Trabajo fin de grado

DISEÑO Y OPTIMIZACIÓN DE ALGORITMO PARA LA GENERACIÓN DE MAPAS DE CASCADAS DE COMPRESOR MEDIANTE CFD

Universitat Politècnica de València

Escuela Técnica Superior de Ingeniería del Diseño
(ETSID)

Grado en Ingeniería Aeroespacial – Curso 2018/2019

Presentado por: **Javier Soriano Lluch**

Tutor UPV: **Dr. Roberto Navarro García**

Valencia, Julio de 2019

Título

Diseño y Optimización de Algoritmo para la Generación de Mapas de Cascadas de Compresor mediante CFD

Resumen

Este trabajo se basa en el desarrollo de un algoritmo integrado en Star CCM+ para el cálculo automatizado de mapas de compresores axiales. Se analizará una etapa de este, simplificándolo a una cascada de álabes para obtener cálculos rápidos y representativos de su comportamiento. El algoritmo cambiará las opciones de configuración numérica de la simulación para tener el menor coste computacional posible, sin sobrepasar un umbral de error para cada punto de funcionamiento. Este estará definido por sus parámetros más representativos.

Palabras Clave

StarCCM+, Numerical Configuration, Off-Design Condition, Automatization

Autor del TFG: **Javier Soriano Lluch**

Valencia, Julio de 2019.

Tutor Académico: **Dr. Roberto Navarro García**

Title

Design and Optimization of an Algorithm for Axiam Compresors Cascade Maps Generation with CFD

Abstract

This work is based on the development of an algorithm integrated in Star CCM + for the calculos of Automated axial compressor maps. It will analyze a single staged compressor. In order to obtaint faster, but also representative, results of its behavior, the mean radius of the compressor is just computed. This way, the problem becomes two dimensional. The algorithm will change the numerical configuration options of the simulation to have the lowest possible computational cost, without exceeding an error threshold defined by its most significant parameters.

Key Words

StarCCM+, Numerical Configuration, Off-Design Conditions, Automatization

Bachelor Thesis Author: **Javier Soriano Lluch**

Valencia, Julio de 2019.

Academic Tutor: **PhD. Roberto Navarro García**

Títol

Disseny i Optimització d'Algoritme per a la Generació de Mapes de Cascades de Compressor mitjançant CFD

Resum

Aquest treball es basa en el desenvolupament d'un algoritme integrat en Star-CCM+ per al càlcul automatitzat de mapes de compressors axials. S'analitzarà una etapa d'aquest, simplificant-ho a una cascada d'àleps per a obtindre càlculs ràpids i representatius del seu comportament. L'algoritme canviarà les opcions de configuració numèrica de la simulació per a tindre el menor cost computacional possible, sense sobrepassar un llindar d'error per a cada punt de funcionament. Aquest estarà definit pels seus paràmetres més representatius.

Paraules Clau

Star-CCM+, Configuració Numèrica, Condicions Fora de Diseny, Automatització

Autor del TFG: **Javier Soriano Lluch**

Valencia, Julio de 2019.

Tutor Acadèmic: **Dr. Roberto Navarro García**

Índice

Índice de figuras

Índice de tablas

Glosario

Notación

C	Cuerda	m
γ	Calado (ángulo)	º
C_x	Cuerda Axial	m
V_a	Velocidad Axial	m/s
H	Altura	m
s	Paso	m
β'	Ángulo perfil	º
φ	Curvatura (ángulo)	º
β	Ángulo flujo	º
ε	Deflexión (ángulo)	º
i	Incidencia (ángulo)	º
δ	Desviación (ángulo)	º
ψ	Coeficiente de carga	–
ϕ	Coeficiente de flujo	–
σ	Solidez	–
R	Grado de reacción	–
$\dot{W}$	Potencia por unidad de longitud	W/m
$\dot{m}$	Gasto másico por unidad de longitud	kg/ms
η	Rendimiento isentrópico	–

Subíndices

LE	Borde de ataque
TE	Borde de fuga
PS	Cara de presión
SS	Cara de succión
0	Aguas arriba rotor
1	Interfaz rotor − estátor
2	Aguas abajo estátor
$*$	Parámetro Corregido

MEMORIA

1. Planteamiento del Proyecto

1.1. Introducción

Una turbina de gas es un motor diseñado para convertir la energía de un fluido en otro tipo de potencia útil. Esta tiene como componentes un generador de gas y un convertidor de energía. Este primero está formado (principalmente) por un compresor, una cámara de combustión y una turbina.

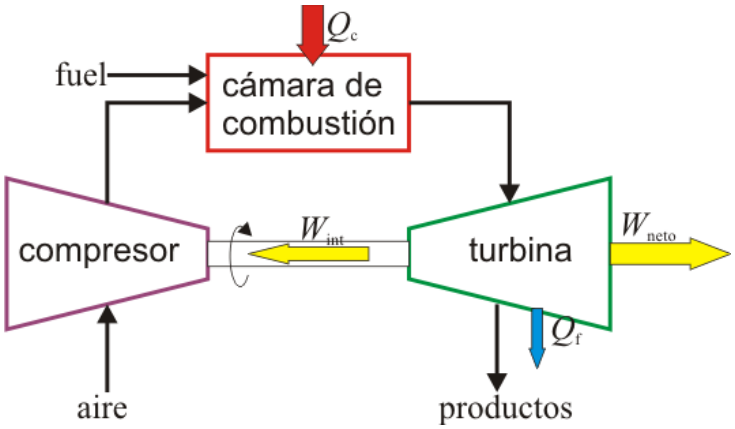


Figura 1: Esquema Turbina de Gas

Dentro del esquema del generador de gas (Figura 1), el compresor tiene la función de aportar energía al flujo antes de entrar a la cámara de combustion. Esta energía que aporta el compresor es extraida del flujo mediante la turbina. Pese a que la teoría y funcionamiento de la turbina de gas fueron establecidos tempranamente[1], se tuvo que esperar al desarrollo de nuevos materiales que pudieses soportar las altas temperaturas y la mejora en la eficiencia de sus componentes para que estas fuesen de utilidad. [3]

Entonces, siendo el principal uso del compresor ser la primera etapa de una turbina de gas, el progreso de las turbinas de gas y compresores han estado mutuamente condicionados (en el caso de compresores axiales, han estado condicionados por el desarrollo de turborreactores).

El propósito de un compresor axial es aumentar la presión total del fluido, es decir, producir un aumento la energía del flujo. Para esto, se necesita la transformación de energía mecánica en energía del flujo, lo cual no es un proceso directo. Esto se realiza mediante la unidad básica dentro de las turbomáquinas térmicas, el escalonamiento. Este está compuesto por una corona de álabes rotativos (rotor) y otra de fijos (estátor), que conforman los canales en los que se modificará la energía cinética del flujo.

Como el propósito del compresor es aumentar la energía del flujo, esto se realiza mediante la acción del rotor. Esta acción acelera el fluido, incrementando así su entalpía de parada. Con el correcto diseño de los álabes, se consigue que además se incremente la presión estática de este.

[1] El ciclo Brayton aparece por primera vez en la patente de John Barber en 1791

$$\frac{dV}{V} = -\frac{1}{1-M^2}\frac{dA}{A} \tag{1}$$

$$\frac{dp}{\rho V^2} = \frac{1}{1-M^2}\frac{dA}{A} \tag{2}$$

Tomando como referencia las ecuaciones de la teoría unidimensional de flujo isentrópico, observamos que para régimen subsónico la sección se debería diseñar tal que la sección sea creciente para conseguir un incremento de presión (Ecuación 2).

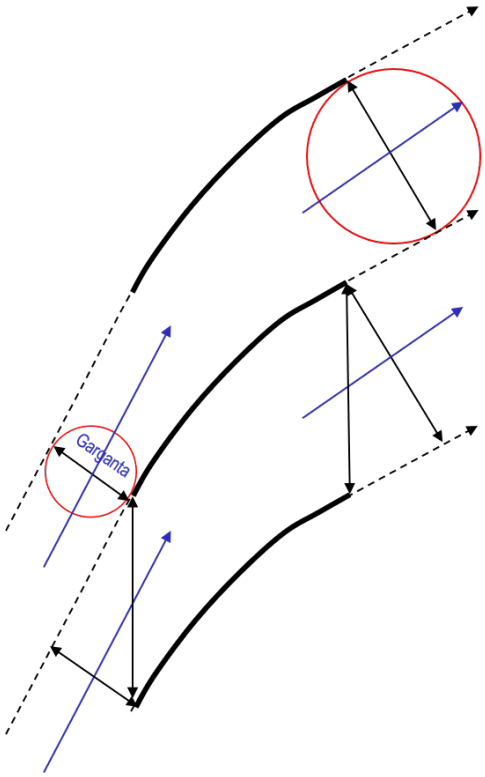


Figura 2: Ejemplo de Distribución de Sección Creciente

En la Figura 2 se puede observar claramente como para conseguir esta ley de sección, el calado tiene un papel crucial. No obstante, la correcta elección del perfil es de vital importancia. Esto se debe a que el estudio aerodinámico de estos ha supuesto una gran mejora en el rendimiento de las turbomáquinas. Principalmente, se ha conseguido una reducción significativa de las pérdidas así como aumentar el rango de funcionamiento estable.
Centrándonos en la Ecuacion 1, observamos que este incremento de la presión producirá una disminución en la velocidad del fluido al pasar por el canal. No obstante, este efecto es sobre la velocidad relativa a la corona. Si nos centramos en la velocidad absoluta del flujo, el incremento en esta será positivo gracias a la cantidad de movimiento aportada por el rotor.

En el estátor, ante la ausencia de aportación de energía al flujo, tenemos que la entalpía será prácticamente constante[2]. Por tanto, en este caso el aumento de presión estática si tendrá condicionado una disminución en la velocidad absoluta del fluido.

Debido a que la máxima difusión posible está limitada por el número de Haller ($\simeq$ 0,7), el incremento en relación de presiones necesaria para el eficiente funcionamiento de los

[2]Siempre que las pérdidas no sean muy importantes

turboreactores no se puede conseguir mediante un solo escalonamiento. Por tanto, los compresores axiales de este tipo de motores están conformados por múltiples etapas.

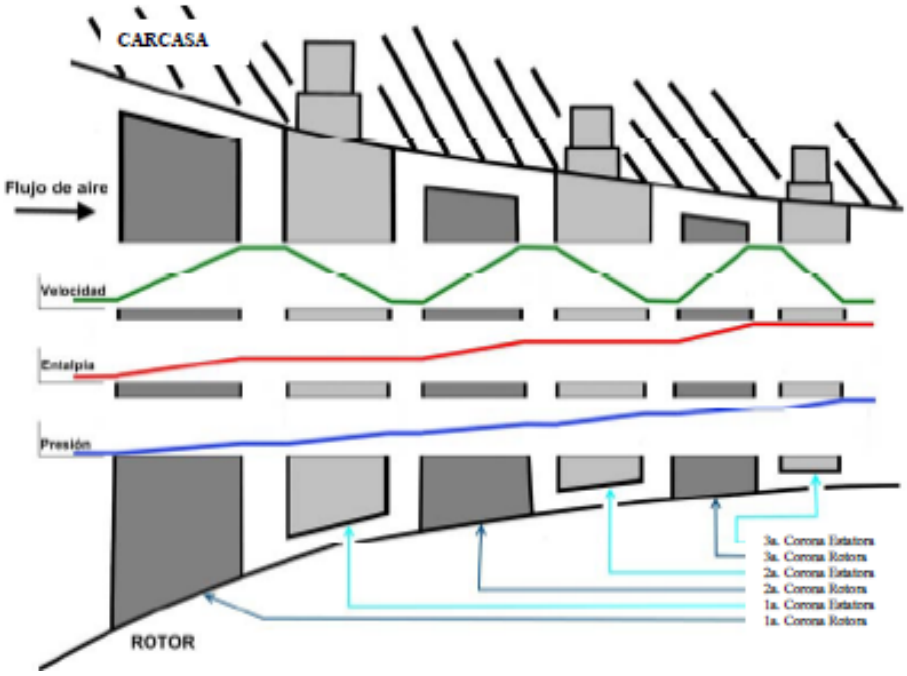


Figura 3: Evolución de la Velocidad, Presión y Entalpía en un Compresor Axial Multietapa

Al aumentar el número de escalonamientos, se consigue aumentar progresivamente la presión y entalpía total del fluido, manteniendo la velocidad axial prácticamente constante (Figura 3).

1.2. Motivación

Una vez establecidos los parámetros de diseño en un compresor, estos optimizan su operación para un punto de funcionamiento. Las condiciones que se han asumido para este cálculo no tienen porque cumplirse, habiendo una incidencia no nula sobre el perfil. Si tenemos un gasto menor al de diseño o un régimen mayor, se producen incidencias positivas sobre el perfil. Esto produce un aumento en el trabajo y relación de presiones, siendo más propenso al desprendimiento. Contrariamente, una incidencia negativa está causada por un gasto mayor al de diseño o un régimen mayor. Esta aumenta la velocidad en la cara de presión, llegando a desprender el flujo en esta cuando es elevado. Se produce una disminución en el trabajo y relación de compresión, siendo más propenso al bloqueo.

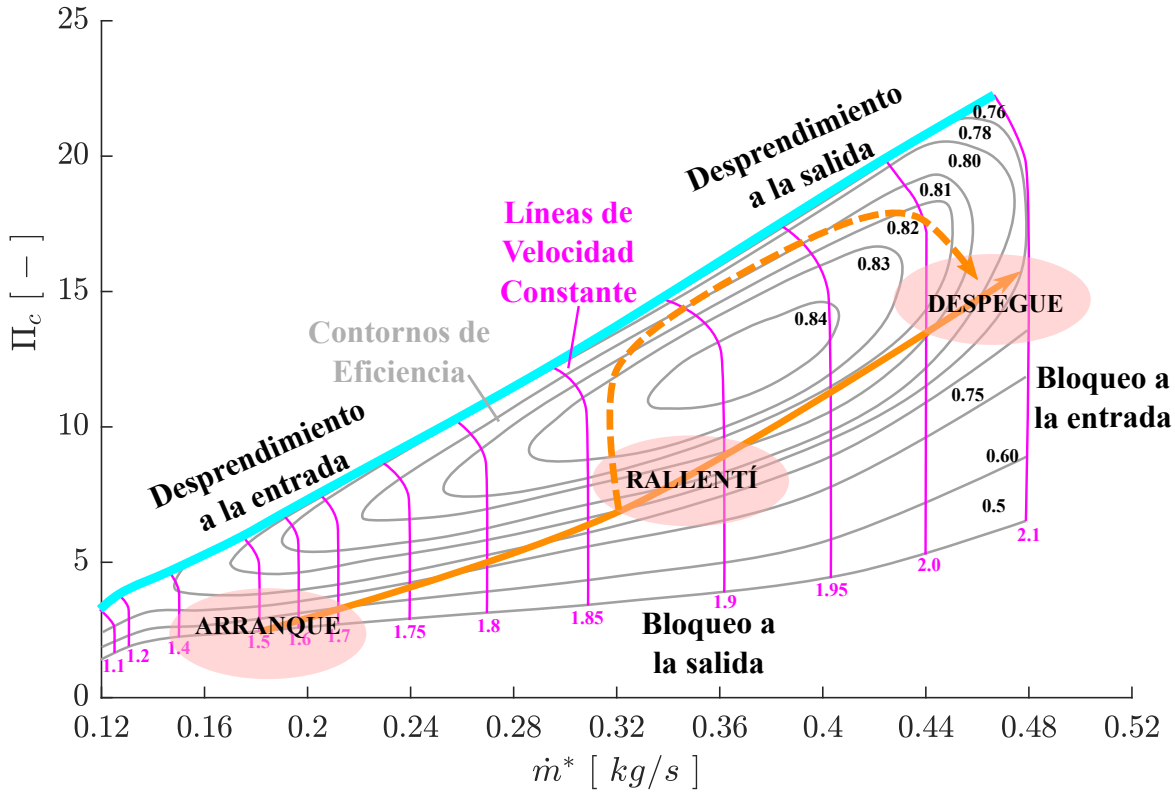


Figura 4: Compresor en las Distintas Fases de Operación

Como se observa en la Figura 4, es necesario que el compresor pueda trabajar fuera de diseño durante su operación normal. Además, una aceleración brusca puede provocar que el punto de funcionamiento se acerque a la línea de bombeo. Teniendo en cuenta que fenómenos como el Lazo de Bombeo y Despredimiento Rotativo afectan a la estabilidad del compresor, pudiendo llegar a ser destructivos para este, es imprescindible conocer en detalle la operación fuera de diseño de este. Además, otros fenómenos como el bloqueo pueden llegar a impedir el arranque del motor.

1.3. Antecedentes

Este proyecto se ha realizado mediante una integración entre el programa STAR-CCM+ y Macros en Java desarrolladas en el entorno NetBeans. STAR-CCM+ es un sofware proporcionado por el DMMT de la Universidad Politécnica de Valencia (UPV), cuyo uso ha incrementado sustancialmente en los últimos años. Actualmente, existe un gran volumen de proyectos en el DMMT. Estos se pueden agrupar en trabajos con empresas así como trabajos de final de grado y máster.

Con el uso de los distintos subentornos disponibles en este programa, se ha creado la geometría de estudio así como el mallado y la resolución de los distintos puntos de funcionamiento involucrados en el trabajo. El entorno NetBeans es un software gratuito que permite la programación de macros para STAR-CCM+ ya que permite incluir sus librerías dentro del entorno de trabajo.

R.Navarro realizó una investigación compleja y detallada en su tesis doctoral sobre el estudio acústico CFD sobre la geometría de un compresor centrífugo.
Evidentemente, los trabajos anteriores realizados en el DMMT condicionarán el punto de

partida de este TFG. En el DMMT se han realizado una serie de trabajos CFD con compresor, pero en su variante centrífuga, por ser la más habitual en vehículos de automoción. Por resumir, las tesis doctorales de R.Navarro y D.Tarí incluyen simulaciones CFD con StarCCM+ de sendos compresores centrífugos, observando la predicción de prestaciones globales, detalles locales del flujo, ruido generado, estabilidad, etc. En compresores axiales no se dispone de este know-how.

Este estudio surge del análisis de cascadas de compresor realizado en el DMMT para la docencia que imparte este. Analizando las características de este tipo de análisis, se dictaminó que el desarrollo de una herramienta que permita obtener estos mapas podría ser interesante en lo que se refiere a labores de prediseño. Esto se debe a que los resultados obtenidos no son suficientemente precisos para una etapa de diseño, pero si que son representativos del problema estudiado así como rápidos.

Este podría tener interés en labores de docencia, ya sea en el Grado de Ingeniería Aeroespacial o su máster habilitante asociado, centrándose en la rama de aeromotores. La rapidez de estos estudios, así como la herramienta desarrollada para la visualización de los resultados pueden ser usados como una herramienta interactiva en la docencia de turbomaquinaria. Así mismo, ciertas condiciones que se han aplicado para aumentar la rapidez del algoritmo sin afectar a sus resultados pudieren ser usadas en la docencia en mecánica de fluidos computacional, concretamente en la correcta selección de modelos dentro de un problema.

1.4. Objetivos

Para poder diseñar mejores compresores, es necesario poder predecir las consecuencias de cualquier cambio de diseño en el funcionamiento de estos. Una fase muy importante dentro de las etapas de desarrollo es el prediseño. En esta etapa se tiene un producto cuyo diseño está poco refinado (o no hay diseño alguno) y se busca plantear las modificaciones que mejoren el funcionamiento de este (restringiendo así ciertos parámetros de su diseño). Para esto, es necesario obtener información representativa de la turbomáquina al mismo tiempo que esta se está desarrollando, para así poder iterar su diseño de acorde a los resultados obtenidos. Estas podrían obtenerse mediante resultados experimentales, no obstante, el coste y tiempo de preparación asociado a estos podría resultar prohibitivo ya que se necesitaría experimentar con muchos prototipos para pasar a la siguiente fase de desarrollo.

En los últimos años, el gran avance en simulaciones numéricas de problemas físicos han permitido sustituir parte del ensayo experimental por cálculos realizados con este tipo de herramientas. No obstante, la resolución detallada del problema puede no ser viable en una fase de prediseño para el estudio de turbomaquinária. Teniendo en cuenta los altos costes computacionales (y económicos) que el análisis en detalle de estos productos puede acarrear, se aumentaría tanto el coste temporal, computacional y económico del proceso de diseño.

Teniendo en cuenta esto, este trabajo tiene por objetivo el desarrollo de un algoritmo integrado en Star CCM+ para el cálculo automatizado de mapas de compresor axial. Este debe ser capaz de obtener soluciones rápidas y representativas del problema a resolver, pudiendo investigar un amplio abanico de configuraciones sin que esto suponga una gran carga de trabajo para el operario. Por esto, se concluye que las características y/o requerimientos que este debe presentar son:

- Rapidez
- Eficiencia
- Bajo Coste Computacional
- Obtener Resultados Representativos
- Estabilidad
- Modularidad
- Automaticidad

1.5. Estructura del Trabajo

El trabajo que aquí se presenta está conformado por cuatro documentos diferenciados: memoria, presupuesto, pliego de condiciones y anexos

- **Memoria:**
  - **Planteamiento del Proyecto:** En este capítulo se expondrá el contexto general mediante una introducción. Así mismo, también se incluirán la motivación, antecendentes y objetivos que tendrá este proyecto.
  - **Modelado CFD de Escalonamiento de Compresor:** En este capítulo se incluirá toda la información relativa a las simulaciones realizadas para calcular los diferentes puntos de funcionamiento.
  - **Desarrollo del Algoritmo de Mapeado:** En este capítulo, se expondrá ordenadamente el proceso de desarrollo del algoritmo de mapeado de mapas de compresores axiales.
  - **Resultados y Discusión:** En este capítulo, se expondrán los resultados obtenidos mediante la ejecución de la macro.
  - **Comentarios Finales:** En este capítulo, se resumirá el análisis del funcionamiento del algoritmo así como sus limitaciones. Como extensión, se incluirán posibles extensiones que puedan desarrollarse a partir del algoritmo desarrollado.
- **Presupuesto:** este documento recogerá los costes de inversión que habría supuesto la realización de este proyecto. Se expondran tanto los presupuestos parciales como globales, indicando los precios unitarios en los primeros.
- **Pliego de Condiciones:** este documento recogerá las especificaciones de los equipos y material necesarios durante el desarrollo de este trabajo, con especial interés en la normativa de prevención de riesgos laborales vigente.

2. Modelado CFD de Escalonamiento de Compresor

2.1. Introducción al CFD

Dentro de la mecánica de fluidos, tenemos como pilar fundamental a las ecuaciones de Navier-Stokes. Estas, conjuntamente con las ecuaciones adicionales que permiten el cierre del problema, son un conjunto de 6 ecuaciones no lineales acopladas entre ellas, dependientes del tiempo. Es por esto que, en la mayoría de casos, no es posible obtener soluciones analíticas a un problema dado. Por esta razón, se desarrollo la mecánica de fluidos computacional (CFD). Mediante la división del problema en múltiples volúmenes de control, permite obtener soluciones numéricas a estas ecuaciones (de forma discreta).

Aunque con el aumento progresivo en la velocidad de cálculo de los ordenadores en los últimos años se ha permitido el acceso a este tipo de herramientas en múltiples campos, el gran coste computacional que presentan algunos problemas para obtener soluciones en detalle resulta prohibitivo (ya sea monetaria o computacionalmente). No obstante, es una herramienta que presenta una gran variedad de ventajas. Centrándonos en el campo de la turbomaquinaria, tenemos:

- Es una buena herramienta para prediseño y diseño, ya que permite obtener resultados sobre el producto que se está desarrollando (sin tener que construir este). Esto reduce significativamente los tiempos de prototipado.
- Respecto a los resultados experimentales, permite obtener cualquier solución en cualquier punto del dominio de estudio sin perturbar al flujo.
- Pese a tener unos costes relativamente altos de licencias y hardware, con la correcta combinación entre métodos analíticos, experimentales y numéricos, se consigue reducir el coste global del proceso.

Dentro del proceso de realizar una simulación CFD, se pueden separar las siguientes partes:

- **Pre-proceso:** En esta primera etapa, se "construye" el problema a resolver. A partir de la geometría del problema, se genera la malla para esta así como los modelos físicos y propiedades del fluido que mejor se adapten al caso de estudio. Como último paso, se aplican las condiciones iniciales (si son necesarias) y de contorno que mejor se adapten al problema físico.
- **Solver:** En esta etapa se seleccionan el método o métodos para su resolución, pudiendo optar entre lanzar cálculos en serie o paralelo o el método para resolver el conjunto de ecuaciones resultantes de la discretización del problema.
- **Post-proceso:** Una vez resuelto el sistema de ecuaciones, se han obtenido una gran cantidad de datos asociados a este. Esta tapa tiene como finalidad la correcta interpretación y visualización de estos resultados.

Este no es un proceso directo y lineal, ya que tanto el pre-proceso como el solver deben estar realimentados por la información obtenida en el post-proceso. Esto conforma un proceso iterativo que se puede observar claramente en la Figura 5.

