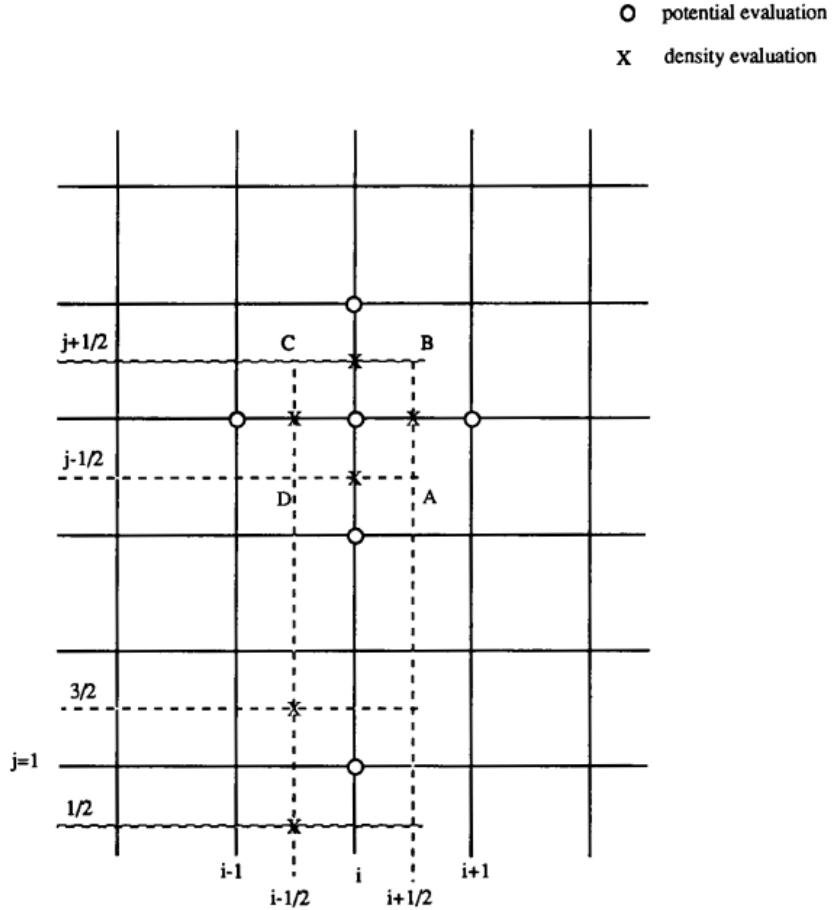


Figura 4.5: Malla original y mallas escalonadas [24]

La figura 4.5 es una representación de la localización de los puntos de las mallas intercaladas. De dicha figura se puede deducir la existencia de dos mallas escalonadas, una malla vertical y otra horizontal. En la figura se observa una “molecula” delimitada por los puntos $ABCD$, sobre dicho contorno se encuentran marcados con una X los puntos de intersección de las mallas escalonadas con la malla original, es importante resaltar que los puntos $ABCD$ no forman parte de ninguna de las mallas y su único propósito en la imagen es delimitar una partícula.

Si a la ecuación 4.7 se le eliminan los términos relacionados con el cambio respecto al tiempo, dado que se está calculando flujo estacionario, y además utilizamos una diferencia de tipo central para los términos de la densidad y las velocidades se obtiene

$$\frac{(\rho_J^U)_{i+\frac{1}{2},j} - (\rho_J^U)_{i-\frac{1}{2},j}}{2\Delta\xi} + \frac{(\rho_J^V)_{i,j+\frac{1}{2}} - (\rho_J^V)_{i,j-\frac{1}{2}}}{2\Delta\eta} = 0 \quad (4.21)$$

la cual se simplifica si se toman $\Delta\xi = \Delta\eta = 1$, quedando

$$\left(\rho \frac{U}{J}\right)_{i+\frac{1}{2},j} - \left(\rho \frac{U}{J}\right)_{i-\frac{1}{2},j} + \left(\rho \frac{V}{J}\right)_{i,j+\frac{1}{2}} - \left(\rho \frac{V}{J}\right)_{i,j-\frac{1}{2}} = 0 \quad (4.22)$$

A partir de las ecuaciones 4.10 y 4.11 (mostradas nuevamente por conveniencia) se obtienen los valores correspondientes a las velocidades mediante la discretización de las mismas por aproximaciones centrales.

$$\begin{aligned} U &= g^{11}\phi_\xi + g^{12}\phi_\eta \\ V &= g^{21}\phi_\xi + g^{22}\phi_\eta \end{aligned}$$

En la malla intercalada vertical, es decir donde la variable ξ es constante, la variación del potencial respecto a ξ se obtiene calculando por aproximaciones centrales de dicha variación en los puntos adyacentes pertenecientes a la malla original y calculando la media de las mismas. Para la variación con respecto a η se calcula únicamente una aproximación central entre los puntos adyacentes. Esto es, para un punto $(i, j + \frac{1}{2})$ las variaciones con respecto a ξ se obtienen promediando las aproximaciones centrales con respecto a ξ de los puntos (i, j) e $(i, j + 1)$; para las variaciones respecto a η se hace una aproximación central entre los puntos (i, j) e $(i, j + 1)$.

Por otro lado, en la malla intercalada horizontal, donde η es constante, la variación del potencial respecto a ξ es simplemente la aproximación central de dicha variación a partir de los puntos adyacentes de la malla original. En el caso de la variación del potencial respecto a η , primero se debe calcular dicha variación en los puntos adyacentes de la malla y posteriormente obtener la media de las mismas. Es decir, para un punto $(i + \frac{1}{2}, j)$, las variaciones con respecto a ξ son una aproximación central entre los puntos (i, j) e $(i + 1, j)$ y para las variaciones con respecto a η se deben calcular las aproximaciones centrales con respecto a η en los puntos (i, j) e $(i + 1, j)$ para posteriormente calcular la media.

La discretización de las velocidades, considerando $\Delta\xi = \Delta\eta = 1$, finalmente queda:

$$U_{i+\frac{1}{2},j} = g_{i+\frac{1}{2},j}^{11}(\phi_{i+1,j} - \phi_{i,j}) + \frac{1}{4}g_{i+\frac{1}{2},j}^{12}(\phi_{i+1,j+1} - \phi_{i+1,j-1} + \phi_{i,j+1} - \phi_{i,j-1}) \quad (4.23a)$$

$$U_{i-\frac{1}{2},j} = g_{i-\frac{1}{2},j}^{11}(\phi_{i,j} - \phi_{i-1,j}) + \frac{1}{4}g_{i-\frac{1}{2},j}^{12}(\phi_{i,j+1} - \phi_{i,j-1} + \phi_{i-1,j+1} - \phi_{i-1,j-1}) \quad (4.23b)$$

$$U_{i,j+\frac{1}{2}} = \frac{1}{4}g_{i,j+\frac{1}{2}}^{11}(\phi_{i+1,j+1} - \phi_{i-1,j+1} + \phi_{i+1,j} - \phi_{i-1,j}) + g_{i,j+\frac{1}{2}}^{12}(\phi_{i,j+1} - \phi_{i,j}) \quad (4.23c)$$

$$U_{i,j-\frac{1}{2}} = \frac{1}{4}g_{i,j-\frac{1}{2}}^{11}(\phi_{i+1,j} - \phi_{i-1,j} + \phi_{i+1,j-1} - \phi_{i-1,j-1}) + g_{i,j-\frac{1}{2}}^{12}(\phi_{i,j} - \phi_{i,j-1}) \quad (4.23d)$$

$$V_{i+\frac{1}{2},j} = g_{i+\frac{1}{2},j}^{21}(\phi_{i+1,j} - \phi_{i,j}) + \frac{1}{4}g_{i+\frac{1}{2},j}^{22}(\phi_{i+1,j+1} - \phi_{i+1,j-1} + \phi_{i,j+1} - \phi_{i,j-1}) \quad (4.23e)$$

$$V_{i-\frac{1}{2},j} = g_{i-\frac{1}{2},j}^{21}(\phi_{i,j} - \phi_{i-1,j}) + \frac{1}{4}g_{i-\frac{1}{2},j}^{22}(\phi_{i,j+1} - \phi_{i,j-1} + \phi_{i-1,j+1} - \phi_{i-1,j-1}) \quad (4.23f)$$

$$V_{i,j+\frac{1}{2}} = \frac{1}{4}g_{i,j+\frac{1}{2}}^{21}(\phi_{i+1,j+1} - \phi_{i-1,j+1} + \phi_{i+1,j} - \phi_{i-1,j}) + g_{i,j+\frac{1}{2}}^{22}(\phi_{i,j+1} - \phi_{i,j}) \quad (4.23g)$$

$$V_{i,j-\frac{1}{2}} = \frac{1}{4}g_{i,j-\frac{1}{2}}^{21}(\phi_{i+1,j} - \phi_{i-1,j} + \phi_{i+1,j-1} - \phi_{i-1,j-1}) + g_{i,j-\frac{1}{2}}^{22}(\phi_{i,j} - \phi_{i,j-1}) \quad (4.23h)$$

En las ecuaciones 4.23 se ubican términos que se mantienen iguales para un mismo punto intercalado, por simplicidad dichos términos se agrupan en el término P , quedando como

$$P_{i+\frac{1}{2},j} = \phi_{i+1,j+1} - \phi_{i+1,j-1} + \phi_{i,j+1} - \phi_{i,j-1} \quad (4.24a)$$

$$P_{i-\frac{1}{2},j} = \phi_{i,j+1} - \phi_{i,j-1} + \phi_{i-1,j+1} - \phi_{i-1,j-1} \quad (4.24b)$$

$$P_{i,j+\frac{1}{2}} = \phi_{i+1,j+1} - \phi_{i-1,j+1} + \phi_{i+1,j} - \phi_{i-1,j} \quad (4.24c)$$

$$P_{i,j-\frac{1}{2}} = \phi_{i+1,j} - \phi_{i-1,j} + \phi_{i+1,j-1} - \phi_{i-1,j-1} \quad (4.24d)$$

Una vez discretizados los valores de las componentes de la velocidad, se puede despejar el valor del potencial a partir de la ecuación 4.22

$$\begin{aligned} \phi_{i,j} = & \frac{1}{\left(\frac{\rho g^{11}}{J}\right)_{i+\frac{1}{2},j} + \left(\frac{\rho g^{11}}{J}\right)_{i-\frac{1}{2},j} + \left(\frac{\rho g^{22}}{J}\right)_{i,j+\frac{1}{2}} + \left(\frac{\rho g^{22}}{J}\right)_{i,j-\frac{1}{2}}} \left(\left(\frac{\rho}{J}\right)_{i+\frac{1}{2},j} \left((g^{12}P)_{i+\frac{1}{2},j} + g_{i+\frac{1}{2},j}^{11} \phi_{i+1,j} \right) \right. \\ & - \left(\left(\frac{\rho}{J}\right)_{i-\frac{1}{2},j} \left((g^{12}P)_{i-\frac{1}{2},j} - g_{i-\frac{1}{2},j}^{11} \phi_{i-1,j} \right) + \left(\frac{\rho}{J}\right)_{i,j+\frac{1}{2}} \left((g^{21}P)_{i,j+\frac{1}{2}} + g_{i,j+\frac{1}{2}}^{22} \phi_{i,j+1} \right) \right. \\ & \left. \left. - \left(\frac{\rho}{J}\right)_{i,j-\frac{1}{2}} \left((g^{21}P)_{i,j-\frac{1}{2}} - g_{i,j-\frac{1}{2}}^{22} \phi_{i,j-1} \right) \right) \right) \end{aligned} \quad (4.25)$$

Mediante la ecuación 4.4 se pueden obtener los valores de la densidad, sustituyendo las ecuaciones de la velocidad en ella. Sin embargo dicha ecuación requiere el cuadrado del módulo de la velocidad en el sistema de coordenadas del dominio físico u y v las cuales se describen como

$$u = x_\xi U + x_\eta V \quad (4.26a)$$

$$v = y_\xi U + y_\eta V \quad (4.26b)$$

$$u^2 + v^2 = U^2(x_\xi^2 + y_\xi^2) + V^2(x_\eta^2 + y_\eta^2) + 2UV(x_\xi x_\eta + y_\xi y_\eta) \quad (4.26c)$$

con lo cual la ecuación de la densidad queda

$$\frac{\rho}{\rho_0} = \left[1 - \frac{U^2(x_\xi^2 + y_\xi^2) + V^2(x_\eta^2 + y_\eta^2) + 2UV(x_\xi x_\eta + y_\xi y_\eta)}{2H_0} \right] \quad (4.27)$$

Para este punto es evidente la necesidad de obtener los valores en las mallas intercaladas tanto de los elementos del tensor de la transformación (elementos g^{mn}) como de las razones de cambio $x_\xi, x_\eta, y_\xi, y_\eta$, los cuales se calculan de la misma manera. Sea s un parámetro de los previamente mencionados, sus aproximaciones en las mallas escalonadas quedan:

$$s_{i+\frac{1}{2},j} = \frac{1}{2}(s_{i+1,j} + s_{i,j}) \quad (4.28a)$$

$$s_{i-\frac{1}{2},j} = \frac{1}{2} (s_{i,j} + s_{i-1,j}) \quad (4.28b)$$

$$s_{i,j+\frac{1}{2}} = \frac{1}{2} (s_{i,j+1} + s_{i,j}) \quad (4.28c)$$

$$s_{i,j-\frac{1}{2}} = \frac{1}{2} (s_{i,j} + s_{i,j-1}) \quad (4.28d)$$

4.2.1. Condiciones de frontera

En el caso del perfil alar, se considera que la velocidad normal al mismo es cero, por lo que la ecuación 4.11 se iguala a cero y se discretiza

$$\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta\xi} g^{21} + \frac{\phi_{i,j-2} - 4\phi_{i,j-1} + 3\phi_{i,j}}{2\Delta\eta} g^{22} = 0 \quad (4.29)$$

si se despeja el potencial se obtiene:

$$\phi_{i,j} = \frac{1}{3} \left(4\phi_{i,j-1} - \phi_{i,j-2} - \frac{g^{21}}{g^{22}} (\phi_{i+1,j} - \phi_{i-1,j}) \right) \quad (4.30)$$

Como ya se comentó, en la simulación se debe especificar un corte a lo largo del cual existirá una discontinuidad en el potencial, dicho corte se hace coincidir por conveniencia con el corte ya presente en la malla durante su generación.

$$\phi_{M-1,j} = \phi_{0,j} + \Gamma \quad (4.31)$$

El valor de la circulación, como ya se ha mencionado, debe ser actualizado durante la ejecución del proceso iterativo, y como lo indica la condición de Kutta para este tipo de perfiles la velocidad en el borde de salida es cero. Como ya se mencionó en la frontera interna la velocidad normal es cero, solo resta igualar la velocidad tangencial, también a cero. La velocidad tangencial discretizada queda

$$\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta\xi} g^{11} + \frac{\phi_{i,j-2} - 4\phi_{i,j-1} + 3\phi_{i,j}}{2\Delta\eta} g^{12} = 0 \quad (4.32)$$

la cual en el borde de salida (punto $(M-1, 0)$), y considerando de nuevo $\Delta\xi = \Delta\eta = 1$, se simplifica a

$$(\phi_{M-1,0} - \phi_{1,0} - \Gamma) g^{11} + (\phi_{M-1,0} - 4\phi_{M-1,1} + 3\phi_{M-1,0}) = 0 \quad (4.33)$$

entonces la circulación viene dada por:

$$\Gamma = \phi_{M-2,0} - \phi_{1,0} + \frac{g^{12}}{g^{11}} (\phi_{M-1,2} - 4\phi_{M-1,1} + 3\phi_{M-1,0}) \quad (4.34)$$

Capítulo 5

Desarrollo Práctico

Los códigos de este trabajo fueron implementados en lenguaje Python 3, utilizando el paradigma de programación orientada a objetos, en éste los datos son trabajados como objetos con atributos y métodos pertenecientes a una clase, y que pueden ser heredados y aplicados a objetos o instancias de alguna subclase. En esta sección se describirá a fondo todo el desarrollo del proyecto, lo que brindará una mejor comprensión del trabajo para facilitar el uso del mismo. Por su parte, el código mismo contiene una documentación exhaustiva en la definición de cada clase y cada método, en donde se explica a fondo cada uno de los parámetros, así como el resultado obtenido después de la ejecución.

El presente trabajo requiere tener instaladas las siguientes dependencias (módulos y programas)

- Python 3
- Matplotlib
- Numpy
- Numba

el primero es el interprete del lenguaje de programación, éste es el encargado de ejecutar el código fuente, matplotlib es un modulo dedicado a la generación de gráficas, numpy es la librería de computación científica de Python y por último numba es una librería desarrollada para computación científica de alto performance.

El presente desarrollo se divide en diferente módulos, cada uno de ellos enfocado a una tarea distinta, y que en su conjunto logran el objetivo del proyecto, es decir, la generación de una malla estructurada que sea util para realizar análisis de DFC. Los modulos se enlistan a continuacion, junto con una breve descripcion de los mismos.

- **airfoil**. Contiene la definición de las clases airfoil y NACA4, lo que permite generar los perfiles correspondientes a la frontera interna de la malla. Proporciona métodos para la manipulación del (los) mismo (s) previo al inicio de la generación de la malla.
- **mesh**. Contiene la definicion general de una malla estructurada. Define la estructura interna y parámetros que la forman. Así como la funcionalidad general de una malla estructurada.

- **mesh_c.** Contiene la definición de la clase mesh_c. A su vez cuenta con los métodos exclusivos para la generación de mallas con tipología C.
- **mesh_o.** Contiene la definición de la clase mesh_c. A su vez cuenta con los métodos exclusivos para la generación de mallas con tipología O.
- **mesh_su2.** Contiene las funciones que permiten exportar las mallas del formato propio de este proyecto, al formato utilizado por el software SU2.
- **potential.** Contiene todas las funciones relacionadas con el análisis de flujo potencial, como es la misma función de flujo potencial, el campo de velocidades, el campo de presiones y el coeficiente de presión, la función de corriente, entre otros.
- **util.** Contiene funciones de utilidad para los demás módulos. Proporciona soporte al funcionamiento del resto de los módulos.

5.1. Estructura de las clases

5.1.1. Clase Airfoil

La clase airfoil se desarrolló para la creación de perfiles alares a partir de una archivo de texto que contenga la nube de puntos del perfil deseado, como dato de entrada se requiere el nombre del archivo texto y la dimensión, en metros, de la cuerda aerodinámica del perfil. La creación de un perfil aerodinámico a partir de una nube de puntos presenta una restricción, la densidad de la malla en la dirección tangencial al perfil (eje ξ) está limitada por el número de puntos proporcionados para su creación.

La clase airfoil hereda a sus subclases sus métodos, logrando así que los métodos comunes a todos los perfiles, sin importar el método mediante el cual fueron generados, queden condensados en esta clase, delegando así las particularidades referentes a la generación de diversos tipos de perfiles a su correspondiente subclase. La clase airfoil tiene una subclase de nombre “NACA4” la cual genera perfiles aerodinámicos NACA de la serie de 4 dígitos. La creación del perfil se hace a partir de la ecuación constitutiva de la serie. Como datos de entrada se requieren los valores de m (combudura máxima), p (posición de la combadura máxima), t (espesor máximo) y c (cuerda aerodinámica) característicos de esta serie. La subclase NACA4 tiene dos métodos para la creación del perfil, el primero de ellos, el método “create_linear” crea la nube de puntos con una distribución equidistante de los puntos en dirección del eje x . El segundo método, llamado “create_sin” genera a partir de una función senoidal una nube de puntos que permite que haya mayor densidad de nodos tanto en el borde de ataque como en el borde de salida del perfil.

Las tablas 5.1 y 5.2 presentan los métodos pertenecientes a la clase airfoil y a la clase NACA4 respectivamente, así como una breve descripción de la funcionalidad de cada método. A su vez, la figura 5.1 presenta un diagrama de clase donde se muestran los atributos necesarios para la creación de un perfil y los métodos que pueden ser utilizados sobre el mismo, condensados en la clase airfoil, así como su herencia con respecto a la clase NACA4, para la cual también se muestran los atributos requeridos para la generación de un perfil NACA de la serie de 4 dígitos y los métodos que sobre él pueden actuar.

Método	Descripción
<code>__init__(c, number)</code>	Crea objeto airfoil, con cuerda c [m]. number representa el número de perfil (en caso de flaps), por default es igual a uno.
<code>get_chord()</code>	Regresa la dimensión de la cuerda aerodinámica.
<code>get_number()</code>	Regresa el número de perfil (en caso de flaps o varios perfiles).
<code>get_x()</code>	Regresa un arreglo que contiene todas las coordenadas en el eje <i>x</i> de los puntos del perfil.
<code>get_y()</code>	Regresa un arreglo que contiene todas las coordenadas en el eje <i>y</i> de los puntos del perfil.
<code>get_union()</code>	Regresa el número de puntos que unen a los perfiles. Por default es cero.
<code>is_alone()</code>	Regresa verdadero o falso, dependiendo si existen 2 o más perfiles.
<code>is_boundary_()</code>	Regresa un arreglo de componentes verdadero o falso, dependiendo si el punto correspondiente forma parte de una frontera (perfil) o es un punto de unión.
<code>size()</code>	Regresa el número de puntos que forman el perfil.
<code>plot()</code>	Grafica el perfil alar.
<code>create(filename)</code>	Crea perfil aerodinámico a partir de un archivo que contiene la nube de puntos que lo describen.
<code>rotate(degrees)</code>	Rota el perfil degrees grados. Se considera rotación positiva en sentido horario.
<code>join(other, dx, dy, union)</code>	Une dos perfiles alares. El que invoca el método con other. dx y dy son las distancias sobre los ejes entre el borde de salida de un perfil y el borde de ataque del otro.
<code>to_csv(filename)</code>	Exporta la nube de puntos del perfil a un archivo de valores separados por comas (.csv)

Tabla 5.1: Métodos de la clase airfoil

Método	Descripción
<code>__init__(m, p, t, c, number)</code>	Crea objeto NACA4, con combadura máxima m, posición de ésta p, espesor máximo t y cuerda aerodinámica c[m]. number representa el número de perfil (en caso de flaps), por default es igual a uno.
<code>create_linear(points)</code>	Crea perfil NACA4 con una distribución equidistante de los puntos en el eje <i>x</i> con un número de puntos points.
<code>create_sin(points)</code>	Crea perfil NACA4 con una distribución senoidal de los puntos en el eje <i>x</i> con un número de puntos points. Esta distribución permite obtener mayor densidad de puntos tanto en el borde de ataque como de salida con respecto a la distribución lineal para un mismo número de puntos.

Tabla 5.2: Métodos de la subclase NACA4

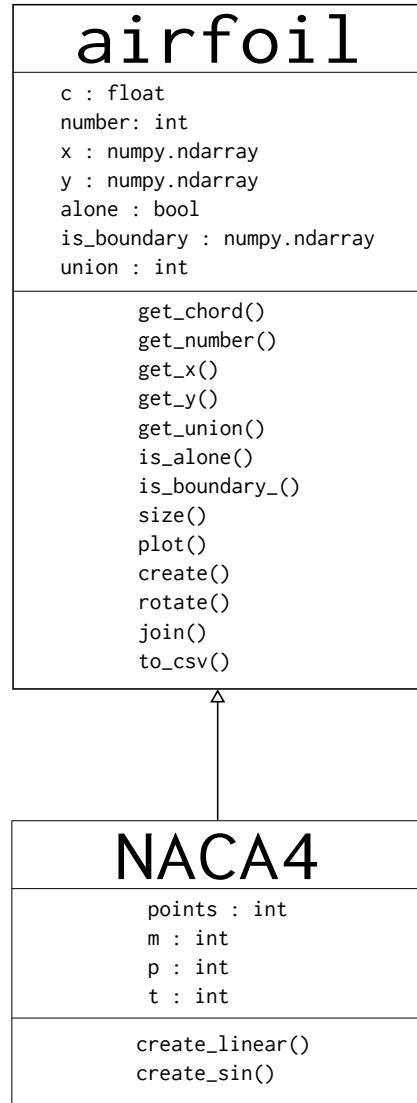


Figura 5.1: Diagrama de clase airfoil y subclase NACA4

5.1.2. Clase Mesh

La clase mesh se desarrolló para la generación de mallas estructuradas. Además es la superclase a la que pertenecen las subclases `mesh_c` y `mesh_o`. Esto resulta en una estructura similar al caso de la clase `airfoil`, es decir la clase `mesh` contiene la definición general de los parámetros necesarios para la creación de una malla estructurada, además de también contener los métodos de uso general, delegando así las particularidades de cada tipología a su correspondiente subclase. Es en esta clase donde están definidos los métodos de creación de malla mediante ecuaciones algebráicas y los métodos que evalúan la calidad de la malla.

La tabla 5.3 enumera los métodos correspondientes a la clase `mesh` dando a su vez una breve descripción de cada uno de ellos.

Método	Descripción
<code>__init__(R, M, N, airfoil)</code>	Crea un objeto mesh donde R es la dimensión de la frontera externa, M es el número de niveles en el eje ξ , N es el número de niveles en el eje η y airfoil es un objeto de la clase homónima que contiene la información del perfil aerodinámico a analizar.
<code>plot()</code>	Representación gráfica de la malla generada.
<code>to_txt_mesh(filename)</code>	Exporta la malla a un archivo de texto de extensión <code>.txt_mesh</code> . Este formato es de desarrollo propio y se implementó con la intención de importar al software mallas previamente generadas para su análisis.
<code>gen_inter_pol(eje)</code>	Genera una malla mediante el método de interpolación polinomial, la cual se basa en el polinomio de Lagrange. eje representa el eje a lo largo del cual se realiza la interpolación, por default es el eje η .
<code>gen_TFI()</code>	Genera una malla mediante el método de interpolación transfinita (TFI).
<code>gen_inter_Hermite()</code>	Genera una malla mediante el método de interpolación de Hermite.
<code>get_aspect_ratio()</code>	Calcula el aspect ratio de cada celda para toda la malla.
<code>get_skew()</code>	Calcula el skew de cada celda para toda la malla.

Tabla 5.3: Métodos de la clase mesh

Como previamente se ha mencionado, la clase mesh sirve de superclase de las subclases `mesh_c` y `mesh_o`. Como el nombre de las mismas lo indica son subclases desarrolladas para la generación de mallas tipo C y tipo O respectivamente. En este trabajo se han desarrollado los mismos métodos para ambas subclases, con lo que se logra la misma funcionalidad en ambos casos. La razón de desarrollar dos clases distintas atiende a la necesidad de manejar las ecuaciones de diferente manera en las fronteras de la malla dependiendo de la tipología de ésta.

La tabla 5.4 enumera los métodos disponibles para ambos tipos de malla, así como una breve descripción de ellos. En dicha tabla se puede observar que para la solución de las ecuaciones tanto de Laplace como de Poisson se proporcionan tres diferentes métodos, y que los tres presentan la misma descripción y por lo tanto mismos resultados, esto se hizo con la idea de dotar de versatilidad al programa desarrollado, para ambas ecuaciones el primer método no tiene “sufijo”, esto implica que es el método clásico de solución, el cual implica dos ciclos for anidados para recorrer toda la matriz, sin embargo este enfoque deja de ser asequible si se pretende generar mallas con un numero de nodos grande, dado que el tiempo de ejecucion crece exponencialmente con el numero de nodos, por un lado porque a mayor numero de nodos ya sea en el eje ξ , η o ambos, mayor número de ejecuciones dentro de los ciclos for, y por otro lado se ha observado que a mayor numero de nodos mayor numero de iteraciones es necesario para lograr el criterio de convergencia. Para contrarrestar dicho efecto se crearon dos metodos adicionales, los cuales buscan entregar los mismos resultados ayudandose de otras herramientas para mejorar el performance. El primer metodo tiene “sufijo” `_v`, en este método se utilizan operaciones de vectorizado para agilizar la ejecución del programa, y que significa una reducción exponencial del tiempo de ejecucion, sin embargo la característica

del vectorizado implica realizar todas las operaciones de forma paralela, con lo que se pierde la habilidad de llevar a cabo los métodos de Gauss-Seidel y SOR de manera estricta, para compensar este hecho se divide la malla en submallas y se resuelve cada una a la vez, logrando así que los valores obtenidos en una submalla previamente calculada, tengan efecto en las subsecuentes submallas dentro de la misma iteración. Por último se proporciona un método con “sufijo” `_n`, dicho método ejecuta las operaciones de la misma manera que el enfoque clásico, pero además utiliza el módulo **numba**, el cual es un módulo creado específicamente para computación científica de alto performance, y que tiene una mejora en la velocidad del ejecución de aproximadamente el doble cuando se le compara con el método de vectorizado. Es este último método el recomendado para utilizar, y solo en caso de no ser posible su uso, se debe optar en primera instancia por el método de vectorizado y por último el método clásico.

Método	Descripción
<code>__init__(R, M, N, airfoil)</code>	Crea un objeto <code>mesh_c</code> o <code>mesh_o</code> , donde <code>R</code> es la dimensión de la frontera externa, <code>M</code> es el número de niveles en el eje ξ , <code>N</code> es el número de niveles en el eje η y <code>airfoil</code> es un objeto de la clase homónima que contiene la información del perfil aerodinámico a analizar.
<code>gen_Laplace(metodo)</code> <code>gen_Laplace_v(metodo)</code> <code>gen_Laplace_n(metodo)</code>	Genera la malla resolviendo la ecuación de Laplace. La variable <code>metodo</code> se refiere al método iterativo de solución, el usuario puede escoger entre Jacobi, Gauss-Seidel y SOR. Por default se utiliza el método SOR
<code>gen_Poisson(metodo)</code> <code>gen_Poisson_v(metodo)</code> <code>gen_Poisson_n(metodo)</code>	Genera la malla resolviendo la ecuación de Poisson. La variable <code>metodo</code> se refiere al método iterativo de solución, el usuario puede escoger entre Jacobi, Gauss-Seidel y SOR. Por default se utiliza el método SOR
<code>tensor()</code>	Calcula el tensor métrico de la transformación, tanto la transformación directa como indirecta. También calcula el Jacobiano de dicha transformación y los valores de las derivadas parciales x_ξ , x_η , y_ξ y y_η . Los valores que regresa el método son matrices de $M \times N$, ya que todos los términos se calculan para cada uno de los puntos de la malla.
<code>to_su2()</code>	Convierte la malla al formato del software SU2, con el objetivo de realizar simulaciones en dicho programa sobre las mallas desarrolladas en este proyecto.

Tabla 5.4: Métodos de las clases `mesh_c` y `mesh_o`

La figura 5.2 es una representación de la estructura de la clase `mesh` y su relación (herencia) con respecto a las subclases `mesh_c` y `mesh_o`. En este diagrama se enlistan los atributos y métodos generales pertenecientes a la clase `mesh`, a su vez se muestran los parámetros y métodos que se requieren para la generación de mallas tanto tipo C como tipo O, pertenecientes a su respectiva clase.

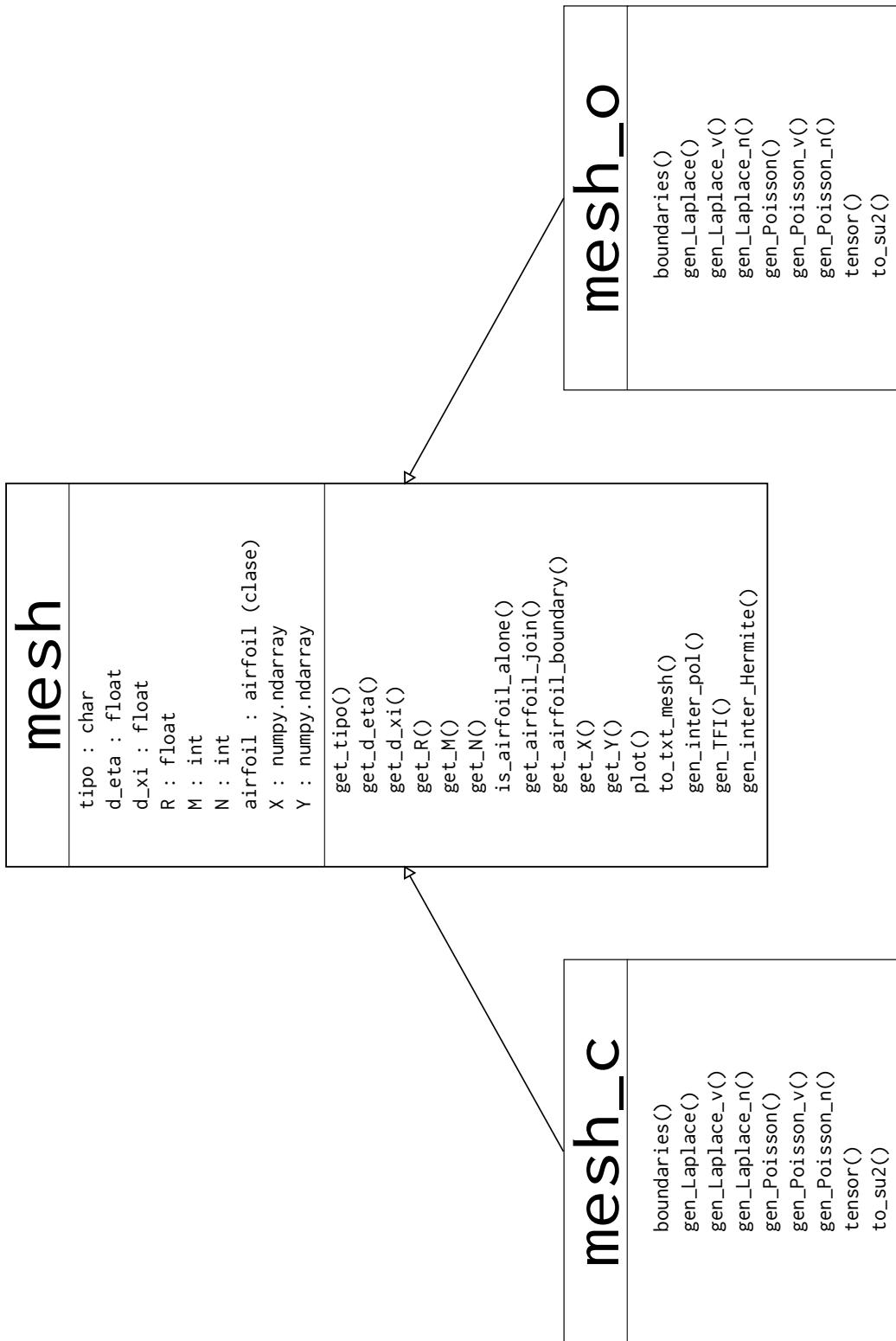


Figura 5.2: Diagrama de clase mesh y subclases mesh_c y mesh_o

5.2. Diagramas de flujo

5.2.1. Estructura global del programa

Siguiendo la estructura adoptada por los software de simulación y análisis de ingeniería, donde la solución del problema se divide en 3 procesos generales

1. Preproceso. Introducción de los datos de entrada del programa, definición de la geometría, definición de las propiedades del fluido.
2. Solver. Parte central del programa de solución. Resuelve de forma iterativa el sistema de ecuaciones.
3. Postproceso. Representación gráfica tanto del dominio como de la malla. Mapas de contorno para las variables de importancia.

se muestra un diagrama en la figura 5.3 donde se relaciona cada parte de este proyecto con el correspondiente proceso de los arriba mencionados.

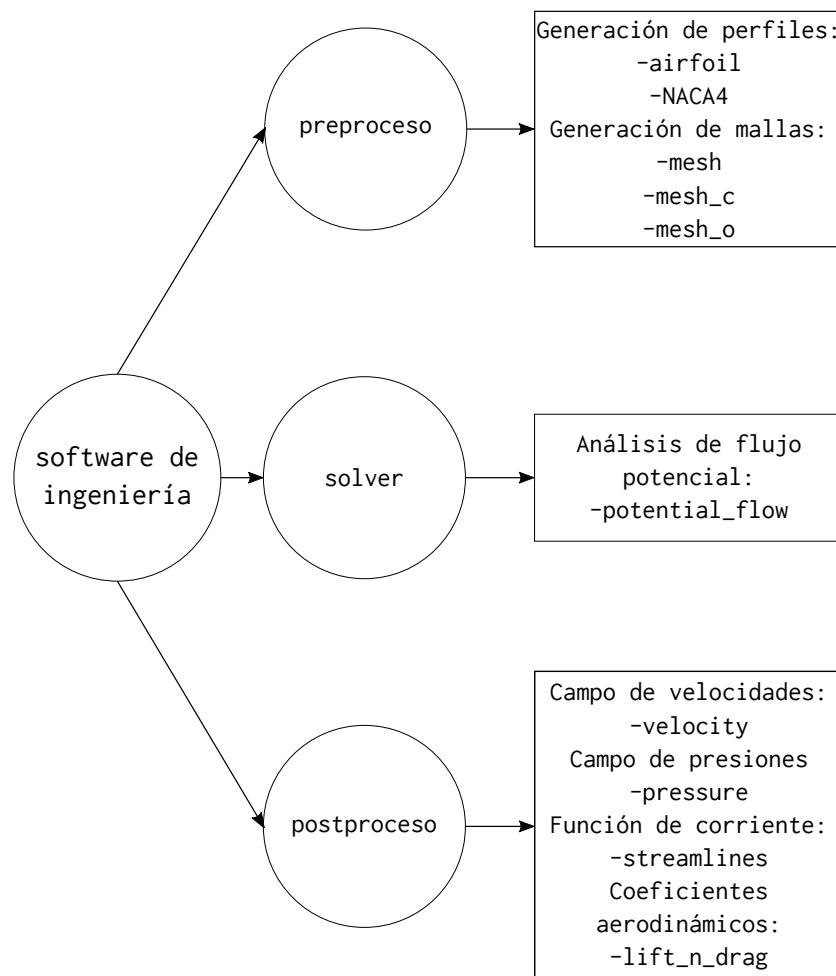


Figura 5.3: Estructura global del programa, dividido en preproceso, solver y postproceso

A continuación se muestran los diagramas de flujo que explican el funcionamiento de los 3 procesos generales dentro de este proyecto.

