# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

## FACULTAD DE INGENIERÍA

# Aplicación de aprendizaje automático en la predicción de presión y velocidad de un perfil aerodinámico

**TESIS**

Que para obtener el título de

**Ingeniero Aeroespacial**

**P R E S E N T A**

Bruno Alexis Teran Torres

**DIRECTOR DE TESIS**

Dr. Raúl Sánchez García

**Ciudad Universitaria, Cd. Mx., 2026**

**PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL**
**(Titulación con trabajo escrito)**

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado APLICACION DE APRENDIZAJE AUTOMATICO EN LA PREDICCION DE PRESION Y VELOCIDAD DE UN PERFIL AERODINAMICO que presenté para obtener el titulo de INGENIERO AEROESPACIAL es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

**BRUNO ALEXIS TERAN TORRES**
**Número de cuenta: 315268360**

# Agradecimientos

A mi mamá, Tere, por su eterno apoyo en mi camino, por ser mi mayor inspiración y por siempre creer que este día llegaría aun si yo lo dudé. Este logro es suyo también.

A mi familia, por siempre estar presente y acompañarme en mis metas, por ayudarme en facilidades como comida, transporte, compañía y palabras cuando las necesité.

A mi novia, Andy, por ser mi compañera de carrera, de clases, de tareas, de proyectos, de exámenes, de estudio, de motivación, de estrés, de consolación, de decepciones, de comidas, de juegos, de fiestas, de botanas, de viajes, de libros, de películas, de series, de trabajo, de regalos, de salidas, de podcast, de baile, de tesis y, para resumir, de vida. Con la promesa de que es solo el principio, muchas gracias.

A Dalinar Kholin, porque el paso más importante es siempre el próximo y, si debo caer, cada vez me alzaré como un hombre mejor.

A mi querida Facultad de Ingeniería, institución que me formó profesional y personalmente, que me brindó retos y satisfacciones, y en la cual aprendí el verdadero significado del esfuerzo y la dedicación. Siempre llevaré con orgullo su nombre y sus enseñanzas.

A mi director de tesis, el Dr. Raúl Sánchez García, por su guía constante, por su disposición para compartir sus conocimientos y por la confianza depositada en mí durante la realización de este trabajo. Su paciencia y compromiso han sido un ejemplo invaluable.

A mis sinodales, el Dr. Roberto Gómez Martínez, Dr. David Israel Posadas Navarro, Mtra. Zaira Linnete Torres Cortés y el Dr. César Abraham Luna Estrada, por su apoyo, comprensión y observaciones.

# Resumen

En este trabajo se propone una implementación del aprendizaje automático para la dinámica de fluidos, ajustando y entrenando una red neuronal con arquitectura MLPs de tipo regresión, esto para la predicción de presión y velocidad en una nueva malla programada en Python a partir de datos obtenidos mediante la simulación CFD de un perfil aerodinámico tipo NACA 4412. Se obtienen los hiperparámetros que mejor ajustan el modelo, evaluando la precisión y generalización mismo con el *Mean Absolute Error* para distintos casos, resultando en un porcentaje de error de entre el 10 y 17 por ciento para el caso objetivo. Igualmente se analiza y evalúa la similitud estructural de los datos precedidos con los originales mediante la *Structural similarity index measure*, con resultados favorables y casi idénticos de entre el 0.97 y 0.99 en el caso objetivo. Concluyendo que la implementación es posible y eficiente, pero se requiere mayor énfasis en algunas zonas como el contorno del perfil para una mayor precisión en la regresión, aunado con un aumento en el número de simulaciones realizadas, sin embargo, estructural y geométricamente responde bien debido a la naturaleza de los datos de entrada.

# Glosario

- **Dinámica de Fluidos Computacional (Computational Fluid Dynamics, CFD)**: Es una disciplina de la mecánica de fluidos que utiliza métodos numéricos y algoritmos computacionales para resolver y analizar el comportamiento de fluidos y sus interacciones con superficies.
- **Aprendizaje automático (Machine Learning)**: Rama de la inteligencia artificial que desarrolla algoritmos capaces de aprender patrones a partir de datos y mejorar su desempeño en una tarea sin ser programados explícitamente.
- **Aprendizaje profundo (Deep Learning)**: Es una subárea del aprendizaje automático que utiliza redes neuronales artificiales con muchas capas (profundas).
- **Característica (Feature)**: Variable de entrada al modelo. Ej: coordenadas, ángulo de ataque, etc.
- **Valor objetivo (Target value)**: Valor de la variable de salida que queremos predecir.
- **Hiperparámetro (Hyperparameter)**: Valor que define cómo se entrena la red, no se aprende automáticamente. Ej: tasa de aprendizaje, número de capas, batch size..
- **Entrenamiento (Training)**: Proceso de ajustar los pesos y sesgos internos del modelo a los datos.
- **Validación (Validation)**: Conjunto de datos separado para evaluar el rendimiento del modelo mientras se entrena.
- **Prueba (Test)**: Conjunto de datos completamente separado que se usa únicamente al final para evaluar el desempeño real del modelo.

- **Sobreajuste (Overfitting)**: Cuando el modelo aprende demasiado bien los datos de entrenamiento, incluyendo el ruido, y falla al generalizar en datos nuevos.
- **Ajuste (Tuning)**: Buscar los mejores hiperparámetros de un modelo.
- **Época (Epoch)**: Una pasada completa del modelo sobre todos los datos de entrenamiento.
- **Métrica (Metric)**: Función de evaluación que se utiliza para medir el desempeño de un modelo, comparando sus predicciones con los valores reales.
- **Función de pérdida (Loss Function)**: Función matemática que mide el error entre la predicción y el valor real, usada durante el entrenamiento.
- **Función de costo (Cost Function)**: Promedio de la pérdida sobre todo el conjunto de entrenamiento.