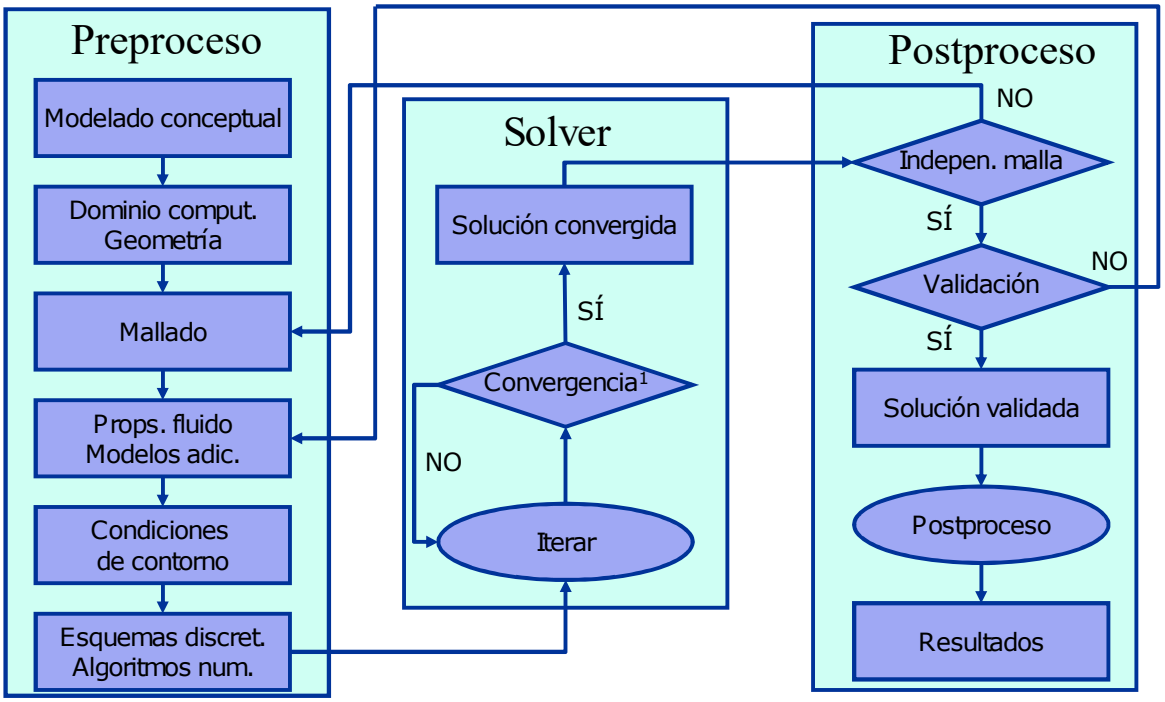


Figura 5: Proceso Iterativo de Resolución de un Caso CFD

2.2. Geometría y Dominio

Dado que la finalidad principal del algoritmo es servir para la etapa de prediseño de un compresor axial, no sería posible analizar turbomáquina completa. El estudio completo de esta comportaría un coste computacional y temporal (y económico) tan grande que no sería realizable o, si fuera realizable con los recursos disponibles, el coste temporal sería tan elevado que no sería posible usar esta herramienta para prediseño. Es por esto que se realiza un compromiso entre obtener resultados precisos y tener un bajo coste computacional mediante el análisis de un canal de una única etapa de compresor. Con esta restricción, no sería posible calcular puntos de alto desprendimiento porque estamos forzando a que la solución en todos los canales sea la misma. En el caso del desprendimiento rotativo, este directamente no se podría calcular. No obstante, mediante soluciones estacionarias, si sería posible encontrar aquellas zonas a partir de las cuales el desprendimiento sea significativo y pudiera conducir a efectos transitorios no deseados. Dado que en prediseño es interesante conocer donde están estos para poder realizar un análisis más detallado, sería posible estimar a partir de que punto de funcionamiento pueden situarse estos.

Con estas simplificaciones, todavía sería demasiado costoso realizar el mapeado completo del compresor. Aunque este tuviera sentido computacionalmente y temporalmente hablando, sería menos costoso y preciso construir un prototipo y probar este. Por tanto, reducimos el coste computacional resolviendo solo el radio medio del compresor. Con esto estamos despreciando efectos tridimensionales que pueden afectar al comportamiento del compresor. En el caso de este algoritmo, al centrarse en la obtención de parámetros globales, estos podrían ser corregidos mediante correlaciones empíricas.

Siendo este un caso 2D, por el bajo coste computacional asociados a estos, podría realizarse un análisis de todos los canales. Este no se va a desarrollar en el estudio que aquí se presenta debido al coste temporal que estos cálculos supondrían con los medios disponibles.

No obstante, el algoritmo desarrollado es capaz de trabajar con todos los canales de la geometría.

En la Figura 6 se puede ver la geometría que se va a analizar. Esta se ha creado en el mismo editor CAD de Star CCM+, para así evitar problemas de compatibilidad o preprocesar la geometría.

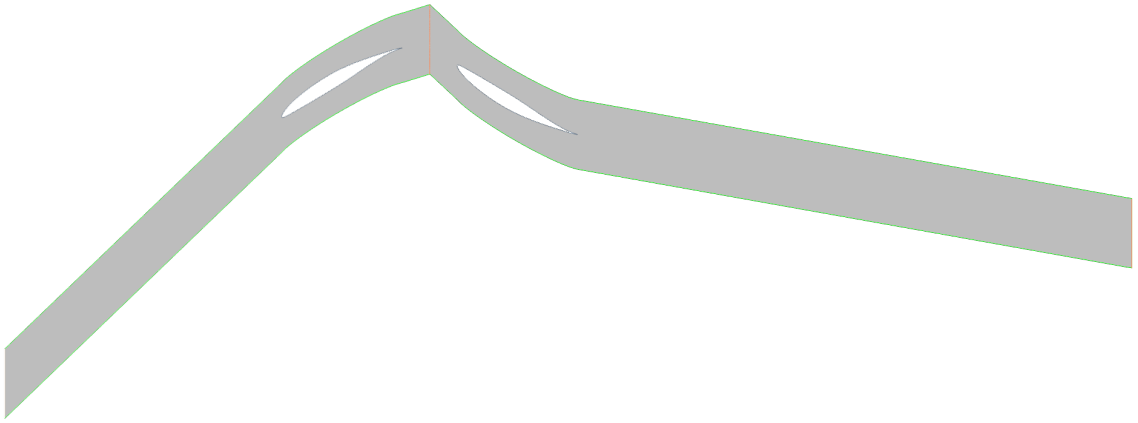


Figura 6: Geometría del Canal Analizado

Esta se corresponde con la superfície 2D que resultaría al desplegar el cilindro que se corresponde con el radio medio del compresor axial. Para generar esta geometría, se ha optado por generar el estátor y rotor como partes diferentes. Como ambos cuerpos se generan con el mismo procedimiento, vamos a proceder a explicar solo el rotor.

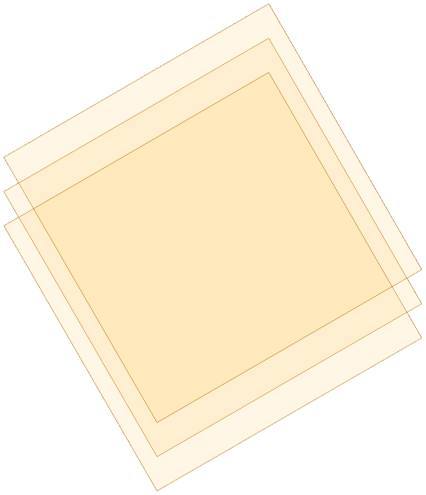


Figura 7: Planos Creados a partir de Transformaciones sobre el Plano XY

Para tener un diseño donde tanto la solidez, calado y distancia aguas arriba y aguas abajo del perfil sea paramétrico, se han generado 3 planos adicionales (ver Figura 7). Estos tienen todos el mismo ángulo ($alpha$), habiéndose desplazado el superior e inferior una distancia $d = paso/2$. De esta forma, modificando $alpha$ y $paso$, somos capaces de modificar tanto el calado como el paso del canal.

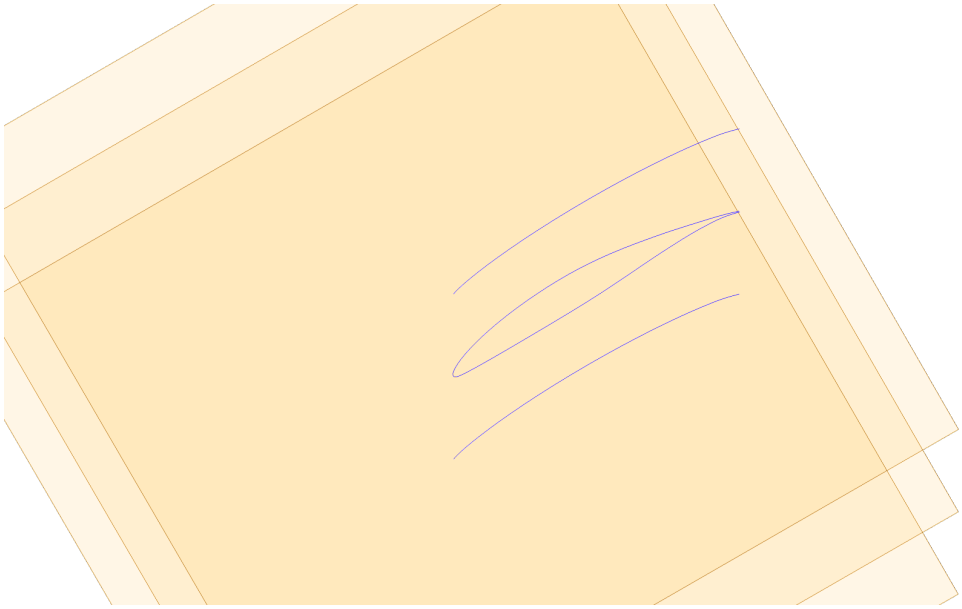


Figura 8: Perfil y Líneas Medias Importadas en Planos

Una vez tenemos los planos creados y posicionados, debemos importar la línea media en los planos superior e inferior y el perfil en el medio (Figura 8). Es importante destacar que el borde de ataque tanto de perfil como de línea media tiene que estar en la coordenada (0, 0, 0).

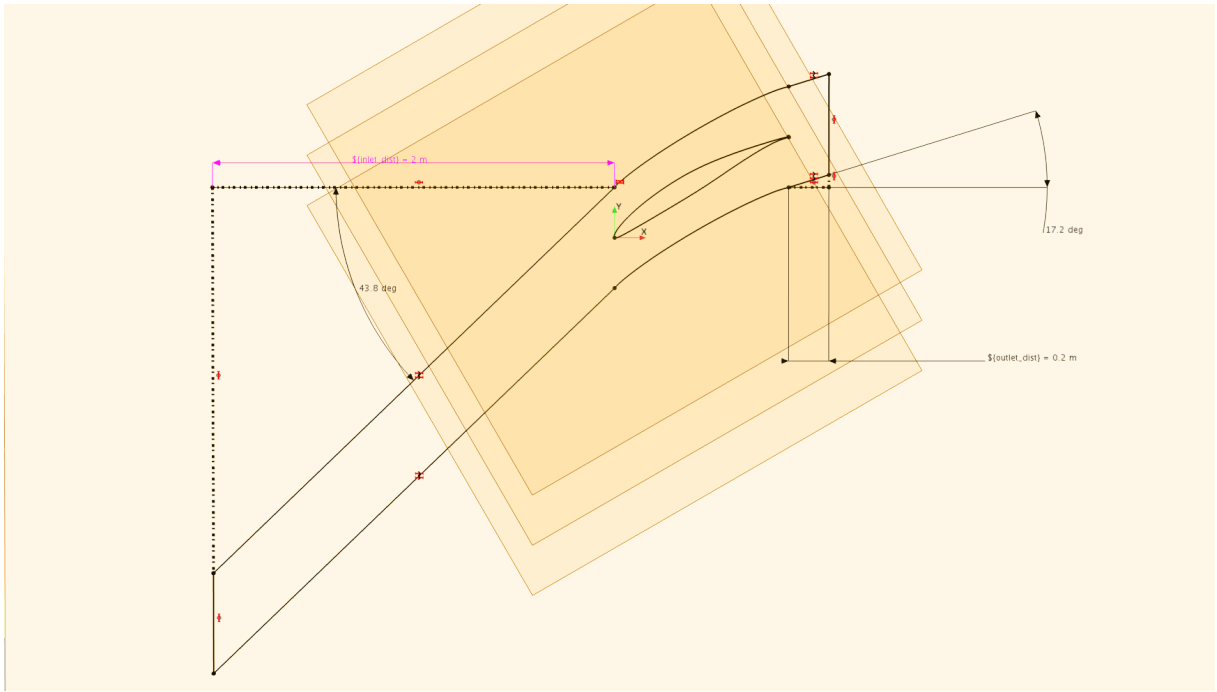


Figura 9: Skecth de Geometría Creado

Cuando ya tenemos tanto las líneas medias como perfil importados, debemos crear un nuevo sketch en el plano XY y proyectar estos. Después de esto, cerramos la zona aguas arriba y aguas abajo del perfil con líneas rectas (ver Figura 9). Es imprescindible que la línea periodica inferior y superior sean paralelas y el inlet y outlet sean líneas verticales. A estas se les da como ángulo la incidencia que tendría el perfil en diseño. Resulta interesante definir la distancia aguas arriba y aguas abajo mediante parámetros para así tener un diseño totalmente parametrizado.

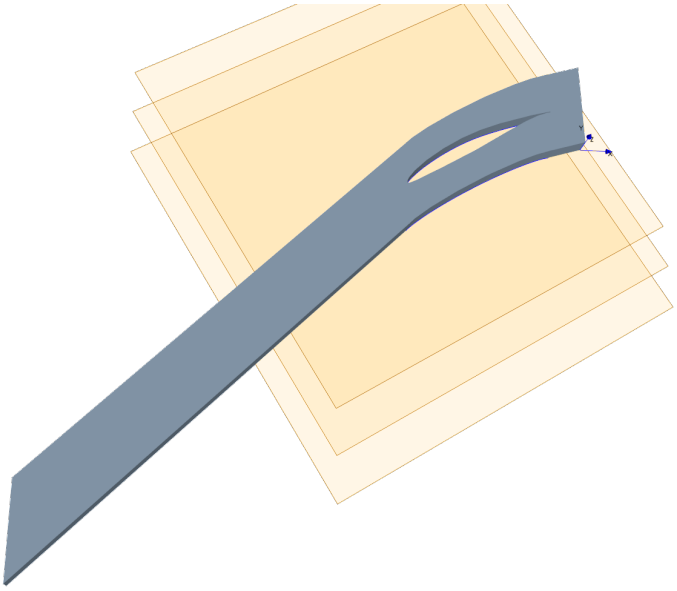


Figura 10: Extrusión de la Geometría y Sistema de Coordenadas

Al estar el Sketch de la geometría ya creado, debemos extruir este y nombrar adecuadamente sus caras.

- Entrada: inlet
- Salida: outlet
- Superficie superior: periodic_1
- Superficie inferior: periodic_2
- Perfil: airfoil
- Plano $z = 0$: domain

Siguiendo este mismo proceso para la generación del estátor, obtenemos la siguiente geometría (Figura 11).

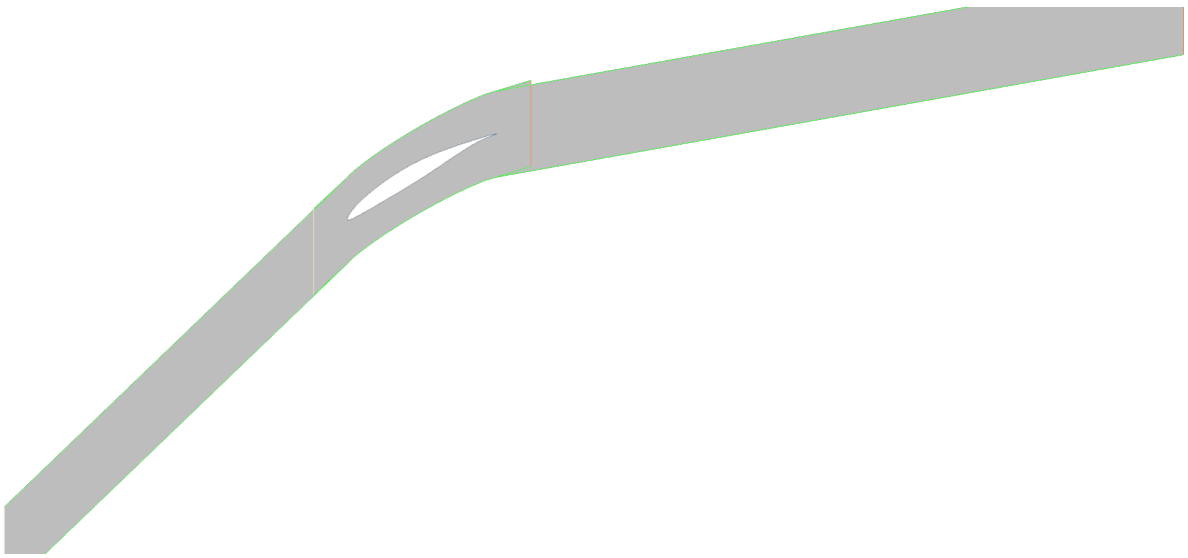


Figura 11: Geometría antes de Transformaciones

Se observa claramente como están superpuestas ambas partes y el estátor está reflejado en el eje X respecto a la Figura 6. Para solucionar esto, debemos definir las siguientes operaciones:

1. Badge for 2D Meshing (Rotor y Estátor)
2. Transform (Rotor)
    a) Coordinate Transform: rotor_outlet → Laboratory
3. Transform (Estátor)
    a) Reflect
    b) Coordinate Transform: stator_inlet → Laboratory

De esta forma, si hemos colocado y exportado correctamente los sistemas de coordenadas, obtenemos una geometría como la que se observa en la Figura 6. Llegados a este punto, a partir del archivo ya creado, solo debemos asignar las superficies nuestras "Parts" (Rotor y Estátor) a las regiones físicas ya creadas (Figura 12). Esto nos asegura que no hay problemas de interacción entre la Macro y la simulación.

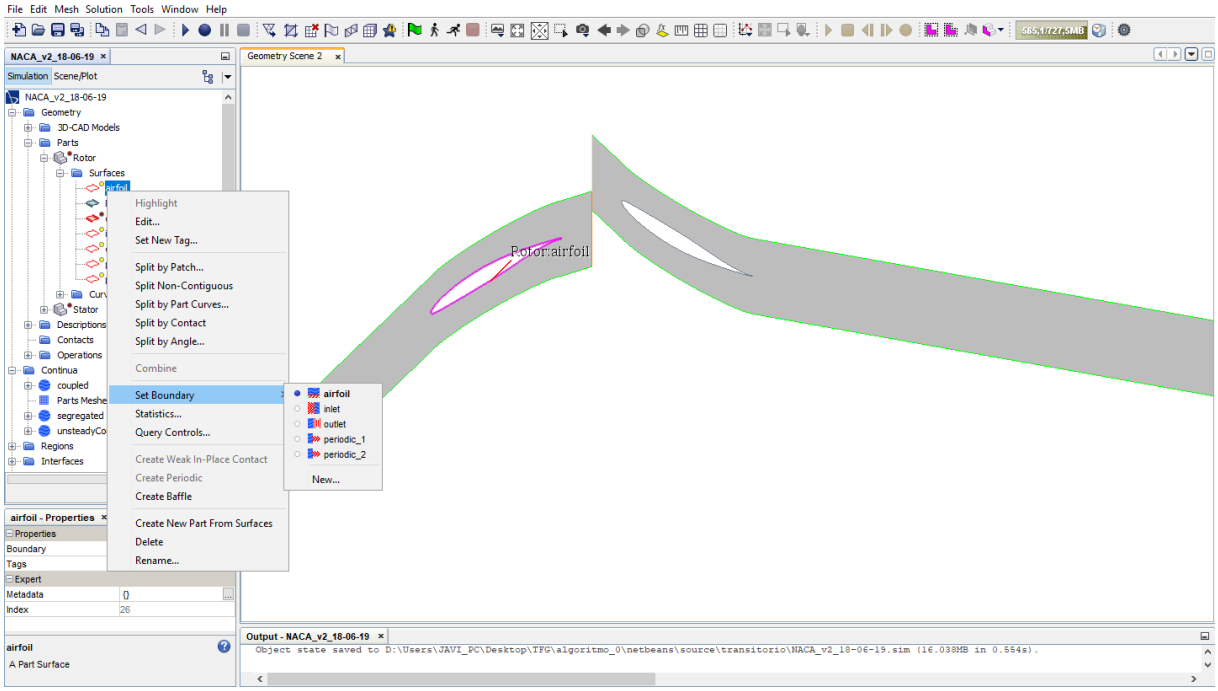


Figura 12: Asignar Superficies a Condiciones de Contorno ya Creadas

2.3. Mallado

El mallado es uno de los aspectos más importantes dentro de la configuración de un caso CFD. Esto se debe a que esta determina:

- **Coste computacional:** Según el número de elementos usados, el tiempo de cálculo necesario aumentará de forma significativa. Por tanto, es imprescindible escoger una malla adecuada al problema estudiado, intentando mantener el número de elementos al mínimo posible (sin que esto afecte a los resultados).
- **Precisión:** La calidad del mallado es muy relevante de cara a los resultados que vayamos a obtener. Una malla inadecuada puede causar que, pese a tener un número alto de elementos, el resultado difiera del problema real.

- **Tasa de convergencia:** Así como en la precisión, la calidad del mallado afecta directamente a la tasa de convergencia del problema a resolver, pudiendo llegar a imposibilitar la convergencia de este.

Dentro de los tipos de malla, nos encontramos que puede ser de dos tipos: estructurada o no estructurada. En esta primera, nos encontramos que los elementos están identificados por índices (un índice por cada dimensión del problema). Esta presenta numerosas ventajas, ya que su discretización es más sencilla y sus cálculos más precisos si los elementos están alineados con las líneas de corriente. No obstante, este tipo de mallado requiere un gran esfuerzo por parte del usuario, llegando a ser difícil (o incluso imposible) de realizar en el caso de geometrías complejas. Por otro lado tenemos el mallado no estructurado, el cual se genera mediante un algoritmo automático de mallado dentro del mismo programa. Este permite usar cualquier tipo de elementos y geometrías complejas. No obstante, se debe prestar especial atención a que la malla sea de buena calidad.

Dentro de la calidad de malla, tenemos varios factores que son de especial importancia:

- **Densidad de Malla:** Si tenemos una zona con altos gradientes, se debe aumentar la densidad de mallado en esta para obtener un y^+ adecuado.
- **Ortogonalidad:** Si bien un mallado estructurado nos permite usar elementos no regulares, esto incrementará el error de discretización.
- **Transición suave:** Debemos mantener la diferencia en tamaño entre elementos contiguos por debajo del 20 %.
- **Relación de Aspecto:** Se debe tener en cuenta especialmente si los elementos no están orientados en la dirección del flujo.

Considerando los conceptos mencionados hasta el momento, pasamos a realizar el mallado de nuestra geometría. Este se realiza en Star CCM+, mediante la herramienta de mallado que el programa proporciona. Dentro de las opciones que este presenta, para un mallado 2D la más adecuada es el "Automatic (2D) Mesher". Esta es una herramienta que permite elegir entre mallados de tipo "polygonal", "quadrilateral" y "triangular", además de poder incluir un mallado prismático.

Comparando entre los tres modelos base para el mallado, se ha descartado el "quadrilateral". Esto se debe a que, teniendo en cuenta que en el dominio tenemos superficies curvas tanto en el perfil como en el contorno periódico, este no es el más adecuado ya que no se adapta tan bien a los contornos curvos. Evaluando las ventajas y desventajas que ofrecen los modelos restantes, se ha optado por usar la malla "polygonal".

Para obtener una mejor resolución en la región cercana al perfil, se ha usado adicionalmente el mallado prismático. La razón principal es poder tener una buena resolución en esta zona, ya que está sometida a altos gradientes, así como reducir el número de elementos totales del problema (Figura 13).

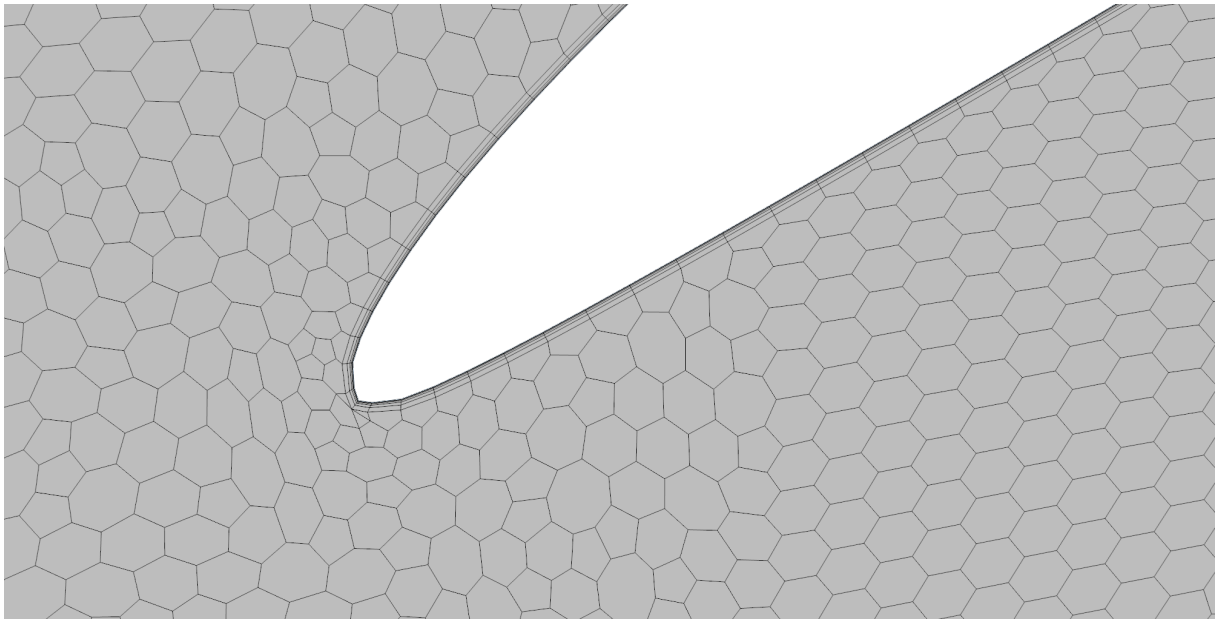


Figura 13: Malla Alrededor del Perfil

No obstante, usando solo estos parámetros de mallado el número de elementos usados es relativamente alto ya que para capturar los efectos que se corresponden con el desprendimiento del flujo deberíamos tener un tamaño de elemento relativamente pequeño. Es por esto que ha resultado conveniente realizar varios refinamientos locales en el dominio.