La figura 5.4 presenta el diagrama de flujo que se sigue para generar una malla y poder guardar dichos datos tanto en el formato de este proyecto como en el formato de SU2. Estas acciones corresponden al preproceso del análisis.

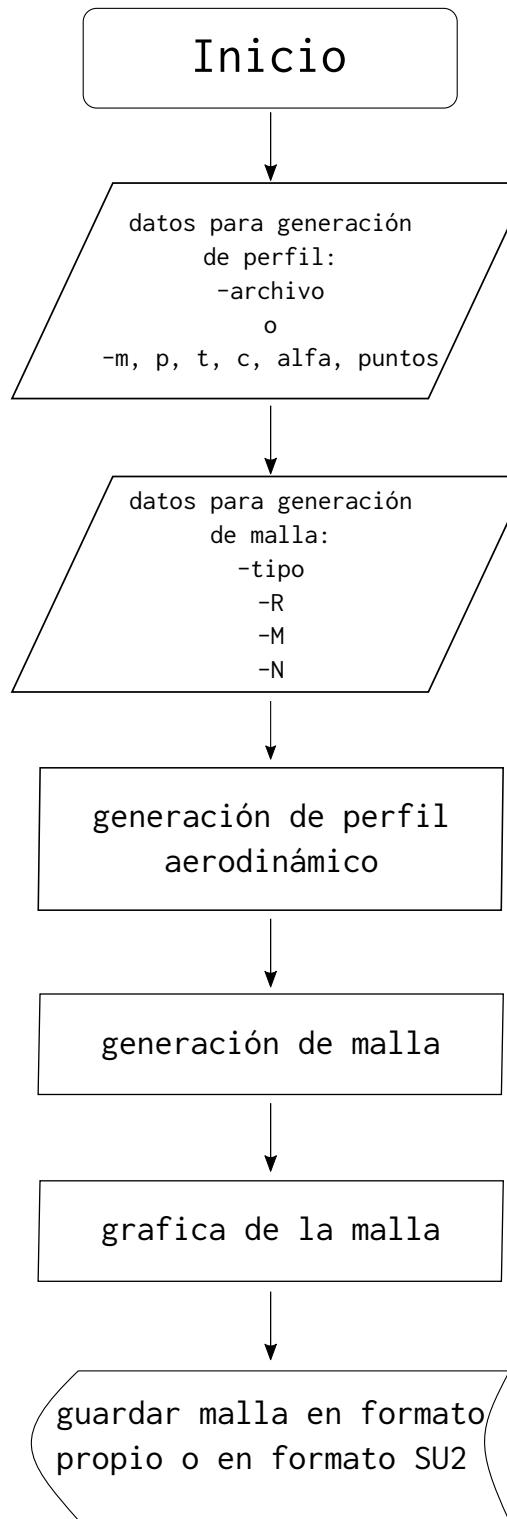


Figura 5.4: Diagrama de flujo para generación de mallas (preproceso)

El diagrama de flujo de la figura 5.5 representa el algoritmo empleado para el análisis de flujo potencial, lo cual corresponde al “solver” del programa.

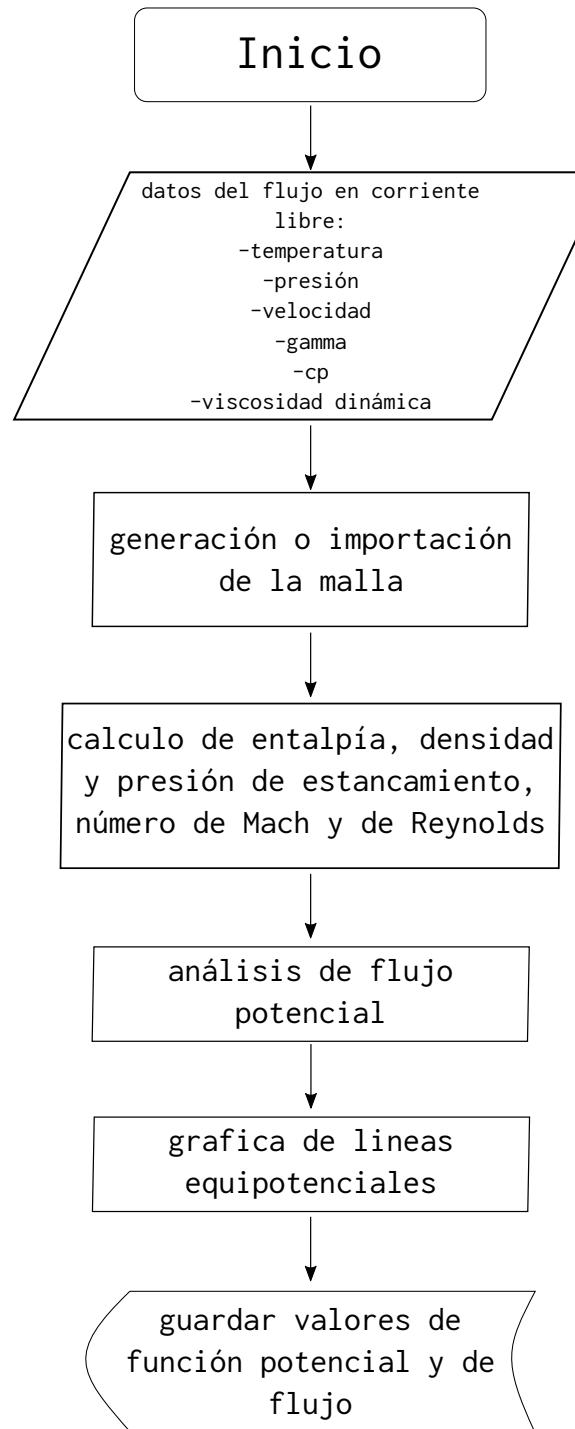


Figura 5.5: Diagrama de flujo para análisis de flujo potencial(solver)

El postproceso se ve ilustrado en la figura 5.6, la cual representa el algoritmo para la obtención de diferentes parámetros de interés en los análisis aerodinámicos, entre ellos los coeficientes de presión (c_p), de sustentación (c_l) y de resistencia al avance (c_d).

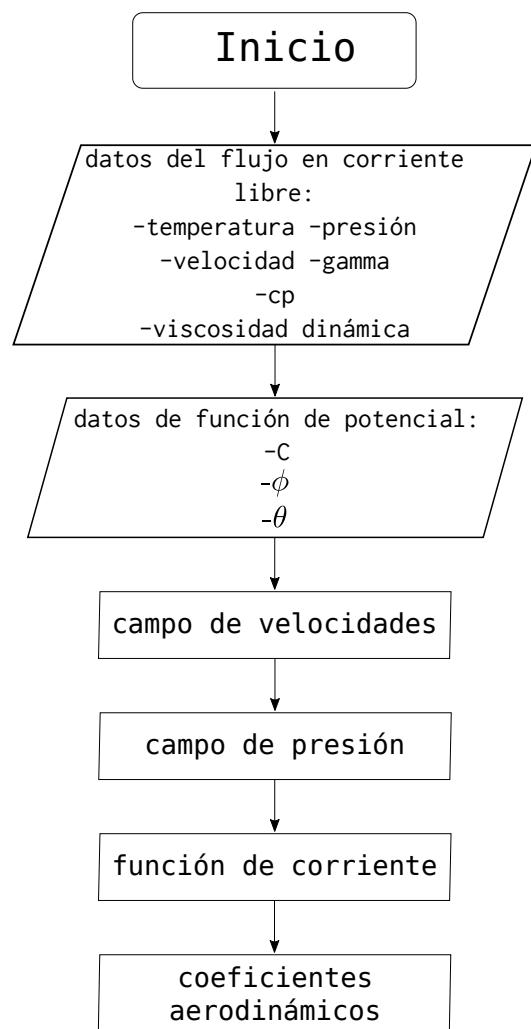


Figura 5.6: Diagrama de flujo para análisis de postproceso

5.2.2. Funciones principales

Hasta ahora, solo se han presentado diagramas de flujo que presentan el funcionamiento general del programa sin entrar en específicos sobre los algoritmos de las funciones y los métodos que lo componen. Se muestran a continuación diagramas de flujo para la generación de mallas mediante la ecuación de Laplace y Poisson, además de la solución a la ecuación de potencial, dado que son los algoritmos más complejos en este trabajo.

La figura 5.7 presenta el algoritmo utilizado para la generación de mallas mediante la solución de la ecuación de Laplace. De igual manera la figura 5.8 presenta el diagrama de flujo para la solución de la ecuación de Poisson. Por último, el diagrama de flujo ilustrado en la figura 5.9 muestra el algoritmo empleado para resolver la función potencial.

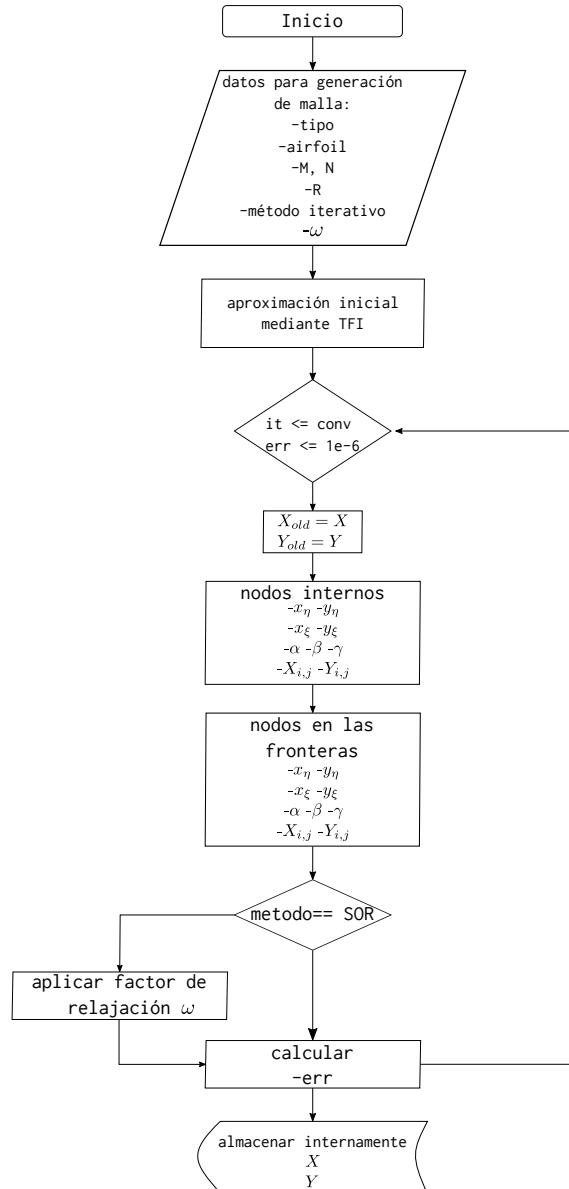


Figura 5.7: Diagrama de flujo para generación de mallas mediante la ecuación de Laplace

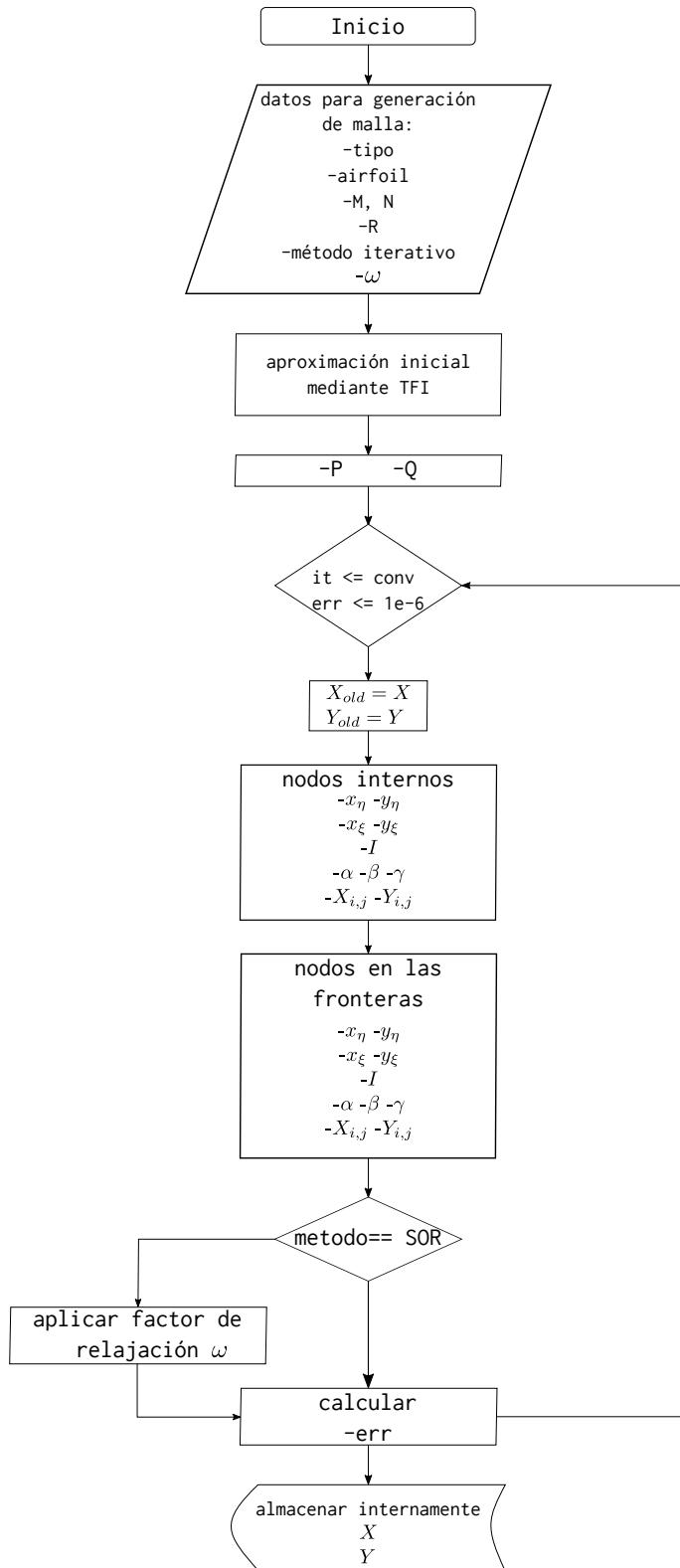


Figura 5.8: Diagrama de flujo para generación de mallas mediante la ecuación de Poisson

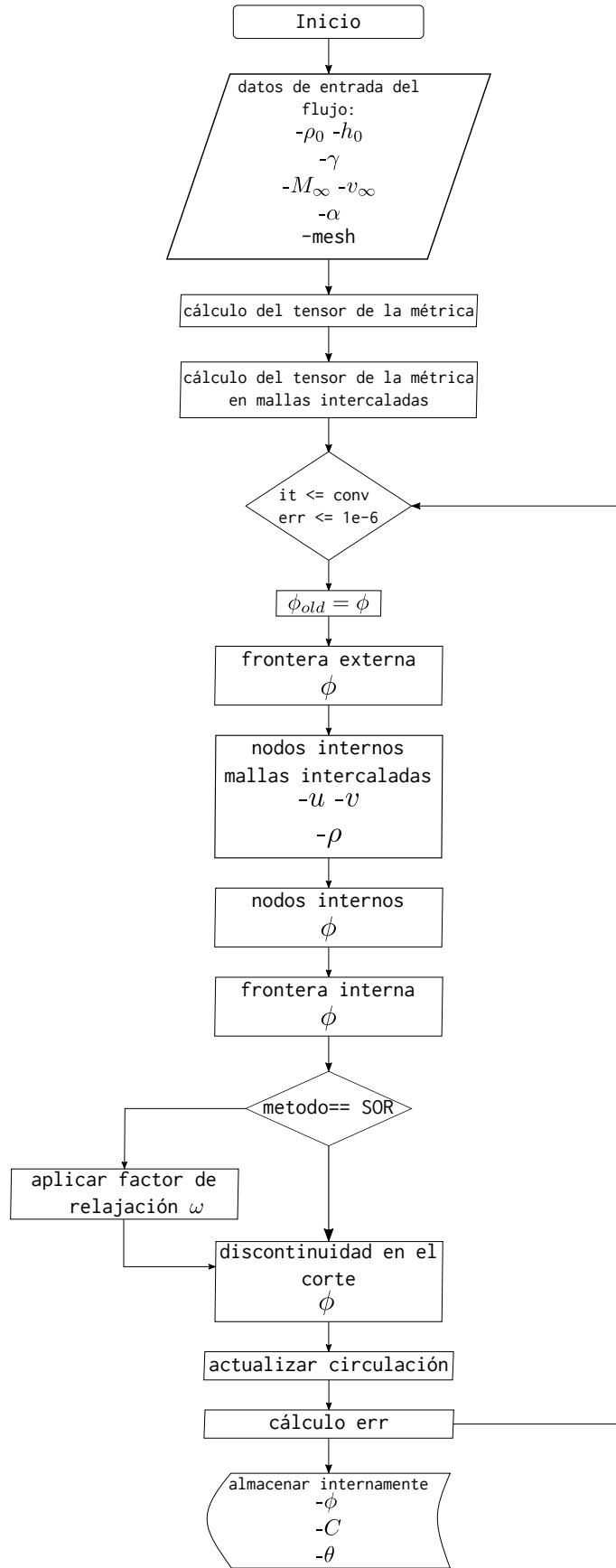


Figura 5.9: Diagrama de flujo para generación de mallas mediante la ecuación de Poisson

Capítulo 6

Resultados

Los resultados de este trabajo se dividen en tres secciones diferentes, la primera de ellas lidia con el alcance de las mallas generadas mediante este programa, analizando para diferentes configuraciones la distribución de los nodos. En segundo lugar para diferentes configuraciones se evalúa la calidad de la malla mediante el “aspect ratio” y el “skewness” de las celdas de la misma, con la intención de mostrar la calidad lograda, lo que dota de confiabilidad al proyecto para su uso en posteriores proyectos. Por último se llevan a cabo simulaciones de flujo potencial, también se realizan simulaciones a través del uso del software SU2, con el fin de confirmar la correcta generación de las mallas y de mostrar su aplicabilidad.

6.1. Alcance y Límites de Aplicación

En esta sección se analizan las mallas resultantes para diferentes ángulos de ataque de un perfil seleccionado, con lo que se obtiene un rango de valores para el ángulo de ataque en los que todavía se cuenta con una malla que resulte de utilidad para análisis de DFC. Normalmente este límite se alcanza cuando se presentan cruces en las líneas que forman la malla.

6.1.1. Malla O. Frontera interna única

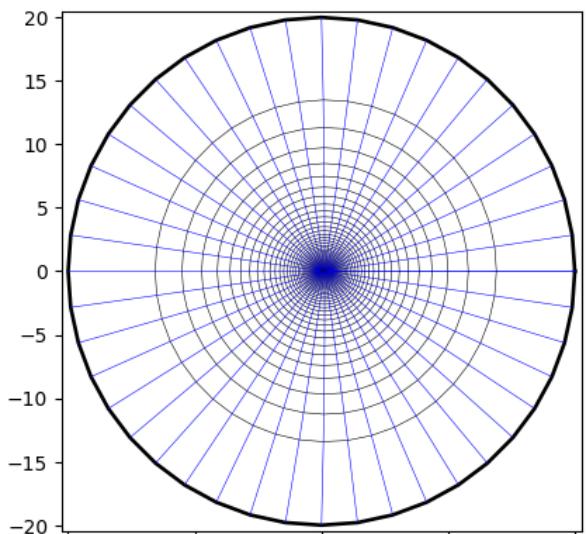
La primera malla que se presenta es una malla tipo O con un solo perfil. La tabla 6.1 presenta las características de la malla, así como los casos límite de aplicabilidad. La figura 6.1 muestra la malla generada para un ángulo de ataque $\alpha = 0^\circ$. A su vez, las figuras 6.2a y 6.2b muestran un acercamiento al perfil para un ángulo de ataque $\alpha = 50^\circ$, condición en la cual todavía se muestra una malla dentro del rango de usabilidad y que corresponde al ángulo máximo recomendado de rotación de la frontera interna. Mientras que las figuras 6.2c y 6.2d muestran un acercamiento al perfil con un ángulo de ataque $\alpha = -50^\circ$, que corresponde al ángulo mínimo recomendado de rotación de la frontera interna.

6.1.2. Malla O. Frontera interna múltiple

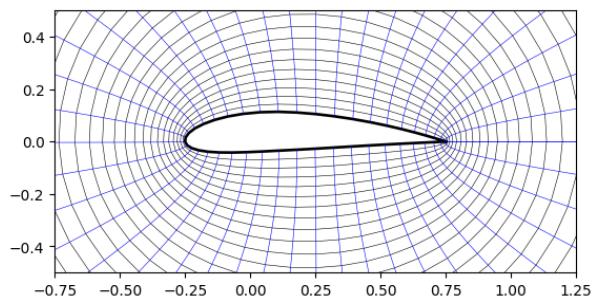
El segundo caso de análisis presenta una malla tipo O con un perfil NACA 4415 y un flap del tipo perfil externo, el cual también es un perfil NACA 4415. En el caso de esta configuración, el ángulo de ataque del perfil se mantiene constante a 0° , mientras que el ángulo de deflexión del flap se va variando. La tabla 6.2 presenta las características de esta configuración. La figura 6.3 presenta la malla generada, tanto una vista completa de la misma, así como un acercamiento en el cual se

Malla O	
Característica	Valor
Perfil	NACA 4415
M	45
N	45
Ecuación	Poisson
aa_m	20.5
cc_m	6.5
Método	SOR
Ángulo máximo	60°
Ángulo máximo recomendado	50°
Ángulo mínimo	-60°
Ángulo mínimo recomendado	-50°

Tabla 6.1: Características de malla O con perfil NACA 4415

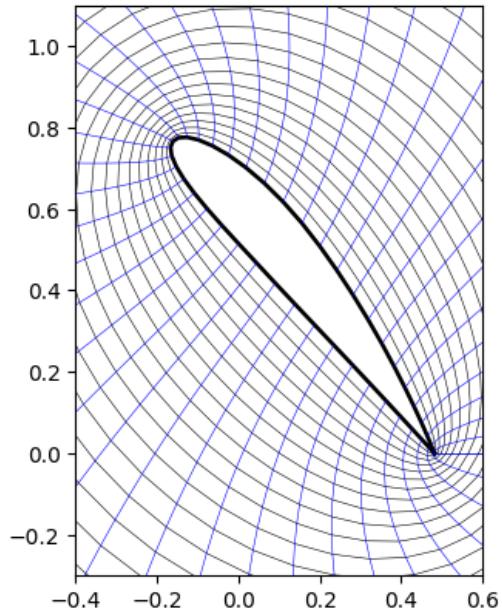


(a) Vista completa

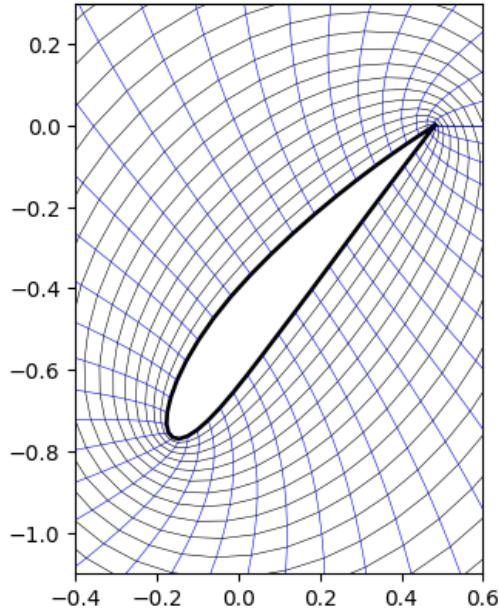


(b) Acercamiento a perfil

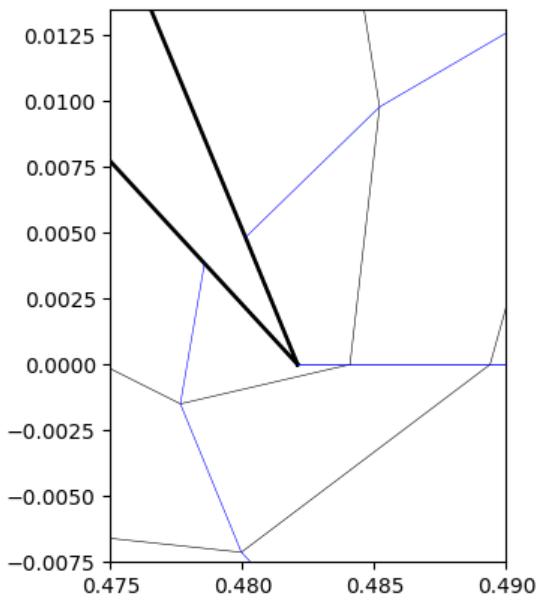
Figura 6.1: Malla tipo O con perfil NACA 4415 con ángulo de ataque 0°



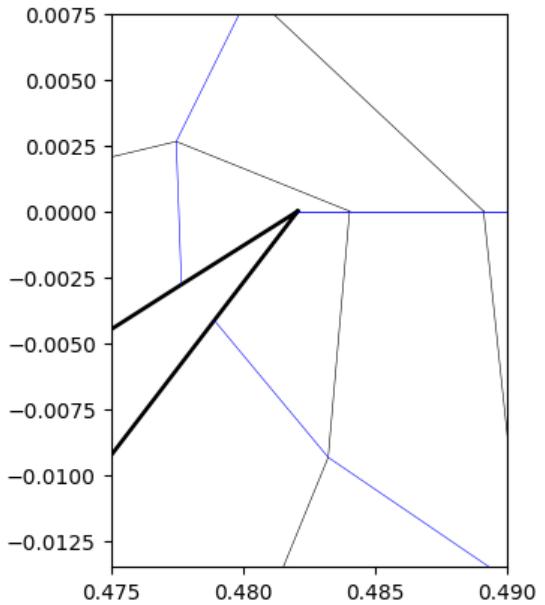
(a) 50° Vista del perfil



(c) -50° Vista del perfil



(b) 50° Vista del borde de salida



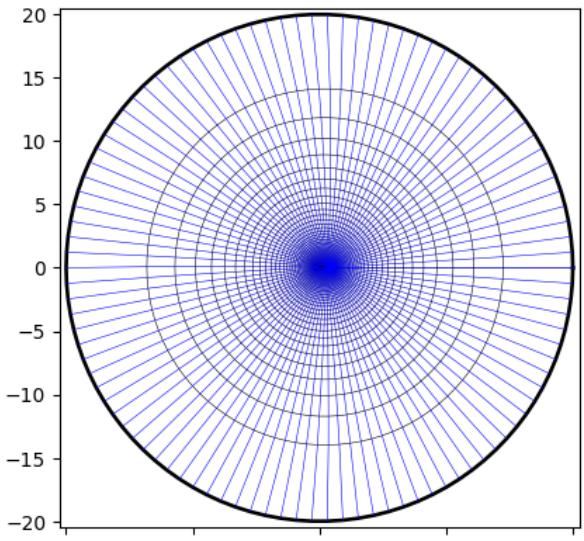
(d) -50° Vista del borde de salida

Figura 6.2: Malla tipo O con perfil NACA 4415 con ángulos de ataque $\alpha = 50^\circ$ y $\alpha = -50^\circ$. Acercamiento al perfil y al borde de salida

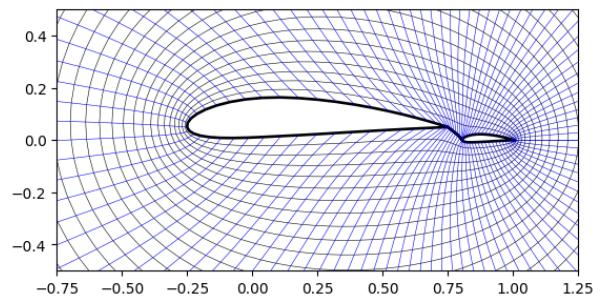
logra apreciar tanto el perfil como el flap, el cual tiene un ángulo de deflexión igual a 0° . Por su parte la figura 6.4 muestra una malla generada bajo los mismos parámetros y una deflexión del flap igual a 50° , la cual cumple en términos generales con las características de una malla de calidad, y corresponde al ángulo máximo recomendado de deflexión del flap. En contraste, la figura 6.5 muestra una malla para un caso en el cual el flap tiene un angulo de deflexión igual a -10° , dicha malla corresponde al ángulo mínimo recomendado de deflexión del flap, y como se observa, es una malla todavía de utilidad práctica.

Malla O	
Característica	Valor
Perfil	NACA 4415
Flap	NACA 4415
M	103
N	45
Union entre perfil y flap	6
Ecuación	Poisson
aa_m	20.5
cc_m	6.5
Método	SOR
Ángulo máximo	60°
Ángulo máximo recomendado	50°
Ángulo mínimo	-15°
Ángulo mínimo recomendado	-10°

Tabla 6.2: Características de malla O con perfil y flap NACA 4415

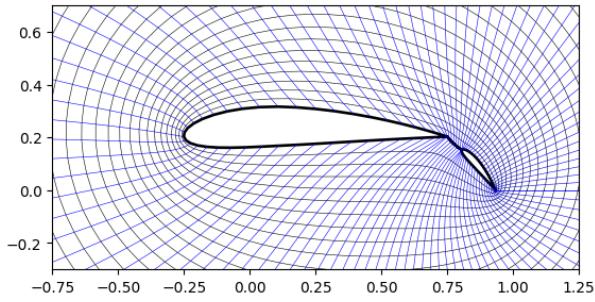


(a) Vista Completa

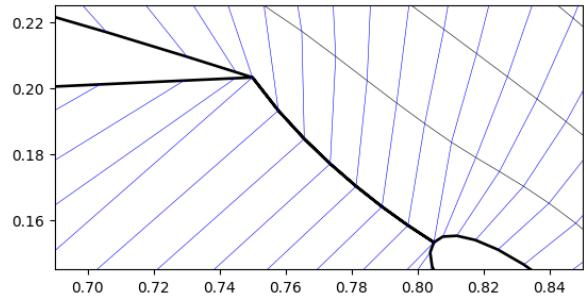


(b) Acercamiento a perfil

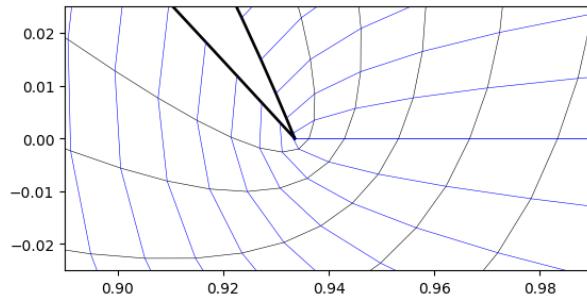
Figura 6.3: Malla tipo O con perfil y flap NACA 4415, ambos con ángulo de ataque $\alpha = 0^\circ$



(a) Flap a 50°

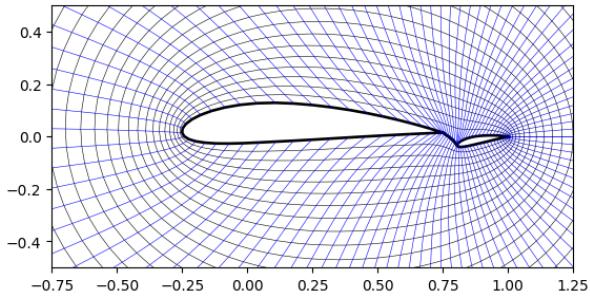


(b) Acercamiento a borde de salida de perfil y borde de ataque de flap

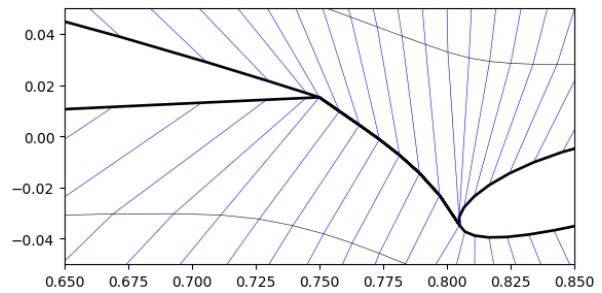


(c) Acercamiento a borde de salida de flap

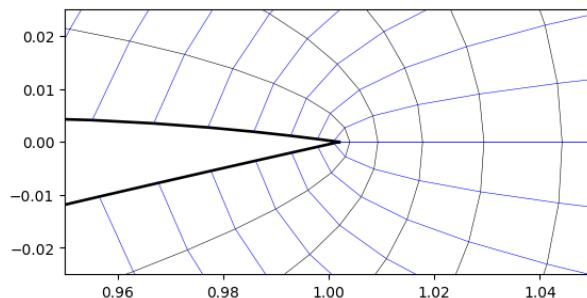
Figura 6.4: Malla tipo O con perfil y flap NACA 4415. El perfil tiene un ángulo de ataque $\alpha = 0^\circ$ y el flap un ángulo de deflexión $\delta = 50^\circ$



(a) Acercamiento a perfil y flap



(b) Acercamiento a borde de salida de perfil y borde de ataque de flap



(c) Acercamiento a borde de salida de flap

Figura 6.5: Malla tipo O con perfil y flap NACA 4415. El perfil tiene un ángulo de ataque $\alpha = 0^\circ$ y el flap un ángulo de deflexión de $\delta = -10^\circ$

De estos resultados se observa, que para las mallas con tipología tipo O el principal aspecto a tener en cuenta al momento de la generación de las mismas, es el cruce entre las líneas que la definen, ya que al aumentar el ángulo de ataque la tendencia hacia el cruce entre líneas aumenta. La zona donde en su mayoría se presenta la dificultad ya mencionada es la zona cercana al borde de salida, ya sea del perfil o del flap según sea el caso. La explicación a este fenómeno radica en la tipología (tipo O) ya que conforme se aumenta la inclinación de la frontera interna algunas líneas se ven forzadas a “rodear” el perfil para lograr la conectividad entre los correspondientes puntos entre las fronteras.

6.1.3. Malla C. Frontera interna única.

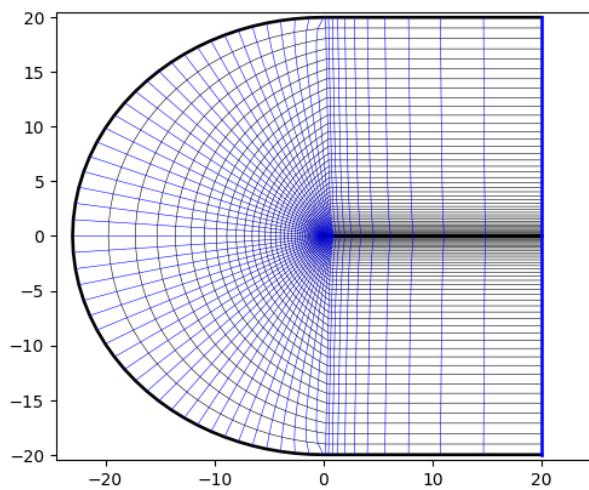
El tercer caso de análisis es similar al primer caso, es decir se genera una malla alrededor de un perfil aerodinámico NACA 4415, sin embargo la malla resultante es una malla tipo C. En la tabla 6.3 se muestran los valores asignados a cada parámetro con los que esta malla ha sido generada. La figura 6.6 muestra la vista de la malla, brinda una vista total de la misma así como un acercamiento a la región donde se encuentra el perfil. La figura 6.7 presenta la malla generada para los ángulos de ataque, tanto máximo como mínimo sugeridos, los cuales son $\alpha = 50^\circ$ (figura 6.7a) y $\alpha = -50^\circ$ (figura 6.7b) respectivamente. Se observa todavía una buena calidad de la malla.