# Índice general

# Índice de figuras

# Índice de tablas

# 1 Introducción

## 1.1. Contexto

En la industria aeronáutica, la optimización de la forma de un perfil aerodinámico es una tarea crucial en la fase de diseño de cualquier vehículo aéreo, ya que su geometría afecta directamente las características aerodinámicas generales de la aeronave. Además, también contribuye a la reducción de costos, la disminución de emisiones, el diseño del sistema de control de vuelo y la predicción de fenómenos complejos de interacción fluido-estructura, como las inestabilidades aeroelásticas [12] [13].
Tradicionalmente, para evaluar el rendimiento de un perfil aerodinámico se recurre a pruebas experimentales en túneles de viento o, dependiendo de la precisión requerida, a simulaciones numéricas basadas en las ecuaciones de la mecánica de fluidos [2]. En el primer método, es necesario fabricar un modelo de prueba instrumentado con diversos sensores y realizar ensayos en un túnel de viento. En el segundo método, se deben llevar a cabo simulaciones numéricas que suelen implicar un alto costo computacional. La aplicación de cualquiera de estos métodos requiere cierto tiempo, lo cual es crucial en las fases de diseño de las aeronaves.
Por otro lado, el uso de la inteligencia artificial (IA) en simulaciones de dinámica de fluidos computacional (CFD) ha ganado relevancia en diversas áreas de la ingeniería [14]. La aplicación de algoritmos de aprendizaje automático permite explorar automáticamente múltiples opciones de diseño y encontrar soluciones más eficientes y seguras. Además, puede reducir el tiempo de cálculo de CFD y, por lo tanto, minimizar el uso de recursos computacionales. También se puede aplicar para mejorar los modelos de turbulencia utilizados en CFD, crear mallas de alta calidad para la simulación, realizar un modelado más completo y realista de sistemas complejos mediante el acoplamiento de CFD con otras simulaciones, como transferencia de calor, electromagnetismo o mecánica estructural [15]. Finalmente, el uso de IA puede detectar patrones anómalos basados en los resultados de la simulación y permitir la identificación temprana de problemas potenciales en sistemas con fluidos [16].

## 1.2. Justificación

Con base en la tendencia de las últimas décadas en el continuo mejoramiento de la capacidad de las computadoras y el desarrollo de algoritmos computacionales aplicados a la ingeniería aeroespacial y aeronáutica, es de esperarse que en un futuro próximo muchas de las simulaciones realizadas en la actualidad con un túnel de viento o códigos numéricos computacionales de CFD, se puedan hacer con algoritmos de inteligencia artificial con las grandes ventajas que esto conlleva; por esta razón, en este tema de investigación es de interés la aplicación de la inteligencia artificial en la predicción de parámetros para caracterizar perfiles aerodinámicos.
En este trabajo se investiga la viabilidad de aplicar redes neuronales artificiales (RNA) para acelerar los cálculos de CFD y predecir soluciones aproximadas del campo fluido de un perfil aerodinámico antes de ejecutar simulaciones completas. Esto permitiría una reducción significativa del tiempo de procesamiento y del uso de recursos computacionales, lo cual sería de gran utilidad en las fases de diseño de aeronaves.

## 1.3. Objetivos

Los objetivos principales de este proyecto son:

1. Realizar simulaciones numéricas de CFD con un solo perfil aerodinámico, considerando distintas velocidades del viento y ángulos de incidencia. Estos resultados servirán como referencia y como base de datos para la entrada del entrenamiento de la red neuronal.

2. Explorar diferentes tipos de redes neuronales y seleccionar la que mejor se ajuste a la problemática de este trabajo

3. Configurar y entrenar la red neuronal.

4. Poner a prueba la red neuronal con un perfil aerodinámico de referencia

## 1.4. Premisa

La tesis que se intenta probar es que existe la posibilidad de implementar metodologías del aprendizaje automático en la optimización de la técnica para caracterizar perfiles aerodinámicos reduciendo tiempos y costos computacionales.

# 2 Marco teórico

## 2.1. Aerodinámica de los perfiles alares

En primer lugar, al construir los conceptos básicos manejados en la dinámica de fluidos, *The American Heritage Dictionary of the English* Language definió la aerodinámica como "la dinámica de los gases, especialmente las interacciones atmosféricas con objetos en movimiento", [4] es decir, que cualquier cuerpo que interactué con un fluido a una cierta velocidad se ve afectado por las propiedades de este campo, obteniendo valores específicos de presión y velocidad en los diferentes puntos alrededor de este mismo, moldeando así el flujo según la forma y dimensiones del objeto.
A partir de lo anterior, la aerodinámica hace una distinción entre los tipos de cuerpos a estudiar, en primer lugar se denominan cuerpos romos a aquellos cuya geometría favorece el desprendimiento de la capa límite y genera una elevada resistencia de presión debido a la diferencia de esta magnitud entre la parte frontal y trasera del cuerpo; ejemplos de este tipo son edificios, puentes, automóviles, pelotas de golf, paracaídas, entre otros. Por el contrario, cuando los cuerpos están diseñados específicamente para evitar el desprendimiento de la capa límite; es decir, interesa que dicha capa sea laminar en todo su recorrido, se denominan cuerpos fuselados [17]. En los cuerpos fuselados, la resistencia se debe principalmente a la fricción de las moléculas del aire con la superficie del cuerpo; ejemplos de este tipo son los aviones, los automóviles de carrera y los submarinos.
Cabe resaltar que, para estudiar ambos tipos de cuerpos, es necesario contar con parámetros cuyos valores permitan obtener resultados cuantitativos, ya que la mera distinción entre ellos se basa en lo definido como resistencia de presión. Como se menciona en [2], actualmente existen tres técnicas para tratar los flujos externos: soluciones numéricas en computadora, experimentación y teoría de la capa límite, y, de estas, la única manera satisfactoria para conocer las fuerzas de resistencia sobre un cuerpo cualquiera en una corriente de aire a un número de Reynolds arbitrario son los resultados experimentales y los de CFD (Dinámica de fluidos computacionales),

cuyas metodología se presentara más adelante.