Figura 14: Malla con Refinamientos Locales

En la Figura 14 se puede observar como se han realizado estos refinamientos locales. Se ha optado por realizar un primer refinamiento del 50 % del tamaño base. Este está destinado a capturar correctamente el flujo desprendido resultante de tener una cierta incidencia sobre el perfil, así como efectos de compresibilidad que puedan aparecer en el canal. Adicionalmente, se ha incorporado un segundo refinamiento del 25 % del tamaño base. Este se destina a tener una mejor resolución de la estela del perfil así como contemplar el engrosamiento de la capa límite (sobretodo en la cara de succión).

Para evitar tener las paredes de las condiciones de contorno (salvo el perfil tanto de rotor como de estátor), se ha aplicado un control volumétrico en todo el dominio para que el tamaño de elemento fuera lo más similar posible al tamaño base definido.

2.4. Configuración de los Casos

2.4.1. Ecuaciones de Conservación y Discretización

La mecánica de fluidos está regida por las ecuaciones de transporte de masa, cantidad de movimiento. Estas se derivan a partir del Teorema de Transporte de Reynolds y componen de un conjunto de 5 ecuaciones de transporte, que están completadas por la ecuación de estado del fluido estudiado. Aunque estas se pueden plantear de forma integral o diferencial, la primera es la forma adoptada en la mecánica de fluidos computacional, porque al dividir el problema en volúmenes de control esta es más sencilla de aplicar para cada uno de estos. Describimos a continuación estas ecuaciones:

- **Conservación de la Masa:** Es una ecuación escalar que establece que la masa dentro de un volumen de control es función de la masa que sale o entra a través de sus límites.

$$\frac{d}{dt}\int_{V_c}\rho dV + \int_{S_c}\rho\cdot(\vec{u}-\vec{u}_c)\cdot\vec{n}dS = 0 \tag{3}$$

- **Conservación de la Cantidad de Movimiento:** Es el resultado de aplicar la segunda ley de newton sobre un volumen de control, donde tenemos que las fuerzas asociadas son las debidas a presión, cortante, fuerzas másicas y de reacción (si las hay). Es una ecuación vectorial de 3 elementos (x, y, z).

$$\sum\vec{F}_R + \int_{S_c}\tau\cdot\vec{n}dS + \int_{V_c}\rho\vec{f}_m dV = \frac{d}{dt}\int_{V_c}\rho\vec{u}dV + \int_{S_c}\rho\vec{u}\,(\vec{u}-\vec{u}_c)\cdot\vec{n}dS \tag{4}$$

- **Conservación de la Energía:** Tiene un sentido similar a la ecuación de conservación de la masa, aplicándolo a la energía con la primera ley de la termodinámica.

$$\begin{aligned}&\frac{d}{dt}\int_{V_c}\rho\left(e+\frac{1}{2}u^2\right)dV + \int_{S_c}\rho\left(e+\frac{1}{2}u^2\right)(\vec{u}-\vec{u}_c)\cdot\vec{n}dS = \\ =&-\int_{S_c}\vec{n}\cdot p\vec{u}dS + \int_{S_c}\vec{n}\cdot\tau'\cdot\vec{u}dS + \int_{V_c}\rho\vec{f}_m\cdot\vec{u}dV - \int_{S_c}\vec{q}\cdot\vec{n}dS + \int_{V_c}QdV\end{aligned} \tag{5}$$

De esta forma, al añadir a estas cinco ecuaciones la ecuación de estado del fluido estudiado, cerraríamos el problema al tener 6 ecuaciones y 6 incógnitas (ρ, u, v, w, T y p).

Entonces, dentro del caso de estudio, se aplicarían las condiciones de contorno adecuadas a las caras de cada elemento, considerando las condiciones de contorno externas del problema y condiciones iniciales. Una vez discretizado el caso, el problema pasará a ser un conjunto de ecuaciones algebraicas no lineales que se resolveran de manera iterativa.

No obstante, la resolución directa y completa del sistema de ecuaciones completo es muy costoso computacionalmente debido a la turbulencia. Dado que la mayoría de los fluidos de estudio en el campo de la ingeniería poseen una naturaleza turbulenta en gran parte de la situaciones de estudio, esta se debe modelar.

2.4.2. Modelos de Turbulencia

Un flujo turbulento se define como un flujo cuyas propiedades presentan fluctuaciones irregulares y caóticas. Este está caracterizado por una alta difusividad y disipación, así como tridimensional, transitorio y rotacional.

Actualmente, hay varios métodos para tratar la turbulencia. Estos se basan en las escalas turbulentas que se desean resolver, aumentando el coste computacional y precisión de los resultados cuantas más se resuelvan. Dentro de estos, existen tres modelos principales en el cálculo de flujos turbulentos:

- **DNS (Direct Numerical Simulation):** Con este método no se modela ninguna escala de turbulencia, sinó que se calculan todos hasta la escala de Komogorov. Este método tiene la ventaja de obtener resultados muy precisos pero, teniendo en cuenta su alto coste computacional, este sólo se ha usado en problemas con geometrías simples y habitualmente a un número de Reynolds Bajo. [3]
- **LES (Large Eddy Simulation):** Es una solución compromiso entre resolver todas las escalas de turbulencia y tener mucha precisión en los resultado, o modelar estas. Con esta técnica, se modelan las escalas más pequeñas y se resuelven las de mayor tamaño.
- **RANS (Reynolds Averaged Navier-Stokes):** La principal característica de este método es la resolución de las ecuaciones promedio del flujo. Esto permite reducir significativamente el coste computacional, convirtiéndose en una herramienta muy útil dentro de la rama de la ingeniería. El problema que presenta este tipo de resolución es la aparición de una nueva variable a resolver, el tensor de esfuerzos de Reynolds. Es por esto que, para solucionar este problema de cierre, se incluyen ecuaciones de transporte adicionales. Según las ecuaciones adicionales, aparecen varios modelos de turbulencia: $k-\epsilon$, $k-\omega$, $k-\omega\ SST$ $(Shear\ Stress\ Transport)$ y $Spalart-Allmaras$.

En el proyecto que en este informe se detalla, se ha optado por seleccionar el modelo $k-\omega\ SST$. Las razones de elegir este han sido su habilidad de predecir la separación del flujo y readherencia mejor que el $k-\epsilon$ y $k-\omega$, así como que este es un modelo ampliamente usado y validado en el campo de la turbomaquinaria.

2.4.3. Condiciones de Contorno

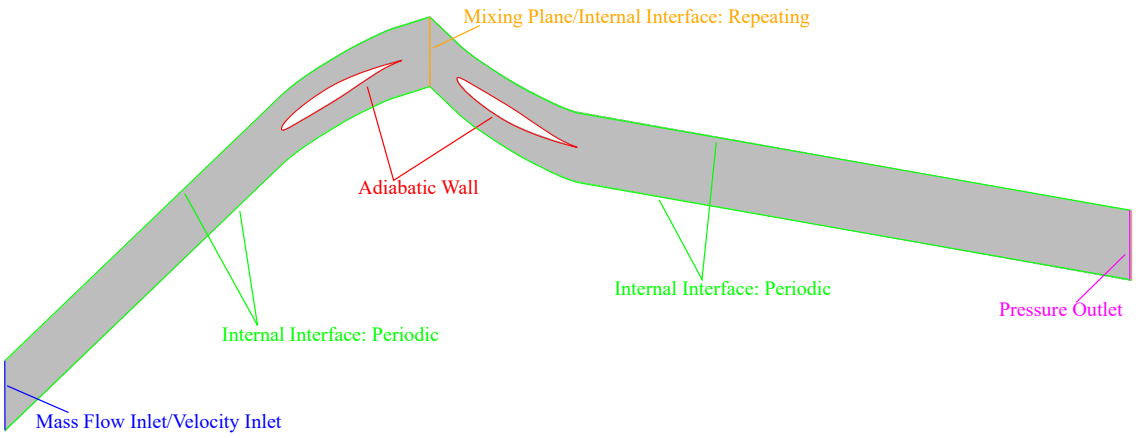


Figura 15: Condiciones de Contorno Aplicadas

En la Figura 15 podemos observar las distintas condiciones de contorno que se han aplicado en el dominio. Las separadas entre "/" indican que se puede elegir una u otra, según convenga.

[3] Como referencia, un "Backward-Facing Step" necesitó 54 días de cálculo en un superordenador con velocidad de procesamiento de 4 GigaFlops [2]

Mientras que la elección entre “Mass Flow Inlet” y “Velocity Inlet” está más condicionada por la preferencia del usuario, la elección entre “Mixing Plane” y “Internal Interface: Repeating” está condicionada por la elección entre una resolución estacionaria o transitoria. En el caso de una resolución estacionaria, es recomendable usar el Mixing Plane. Esta es una condición de contorno que promedia la solución en todo el contorno. Pese a que este tipo de CC elimine algunos efectos causados por la estela del rotor sobre el estátor, también elimina el efecto Clocking, obteniendo así unas variables promedio del problema. Por otro lado, la condición de contorno “Internal Interface: Repeating” es una combinación entre una “Internal Interface: Periodic” y “Internal Interface: In Place”. Esta usa la segunda en aquellas zonas que están en contacto y la primera en las zonas que no (Figura 16).

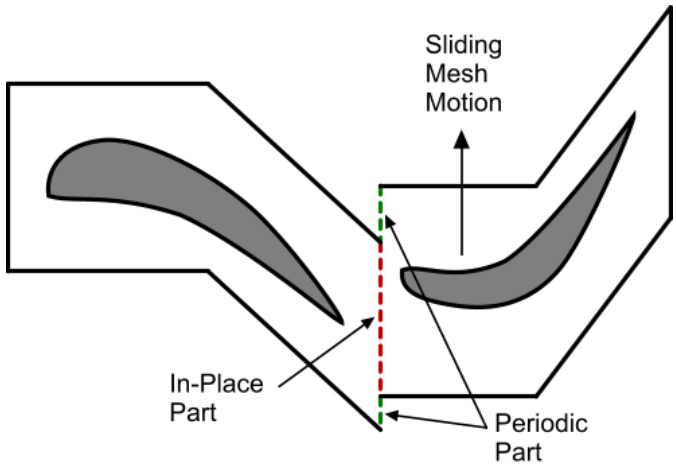


Figura 16: Condiciones de Contorno “Repeating”

Por tanto, si usáramos este tipo de CC en un resultado estacionario la posición relativa entre el rotor y el estátor afectaría a la solución, ya que la estela del primero afectaría de forma diferente al segundo. No obstante, este tipo de interfaz es la idónea en una simulación transitoria ya que en este tipo de simulaciones el rotor se estaría desplazando a cada paso temporal, obteniendo así la interacción rotor-estátor.

2.4.4. Modelos Físicos y Solvers

Dentro de las simplificaciones y modelos físicos disponibles se han elegido los siguientes:

- Espacio bidimensional (2D)
- Estacionario/Transitorio: en una primera estimación, resulta interesante simplificar el problema a uno estacionario ya que proporciona resultados bastante precisos a un bajo coste computacional. No obstante, cuando la incidencia sobre los perfiles es alta, esta suposición no sería correcta y se debe ejecutar una simulación transitoria. En nuestro caso, se ha usado una implícita por las ventajas que presenta sobre la explícita.
- Régimen Turbulento: RANS $k-\omega$ SST
- Segregated/Coupled Flow: en una primera estimación, el solver segregado tiene numerosos beneficios como su bajo coste computacional, convergencia rápida y estabilidad. No obstante, cuando nos encontramos con incidencias altas y/o altos números de mach, los resultados que este proporciona empiezan a diferir con la realidad y nos vemos forzados a usar el solver acoplado, el cual es computacionalmente más costoso.
- El fluido de trabajo seleccionado ha sido un gas ideal, en concreto, aire.

2.5. Criterios de Convergencia

Al tratar con métodos iterativos para la resolución de las ecuaciones obtenidas, nunca vamos a llegar a la solución "exacta" del sistema sino que el error de la solución irá disminuyendo hasta que sea despreciable. Es por esto que se deben considerar ciertos criterios a partir de los cuales el error en la resolución de las ecuaciones así como la variación de las variables del problema en cada iteración sea despreciable, evitanto incurrir en tiempos de cálculo mayores por una variación mínima en nuestra solución. Es por esto que se usan las siguientes variables como referencias:

- **Residuales:** En el caso ideal de una resolución exacta, la parte derecha de una ecuación debería ser idéntica a la izquierda. No obstante, con los métodos iterativos trabajan minimizando este error entre ambas partes. Cuando estos están por debajo de un cierto valor (10^-4 se suele considerar un número aceptable), se considera que este criterio estaría cumplido. No obstante, es importante mencionar que estos están adimensionalizados habitualmente con el resultado de la primera iteración, por lo que el valor a partir del cual son considerados aceptables varía según sea el estado de la solución antes de empezar el cálculo.
- **Variables de Interés:** Más que los residuales, el interés principal de una simulación es que la variables de interés para el estudio lleguen a un cierto nivel de convergencia, es decir, que el valor de estas no varíe por encima de cierto umbral a medida que avanza el cálculo. Es por esto que se han considerado como importantes las siguientes variables del sistema:

  - Eficiencia Isentrópica
  - Fuerza sobre el Rotor
  - Mach Máximo

  Se ha definido como criterio sobre estas que su variación no sea superior al 0.5 % durante 200 iteraciones. Este sistema es tan restrictivo debido a que se ha comprobado que el solver acoplado tenía tendencia a tener zonas relativamente planas según avanzaba su cálculo, lo cual cumplía los criterios de convergencia si estos no eran suficientemente restrictivos.

Debido a que este proyecto se basa en la creación de un algoritmo automatizado, se ha definido un cierto número de iteraciones por debajo del cual cualquier solución del problema sea capaz de entrar en estos criterios (si es que puede llegar a cumplirlos). Para comprobar si nuestros criterios de convergencia se han llegado a cumplir, se ha creado un "Update Events" que se activa solo cuando todos ellos están cumplidos. Además de esto, se ha definido un "Stopping Criteria" para no realizar más iteraciones de las necesarias cuando la solución ya está convergida pero no ha llegado al número máximo de iteraciones. En la Figura 17 y 18 podemos ver como Star CCM+ detecta que los resultados han convergido según nuestros criterios y, ante esta señal, el algoritmo varía el punto de funcionamiento para así obtener el mapa completo del compresor.

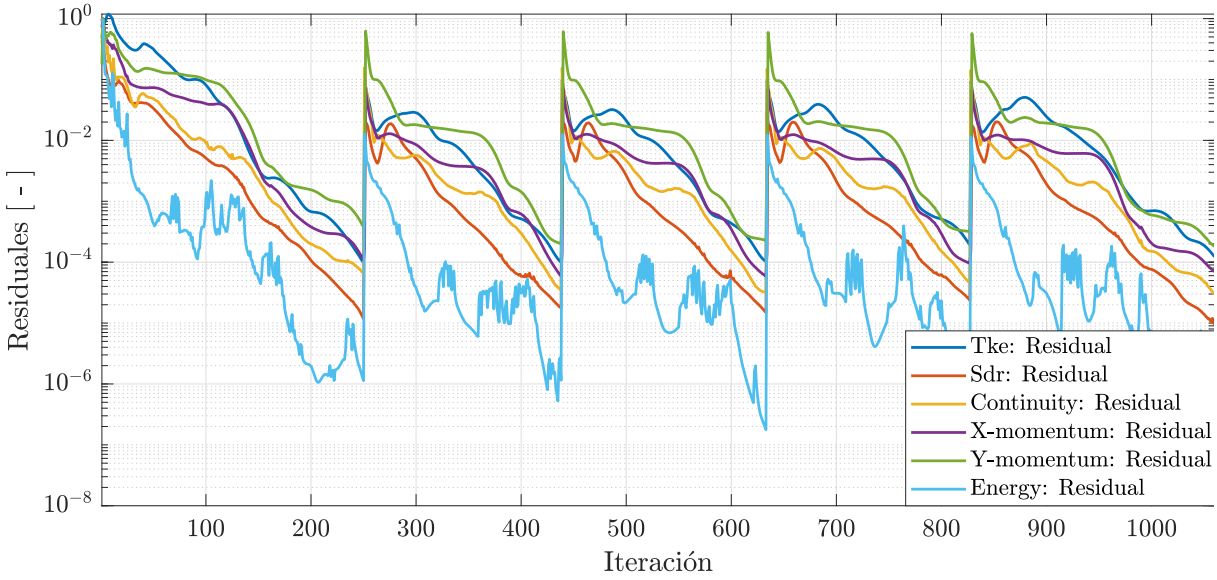


Figura 17: Criterio de Convergencia Cumplido en Residuales

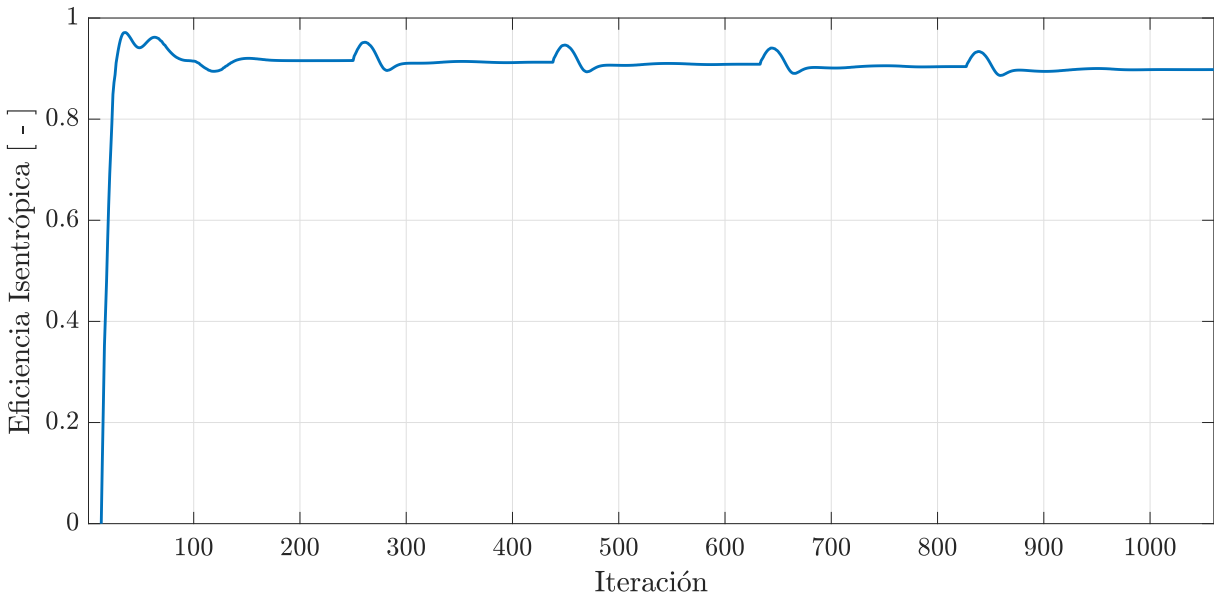


Figura 18: Criterio de Convergencia Cumplido en Eficiencia Isentrópica

2.6. Estudio de Sensibilidad de Malla

Para asegurar la independencia de los resultados de la malla usado, se ha disminuido el tamaño base así como el y+ del "Prism Layer" de la malla base mostrada en la Figura 14 hasta asegurar la convergencia de los parámetros globales (ver Tabla 1).

Tabla 1: Parámetros del Mallado

Mallado	**Tamaño Base**	**"Prism Layer Number"**	**Número de Elementos**	**y+**	η	$\dot{W}$	**Mach Máximo**	Π_c
0	0.045 m	3	7668	83	0.92	460912	0.782	1.163
1	0.030 m	6	18609	5.30	1.5 %	2.5 %	0.5 %	0.6 %
2	0.020 m	8	39542	1.58	2.0 %	1.4 %	1.8 %	0.5 %
3	0.0125 m	12	97944	0.32	0.7 %	1.8 %	0.7 %	0.4 %

Además de estos, también se han comparado las distribuciones de presiones sobre los perfiles del rotor y estátor para asegurar la correcta independencia de los resultados de la malla (Figura 19 y 20).

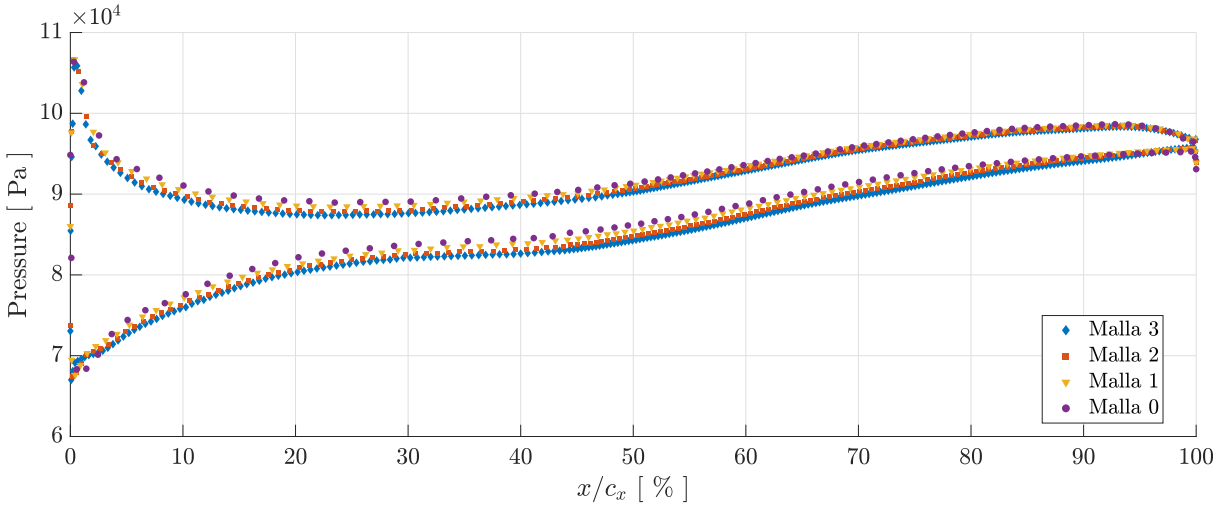


Figura 19: Distribución de Presiones sobre Rotor para cada Malla

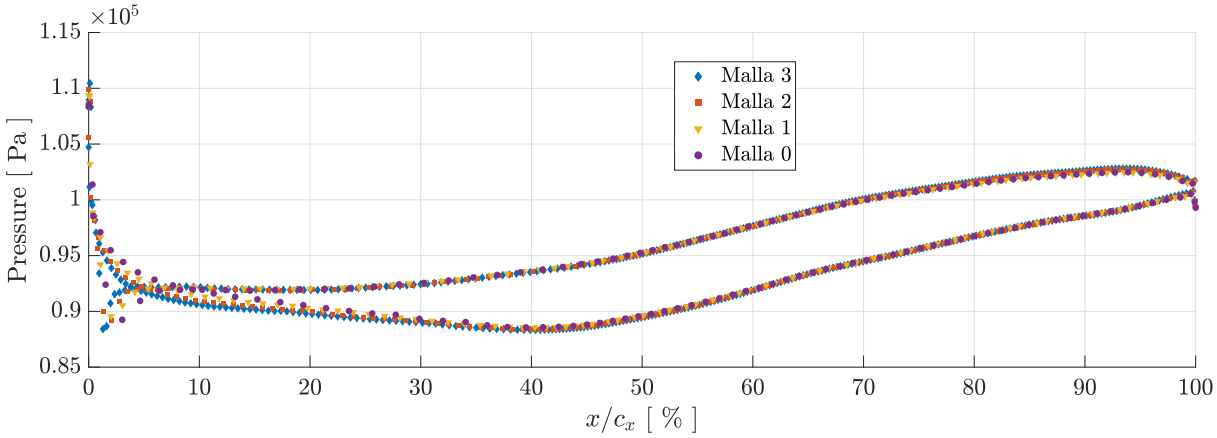


Figura 20: Distribución de Presiones sobre Estátor para cada Malla

En los resultados mostrados anteriormente (Tabla 1, Figura 19 y 20) podemos observar que, aunque la Malla 3 sería la que nos proporcionaría los resultados más precisos, la Malla 1 sería más adecuada para el estudio. Esto se debe a que, debido a que hemos hecho simplificaciones que obtendrán resultados representativos pero no con demasiada exactitud del problema a resolver[4], no tendría demasiado sentido quinduplicar el número de elementos para obtener resultados no exactos. Dicho esto, resulta más importante reducir los tiempos de cálculo que refinar en exceso el mallado.

[4]Pasar a un dominio 2D elimina pérdidas secundarias que pueden representar una parte relevante del problema si deseamos obtener resultados exactos

3. Desarrollo del Algoritmo de Mapeado

Introducción

Los cálculos de los distintos puntos de funcionamiento necesarios para obtener el mapa de operación del compresor se obtendrán mediante Star CCM+. En este se ha creado un modelo donde el punto de funcionamiento del sistema se selecciona mediante la modificación de 2 parámetros, para así tener una estructura más sencilla en el algoritmo. Para aumentar la velocidad de cálculo, se analizará el radio medio del compresor. De esta forma, pasamos a tener un problema 2D.

Como lenguaje para la implementación de los algoritmos se ha elegido Java. Esto se debe a que Star CCM+ cuenta con un sistema integrado de Macros en este lenguaje, por lo que esto posibilita tener una buena integración entre algoritmo y solver. Esta buena integración entre ellos resulta en una gran libertad de modificación del caso a resolver desde la Macro. El sistema de mapeado se ha dividido en varias macros, para así tener un programa modular, permitiendo así la actualización y mejora del código de una forma más sencilla y práctica. Estas tienen la siguiente estructura:

- **Estimador:** Se encarga del mapeado externo del mapa, para tener una primera aproximación de su tamaño así como para poder elegir una distribución eficiente de los distintos puntos de funcionamiento que se van a calcular.
- **Mapeado Estacionario:** A partir de los datos proporcionados en la primera etapa del sistema, esta macro realiza un barrido de los diferentes puntos de funcionamiento para ciertos valores de gasto másico.
- **Mapeado Transitorio:** Las zonas más exteriores del mapeado están afectadas por efectos transitorios del problema, por lo que sus resultados pueden diferir significativamente respecto al problema real estudiado. Por tanto, como última etapa del sistema se realiza un cálculo transitorio de aquellos puntos de funcionamiento que hayan podido ser afectados.