Malla C	
Característica	Valor
Perfil	NACA 4415
M	73
N	45
Ecuación	Poisson
aa_m	20.5
cc_m	7.5
Método	SOR
Ángulo máximo	70°
Ángulo máximo recomendado	50°
Ángulo mínimo	-70°
Ángulo mínimo recomendado	-50°

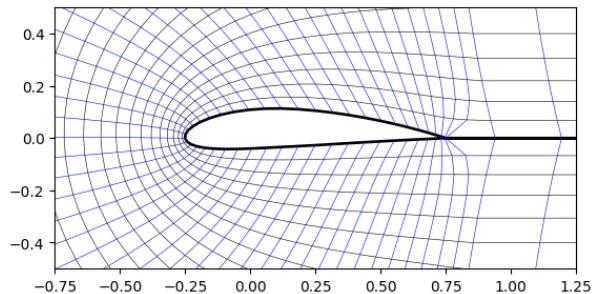
Tabla 6.3: Características de malla C con perfil NACA 4415

6.1.4. Malla C. Frontera interna múltiple

El último caso de estudio en esta sección corresponde a una malla de tipología C, la cual tiene como frontera interna un perfil NACA 4415 el cual cuenta con un flap con el mismo diseño NACA 4415. El perfil se mantiene constante con un ángulo de ataque $\alpha = 0^\circ$ mientras que el ángulo de deflexión del flap se varía. La figura 6.8 muestra la malla generada de acuerdo a los parámetros establecidos en la tabla 6.4, en la figura 6.8a se observa una vista completa de la malla, por su parte la figura 6.8b proporciona un acercamiento a la zona donde se encuentran el perfil y el flap. De igual manera la figura 6.9 muestra la malla resultante para una deflexión de flap de 20° , caso establecido para la presente configuración como el ángulo de deflexión máximo recomendado. Por

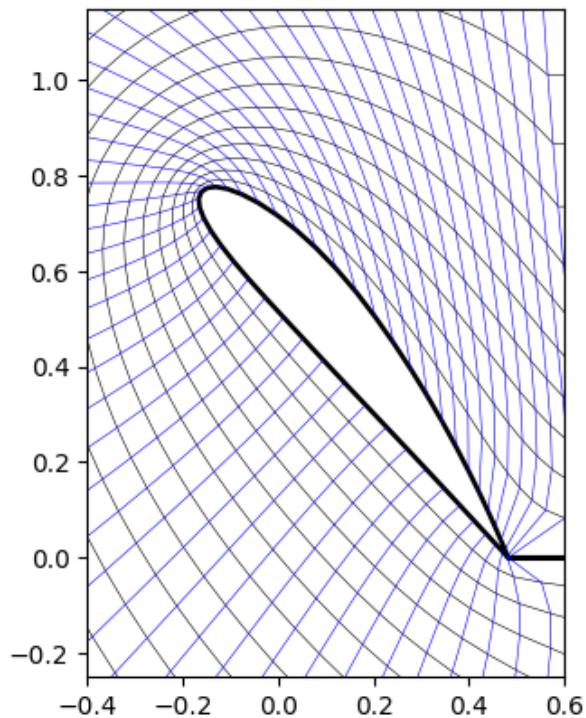


(a) Vista completa

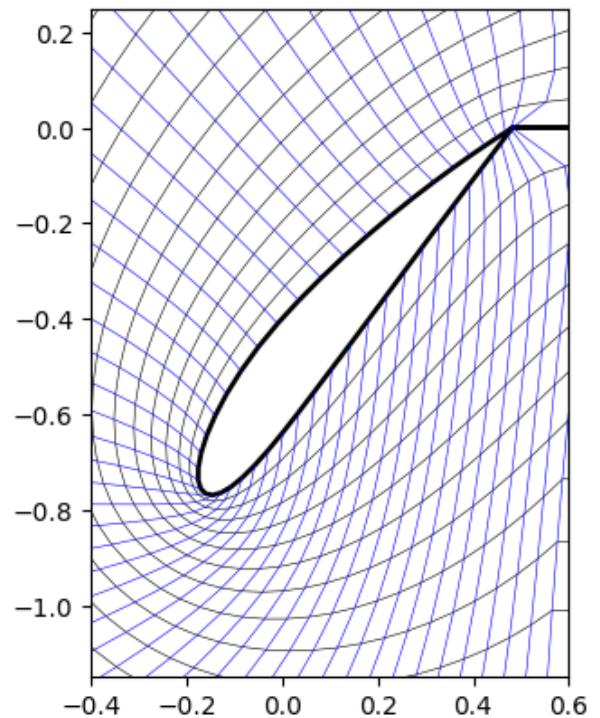


(b) Acercamiento a perfil

Figura 6.6: Malla tipo C con perfil NACA 4415, a un ángulo de ataque $\alpha = 0^\circ$



(a) 50° Vista del perfil



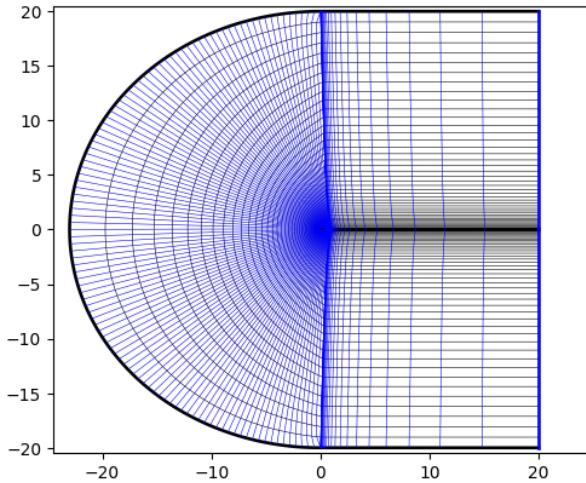
(b) -50° Vista del perfil

Figura 6.7: Malla tipo C con perfil NACA 4415 con ángulos de ataque $\alpha = 50^\circ$ y $\alpha = -50^\circ$. Límites recomendados

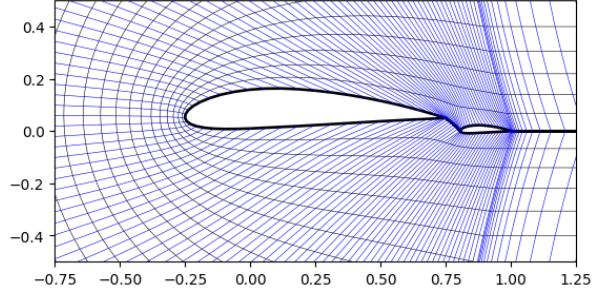
último la figura 6.10 corresponde al caso del mínimo ángulo sugerido de deflexión del flap, el cual es de -30° .

Malla C	
Característica	Valor
Perfil	NACA 4415
M	163
N	45
Ecuación	Poisson
aa_m	20.5
cc_m	7.5
Método	SOR
Ángulo máximo	30°
Ángulo máximo recomendado	20°
Ángulo mínimo	-40°
Ángulo mínimo recomendado	-30°

Tabla 6.4: Características de malla C con perfil y flap NACA 4415



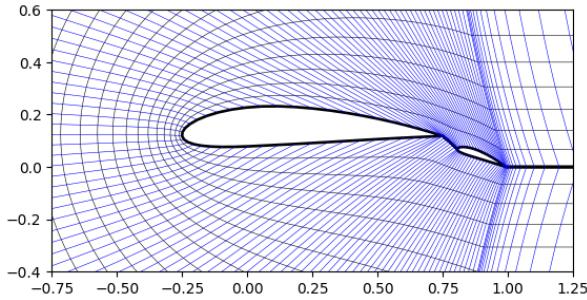
(a) Vista completa



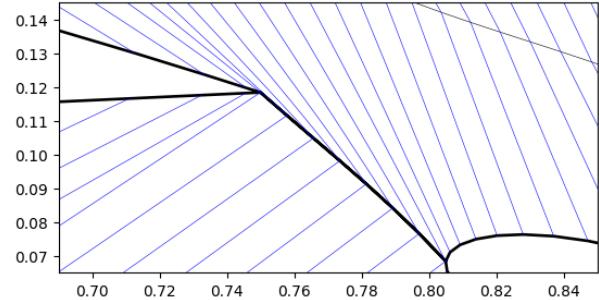
(b) Acercamiento a perfil

Figura 6.8: Malla tipo C con perfil y flap NACA 4415, ambos con ángulo de ataque y deflexión 0°

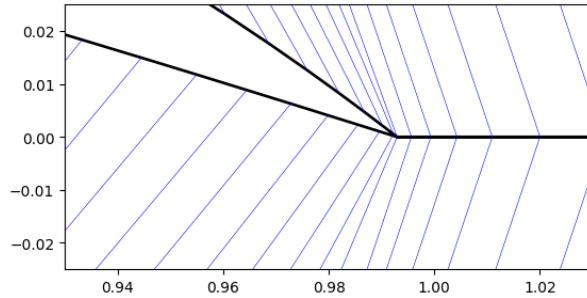
De los resultados obtenidos se puede observar que el principal aspecto de cuidado durante la generación de mallas tipo C es la posibilidad del cruce entre las líneas que la definen. En contraste con las mallas de tipología O, las mallas tipo C no presentan esta problemática en la zona de los bordes de salida debido a que las líneas continúa con su recorrido de manera horizontal, sin embargo dicho problema se encuentra presente en la sección de unión entre perfil y flap, esto debido a la la distribución de puntos en la frontera externa en la zona que conecta con el perfil. Otro aspecto importante en las mallas tipo C con únicamente un perfil, es el hecho de que al aumentar el ángulo de ataque de éste, las líneas adyacentes al perfil comienzan a separarse de manera brusca, lo cual



(a) Acercamiento a perfil y flap

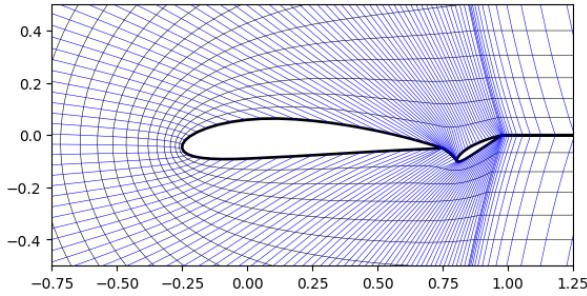


(b) Acercamiento a borde de salida de perfil

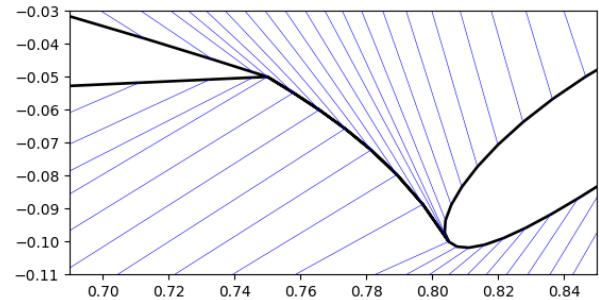


(c) Acercamiento a borde de salida de flap

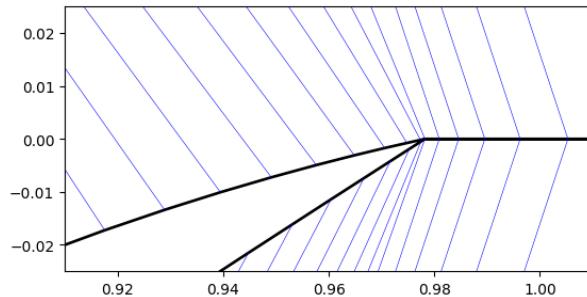
Figura 6.9: Malla tipo C con perfil y flap NACA 4415. El perfil tiene un ángulo de ataque $\alpha = 0^\circ$ y el flap un ángulo de deflexión $\delta = 20^\circ$



(a) Acercamiento a perfil y flap



(b) Acercamiento a borde de salida de perfil y borde de ataque de flap



(c) Acercamiento a borde de salida de flap

Figura 6.10: Malla tipo C con perfil y flap NACA 4415. El perfil tiene un ángulo de ataque $\alpha = 0^\circ$ y el flap un ángulo de deflexión $\delta = -30^\circ$

vuelve a la malla inservible dado que las celdas se deforman cambiando su aspect ratio y dichas celdas estarán localizadas en zonas de grandes gradientes donde lo que se busca es la mejor calidad posible en las celdas.

6.2. Calidad de la malla

En esta sección se discuten los resultados obtenidos para diferentes configuraciones de malla, analizando la calidad de la misma. Este rubro es evaluado mediante el método utilizado en la librería “The Verdict Geometric Quality Library”, mencionado previamente en el capítulo 3. Los aspectos que se analizan son el “aspect ratio” y el “skewness” en cada una de las celdas de la malla.

Para esto es necesario que se defina una configuración inicial, es decir se define la tipología de la malla, el perfil que fungirá de frontera interna, las dimensiones de la malla M y N , y los valores de entrada para las funciones que afectan la densidad del mallado en la ecuación de Poisson. A partir de dicha configuración se varía un parámetro a la vez, de todos los previamente mencionados, con la idea de mostrar el efecto que cada uno de ellos tiene sobre la calidad de la malla resultante.

Buscando la continuidad en los resultados, se opta por hacer el análisis de calidad de la malla para las mallas generadas en la sección anterior en la configuración inicial, es decir, se analizan los mismos cuatro casos, malla C con un único perfil y con un perfil con flap, lo mismo para la tipología O, un caso con perfil único y otro con perfil y flap. Se hace el análisis a un ángulo de ataque constante $\alpha = 0^\circ$.

El primer caso corresponde al descrito por la tabla 6.1 y representado por la malla de la figura 6.1. A esta configuración se le realizarán variaciones en sus parámetros con el fin de identificar el efecto de cada uno de ellos en la calidad de la malla. Las figuras 6.11 y 6.12 muestran el efecto de las variables en el aspect ratio, y las figuras 6.13 y 6.14 el efecto en el skew de las celdas. El primer cambio es el aumento de la dimensión M de la malla desde $M = 45$ a $M = 65$. El segundo cambio implica un incremento en el número de nodos en la dirección del eje η , es decir, aumentando el número de nodos desde $N = 45$ a $N = 65$. Por último se cambia el valor de la variable aa_m correspondiente a la ecuación de Poisson, de $aa_m = 20.5$ a $aa_m = 21.5$.

El siguiente caso corresponde al descrito en la tabla 6.2, al cual se le realizan los mismos incrementos, el valor de M cambia de $M = 103$ a $M = 123$ y el valor de aa_m de $aa_m = 20.5$ a $aa_m = 21.5$. Las figuras 6.15 y 6.16 muestran el aspect ratio, mientras que las figuras 6.17 y 6.18 presentan el skew de la malla.

La tabla 6.3 corresponde a una malla tipo C con un perfil NACA 4415, se analizan el aspect ratio y el skew de la misma, haciendo variaciones en los parámetros que la definen. Los cambios realizados son M de $M = 73$ a $M = 91$, N de $N = 45$ a $N = 65$ y aa_m de $aa_m = 20.5$ a $aa_m = 21.5$. Las gráficas de las figuras 6.19 y 6.20 corresponden al efecto que tienen los parámetros en el aspect ratio, mientras que las figuras 6.21 y 6.22 corresponden al skew de las celdas.

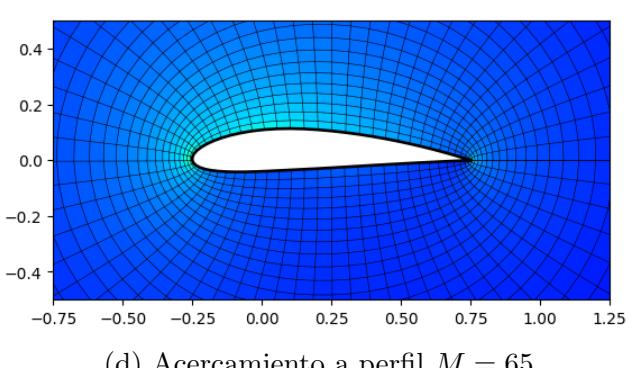
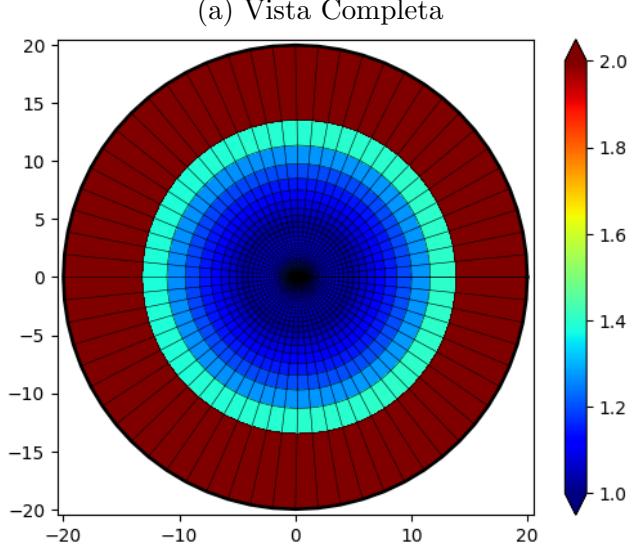
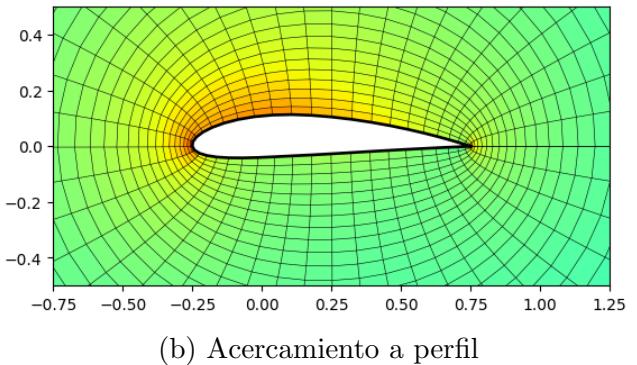
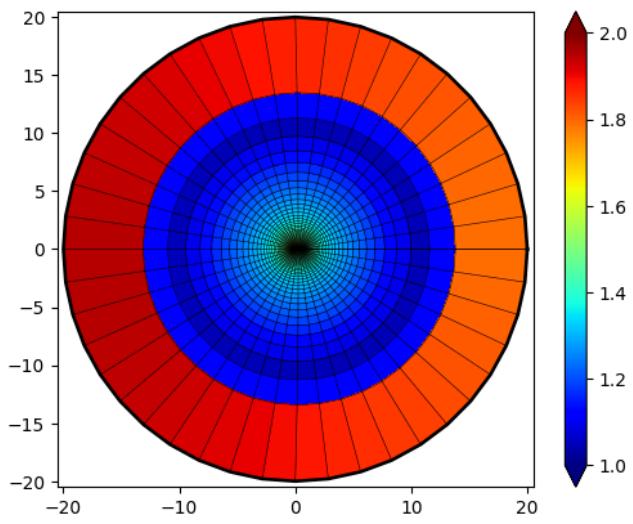


Figura 6.11: Aspect ratio para malla tipo O con perfil NACA 4415

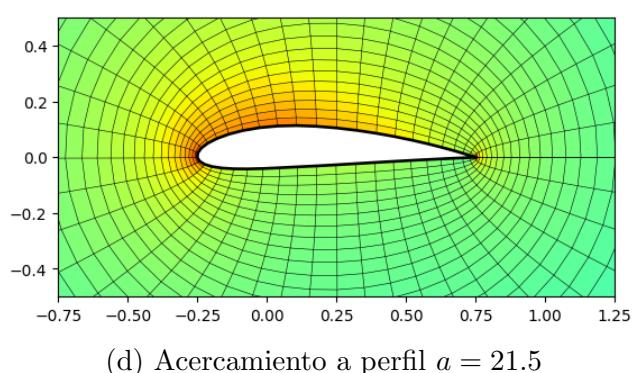
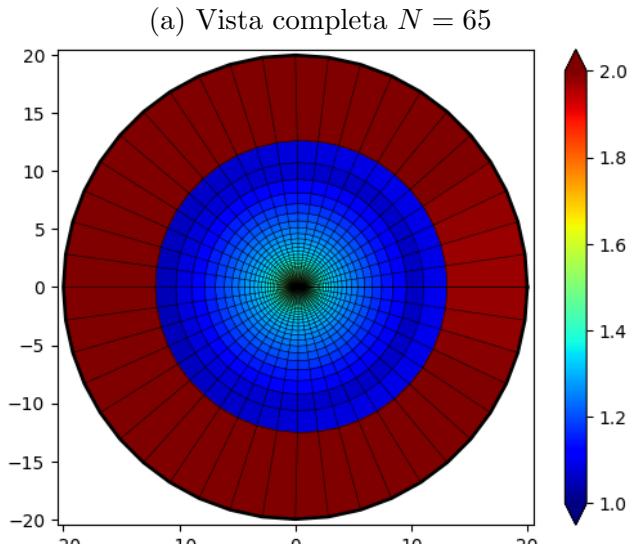
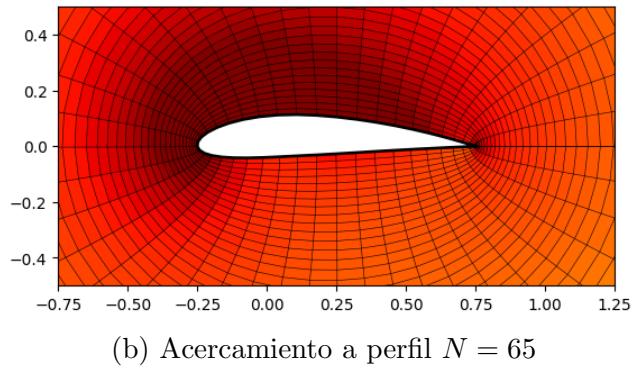
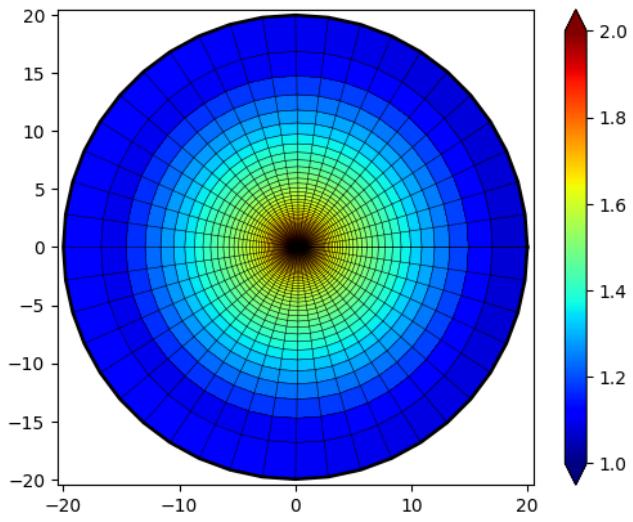
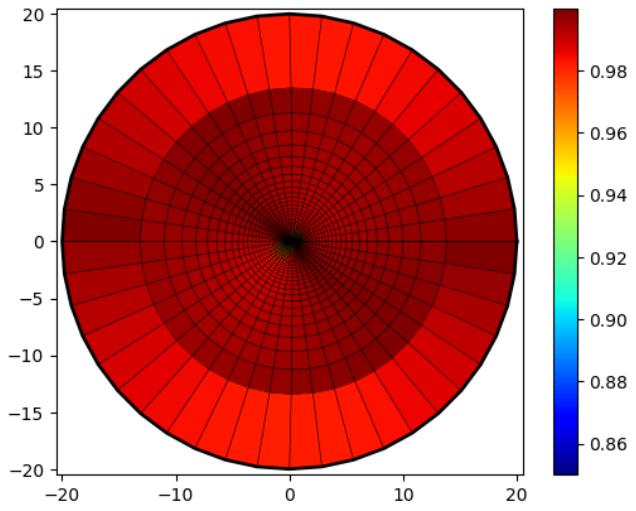
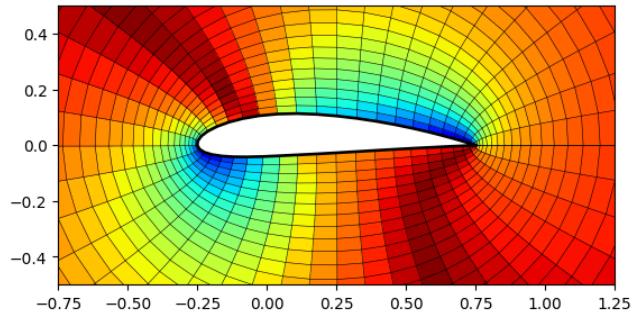


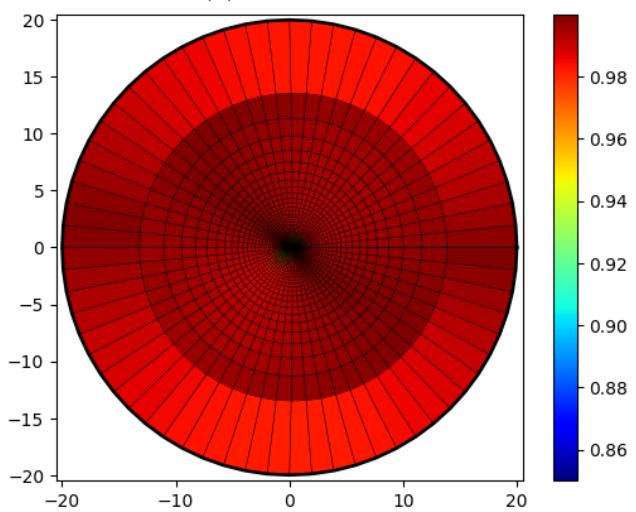
Figura 6.12: Aspect ratio para malla tipo O con perfil NACA 4415



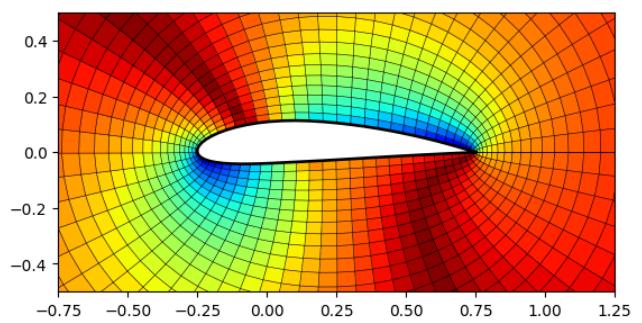
(a) Vista completa



(b) Acercamiento a perfil

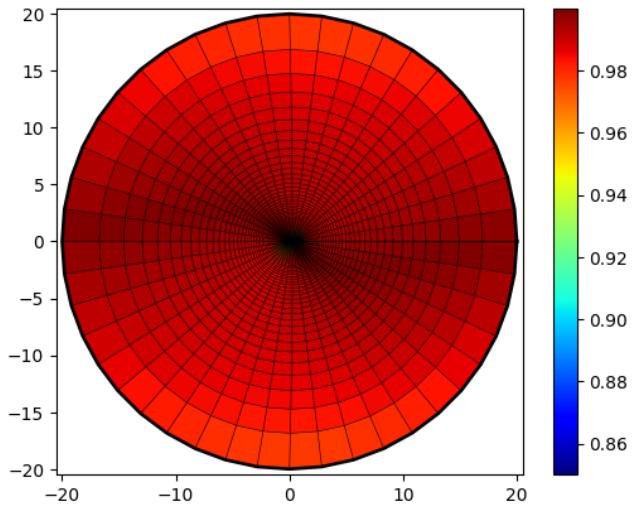


(c) Vista completa $M = 65$

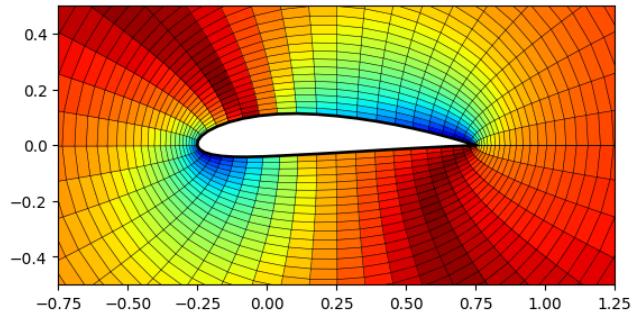


(d) Acercamiento a perfil $M = 65$

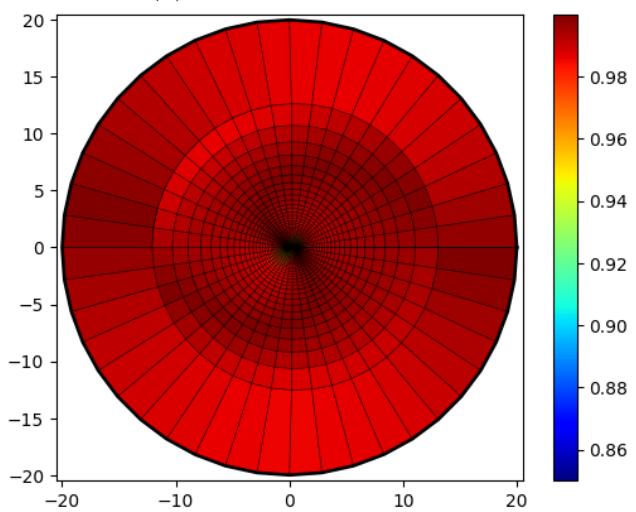
Figura 6.13: Skew pata malla tipo O con perfil NACA 4415



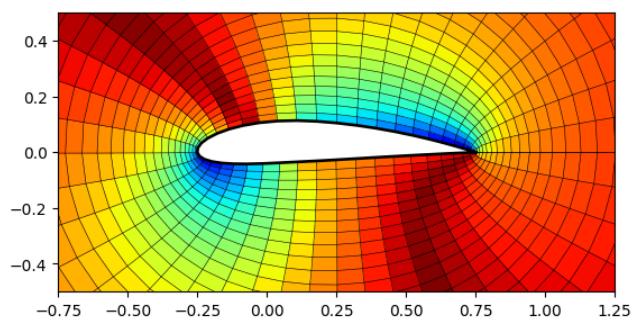
(a) Vista completa $N = 65$



(b) Acercamiento a perfil $N = 65$

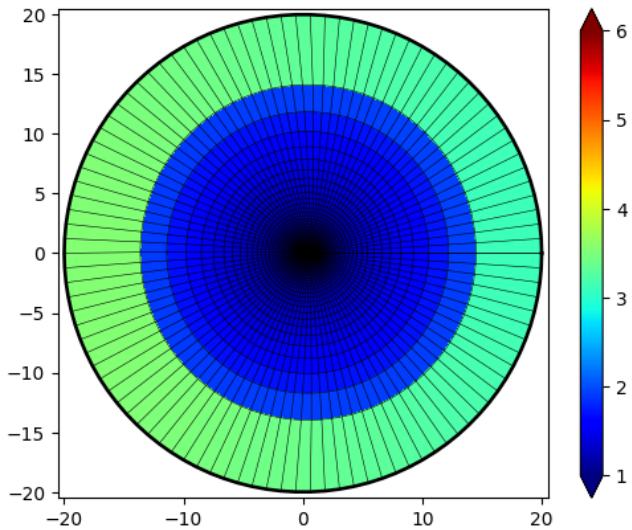


(c) Vista completa $a = 21.5$

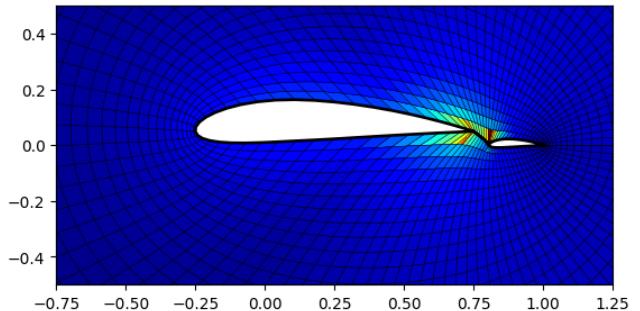


(d) Acercamiento a perfil $a = 21.5$

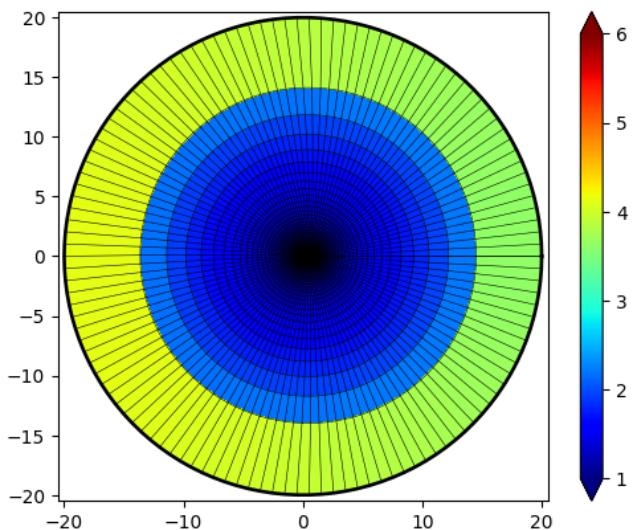
Figura 6.14: Skew para malla tipo O con perfil NACA 4415



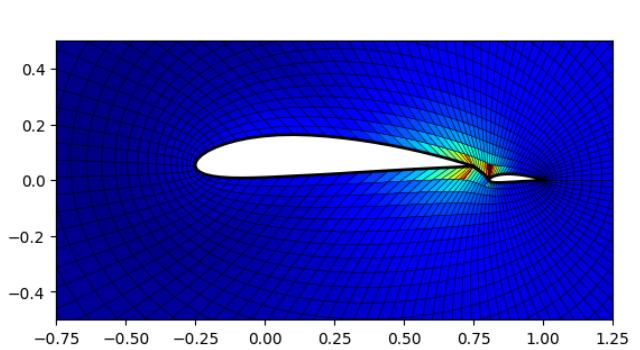
(a) Vista Completa



(b) Acercamiento a perfil



(c) Vista completa $M = 123$



(d) Acercamiento a perfil $M = 123$

Figura 6.15: Aspect ratio para malla tipo O con perfil y flap NACA 4415

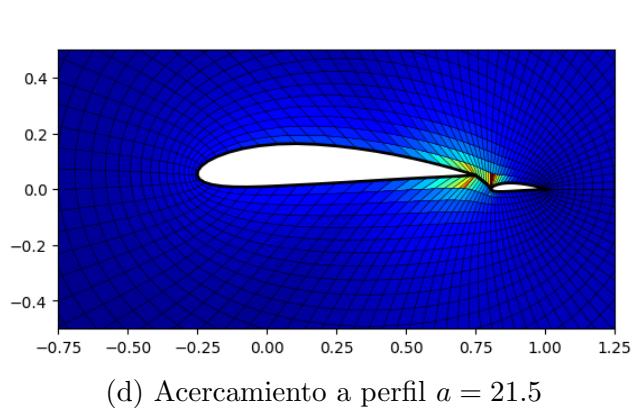
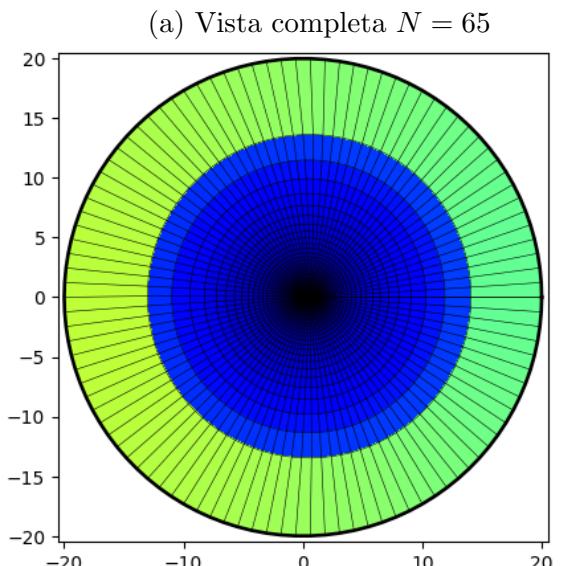
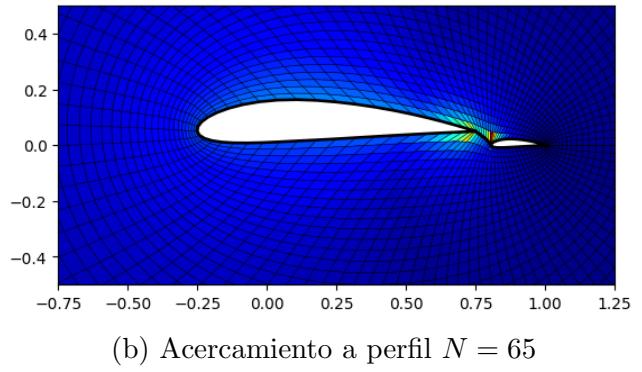
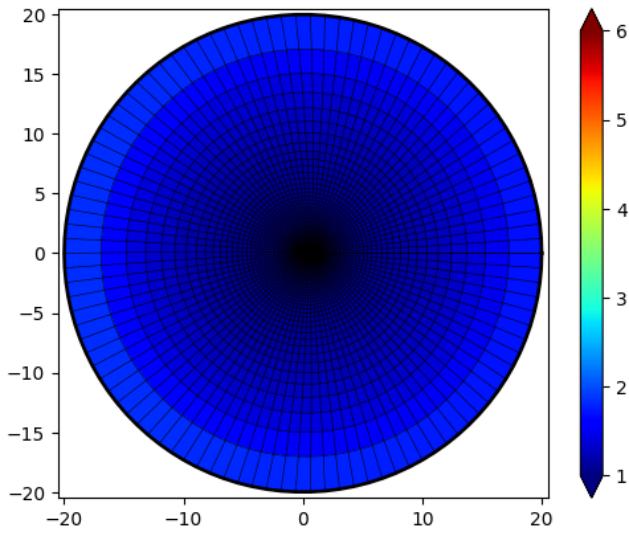
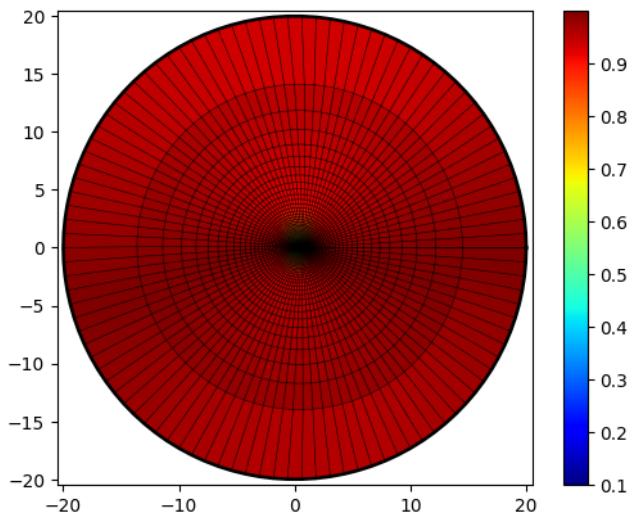
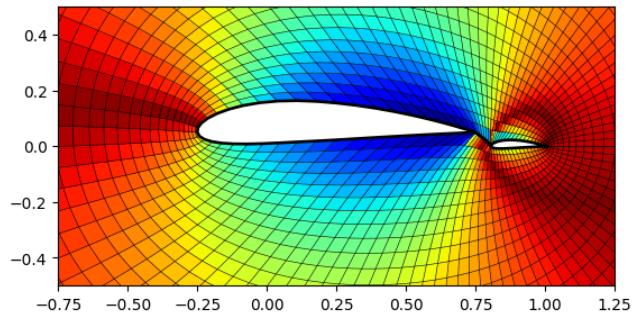