### 2.1.1. Ecuaciones gobernantes

Haciendo un paréntesis, y, antes de pasar completamente a la parte experimental de la aerodinámica, se debe entender que el estudio de esta se remonta únicamente a tres ecuaciones que gobiernan el comportamiento de los fluidos: continuidad (definida por la ecuación 2.1), cantidad de movimiento (ecuación 2.2) y energía. Dichas ecuaciones no son mas que los principios fundamentales de la física, la conservación de la masa, la segunda ley de Newton y la conservación de la energía. De ellas, las ecuaciones de cantidad de movimiento o de Navier Stokes (ecuación 2.2) son el motivo por la que existen muy pocas soluciones exactas en la dinámica de fluidos, pues, esta es una ecuación diferencial parcial de segundo orden no lineal e inestable debido principalmente a la turbulencia, esto da como resultado que la experimentación y la potencia numérica de las computadoras modernas sean herramientas imprescindibles en el estudio de esta disciplina.
La ecuación de continuidad, para fluidos incompresibles, se expresa como [18]:

$$\frac{\partial U_j}{\partial x_j} = 0 \tag{2.1}$$

donde $U_j$ son las componentes de la velocidad del viento (U, V, W) y $x_j$ son las direcciones principales del viento (x, y, z).
En el caso de la ecuación de Navier-Stokes, esta se puede definir de la siguiente manera:

$$\underset{(\mathrm{I})}{\frac{\partial U_i}{\partial t}} + \underset{(II)}{U_j \frac{\partial U_i}{\partial x_j}} = \underset{(III)}{-\delta_{i3} g} + \underset{(IV)}{f_c \varepsilon_{ij3} U_j} - \underset{(V)}{\frac{1}{\rho}\frac{\partial p}{\partial x_i}} + \underset{(VI)}{\nu \frac{\partial^2 U_i}{\partial x_j^2}} \tag{2.2}$$

donde $\delta_{i3}$ es la delta de Kronecker, $g$ es la gravedad, $f_c$ es el parámetro de Coriolis, $\varepsilon_{ij3}$ es el tensor unitario, $p$ es la presión y $\nu$ es la viscosidad cinemática. El término I representa el almacenamiento del momento (inercia), el término II describe la advección, III describe la acción de la gravedad presente solo en dirección vertical, IV describe la influencia de la rotación de la tierra (efecto de Coriolis), el término V describe las fuerzas del gradiente de presión, y VI representa la influencia de los esfuerzos viscosos [19].
Como la turbulencia (término VI) es una parte intrínseca del viento y los fenómenos a pequeña escala son transitorios, la descripción de cada uno de estos es imposible y, por lo tanto, no se puede hacer una descripción determinista del flujo turbulento [18]. Con lo anterior, una forma de acercarnos a ello es considerar la velocidad del viento

como la suma de una velocidad media más una velocidad turbulenta, esto se expresa en la ecuación 2.3.

$$U = \overline{U}_j + u'_j \tag{2.3}$$

donde $\overline{U}_j$ es la velocidad media y $u'_j$ es la velocidad turbulenta.

### 2.1.2. Promedios de Reynolds

Haciendo un procedimiento similar al anterior para valores promedio de todas las variables en la ecuación 2.2, es posible tratar de eliminar la parte turbulenta y contar con una descripción media del viento [19], obteniendo la siguiente ecuación:

$$\underset{(\mathrm{I})}{\frac{\partial \overline{U}_i}{\partial t}} + \underset{(II)}{\overline{U}_j \frac{\partial \overline{U}_i}{\partial x_j}} = \underset{(III)}{-\delta_{i3} g} + \underset{(IV)}{f_c \varepsilon_{ij3} \overline{U}_j} - \underset{(V)}{\frac{1}{\rho} \frac{\partial \overline{P}}{\partial x_i}} + \underset{(VI)}{\nu \frac{\partial^2 \overline{U}_i}{\partial x_j^2}} - \underset{(VII)}{\frac{\partial (\overline{u'_j u'_j})}{\partial x_j}} \tag{2.4}$$

donde de forma similar, el término I representa el almacenamiento promedio del momento (inercia), el término II describe la advección del momento medio por la velocidad media, III describe la acción de la gravedad presente solo en dirección vertical, IV describe la influencia de la rotación de la tierra (efecto de Coriolis), el término V describe las fuerzas medias del gradiente de presión, VI representa la influencia de los esfuerzos viscosos sobre los movimientos medios y el término VII representa la influencia de los esfuerzos de Reynolds sobre los movimientos medios.
Es en el último término donde surge nuevamente el problema, pues la turbulencia aun debe ser considerada en la descripción del flujo para flujos desconocidos ($\overline{u'_j u'_j}$), de manera que, si se intenta utilizar la ecuación 2.4 derivando ecuaciones predictivas adicionales nos encontraremos con un término adicional, $\overline{u'_j u'_j u'_k}$. Si ahora se deriva una ecuación adicional para este tercer término, es de esperarse que, nuevamente, se tengan cantidades desconocidas una y otra vez. De tal forma que, para cualquier conjunto finito de ecuaciones, la descripción de la turbulencia no está cerrada (problema de cierre) [18].
Más adelante se muestra que para la Dinámica de Fluidos Computacionales se aproximan los términos desconocidos con cantidades conocidas (parametrizaciones), con el fin de cerrar dichas ecuaciones y resolverlas. A estas parametrizaciones se les conoce como modelos de turbulencia [19]. Sin embargo, con los problemas presentados, ahora es posible ver el grave problema que representa modelar fluidos de forma analítica, es por ello que se cuenta con las dos metodologías ya mencionadas para sistemas de cualquier tipo de complejidad.

### 2.1.3. Fuerzas y momentos aerodinámicos

El objetivo último, tanto del método experimental como del método de CFD, independientemente de lo complejo que sea el flujo alrededor del cuerpo, es determinar las fuerzas y momentos de origen aerodinámico que actúan sobre ellos. Dichas fuerzas se deben a dos causas principales, por un lado, a la distribución de presiones sobre la superficie del cuerpo y, por el otro, a la distribución de esfuerzos viscosos sobre esta misma superficie [17]. A partir de ambos efectos, se obtienen tres componentes de fuerza fundamentales para el análisis aerodinámico: la resistencia (o arrastre), la sustentación y el momento.
La primera de estas fuerzas, el arrastre, representa a una pérdida de cantidad de movimiento que el cuerpo debe vencer para desplazarse en contra del flujo. La sustentación, orientada perpendicularmente al arrastre, se opone al peso y resulta fundamental para mantener el equilibrio del cuerpo en el medio. La tercera componente, el momento, se presenta como una fuerza que, como su nombre lo indica, provoca un giro sobre uno de los ejes del sistema, cambiando así su nombre en función del eje sobre el que actúe, puede presentarse como guiñada, cabeceo y alabeo [2]. Todas estas componentes y su respectiva dirección se ilustran en la Figura 2.1.
Aunque, las fuerzas previamente mencionadas son esenciales para obtener la denomi-