Por último, también se creará una GUI en MATLAB para poder visualizar el flujo de la cascada y sus variables para cada punto de funcionamiento.

3.1. Estimador

Antes de lanzar un algoritmo más costoso, es interesante poder obtener información sobre el tamaño del mapa que queremos calcular mediante métodos más rápidos, así como tener una primera estimación del coste computacional y tiempo de cálculo. Para ello, se ha desarrollado un algoritmo capaz de buscar los bordes inferior y superior del mapa[5].

Teniendo en cuenta que los aspectos más relevantes que debe tener este algoritmo son estabilidad en su funcionamiento y búsqueda de resultados únicos, se ha optado por usar un método "Newton-Rhapson". Para evitar la sobreoscilación por bajo gradiente, se ha limitado la correción máxima por paso.

[5]Llamamos bordes a las líneas de bloqueo a la salida y stall/surge (Figura 4)

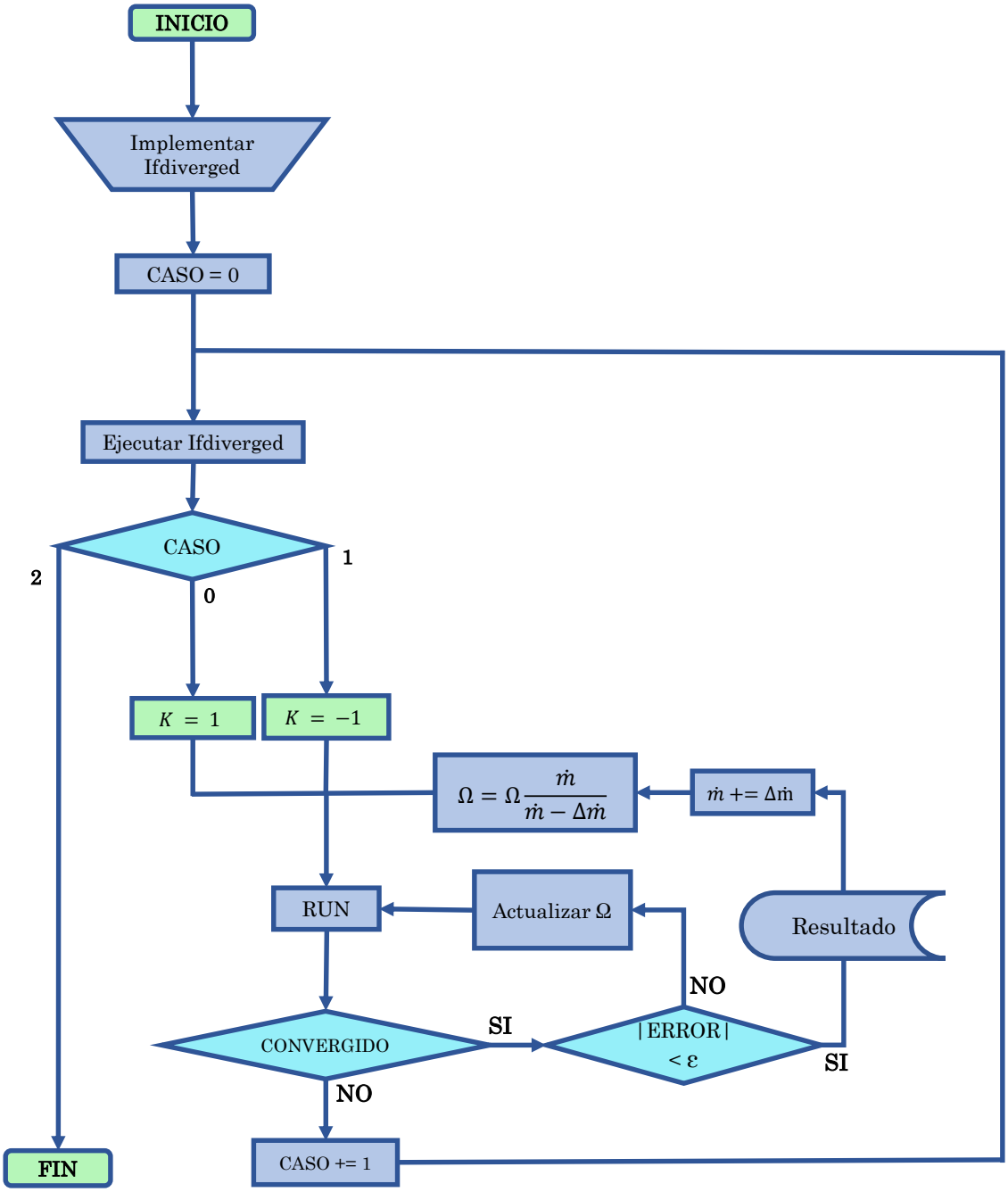


Figura 21: Diagrama de Flujo de Algoritmo Estimador de Mapa

En la Figura 21 se puede observar la estructura del algoritmo. Ya que programar un proceso que asegure la convergencia del primer cálculo independientemente de la geometría del compresor resulta prácticamente imposible, esta tarea queda destinada al usuario. Para esto, el usuario puede grabar el proceso realizado para llegar a un primer cálculo convergido y copiar el código resultante dentro del método “Ifdiverged”, destinado para esto.

A partir de esta solución, el algoritmo avanza hacia gastos másicos superiores buscando el régimen que cumple que el error[6] esté por debajo de cierto margen. Esto lo hace tanto para el borde superior como inferior.

"Actualizar Ω"se encarga de actualizar el régimen de giro para buscar el borde del mapa. En este se ha limitado la corrección máxima por paso al 5 % para evitar overshooting por bajo gradiente. Además, para que la solución no dependa de las condiciones iniciales, el sentido de desplazamiento está determinado por K, donde $K = 1$ busca el borde superior y $K = -1$ busca el inferior.

Es importante destacar que este mapeado se realiza con el modelo físico llamado "segregated". Esto se debe a la mayor estabilidad y rapidez de este frente al "coupled". Como el estimador solo busca obtener soluciones aproximadas y posteriormente estas van a ser recalculadas con más precisión, no resulta relevante que los resultados que pueda obtener este sean menos exactos.

3.2. Mapeado Estacionario

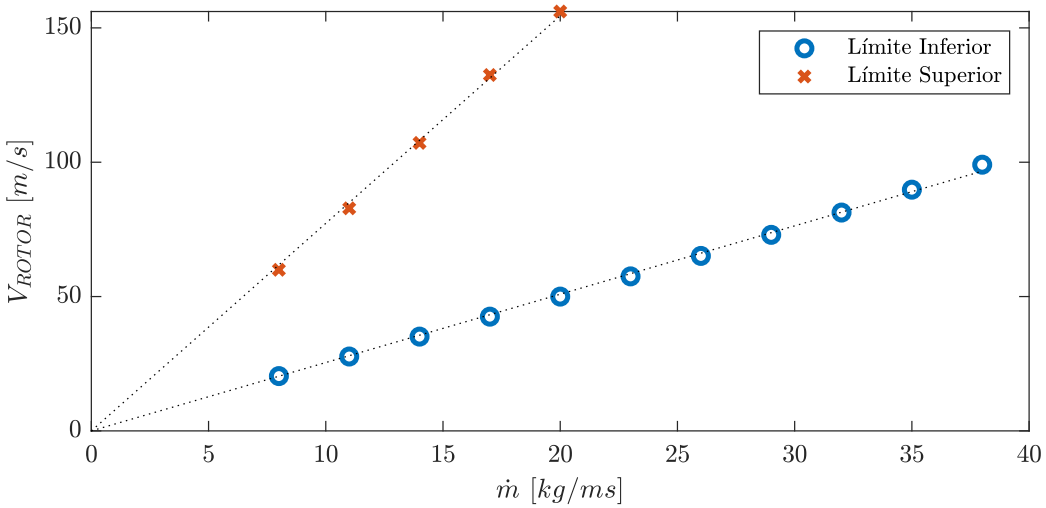


Figura 22: Resultado Algoritmo Estimador

Una vez ejecutado el algoritmo estimador, obtenemos resultados como los obtenidos en la Figura 22. Esta se ha realizado buscando la isolínea de eficiencia 80 % para el borde superior y 75 % para el inferior. Como se puede observar claramente, estas líneas se ajustan una recta que pasa por el origen. Esto se debe a que la eficiencia depende de la incidencia de los perfiles, por lo que los puntos que tengan $V_{ROTOR}/\dot{m} = cte$ tendrán una eficiencia aproximadamente constante[7]. Comparando sobre el mapeado resultado, vemos que este ajuste se cumple bien en la zona mencionada (Figura 23).

[6]Esto puede ser cualquier variable definida por el usuario, por ejemplo buscar cierta eficiencia o $\frac{\partial \varepsilon}{\partial \pi_c}$

[7]Esto se cumple siempre que el desprendimiento no sea excesivo y/o el Mach no sea elevado

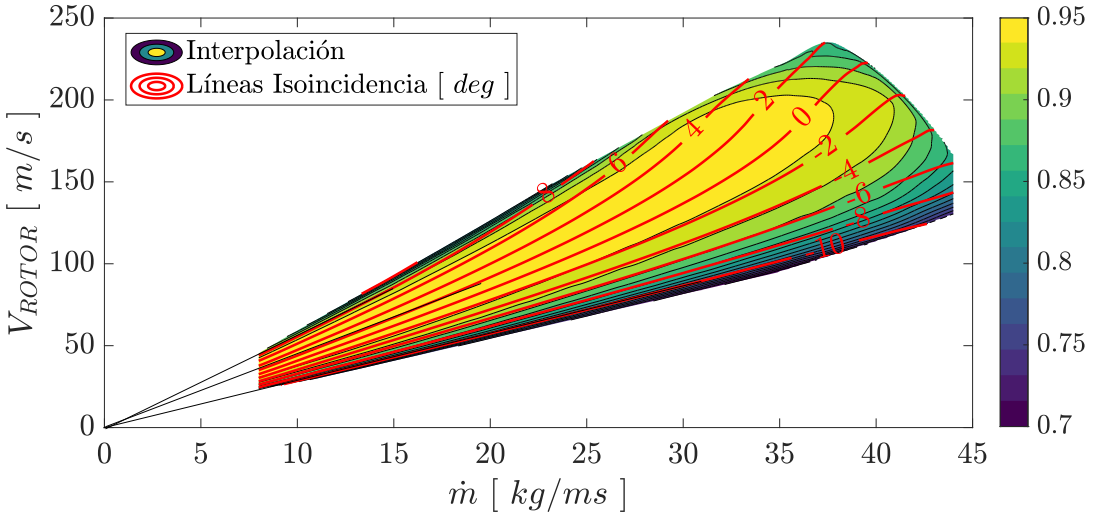


Figura 23: Comprobación Ajuste Lineal del Algoritmo Estimador

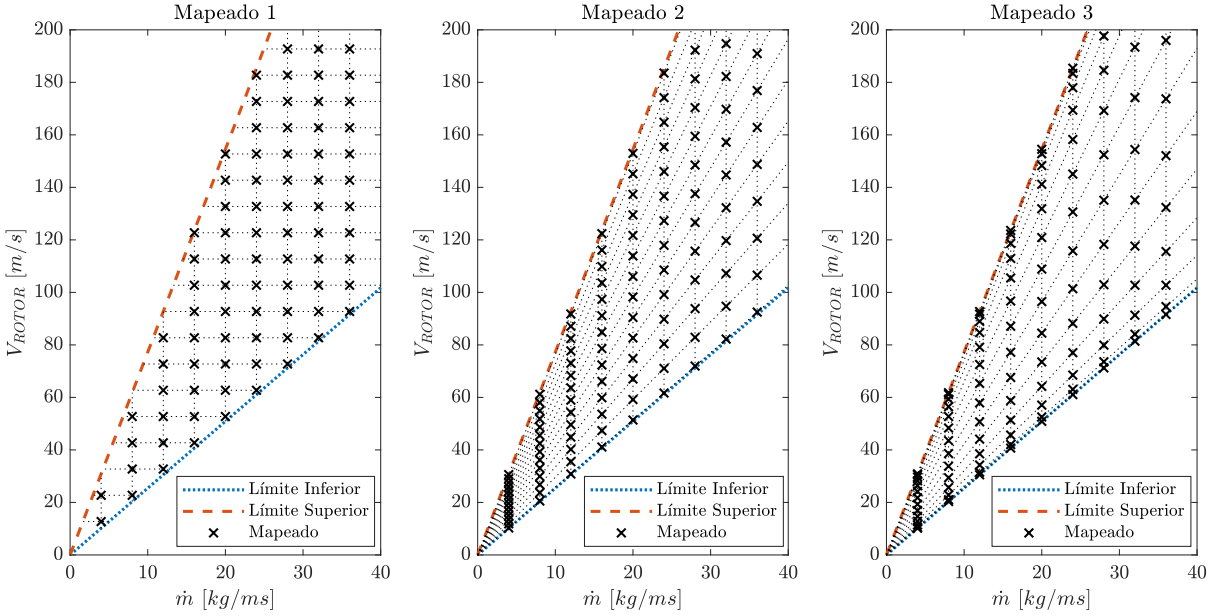


Figura 24: Tipos de Mapeado

Con esto en mente, resulta interesante usar esta propiedad para mapear la zona de bajo gasto másico del mapa. Por tanto se plantean los tres tipos de mapeado que se observan en la Figura 24.

Procedemos a exponer sus características, así como sus ventajas e inconvenientes (Tabla 2):

Tabla 2: Características, Ventajas e Inconvenientes de cada Mapeado

Tipo	Características	Ventajas	Inconvenientes
Mapeado 1	Datos obtenidos a valores de $\dot{m}$ y V_{ROTOR} equiespaciados	▪ Interpolación de resultados más sencilla	▪ Pocos puntos a bajo gasto másico ▪ Muy pocos puntos en zonas de altos gradientes
Mapeado 2	Datos obtenidos a valores de $\dot{m}$ y $V_{ROTOR}/\dot{m}$ equiespaciados	▪ Buena resolución en bordes	▪ Interpolación complicada ▪ Bastantes puntos en zonas de bajos gradientes ▪ Igual cantidad de puntos para cada $\dot{m}$
Mapeado 3	Datos obtenidos a valores de $\dot{m}$ equiespaciados y distribución senoidal en $V_{ROTOR}/\dot{m}$	▪ Mucha resolución en bordes ▪ Poca resolución en zona de bajos gradientes	▪ Interpolación más complicada ▪ Igual cantidad de puntos para cada $\dot{m}$

Se ha observado que la convergencia de resultados es más rápida ante una variación de V_{ROTOR} que ante una variación de $\dot{m}$. Por tanto, el mapeado se ha centrado en realizar un barrido de V_{ROTOR} para cada valor de $\dot{m}$. Además, teniendo en cuenta que es primordial evitar una divergencia que produzca errores tales como "Float Point" en la solución, el mapeado se realiza de centro a borde. Para aumentar esta estabilidad, este se realiza siguiendo el patrón centro−inferior centro−superior (ver Figura 25). Esto se debe a que la zona superior del mapa es más inestable, por lo que con esta estructura aseguramos que solo haya que limpiar la solución 1 vez por cada gasto másico como máximo.

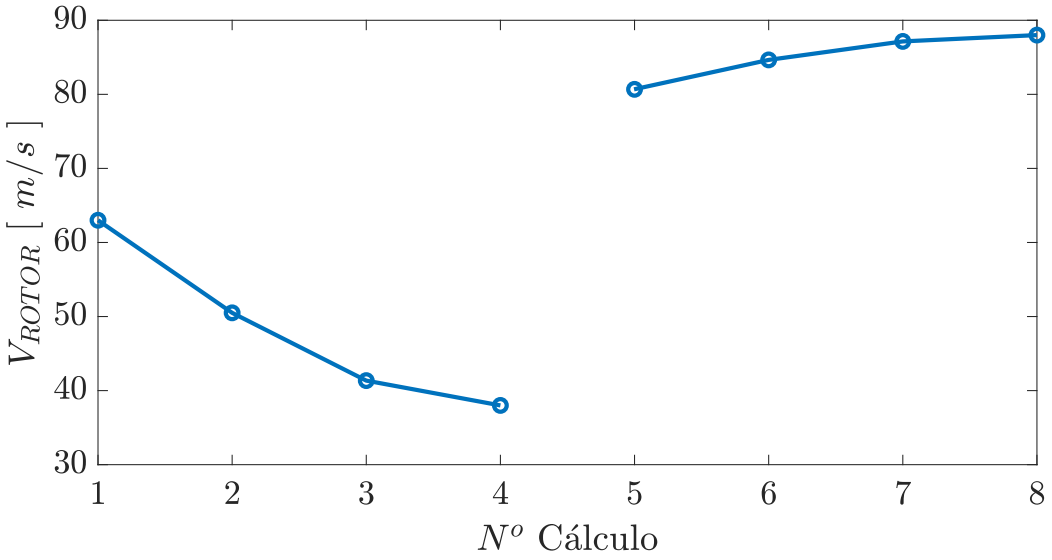


Figura 25: Barrido de Velocidad del Rotor para un $\dot{m}$

Este proceso de mapeado también es interesante para la parte de gasto másico alto del mapa. Aunque estos puntos no sigan la ley lineal que se observa a bajos gastos másicos, estos puntos centrales cumplen que serán habitualmente los de mayor eficiencia[8] al tener incidencia nula o casi nula sobre el perfil.

Respecto al gasto másico hasta el cual se mapea, es decisión del usuario si se limita bien por un régimen máximo o una línea de isoeficiencia. En caso de requerir que pare por cualquier otro parámetro, se puede incluir en el algoritmo.

[8]Para un gasto másico dado

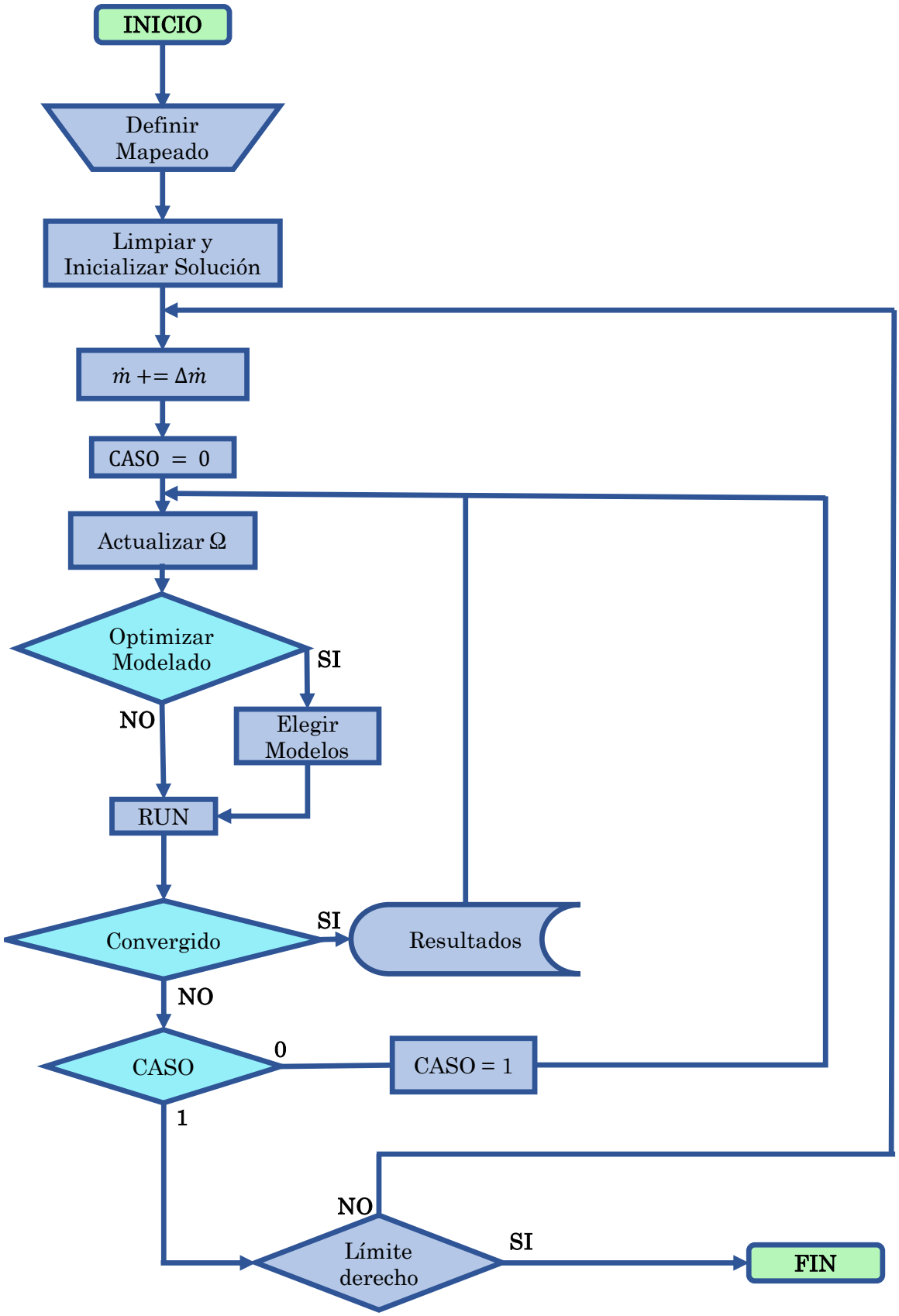


Figura 26: Diagrama de Flujo del Algoritmo de Mapeado Estacionario

Como se observa en la Figura 26, además de dar la opción de elegir el tipo de mapeado que se quiere realizar, el algoritmo da la opción de reducir el tiempo de cálculo mediante una selección automática del solver usado en cada momento. Para el algoritmo, esto consistirá en elegir entre el solver acoplado (Coupled) y segregado (Segregated) que presenta Star CCM+. Mientras que "Coupled" presenta unos resultados más similares a los del problema estudiado, este resulta más costoso computacionalmente y tiene una convergencia más lenta y un tanto más inestable. Por tanto, resulta útil considerar el "Segregated" cuando ambos dan resultados similares, por su convergencia más rápida y menor coste computacional. Teniendo en cuenta que cuando nos encontramos a Mach bajo y la incidencia sobre los perfiles no es demasiado elevada esto se cumple, podemos determinar la zona en la que interesa realizar un mapeado con "Segregated".

Por tanto, para reducir el tiempo de cálculo empleado en un mapa, se ha optado por definir un umbral de error entre modelos que consideramos aceptable. A partir de la información del problema, se definirán ciertas condiciones de incidencia sobre el perfil y número de Mach a partir de los cuales se hará un cambio de modelos[9] para así comparar el error entre ambos resultados. En caso de obtener un error superior al umbral, se cambiará el solver a "Coupled" para el segmento estudiado.

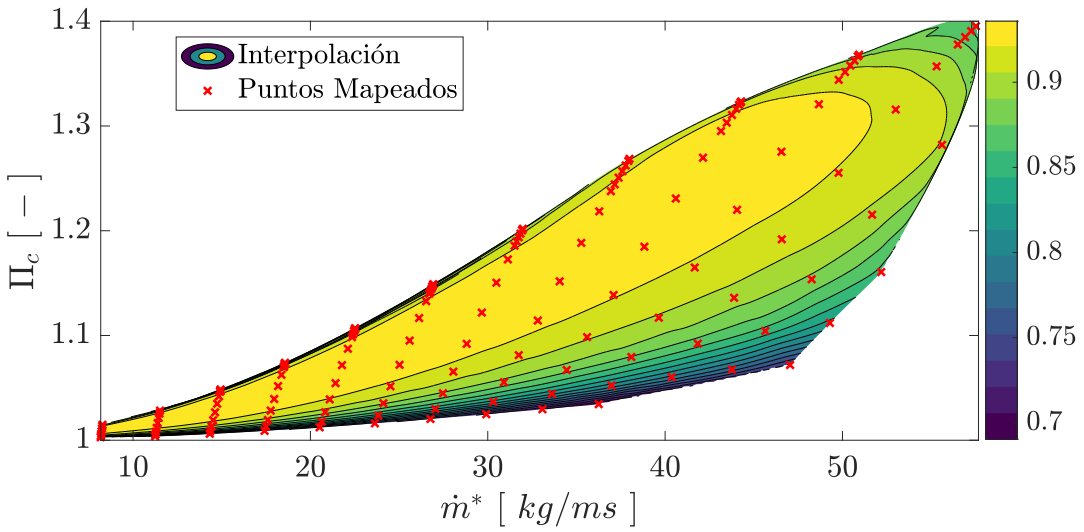


Figura 27: Ejemplo Mapeado

Para definir los umbrales a partir de los cuales se empieza a comprobar el error entre modelos, se ha optado por ejecutar el algoritmo con el modelo "Segregated" y "Coupled", para proceder a comparar los resultados entre estos modelos. A partir de los resultados obtenidos, se definirán los umbrales a partir de los cuales operará el selector de modelos y, para comparar el correcto funcionamiento del selector, se ejecutará el algoritmo en un compresor de distinta geometría.

[9]El modelo "Segregated" es el que tenemos por defecto al ser el más rápido y estable

3.3. Interfaz Gráfica para Visualizar Resultados

Para ayudar con la visualización de toda la información que este algoritmo puede generar, se ha optado por crear una GUI en Matlab para ayudar en esta tarea. Esta GUI no está concebida como una herramienta para obtener información en profundidad sinó como una herramienta que permita una vista rápida del Mapa del compresor analizado, así como poder visualizar el flujo para un punto de funcionamiento dado.