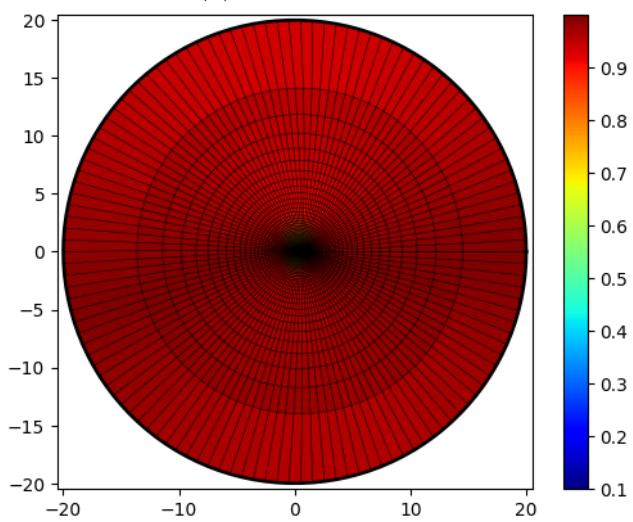
Figura 6.16: Aspect ratio para malla tipo O con perfil y flap NACA 4415



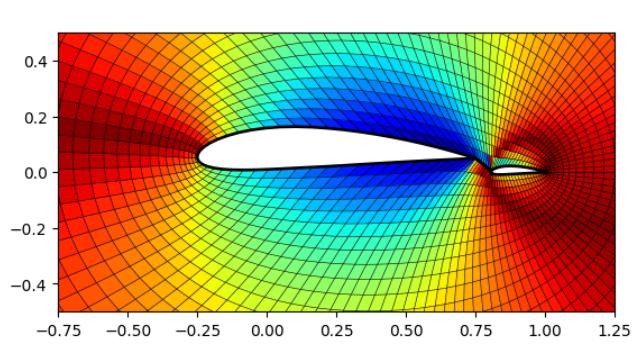
(a) Vista Completa



(b) Acercamiento a perfil

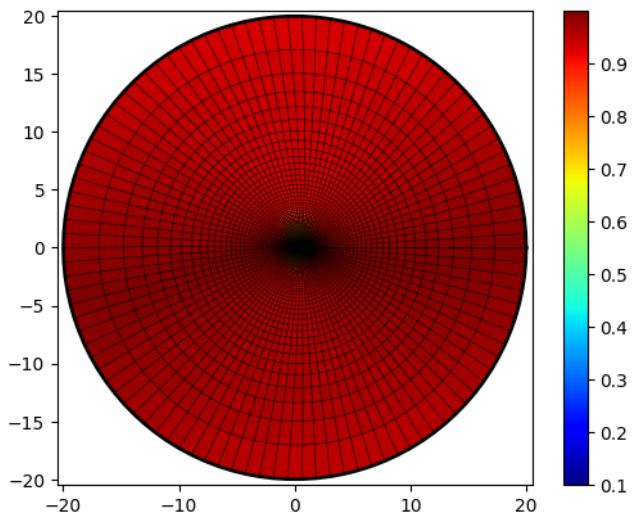


(c) Vista completa $M = 123$

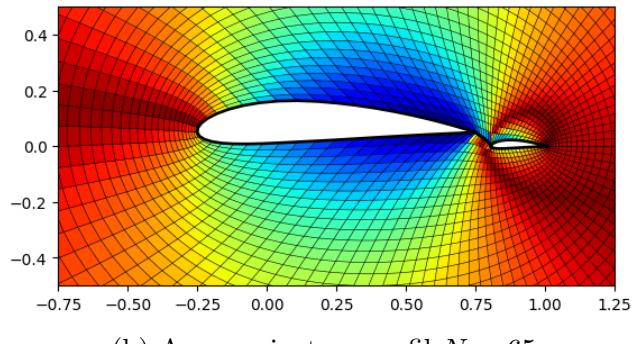


(d) Acercamiento a perfil $M = 123$

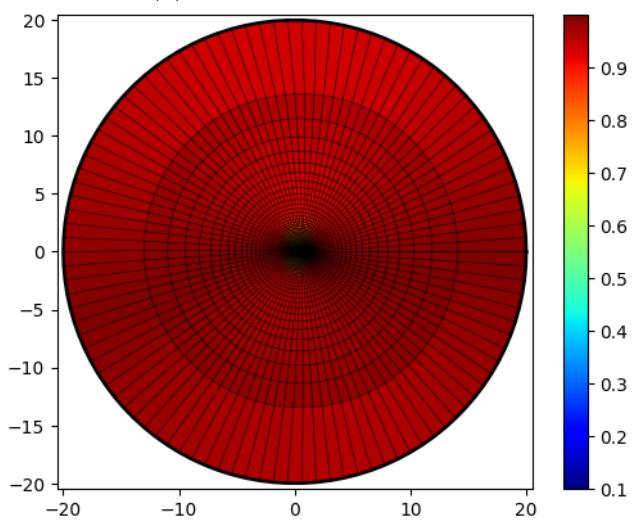
Figura 6.17: Skew para malla tipo O con perfil y flap NACA 4415



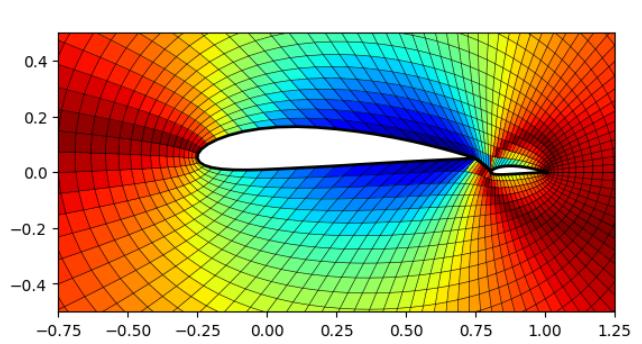
(a) Vista completa $N = 65$



(b) Acercamiento a perfil $N = 65$

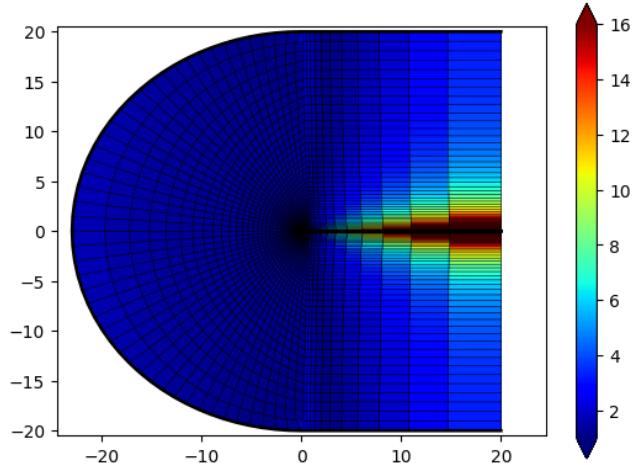


(c) Vista completa $a = 21.5$

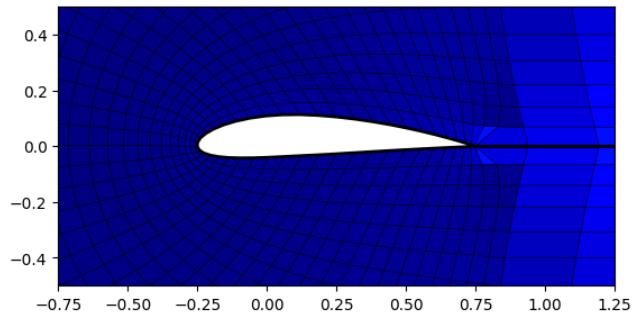


(d) Acercamiento a perfil $a = 21.5$

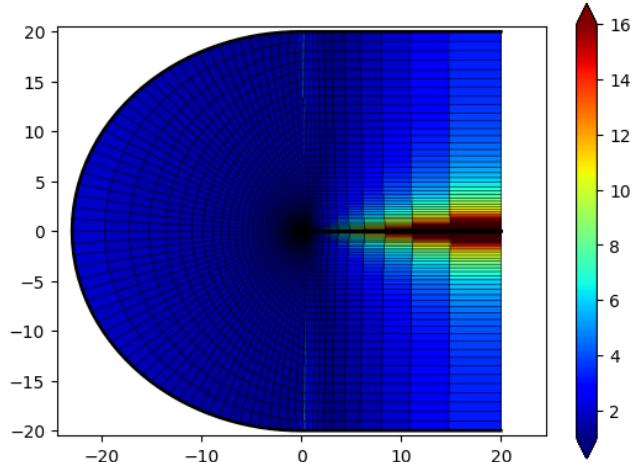
Figura 6.18: Skew para malla tipo O con perfil y flap NACA 4415



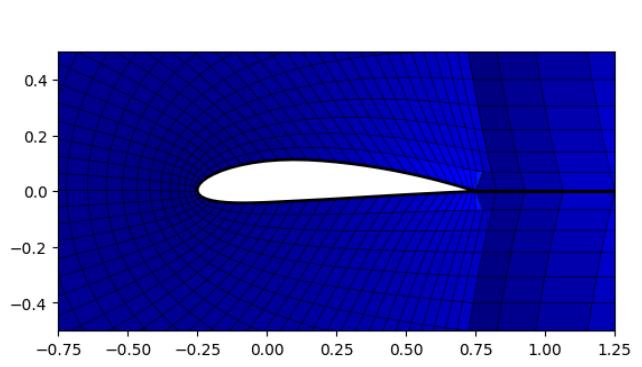
(a) Vista Completa



(b) Acercamiento a perfil

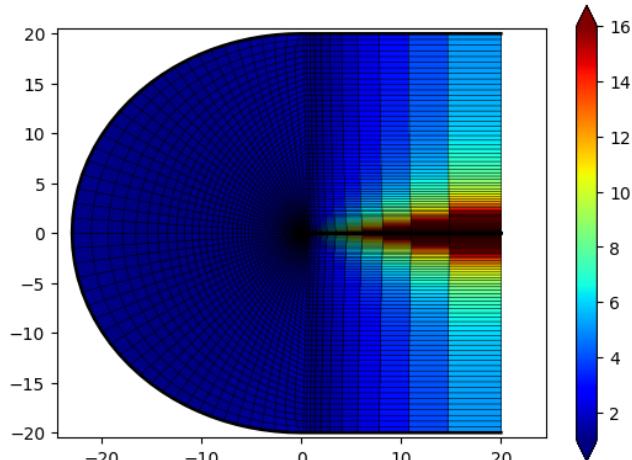


(c) Vista completa $M = 91$

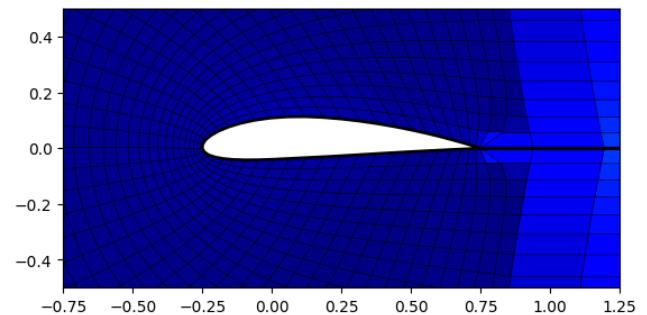


(d) Acercamiento a perfil $M = 91$

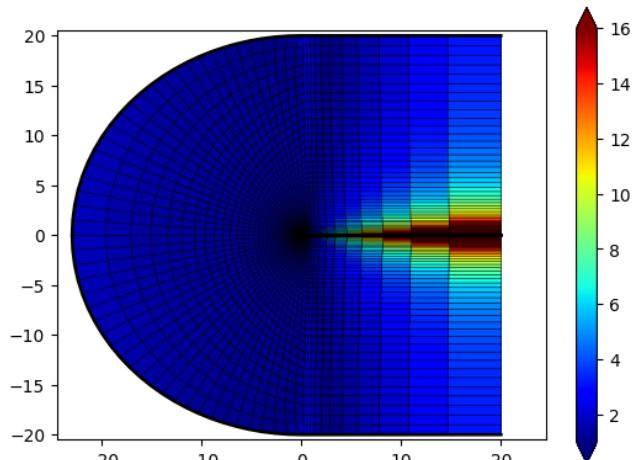
Figura 6.19: Aspect ratio para malla tipo C con perfil NACA 4415



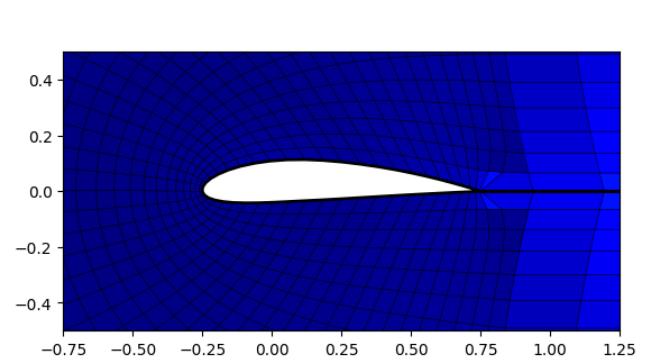
(a) Vista completa $N = 65$



(b) Acercamiento a perfil $N = 65$

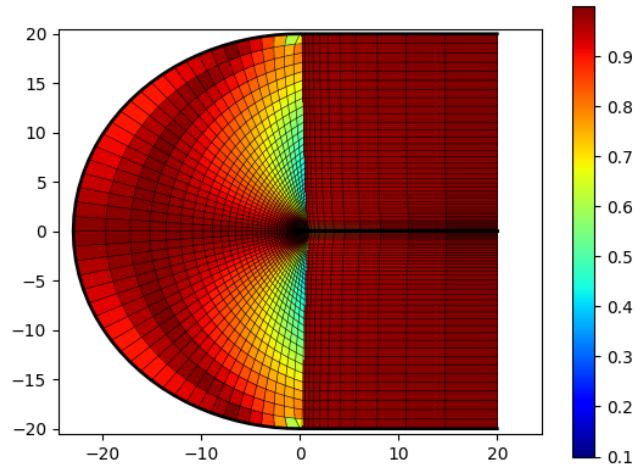


(c) Vista completa $a = 21.5$

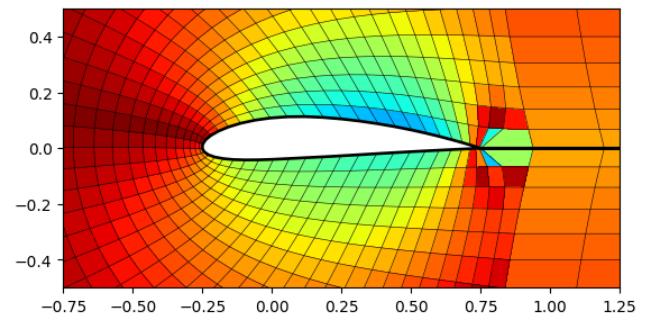


(d) Acercamiento a perfil $a = 21.5$

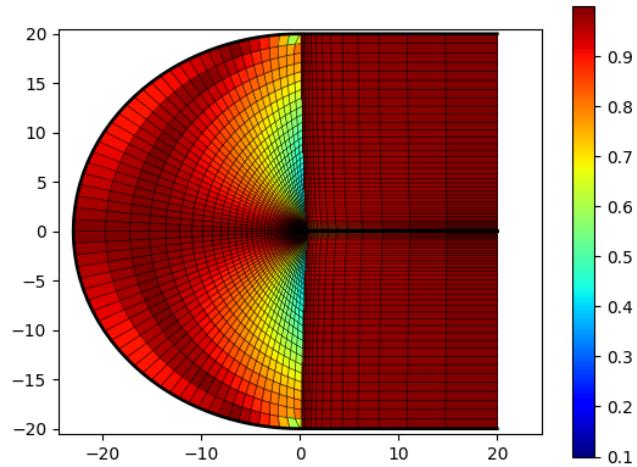
Figura 6.20: Aspect ratio para malla tipo C con perfil NACA 4415



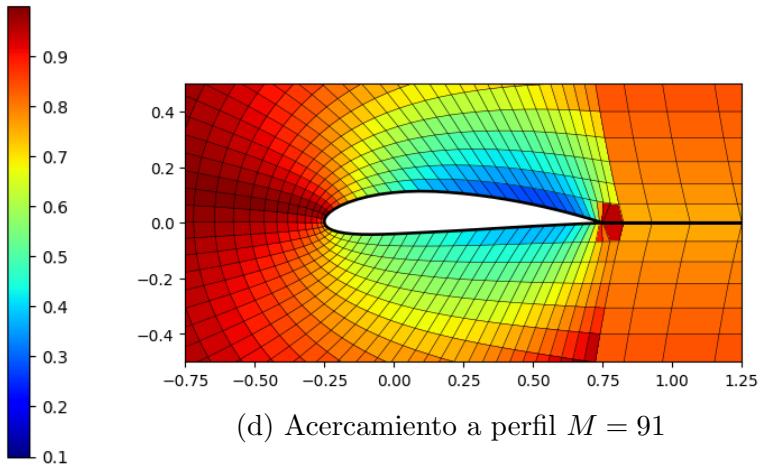
(a) Vista completa



(b) Acercamiento a perfil

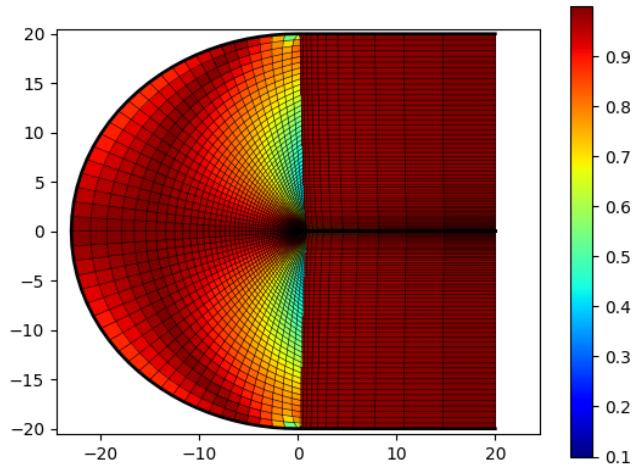


(c) Vista completa $M = 91$

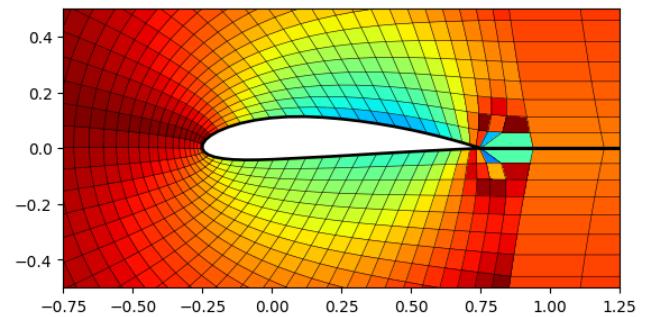


(d) Acercamiento a perfil $M = 91$

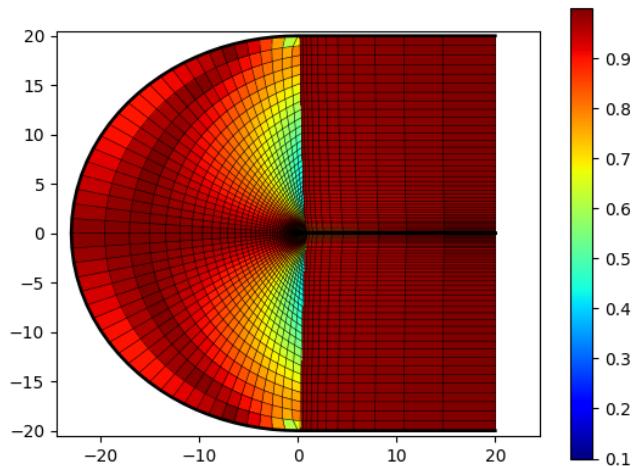
Figura 6.21: Skew para malla tipo C con perfil NACA 4415



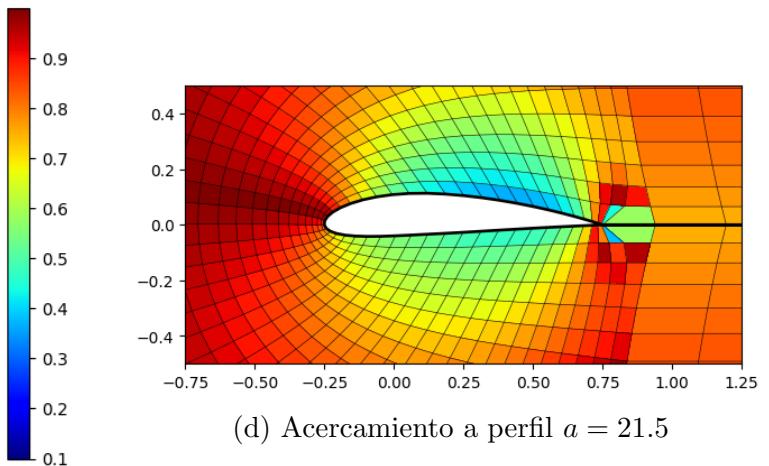
(a) Vista completa $N = 65$



(b) Acercamiento a perfil $N = 65$



(c) Vista completa $a = 21.5$



(d) Acercamiento a perfil $a = 21.5$

Figura 6.22: Skew para malla tipo C con perfil NACA 4415

La tabla 6.4 corresponde a una malla tipo C con perfil y flap, ambos NACA 4415, los resultados para el aspect ratio y el skew de ésta para diferentes configuraciones se muestran en las figuras 6.23, 6.24 y 6.25, 6.26 respectivamente. Los cambios realizados son M de $M = 163$ a $M = 193$, N de $N = 45$ a $N = 65$ y aa_m de $aa_m = 20.5$ a $aa_m = 21.5$.

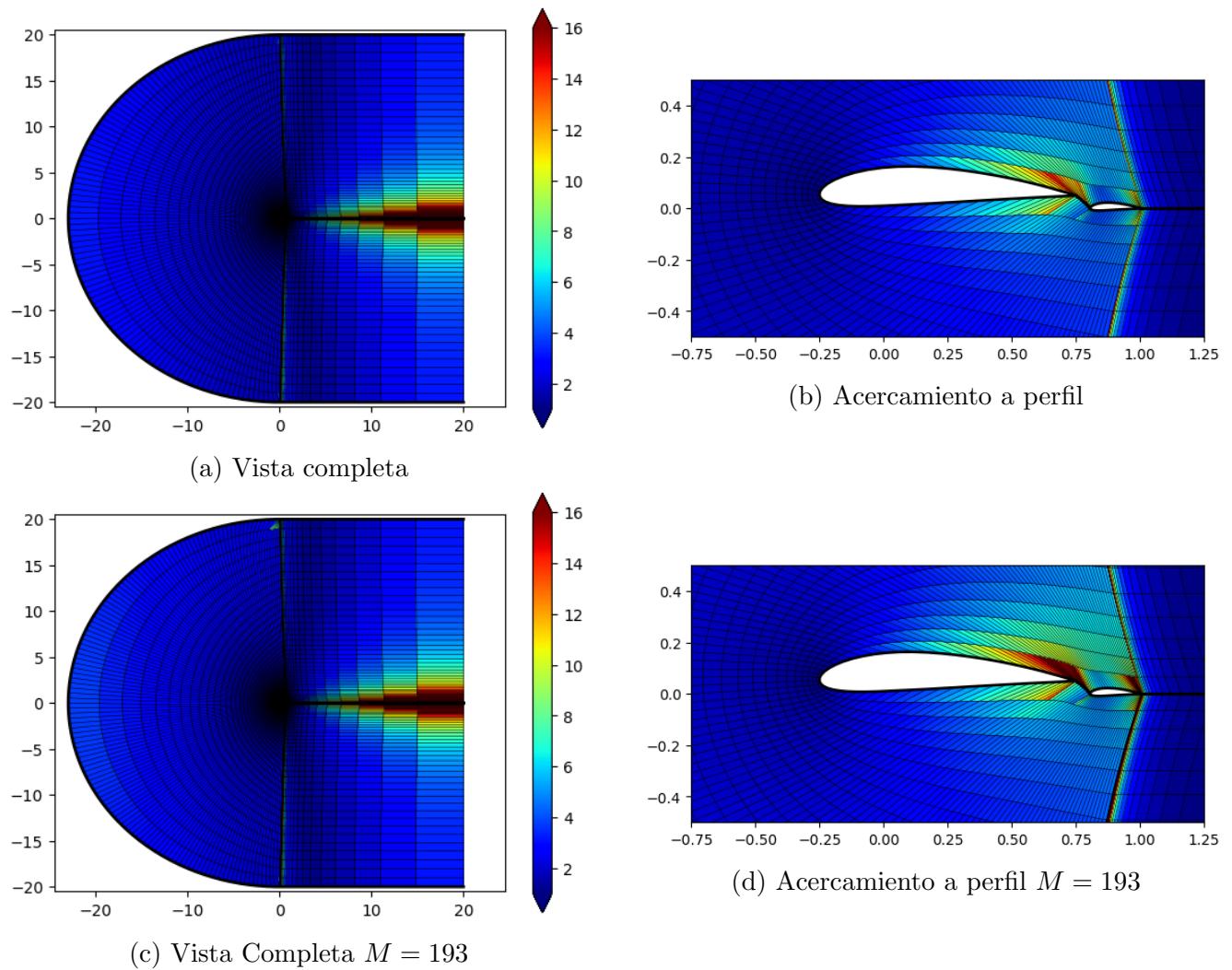
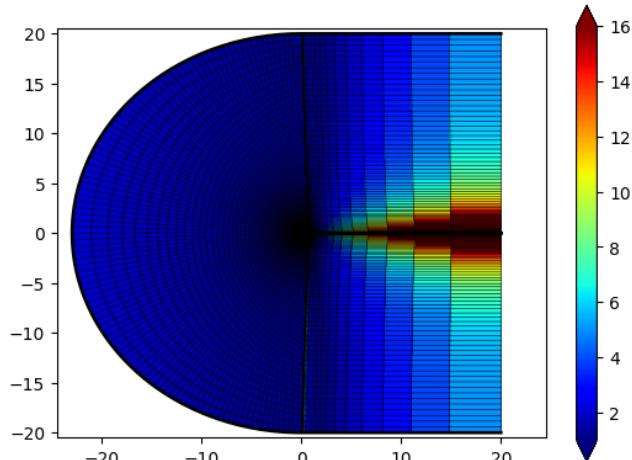
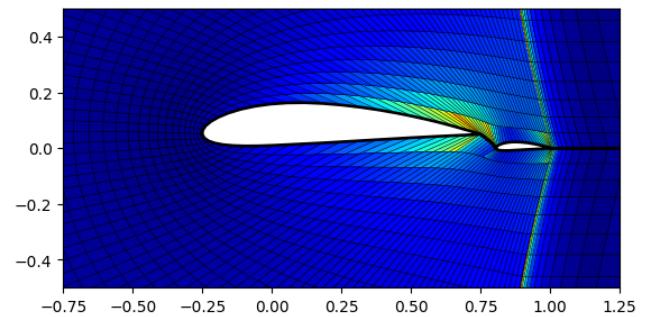


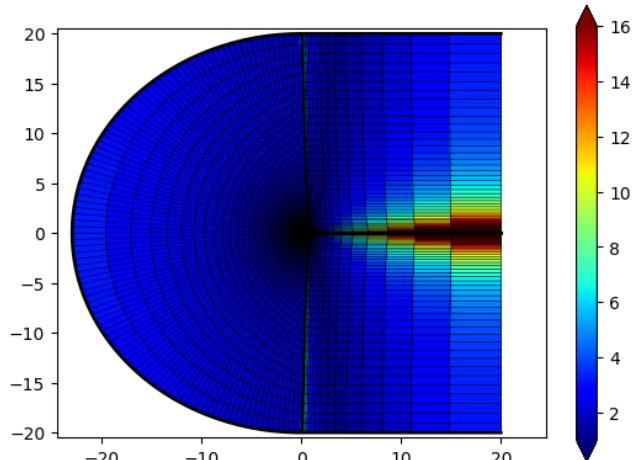
Figura 6.23: Aspect ratio para malla tipo C con perfil y flap NACA 4415



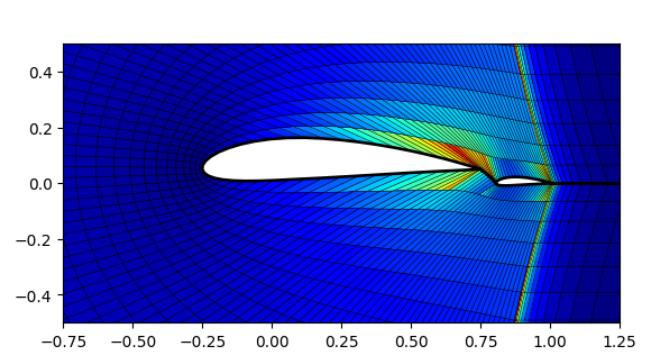
(a) Vista completa $N = 65$



(b) Acercamiento a perfil $N = 65$

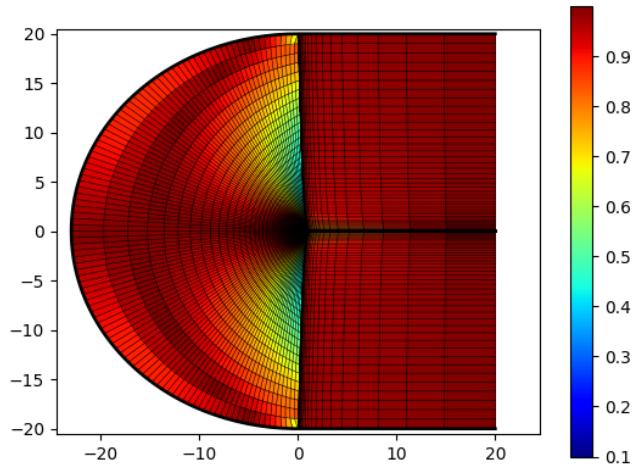


(c) Vista completa $a = 21.5$

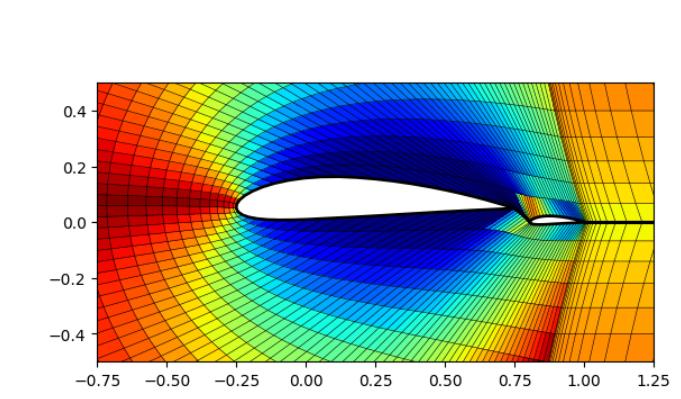


(d) Acercamiento a perfil $a = 21.5$

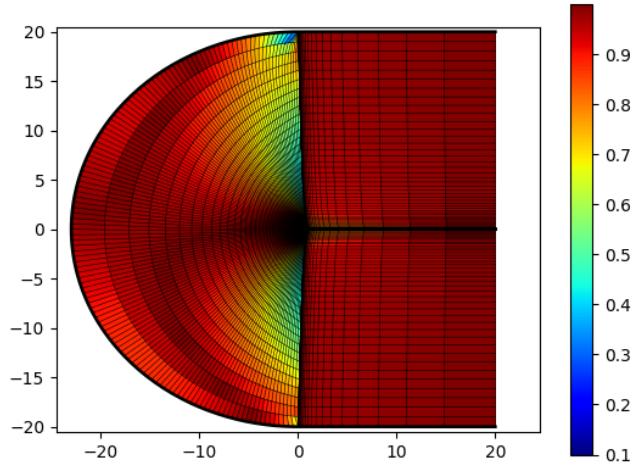
Figura 6.24: Aspect ratio para malla tipo C con perfil y flap NACA 4415



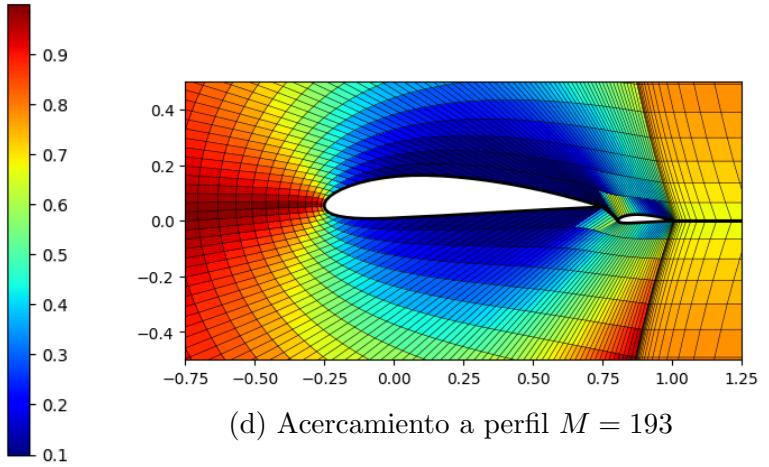
(a) Vista completa



(b) Acercamiento a perfil

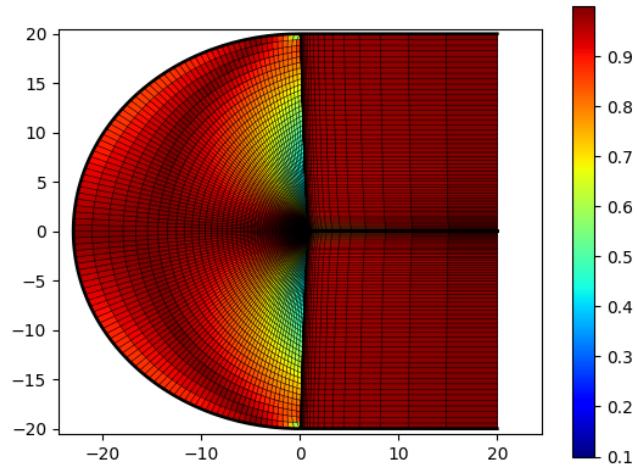


(c) Vista completa $M = 193$

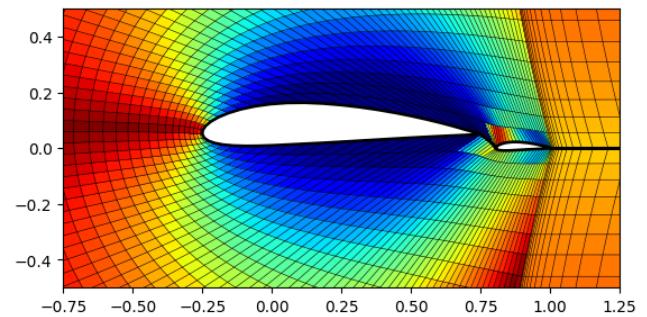


(d) Acercamiento a perfil $M = 193$

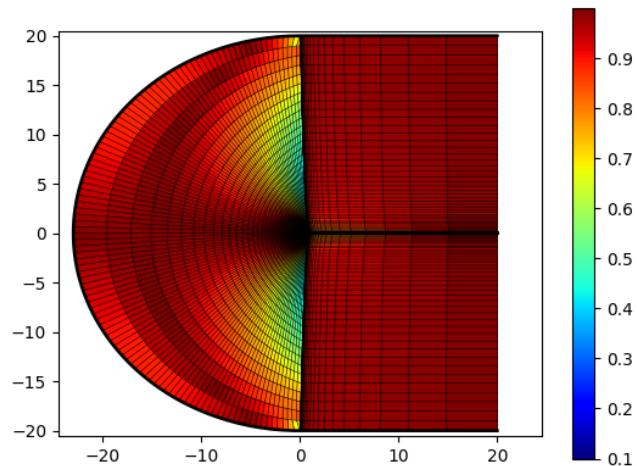
Figura 6.25: Skew para malla tipo C con perfil y flap NACA 4415



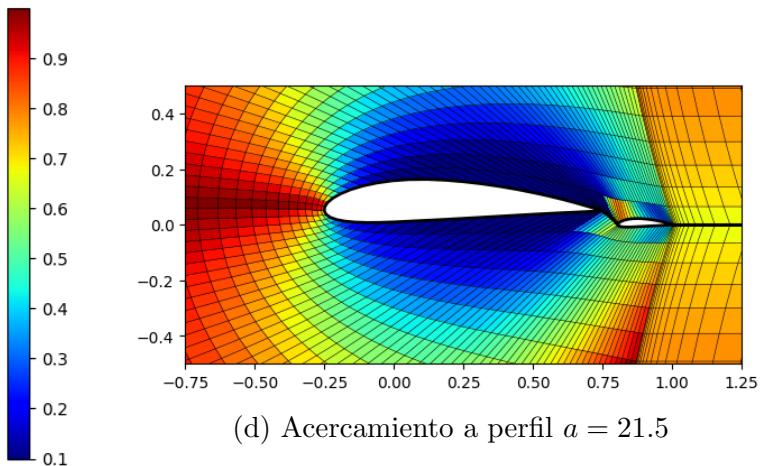
(a) Vista completa $N = 65$



(b) Acercamiento a perfil $N = 65$



(c) Vista completa $a = 21.5$



(d) Acercamiento a perfil $a = 21.5$

Figura 6.26: Skew para malla tipo C con perfil y flap NACA 4415

6.3. Análisis de DFC

En este apartado se muestran los resultados obtenidos en diferentes simulaciones para las cuales se ocuparon mallas generadas mediante el uso de los códigos desarrollados como parte de este proyecto. En primera instancia se llevan a cabo análisis de flujo potencial, tema que se ha desarrollado en el capítulo 4. Por otro lado, se llevaron a cabo simulaciones para flujos turbulentos y estacionario en el programa de código abierto SU2.