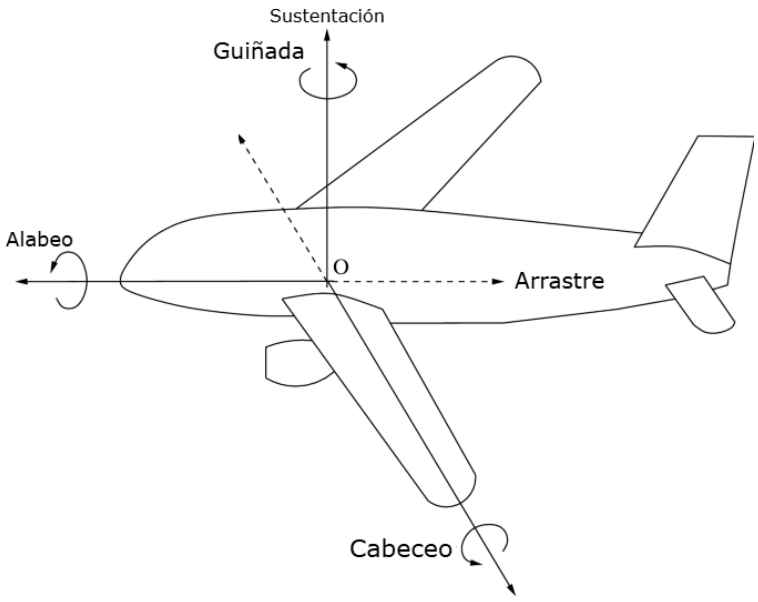


Figura 2.1: Definición de fuerzas y momentos sobre un cuerpo inmerso en una corriente uniforme [1].

nada aerodinámica de un cuerpo, no se suele tratar con ellas, en su lugar se usan los coeficientes correspondientes de arrastre ($C_D$), sustentación ($C_L$) y momento ($C_M$), definidos de tal manera que son mucho más útiles para la ingeniería y los cálculos [4]. Para la superficie sustentadora de interés, los perfiles aerodinámicos o alares, estos coeficientes dependen solo de la forma, orientación (ángulo de ataque) del perfil alar y de la velocidad del flujo [4].
En la definición de perfil aerodinámico se considera el ala de un avión, la forma de la sección transversal obtenida al cortar el ala con el plano perpendicular se denomina perfil alar, cuyo diseño convencional es con un borde de ataque redondeado y un borde de salida agudo [4] [2]. Para una mejor referencia en la Figura 2.2 se puede observar las fuerzas aerodinámicas, así como el ángulo de ataque asociado al flujo incidente.

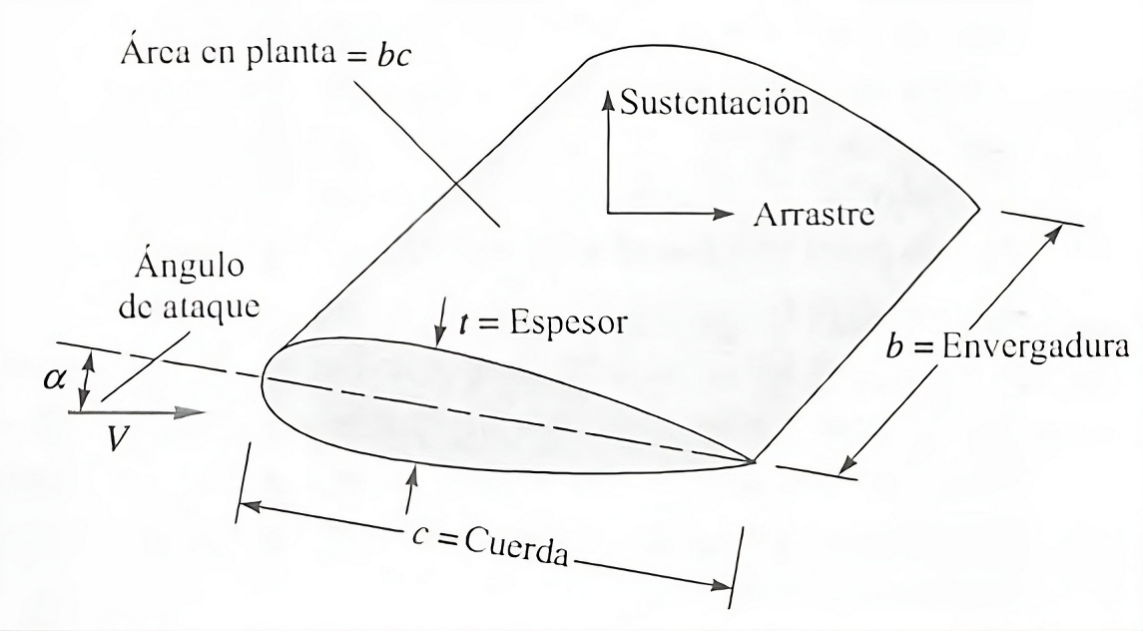


Figura 2.2: Definición de parámetros geométricos de un perfil alar [2].

La forma tan característica de un perfil aerodinámico facilita una distribución de presión optima en la generación máxima de sustentación y la generación mínima de arrastre, tal como se muestra en la Figura 2.3, pues, siguiendo la ecuación de Bernoulli, el flujo por encima del perfil tiene una velocidad mayor que por debajo, generando la diferencia de presión que permite la sustentación, además de favorecer un flujo laminar sin apenas desprendimiento del mismo, efecto que, como veremos enseguida, es una de las causas que puede generar arrastre en nuestro perfil. Algo

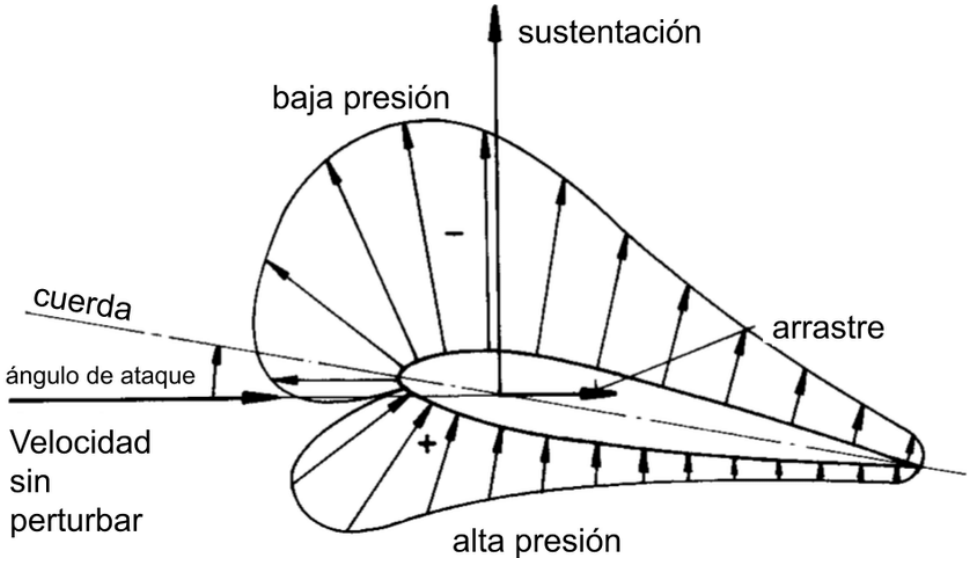


Figura 2.3: Distribución de presiones en un perfil aerodinámico [3].