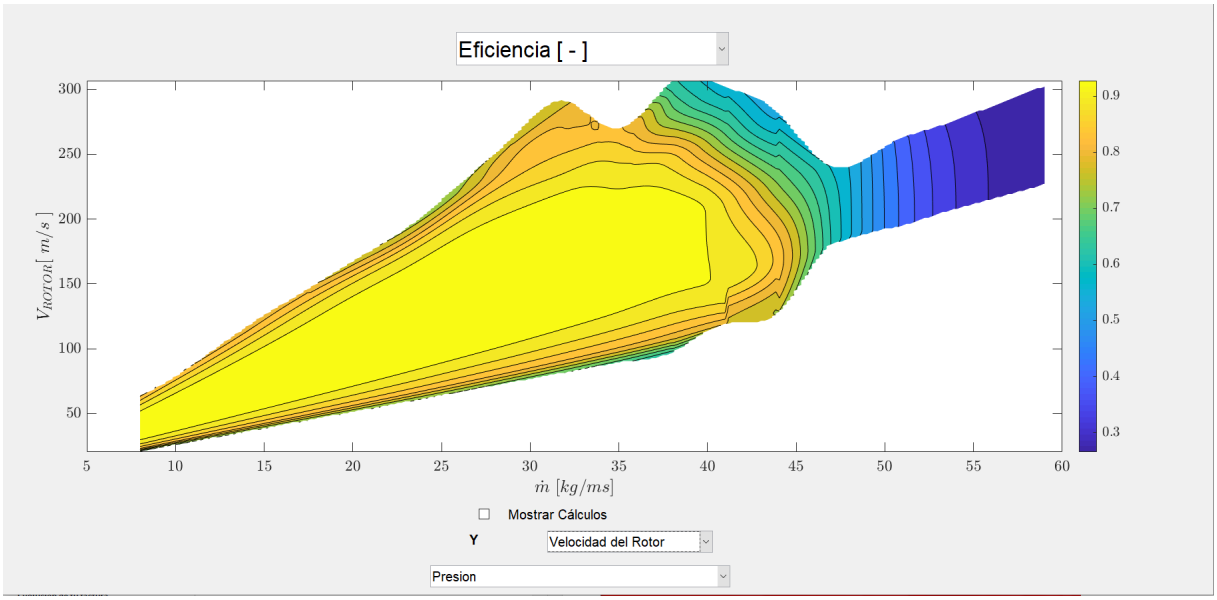


Figura 28: Interfaz GUI MATLAB

En la Figura 28 se puede observar la estructura de la interfaz gráfica. Esta se ha diseñado siguiendo un modelo sencillo y fácil de utilizar, que permita ejecutar todas las acciones que hemos mencionado. Esta tiene un "Selector eje Z" y "Selector eje Y", que permite seleccionar que variable se va a mapear así como la variable correspondiente al eje Y. Además, se pueden visualizar que puntos de funcionamiento dentro del mapa son los que se han calculado mediante la opción "Mostrar Cálculos".

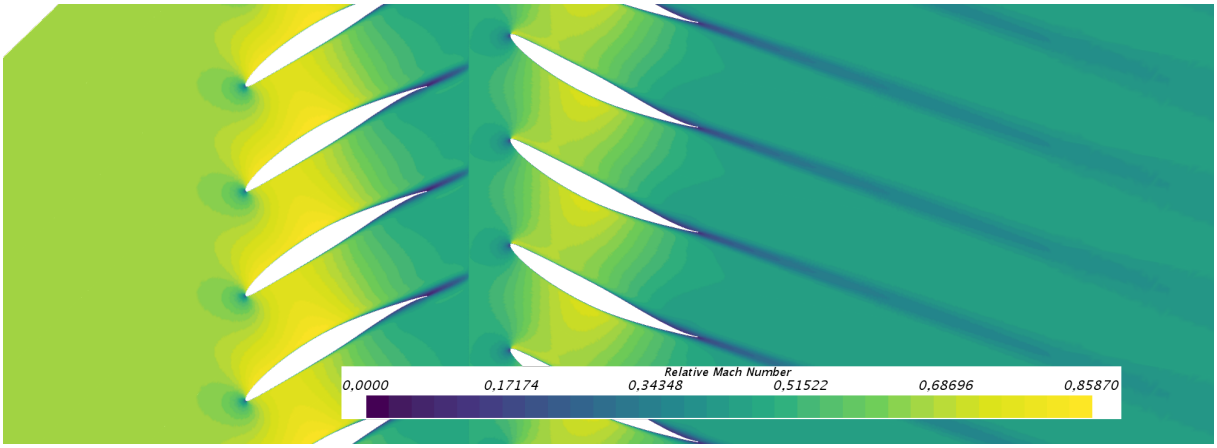


Figura 29: Visualizador de Punto de Funcionamiento

Respecto a la visualización del flujo, el selector "Visualizar Flujo" nos permite seleccionar que función escalar queremos visualizar. Para visualizar un punto en concreto, debemos hacer click izquierdo sobre el punto del mapa que queremos visualizar y se mostrará una imagen correspondiente al punto de funcionamiento más cercano al punto en el que hemos clicado (Figura 29).

4. Resultados y Discusión

4.1. Mapeado de las Distintas Variables de Interés

Una vez ejecutado el algoritmo dentro de Star-CCM+, obtenemos como resultado una carpeta con todas las imágenes asociadas a los puntos de funcionamiento calculados y un fichero csv con todos los parámetros globales guardados. Procesando estos resultados en Matlab, se pueden representar respecto al gasto másico corregido y relación de presiones (totales) entrada - salida. Los resultados mostrados a continuación se han realizado mediante el solver acoplado.

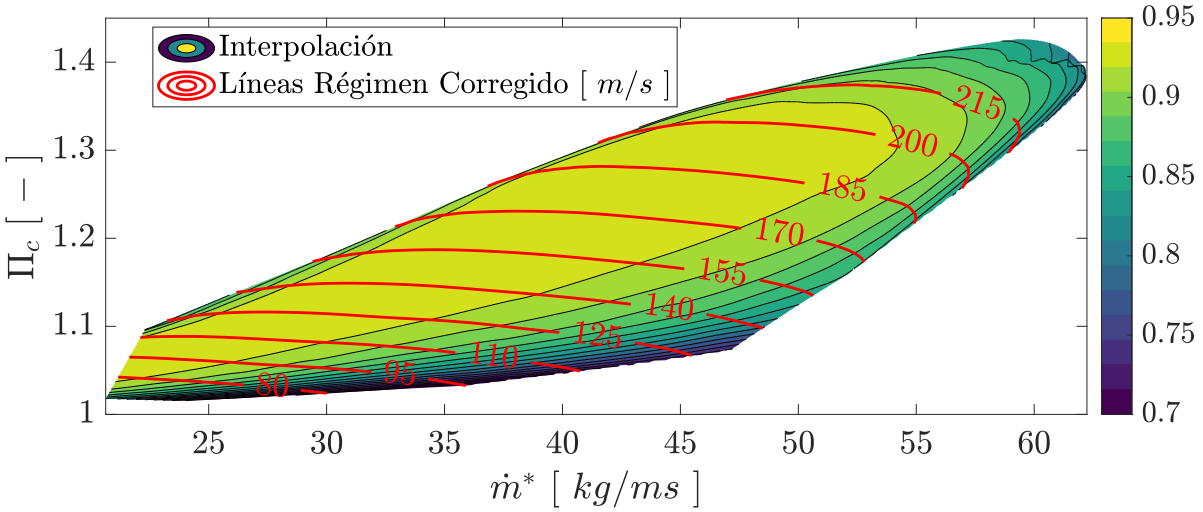


Figura 30: Mapa Eficiencia con Solver Acoplado [-]

Si realizamos este mapeado para la eficiencia (Figura 30), observamos que este tiene un aspecto similar al mostrado en la Figura 4. Esta tiene eficiencias muy inferiores a las calculadas en el estudio realizado (84 % frente a 95 %). No obstante, esto se debe porque en el estudio realizado se ha analizado una sola etapa mientras que la Figura 4 corresponde a un compresor axial multietapa. Para comparar esto de forma correcta, se debe comparar con un compresor con una sola etapa. Para ello, se ha obtenido los resultados a partir de un análisis realizado en un compresor donde se han usado el mismo tipo de perfiles que los del compresor estudiado en este proyecto [1]. Para que la comparación tenga sentido, el calado usado en el artículo y en este estudio tiene un valor similar.

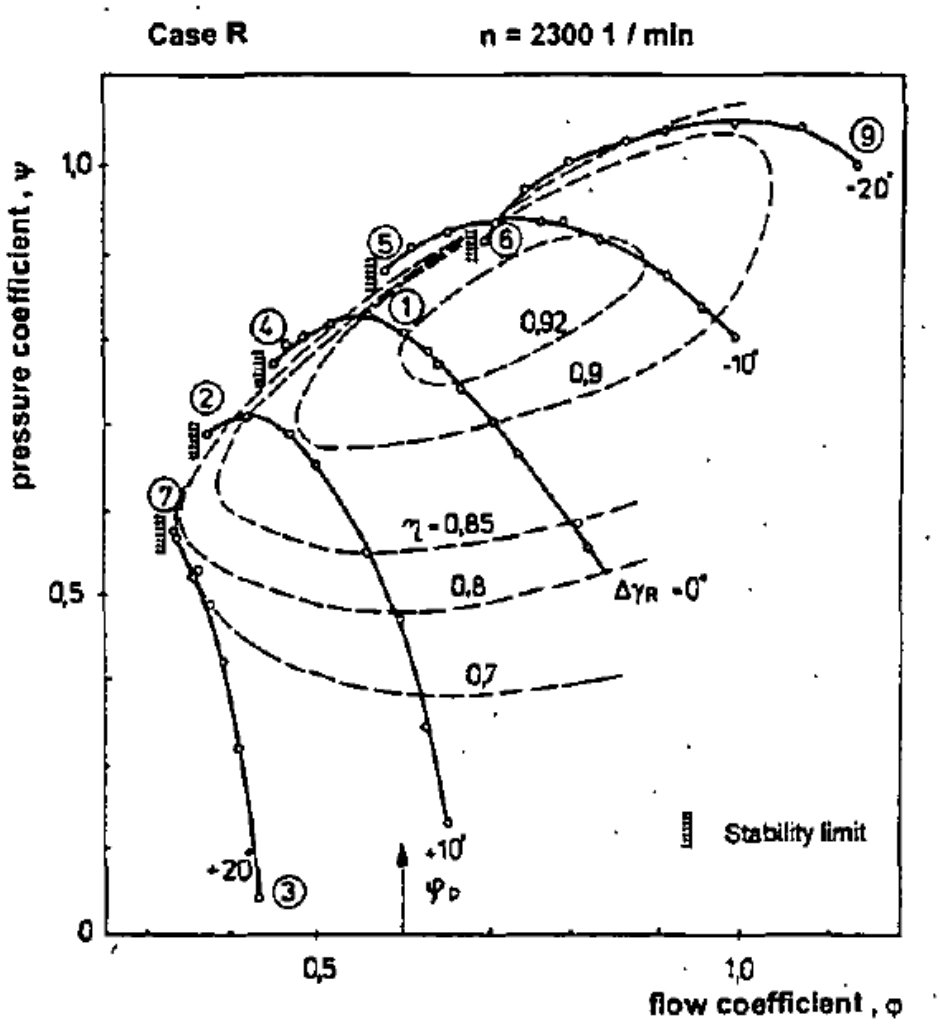


Figura 31: Mapa Eficiencia Compresor una Etapa con Perfil Serie NACA 65 con TE reforzado

En la Figura 31 se observa que, variando el calado del rotor, la eficiencia máxima que se ha obtenido es en torno al 92 % frente al 95 % obtenido en el presente estudio. Esta diferencia se puede deber a que estudiando solo el radio medio del compresor estamos despreciando las pérdidas debidas a los fenómenos tridimensionales (Figura 32). Entre ellos destacan:

- La velocidad de álabe aumenta con el radio
- Aumento del paso y espesor del perfil con el radio
- Curvatura en el plano meridional
- Crecimiento de capa límite en la carcasa (endwall) y tubo
- Flujos secundarios
- Flujo intersticial (tip flow)
- Fuerzas centrífuga y coriolis

No obstante, estas diferencias causadas por los efectos tridimensionales podrían ser corregidas mediante correlaciones.

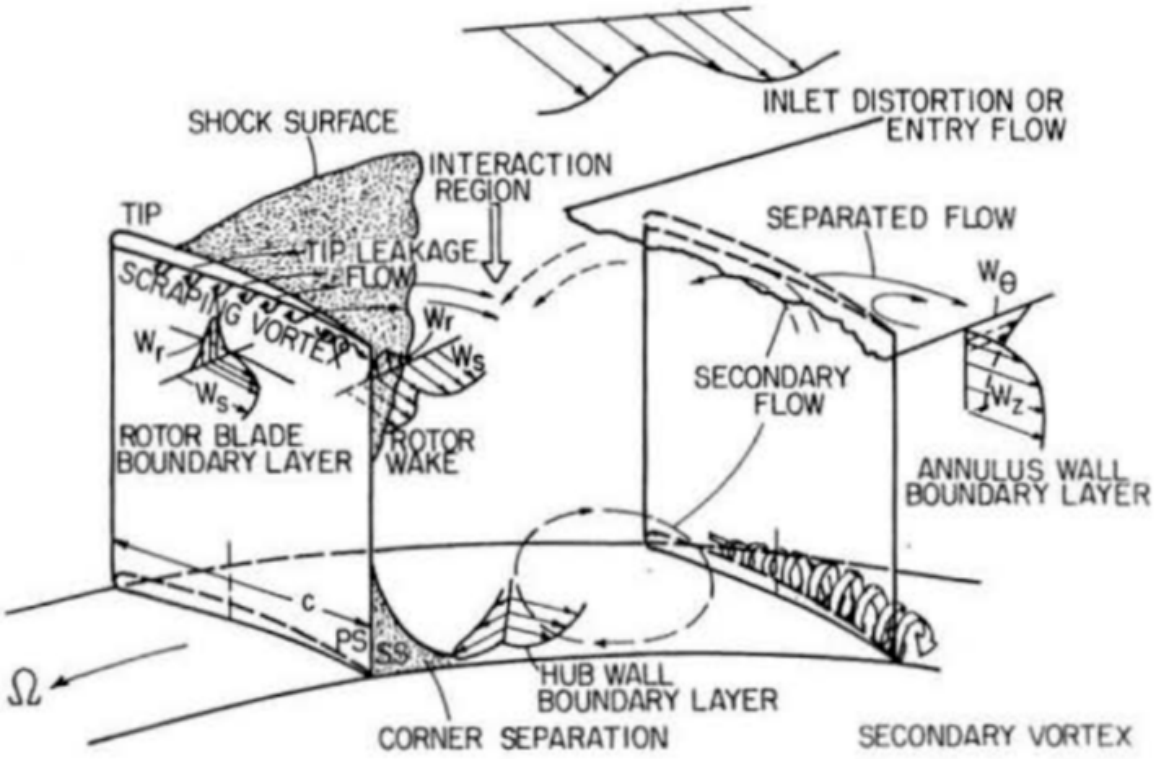


Figura 32: Esquema Pérdidas 3D

En el estudio realizado (con la configuración de Star-CCM+ usada) no ha sido posible llegar a la zona de choque. Calcular la zona con choque no era el propósito de este proyecto, el cual era poder estimar donde se empezaba a producir. Esto se puede observar en la esquina inferior derecha de la Figura 30, a partir de régimen 155 m/s, ya que a partir de este las curvas de isorégimen presentan un alto gradiente (son casi verticales).

Con la configuración de Star-CCM+ seleccionada, se han podido calcular las zonas a partir de las cuales podrían suceder los fenómenos de lazo de bombeo/desprendimiento rotativo. Así mismo, también se ha podido mapear correctamente la zona inferior del mapa (la cual se correspondería con un bloqueo a la salida en un compresor multietapa). Si se deseara llegar a una eficiencia inferior en esta zona, se debería bajar el límite de eficiencia hasta el cual calcula el algoritmo.

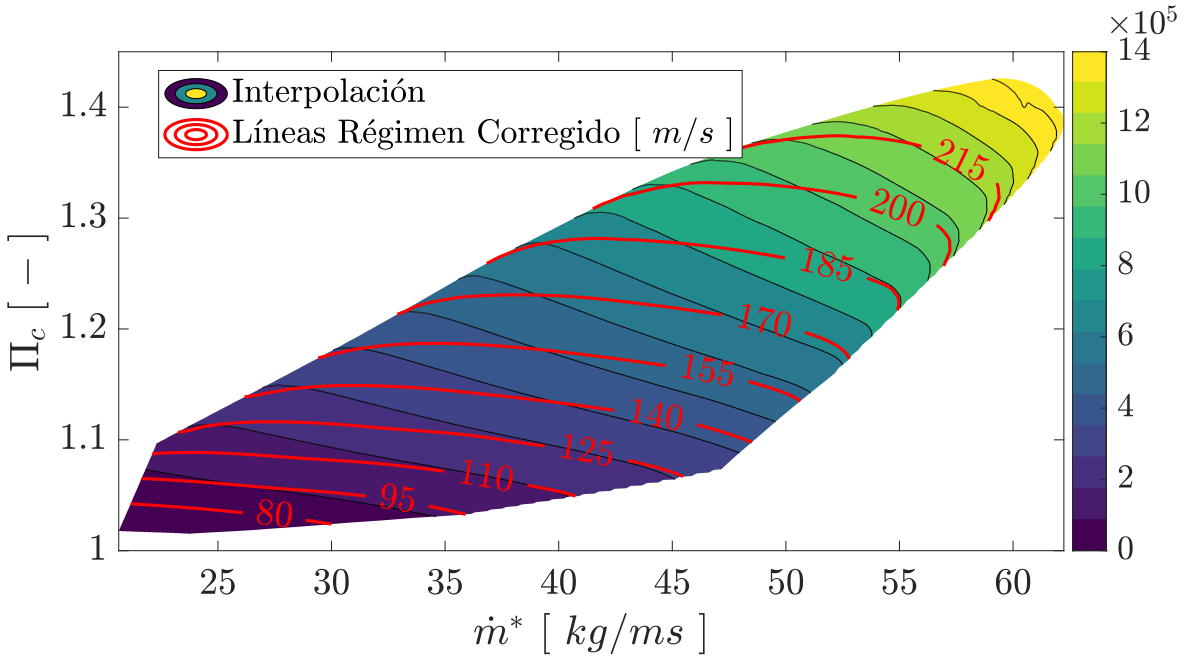


Figura 33: Mapeado Potencia con Solver Acoplado [W/m]

Respecto a la Figura 33, se podría obtener como conclusiones que a cuanto mayor es la relación de compresión del motor la potencia que se debe aportar es mayor. Coherentemente, un mayor gasto másico también comporta un aumento en la potencia aportada. Dado que la turbina debe extraer del flujo una potencia mayor que la necesaria por el compresor (para compensar las pérdidas), este mapa (Figura 33) es de vital importancia ya que nos mostraría aquellas zonas en las que el sistema no podría operar de forma estable (si disponemos de un mapa de potencia de la turbina).

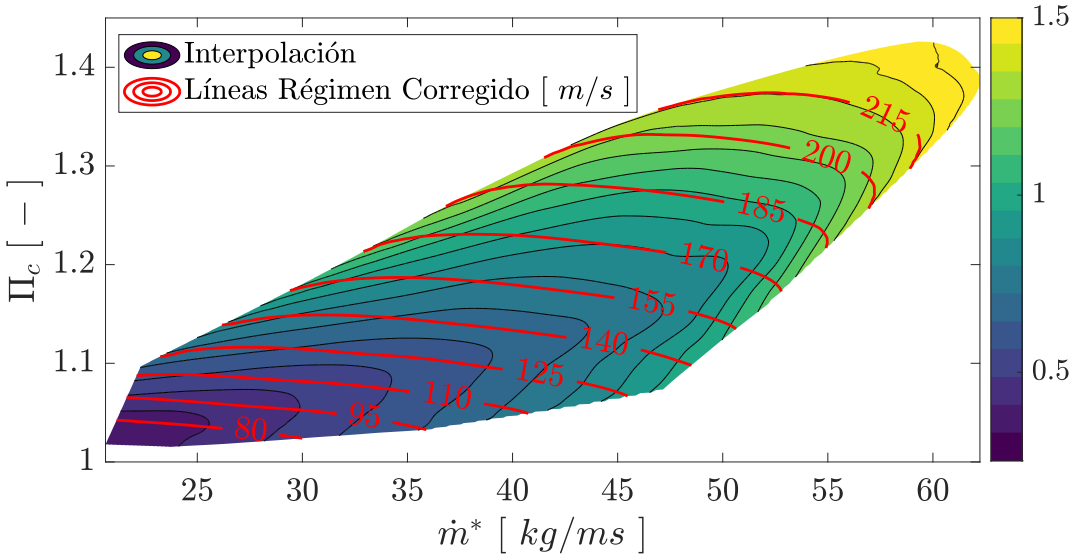


Figura 34: Mapeado Mach Máximo Relativo con Solver Acoplado

Debido a que tanto una mayor relación de presiones como un mayor gasto másico están directamente relacionados con una mayor velocidad del flujo, el mach máximo en el dominio aumenta con estos. Además, cabe considerar que si la incidencia sobre los perfiles no es cero, el flujo debe acelerarse para adaptarse a la geometría, aumentando significativamente el número de Mach máximo en el dominio (Figura 34).

4.2. Comparación entre Modelos de Resolución

Ya que el solver acoplado es más costoso que el segregado y el resultados proporcionado por estos coincide en gran parte del mapeado, resulta interesante definir en que zonas del mapa sería interesante usar uno u otro para así ahorrar en tiempo y coste computacional. Para ello, se identifican entre las principales razones de esta diferencia el número de Mach y la incidencia sobre los perfiles. Debido a que el solver acoplado resuelve mediante un "pseudo-paso temporal", predice mejor los resultados en aquellos casos donde la incidencia y/o el mach es alto. Usando la incidencia sobre el rotor y la velocidad de entrada axial como parámetros de interés del problema, podemos representar estos sobre un mapa correspondiente al error relativo entre modelos. De esta forma, se identificarán las zonas de cambios de modelos.

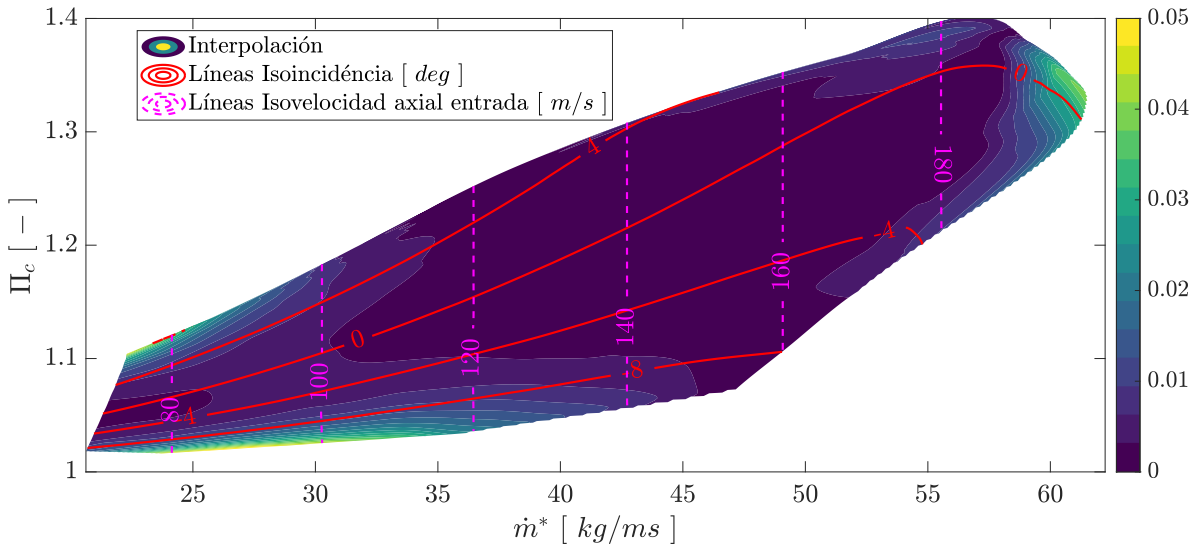


Figura 35: Error Relativo Eficiencia

Respecto al error relativo en la eficiencia entre el modelo segregado y acoplado (Figura 35), se observa que tal como se esperaba que los máximos en este se encuentran en las zonas de alta y baja relación de presiones (alta incidencia) y alto gasto másico (alto mach). Usando como referencia las isolíneas dibujadas sobre el mapa, se llega a la conclusión que dentro de la región encerrada por $-8° \leq i_0 \leq 4°$ y $V_a \leq 180\ m/s$, se cumple que el error relativo es menor que el 2.5 % por lo que se consideraría aceptable la solución proporcionada por el solver segregado.

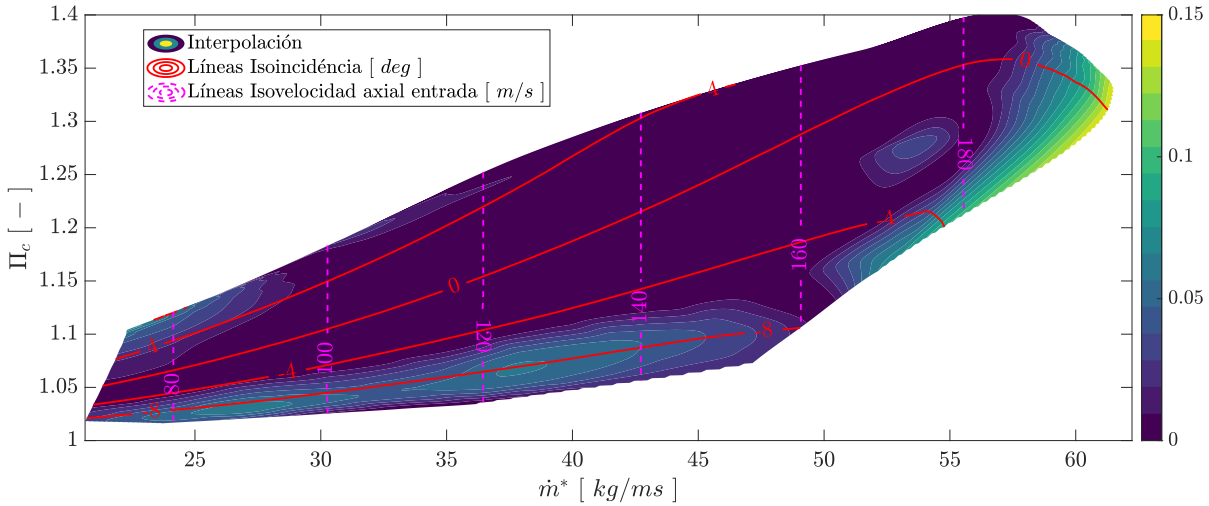


Figura 36: Error Relativo Máximo Mach Relativo

Respecto al error relativo en el máximo mach relativo entre ambos solvers, se observa que en este caso la zona más afectada es aquella donde se producen efectos de compresibilidad (alto gasto másico). Las zonas de alta incidencia también están afectadas, por lo que para cumplir el criterio del 2.5 % la región sería la que cumpla que $-4° \leq i_0 \leq 2°$ y $V_a \leq 160\ m/s$. Es bastante notable que en la zona de incidencia negativa se aprecian errores relativos mayores que en la zona de incidencia positiva. Esto se puede deber a que, ante una incidencia negativa, el flujo sufre un gran aumento de velocidad en la cara de presión (pudiendo llegar a producir desprendimiento).