6.3.1. Flujo Potencial

En el análisis de flujo potencial se analizó el flujo alrededor de un perfil aerodinámico NACA 0012 de la serie de cuatro dígitos, con cuerda unitaria $c = 1$ m. La simulación se llevó a cabo mediante la generación de una malla tipo O, y para un intervalo de ángulos de ataque desde $\alpha = -4^\circ$ hasta $\alpha = 10^\circ$, con incrementos de 2° . La información del flujo se presenta en la tabla 6.5.

La figura 6.27 muestra la malla utilizada para el análisis previamente descrito, la cual es una malla de dimensiones $M = 229$ y $N = 649$, y es utilizada para todos los ángulos de ataque estudiados, ya que lo único que se modifica con el cambio en el ángulo de ataque son las componentes del vector de velocidad. Por su parte la figura 6.28 muestra las líneas equipotenciales obtenidas como resultado de la solución de la ecuación de potencial para un ángulo de ataque $\alpha = 0^\circ$. En ambas imágenes se muestran tanto una vista completa del dominio, como un acercamiento a la zona del perfil aerodinámico.

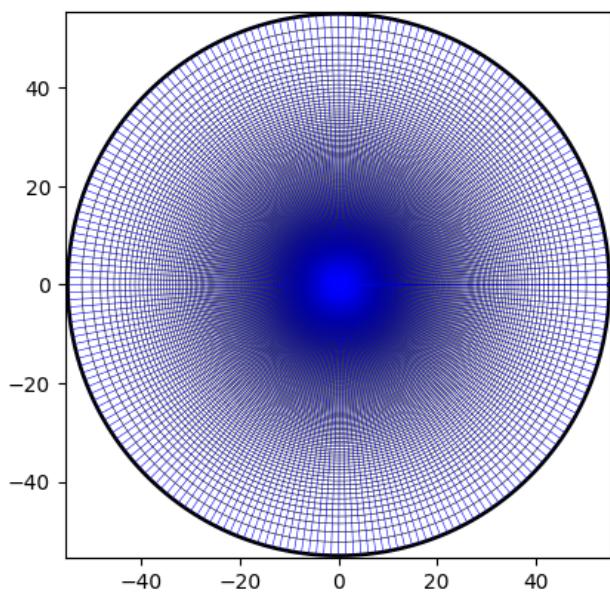
Las figuras 6.29 y 6.30 muestran el campo de presiones generado alrededor del perfil alar para todos los ángulos de ataque estudiados. Por su parte, las figuras 6.31 y 6.32 presentan las líneas de corriente del flujo correspondiente a cada uno de los casos previamente mencionados.

$V_\infty[m/s]$	Re	$P_\infty[Pa]$	$T_\infty[K]$
48	3.1596×10^6	101325	293.15

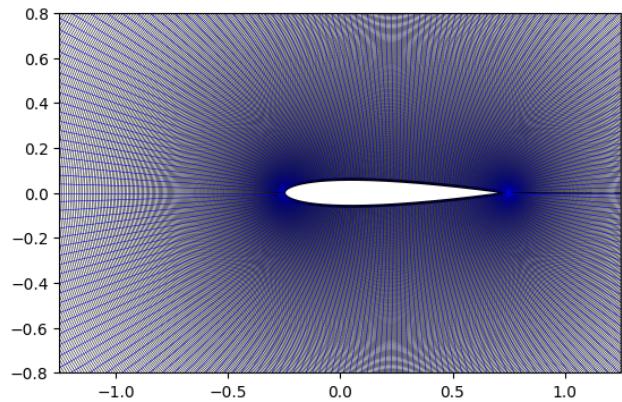
Tabla 6.5: Características de la corriente libre en el análisis de flujo potencial

La figura 6.33 muestra la distribución del coeficiente de presión en el perfil para diferentes ángulos de ataque.

A su vez, la gráfica que se muestra en la figura 6.34 compara los valores del coeficiente de sustentación obtenido para diferentes ángulos de ataque del perfil mediante la integración de los valores del coeficiente de presión a lo largo de la cuerda del mismo, con los datos presentados por Abbott en su libro “Theory of wing sections” [26], hay que recalcar que existen dos principales factores a tomar en cuenta al hacer esta comparación, el primero es la precisión del modelo de flujo potencial, y por otro lado el error de lectura que se pueda presentar al momento de obtener los datos proporcionados en el libro previamente mencionado.

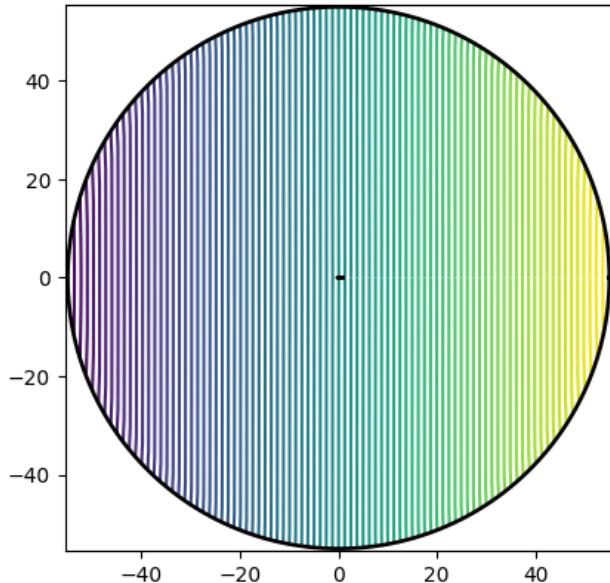


(a) Vista completa

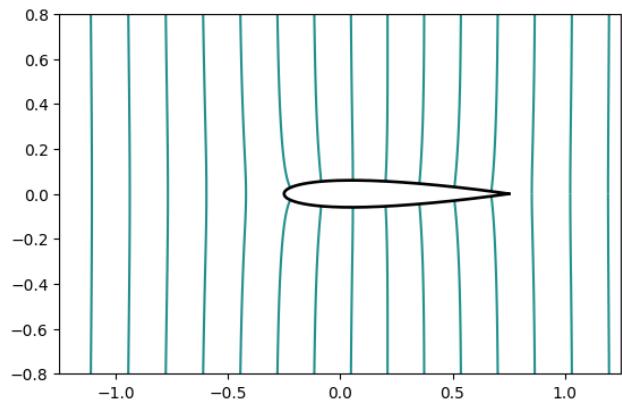


(b) Acercamiento a perfil

Figura 6.27: Malla generada para el análisis de flujo potencial. Perfil NACA 0012



(a) Vista Completa



(b) Acercamiento a perfil

Figura 6.28: Líneas equipotenciales para $\alpha = 0^\circ$ del análisis de flujo potencial de perfil NACA 0012

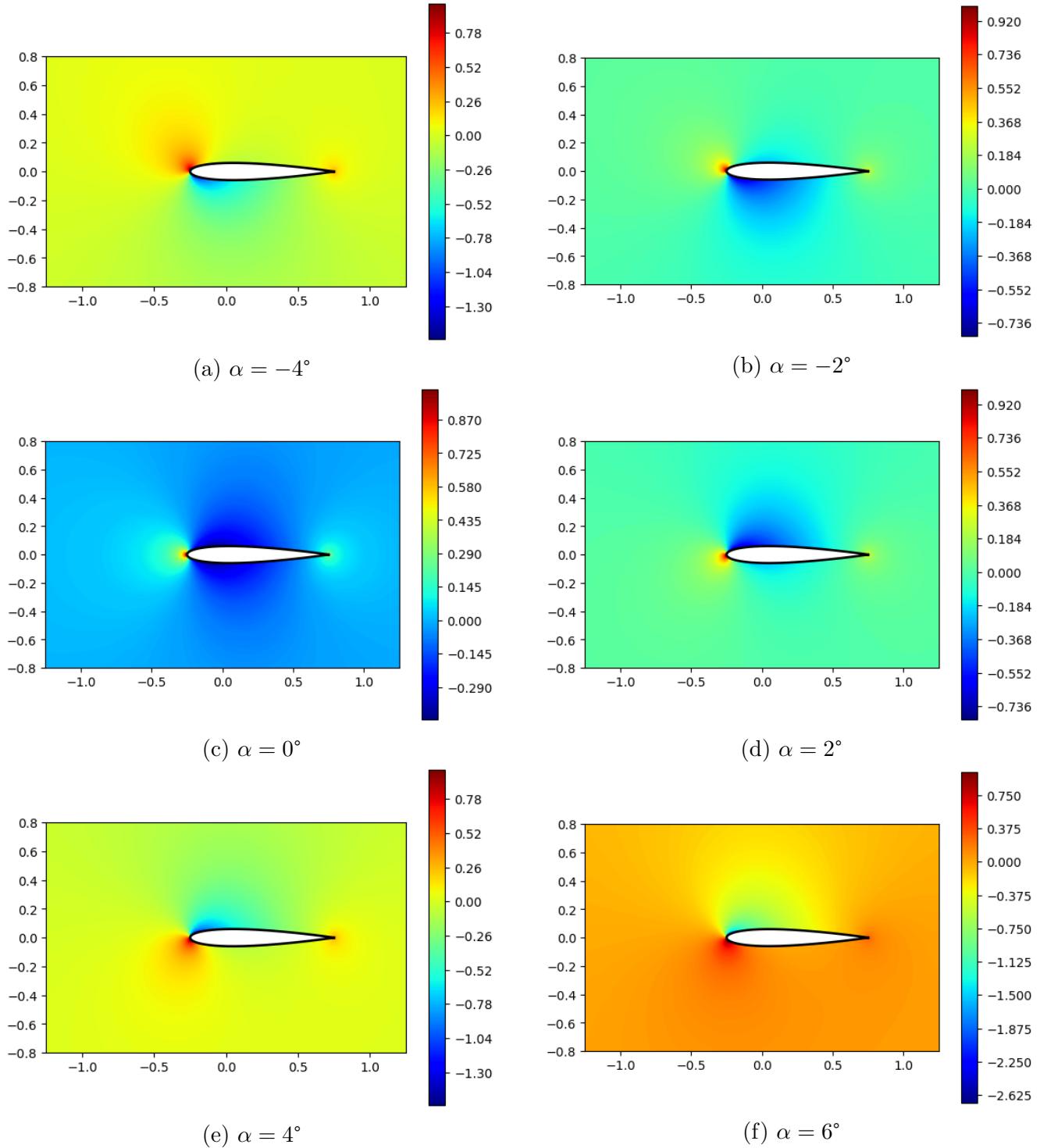
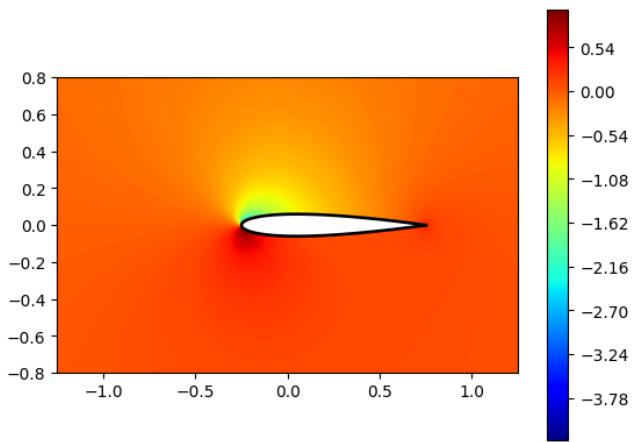
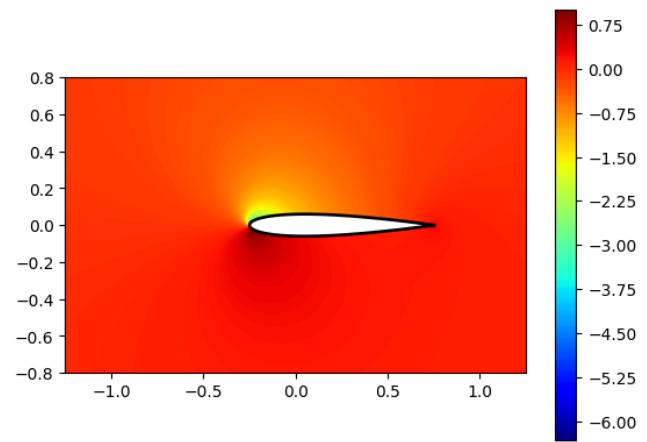


Figura 6.29: Campo de presiones alrededor de perfil NACA 0012

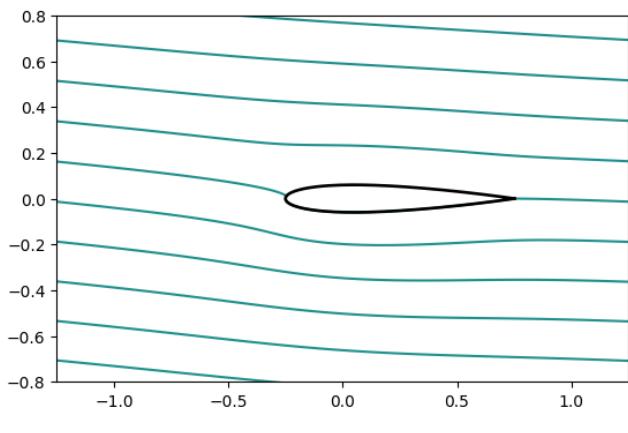


(a) $\alpha = 8^\circ$

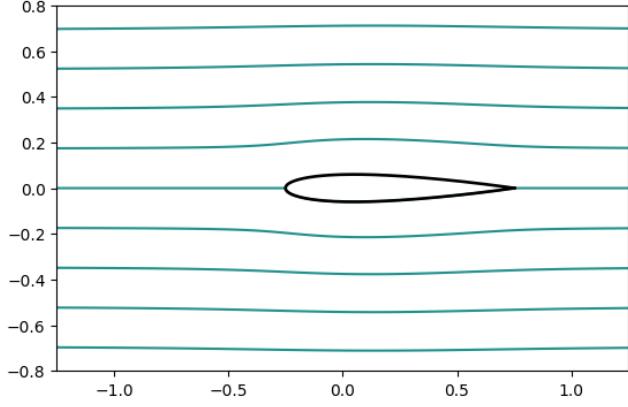


(b) $\alpha = 10^\circ$

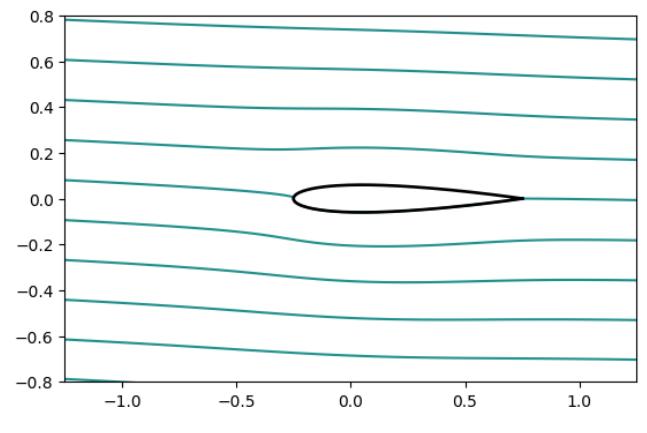
Figura 6.30: Campo de presiones alrededor de perfil NACA 0012



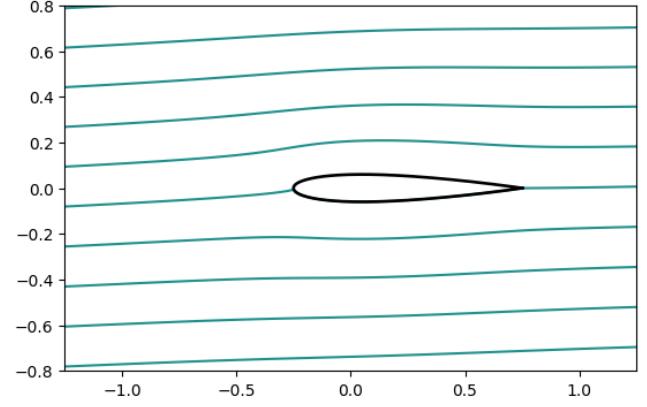
(a) $\alpha = -4^\circ$



(c) $\alpha = 0^\circ$

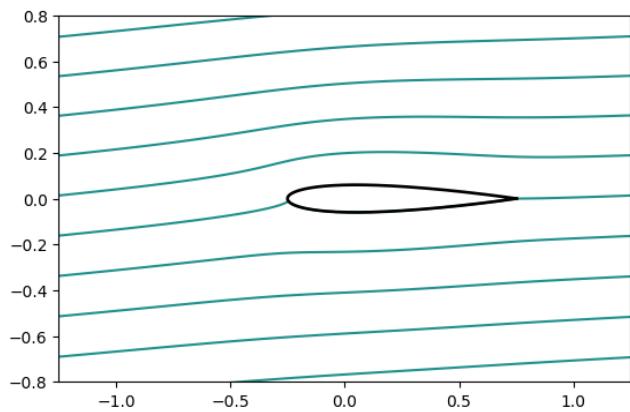


(b) $\alpha = -2^\circ$

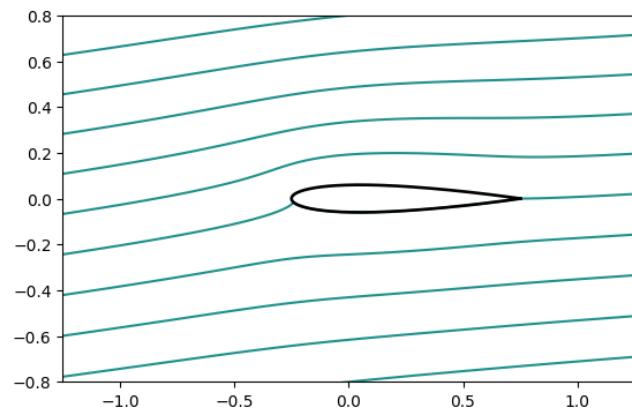


(d) $\alpha = 2^\circ$

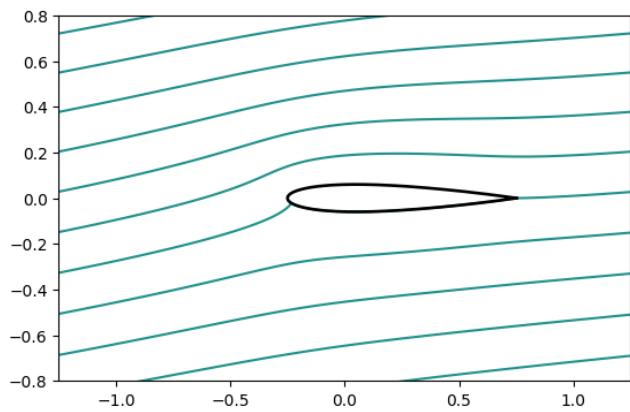
Figura 6.31: Líneas de corriente alrededor de perfil NACA 0012



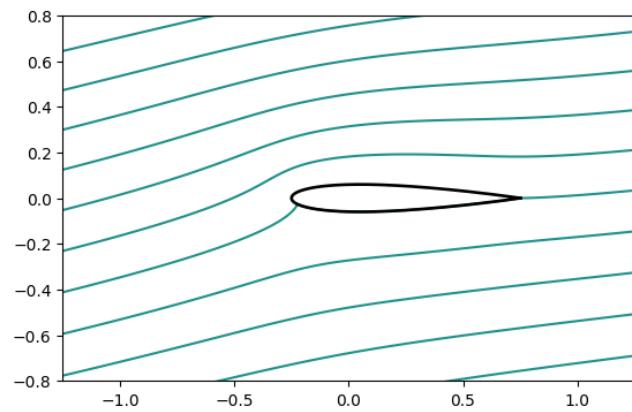
(a) $\alpha = 4^\circ$



(b) $\alpha = 6^\circ$



(c) $\alpha = 8^\circ$



(d) $\alpha = 10^\circ$

Figura 6.32: Líneas de corriente alrededor de perfil NACA 0012

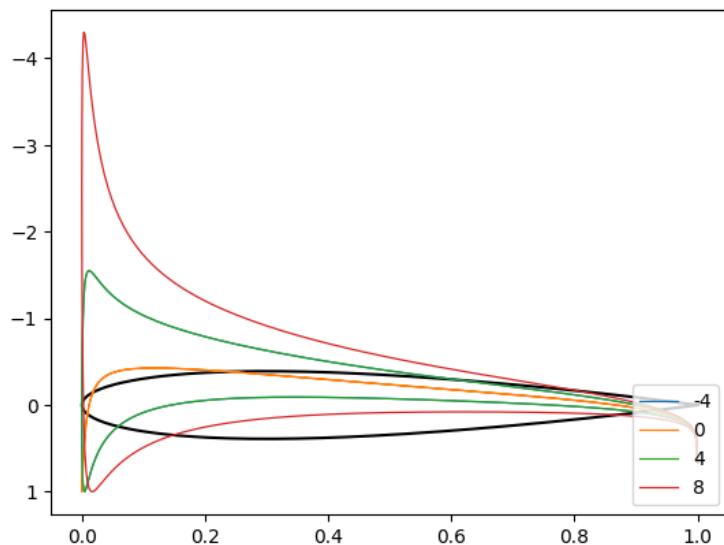


Figura 6.33: Coeficiente de presión a lo largo de la cuerda de perfil NACA 0012 para diferentes ángulos de ataque

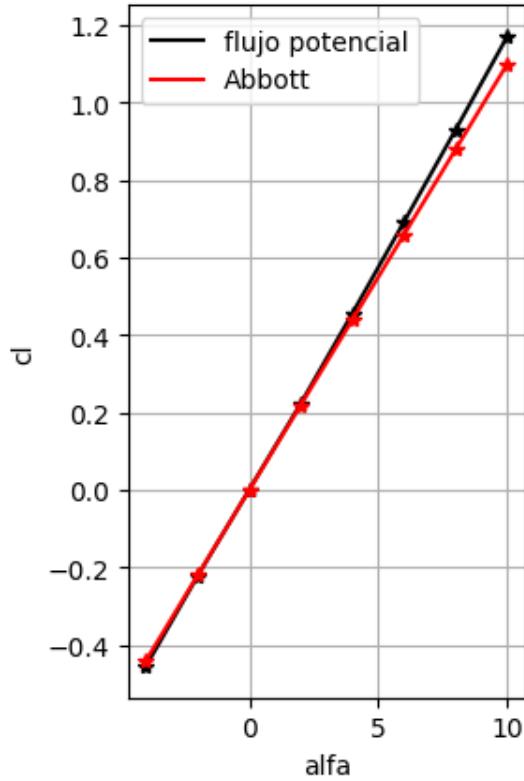


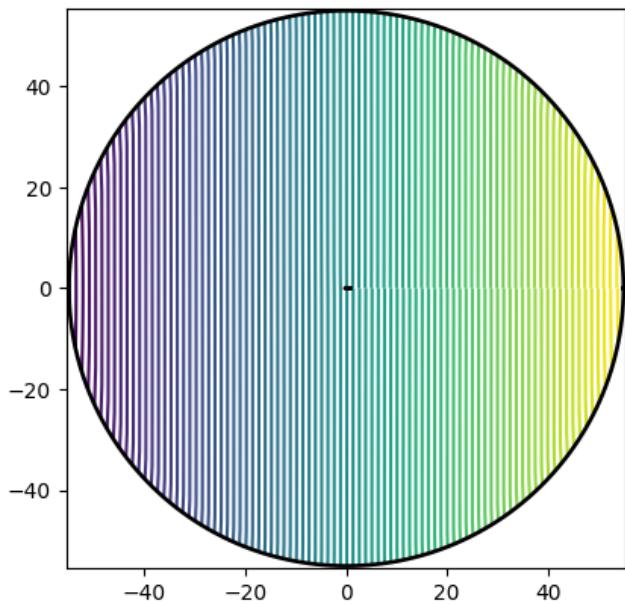
Figura 6.34: Gráfica de coeficiente de sustentación vs. ángulo de ataque para perfil NACA 0012

Se realizó el mismo estudio de flujo potencial para el análisis de un perfil NACA 2412, en el cual las condiciones del flujo en corriente libre se mantienen iguales que para el caso del perfil NACA 0012, y se especifican en la tabla 6.5. Los parámetros para la generación de la malla tampoco han sido modificados, es decir, el único cambio realizado fue la elección del perfil a analizar. Los resultados obtenidos de este análisis se muestran a continuación.

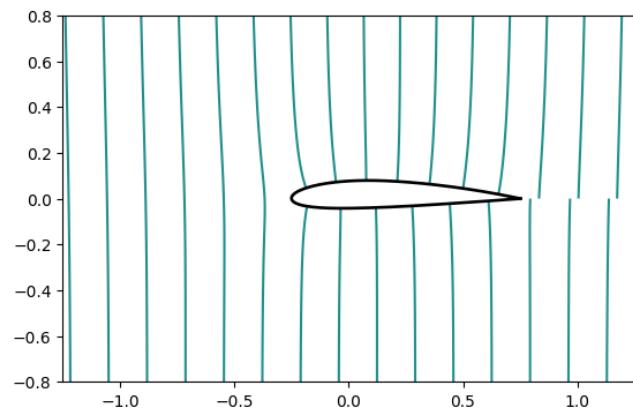
En primera instancia la figura 6.35 presenta las líneas equipotenciales obtenidas mediante la solución de la ecuación de flujo potencial, tanto en vista completa del domino, como un acercamiento al perfil alar. En la vista con acercamiento a la zona donde se ubica el perfil aerodinámico se aprecia la discontinuidad en las líneas equipotenciales a partir del borde de salida del perfil, hecho predecido por la teoría.

A su vez, las figuras 6.36 y 6.37 presentan el campo de presiones que se genera alrededor del perfil para diferentes ángulos de ataque. Por otro lado las figuras 6.38 y 6.39 muestran las líneas de corriente del flujo para los mismos ángulos de ataque.

Por último, la figura 6.40 presenta la distribución del coeficiente de presión a lo largo de la cuerda del perfil aerodinámico para diferentes ángulos de ataque α , y la figura 6.41 muestra la gráfica de la variación del coeficiente de sustentación con el ángulo de ataque del perfil.

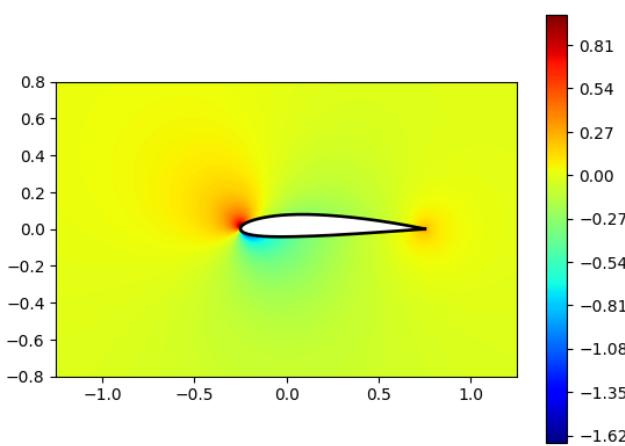


(a) Vista Completa

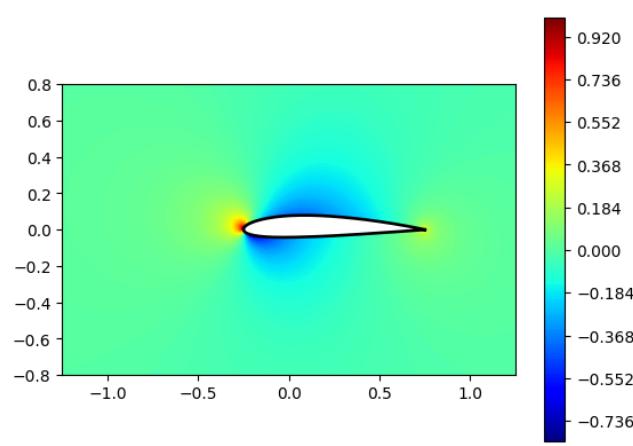


(b) Acercamiento a perfil

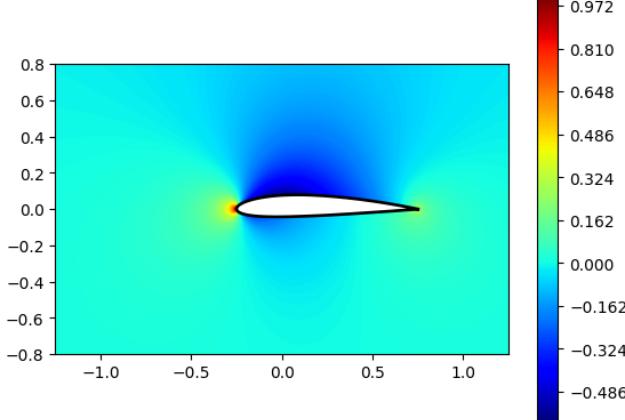
Figura 6.35: Líneas equipotenciales para $\alpha = 0^\circ$ del análisis de flujo potencial de perfil NACA 2412