importante a considerar, es que la distribución de presión mostrada en la Figura 2.3 es la más frecuente en ángulos de ataque cercanos al cero, sin embargo, esta cambia drásticamente a medida que dicho angulo aumenta o disminuye en un cierto rango, típicamente ± 12 grados [17], pues tal como observamos en la Figura 2.4, se crea una zona donde el flujo se desprende por consecuencia a la propia geometría del perfil, con ello se incrementa el arrastre y se disminuye la sustentación en el proceso. Otro mecanismo por el que se genera resistencia al flujo, es debido a los esfuerzos cortantes en la piel del perfil, producidos por la fricción entre el material y el fluido. La suma de ambos se conoce como arrastre total y, según el régimen en que se encuentre, existe una mayor presencia de alguno de los dos, mayor desprendimiento en laminar, y mayor fricción en el turbulento [4].

## 2.2. Metodologías para medir la aerodinámica de perfiles alares

Como se mencionó previamente, para caracterizar la aerodinámica de un perfil alar resulta más útil utilizar los coeficientes de arrastre, sustentación y momento, dichos coeficientes, obtenidos a partir de un análisis dimensional, los encontramos en

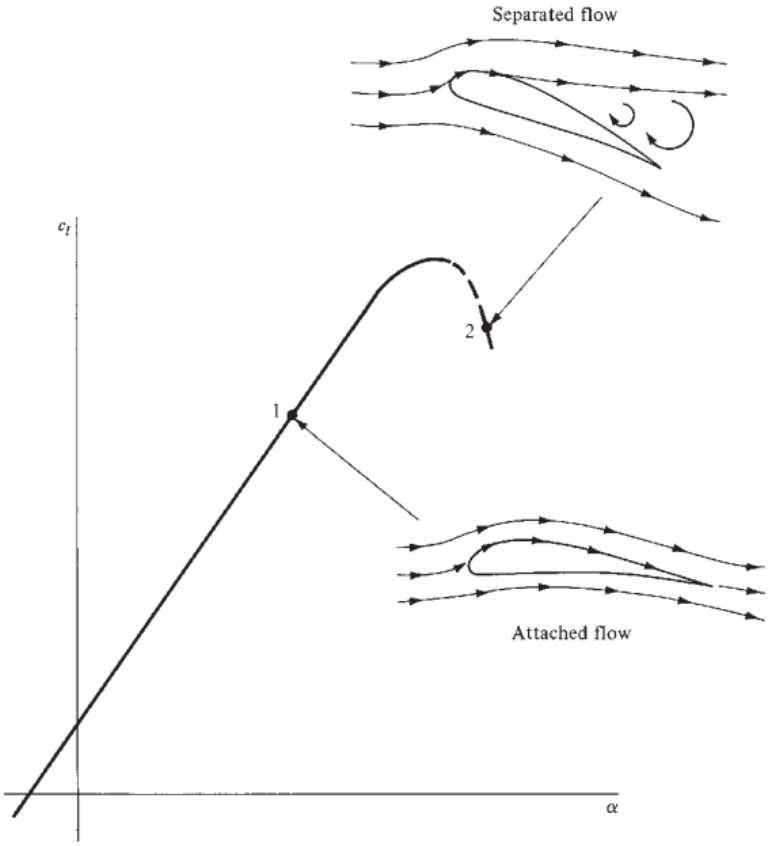


Figura 2.4: Coeficiente de sustentación, $C_L$, en función del ángulo de ataque. [4].

la ecuación 2.5:

$$C_L = \frac{L}{q_\infty S} \qquad C_D = \frac{D}{q_\infty S} \qquad C_M = \frac{M}{q_\infty S} \tag{2.5}$$

$$q_\infty = \frac{1}{2}\rho_\infty V_\infty^2 \tag{2.6}$$

donde L, D y M son las fuerzas de sustentación, arrastre y momento, $S$ es el área del perfil y $q_\infty$ es la presión dinámica, ecuación 2.6, donde $\rho_\infty$ y $V_\infty$ es la densidad y velocidad del fluido respectivamente. Estas expresiones son fundamentales para toda la aerodinámica aplicada y, dentro de ellas, están contenidos valores geométricos y de presión que, tal como se expuso previamente, resultan en la base para el calculo de estas fuerzas.

Antes de pasar propiamente con la metodología, mediante la cual se obtienen dichos coeficientes, es necesario abordar los diferentes diseños que se pueden encontrar de perfiles aerodinámicos, pues, la teoría para las secciones modernas se remonta hasta las pruebas realizadas en Gotinga, Alemania, durante la primera guerra mundial, utilizada, igualmente, durante la segunda. Sin embargo, fue también en este periodo donde un gran número de familias de perfiles alares fueron probadas en laboratorios de distintos países, donde, el trabajo realizado por NACA, el Comité Asesor Nacional de Aeronáutica, fue el mas sobresaliente [20]. Esta familia, al día de hoy, es la más usada y, la que cuenta con una mayor cantidad de estudios experimentales en su haber.
La familia NACA, propiamente dicha, cuenta a su vez con series de 4, 5 y 6 dígitos, donde cada número de la serie representa una característica geométrica del perfil. Con esto dicho, únicamente se enfocará en la serie de 4 dígitos para este estudio.
Igualmente, con el propósito de explicar la nomenclatura de esta serie, debemos abordar más a fondo algunas otras secciones geométricas de un perfil aerodinámico. Para ello, si observamos la Figura 2.5 podremos identificar las de mayor importancia, como es la cuerda; definida como la línea recta que une el borde de ataque con el borde de salida, la curvatura; la distancia máxima entre la línea media y la cuerda, y el espesor; la distancia entre la parte superior e inferior del perfil [21].