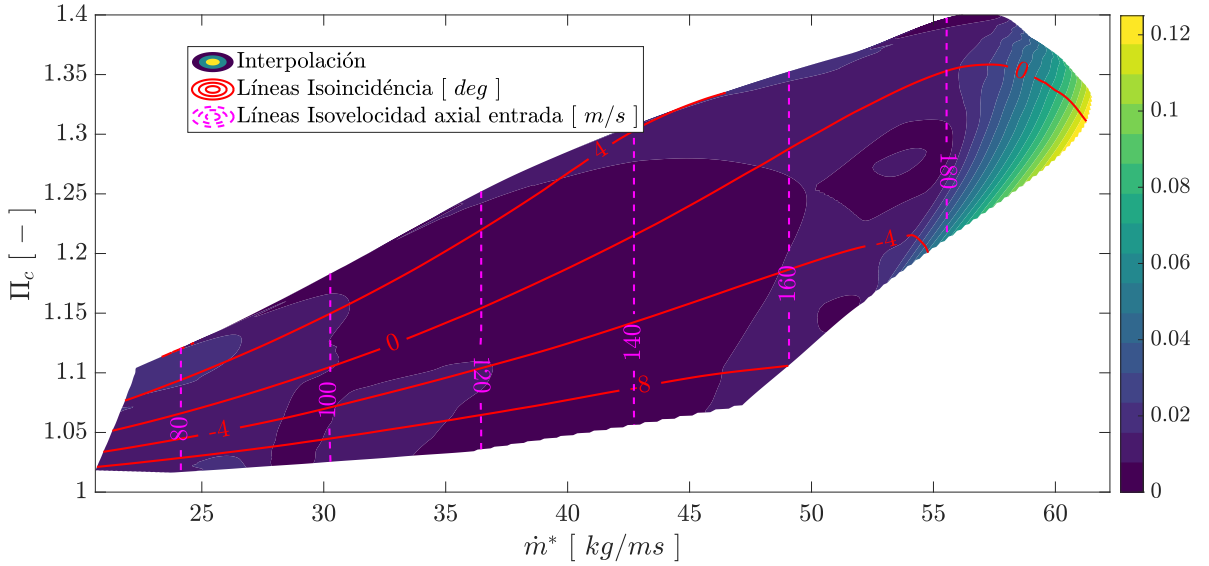


Figura 37: Error Relativo Potencia

Observando detalladamente el error relativo del mapeado de la potencia, se aprecia que no existe una gran diferencia entre ambos solvers en las zonas de alta incidencia, en las cuales se aprecia un error bastante uniforme. Por tanto, para el perfil estudiado y barrido realizado en este estudio no se aprecian cambios tan significativos entre los dos tipos de resolución como en los mapas anteriores (Figura 35 y 36). Si que se observa que la zona de alta relación de presiones y bajo gasto másico el error es el máximo de la zona de gasto másico bajo, correspondiéndose con la zona de error relativo máximo de la Figura 36.

Centrándonos en la zona de altos gastos másicos, a partir de un gasto corregido de 45 kg/ms ya se empieza a observar que el error entre solvers empieza a aumentar, siendo muy clara esta diferencia a partir de cuando este es de 55 kg/ms. Por tanto, los límites que se definirían en este caso serían $-8° \leq i_0 \leq 4°$ y $V_a \leq 170\ m/s$.

A partir de la información proporcionada por los tres mapeados de error, se concluye que los límites que cumplirían con las restricciones de los tres modelos es $-4° \leq i_0 \leq 2°$ y $V_a \leq 160\ m/s$. Para comprobar que estos han sido seleccionados de forma correcta, se han definido las zonas donde alguno de los errores de mapeados sean mayores al 2.5 % con un valor y el resto con otro para poder diferenciar claramente estas regiones.

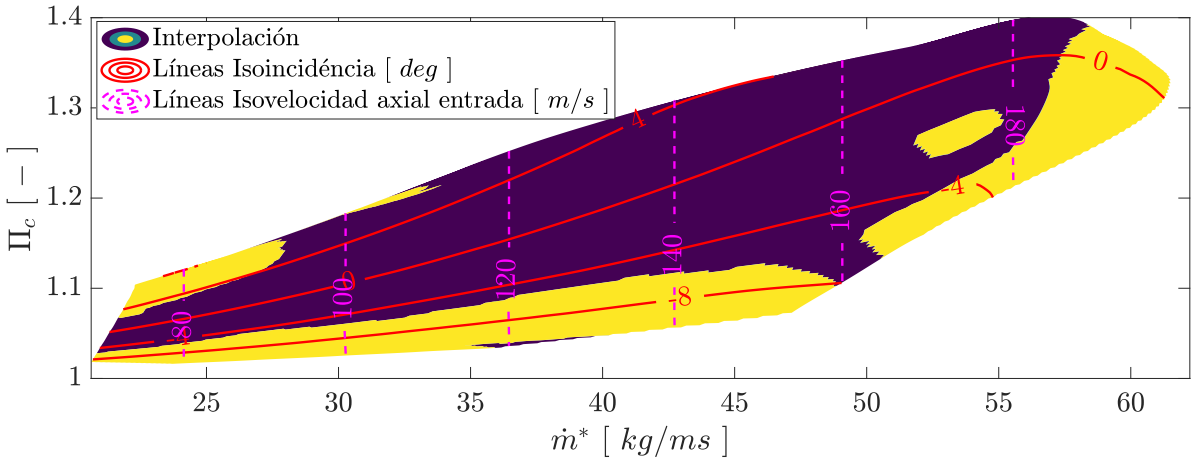


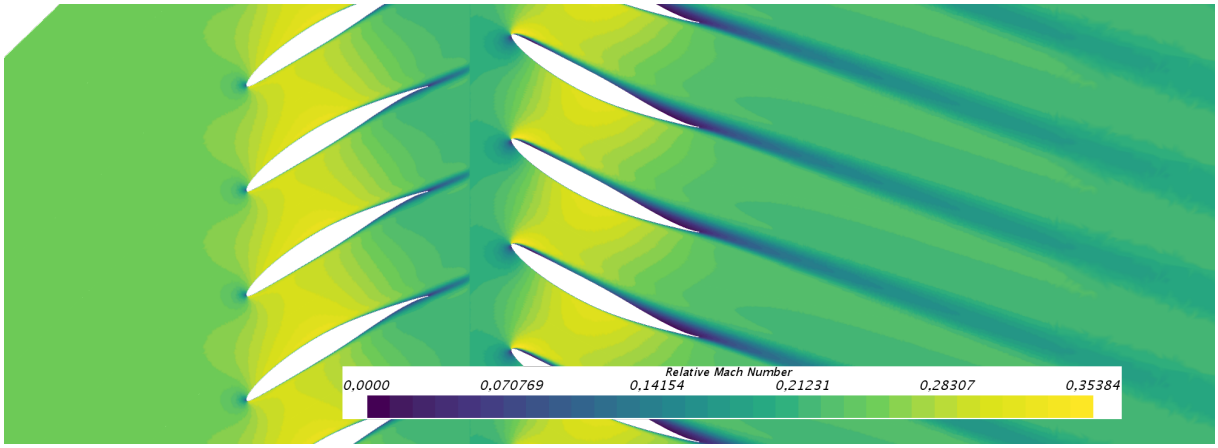
Figura 38: Regiones Diferenciadas por Error Relativo de 2.5 %

En la Figura 38 se puede comprobar como estos límites definidos cumplen con las restricciones.

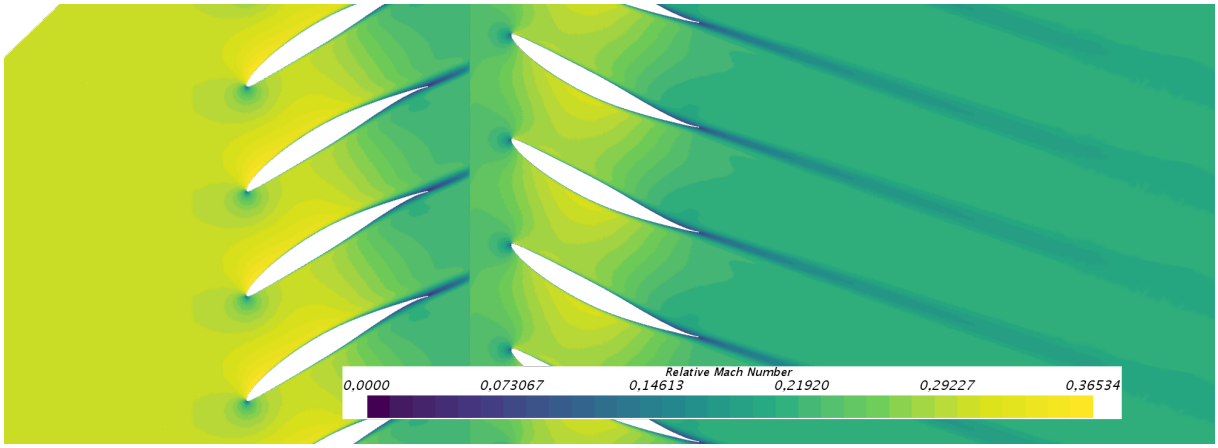
4.3. Estudio del Flujo

4.3.1. Variación en Ratio de Presiones a Mach Bajo

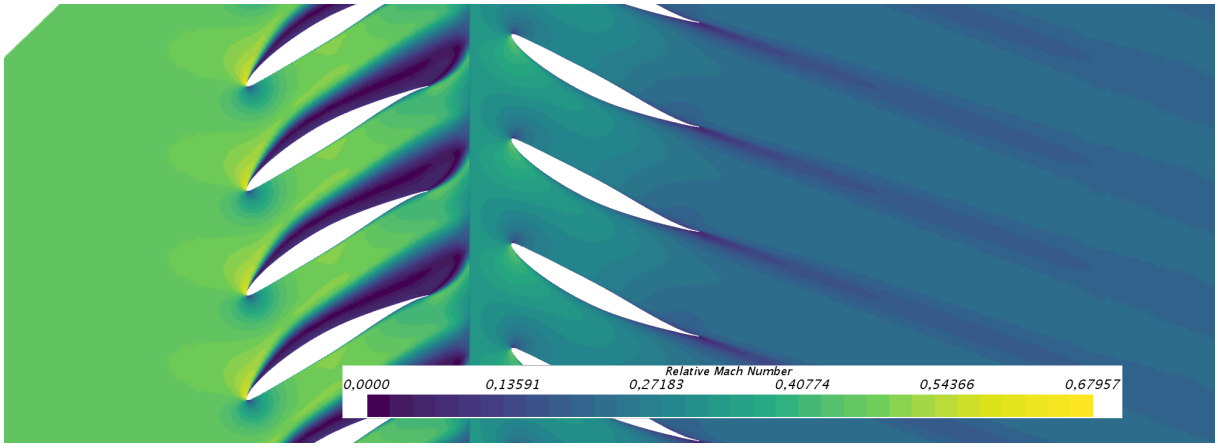
En la Figura 39 se representa la evolución del flujo cuando, para un gasto másico corregido constante de 20 kg/ms, variamos la relación de presiones en el flujo. Una de las principales diferencias se aprecia a la salida del estátor. En esta, la estela del álabe es más estrecha en el caso en que la incidencia es prácticamente nula (Figura 39b). Si nos centramos en el caso de incidencia negativa (Figura 39a), se observa que el mayor desprendimiento se produce en el estátor. En este se aprecia claramente un pico de succión en el borde de ataque del álabe (PS), a causa de la gran curvatura que debe experimentar el flujo para bordear este. No obstante, este no es capaz de adaptarse a una curvatura tan acentuada y el flujo se desprende en la cara de presión. Por el contrario, en el caso de la incidencia positiva (Figura 39c), el desprendimiento se produce en el álabe del rotor (SS). De forma similar a como pasaba en el estátor, el flujo no es capaz de adaptarse a la geometría al ser muy abrubta la curvatura que este debería experimentar. Es interesante remarcar que la estela del rotor no se transmite al estátor por estar usando una condición de contorno “Mixing Plane” entre estos, la cual promedia el resultado. Esto perjudica a la solución, en los casos en los que desprende el rotor pero permite obtener un resultado independiente de la posición relativa rotor-estátor.



(a) Incidencia Negativa



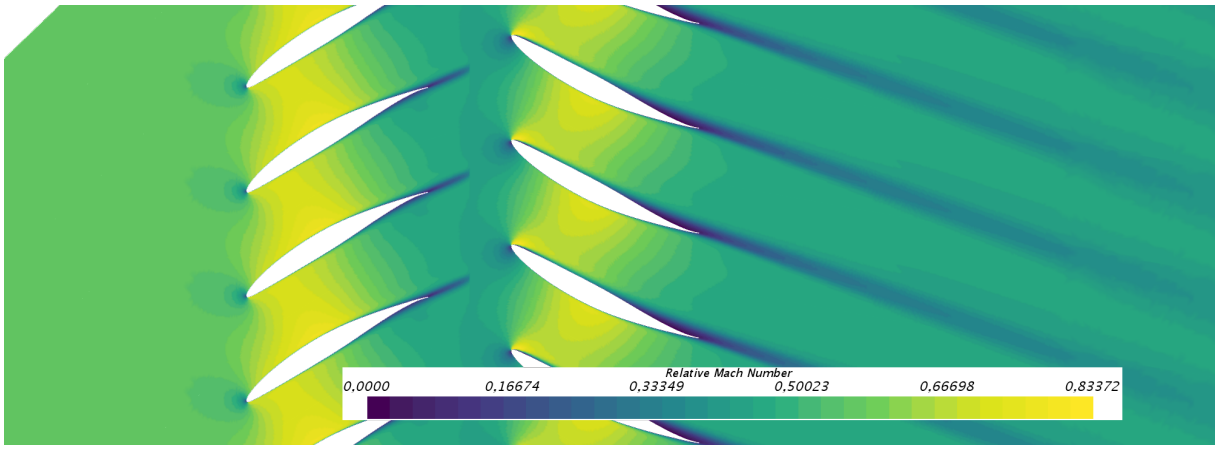
(b) Incidencia Aproximadamente Nula



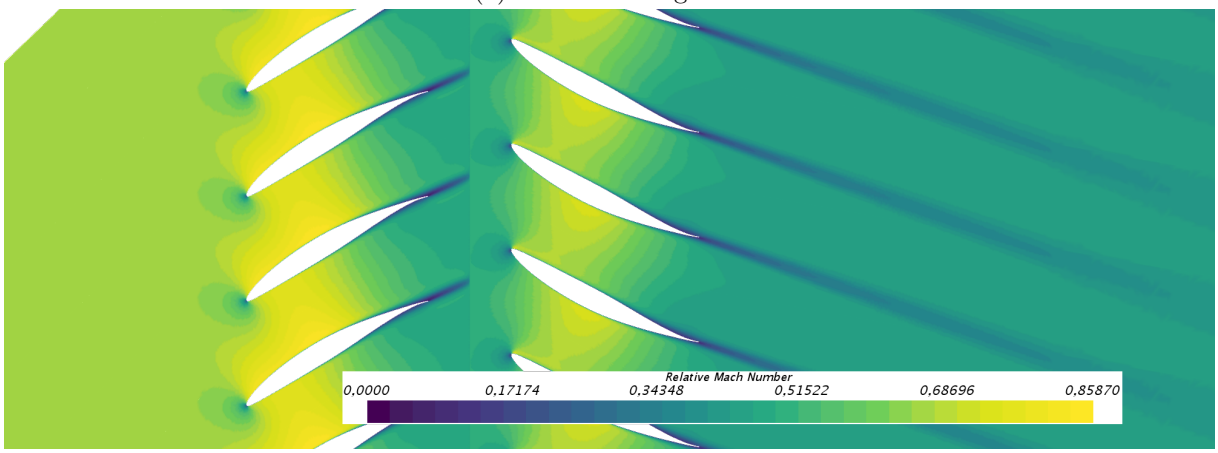
(c) Incidencia Positiva

Figura 39: Comportamiento del Flujo al Variar el Ratio de Presiones a $\dot{m}^* = 20\ kg/ms$

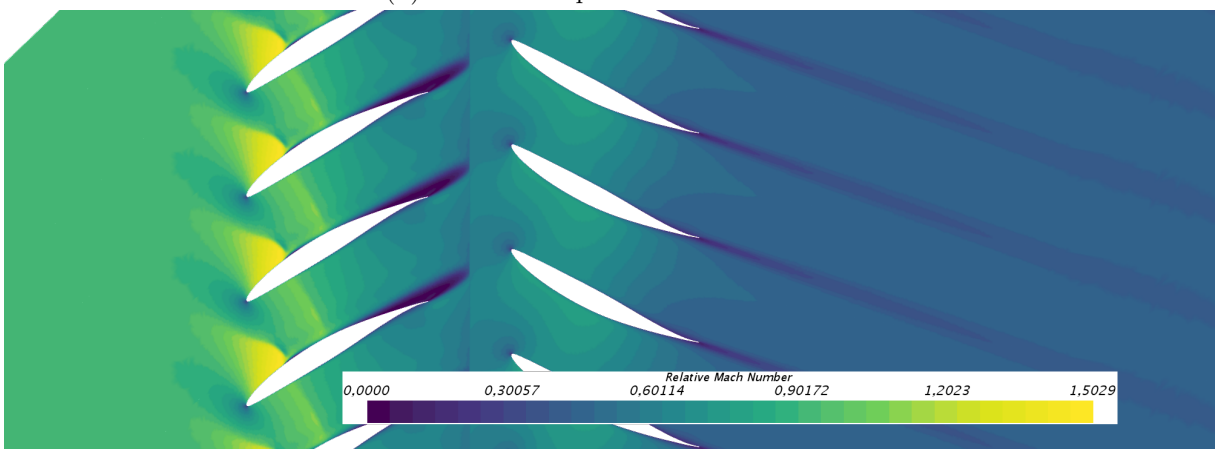
4.3.2. Variación en Ratio de Presiones a Mach Alto



(a) Incidencia Negativa



(b) Incidencia Aproximadamente Nula



(c) Incidencia Positiva

Figura 40: Comportamiento del Flujo al Variar el Ratio de Presiones a $\dot{m}^* = 41\ kg/ms$

En la Figura 39 se representa la evolución del flujo cuando, para un gasto másico corregido constante de 41 kg/ms. En los casos de incidencia negativa (Figura 40a) y incidencia aproximadamente nula (Figura 40b), no se aprecian efectos de compresibilidad muy marcados como pueden serlo las ondas de choque. Esto se debe a que, aunque si que se puede observar claramente en la imagen como el flujo se acelera hasta aproximadamente la mitad de los álabes porque el espesor de estos genera una garganta en el canal, la velocidad relativa de entrada no es

suficiente para llegar a régimen sónico. Por el contrario, cuando la incidencia es suficientemente alta (Figura 40c) hay dos fenómenos a causa del aumento en velocidad de desplazamiento del rotor que producen que se aprecien ondas de choque:

- Una mayor U implica que la velocidad relativa a la entrada del rotor sea mayor
- Una mayor U aumenta el flujo que superar una curvatura mayor, aumentándose la velocidad del flujo en el pico de succión.

Además de estos, la mayor incidencia puede cambiar de posición y/o la sección relativa[10] de la garganta (Figura 41).

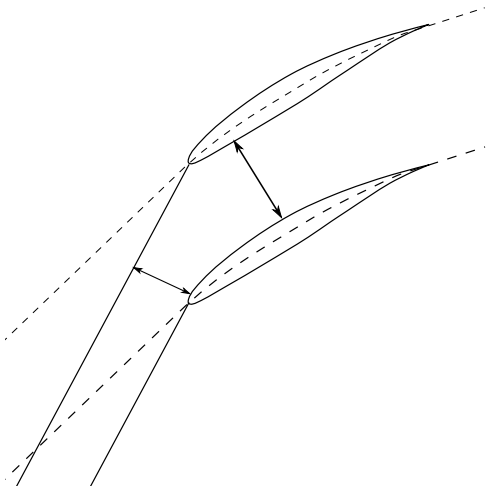


Figura 41: Reducción de Garganta al Aumentar Incidencia

Si comparamos el resultado de entrada subsónica cercana a $Mach = 1$ con el proporcionado en la asignatura de turbomáquinas vemos que el resultado de las ondas de choque coincide (Figura 42).

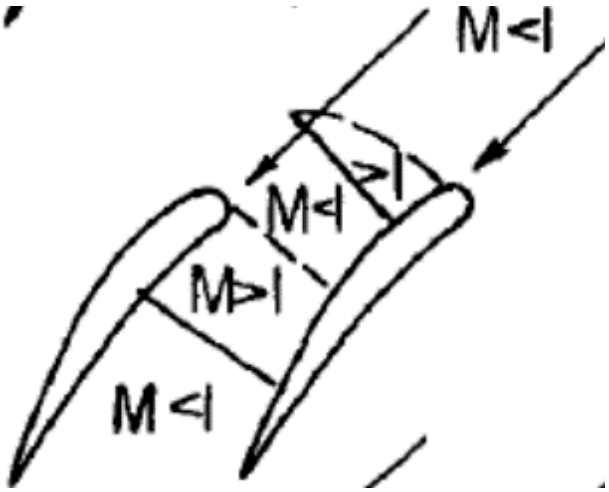
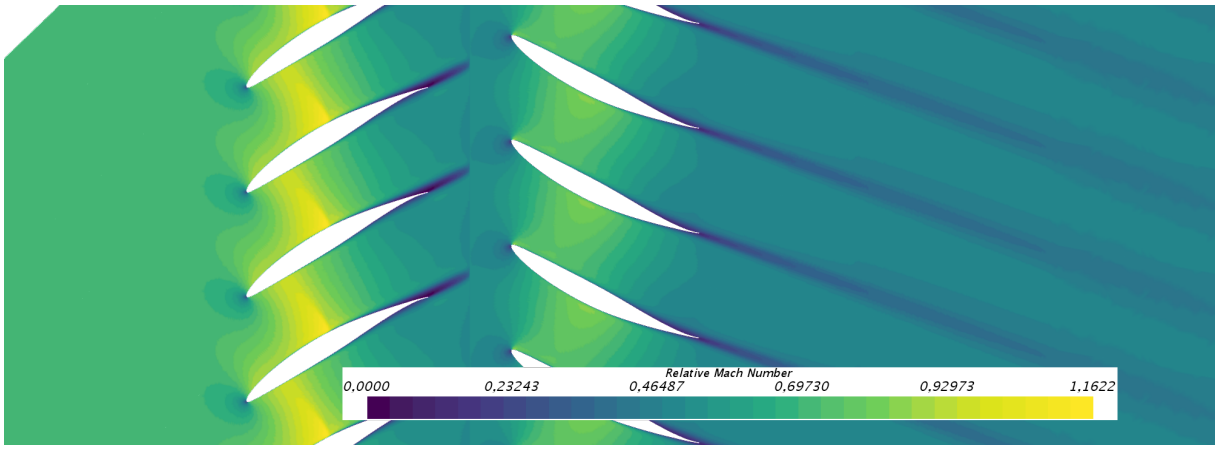


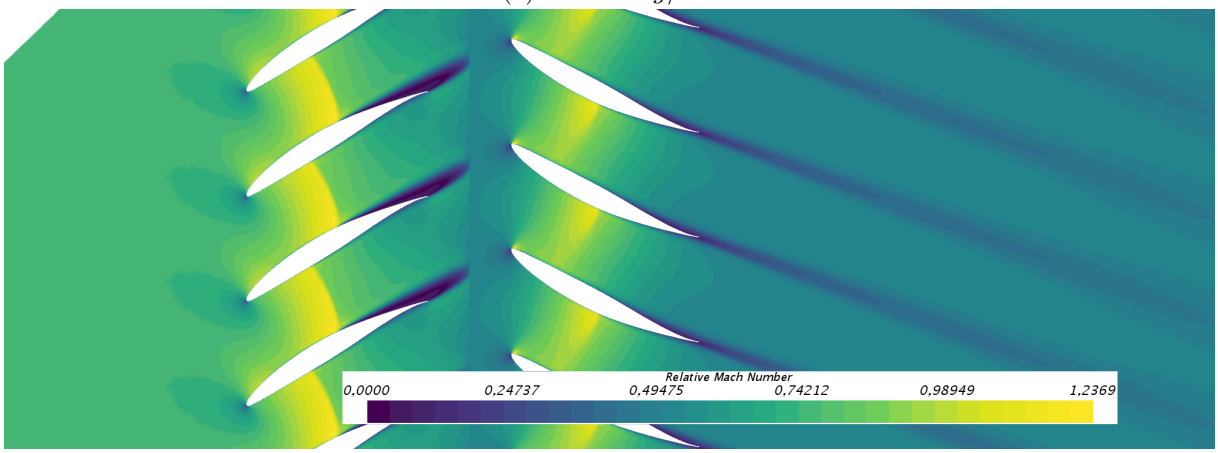
Figura 42: Comportamiento Ondas de Choque ante Entrada Subsónica cercana a Condiciones Sónicas

[10]Respecto al área de entrada perpendicular al flujo

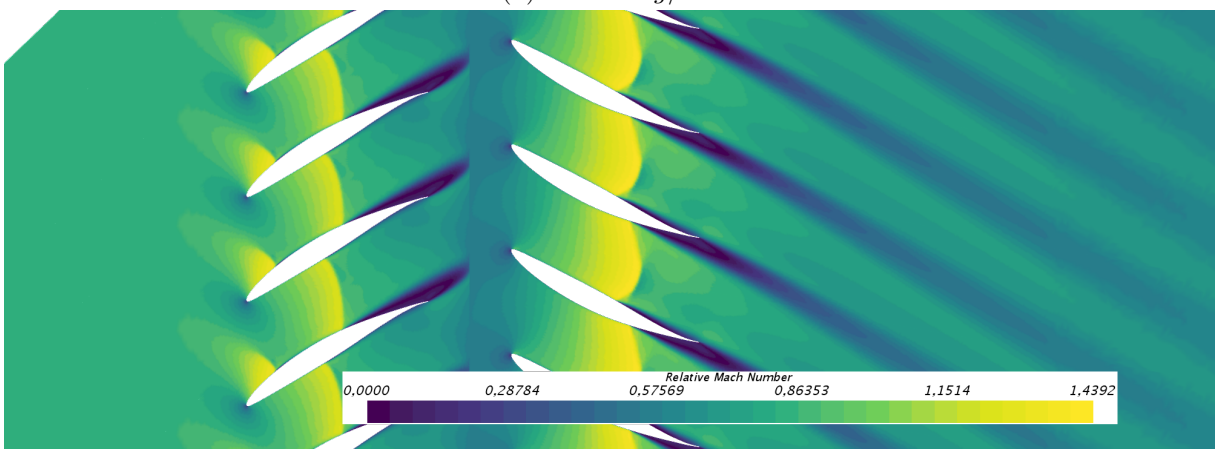
4.3.3. Variación en Gasto Másico a Incidencia Aproximadamente Nula



(a) $\dot{m} = 44\ kg/ms$



(b) $\dot{m} = 47\ kg/ms$



(c) $\dot{m} = 56\ kg/ms$

Figura 43: Comportamiento del Flujo al Variar el Gasto Másico a $i_0 \simeq 0$

En la Figura 43 podemos observar como varía el comportamiento del flujo al variar el gasto másico con incidencia casi nula. A partir del flujo con incidencia prácticamente nula en régimen subsónico alto (Figura 40b), observamos que al aumentar el gasto másico aparece una onda de choque normal en el canal del rotor. Esta va avanzando aguas abajo del canal con el aumento del gasto másico. Esta además provoca el desprendimiento del flujo al interactuar con la capa límite.