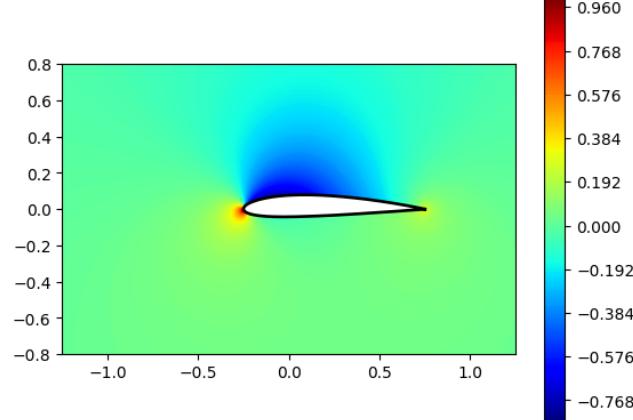
(a) $\alpha = -4^\circ$



(b) $\alpha = -2^\circ$



(c) $\alpha = 0^\circ$



(d) $\alpha = 2^\circ$

Figura 6.36: Campo de presiones alrededor de perfil NACA 2412

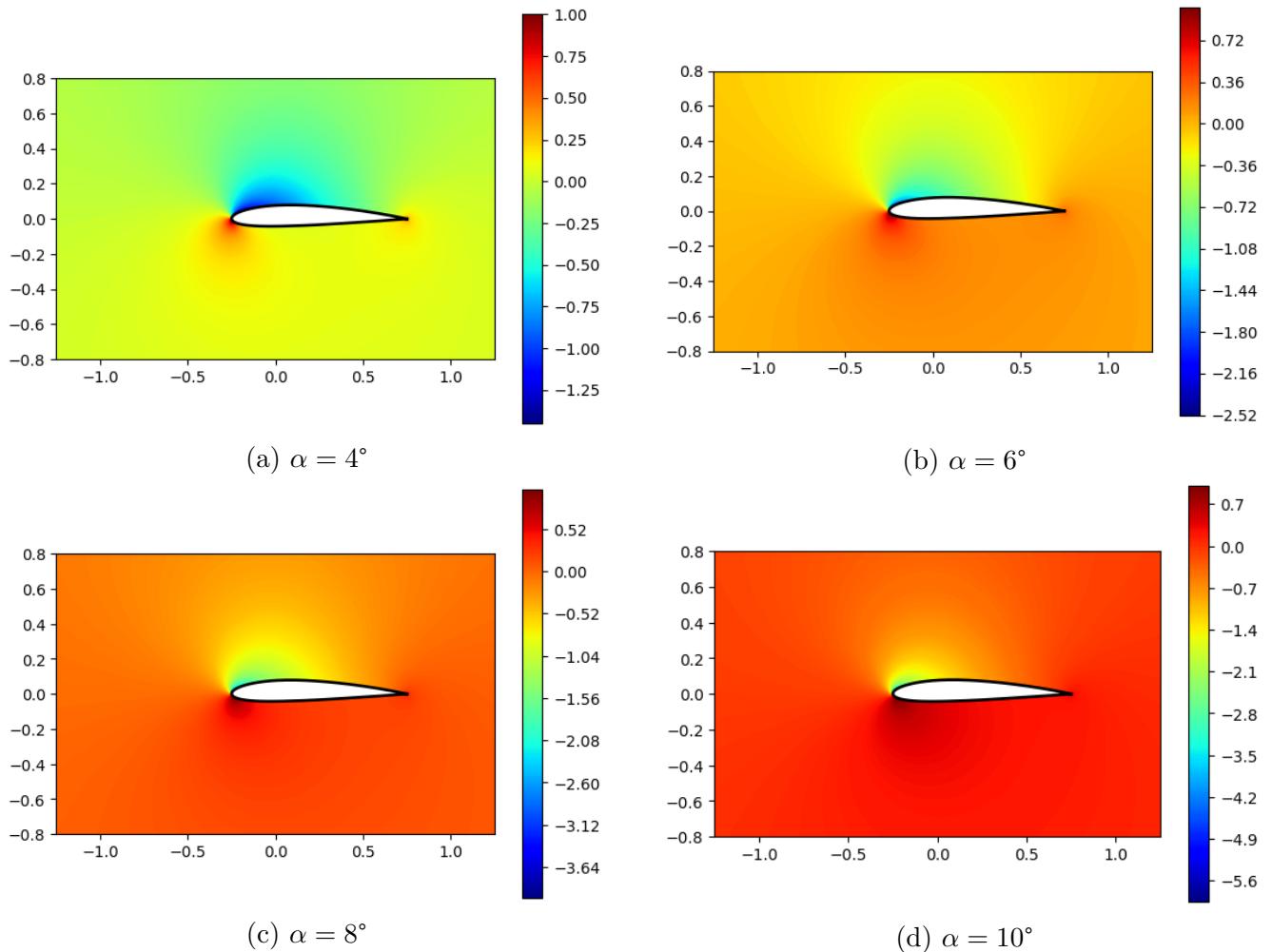


Figura 6.37: Campo de presiones alrededor de perfil NACA 2412

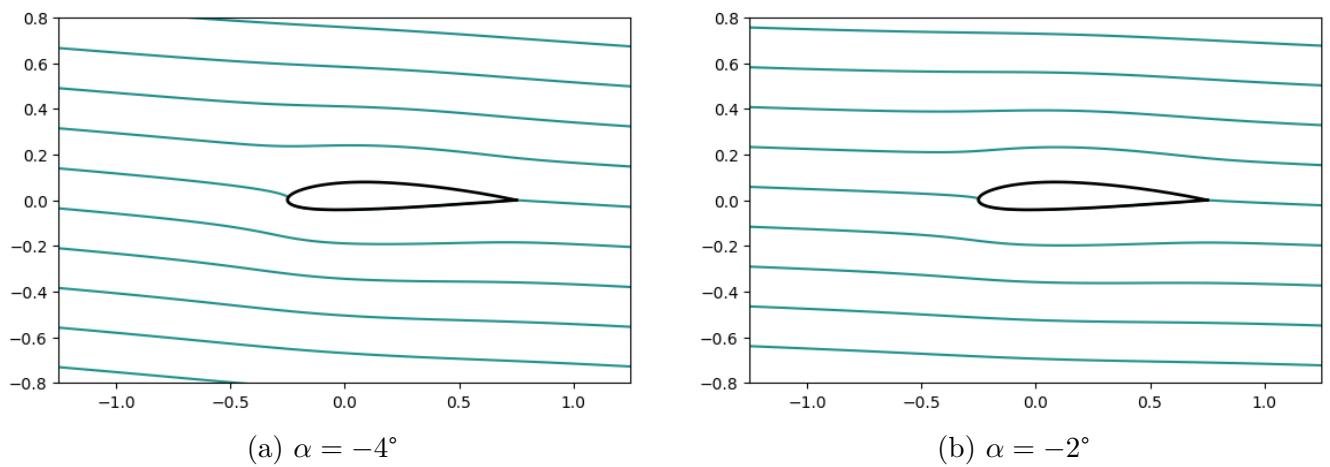
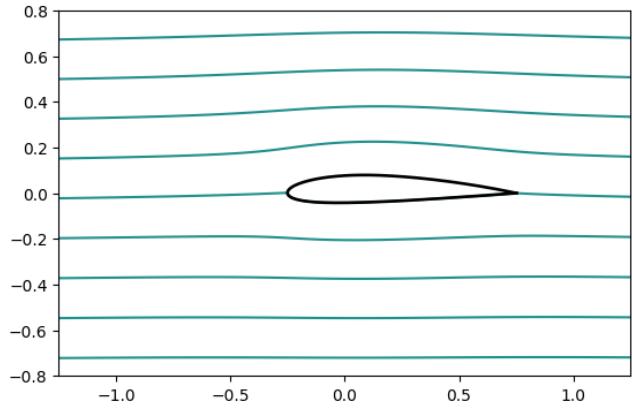
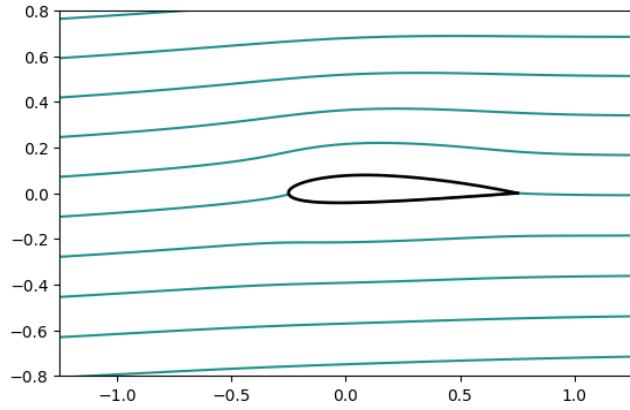


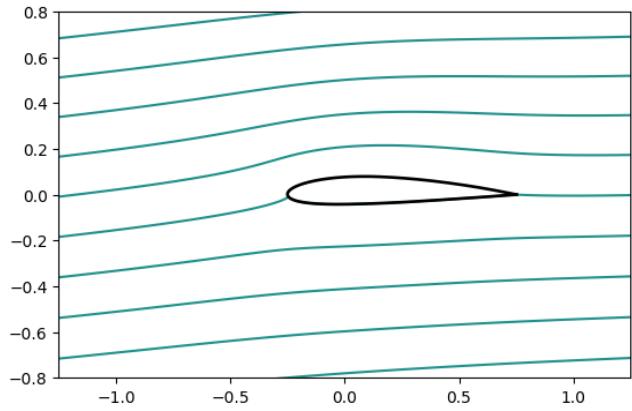
Figura 6.38: Líneas de corriente alrededor de perfil NACA 2412



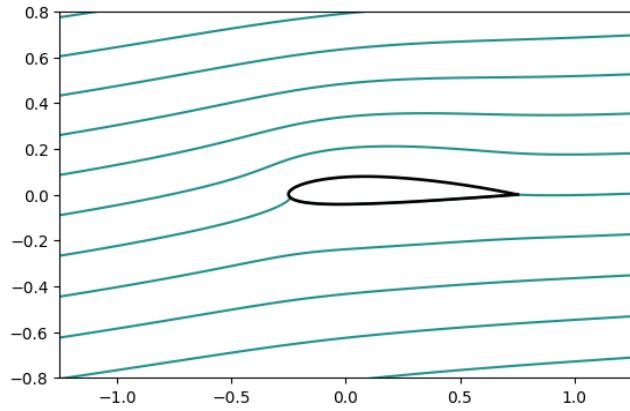
(a) $\alpha = 0^\circ$



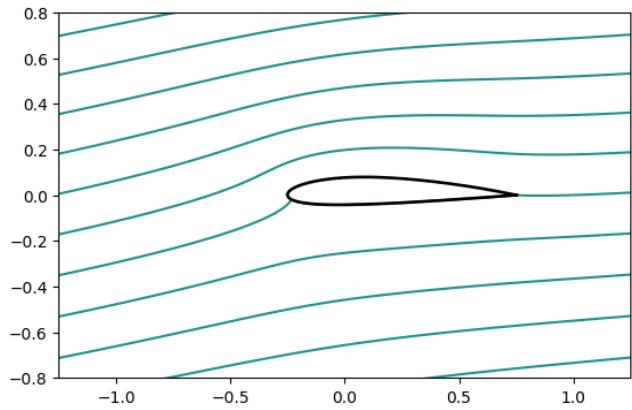
(b) $\alpha = 2^\circ$



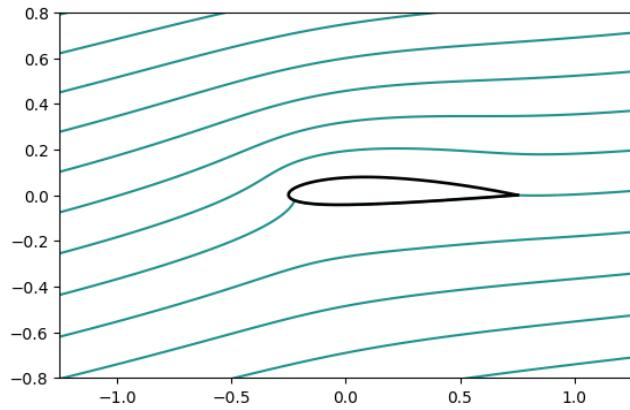
(c) $\alpha = 4^\circ$



(d) $\alpha = 6^\circ$



(e) $\alpha = 8^\circ$



(f) $\alpha = 10^\circ$

Figura 6.39: Líneas de corriente alrededor de perfil NACA 2412

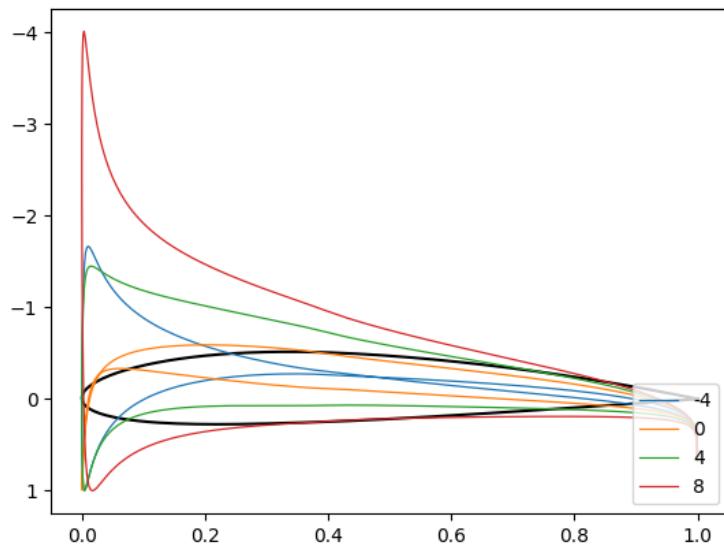


Figura 6.40: Coeficiente de presión a lo largo de la cuerda de perfil NACA 2412 para diferentes ángulos de ataque

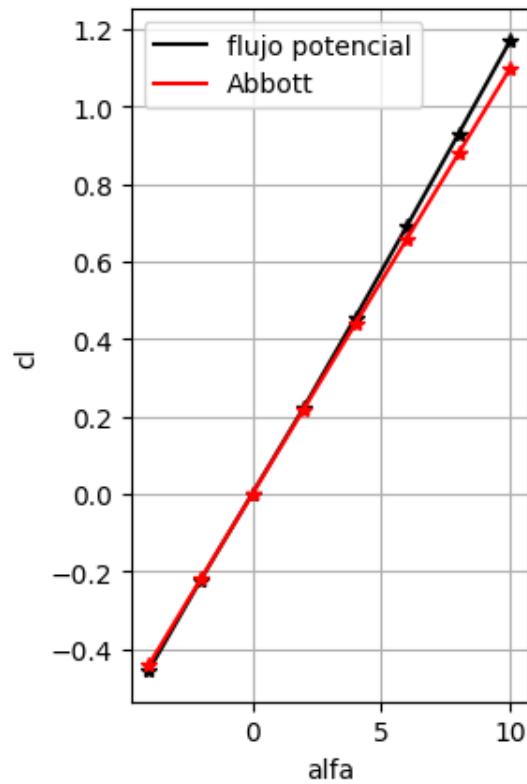


Figura 6.41: Gráfica de coeficiente de sustentación vs. ángulo de ataque para perfil NACA 2412

6.3.2. SU2

Para esta sección se ha decidido utilizar como base los tutoriales creados por el equipo de desarrollo del software SU2, los cuales están disponibles al público desde su página web [11].

SU2 es un programa cuyo flujo de trabajo se basa en un archivo de configuración con extensión “.cfg”, es en este archivo donde se especifican todas las propiedades que caracterizan al problema que se pretende simular. Uno puede por ejemplo, especificar las condiciones del flujo en corriente libre, el ángulo de ataque, el número de Reynolds, el “solver” que se va a utilizar, las ecuaciones que se van a resolver, condiciones de frontera, criterios de convergencia, entre muchos otros parámetros más. De entre todos estos parámetros, el de mayor importancia para este trabajo y los resultados presentados a continuación, es el que nos permite especificar un archivo que contenga la descripción de la malla a utilizarse. SU2 es capaz de leer diferentes tipos de archivo para la importación de la malla, el principal es su formato nativo el cual se asocia a un archivo con extensión “.su2”, es en este formato nativo en el cual el software desarrollado en este proyecto exporta las mallas, además del ya mencionado formato, SU2 también es capaz de interpretar archivos que sigan el formato CGNS (CFD General Notation System), el cual es un esfuerzo de la comunidad para estandarizar la representación de la información para la solución de simulaciones para DFC.

En este trabajo, cuyo desarrollo gira alrededor de la generación de mallas, se ha optado por utilizar las mismas condiciones que se utilizan en los tutoriales y únicamente proporcionar la malla correspondiente al perfil aerodinámico de interés, con el fin de comprobar la calidad de las mallas que se puedan generar con este proyecto, y su compatibilidad con software reconocido de dinámica de fluidos computacional.

El caso de análisis seleccionado implica flujo bidimensional, turbulento y estacionario, los parámetros del flujo se muestran en la tabla 6.6

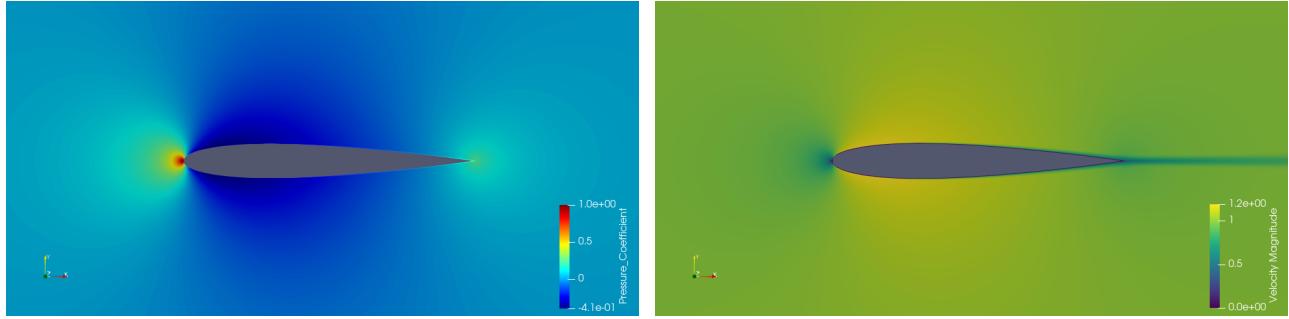
Parámetro	Valor
$\rho [kg/m^3]$	2.13163
$V_\infty [m/s]$	$\alpha = 0^\circ .(52.157, 0.0, 0.0)$ $\alpha = 10^\circ .(51.365, 9.057, 0.0)$
$\mu [kg/(m \cdot s)]$	1.853×10^{-5}
Re	6×10^6

Tabla 6.6: Características del flujo para el análisis en SU2

Se realizaron diferentes casos de estudio, un caso con un único perfil alar y otro caso con perfil alar y flap tipo perfil externo, para cada tipo de malla (malla tipo O y C), dando un total de cuatro análisis. Todos los casos simulan el mismo flujo, descrito previamente. En los cuatro casos mencionados, el perfil alar a analizar es el perfil NACA 0012, con cuerda unitaria (1 m). En los casos donde el perfil cuenta con un flap, se seleccionó la configuración de flap de perfil externo, el perfil correspondiente al flap es también un NACA 0012, en este caso su cuerda es 0.2 de la cuerda del perfil

A continuación se presentan los resultados obtenidos en estos cuatro casos. Para cada caso se muestra las gráficas tanto del campo de presión como del campo de velocidad. En los casos donde se analiza únicamente al perfil NACA 0012 se realizaron simulaciones para dos casos diferentes, en el primero de ellos el ángulo de ataque es $\alpha = 0^\circ$, en el segundo caso se cambia su valor a $\alpha = 10^\circ$.

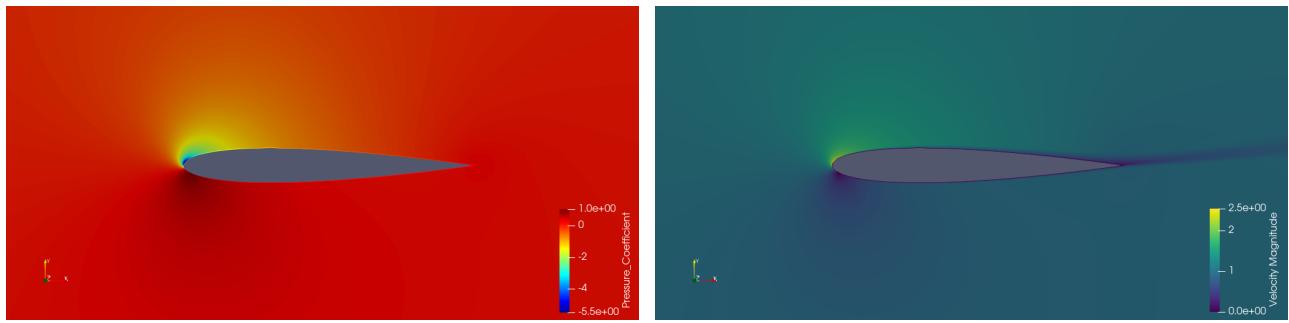
Los casos en los que se analiza un perfil con flap, se simularon con el perfil a un ángulo de ataque $\alpha = 0^\circ$ y el flap con una deflexión $\delta = 15^\circ$.



(a) Campo de presión

(b) Campo de velocidad

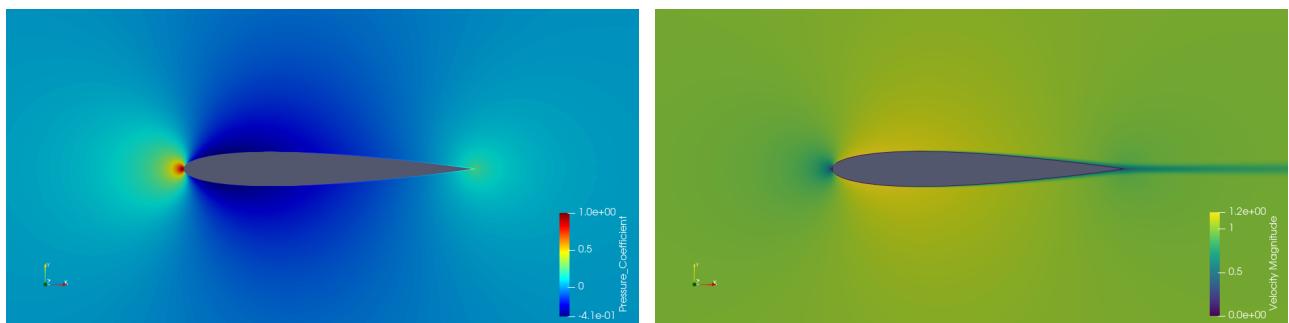
Figura 6.42: Resultados de la simulación en SU2 con una malla tipo C de un perfil NACA 0012 con ángulo de ataque $\alpha = 0^\circ$



(a) Campo de presión

(b) Campo de velocidad

Figura 6.43: Resultados de la simulación en SU2 con una malla tipo C de un perfil NACA 0012 con ángulo de ataque $\alpha = 10^\circ$

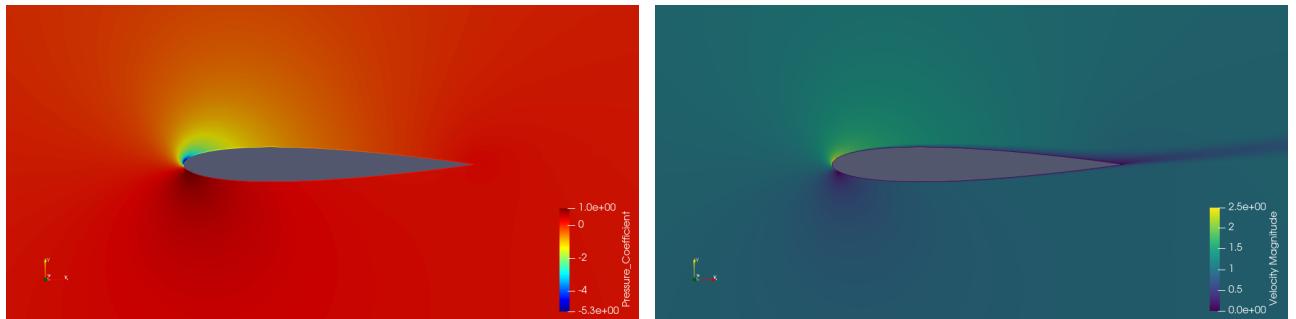


(a) Campo de presión

(b) Campo de velocidad

Figura 6.44: Resultados de la simulación en SU2 con una malla tipo O de un perfil NACA 0012 con ángulo de ataque $\alpha = 0^\circ$

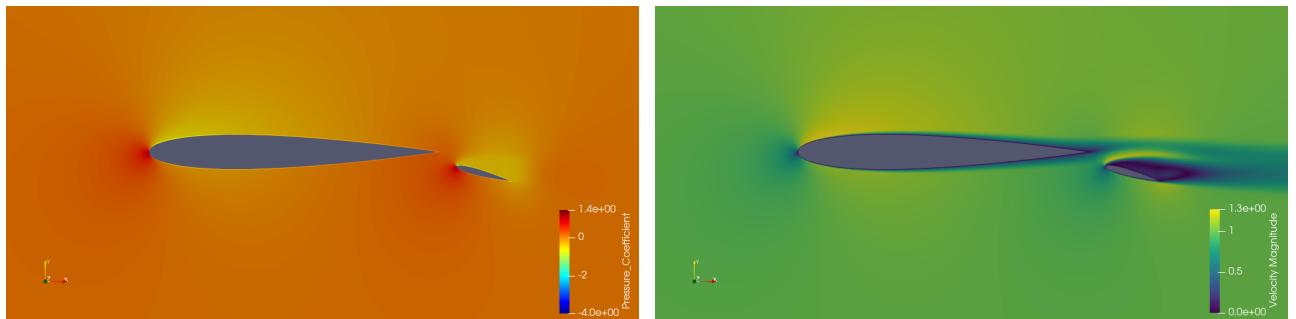
Por último se muestra en la figura 6.48 una comparación del valor del coeficiente de presión para el perfil NACA 0012 a un ángulo de ataque $\alpha = 10^\circ$ para tres casos diferentes, dos de ellos son



(a) Campo de presión

(b) Campo de velocidad

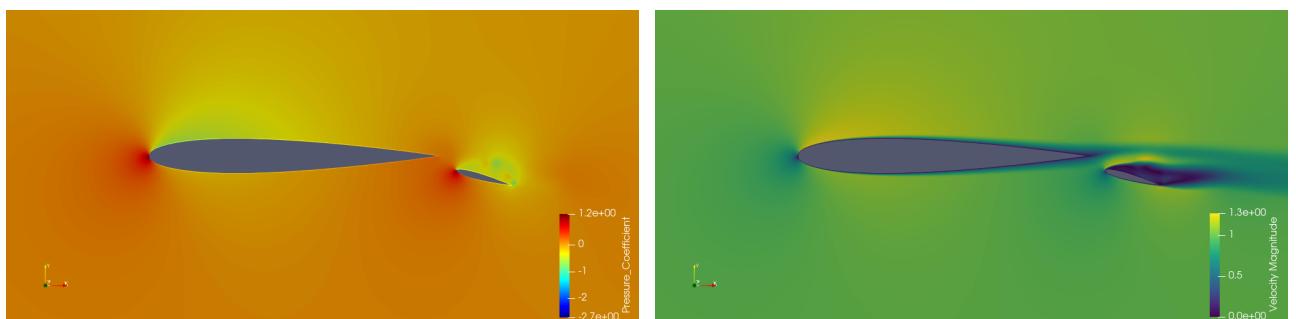
Figura 6.45: Resultados de la simulación en SU2 con una malla tipo O de un perfil NACA 0012 con ángulo de ataque $\alpha = 10^\circ$



(a) Campo de presión

(b) Campo de velocidad

Figura 6.46: Resultados de la simulación en SU2 con una malla tipo C de un perfil NACA 0012 con ángulo de ataque $\alpha = 0^\circ$ y un flap con deflexión $\delta = 15^\circ$



(a) Campo de presión

(b) Campo de velocidad

Figura 6.47: Resultados de la simulación en SU2 con una malla tipo O de un perfil NACA 0012 con ángulo de ataque $\alpha = 0^\circ$ y un flap con deflexión $\delta = 15^\circ$

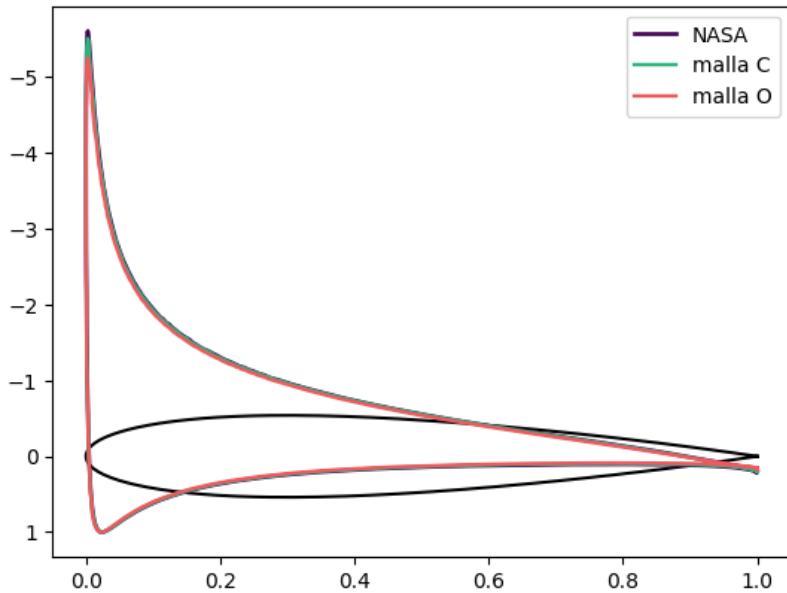


Figura 6.48: Comparación del coeficiente de presión a lo largo de un perfil NACA 0012 para un ángulo de ataque $\alpha = 10^\circ$

los resultados presentados en este trabajo, tanto malla tipo C como tipo O, además de éstos se presentan los resultados de la simulación del mismo problema utilizando una malla estructurada tipo C creada en el Centro de Investigación Langley de la NASA [27], dicha malla es proporcionada como parte del tutorial del programa SU2.

Capítulo 7

Conclusiones

En lo general, se cumplen con los objetivos planteados al inicio del desarrollo del proyecto, es decir, se ha desarrollado un paquete de códigos en lenguaje Python 3 que generan mallas con la calidad requerida para su uso en análisis de dinámica de fluidos computacional. En lo particular, los objetivos planteados también fueron alcanzados, se proporciona entonces, un módulo que permite generar cualquier perfil NACA de la serie de 4 dígitos con cualquier densidad de puntos deseada, además de poder importar una nube de puntos que describa el perfil deseado, a través de un archivo de valores separados por comas. Además de esto se desarrollaron los módulos necesarios para la generación de mallas, tanto tipo O como tipo C, mediante diversos algoritmos, ya sea mediante la solución de ecuaciones algebraicas o ecuaciones diferenciales parciales. Por último se llevaron a cabo simulaciones tanto de flujo potencial, como de flujo turbulento y estacionario.

Derivado de este proyecto se cuenta con toda una suite robusta de software para la generación de mallas estructuradas. Se proporcionan como opciones dos diferentes tipologías, mallas tipo C y tipo O, además de la tipología también se tiene la capacidad de analizar diferentes configuraciones de frontera interna, el estado particular del programa permite el análisis de perfiles aerodinámicos con un flap en configuración de perfil externo.

Es importante aclarar que el desarrollo del programa se ha hecho considerando cualquier tipo de configuración, por lo que la generación de mallas puede ser extensible a cualquier problema ya sea una configuración diferente de flaps como podría ser un flap simple, ranurado (sencillo, doble o triple), tipo Fowler, así como slats. El usuario puede simplemente importar una nube de puntos que describa la configuración geométrica de su interés. Por otro lado, vale la pena recalcar que queda abierta la posibilidad de agregar nuevos modulos que proporcionen nuevas configuraciones para la frontera interna de manera relativamente sencilla ya que desde el inicio del proyecto se contempló esta posibilidad, por lo que la funcionalidad ya existe en el software, solo habrá que adaptar las particularidades de cada configuración para cada módulo.

Las mallas generadas presentan en general buena calidad en la mayoría de sus celdas, siendo normalmente la excepción en las vecindades de la frontera externa, resultado de las funciones de forzado utilizadas para obtener mayor densidad de nodos en la frontera interna. La localización en el dominio físico de estas celdas no tiene un efecto negativo en los resultados dado que es una zona donde los gradientes suelen ser casi nulos. Cabe mencionar que estos resultados no son constantes para todas las mallas, y que obtener una malla de calidad representa un proceso iterativo y exhaustivo dado que cada problema requiere diferentes parámetros.

Se realizaron análisis de flujo potencial y de flujo turbulento y estacionario para poner a prueba la calidad y usabilidad de las mallas resultantes. El programa ha probado generar mallas con la calidad necesaria para que sean de utilidad práctica para análisis de flujos bajo diversas condiciones.

A parte de la generación de mallas, se ha desarrollado un código capaz de resolver la ecuación de flujo potencial para mallas tipo O con una única frontera interna. Una vez obtenida la solución del potencial de velocidad se puede calcular los coeficientes de presión y de sustentación, así como la función del corriente.

Los resultados obtenidos a partir del análisis de flujo potencial muestran congruencia con los resultados presentados por Abbott [26] para un rango de ángulos de ataque a los cuales el flujo incidente sobre el perfil puede considerarse no viscoso e irrotacional. Se observa como la comparación diverge conforme aumenta el ángulo de ataque y éste se acerca a valores donde comienza a haber separación del flujo sobre la superficie del perfil, dicho efecto no puede ser representado bajo el modelo de flujo potencial.

Las simulaciones realizadas mediante SU2 fueron contrastadas con los resultados que el mismo software proporciona para el problema analizado, donde se observa una similitud en los valores obtenidos para el coeficiente de presión a lo largo de la cuerda aerodinámica.

Capítulo 8

Recomendaciones y trabajos futuros

En general se debe buscar herramientas que su integración resulte en mejoras sustanciales en el performance de la aplicación, entre ellas se pueden destacar

- Cython. Extensión del lenguaje Python 3 que permite la ejecución de código escrito en C o C++.
- Numba. Módulo de python enfocado en la computación científica de alto performance. Convierte el código de Python a código optimizado en lenguaje máquina.
- C / C++.
- Técnicas de paralelización. Multihilos, vectorización, aceleración mediante GPU (procesadores gráficos), etc.

También debe prestarse especial atención a las técnicas de programación aplicadas sin importar las herramientas sobre las cuales se esté desarrollando. La DFC es una herramienta cuyas características implican una gran demanda de recursos computacionales, lo que la mantiene siempre a la vanguardia tecnológica, sin embargo el estado actual del arte es tal que aún representa procesos largos e iterativos, por lo que una mala implementación puede resultar en crecimientos exponenciales de los tiempos de ejecución. Si se pretende desarrollar programas que cumplan las exigencias y expectativas requeridas en la DFC, este aspecto debe ser de primordial importancia.

Se recomienda revisar los múltiples trabajos del Ph.D. Robert A. van de Geijn, profesor del departamento de Ciencias de la Computación en la Universidad de Texas. Dichos trabajos y publicaciones están enfocados al desarrollo de algoritmos y técnicas para la computación científica de alto performance.

Volviendo a la generación de mallas, se recomienda revisar a fondo las técnicas de generación mediante EDPs hiperbólicas y parabólicas, poniendo particular interés en la estabilidad numérica de los métodos. También se debe extender la generación de mallas a otras técnicas como son mallas adaptativas, no estructuradas, e incluso mediante algoritmos de aprendizaje automatizado.

Se propone la extensión hacia otras herramientas de análisis de CFD, es decir, crear métodos que exporten las mallas generadas a formatos reconocibles por otros paquetes, por ejemplo OpenFOAM.

Apéndice A

Códigos

A continuación, se muestran únicamente los códigos de mayor relevancia para este trabajo, debido a la extensión del mismo, sin embargo, el proyecto completo se puede encontrar y descargar en el siguiente repositorio de github <https://github.com/Cardoso1994/Grid-Generator—CFD>.

Los códigos que se muestran a continuación son los siguientes

- Malla C. Generación de malla mediante la ecuación de Poisson, con el uso del módulo numba.
- Malla O. Generación de malla mediante la ecuación de Poisson, con el uso del módulo numba.
- Conversión de mallas a formato SU2
- Archivo main. Ejemplo para la ejecución del programa.

A.1. Generación de Malla C con uso de librería numba

```
"""
@author:      Marco Antonio Cardoso Moreno
@mail:        marcoacardosom@gmail.com
```

Extiende subclase mesh_C.

Metodos de generacion para mallas tipo C mediante la ecuacion de Poisson, apoyandose de la libreria numba y de metodos de vectorizado

```
import numpy as np
import matplotlib.pyplot as plt
from numba import jit

from mesh import mesh
import mesh_su2
```

```
def gen_Poisson_n(self, metodo='SOR', omega=1, a=0, c=0, linea_xi=0,  
                  aa=0, cc=0, linea_eta=0):
```

```
    """
```

Resuelve la ecuacion de Poisson para generar la malla.

Metodo que se apoya en el uso de la libreria Numba.

El codigo es el mismo que en el metodo clasico.

```
...
```

Parametros

metodo : str

*Metodo iterativo de solucion. Jacobi (J), Gauss Seidel (GS) y
sobrerelajacion (SOR)*

omega : float64

*Valor utilizado para acelerar o suavizar la solucion. Solo se
utiliza si metodo == 'SOR'*

omega < 1 —> suaviza la solucion

omega = 1 —> metodod Gauss Seidel

omega > 1 —> acelera la solucion

a, c : float64

valores ocupados para la funcion de forzado P, en el eje xi

linea_xi : int

linea en el eje xi hacia la cual se realiza el forzado.

0 <= linea_xi <= self.M

aa, cc : float64

valores ocupados para la funcion de forzado Q, en el eje eta

linea_eta : int

linea en el eje eta hacia la cual se realiza el forzado.

0 <= linea_eta <= self.N

Return

None

```
"""
```

```
# aproximacion inicial  
self.gen_TFI()
```

Xn = self.X

Yn = self.Y

Xo = Xn.copy()

Yo = Yn.copy()

m = self.M

n = self.N

```

d_eta    = self.d_eta
d_xi     = self.d_xi

# funciones de forzado
P_ = np.arange(1, m)
Q_ = np.arange(1, n)
P_ = -a * (P_ / (m-1) - linea_xi) \
      / np.abs(P_ / (m-1) - linea_xi) \
      * np.exp(-c * np.abs(P_ / \
                           (m-1) - linea_xi))

Q_ = -aa * (Q_ / (n-1) - linea_eta) \
      / np.abs(Q_ / (n-1) - linea_eta) \
      * np.exp(-cc \
      * np.abs(Q_ / (n-1) - linea_eta))

mask = np.isnan(P_)
P_[mask] = 0
mask = np.isnan(Q_)
Q_[mask] = 0

# obteniendo el indice de la union de los perfiles
if not self.airfoil_alone:
    union_start = 0
    while self.Y[union_start, 0] == 0:
        union_start += 1
    i = 0
    while self.airfoil_boundary[i] != 0:
        union_start += 1
        i += 1
    union_start -= 1

it = 0
mesh.it_max = 750000
mesh.err_max = 1e-6

# inicio del metodo iterativo, separar el metodo para perfil con
# y sin flap
print(f"Generando malla tipo C. \nDimensiones M: {self.M}")
      + f"N: {self.N}")
if self.airfoil_alone:
    print("Perfil")
    print("Poisson numba:")

# while it < mesh.it_max:
for it in range(mesh.it_max):
    if it % 150000 == 0:
        self.X = np.copy(Xn)

```

```

        self.Y = np.copy(Yn)
        self.plot()
        print()

# imprime informacion
print('it = ' + str(it) + ' aa = ' + str(aa) + ' cc = '
      + str(cc)
      + ' err_x = ' + '{:.3e}'.format(abs(Xn - Xo).max())
      + ' err_y = ' + '{:.3e}'.format(abs(Yn - Yo).max())
      + '\t\t', end="\r")

Xo = Xn.copy()
Yo = Yn.copy()
# si el metodo iterativo es Jacobi
if metodo == 'J':
    X = Xo
    Y = Yo
else: # si el metodo es Gauss-Seidel o SOR
    X = Xn
    Y = Yn

(Xn, Yn) = _gen_Poisson_n(X, Y, self.M, self.N, P_, Q_)

# se aplica sobre-relajacion si el metodo es SOR
if metodo == 'SOR':
    Xn = omega * Xn + (1 - omega) * Xo
    Yn = omega * Yn + (1 - omega) * Yo

    if abs(Xn - Xo).max() < mesh.err_max \
        and abs(Yn - Yo).max() < mesh.err_max:
        print('Poisson: ' + metodo + ': saliendo... ')
        print('it=' , it)
        break

else:
    print("Perfil con flap")
    print("Poisson numba:")

# while it < mesh.it_max:
for it in range(mesh.it_max):
    if (it % 150e3 == 0):
        self.X = np.copy(Xn)
        self.Y = np.copy(Yn)
        self.plot()
        print()

# printing info
print('it = ' + str(it) + ' aa = ' + str(aa) + ' cc = '

```

```

+ str(cc)
+ ' err_x = ' + '{:.3e}'.format(abs(Xn - Xo).max())
+ ' err_y = ' + '{:.3e}'.format(abs(Yn - Yo).max())
+ '\t\t', end="\r")

Xo = Xn.copy()
Yo = Yn.copy()
# si el metodo iterativo es Jacobi
if metodo == 'J':
    X = Xo
    Y = Yo
else: # si el metodo es Gauss-Seidel o SOR
    X = Xn
    Y = Yn

(Xn, Yn) = _gen_Poisson_n_flap(X, Y, self.M, self.N, P_,
                                 Q_, self.airfoil_boundary,
                                 union_start)

# se aplica sobre-relajacion si el metodo es SOR
if metodo == 'SOR':
    Xn = omega * Xn + (1 - omega) * Xo
    Yn = omega * Yn + (1 - omega) * Yo

    if abs(Xn - Xo).max() < mesh.err_max \
        and abs(Yn - Yo).max() < mesh.err_max:
        print('\nPoisson: ' + metodo + ': saliendo... ')
        print('it:', it)
        break

self.X = Xn
self.Y = Yn
return (self.X, self.Y)

@jit
def _gen_Poisson_n_flap(X, Y, M, N, P_, Q_, airfoil_boundary,
                        union_start):
"""
Resuelve los for loops anidados correspondientes a la solucion de la
ecuacion de Poisson para generar la malla.

Metodo que se apoya en el uso de la libreria Numba.
El codigo es el mismo que en el metodo clasico.
...