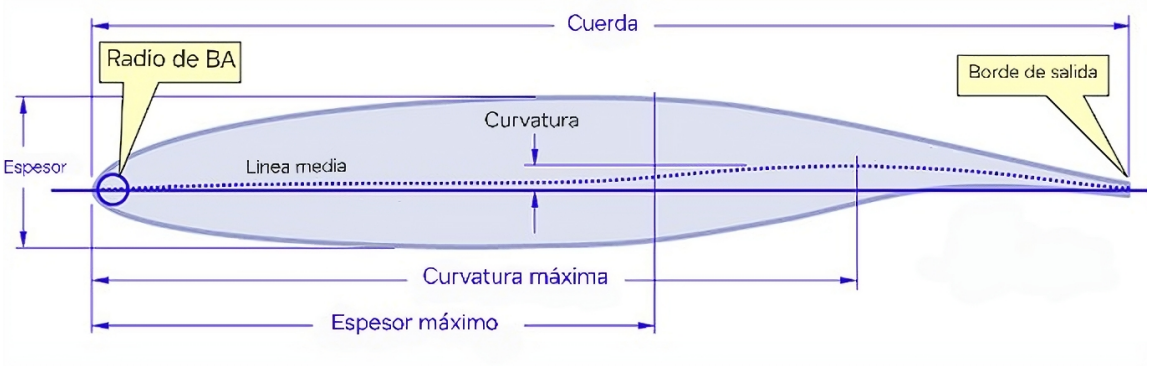


Figura 2.5: Partes de un perfil alar. [5].

Una vez definidas las partes del perfil es posible explicar la nomenclatura NACA de 4 dígitos, en esta, el primer dígito especifica la curvatura máxima (m) en porcentaje de la cuerda, el segundo indica la posición de la curvatura máxima (p) expresado en décimas, y los dos últimos dos números indican el espesor máximo (t) en porcentaje de la cuerda [21]. Es decir que, por ejemplo, el perfil NACA 2412 cuenta con una curvatura del 2 por ciento de la cuerda, su curvatura máxima se encuentra al 40 por

ciento de la cuerda (0.4) y su grosor máximo es del 12 por ciento de la cuerda.
Con los parámetros m, p y t ahora es posible calcular las coordenadas de un perfil completo de la siguiente manera:

1. Elegir el número de puntos sobre la dirección x desde 0 hasta el tamaño de la cuerda, uniformemente espaciados.

2. Calcular las coordenadas de la línea utilizando las ecuaciones para las coordenadas de la curvatura, ecuaciones 2.7 y 2.8, para x=0 hasta x=p y para x=p hasta x=c respectivamente.

$$y_c = \frac{m}{p^2}(2px - x^2) \tag{2.7}$$

$$y_c = \frac{m}{(1-p)^2}[(1-2p) + 2px - x^2] \tag{2.8}$$

   donde $x$ y $y_c$ son las coordenadas de la curvatura, y m, p y t son los ya definidos curvatura máxima, posición de la curvatura máxima y espesor máximo y c es el tamaño de la cuerda.

3. Calcular las coordenadas del espesor para usar por encima y por debajo de la línea media con la ecuación 2.9.

$$y_t = \frac{t}{(0.2)}(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4) \tag{2.9}$$

   donde $y_t$ es la coordenada $y$ considerando el espesor.

4. Determinar las coordenadas finales para la parte superior e inferior del perfil con las ecuaciones 2.10 y 2.11.

$$y_{upper} = yc + yt \tag{2.10}$$

$$y_{lower} = yc - yt \tag{2.11}$$

   donde $y_{upper}$ y $y_{lower}$ son las coordenadas en $y$ finales por encima y por debajo de la cuerda del perfil.

Siguiendo los pasos anteriores, es posible obtener las coordenadas de cualquier perfil NACA de cuatro dígitos, para comprobar dicha metodología se puede observar la Figura 2.6 cuyos puntos fueron obtenidos de esta forma.

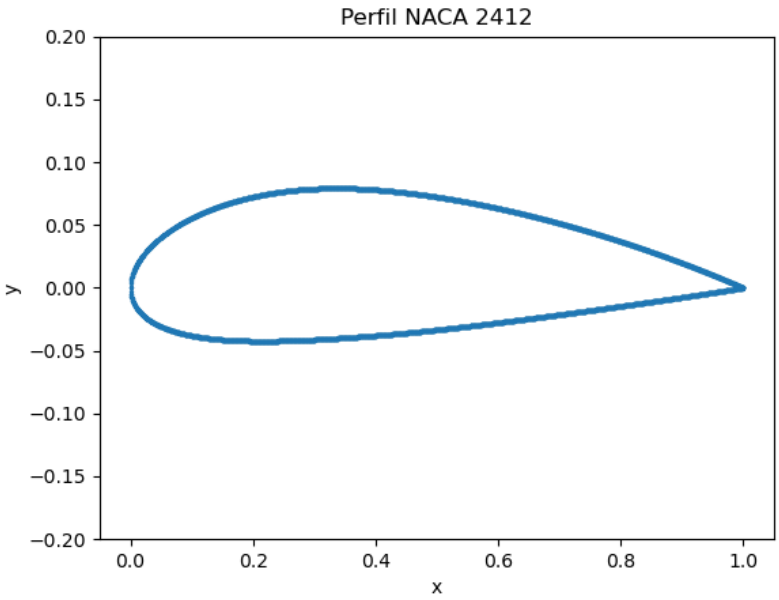


Figura 2.6: Perfil aerodinámico NACA 2412.

La mención de la serie NACA de perfiles aerodinámica es necesaria, pues, como ya se mencionó previamente, existen una gran cantidad de estudios y pruebas en dicha serie; muchos de estos perfiles ya han sido caracterizados utilizando esencialmente los coeficientes definidos al principio de este capítulo, así, cada perfil NACA tiene sus propias curvas de sustentación, arrastre o momento, en función del ángulo de ataque, por lo que, es fácil encontrar un perfil el cual se adecué a los objetivos y al tipo de aeronave que se pueda necesitar. Sin embargo, es precisamente la metodología para caracterizar estos perfiles lo que es de interés, pues, como se muestra a continuación, el estudio y el mejoramiento de estas pruebas es tema de gran interés aún en la actualidad.