Si bien se aprecia el aumento de velocidad en el pico de succión, en las Figuras 43a y 43b la velocidad de entrada no es suficiente para alcanzar condiciones sónicas en este. En la Figura 43c, al si ser esta suficiente, el Mach en este punto es superior a 1. Se obtiene por tanto un resultado con flujo similar al mostrado en la Figura 42.

De igual forma, el flujo aguas arriba del estátor es subsónico pero la velocidad de entrada es suficiente para llegar a condiciones bloquear la garganta. Se aprecia que está totalmente bloqueada a partir de $\dot{m} = 56\ kg/ms$.

5. Comentarios Finales

Sobre los resultados obtenidos por el algoritmo con la configuración de Star-CCM+ usada en este estudios, si bien los resultados obtenidos no son lo suficientemente precisos para ser usados directamente, estos si que son representativos del problema resuelto. Es decir, el error de los resultados con respecto a un análisis experimental sería demasiado elevado para considerar estos resultados directamente. No obstante, estos si que seguirán tendencias similares a los obtenidos en el experimento, por lo que podrán ser usados en la etapa de prediseño (siempre teniendo en cuenta las limitaciones del cálculo).

Si se requiriera que estos fuesen más similares al análisis del compresor completo (sin aumentar el coste computacional), se podría incluir el efecto de los términos despreciados con las simplificaciones tomadas mediante correlaciones empíricas y/o semi-empíricas. Si se dispone de suficientes recursos, se podría haber mejorado el estudio con el análisis de todos los canales ya que esto no supondría cambios sobre el algoritmo.

Aunque el algoritmo ha cumplido con los requerimientos esperados, este requeriría ciertas mejoras para mejorar su funcionamiento. La primera mejora que se contempla es una definición más exhaustiva de las diferentes regiones del compresor para especificar de forma más exacta cuales son los límites hasta los que deseamos calcular. Otra mejora significativa sería la inclusión de cálculos transitorios para la resolución del mapa. Actualmente, está implementado un sistema que permite realizar estos pero no cambiar los modelos físicos. La mejora que aquí se presenta es incluir los cálculos transitorios en el selector de modelos físicos para poder variar entre solver segregado, solver acoplado y solver acoplado transitorio[11]. Por tanto, se definirían tres regiones dentro del problema, tal que se seleccionaría el modelo físico meno costoso (computacionalmente) que cumpliera con las restricciones de error entre parámetros.

Dado que estas mejoras incrementarían los recursos computacioneles necesarios, si se dispone de un ordenador con múltiples procesadores que pueda lanzar varios cálculos simultáneamente, se podría lanzar el algoritmo por sectores simultáneamente[12]. De esta forma, logaríamos disminuir el tiempo de cálculo de forma lineal respecto al número de sectores usados.

[11] Ya que el incluir cálculos transitorios aumenta significativamente el tiempo de cálculo, no tiene demasiado sentido alternar entre transitorio segregado y transitorio acoplado ya que la ganancia en tiempo no sería significativa

[12] Ejemplo: un caso se encargaría del barrido entre $\dot{m} \in\ [10, 20]\ kg/ms$, otro entre $\dot{m} \in\ ]20, 30]\ kg/ms$, etc

BIBLIOGRAFÍA

[1] Vaclav Cyrus. *AERODYNAMIC PERFORMANCE OF AN AXIAL COMPRESSOR STAGE WITH VARIABLE ROTOR BLADES AND VARIABLE INLET GUIDE VANES.* International Gas Turbine & Aeroengine Congress & Exhibition, 1998. URL: `https://proceedings.asmedigitalcollection.asmeS.otrogc%20oknh%200o6/l0m5/2%200S1w9%20eTedremns%20o%E2%80%94f%20U%20sJeu:%20nhtetp%202://-wJwuwn.aes%20m5,e%20.1or9g9/a8b`.

[2] ShamoonJamshed. *Using HPC for Computational Fluid Dynamics, Chapter 4.* ScienceDirect, 2015. URL: `https://doi.org/10.1016/B978-0-12-801567-4.00004-0`.

[3] Bathie William W. *Fundamentals of Gas Turbines.* Editorial John Wiley & Sons., 1996.

PRESUPUESTO

1. Introducción

La realización de este Trabajo Final de Grado titulado “Diseño y optimización de algoritmo para la generación de mapas de cascadas de compresor mediante CFD ” ha supuesto unos costes. Por una parte, se detallarán los costes parciales de todos los procesos y materiales y, por otra, el presupuesto global del trabajo.

Tanto para los costes parciales como para los globales, se debe de tener en cuenta los costes de amortización de equipos, los recursos humanos necesarios y los materiales. Además se debe tener en cuenta a todos los elementos y personas que han intervenido.

Después de calcular los costes parciales, se elabora el presupuesto global del proyecto considerado como una suma total de todos los presupuestos parciales obtenidos. Además, se contabilizará un 10 % de incremento debido a los gastos generales, un 5 % de incremento de beneficio industrial y un 21 % de incremento debido al Impuesto sobre el Valor Añadido (IVA).

2. Presupuestos Parciales

La mano de obra comprende el coste de un ingeniero doctor (Ph.D engineer), que es el Director de este proyecto, y un ingeniero superior (MSc engineer), como es el doctorando que también ha colaborado. Ambos ingenieros llevan un coste asociado de 45 €/h y 30 €/h, respectivamente. El ingeniero junior, autor del trabajo, realizó reuniones durante un tiempo de 1.25 horas a la semana durante 4 meses con el ingeniero doctor, y 2 horas a la semana durante 4 meses con el ingeniero superior. El ingeniero junior tiene un salario asociado de 20 €/h con un régimen de trabajo de 6h.

Por otro lado, hay que tener en cuenta el equipo empleado por el autor del proyecto, para llevar a cabo las simulaciones realizadas.

2.1. Mano de obra

En las reuniones en las que se trató el trabajo, participó tanto el director, cómo el doctorando. Pero no tendremos en cuenta sólo los recursos humanos, si no tambien los recursos informaticos utilizados. El desarrollo y depuración del algoritmo ocupan el mayor volumen de esta parte del presupuesto, contabilizado en las horas del ingeniero junior. El coste se detalla en Tabla 3.

No se tendrá en cuenta, en esta parte, la licencia de STAR-CCM+ utilizada por todos ellos.

Tabla 3: Resumen Costes Asociados a Mano de Obra

Categoria	€/h	horas/semana	Horas Total	Coste Total (€)
Ph.D engineer	45.00	1.25	21.43	964.35
Msc engineer	30.00	2.00	34.29	1028.70
BSc engineer	20.00	30.00	514.29	10285.80
				12278.85

2.2. Adquisición y Amortización de los Equipos

Los costes de amortización de los equipos utilizados en la simulación y en el post-procesado quedan recogidos en la Tabla 4. Se muestran las herramientas informáticas utilizadas. El

programa más utilizado, ha sido el software STAR-CCM+, cuya licencia de trabajo se expone a más adelante. Además, se incluyen otros programas que se han utilizado para la realización del trabajo.

Tabla 4: Costes de amortización de los programas utilizados

Programa		
Office 365	240.00	€
Matlab	800.00	€
Periodo de amortización	1.00	(año)
Tiempo de amortización	4.00	(mes)
Subtotal	346.67	

A continuación, se detalla el coste mismo de amortización en la Tabla 5.

Tabla 5: Coste de amortización Equipos

Descripción		
Coste de equipo 1	1100.00	€
Coste de equipo 2	5000.00	€
Periodo de amortización	4.00	(año)
Tiempo de amortización	4.00	(mes)
Coste/mes	127.08	€
Subtotal (€)	508.32	€

2.3. Gastos Generales e IVA

A todos los costes parciales hay que añadirles los costes generales (electricidad, secretaria, etc.) que no se ven reflejados, estos se tomarán como un porcentaje (10 %) de la suma de los costes parciales.

Además, habrá que tener en cuenta el impuesto sobre el valor añadido (I.V.A.) en cada uno de los costes parciales, excepto en el de amortización de equipos. Estos presupuestos se recogen en la Tabla 6 y en la Tabla 7.

Tabla 6: Costes Parciales + IVA

Descripción	**Coste Parcial (€)**	**I.V.A 21 % (€)**	**Coste Parcial I.V.A 21 % (€)**
Mano de Obra	12278.85	2578.56	14857.41
Adquisición Licencia CFD	4500.00	945.00	5445.00
Amortización de Equipos	508.32	0.00	508.32
Amortización de Programas	346.67	0.00	346.67
Subtotal	17633.84	3523.56	21157.40

Tabla 7: Costes Globales

Descripción	Parcial + I.V.A (€)	Generales (€)	Coste Total (€)
Parcial + I.V.A	21157.40	2115.74	23273.14

3. Presupuesto Global

Con el fin de calcular el presupuesto global, es necesario aplicar el beneficio industrial buscado por la empresa al realizar el presente estudio. Como se indicaba en el apartado de introducción, este beneficio industrial será del 5 % respecto del coste total que le suponga a la empresa. En la tabla se presenta la información.

Tabla 8: Presupuesto Proyecto

Coste Total (€)	Beneficio Industrial (€)	Importe Total (€)
23273.14	1163.66	25134.99

El coste total del Trabajo Fin de Grado: " Diseño y optimización de algoritmo para la generación de mapas de cascadas de compresor mediante CFD ", asciende a la cantidad de:

25134.99 € (EUROS)

 # Veinticinco mil ciento treinta y cuatro euros con noventa y nueve céntimos #

PLIEGO DE CONCICIONES

1. Hardware

Las características de sistema empleado son las siguientes:

- **Fabricante:** MSI
- **Modelo:** GE60 2PE Apache Pro
- **Procesador:** Intel(R) Core(TM) i7−4720HQ CPU @ 2.60GHz 2.60GHz
- **Memoria:** 8GB DDR3
- **Almacenamiento:** HDD SATA 1TB
- **Tarjeta gráfica integrada:** Intel(R)HD Graphics 4600
- **Tarjeta gráfica dedicada:** NVIDIA GTX860M
- **Sistema operativo:** Windows 10 64bits

Adicionalmente, se ha usado una estación de trabajo con las siguientes características:

- **Nombre del Equipo:** CMT505
- **Procesador:** Intel(R) Xeon(R) CPU E5-2640 @ 2.5 GHz, 24 CPUs.
- **Memoria:** 128 GB
- **Almacenamiento:** 8 TB
- **Tarjeta gráfica dedicada:** NVIDIA Quadro 4000
- **Sistema operativo:** Windows 1.

2. Software

- **STAR-CCM+:** Software CFD comercial, proporcionado por el Departamento. En este se han realizado las simulaciones de los diversos puntos de funcionamiento del compresor. Se ha usado la versión 12.06.011-R8.
- **MATLAB:** Software para cálculo numérico con un entorno de desarrollo integrado así como lenguaje de programación propio. Ha sido utilizado principalmente como herramienta de postproceso de datos, así como para la programación de la GUI para mostrar los resultados. Se ha usado la versión 2018b.
- **NetBeans:** Entorno de desarrollo integrado. Se ha usado para la programación del código correspondiente a las macros que conformaban el algoritmo.
- **LaTeX:** Programa ofimático usado para la redacción de este informe. Se han usado MikTex y TexMaker como compilador e interfaz.

3. Condiciones Generales

Aquellos lugares de trabajos en los que se utilice un equipo con pantalla está relacionado con riesgos como problemas oculares y visión, así como posturales y estrés. Para reducir estos, se debe tener en cuenta el Real Decreteo 488/1997 del 14 de abril. Este establece las condiciones de seguridad mínima (Ley 31/1997). Los factores más relevantes son:

- Tiempo de visualización de la pantalla.
- Complejidad de la tarea.
- Tipo de visualización: continua o discontinua

Se debe asegurar la seguridad del entorno de trabajo respecto a golpes, accidentes eléctricos, etc. Además, es necesario que el lugar de trabajo proporcione las condiciones adecuadas con respecto a luminosidad, espacio, ergonomía, temperatura, ruido y reducción de movimientos innecesarios y repetitivos.

ANEXO

ANEXO I: Extensión del Dominio a Múltiples Canales

Para aumentar el número de canales a simular, debemos crear un patrón lineal sobre el cuerpo que ya tenemos creado (Figura 44).

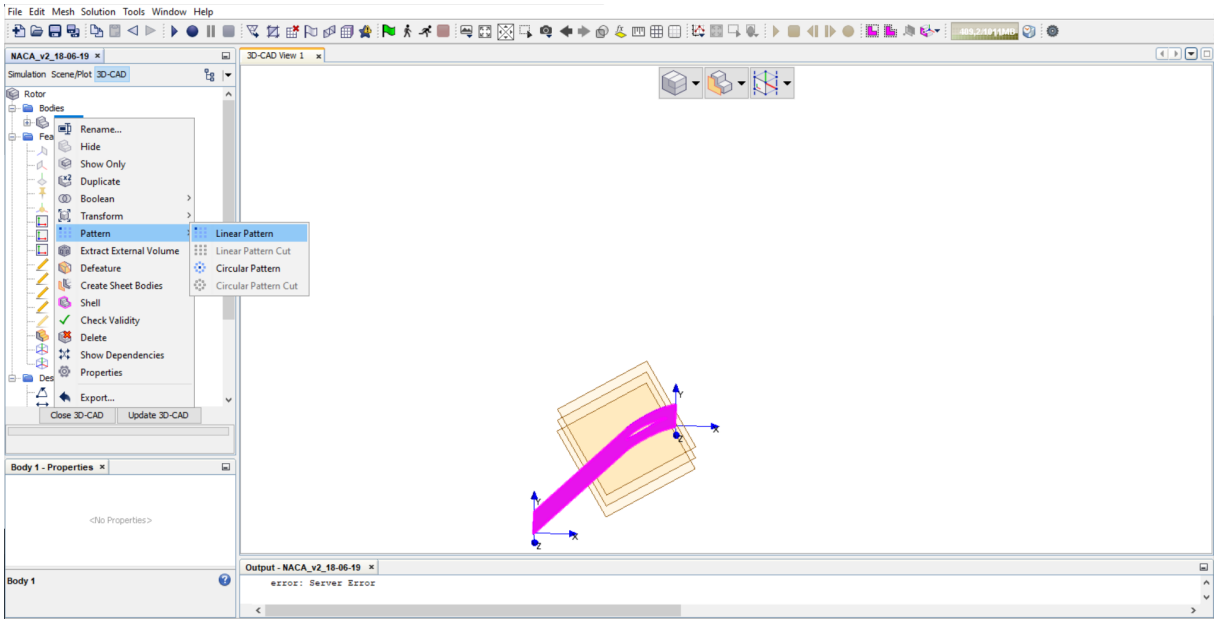


Figura 44: “Linear Pattern”

Una vez entramos en la configuración de este patrón, solo se requiere cambiar el valor por defecto por el paso del canal y escribir en nombre de repeticiones necesarias (Figura 45) y pulsar “OK”.

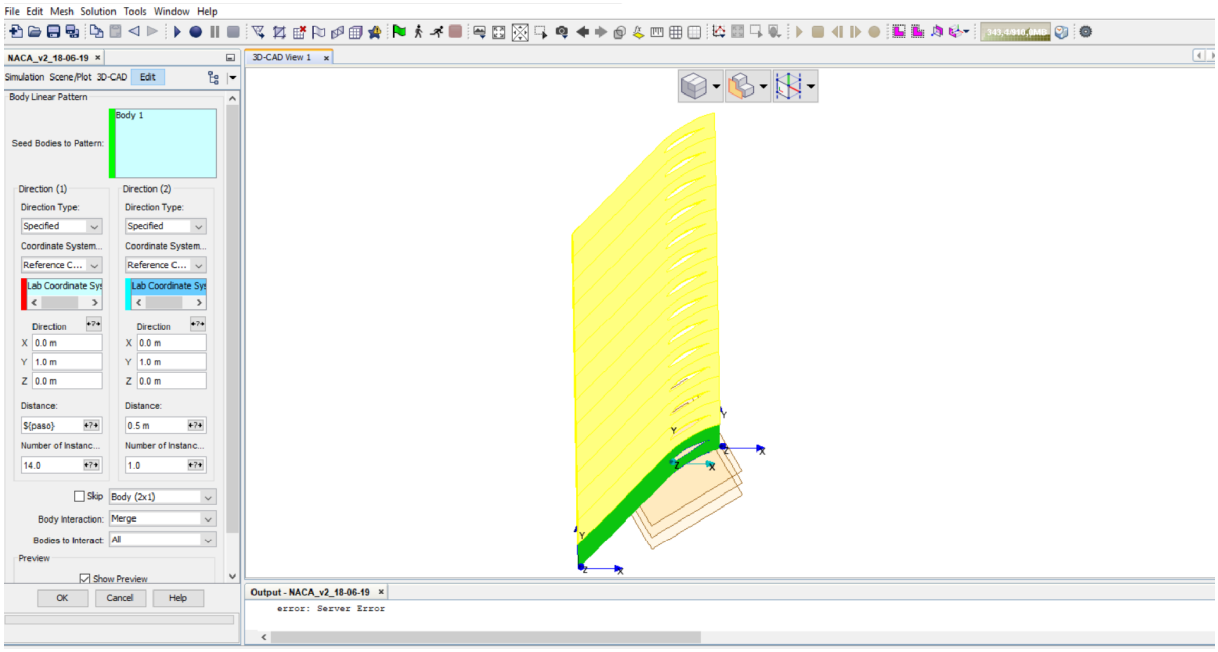


Figura 45: Configuración “Linear Pattern”

Una vez hecho esto, solo queda ejecutar las operaciones de mallado y translación. Puede resultar interesante nombrar uno de los perfiles con otro nombre para analizar la evolución temporal sobre este. Para esto, se debe crear otra condición de contorno en la región a la que pertenece este objeto y asignarle esta cara.

ANEXO II: Código Macros

1. Estimador

```
1
2  package macro;
3
4  import java.util.*;
5  import java.util.Arrays;
6  import java.io.*;
7  import java.nio.*;
8  import star.common.*;
9  import star.base.neo.*;
10 import star.base.report.*;
11 import star.flow.*;
12 import star.segregatedflow.*;
13 import star.energy.*;
14
15 /**
16  *
17  * @author JAVI_PC
18  */
19 public class size_finder_v extends StarMacro{
20
21     public void execute() {
22     execute0();
23   }
24     private void execute0() {
25         //Variable Definitions
26     boolean update_state;
27     double v_base0;
28     double iter_rotorvelocity;
29     double m_base0;
30     double iter_massflow;
31     double step_massflow;
32     double sign_rotorvelocity;
33     double sign_massflow;
34
35     double efficiency;      double[] d_efficiency = new double[0];
36     double massflow;        double[] d_massflow = new double[0];
37     double rotorvelocity;   double[] d_rotorvelocity = new double[0];
38     double maxRelMach;          double[] d_maxRelMach = new double[0];
39     double[] d_RP = new double[0];
40 double[] solEfficiency = new double[0];
41 double[] solRotorvelocity = new double[0];
42 //simulation_0 variable definitions
43     Simulation simulation_0 = getActiveSimulation();
44
45     Solution solution_0 = simulation_0.getSolution();
46
47     ScalarGlobalParameter parameterMassFlow =
48       ((ScalarGlobalParameter) simulation_0.
49       get(GlobalParameterManager.class).
50       getObject("parameterMassFlow"));
51
52     ScalarGlobalParameter parameterRotorVelocity =
53       ((ScalarGlobalParameter) simulation_0.
54       get(GlobalParameterManager.class).
```

```
      getObject("parameterRotorVelocity"));


    MaxReport reportMaxRelMach =
            ((MaxReport) simulation_0.
            getReportManager().getReport("reportMaxRelMach"));

    LogicUpdateEvent logicUpdateEvent_0 =
      ((LogicUpdateEvent) simulation_0.
      getUpdateEventManager().
      getUpdateEvent("updateEventConvergido"));


    IsentropicEfficiencyReport reportEfficiency =
      ((IsentropicEfficiencyReport) simulation_0.
      getReportManager().getReport("reportEfficiency"));

    ExpressionReport reportMassFlow =
      ((ExpressionReport) simulation_0.
      getReportManager().getReport("reportMassFlow"));

    ExpressionReport reportRotorVelocity =
      ((ExpressionReport) simulation_0.
      getReportManager().
      getReport("reportRotorVelocity"));

    AreaAverageReport reportPTinlet =
            ((AreaAverageReport) simulation_0.
            getReportManager().
            getReport("reportPTinlet"));

    AreaAverageReport reportPToutlet =
            ((AreaAverageReport) simulation_0.
            getReportManager().
            getReport("reportPToutlet"));

//Initialization
    clear("all");
    solution_0.initializeSolution();
     //caso = 0;

    v_base0 = 33;

    m_base0 = 8;

    iter_rotorvelocity = v_base0;
    iter_massflow = m_base0;


    parameterMassFlow.
    getQuantity().setValue(iter_massflow);
    parameterRotorVelocity.
    getQuantity().setValue(iter_rotorvelocity);

    double effline;

    double error;
    double corr;
```

```
    double gradient;
    int iter = 2000;

    for(int caso = 0; caso < 2; caso++){
        d_massflow = new double[0];
        d_rotorvelocity = new double[0];
        d_efficiency = new double[0];
        d_maxRelMach = new double[0];
        iter_massflow = m_base0;
        iter_rotorvelocity = v_base0;
        sign_rotorvelocity = vrotSigno(caso);
        sign_massflow = mflowSigno(caso);
        parameterMassFlow.getQuantity().
        setValue(iter_massflow);
        parameterRotorVelocity.getQuantity().
        setValue(iter_rotorvelocity);
        switch(caso){
            case 0:
                step_massflow = 3;
                effline = 0.9;
                break;
            default: //case 1:
                step_massflow = 3;
                effline = 0.7;
                break;
        }
        clear("all");
        solution_0.initializeSolution();
        massflowLoop: while(true){
            error = 2;
            parameterMassFlow.getQuantity().
            setValue(iter_massflow);
            solEfficiency = new double[0];
            solRotorvelocity = new double[0];
            velocityLoop: while(true){
                parameterRotorVelocity.getQuantity().
                setValue(iter_rotorvelocity);
                try {
                    simulation_0.getSimulationIterator().step(150);
                    simulation_0.getSimulationIterator().run(iter);
                    simulation_0.println("\n\n\n Caso:" + caso + "\n\n\n");
                }
                catch (Exception iOException){
                    simulation_0.println("Caso: " + caso);
                    simulation_0.println(iOException + "\n");
                    break massflowLoop;
                }
                update_state = logicUpdateEvent_0.getEvaluatedState();
                if(update_state){
                    efficiency = reportEfficiency.getReportMonitorValue();
                    massflow = reportMassFlow.getReportMonitorValue();
                    rotorvelocity = reportRotorVelocity.
                    getReportMonitorValue();
                    maxRelMach = reportMaxRelMach.getReportMonitorValue();

                    error = efficiency - effline;

                    solEfficiency = join(solEfficiency,efficiency);
```

```
                    solRotorvelocity = join(solRotorvelocity,rotorvelocity);
                    if(Math.abs(error) < 0.02){
                        d_efficiency = join(d_efficiency,efficiency);
                        d_massflow = join(d_massflow,massflow);
                        d_rotorvelocity =
                         join(d_rotorvelocity,rotorvelocity);
                        d_maxRelMach = join(d_maxRelMach,maxRelMach);
                        d_RP = join(d_RP,reportPToutlet.
                        getReportMonitorValue()
                        /reportPTinlet.getReportMonitorValue());
                        iter_massflow = iter_massflow +
                         sign_massflow*step_massflow;
                        switch(caso){
                            case 0:
                                write("limiteSuperior.csv",
                                d_massflow,d_rotorvelocity,
                                d_efficiency,d_maxRelMach,d_RP);
                                break;
                            case 1:
                                write("limiteInferior.csv",
                                d_massflow,d_rotorvelocity,
                                d_efficiency,d_maxRelMach,d_RP);
                                break;
                        }
                        iter_rotorvelocity = rotorvelocity*iter_massflow
                        /massflow;
                        if(maxRelMach > 1.3){
                            break massflowLoop;
                        }
                        else{
                            break velocityLoop;
                        }
                    }
                    else{

                        corr = rotorvelocity*0.1;

                        if (solEfficiency.length > 1){
                            gradient = (solEfficiency[solEfficiency.length-1]
                             - solEfficiency[solEfficiency.length-2])/
                            (solRotorvelocity[solRotorvelocity.length-1] -
                             solRotorvelocity[solRotorvelocity.length-2]);

                            double corr2 = 0.5*Math.abs(error/gradient);
                            corr = Math.min(corr,corr2);
                        }


                        iter_rotorvelocity +=
                         sign_rotorvelocity*Math.signum(error)*corr;
                    }
                }
                else{
                    // Hemos llegado a la derecha, bajar y cambiar de caso
                    break massflowLoop;
                }
            }
        }
```

```
229     }
230 }
231
232
233     private void clear(String clearType){
234
235     Simulation simulation_0 = getActiveSimulation();
236     Solution solution_0 = simulation_0.getSolution();
237
238     switch(clearType){
239         case "history":
240             solution_0.clearSolution(Solution.Clear.History);
241             break;
242         case "solution":
243             solution_0.clearSolution(Solution.Clear.Fields,
244              Solution.Clear.LagrangianDem);
245         default:
246             solution_0.clearSolution(Solution.Clear.History,
247              Solution.Clear.Fields, Solution.Clear.LagrangianDem);
248             break;
249     }
250 }
251
252
253     private int vrotSigno(int caso){
254
255         int temp = 0;
256         // caso 0: barriendo la zona superior derecha, izquierda a derecha
257         // caso 1: barriendo la zona inferior derecha, izquierda a derecha
258         switch (caso) {
259             case 0:
260                 temp = +1;
261                 break;
262             case 1:
263                 temp = -1;
264                 break;
265         }
266         return temp;
267     }
268
269     private int mflowSigno(int caso){
270         // caso 0: barriendo la zona superior derecha, izquierda a derecha
271         // caso 1: barriendo la zona inferior derecha, izquierda a derecha
272
273         int temp = 0;
274
275         switch (caso) {
276             case 0:
277                 temp = +1;
278                 break;
279             case 1:
280                 temp = +1;
281                 break;
282         }
283         return temp;
284     }
285
286     // Mass_Flow,Rotor_Velocity,Efficiency
```

```
287     private void write(String fileName, double[] massflow, double[] vrot,
288      double[] eff, double[] mach, double[] rp){
289         BufferedWriter bwout;
290         try{
291             bwout = new BufferedWriter(new FileWriter(
292             resolvePath(fileName)));
293             bwout.write("Mass_Flow,Rotor_Velocity,
294             Efficiency,MaxRelMach,Pi_c\n");
295             for(int i=0;i<eff.length;i++){
296                 bwout.write(Double.toString(massflow[i])+","+
297                 Double.toString(vrot[i])+","+Double.toString(eff[i]) + ","
298                  +Double.toString(mach[i])+","
299                   +Double.toString(rp[i]) +"\n");
300             }
301             bwout.close();
302         }
303         catch(IOException ioException){
304 
305         }
306     }
307 
308     private double[] join(double[] base, double addto){
309     double[] vector;    vector = new double[base.length+1];
310     for(int i=0;i<base.length;i++){
311         vector[i]= base[i];
312     }
313     vector[vector.length-1] = addto;
314     return vector;
315     }
316 }
```