```

Parametros

X : numpy.array
Matriz que contiene las coordenadas X que describen a la malla

Y : numpy.array
Matriz que contiene las coordenadas Y que describen a la malla

M : int
Numero de divisiones en el eje xi.

N : int
Numero de divisiones en el eje eta.

P_ : numpy.array
Valores de la funcion de forzado P para el eje xi

Q_ : numpy.array
Valores de la funcion de forzado Q para el eje eta

airfoil_boundary : numpy.array
Cada elemento del array indica el punto a que perfil pertenece (numero positivo) o cero si no forma parte de una frontera

union_start : int
indice "i" en el que inicia la seccion de union entre los perfiles

Return

(X, Y) : numpy.array, numpy.array
Matrices X y Y que describen la malla. Actualizadas.
"""

```
d_eta = 1
d_xi = 1
m = M
n = N

begin_perfil = 118
end_perfil = begin_perfil + 1083
limit = 0
for j in range(n-2, 0, -1):
    for i in range(1, m-1):
        x_eta = (X[i, j+1] - X[i, j-1]) / 2 / d_eta
        y_eta = (Y[i, j+1] - Y[i, j-1]) / 2 / d_eta
        x_xi = (X[i+1, j] - X[i-1, j]) / 2 / d_xi
        y_xi = (Y[i+1, j] - Y[i-1, j]) / 2 / d_xi

        alpha = x_eta ** 2 + y_eta ** 2
        beta = x_xi * x_eta + y_xi * y_eta
        gamma = x_xi ** 2 + y_xi ** 2
        I = x_xi * y_eta - x_eta * y_xi
```

```

X[i , j] = (d_xi * d_eta) ** 2 \
/ (2 * (alpha * d_eta ** 2 + gamma * d_xi ** 2)) \
* (alpha / (d_xi ** 2) * (X[i+1, j] + X[i-1, j]) \
+ gamma / (d_eta ** 2) * (X[i , j+1] + X[i , j-1]) \
- beta / (2 * d_xi * d_eta) * (X[i+1, j+1] \
- X[i+1, j-1] + X[i-1, j-1] - X[i-1, j+1]) \
+ I ** 2 * (P_[i-1] * x_xi + Q_[j-1] * x_eta))
Y[i , j] = (d_xi * d_eta) ** 2 \
/ (2 * (alpha * d_eta ** 2 + gamma * d_xi ** 2)) \
* (alpha / (d_xi ** 2) * (Y[i+1, j] + Y[i-1, j]) \
+ gamma / (d_eta ** 2) * (Y[i , j+1] + Y[i , j-1]) \
- beta / (2 * d_xi * d_eta) * (Y[i+1, j+1] \
- Y[i+1, j-1] + Y[i-1, j-1] - Y[i-1, j+1]) \
+ I ** 2 * (P_[i-1] * y_xi + Q_[j-1] * y_eta))

# se calculan los puntos en la seccion de salida de la malla
i = 0
x_eta = (X[i , j+1] - X[i , j-1]) / 2 / d_eta
y_eta = (Y[i , j+1] - Y[i , j-1]) / 2 / d_eta
x_xi = (X[i+1, j] - X[i , j]) / d_xi
y_xi = (Y[i+1, j] - Y[i , j]) / d_xi

alpha = x_eta ** 2 + y_eta ** 2
beta = x_xi * x_eta + y_xi * y_eta
gamma = x_xi ** 2 + y_xi ** 2
I = x_xi * y_eta - x_eta * y_xi

Y[i , j] = (d_xi * d_eta) ** 2 \
/ (2 * gamma * d_xi ** 2 - alpha * d_eta ** 2) \
* (alpha / d_xi ** 2 * (Y[i+2, j] - 2 * Y[i+1, j]) \
- beta / d_xi / d_eta \
* (Y[i+1, j+1] - Y[i+1, j-1] - Y[i , j+1] + Y[i , j-1]) \
+ gamma / d_eta ** 2 * (Y[i , j+1] + Y[i , j-1]) \
+ I ** 2 * (P_[i-1] * y_xi + Q_[j-1] * y_eta))

i = m-1
x_eta = (X[i , j+1] - X[i , j-1]) / 2 / d_eta
y_eta = (Y[i , j+1] - Y[i , j-1]) / 2 / d_eta
x_xi = (X[i , j] - X[i-1, j]) / d_xi
y_xi = (Y[i , j] - Y[i-1, j]) / d_xi

alpha = x_eta ** 2 + y_eta ** 2
beta = x_xi * x_eta + y_xi * y_eta
gamma = x_xi ** 2 + y_xi ** 2
I = x_xi * y_eta - x_eta * y_xi

Y[i , j] = (d_xi * d_eta) ** 2 \

```

```

/ (2 * gamma * d_xi ** 2 - alpha * d_eta**2) \
* (alpha / d_xi**2 * (-2 * Y[i-1, j] + Y[i-2, j])
   - beta / d_xi / d_eta
   * (Y[i, j+1] - Y[i, j-1] - Y[i-1, j+1] + Y[i-1, j-1])
   + gamma / d_eta**2 * (Y[i, j+1] + Y[i, j-1])
   + I**2 * (P_[i-1] * y_xi + Q_[j-1] * y_eta))

```

seccion de union entre perfiles

```

i_ = 0
while airfoil_boundary[i_] != 0:
    i_ += 1
i = union_start

while airfoil_boundary[i_] == 0:
    x_eta = (X[i, 1] - X[-i-1, 1]) / 2 / d_eta
    y_eta = (Y[i, 1] - Y[-i-1, 1]) / 2 / d_eta
    x_xi = (X[i+1, 0] - X[i-1, 0]) / 2 / d_xi
    y_xi = (Y[i+1, 0] - Y[i-1, 0]) / 2 / d_xi

    alpha = x_eta ** 2 + y_eta ** 2
    beta = x_xi * x_eta + y_xi * y_eta
    gamma = x_xi ** 2 + y_xi ** 2
    I = x_xi * y_eta - x_eta * y_xi

    X[i, 0] = (d_xi * d_eta) ** 2 \
        / (2 * (alpha * d_eta ** 2 + gamma * d_xi ** 2)) \
        * (alpha / (d_xi ** 2) * (X[i+1, 0] + X[i-1, 0])
           + gamma / (d_eta ** 2) * (X[i, 1] + X[-i-1, 1])
           - beta / (2 * d_xi * d_eta) * (X[i+1, 1]
               - X[-i-2, 1] + X[-i, 1] - X[i-1, 1]))
    Y[i, 0] = (d_xi * d_eta) ** 2 \
        / (2 * (alpha * d_eta ** 2 + gamma * d_xi ** 2)) \
        * (alpha / (d_xi ** 2) * (Y[i+1, 0] + Y[i-1, 0])
           + gamma / (d_eta ** 2) * (Y[i, 1] + Y[-i-1, 1])
           - beta / (2 * d_xi * d_eta) * (Y[i+1, 1]
               - Y[-i-2, 1] + Y[-i, 1] - Y[i-1, 1]))

    X[-i-1, 0] = X[i, 0]
    Y[-i-1, 0] = Y[i, 0]
    i += 1
    i_ += 1

return (X, Y)

```

```

@jit
def _gen_Poisson_n(X, Y, M, N, P_, Q_):

```

```
"""
```

Resuelve los for loops anidados correspondientes a la solucion de la ecuacion de Poisson para generar la malla.

Metodo que se apoya en el uso de la libreria Numba.

El codigo es el mismo que en el metodo clasico.

```
...
```

Parametros

X : numpy.array

Matriz que contiene las coordenadas X que describen a la malla

Y : numpy.array

Matriz que contiene las coordenadas Y que describen a la malla

M : int

Numero de divisiones en el eje xi.

N : int

Numero de divisiones en el eje eta.

P_ : numpy.array

Valores de la funcion de forzado P para el eje xi

Q_ : numpy.array

Valores de la funcion de forzado Q para el eje eta

Return

(X, Y) : numpy.array, numpy.array

Matrices X y Y que describen la malla. Actualizadas.

```
"""
```

d_eta = 1

d_xi = 1

m = M

n = N

begin_perfil = 118

end_perfil = begin_perfil + 1083

limit = 0

for j in range(n-2, 0, -1):

for i in range(1, m-1):

x_eta = (X[i, j+1] - X[i, j-1]) / 2 / d_eta

y_eta = (Y[i, j+1] - Y[i, j-1]) / 2 / d_eta

x_xi = (X[i+1, j] - X[i-1, j]) / 2 / d_xi

y_xi = (Y[i+1, j] - Y[i-1, j]) / 2 / d_xi

alpha = x_eta ** 2 + y_eta ** 2

beta = x_xi * x_eta + y_xi * y_eta

gamma = x_xi ** 2 + y_xi ** 2

```

I = x_xi * y_eta - x_eta * y_xi

X[i , j] = (d_xi * d_eta) ** 2 \
/ (2 * (alpha * d_eta ** 2 + gamma * d_xi ** 2)) \
* (alpha / (d_xi ** 2) * (X[i+1, j] + X[i-1, j]) \
+ gamma / (d_eta ** 2) * (X[i, j+1] + X[i, j-1]) \
- beta / (2 * d_xi * d_eta) * (X[i+1, j+1] \
- X[i+1, j-1] + X[i-1, j-1] - X[i-1, j+1]) \
+ I ** 2 * (P_[i-1] * x_xi + Q_[j-1] * x_eta))

Y[i , j] = (d_xi * d_eta) ** 2 \
/ (2 * (alpha * d_eta ** 2 + gamma * d_xi ** 2)) \
* (alpha / (d_xi ** 2) * (Y[i+1, j] + Y[i-1, j]) \
+ gamma / (d_eta ** 2) * (Y[i, j+1] + Y[i, j-1]) \
- beta / (2 * d_xi * d_eta) * (Y[i+1, j+1] \
- Y[i+1, j-1] + Y[i-1, j-1] - Y[i-1, j+1]) \
+ I ** 2 * (P_[i-1] * y_xi + Q_[j-1] * y_eta))

# se calculan los puntos en la seccion de salida de la malla
i = 0
x_eta = (X[i , j+1] - X[i , j-1]) / 2 / d_eta
y_eta = (Y[i , j+1] - Y[i , j-1]) / 2 / d_eta
x_xi = (X[i+1, j] - X[i, j]) / d_xi
y_xi = (Y[i+1, j] - Y[i, j]) / d_xi

alpha = x_eta ** 2 + y_eta ** 2
beta = x_xi * x_eta + y_xi * y_eta
gamma = x_xi ** 2 + y_xi ** 2
I = x_xi * y_eta - x_eta * y_xi

Y[i , j] = (d_xi * d_eta) ** 2 \
/ (2 * gamma * d_xi ** 2 - alpha * d_eta ** 2) \
* (alpha / d_xi ** 2 * (Y[i+2, j] - 2 * Y[i+1, j]) \
- beta / d_xi / d_eta \
* (Y[i+1, j+1] - Y[i+1, j-1] - Y[i, j+1] + Y[i, j-1]) \
+ gamma / d_eta ** 2 * (Y[i, j+1] + Y[i, j-1]) \
+ I ** 2 * (P_[i-1] * y_xi + Q_[j-1] * y_eta))

i = m-1
x_eta = (X[i , j+1] - X[i , j-1]) / 2 / d_eta
y_eta = (Y[i , j+1] - Y[i , j-1]) / 2 / d_eta
x_xi = (X[i , j] - X[i-1, j]) / d_xi
y_xi = (Y[i , j] - Y[i-1, j]) / d_xi

alpha = x_eta ** 2 + y_eta ** 2
beta = x_xi * x_eta + y_xi * y_eta
gamma = x_xi ** 2 + y_xi ** 2
I = x_xi * y_eta - x_eta * y_xi

```

```

Y[i , j] = (d_xi * d_eta) ** 2 \
/ (2 * gamma * d_xi ** 2 - alpha * d_eta**2) \
* (alpha / d_xi**2 * (-2 * Y[i-1, j] + Y[i-2, j]) \
- beta / d_xi / d_eta \
* (Y[i , j+1] - Y[i , j-1] - Y[i-1, j+1] + Y[i-1, j-1]) \
+ gamma / d_eta**2 * (Y[i , j+1] + Y[i , j-1]) \
+ I**2 * (P_[i-1] * y_xi + Q_[j-1] * y_eta))

```

return (X, Y)

A.2. Generación de Malla O con uso de librería numba

```

"""
@author: Marco Antonio Cardoso Moreno
@mail: marcoarcardosom@gmail.com

```

Extiende subclase mesh_O.

Diversos metodos de generacion para mallas tipo O, apoyandose de la libreria numba y de metodos de vectorizado

```

import numpy as np
import matplotlib.pyplot as plt
from numba import jit

from mesh import mesh
import mesh_su2

```

```

def gen_Poisson_n(self , metodo='SOR' , omega=1, a=0, c=0, linea_xi=0,
                  aa=0, cc=0, linea_eta=0):
    """

```

Resuelve la ecuacion de Poisson para generar la malla.

*Metodo que se apoya en el uso de la libreria Numba.
El codigo es el mismo que en el metodo clasico.*

...

Parametros

metodo : str

Metodo iterativo de solucion. Jacobi (J), Gauss Seidel (GS) y sobrerelajacion (SOR)

omega : float64

Valor utilizado para acelerar o suavizar la solucion. Solo se utiliza si metodo == 'SOR'

```

    omega < 1 —> suaviza la solucion
    omega = 1 —> metodo Gauss Seidel
    omega > 1 —> acelera la solucion
a, c : float64
    valores ocupados para la funcion de forzado P, en el eje xi
linea_xi : int
    linea en el eje xi hacia la cual se realiza el forzado.
0 <= linea_xi <= self.M
aa, cc : float64
    valores ocupados para la funcion de forzado P, en el eje eta
linea_eta : int
    linea en el eje eta hacia la cual se realiza el forzado.
0 <= linea_eta <= self.N

```

Return

None
"""

```

# aproximacion inicial
self.gen_TFI()

# asignacion de variable para metodo
Xn = self.X
Yn = self.Y
Xo = Xn.copy()
Yo = Yn.copy()

m      = self.M
n      = self.N

d_eta  = self.d_eta
d_xi   = self.d_xi

P_ = np.arange(1, m)
Q_ = np.arange(1, n)
P_ = -a * (P_ / (m-1) - linea_xi) \
      / np.abs(P_ / (m-1) - linea_xi) \
      * np.exp(-c * np.abs(P_ / \
                           (m-1) - linea_xi))

Q_ = -aa * (Q_ / (n-1) - linea_eta) \
      / np.abs(Q_ / (n-1) - linea_eta) \
      * np.exp(-cc \
                * np.abs(Q_ / (n-1) - linea_eta))

mask = np.isnan(P_)
P_[mask] = 0

```

```

mask = np.isnan(Q_)
Q_[mask] = 0

# obteniendo el indice de la union de los perfiles
if not self.airfoil_alone:
    union_start = 0
    while self.airfoil_boundary[union_start] != 0:
        union_start += 1

mesh.it_max = 750000
mesh.err_max = 1e-6
# inicio del metodo iterativo , separa el metodo para perfil con
# y sin flap
print(f"Generando malla tipo O.\nDimensiones M: {self.M}"
      + f" N: {self.N}")
if self.airfoil_alone:
    print("Perfil")
    print("Poisson numba:")
    for it in range(mesh.it_max):
        if (it % 150e3 == 0):
            self.X = np.copy(Xn)
            self.Y = np.copy(Yn)
            self.plot()
            print()

# imprime informacion
print('it = ' + str(it) + ' aa = ' + str(aa) + ' cc = '
      + str(cc)
      + ' err_x = ' + '{:.3e}'.format(abs(Xn - Xo).max())
      + ' err_y = ' + '{:.3e}'.format(abs(Yn - Yo).max())
      + '\t\t', end="\r")

Xo = Xn.copy()
Yo = Yn.copy()
# si el metodo iterativo es Jacobi
if metodo == 'J':
    X = Xo
    Y = Yo
else: # si el metodo es Gauss-Seidel o SOR
    X = Xn
    Y = Yn

(Xn, Yn) = _gen_Poisson_n(X, Y, self.M, self.N, P_, Q_)

# se aplica sobre-relajacion si el metodo es SOR
if metodo == 'SOR':
    Xn = omega * Xn + (1 - omega) * Xo

```

```

Yn = omega * Yn + (1 - omega) * Yo

if abs(Xn - Xo).max() < mesh.err_max \
    and abs(Yn - Yo).max() < mesh.err_max and it > 10:
    print('Poisson: ' + metodo + ': saliendo... ')
    print('it= ', it)
    break

else:
    print("Perfil con flap")
    print("Poisson numba:")
    for it in range(mesh.it_max):
        if (it % 650e3 == 0):
            self.X = np.copy(Xn)
            self.Y = np.copy(Yn)
            self.plot()
            print()

# imprime informacion
print('it = ' + str(it) + ' aa = ' + str(aa) + ' cc = ' +
      + str(cc) +
      + ' err_x = ' + '{:.3e}'.format(abs(Xn - Xo).max()) +
      + ' err_y = ' + '{:.3e}'.format(abs(Yn - Yo).max()) +
      + '\t\t', end="\r")

Xo = Xn.copy()
Yo = Yn.copy()
# si el metodo iterativo es Jacobi
if metodo == 'J':
    X = Xo
    Y = Yo
else:    # si el metodo es Gauss-Seidel o SOR
    X = Xn
    Y = Yn

(Xn, Yn) = _gen_Poisson_n_flap(X, Y, self.M, self.N, P_,
                                 Q_, self.airfoil_boundary,
                                 union_start)

# se aplica sobre-relajacion si el metodo es SOR
if metodo == 'SOR':
    Xn = omega * Xn + (1 - omega) * Xo
    Yn = omega * Yn + (1 - omega) * Yo

if abs(Xn - Xo).max() < mesh.err_max \
    and abs(Yn - Yo).max() < mesh.err_max:
    print('Poisson: ' + metodo + ': saliendo... ')

```

```

        print('it=' , it)
        break

self.X = Xn
self.Y = Yn

return ( self.X, self.Y)

@jit
def _gen_Poisson_n_flap(X, Y, M, N, P_, Q_, airfoil_boundary,
                        union_start):
    """
    Resuelve los for loops anidados correspondientes a la solucion de
    la ecuacion de Poisson para generar la malla.

    Metodo que se apoya en el uso de la libreria Numba.
    El codigo es el mismo que en el metodo clasico.
    ...

```

Parametros

X : numpy.array
Matriz que contiene las coordenadas X que describen a la malla
Y : numpy.array
Matriz que contiene las coordenadas Y que describen a la malla
M : int
Numero de divisiones en el eje xi.
N : int
Numero de divisiones en el eje eta.
P_ : numpy.array
Valores de la funcion de forzado P para el eje xi
Q_ : numpy.array
Valores de la funcion de forzado Q para el eje eta
airfoil_boundary : numpy.array
*Cada elemento del array indica el punto a que perfil pertenece
 (numero positivo) o cero si no forma parte de una frontera*
union_start : int
*indice "i" en el que inicia la seccion de union entre los
 perfiles*

Return

(X, Y) : numpy.array, numpy.array
Matrices X y Y que describen la malla. Actualizadas.
"""

d_eta = 1

```

d_xi = 1
m = M
n = N

for j in range(n-2, 0, -1):
    for i in range(1, m-1):
        x_eta = (X[i, j+1] - X[i, j-1]) / 2 / d_eta
        y_eta = (Y[i, j+1] - Y[i, j-1]) / 2 / d_eta
        x_xi = (X[i+1, j] - X[i-1, j]) / 2 / d_xi
        y_xi = (Y[i+1, j] - Y[i-1, j]) / 2 / d_xi

        alpha = x_eta ** 2 + y_eta ** 2
        beta = x_xi * x_eta + y_xi * y_eta
        gamma = x_xi ** 2 + y_xi ** 2
        I = x_xi * y_eta - x_eta * y_xi

        X[i, j] = (d_xi * d_eta) ** 2 \
            / (2 * (alpha * d_eta ** 2 + gamma * d_xi ** 2)) \
            * (alpha / (d_xi ** 2) * (X[i+1, j] + X[i-1, j]) \
            + gamma / (d_eta ** 2) * (X[i, j+1] + X[i, j-1]) \
            - beta / (2 * d_xi * d_eta) * (X[i+1, j+1] \
            - X[i+1, j-1] + X[i-1, j-1] - X[i-1, j+1]) \
            + I ** 2 * (P_[i-1] * x_xi + Q_[j-1] * x_eta))

        Y[i, j] = (d_xi * d_eta) ** 2 \
            / (2 * (alpha * d_eta ** 2 + gamma * d_xi ** 2)) \
            * (alpha / (d_xi ** 2) * (Y[i+1, j] + Y[i-1, j]) \
            + gamma / (d_eta ** 2) * (Y[i, j+1] + Y[i, j-1]) \
            - beta / (2 * d_xi * d_eta) * (Y[i+1, j+1] \
            - Y[i+1, j-1] + Y[i-1, j-1] - Y[i-1, j+1]) \
            + I ** 2 * (P_[i-1] * y_xi + Q_[j-1] * y_eta))

i = m-1
x_eta = (X[i, j+1] - X[i, j-1]) / 2 / d_eta
y_eta = (Y[i, j+1] - Y[i, j-1]) / 2 / d_eta
x_xi = (X[1, j] - X[i-1, j]) / 2 / d_xi
y_xi = (Y[1, j] - Y[i-1, j]) / 2 / d_xi

alpha = x_eta ** 2 + y_eta ** 2
beta = x_xi * x_eta + y_xi * y_eta
gamma = x_xi ** 2 + y_xi ** 2
I = x_xi * y_eta - x_eta * y_xi

X[i, j] = (d_xi * d_eta) ** 2 \
    / (2 * (alpha * d_eta ** 2 + gamma * d_xi ** 2)) \
    * (alpha / (d_xi ** 2) * (X[1, j] + X[i-1, j]) \
    + gamma / (d_eta ** 2) * (X[i, j+1] + X[i, j-1]) \
    - beta / (2 * d_xi * d_eta) \

```

```

* (X[1, j+1] - X[1, j-1] + X[i-1, j-1] - X[i-1, j+1]) \
+ I**2 * (P_ [i-1] * x_xi + Q_ [j-1] * x_eta))

X[0, 1:-1] = X[m-1, 1:-1]

# seccion de union entre perfiles
i = union_start
while airfoil_boundary [i] == 0:
    x_eta = (X[i, 1] - X[-i - 1, 1]) / 2 / d_eta
    y_eta = (Y[i, 1] - Y[-i - 1, 1]) / 2 / d_eta
    x_xi = (X[i+1, 0] - X[i-1, 0]) / 2 / d_xi
    y_xi = (Y[i+1, 0] - Y[i-1, 0]) / 2 / d_xi

    alpha = x_eta ** 2 + y_eta ** 2
    beta = x_xi * x_eta + y_xi * y_eta
    gamma = x_xi ** 2 + y_xi ** 2
    I = x_xi * y_eta - x_eta * y_xi

    X[i, 0] = (d_xi * d_eta) ** 2 \
        / (2 * (alpha * d_eta ** 2 + gamma * d_xi ** 2)) \
        * (alpha / (d_xi ** 2) * (X[i+1, 0] + X[i-1, 0]) \
            + gamma / (d_eta ** 2) * (X[i, 1] + X[-i - 1, 1]) \
            - beta / (2 * d_xi * d_eta) * (X[i+1, 1] \
                - X[-i - 2, 1] + X[-i, 1] - X[i-1, 1]))
    Y[i, 0] = (d_xi * d_eta) ** 2 \
        / (2 * (alpha * d_eta ** 2 + gamma * d_xi ** 2)) \
        * (alpha / (d_xi ** 2) * (Y[i+1, 0] + Y[i-1, 0]) \
            + gamma / (d_eta ** 2) * (Y[i, 1] + Y[-i - 1, 1]) \
            - beta / (2 * d_xi * d_eta) * (Y[i+1, 1] \
                - Y[-i - 2, 1] + Y[-i, 1] - Y[i-1, 1]))

    X[-i - 1, 0] = X[i, 0]
    Y[-i - 1, 0] = Y[i, 0]
    i += 1

return (X, Y)

```

```

@jit
def _gen_Poisson_n(X, Y, M, N, P_, Q_):
    """

```

Resuelve los for loops anidados correspondientes a la solucion de la ecuacion de Poisson para generar la malla.

*Metodo que se apoya en el uso de la libreria Numba.
El codigo es el mismo que en el metodo clasico.*

...

Parametros

X : `numpy.array`

Matriz que contiene las coordenadas *X* que describen a la malla

Y : `numpy.array`

Matriz que contiene las coordenadas *Y* que describen a la malla

M : `int`

Numero de divisiones en el eje *xi*.

N : `int`

Numero de divisiones en el eje *eta*.

*P*_ : `numpy.array`

Valores de la funcion de forzado *P* para el eje *xi*

*Q*_ : `numpy.array`

Valores de la funcion de forzado *Q* para el eje *eta*

Return

(X, Y) : `numpy.array, numpy.array`

Matrices *X* y *Y* que describen la malla. Actualizadas.

"""

`d_eta = 1`

`d_xi = 1`

`m = M`

`n = N`

for *j* **in** `range(n-2, 0, -1)`:

for *i* **in** `range(1, m-1)`:

x_eta = (*X*[*i*, *j*+1] - *X*[*i*, *j*-1]) / 2 / `d_eta`

y_eta = (*Y*[*i*, *j*+1] - *Y*[*i*, *j*-1]) / 2 / `d_eta`

x_xi = (*X*[*i*+1, *j*] - *X*[*i*-1, *j*]) / 2 / `d_xi`

y_xi = (*Y*[*i*+1, *j*] - *Y*[*i*-1, *j*]) / 2 / `d_xi`

alpha = *x_eta* ** 2 + *y_eta* ** 2

beta = *x_xi* * *x_eta* + *y_xi* * *y_eta*

gamma = *x_xi* ** 2 + *y_xi* ** 2

I = *x_xi* * *y_eta* - *x_eta* * *y_xi*

X[*i*, *j*] = (`d_xi` * `d_eta`) ** 2 \\\n / (2 * (*alpha* * `d_eta` ** 2 + *gamma* * `d_xi` ** 2)) \\\n * (*alpha* / (`d_xi` ** 2) * (*X*[*i*+1, *j*] + *X*[*i*-1, *j*]) \\\n + *gamma* / (`d_eta` ** 2) * (*X*[*i*, *j*+1] + *X*[*i*, *j*-1]) \\\n - *beta* / (2 * `d_xi` * `d_eta`) * (*X*[*i*+1, *j*+1] \\\n - *X*[*i*+1, *j*-1] + *X*[*i*-1, *j*-1] - *X*[*i*-1, *j*+1]) \\\n + *I* ** 2 * (*P*[*i*-1] * *x_xi* + *Q*[*j*-1] * *x_eta*)) \\\n *Y*[*i*, *j*] = (`d_xi` * `d_eta`) ** 2 \\\n

```

/ (2 * (alpha * d_eta**2 + gamma * d_xi**2)) \
* (alpha / (d_xi**2) * (Y[i+1, j] + Y[i-1, j]) \
+ gamma / (d_eta**2) * (Y[i, j+1] + Y[i, j-1]) \
- beta / (2 * d_xi * d_eta) * (Y[i+1, j+1] \
- Y[i+1, j-1] + Y[i-1, j-1] - Y[i-1, j+1]) \
+ I**2 * (P_[i-1] * y_xi + Q_[j-1] * y_eta))

i      = m-1
x_eta  = (X[i, j+1] - X[i, j-1]) / 2 / d_eta
y_eta  = (Y[i, j+1] - Y[i, j-1]) / 2 / d_eta
x_xi   = (X[1, j] - X[i-1, j]) / 2 / d_xi
y_xi   = (Y[1, j] - Y[i-1, j]) / 2 / d_xi

alpha   = x_eta ** 2 + y_eta ** 2
beta    = x_xi * x_eta + y_xi * y_eta
gamma   = x_xi ** 2 + y_xi ** 2
I       = x_xi * y_eta - x_eta * y_xi

X[i, j] = (d_xi * d_eta) ** 2 \
/ (2 * (alpha * d_eta**2 + gamma * d_xi**2)) \
* (alpha / (d_xi**2) * (X[1, j] + X[i-1, j]) \
+ gamma / (d_eta**2) * (X[i, j+1] + X[i, j-1]) \
- beta / (2 * d_xi * d_eta) \
* (X[1, j+1] - X[1, j-1] + X[i-1, j-1] - X[i-1, j+1]) \
+ I**2 * (P_[i-1] * x_xi + Q_[j-1] * x_eta))

X[0, 1:-1] = X[m-1, 1:-1]

return (X, Y)

```

A.3. Conversión de mallas a formato SU2

"""

@author: Marco Antonio Cardoso Moreno
@mail: marcoacardosom@gmail.com

Scripts para convertir mallas a formato de SU2

Documentacion: <https://su2code.github.io/docs/Mesh-File>
"""

```

import numpy as np
import matplotlib.pyplot as plt

from util.helpers import get_size_airfoil, get_size_airfoil_n_flap

def to_su2_mesh_o_airfoil(mesh, filename):

```

```

, , ,
Convierte malla de formato propio a formato de SU2
Para mallas tipo O
Con solo un perfil (o cualquier geometria)
, , ,

# importa coordenadas de mesh, se quita ultima fila (repetida).
X           = np.copy(mesh.X)[-1:, :].transpose()
Y           = np.copy(mesh.Y)[-1:, :].transpose()

# convirtiendo a arreglo 1D
X           = X.flatten()
Y           = Y.flatten()

NPOIN       = np.shape(X)[0]

# creando archivo de malla para SU2
su2_mesh    = open(filename, 'w')

su2_mesh.write('NDIME= 2\n')
su2_mesh.write('NPOIN= ' + str(NPOIN) + '\n')

# se escriben las coordenadas de los nodos
for i in range(NPOIN):
    su2_mesh.write(str(X[i]) + '\t' + str(Y[i]) + '\n')

# se escriben las celdas y la conectividad entre nodos que la
# forman
NELEM = (mesh.M - 1) * (mesh.N - 1)
su2_mesh.write('NELEM= ' + str(NELEM) + '\n')
for i in range(NELEM):
    # condicion para excluir ultimo volumen del nivel. Al terminar
    # vuelta
    if i % (mesh.M - 1) != mesh.M - 2:
        su2_mesh.write('9 ' + str(i) + ', ' + str(i + 1) + ', '
                      + str(i + mesh.M) + ', ' + str(i + mesh.M - 1)
                      + '\n')
    else:
        su2_mesh.write('9 ' + str(i) + ', ' + str(i - (mesh.M - 2))
                      + ', ' + str(i + 1) + ', '
                      + str(i + mesh.M - 1) + '\n')

# se escriben las fronteras. Primero FE, luego FI
NMARK = 2
su2_mesh.write('NMARK= ' + str(NMARK) + '\n')
su2_mesh.write('MARKERTAG= farfield\n')
su2_mesh.write('MARKERELEMS= ' + str(mesh.M - 1) + '\n')

```

```

far1 = (mesh.M - 1) * mesh.N
far0 = far1 - (mesh.M - 1)
for i in range(far0, far1 - 1):
    su2_mesh.write('3 ' + str(i) + ', ' + str(i + 1) + '\n')
su2_mesh.write('3 ' + str(i + 1) + ', ' + str(far0) + '\n')

su2_mesh.write('MARKERTAG= airfoil\n')
su2_mesh.write('MARKERELEMS= ' + str(mesh.M - 1) + '\n')
for i in range(mesh.M - 2):
    su2_mesh.write('3 ' + str(i) + ', ' + str(i + 1) + '\n')
su2_mesh.write('3 ' + str(i + 1) + ', ' + str(0) + '\n')
su2_mesh.close()

return

def to_su2_mesh_o_airfoil_n_flap(mesh, filename):
    """
    Convierte malla de formato propio a formato de SU2
    Para mallas tipo O
    Para perfiles con external airfoil flap (o en general,
    , , , 2 geometrias separadas)
    """

    size_airfoil, size_flap = get_size_airfoil_n_flap(\n
                                                mesh.airfoil_boundary[:-1])

    # creando archivo de malla para SU2
    su2_mesh = open(filename, 'w')

    M_SU2 = mesh.M - 1
    N_SU2 = mesh.N - 1
    NPOIN = M_SU2 * N_SU2 + M_SU2 - 2 - mesh.airfoil_join

    # importa coordenadas de mesh, se quita ultima fila (repetida).
    X = np.copy(mesh.X)[:-1, :]
    Y = np.copy(mesh.Y)[:-1, :]

    # extraer primera columna (perfiles) para eliminar puntos
    # repetidos
    x_perfil = X[:, 0]
    y_perfil = Y[:, 0]
    end = size_flap // 2 + 1 + mesh.airfoil_join \
          + size_airfoil - 1
    x_perf1 = x_perfil[:end]
    y_perf1 = y_perfil[:end]
    begin = end + mesh.airfoil_join + 2
    x_perf2 = x_perfil[begin:]
    y_perf2 = y_perfil[begin:]

```

```

x_perfil      = np.concatenate((x_perf1, x_perf2))
y_perfil      = np.concatenate((y_perf1, y_perf2))
eta_0          = np.shape(x_perfil)[0]

# convirtiendo a arreglo 1D
X             = X[:, 1:]
Y             = Y[:, 1:]
X             = X.transpose().flatten()
Y             = Y.transpose().flatten()
X             = np.concatenate((x_perfil, X))
Y             = np.concatenate((y_perfil, Y))

# se inicia escritura de archivo
su2_mesh.write('NDIME= 2\n')
su2_mesh.write('NPOIN= ' + str(NPOIN) + '\n')

# se escriben las coordenadas de los nodos
for i in range(NPOIN):
    su2_mesh.write(str(X[i]) + 't' + str(Y[i]) + '\n')

# se escriben las celdas y la conectividad entre los nodos que la
# forman
NELEM = M_SU2 * N_SU2
su2_mesh.write('NELEM= ' + str(NELEM) + '\n')

# primera parte de celdas conectadas al perfil
size_airfoils = np.shape(x_perfil)[0]
end           = size_flap // 2 + 1 + mesh.airfoil_join \
                + size_airfoil - 2

for i in range(end):
    su2_mesh.write('9 ' + str(i) + ' ' + str(i+1) + ' '
                  + str(i + size_airfoils + 1) + ' '
                  + str(i + size_airfoils) + '\n')

# segunda parte, cubre el "regreso" en la O, cubre ultimo pedazo
# de perfil y la union
begin         = end
extrados_flap = end + 1
end           += mesh.airfoil_join + 2
diff          = begin - size_airfoil + 2

su2_mesh.write('9 ' + str(begin) + ' ' + str(diff) + ' '
              + str(begin + size_airfoils + 1) + ' '
              + str(begin + size_airfoils) + '\n')