### 2.2.1. Pruebas en túnel de viento

La ingeniería aeroespacial en general y, la aerodinámica en particular, son disciplinas basadas en gran medida en la experimentación empírica, por lo que, el desarrollo o descubrimiento de instrumentos y métodos nuevos que permitan su estudio ha sido de vital importancia en estos campos. La primer gran metodología y que, incluso a día de hoy, se utiliza en la mayoría de los laboratorios de esta índole, son las pruebas

en túnel de viento.
En el sentido más básico, un túnel de viento es una instalación experimental en tierra diseñada para generar flujos de aire (u otros gases) que simulan los flujos naturales que ocurren fuera del laboratorio [4]. Estas instalaciones son verdaderamente útiles para la experimentación, pues, aún en túneles de viento de baja velocidad, permiten utilizar modelos preparados en etapas tempranas del ciclo de diseño, ya sean modelos aeronáuticos o de muchas otras industrias como la automovilística, esto debido a que se incluye toda la complejidad de un flujo real, y, además pueden proporcionar grandes cantidades de datos confiables como un medio rápido, económico y preciso para investigaciones donde se respalden decisiones de diseño [22].
Existen dos tipos básicos de túneles de viento, los de circuito abierto y los de circuito cerrado; el esquema de ambos lo observamos en la Figura 2.7, sin embargo, hay una cantidad prácticamente infinita de variaciones en las características específicas de distintos túneles, cada tipo tiene sus ventajas y desventajas y, la elección depende en general del presupuesto disponible y del propósito experimental. Algunos ejemplos específicos a nombrar, son los que han sido diseñados con fines aeronáuticos, estos suelen tener como particularidad que son túneles de alta velocidad (subsonicos, transónicos o supersónicos) tales como el túnel V/STOL, túneles de propulsión, túneles de viento vertical, túneles de alto número de Reynolds, entre otros [22]. Una característica importante de las pruebas en túnel de viento es que, los modelos usados en ellas usualmente están a una escala reducida, debido a ello, existen coeficientes adimensionales que nos permiten escalar e interpretar los datos resultantes, los tres coeficientes, junto con las propiedades que relacionan (fuerzas representadas directamente en la ecuación 2.2 de Navier Stokes), se observan en las ecuaciones 2.12, 2.13 y 2.14.

$$\frac{\text{Fuerza de inercia}}{\text{Fuerza viscosa}} = \frac{\rho V l}{\mu} = \text{Número de Reynolds} \tag{2.12}$$

$$\frac{\text{Fuerza de inercia}}{\text{Fuerza elástica}} = \frac{V l}{a} = \text{Número de Mach} \tag{2.13}$$

$$\sqrt{\frac{\text{Fuerza de inercia}}{\text{Fuerza de gravedad}}} = \sqrt{\frac{V^2}{lg}} = \text{Número de Froude} \tag{2.14}$$

donde $\rho$ es la densidad del fluido, $V$ es la velocidad, $l$ la longitud característica, $\mu$ es la viscosidad dinámica del aire, $a$ es la velocidad del sonido y $g$ es la gravedad.
En la práctica, el número de Froude solo es un parámetro importante en pruebas

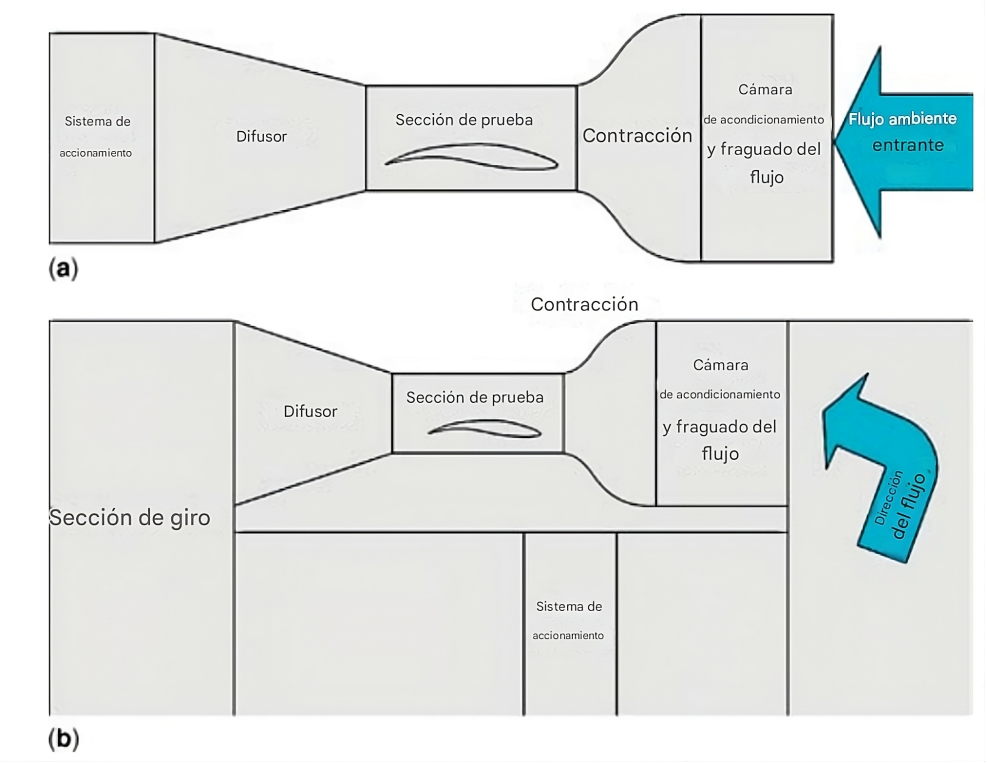


Figura 2.7: Esquemas de túnel (a) circuito abierto y (b) circuito cerrado [6].