2. Mapeado Estacionario

```
1 
2  package macro;
3 
4  import java.util.*;
5  import java.util.Arrays;
6  import java.io.*;
7  import java.nio.*;
8  import star.common.*;
9  import star.base.neo.*;
10 import star.base.report.*;
11 import star.flow.*;
12 import star.segregatedflow.*;
13 import star.energy.*;
14 import star.motion.*;
15 import star.vis.*;
16 
17 import star.segregatedenergy.*;
18 import star.coupledflow.*;
19 
20 import java.io.BufferedReader;
21 import java.io.FileReader;
22 import java.io.IOException;
23 
24 import java.io.File;
```

```
25
26  import javax.swing.JFileChooser;
27  import javax.swing.filechooser.FileSystemView;
28
29  //update event getEvaluatedState()
30  public class algoritm_0_v1_2_seg extends StarMacro {
31
32  String filePath = "";
33
34  String physicsModel = "segregated";
35  String imagesPath = filePath + "imagenes_" + physicsModel + "/";
36  String plotsPath = filePath + "plots_" + physicsModel + "/";
37  int mapeado = 2;
38  //double stepRotorVelocity = 5;
39  int selector = 0;
40  double A, B;
41
42  public void execute() {
43
44      double inf_slope = slope(0, 1, "limiteInferior.csv");
45      double sup_slope = slope(0, 1, "limiteSuperior.csv");
46
47      execute0(inf_slope,sup_slope);
48
49  }
50
51      private void execute0(double inf_slope, double sup_slope) {
52
53  /*////////////////////////////////////////////////////////////////////////////
54                      Definición de variables
55  ////////////////////////////////////////////////////////////////////////////*/
56          boolean converged;
57
58          double iterRotorVelocity;
59
60          double iterMassFlow;
61
62          double stepMassFlow;
63          double calado = 45*3.1415/180;
64          int signMassFlow;
65
66
67          double efficiency, massFlow, maxRelMach, PTinlet, PToutlet, TTinlet;
68          double TToutlet, rotorVelocity, velocityHinlet, velocityHoutlet;
69          double velocityVoutlet, vForceRotor, bladePass;
70          double Pinlet, densityInlet;
71
72          data dataEfficiency = new data();
73          data dataMassFlow = new data();
74          data dataMaxRelMach = new data();
75          data dataPTinlet = new data();
76          data dataPinlet = new data();
77          data dataPToutlet = new data();
78          data dataTTinlet = new data();
79          data dataTToutlet = new data();
80          data dataRotorVelocity = new data();
81          data dataVelocityHinlet = new data();
82          data dataVelocityHoutlet = new data();
```

```
83          data dataVelocityVoutlet = new data();
84          data dataVForceRotor = new data();
85          data dataBladePass = new data();
86          data dataIndex = new data();
87          data dataNumeroImagen = new data();
88          data dataDensityInlet = new data();
89
90
91
92  /*////////////////////////////////////////////////////////////////////
93                  Obtención de reports, update events
94                   y parámetros de la simulación
95  ////////////////////////////////////////////////////////////////////*/
96          Simulation simulation_0 = getActiveSimulation();
97
98          Solution solution_0 = simulation_0.getSolution();
99
100         // Definir parametros
101         ScalarGlobalParameter parameterMassFlow =
102             ((ScalarGlobalParameter) simulation_0.
103             get(GlobalParameterManager.class).
104             getObject("parameterMassFlow"));
105
106         ScalarGlobalParameter parameterRotorVelocity =
107             ((ScalarGlobalParameter) simulation_0.
108             get(GlobalParameterManager.class).
109             getObject("parameterRotorVelocity"));
110
111         // Definir update events
112         LogicUpdateEvent updateEventConverged =
113             ((LogicUpdateEvent) simulation_0.
114             getUpdateEventManager().
115             getUpdateEvent("updateEventConvergido"));
116
117         // Definir reports
118         IsentropicEfficiencyReport reportEfficiency =
119             ((IsentropicEfficiencyReport) simulation_0.getReportManager().
120             getReport("reportEfficiency"));
121
122         ExpressionReport reportMassFlow =
123             ((ExpressionReport) simulation_0.getReportManager().
124             getReport("reportMassFlow"));
125
126         AreaAverageReport reportPTinlet =
127             ((AreaAverageReport) simulation_0.getReportManager().
128             getReport("reportPTinlet"));
129
130         AreaAverageReport reportPinlet =
131             ((AreaAverageReport) simulation_0.getReportManager().
132             getReport("reportPinlet"));
133
134         AreaAverageReport reportPToutlet =
135             ((AreaAverageReport) simulation_0.getReportManager().
136             getReport("reportPToutlet"));
137
138         MassFlowAverageReport reportTTinlet =
139             ((MassFlowAverageReport) simulation_0.getReportManager().
140             getReport("reportTTinlet"));
```

```
141
142         MassFlowAverageReport reportDensityInlet =
143             ((MassFlowAverageReport) simulation_0.getReportManager().
144             getReport("reportDensityInlet"));
145
146         MassFlowAverageReport reportTToutlet =
147             ((MassFlowAverageReport) simulation_0.getReportManager().
148             getReport("reportTToutlet"));
149
150         ExpressionReport reportRotorVelocity =
151             ((ExpressionReport) simulation_0.getReportManager().
152             getReport("reportRotorVelocity"));
153
154         MassFlowAverageReport reportVelocityHinlet =
155             ((MassFlowAverageReport) simulation_0.getReportManager().
156             getReport("reportVelocityHinlet"));
157
158         MassFlowAverageReport reportVelocityHoutlet =
159             ((MassFlowAverageReport) simulation_0.getReportManager().
160             getReport("reportVelocityHoutlet"));
161
162         MassFlowAverageReport reportVelocityVoutlet =
163             ((MassFlowAverageReport) simulation_0.getReportManager().
164             getReport("reportVelocityVoutlet"));
165
166         ForceReport reportVForceRotor =
167             ((ForceReport) simulation_0.getReportManager().
168             getReport("reportVForceRotor"));
169
170         ExpressionReport reportTimeStep =
171             ((ExpressionReport) simulation_0.getReportManager().
172             getReport("reportTimeStep"));
173
174         ExpressionReport reportBladePass =
175             ((ExpressionReport) simulation_0.getReportManager().
176             getReport("reportBladePass"));
177
178         MaxReport reportMaxRelMach =
179             ((MaxReport) simulation_0.getReportManager().
180             getReport("reportMaxRelMach"));
181
182         // Definir Monitor Plots
183         MonitorPlot plotITMonitorMassFlow =
184             ((MonitorPlot) simulation_0.getPlotManager().
185             getPlot("plotITMonitorMassFlow"));
186
187         MonitorPlot plotITMonitorRotorVelocity =
188             ((MonitorPlot) simulation_0.getPlotManager().
189             getPlot("plotITMonitorRotorVelocity"));
190
191         MonitorPlot plotITMonitorEfficiency =
192             ((MonitorPlot) simulation_0.getPlotManager().
193             getPlot("plotITMonitorEfficiency"));
194
195         MonitorPlot plotITMonitorPTinlet =
196             ((MonitorPlot) simulation_0.getPlotManager().
197             getPlot("plotITMonitorPTinlet"));
198
```

```
199         MonitorPlot plotITMonitorVForceRotor =
200             ((MonitorPlot) simulation_0.getPlotManager().
201             getPlot("plotITMonitorVForceRotor"));
202 
203 /*////////////////////////////////////////////////////////////////
204                     Inicialización
205 ////////////////////////////////////////////////////////////////*/
206         simulation_0.println(inf_slope + "    " + sup_slope);
207 
208         clear("all");
209 
210         solution_0.initializeSolution();
211 
212         setPhysicsModel(physicsModel);
213 
214         iterMassFlow = 8;
215 
216         stepMassFlow = 3;
217 
218         signMassFlow = 1;
219 
220         iterRotorVelocity = 30;
221         parameterRotorVelocity.getQuantity().setValue(iterRotorVelocity);
222         parameterMassFlow.getQuantity().setValue(iterMassFlow);
223 
224 
225 
226         double indexStep = 0.2;
227         simulation_0.println(indexStep);
228         double index = -indexStep;
229 
230         double pi = Math.PI;
231 
232 
233 
234 
235 /*////////////////////////////////////////////////////////////////
236                     Inicialización
237 ////////////////////////////////////////////////////////////////*/
238 
239         int iter = 2000;
240         efficiency = 1;
241         converged = true;
242         double numeroImagen = 1;
243 
244     loopMassflow: while(true){
245         double vinf = inf_slope*iterMassFlow;
246         double vsup = sup_slope*iterMassFlow;
247         A = (vinf + vsup)/2;
248         B = Math.abs(vinf - vsup)/2;
249 
250 
251         loopCaso: for(int vrotCaso = 0; vrotCaso < 2; vrotCaso++){
252             indexStep = -indexStep;
253             switch(vrotCaso){
254                 case 0:
255                     index = 0;
256                     break;
```

```
                case 1:
                    index = -indexStep;
                    break;
            }
            loopVrot: while(true){
                /* A partir de los datos obtenidos en el estimador de
                tamaño de mapa, definimos una velocidad inferior y superior
                de referencia. De esta forma, mediante una distribución seno
                podemos obtener un refinamiento mayor en los bordes del mapa*/

                index += indexStep;

                iterRotorVelocity = tipoMapeado(iterMassFlow, mapeado);

                simulation_0.println("\n\n\n Estamos en CASO: " + vrotCaso);
                simulation_0.println("indice: " + index);
                simulation_0.println("massflow: " + iterMassFlow);
                simulation_0.
                println("rotorvelocity: " + iterRotorVelocity + "\n\n\n");

                parameterRotorVelocity.getQuantity().
                setValue(iterRotorVelocity);


                /* Para evitar que el algoritmo pare por un "float point"
                (o errores similares), lanzamos las iteraciones con una
                sentencia try/catch. En caso de catch, limpiamos la solución
                y cambiamos la zona que estamos mapeando
                (mitad superior/inferior)*/
                if((velocityHinlet > 160) && selector == 1){
                    physicsModel = "coupled";
                }
                else{
                    physicsModel = "segregated";
                }
                try{
                    simulation_0.getSimulationIterator().step(150);
                    simulation_0.getSimulationIterator().run(iter);
                }
                catch(Exception e){
                    // Temporal, actualizar para aÃ±adir en ifdiverged bien
                    clear("solution");
                    solution_0.initializeSolution();
                    break loopVrot;
                }
                /* Para reducir tiempo de cÃ¡lculo (y evitar tener una larga
                sentencia de condicionales), tenemos un stopping criteria
                y un update events equivalentes). El primero mÃ¡s
                restrictivo que el segundo, para asegurar que se
                haya activado. Esto es asÃ porque comprobar un update
                events es mÃ¡s sencillo que comprobar una sentencia de
                && de stopping criteria en java*/
                converged = updateEventConverged.getEvaluatedState();
                if(converged){
                    // Obtener datos del cÃ¡lculo
                    efficiency = reportEfficiency.getReportMonitorValue();
                    massFlow = reportMassFlow.getReportMonitorValue();
                    maxRelMach = reportMaxRelMach.getReportMonitorValue();
```

```
                    PTinlet = reportPTinlet.getReportMonitorValue();
                    Pinlet = reportPinlet.getReportMonitorValue();
                    PToutlet = reportPToutlet.getReportMonitorValue();
                    TTinlet = reportTTinlet.getReportMonitorValue();
                    TToutlet = reportTToutlet.getReportMonitorValue();
                    rotorVelocity =
                    reportRotorVelocity.getReportMonitorValue();
                    velocityHinlet =
                     reportVelocityHinlet.getReportMonitorValue();
                    velocityHoutlet =
                     reportVelocityHoutlet.getReportMonitorValue();
                    velocityVoutlet =
                     reportVelocityVoutlet.getReportMonitorValue();
                    vForceRotor = reportVForceRotor.getReportMonitorValue();
                    densityInlet =
                    reportDensityInlet.getReportMonitorValue();
                    simulation_0.
                    println("\n\n\n Resultados obtenidos\n\n\n");
                    switch(physicsModel){
                        case "unsteadySegregated":
                        case "unsteadyCoupled":
                            bladePass =
                             reportBladePass.getReportMonitorValue();
                            break;
                        default:
                            bladePass = -1;
                            break;
                    }
                    dataEfficiency.add(efficiency);
                    dataMassFlow.add(massFlow);
                    dataMaxRelMach.add(maxRelMach);
                    dataPTinlet.add(PTinlet);
                    dataPinlet.add(Pinlet);
                    dataPToutlet.add(PToutlet);
                    dataTTinlet.add(TTinlet);
                    dataTToutlet.add(TToutlet);
                    dataRotorVelocity.add(rotorVelocity);
                    dataVelocityHinlet.add(velocityHinlet);
                    dataVelocityHoutlet.add(velocityHoutlet);
                    dataVelocityVoutlet.add(velocityVoutlet);
                    dataVForceRotor.add(vForceRotor);
                    dataBladePass.add(bladePass);
                    dataIndex.add(Math.round(index/Math.abs(indexStep)));
                    dataNumeroImagen.add(numeroImagen);
                    dataDensityInlet.add(densityInlet);
                    String cabecera = "massFlow,rotorVelocity,
                    PTinlet,Pinlet,PToutlet,efficiency,"
                    "TTinlet,TToutlet,maxRelMach,velocityHinlet,"
                            + "velocityHoutlet,velocityVoutlet,vForceRotor,"
                            "Densityinlet,bladePass,index,numeroImagen\n";
                    write("mapaReports" + physicsModel + ".csv", cabecera,
                        dataMassFlow.getValues(),
                        dataRotorVelocity.getValues(),
                        dataPTinlet.getValues(),
                        dataPinlet.getValues(),
                        dataPToutlet.getValues(),
                        dataEfficiency.getValues(),
                        dataTTinlet.getValues(),
```

```
            dataTToutlet.getValues(),
            dataMaxRelMach.getValues(),
            dataVelocityHinlet.getValues(),
            dataVelocityHoutlet.getValues(),
            dataVelocityVoutlet.getValues(),
            dataVForceRotor.getValues(),
            dataDensityInlet.getValues(),
            dataBladePass.getValues(),
            dataIndex.getValues(),
            dataNumeroImagen.getValues());

        plotITMonitorMassFlow.
        export(resolvePath(plotsPath +
        "plotITMonitorMassFlow" + ".csv"), ",");
        plotITMonitorRotorVelocity.
        export(resolvePath(plotsPath +
        "plotITMonitorRotorVelocity" + ".csv"), ",");
        plotITMonitorEfficiency.
        export(resolvePath(plotsPath +
        "plotITMonitorEfficiency" + ".csv"), ",");
        plotITMonitorPTinlet.
        export(resolvePath(plotsPath +
        "plotITMonitorPTinlet" + ".csv"), ",");
        plotITMonitorVForceRotor.
        export(resolvePath(plotsPath +
        "plotITMonitorVForceRotor" + ".csv"), ",");

        String name = numeroImagen + "_" + "MassFlow_"  +
         Double.toString(Math.round(massFlow*10)/10) +
         "_Vrot_" +
         Double.toString(
         Math.round(Math.abs(rotorVelocity*100)/100));

        Scene sceneMach = simulation_0.
        getSceneManager().getScene("Mach");
        Scene scenePressure = simulation_0.getSceneManager().
        getScene("Presion");

        sceneMach.printAndWait(resolvePath(
        imagesPath + "Mach_" + name + ".png"),
         1, 2000, 725, true, true);
        scenePressure.printAndWait(resolvePath(
        imagesPath + "Presion_" + name + ".png"),
         1, 2000, 725, true, true);

        simulation_0.
        println("\n\n\n Resultados exportados\n\n\n");
        numeroImagen += 1;
        if(selector == 1 && (velocityHinlet > 160 ||
         Math.atan(rotorVelocity/velocityHinlet)
          - calado > 2*3.1415/180 ||
           Math.atan(rotorVelocity/velocityHinlet)
            - calado < -4*3.1415/180)){
            physicsModel = "coupled";
        }
        else{
            physicsModel = "segregated";
        }
```

```
                    if(efficiency < 0.65 ||
                    (vrotCaso == 0 && efficiency < 0.8)){
                        break loopVrot;
                    }
                }
                else{
                    break loopVrot;
                }
            }
        }
        iterMassFlow += signMassFlow*stepMassFlow;
        parameterMassFlow.getQuantity().setValue(iterMassFlow);
    }
}
    public double tipoMapeado(double massFlow, int mapeado){
    switch(mapeado){
        case 0:
            iterRotorVelocity = Math.floor(A/stepRotorVelocity)
            *stepRotorVelocity + stepRotorVelocity*index;
            break;
        case 1:
            iterRotorVelocity = A + B*index;
            break;
        case 2:
            if(Math.abs(index) <= 1){
                iterRotorVelocity = A + B*Math.sin(pi/2*index);
            }
            else{
                iterRotorVelocity += Math.signum(index)*B
                    *(1 - Math.sin(pi/2*(1-indexStep)));
            }
            break;
    }
    }
    public double slope(int xPosition, int yPosition, String file){

        String line;
        String cvsSplitBy = ",";

        double[] x;
        double[] y;

        data datax = new data();
        data datay = new data();

        try (BufferedReader br = new BufferedReader(new FileReader(
        filePath + file))) {
            br.readLine();
            while ((line = br.readLine()) != null) {
                // use comma as separator
                String[] data = line.split(cvsSplitBy);
                datax.add(Double.parseDouble(data[xPosition]));
                datay.add(Double.parseDouble(data[yPosition]));
            }
        }
        catch (IOException e) {
        }

```

```
489         x = datax.getValues();
490         y = datay.getValues();
491 
492         double num = 0;
493         double den = 0;
494 
495         for(int i = 0; i < x.length;i++){
496             num += y[i]*x[i];
497             den += Math.pow(x[i],2);
498         }
499         return num/den;
500     }
501 
502 private void clear(String clearType){
503 
504     Simulation simulation_0 = getActiveSimulation();
505     Solution solution_0 = simulation_0.getSolution();
506 
507     switch(clearType){
508         case "history":
509             solution_0.clearSolution(Solution.Clear.History);
510             break;
511         case "solution":
512             solution_0.clearSolution(Solution.Clear.Fields,
513              Solution.Clear.LagrangianDem);
514         default:
515             solution_0.clearSolution(Solution.Clear.History,
516              Solution.Clear.Fields, Solution.Clear.LagrangianDem);
517             break;
518     }
519 }
520 
521     private void write(String fileName, String cabecera,
522     double[]... valueArray){
523         BufferedWriter bwout;
524         try{
525             bwout = new BufferedWriter(new FileWriter(resolvePath(
526             filePath + fileName)));
527             bwout.write(cabecera);
528             int m = valueArray.length;
529             int n = valueArray[0].length;
530             for(int i = 0; i < n; i++){
531                 for(int j = 0; j < m; j++){
532                     bwout.write(Double.toString(valueArray[j][i]));
533                     if(j == m - 1){
534                         bwout.write("\n");
535                     }
536                     else{
537                         bwout.write(",");
538                     }
539                 }
540             }
541             bwout.close();
542         }
543         catch(IOException ioException){
544         }
545     }
546 
```

```
547
548
549     class data extends algoritm_0_v1_2_seg{
550
551         double[] base;
552         double last;
553
554         public data(){
555             base = new double[0];
556             last = 0;
557         }
558         public void add(double data){
559             last = data;
560             double[] vector = new double[base.length+1];
561             for(int i=0;i<base.length;i++){
562         vector[i]= base[i];
563             }
564             vector[vector.length-1] = last;
565             base = vector;
566         }
567         public double getLast(){
568             return last;
569         }
570         public double[] getValues(){
571             return base;
572         }
573         public boolean checkConvergence(){
574             return true;
575         }
576     }
577
578     private void setPhysicsModel(String model){
579
580         Simulation simulation_0 = getActiveSimulation();
581
582         PhysicsContinuum physicsContinuum =
583             ((PhysicsContinuum) simulation_0.getContinuumManager().
584             getContinuum(model));
585
586         Region rotor =
587           simulation_0.getRegionManager().getRegion("Rotor");
588
589         Region stator =
590           simulation_0.getRegionManager().getRegion("Stator");
591
592         physicsContinuum.add(rotor);
593
594         physicsContinuum.add(stator);
595
596         BoundaryInterface interfaceRotorStator =
597                   ((BoundaryInterface) simulation_0.getInterfaceManager().
598                   getInterface("interfaceRotorStator"));
599
600         switch(model){
601             case "segregated":
602             case "coupled":
603                 MotionSpecification motionSpecification_0 =
604                     rotor.getValues().get(MotionSpecification.class);
```

```
605
606                 StationaryMotion stationaryMotion_0 =
607                   ((StationaryMotion) simulation_0.get(MotionManager.class).
608                   getObject("Stationary"));
609
610                 motionSpecification_0.setMotion(stationaryMotion_0);
611
612                 TranslatingReferenceFrame translatingReferenceFrame_0 =
613                   ((TranslatingReferenceFrame) simulation_0.
614                   get(ReferenceFrameManager.class).
615                   getObject("ReferenceFrame for Translation"));
616
617                 motionSpecification_0.
618                 setReferenceFrame(translatingReferenceFrame_0);
619
620                 MonitorIterationStoppingCriterion
621                  monitorIterationStoppingCriterion_0 =
622                   ((MonitorIterationStoppingCriterion) simulation_0.
623                   getSolverStoppingCriterionManager().
624                   getSolverStoppingCriterion(
625                   "steadyStoppingMonitorEfficiency"));
626
627                 MonitorIterationStoppingCriterion
628                  monitorIterationStoppingCriterion_1 =
629                   ((MonitorIterationStoppingCriterion) simulation_0.
630                   getSolverStoppingCriterionManager().
631                   getSolverStoppingCriterion(
632                   "steadyStoppingMonitorVForceRotor"));
633
634                 monitorIterationStoppingCriterion_0.setIsUsed(true);
635
636                 monitorIterationStoppingCriterion_1.setIsUsed(true);
637
638
639                 MixingPlaneInterface mixingPlaneInterface =
640                     ((MixingPlaneInterface) simulation_0.
641                     get(ConditionTypeManager.class).
642                     get(MixingPlaneInterface.class));
643
644                 interfaceRotorStator.setInterfaceType(mixingPlaneInterface);
645
646                 if("coupled".equals(model)){
647                     CoupledImplicitSolver coupledImplicitSolver_0 =
648                         ((CoupledImplicitSolver) simulation_0.
649                         getSolverManager().
650                         getSolver(CoupledImplicitSolver.class));
651                     coupledImplicitSolver_0.setCFL(20.0);
652                 }
653                     break;
654             case "unsteadyCoupled":
655
656                 break;
657         }
658     }
659 }
```