```

```

begin          += 1

for i in range(begin, end):
    su2_mesh.write('9 ' + str(diff) + ' ' + str(diff - 1) + ' '
                  + str(i + size_airfoils + 1) + ' '
                  + str(i + size_airfoils) + '\n')
diff -= 1

# primer celda extrados flap
begin = end
su2_mesh.write('9 ' + str(diff) + ' ' + str(extrados_flap) + ' '
              + str(begin + size_airfoils + 1) + ' '
              + str(begin + size_airfoils) + '\n')

# a partir de este punto todas las celdas siguen la misma
# secuencia a partir de extrados de flap
begin += 1
diff = begin - extrados_flap
for i in range(begin, M_SU2):
    if i % (M_SU2) != M_SU2 - 1:
        su2_mesh.write('9 ' + str(i - diff) + ' '
                      + str(i - diff + 1) + ' '
                      + str(i + size_airfoils + 1) + ' '
                      + str(i + size_airfoils) + '\n')
    else:
        su2_mesh.write('9 ' + str(i - diff) + ' '
                      + str(i - mesh.M + 2)
                      + ' ' + str(i - diff + 1) + ' '
                      + str(i + size_airfoils) + '\n')

begin = M_SU2
for i in range(begin, NELEM):
    if i % (M_SU2) != M_SU2 - 1:
        su2_mesh.write('9 ' + str(i - diff) + ' '
                      + str(i - diff + 1) + ' '
                      + str(i + size_airfoils + 1) + ' '
                      + str(i + size_airfoils) + '\n')
    else:
        su2_mesh.write('9 ' + str(i - diff) + ' '
                      + str(i - mesh.M + 2 - diff) + ' '
                      + str(i - diff + 1) + ' '
                      + str(i + size_airfoils) + '\n')

# se escriben las fronteras. Primero FE, luego FI
NMARK = 3
su2_mesh.write('NMARK= ' + str(NMARK) + '\n')
su2_mesh.write('MARKERTAG= farfield\n')
su2_mesh.write('MARKERELEMS= ' + str(mesh.M - 1) + '\n')

```

```

far0 = NPOIN - M_SU2
for i in range(far0 , NPOIN - 1):
    su2_mesh.write('3 ' + str(i) + ' ' + str(i + 1) + '\n')
su2_mesh.write('3 ' + str(i + 1) + ' ' + str(far0) + '\n')

# frontera airfoil
su2_mesh.write('MARKERTAG= airfoil\n')
su2_mesh.write('MARKERELEMS= ' + str(size_airfoil - 1) + '\n')

begin = size_flap // 2 + mesh.airfoil_join + 1
end = begin + size_airfoil - 2
for i in range(begin , end):
    su2_mesh.write('3 ' + str(i) + ' ' + str(i + 1) + '\n')
su2_mesh.write('3 ' + str(i + 1) + ' ' + str(begin) + '\n')

# frontera flap
su2_mesh.write('MARKERTAG= flap\n')
su2_mesh.write('MARKERELEM= ' + str(size_flap - 1) + '\n')
for i in range(size_flap // 2):
    su2_mesh.write('3 ' + str(i) + ' ' + str(i + 1) + '\n')

begin = size_flap // 2 + mesh.airfoil_join + size_airfoil
end = begin + (size_flap - 1)
su2_mesh.write('3 ' + str(size_flap // 2) + ' ' + str(begin)
              + '\n')

for i in range(begin , eta_0 - 1):
    su2_mesh.write('3 ' + str(i) + ' ' + str(i + 1) + '\n')

i = eta_0 - 1
su2_mesh.write('3 ' + str(i) + ' ' + str(0) + '\n')
su2_mesh.close()

return

def to_su2_mesh_c_airfoil(mesh , filename):
    """
    Convierte malla de formato propio a formato de SU2
    Para mallas tipo O
    Con solo un perfil (o cualquier geometria)
    """

# guardando X y Y de la malla
X = mesh.X
Y = mesh.Y
size_airfoil = get_size_airfoil(mesh.airfoil_boundary)

```

```

diff           = mesh.M - size_airfoil
diff          /= 2
intrados      = np.zeros((diff))
extrados      = intrados
is_bound      = np.concatenate([intrados, mesh.airfoil_boundary])
is_bound      = np.concatenate([is_bound, extrados])

# extraer primera columna (perfils) para eliminar puntos
# repetidos
x_perfil      = X[:, 0]
y_perfil      = Y[:, 0]

x_perfil      = x_perfil[: size_airfoil - 1 + diff]
y_perfil      = y_perfil[: size_airfoil - 1 + diff]

# se elimina j = 0 y se hace arreglo 1D
X             = X[:, 1: ].transpose().flatten()
Y             = Y[:, 1: ].transpose().flatten()

X             = np.concatenate([x_perfil, X])
Y             = np.concatenate([y_perfil, Y])

NPOIN         = np.shape(X)[0]

# creando archivo
su2_mesh      = open(filename, 'w')

# numero de dimensiones
su2_mesh.write('NDIME= 2\n')

# lista de nodos
su2_mesh.write('NPOIN= ' + str(NPOIN) + '\n')
for i in range(np.shape(X)[0]):
    su2_mesh.write(str(X[i]) + '\t' + str(Y[i]) + '\n')

# se enlistan las celdas que forman el dominio
NELEM = (mesh.M - 1) * (mesh.N - 1)
su2_mesh.write('NELEM= ' + str(NELEM) + '\n')

# primero de inicio de la malla a final del perfil
end = diff + size_airfoil - 2
for i in range(end):
    su2_mesh.write('9 ' + str(i) + ', ' + str(i + 1) + ', '
                  + str(i + end + 2) + ', ' + str(i + end + 1)
                  + '\n')

# de fin del perfil a regreso de la C en eta = 0

```

```

i += 1
su2_mesh.write('9 ' + str(i) + ' ' + str(diff) + ' '
               + str(i + end + 2)
               + ' ' + str(i + end + 1) + '\n')

i += 1
for j in range(diff, 0, -1):
    su2_mesh.write('9 ' + str(j) + ' ' + str(j - 1) + ' '
                  + str(i + end + 2) + ' ' + str(i + end + 1)
                  + '\n')
i += 1

# resto de la malla, a partir de eta = 1
begin      = mesh.M - 1
jump       = mesh.M - 3
i          = 0
diff_NELEM = diff
while begin < NELEM:
    if i == mesh.M - 1:
        diff_NELEM -= 1
        i = 0
    first     = begin - diff_NELEM
    second    = first + 1
    third     = second + mesh.M
    fourth    = first + mesh.M
    su2_mesh.write('9 ' + str(first) + ' ' + str(second) + ' '
                  + str(third) + ' ' + str(fourth) + '\n')
    i         += 1
    begin    += 1

# se especifican las fronteras
su2_mesh.write('NMARK= 2\n')

# frontera externa
su2_mesh.write('MARKERTAG= farfield\n')
MARKER_ELEMS = mesh.M - 1 + (mesh.N - 1) * 2
su2_mesh.write('MARKER_ELEMS= ' + str(MARKER_ELEMS) + '\n')

# parte inferior de C
begin = diff + size_airfoil - 1
su2_mesh.write('3 ' + str(0) + ' ' + str(begin) + '\n')
for i in range(mesh.N - 2):
    end     = begin + mesh.M
    su2_mesh.write('3 ' + str(begin) + ' ' + str(end) + '\n')
    begin   = end

# parte C

```

```

end += mesh.M - 1
for i in range(begin, end):
    su2_mesh.write('3 ' + str(i) + ' ' + str(i + 1) + '\n')

# parte superior de C
begin = end
for i in range(mesh.N - 2):
    end = begin - mesh.M
    su2_mesh.write('3 ' + str(begin) + ' ' + str(end) + '\n')
    begin = end
su2_mesh.write('3 ' + str(begin) + ' ' + str(0) + '\n')

# frontera interna
MARKERELEMS = size_airfoil - 1
begin = diff
end = diff + size_airfoil - 2
su2_mesh.write('MARKERTAG= airfoil\n')
su2_mesh.write('MARKERELEMS= ' + str(MARKERELEMS) + '\n')
for i in range(begin, end):
    su2_mesh.write('3 ' + str(i) + ' ' + str(i + 1) + '\n')
su2_mesh.write('3 ' + str(i + 1) + ' ' + str(begin) + '\n')
su2_mesh.close()

return

def to_su2_mesh_c_airfoil_n_flap(mesh, filename):
    """
    Convierte malla de formato propio a formato de SU2
    Para mallas tipo C
    Para perfiles con external airfoil flap (o en general,
    2 geometrias separadas)
    """

    size_airfoil, size_flap = get_size_airfoil_n_flap(
        mesh.airfoil_boundary[:-1])

    union = mesh.airfoil_join
    diff = mesh.M - (size_airfoil + size_flap + 1
                    + union * 2)
    diff // = 2

# creando archivo de malla para SU2
su2_mesh = open(filename, 'w')

M_SU2 = mesh.M - 1
N_SU2 = mesh.N - 1
NPOIN = size_airfoil - 1 + size_flap - 1 + union + diff

```

```

NPOIN          += mesh.M * (N_SU2)
NELEM          = M_SU2 * N_SU2

# importa coordenadas de mesh, se quita ultima fila (repetida).
X              = np.copy(mesh.X)
Y              = np.copy(mesh.Y)

# extraer primera columna (perfils) para eliminar puntos
# repetidos
x_perfil      = X[:, 0]
y_perfil      = Y[:, 0]
end           = diff + size_flap // 2 + 1 + mesh.airfoil_join \
                + size_airfoil - 1
x_perf1       = x_perfil [: end]
y_perf1       = y_perfil [: end]
begin         = end + mesh.airfoil_join + 2
end           = -diff - 1
x_perf2       = x_perfil [begin : end]
y_perf2       = y_perfil [begin : end]
x_perfil      = np.concatenate((x_perf1, x_perf2))
y_perfil      = np.concatenate((y_perf1, y_perf2))
eta_0          = np.shape(x_perfil)[0]

# convirtiendo a arreglo 1D
X              = X[:, 1:]
Y              = Y[:, 1:]
X              = X.transpose().flatten()
Y              = Y.transpose().flatten()
X              = np.concatenate((x_perfil, X))
Y              = np.concatenate((y_perfil, Y))

# numero de dimensiones
su2_mesh.write('NDIME= 2\n')

# lista de nodos
su2_mesh.write('NPOIN= ' + str(NPOIN) + '\n')
for i in range(NPOIN):
    su2_mesh.write(str(X[i]) + '\t' + str(Y[i]) + '\n')

# se enlistan las celdas que forman el dominio
su2_mesh.write('NELEM= ' + str(NELEM) + '\n')

# primero de inicio de la malla a final del perfil
end = diff + size_flap // 2 + 1 + union + size_airfoil - 2
for i in range(end):
    su2_mesh.write('9 ' + str(i) + ', ' + str(i + 1) + ', '
                  + str(i + eta_0 + 1) + ', ' + str(i + eta_0))

```

```

+ '\n')

# de fin del perfil a inicio de union
i += 1
su2_mesh.write('9 ' + str(i) + ' ' + str(i - size_airfoil + 2)
               + ' ' + str(i + eta_0 + 1) + ' ' + str(i + eta_0)
               + '\n')

# union de regreso
begin = i - size_airfoil + 2
for j in range(begin, begin - union - 1, -1):
    i += 1
    su2_mesh.write('9 ' + str(j) + ' ' + str(j - 1) + ' '
                  + str(i + eta_0 + 1) + ' ' + str(i + eta_0)
                  + '\n')

# primer celda del borde de ataque de flap. regreso
i += 1
j -= 1
end += 1
su2_mesh.write('9 ' + str(j) + ' ' + str(end) + ' '
               + str(i + eta_0 + 1) + ' ' + str(i + eta_0) + '\n')

# resto del flap
begin = end
end = begin + size_airfoil // 2 - 2
for j in range(begin, end):
    i += 1
    su2_mesh.write('9 ' + str(j) + ' ' + str(j + 1) + ' '
                  + str(i + eta_0 + 1) + ' ' + str(i + eta_0)
                  + '\n')

# ultima celda borde de salida flap
i += 1
su2_mesh.write('9 ' + str(eta_0 - 1) + ' ' + str(diff) + ' '
               + str(i + eta_0 + 1) + ' ' + str(i + eta_0)
               + '\n')

# ultima seccion de regreso de la C
for j in range(diff, 0, -1):
    i += 1
    su2_mesh.write('9 ' + str(j) + ' ' + str(j - 1) + ' '
                  + str(i + eta_0 + 1) + ' ' + str(i + eta_0)
                  + '\n')

# resto de la malla, a partir de eta = 1
begin = M_SU2

```

```

jump      = mesh.M - 3
i         = 0
diff_NELEM = begin - eta_0
while begin < NELEM:
    if i == mesh.M - 1:
        diff_NELEM -= 1
        i = 0
    first   = begin - diff_NELEM
    second  = first + 1
    third   = second + mesh.M
    fourth  = first + mesh.M
    su2_mesh.write( '9 ' + str(first) + ' ' + str(second) + ' '
                    + str(third) + ' ' + str(fourth) + '\n')
    i += 1
    begin += 1

# se especifican las fronteras
su2_mesh.write( 'NMARK= 3\n' )

# frontera externa
su2_mesh.write( 'MARKER_TAG= farfield\n' )
MARKER_ELEMS = M_SU2 + N_SU2 * 2
su2_mesh.write( 'MARKER_ELEMS= ' + str(MARKER_ELEMS) + '\n' )

# parte inferior de C
begin = eta_0
su2_mesh.write( '3 ' + str(0) + ' ' + str(begin) + '\n' )
for i in range(mesh.N - 2):
    end = begin + mesh.M
    su2_mesh.write( '3 ' + str(begin) + ' ' + str(end) + '\n' )
    begin = end

# parte de la C en sentido horario
begin = NPOIN - mesh.M
for i in range(begin, NPOIN - 1):
    su2_mesh.write( '3 ' + str(i) + ' ' + str(i + 1) + '\n' )

begin    = NPOIN - 1
end      = begin - mesh.M
for i in range(N_SU2 - 1):
    su2_mesh.write( '3 ' + str(begin) + ' ' + str(end) + '\n' )
    begin    = end
    end      -= mesh.M
su2_mesh.write( '3 ' + str(begin) + ' ' + str(0) + '\n' )

# airfoil
MARKER_ELEMS    = size_airfoil - 1

```

```

begin      = diff + size_flap // 2 + 1 + union
end        = begin + size_airfoil - 2
su2_mesh.write( 'MARKER_TAG= airfoil\n')
su2_mesh.write( 'MARKER_ELEMS= ' + str(MARKER_ELEMS) + '\n')
for i in range(begin, end):
    su2_mesh.write('3 ' + str(i) + ' ' + str(i + 1) + '\n')
i += 1
su2_mesh.write('3 ' + str(i) + ' ' + str(begin) + '\n')
end_airfoil = i

# flap
MARKER_ELEMS = size_flap - 1
su2_mesh.write( 'MARKER_TAG= flap\n')
su2_mesh.write( 'MARKER_ELEMS= ' + str(MARKER_ELEMS) + '\n')

# intrados hasta borde de ataque
begin      = diff
end        = begin + size_flap // 2
for i in range(begin, end):
    su2_mesh.write('3 ' + str(i) + ' ' + str(i + 1) + '\n')

# primer elemento borde de ataque extrados
i          += 1
end_airfoil += 1
su2_mesh.write('3 ' + str(i) + ' ' + str(end_airfoil) + '\n')

# resto de extrados. regreso
begin      = end_airfoil
end        = begin + size_flap // 2 - 2
i          = begin
for i in range(begin, end):
    su2_mesh.write('3 ' + str(i) + ' ' + str(i + 1) + '\n')

su2_mesh.write('3 ' + str(eta_0 - 1) + ' ' + str(diff) + '\n')
su2_mesh.close()

return

```

A.4. Archivo main. Ejemplo de ejecución

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""

@author: cardoso
@mail: marcoacardosom@gmail.com

```

Archivo main. Ejemplo de ejecucion de codigo.
"""

```
# importar dependencias de sistema
import numpy as np
import matplotlib.pyplot as plt

# importar modulos del programa
import airfoil
import mesh
import mesh_c
import mesh_o
import mesh_su2
from potential import potential_flow_o, velocity, pressure,\n    lift_n_drag, streamlines, potential_flow_o_n
import util

#####
# PERFIL AERODINAMICO
#####
# importar desde archivo
airfoil_points = 549
c = 1
filename = 'nombre_archivo_nube_de_puntos.csv'
perfil = airfoil.airfoil(c)
perfil.create(filename)

# perfil NACA 4
m = 0
p = 0
t = 12
c = 1
perfil = airfoil.NACA4(m, p, t, c)
perfil.create_sin(airfoil_points)

#####
# FLAP
#####
# importar desde archivo
filename_flap = 'nombre_archivo_nube_de_puntos_flap.csv'
flap = airfoil.airfoil(filename, number=2)

# flap NACA 4
m_f = 2
```

```

p_f = 4
t_f = 12
c_f = 0.2 * c
flap = airfoil.NACA4(m_f, p_f, t_f, c_f, number=2)
flap.create_sin(airfoil_points)
flap.rotate(15)

# union de perfil y flap
union = 39
dx = 0.055
dy = 0.05
perfil.join(flap, dx, dy, union=union)

# rotacion del angulo de ataque del perfil.
# Configuracion completa
alpha = 5
perfil.rotate(alpha)

#####
# MALLA
#####
# tipo de malla (C, O)
malla_tipo = 'O'

# dimensiones de la malla. N (\eta), airfoil-points = M (\xi)
N = 275

# dimension de frontera externa
R = 400 * c

if malla_tipo == 'O':
    mallaNACA = mesh_o.mesh_O(R, N, perfil)
elif malla_tipo == 'C':
    mallaNACA = mesh_c.mesh_C(R, N, perfil)

# metodos de generacion de malla
metodo_iterativo = 'SOR' # 'GS', 'J'
mallaNACA.gen_Laplace_n(metodo='SOR')
# mallaNACA.gen_Laplace_v(metodo='SOR')
# mallaNACA.gen_Laplace(metodo='SOR')
mallaNACA.gen_Poisson_n(metodo='SOR', omega=0.15, aa=1500, cc=12,
                        linea_eta=0)
# mallaNACA.gen_Poisson_v(metodo='SOR', omega=0.15, aa=1500, cc=12,
    # linea_eta=0)
# mallaNACA.gen_Poisson(metodo='SOR', omega=0.15, aa=1500, cc=12,
    # linea_eta=0)

```

```

# importar malla
malla_importada = 'nombre_archivo_malla.txt_mesh'
mallaNACA = util.from_txt_mesh(malla_importada)

# grafica de malla
mallaNACA.plot()

#####
# CALIDAD DE LA MALLA
#####
aspect_ratio = mallaNACA.get_aspect_ratio()
skew = mallaNACA.get_skew()
cmap = 'viridis'

# grafica aspect_ratio
plt.figure('aspect_ratio')
plt.title('aspect_ratio')
plt.pcolormesh(mallaNACA.X, mallaNACA.Y, aspect_ratio, cmap=cmap)

# guardar grafica
plt.savefig('nombre_de_imagen.png',
            bbox_inches='tight', pad_inches=0.05)

# mostrar grafica
plt.show()

# grafica skew
plt.figure('skew')
plt.title('skew')
plt.pcolormesh(mallaNACA.X, mallaNACA.Y, skew, cmap=cmap)

# guardar grafica
plt.savefig('nombre_de_imagen.png',
            bbox_inches='tight', pad_inches=0.05)

# mostrar grafica
plt.show()

#####
# SU2
#####
malla_su2 = 'nombre_archivo_malla.su2'
mallaNACA.to_su2(malla_su2)

```

```

#####
# FLUJO POTENCIAL. Unicamente mallas tipo O
#####
# variables del flujo
t_inf = 293.15 # [K]
p_inf = 101325 # [Pa]
v_inf = 48 # [m / s]
alfa = 5
gamma = 1.4
cp_ = 1007
mu = 18.25e-6
Rg = cp_ * (gamma - 1) / gamma
d_inf = p_inf / (Rg * t_inf)
h_inf = cp_ * t_inf
c_inf = (gamma * p_inf / d_inf) ** 0.5

# relaciones isentropicas
h0 = h_inf + 0.5 * v_inf ** 2
d0 = d_inf * (1 + (gamma - 1) / 2
               * (v_inf / c_inf) ** 2) ** (1 / (gamma - 1))
p0 = p_inf * (d0 / d_inf) ** gamma
mach_inf = v_inf / c_inf
Re = v_inf * c * d_inf / mu

mach_inf = v_inf / c_inf
Re = v_inf * c * d_inf / mu
alphas = [-4, -2, 0, 2, 4, 6, 8, 10]
if mach_inf > 0.8:
    print('Las condiciones de flujo son invalidas')
    exit()

for alpha_ in alphas:
    alpha = int(alpha_)
    (phi, C, theta, IMA) = potential_flow_o_n(d0, h0, gamma,
                                                mach_inf, v_inf,
                                                alpha, mallaNACA)
    (u, v) = velocity(alfa, C, mach_inf, theta, mallaNACA,
                      phi, v_inf)
    (cp, p) = pressure(u, v, v_inf, d_inf, gamma, p_inf, p0,
                        d0, h0)
    (psi, mach) = streamlines(u, v, gamma, h0, d0, p, mallaNACA)
    (L, _) = lift_n_drag(mallaNACA, cp, alfa, c)

    plt.figure('potential')
    plt.contour(mallaNACA.X, mallaNACA.Y, phi, 95, cmap='viridis')
    plt.colorbar()

```

```

plt.plot(mallaNACA.X[:, 0], mallaNACA.Y[:, 0], 'k')
plt.plot(mallaNACA.X[:, -1], mallaNACA.Y[:, -1], 'k')
plt.axis('equal')

plt.figure('pressure')
plt.plot(mallaNACA.X[:, 0], mallaNACA.Y[:, 0], 'k')
plt.contourf(mallaNACA.X, mallaNACA.Y, cp, 75, cmap='viridis')
plt.colorbar()
plt.axis('equal')

plt.figure('streamlines')
plt.plot(mallaNACA.X[:, 0], mallaNACA.Y[:, 0], 'k')
plt.contour(mallaNACA.X, mallaNACA.Y, np.real(psi), 195,
            cmap='viridis')
plt.colorbar()
plt.axis('equal')

plt.draw()
plt.show()

```

Apéndice B

Métodos Numéricos Iterativos

Existen en general, dos métodos de solución de sistemas de ecuaciones algebráicas lineales simultáneas: métodos directos y métodos iterativos.

Dentro de los métodos directos podemos encontrar los métodos de Cramer y de Gauss. La principal desventaja que presentan estos métodos es el gran número de operaciones aritméticas necesarias para obtener una solución del sistema. Desde el punto de vista computacional otras desventajas que presentan son el uso de memoria y cierta dificultad para ser programados. Los métodos iterativos son simples y fáciles de programar, por lo que se presentan como una mejor herramienta para llevar a cabo la solución del sistema. La idea de estos métodos, tal y como su nombre lo indica, es obtener la solución mediante un proceso iterativo, por lo general se asume una primera solución y con dichos valores propuestos se calculan nuevos valores de las incógnitas, con base en los nuevos valores calculados se obtienen nuevos valores. Este proceso se repite hasta satisfacer el criterio de convergencia establecido.

B.1. Métodos iterativos: Jacobi, Gauss-Seidel, SOR

B.1.1. Método de Jacobi

Para resolver una ecuación matricial de la forma $Au = d$ para el vector u de dimensiones $n \times 1$ suelen utilizarse métodos iterativos. Se pueden escribir las ecuaciones para cada término del vector u (asumiendo que ningún elemento de la diagonal principal de la matriz es igual a 0) de la siguiente manera

$$u_1 = \frac{1}{a_{11}} [d_1 - (a_{12}u_2 + a_{13}u_3 + \cdots + a_{1n}u_n)]$$

$$u_2 = \frac{1}{a_{22}} [d_2 - (a_{21}u_1 + a_{23}u_3 + \cdots + a_{2n}u_n)]$$

...

$$u_n = \frac{1}{a_{nn}} [d_n - (a_{n1}u_1 + a_{n2}u_2 + \cdots + a_{n(n-1)}u_{n-1})]$$

A partir de estas ecuaciones se derivan varios esquemas iterativos, dada una solución inicial $u_i^{(1)}, i = 1, 2, \dots, n$. El primero de ellos es el método de Jacobi, en éste, la variable dependiente se

calcula usando los datos previamente obtenidos, quedando sus ecuaciones como

$$\begin{aligned}
 u_1^{k+1} &= \frac{1}{a_{11}} [d_1 - (a_{12}u_2^k + a_{13}u_3^k + \cdots + a_{1n}u_n^k)] \\
 u_2^{k+1} &= \frac{1}{a_{22}} [d_2 - (a_{21}u_1^k + a_{23}u_3^k + \cdots + a_{2n}u_n^k)] \\
 &\quad \dots \\
 u_n^{k+1} &= \frac{1}{a_{nn}} [d_n - (a_{n1}u_1^k + a_{n2}u_2^k + \cdots + a_{n(n-1)}u_{n-1}^k)]
 \end{aligned} \tag{B.3}$$

La ecuación B.3 es la ecuación constitutiva del método iterativo de Jacobi, en ella k representa los valores previamente calculados, es decir, obtenidos de la iteración anterior o de la solución inicial, según sea el caso. El superíndice $k + 1$ indica el valor obtenido en la iteración actual.

B.1.2. Método de Gauss-Seidel

El siguiente método que puede desarrollarse es el método de Gauss-Seidel, el cual utiliza los valores calculados en la iteración actual para calcular los siguientes valores de la variable dependiente. Sus ecuaciones se escriben como

$$\begin{aligned}
 u_1^{k+1} &= \frac{1}{a_{11}} [d_1 - (a_{12}u_2^k + a_{13}u_3^k + \cdots + a_{1n}u_n^k)] \\
 u_2^{k+1} &= \frac{1}{a_{22}} [d_2 - (a_{21}u_1^{k+1} + a_{23}u_3^k + \cdots + a_{2n}u_n^k)] \\
 &\quad \dots \\
 u_n^{k+1} &= \frac{1}{a_{nn}} [d_n - (a_{n1}u_1^{k+1} + a_{n2}u_2^{k+1} + \cdots + a_{n(n-1)}u_{n-1}^{k+1})]
 \end{aligned} \tag{B.5}$$

en este método se calcula un nuevo valor u_1^{k+1} con los valores de la iteración anterior, dicho valor luego es ocupado para el cálculo de u_2^{k+1} , ambos términos son utilizados para obtener u_3^{k+1} y el proceso continúa de la misma manera hasta llegar a la última ecuación. Es decir, el método de Gauss-Seidel implica el uso inmediato de los valores u_i^{k+1} tan pronto como están disponibles, lo que da como resultado un uso menor de memoria en el ordenador, además de satisfacer el criterio de convergencia en menos iteraciones comparado con el método de Jacobi. La ecuación B.5 puede escribirse de la siguiente manera

$$u_i^{k+1} = \frac{1}{a_{ii}} \left[d_i - \sum_{j=1}^{i-1} a_{ij}u_j^{k+1} - \sum_{j=i+1}^n a_{ij}u_j^k \right], \quad i = 1, 2, \dots, n \tag{B.6}$$

B.1.3. Método de Sobre-relajación SOR

Si durante el proceso de solución se percibe una tendencia en los valores calculados se puede utilizar la dirección de cambio para extrapolar en la siguiente iteración y por lo tanto, acelerar el proceso de solución. A este procedimiento se le conoce como método SOR (Successive Over-Relaxation o

Sobre-relajación sucesiva). Este método introduce un parámetro extra ω conocido como parámetro de aceleración, el cual puede acelerar la convergencia. Dicho método está representado por

$$\begin{aligned} u_i^{k+1} &= u_i^k + \frac{\omega}{a_{ii}} \left[d_i - \sum_{j=1}^{i-1} a_{ij} u_j^{k+1} - a_{ii} u_i^k - \sum_{j=i+1}^n a_{ij} u_j^k \right] \\ &= \frac{\omega}{a_{ii}} \left[d_i - \sum_{j=1}^{i-1} a_{ij} u_j^{k+1} - \sum_{j=i+1}^n a_{ij} u_j^k \right] + (1 - \omega) u_i^k, \quad i = 1, 2, \dots, n \end{aligned} \tag{B.7}$$

Cuando $\omega = 1$ el método SOR se convierte en el método de Gauss-Seidel. Si $1 < \omega < 2$ se está trabajando con sobre-relajación, mientras que al utilizar un valor $0 < \omega < 1$ se lleva a cabo una bajo-relajación. El uso de sobrerelajación se debe utilizar cuando de antemano se sabe que la solución del sistema tiende a converger bajo el método de Gauss-Seidel (cuando $\omega = 1$). La técnica de bajo-relajación es utilizada cuando se sabe que la solución tiende a diverger bajo el método de Gauss-Seidel, en este caso omega sirve como un término disipativo que ayuda a encontrar la convergencia.

Apéndice C

Solución de Sistemas de Bloque Tridiagonal

En el presente proyecto se lleva a cabo la solución del sistema obtenido en la sección anterior mediante el método propuesto por Hoffman [28].

Un sistema de ecuaciones diferenciales parciales es aproximado mediante un sistema de bloque tridiagonal cuando en dicho sistema se involucran tres puntos de la malla en cada nivel. El sistema resultante puede expresarse en su forma general como:

$$S\Delta Q = R \quad (\text{C.1})$$

donde ΔQ y R con vectores con m componentes, y a su vez, S representa el bloque tridiagonal

$$S = \begin{bmatrix} B_2 & C_2 & & & \\ A_3 & B_2 & C_3 & & \\ & A_4 & B_4 & C_4 & \\ & & \ddots & \ddots & \ddots \\ & & & A_{m-2} & B_{m-2} & C_{m-2} \\ & & & & A_{m-1} & B_{m-1} \end{bmatrix}$$

donde A_i, B_i y C_i son matrices de orden m .

Para poder continuar con la obtención de un esquema de solución, habrá que considerar la siguiente factorización

$$S = LU = \begin{bmatrix} \alpha_2 & & & & \\ A_3 & \alpha_3 & & & \\ & A_4 & \alpha_4 & & \\ & & \ddots & \ddots & \\ & & & A_{m-2} & \alpha_{m-2} \\ & & & & A_{m-1} & \alpha_{m-1} \end{bmatrix} \begin{bmatrix} I & \beta_2 & & & \\ & I & \beta_3 & & \\ & & I & \beta_4 & \\ & & & \ddots & \ddots \\ & & & & I & \beta_{m-2} \\ & & & & & I \end{bmatrix} \quad (\text{C.2})$$

donde I es la matriz identidad de orden m y las matrices cuadradas α_i y β_i se determinan mediante

$$\alpha_2 = B_2 \quad y \quad \beta_2 = B_2^{-1}C_2 \quad (\text{C.3})$$

$$\alpha_i = B_i - A_i \beta_{i-1} \quad i = 3, 4, \dots, m-1 \quad (\text{C.4})$$

y

$$\beta_i = \alpha_i^{-1} C_i \quad i = 3, 4, \dots, m-2 \quad (\text{C.5})$$

Ahora, el sistema dado por la ecuación C.1 es equivalente a

$$LY = R \quad (\text{C.6})$$

donde

$$Y = U\Delta Q \quad (\text{C.7})$$

La ecuación C.6 al desarrollarse queda de la siguiente manera

$$\begin{bmatrix} \alpha_2 & & & & \\ A_3 & \alpha_3 & & & \\ & A_4 & \alpha_4 & & \\ & & \ddots & \ddots & \\ & & & A_{m-1} & \alpha_{m-1} \end{bmatrix} \begin{bmatrix} Y_2 \\ Y_3 \\ Y_4 \\ \vdots \\ Y_{m-1} \end{bmatrix} = \begin{bmatrix} R_2 \\ R_3 \\ R_4 \\ \vdots \\ R_{m-1} \end{bmatrix} \quad (\text{C.8})$$

donde

$$Y_2 = \alpha_2^{-1} R_2 \quad (\text{C.9})$$

y

$$Y_i = \alpha_i^{-1} (R_i - A_i Y_{i-1}) \quad i = 3, 4, \dots, m-1 \quad (\text{C.10})$$

La ecuación C.7 se desarrolla como

$$\begin{bmatrix} I & \beta_2 & & & \\ & I & \beta_3 & & \\ & & I & \beta_4 & \\ & & & \ddots & \ddots \\ & & & & I & \beta_{m-2} \\ & & & & & I \end{bmatrix} \begin{bmatrix} \Delta Q_2 \\ \Delta Q_3 \\ \Delta Q_4 \\ \vdots \\ \Delta Q_{m-2} \\ \Delta Q_{m-1} \end{bmatrix} = \begin{bmatrix} Y_2 \\ Y_3 \\ Y_4 \\ \vdots \\ Y_{m-2} \\ Y_{m-1} \end{bmatrix} \quad (\text{C.11})$$

en donde

$$\Delta Q_{m-1} = Y_{m-1} \quad (\text{C.12})$$

y

$$\Delta Q_i = Y_i - \beta_i \Delta Q_{i+1} \quad i = m-1, m-2, \dots, 3, 2 \quad (\text{C.13})$$

Bibliografía

- [1] TJ Chung. *Computational fluid dynamics*. Cambridge university press, segunda edición, 2010.
- [2] Richard H Pletcher, John C Tannehill, y Dale Anderson. *Computational fluid mechanics and heat transfer*. CRC Press, tercera edición, 2012.
- [3] Vladimir D Liseikin. *Grid generation methods*, volumen 1. Springer, 1999.
- [4] Jiri Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, tercera edición, 2015.
- [5] John David Anderson y J Wendt. *Computational fluid dynamics*, volumen 206. Springer, tercera edición, 1995.
- [6] Yunus A Cengel y John M Cimbala. *Mecánica de fluidos: Fundamentos y Aplicaciones*. McGraw Hill Higher Education, segunda edición, 2012.
- [7] CFD Direct. <https://cfd.direct/openfoam/user-guide/introduction/>, 2017.
- [8] Massachusetts Institute of Technology. <http://web.mit.edu/drela/Public/web/xfoil/>, 2001.
- [9] XFLR5. www.xflr5.com/, 2017.
- [10] XFLOW. <https://www.3ds.com/products-services/simulia/products/xflow/>, 2018.
- [11] SU2. <https://su2code.github.io/>, 2018.
- [12] Helyx. <https://engys.com/products/helyx>, 2020.
- [13] Joe F Thompson, Bharat K Soni, y Nigel P Weatherill. *Handbook of grid generation*. 1999. CRC press.
- [14] Joe F Thompson, Zahir UA Warsi, y C Wayne Mastin. *Numerical grid generation: foundations and applications*, volumen 45. North-holland Amsterdam, 1985.
- [15] Vladimir D Liseikin. *Grid generation methods*, volumen 1. Springer, 1999.
- [16] Abhishek Khare, Ashish Singh, y Kishor Nokam. Best practices in grid generation for cfd applications using hypermesh. *Computational Research Laboratories*, página 2, 2009.
- [17] M Farrashkhhalvat y JP Miles. *Basic Structured Grid Generation: With an introduction to unstructured grid generation*. Elsevier, 2003.

- [18] Mato F Siladic. Numerical grid generation and potential airfoil analysis and design. Reporte técnico, AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH, 1988.
- [19] Joe F Thompson, Frank C Thames, y C Wayne Mastin. Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies. *Journal of computational physics*, 15(3):299–319, 1974.
- [20] Joseph L Steger y Denny S Chaussee. Generation of body-fitted coordinates using hyperbolic partial differential equations. *SIAM Journal on Scientific and Statistical Computing*, 1(4):431–437, 1980.
- [21] S Nakamura. Marching grid generation using parabolic partial differential equations. Reporte técnico, OHIO STATE UNIV COLUMBUS DEPT OF MECHANICAL ENGINEERING, 1982.
- [22] Patrick Michael Knupp, CD Ernst, David C Thompson, CJ Stimpson, y Philippe Pierre Pebay. The verdict geometric quality library. Reporte técnico, Sandia National Laboratories, 2006.
- [23] María Virgil Navarro. *Cálculo del flujo potencial compresible alrededor de perfiles aerodinámicos*. 2009.
- [24] Charles Hirsch. Numerical computation of internal and external flows. vol. 2-computational methods for inviscid and viscous flows. *Chichester, England and New York, John Wiley & Sons, 1990, 708 p.*, 1990.
- [25] J.D. Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill Education, 2010.
- [26] Ira H Abbott y Albert E Von Doenhoff. *Theory of wing sections: including a summary of airfoil data*. Courier Corporation, 2012.
- [27] Langley Research Center Turbulence Modeling Resource. <https://turbmodels.larc.nasa.gov/>, 2018.
- [28] Klaus A Hoffmann y Steve T Chiang. Computational fluid dynamics volume 1. *Engineering Education System, Wichita, Kan, USA*, 2000.