dinámicas, es decir, en las que se analiza el movimiento del modelo, mientras que, tanto el número de Reynolds, como el de Mach, son parámetros de similitud significativos en modelos estacionarios. De esta forma, si un experimento el cual usa un modelo a escala tiene los mismos números de Reynolds y de Mach que la aplicación a escala real, entonces, ambos flujos serán dinámicamente similares [22]. Cabe mencionar que, la similaridad del número de Mach suele aplicarse únicamente a vehículos a una alta velocidad, pues predominan los efectos relacionados a esta cantidad y la coincidencia con Reynolds no es tan crítica, por el contrario para vuelos de velocidad baja predominan los efectos del número de Reynolds.
Un hecho importante en el escalamiento es que, a partir de la ecuación 2.5, se deduce que para un flujo cuyas propiedades de temperatura y presión son las mismas, la fuerza también será la misma sin importar la combinación de tamaño y velocidad utilizadas, siempre y cuando estas sean proporcionales.
Un túnel de viento puede ser usado para medir distintas propiedades del flujo o del modelo a utilizar, tales como temperatura, presión (atmosférica, estática, dinámica),

velocidad, fuerzas, entre otras. Para ello, este se equipa con instrumentos de medición según cual sea el objetivo de la instalación, en el caso de la aeronáutica en general es esencial la obtención de los valores de presión y velocidad medidos alrededor de nuestro perfil, es por ello que a continuación se se nombrarán algunos de los dispositivos más comunes utilizados para estos fines.
En primer lugar, para la medición de presión, uno de los dispositivos más antiguos y uno de los más fáciles de construir es el manómetro. Si bien, este término se aplica a cualquier dispositivo utilizado para medir presión diferencial, el mas comúnmente utilizado es el tubo en forma de U, mostrado en la Figura 2.8, en el que, al tomar como dato la diferencia de alturas del líquido dentro de este, se puede obtener asimismo la diferencia entre una presión de referencia, como la atmosférica, y una presión de proceso, como la de un punto en un modelo de ala.

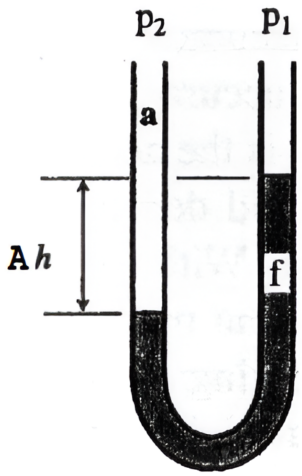


Figura 2.8: Manómetro de tubo en U [6].

Aunque, el manómetro es un dispositivo muy sencillo de usar, el uso más común de este es calibrar y verificar otros dispositivos, pues es difícil obtener un resultado más preciso en el rango de presiones diferenciales comunes. Es por ello que, actualmente se usan instrumentos más modernos como los transductores de presión, este término se aplica a cualquier dispositivo que proporciona una respuesta eléctrica a una presión o cambio de presión [22], los transductores se pueden instalar tanto en el propio túnel, para medir las condiciones de operación, como en los modelos, para obtener medidas de presión en las paredes o superficies que lo requieran.
Existen diversos principios físicos con la que la medición de presión se efectúa, algunos de ellos son: el bombardeo molecular sobre una lámina muy fina, chips sensores

de presión, acelerómetros y tecnología piezoresistiva [23]. Este último es el tipo más frecuente de transductor y se le conoce como de tipo diafragma, en el cual, el mecanismo sensor básico es una lamina delgada de material que se deforma cuando la presión cambia a través de ella, un esquema de este dispositivo se observa en la Figura 2.9.

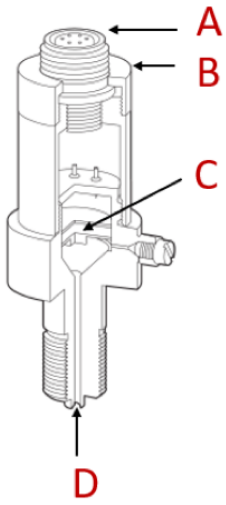


Figura 2.9: Componentes del transductor de presión de galga extensiométrica: conector (A), carcasa (B), galga extensiométrica (C) y entrada de presión (D) [7].

Para el caso de la velocidad del flujo, el dispositivo mas comúnmente usado es el tubo de Pitot, este tiene como característica que, al igual que los instrumentos anteriores, su uso radica en medir la presión, sin embargo, puede obtener valores tanto de la presión total como de la presión estática, de tal forma que, mediante la ecuación 2.15 también es posible conocer la presión dinámica y, con ella, obtener la velocidad. Un esquema del tubo de Pitot se muestra en la Figura 2.10.

$$p + \frac{1}{2}\rho V^2 = p_0 \tag{2.15}$$

Donde: $p$ es la presión estática y $P_0$ es la presión total.
Existen otros instrumentos para medir la velocidad, sin embargo, estos son usados principalmente para la calibración del tubo Pitot, tales como, por ejemplo, el anemómetro de hilo caliente.
Una vez explicada la distinta instrumentación que se suele usar en un túnel de viento, se retoma lo mencionado en la aerodinámica de los perfiles alares, en donde se determinó que con parámetros como la presión es posible calcular la fuerza aerodinámica.

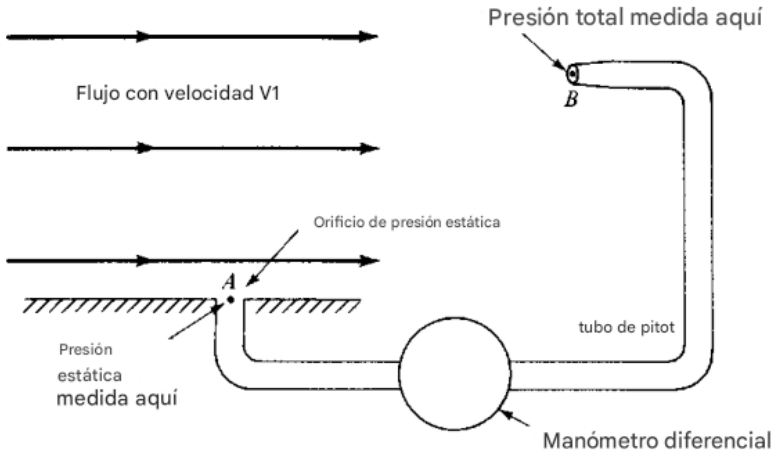


Figura 2.10: Esquema de tubo de pitot para medir presiones [4].

En este caso, usando la presión estática en los distintos puntos a lo largo de un perfil aerodinámico, como se ve la ecuación 2.16, es posible obtener la fuerza total, que, a su vez, se puede descomponer para obtener tanto la fuerza de sustentación como la fuerza de arrastre, con las que, mediante el uso de la ecuación 2.5, es posible obtener los coeficientes para caracterizar dicho perfil.

$$F = \int_{as} (-pI + \tau) n dS \tag{2.16}$$

donde $p$ es la presión que actúa sobre la superficie, $I$ es el tensor identidad, $\tau$ es el tensor de esfuerzos viscosos, $n$ es el vector unitario normal a la superficie y $dS$ es un elemento diferencial de área de la superficie.
Cabe mencionar que la ecuación 2.16 tiene en cuenta los efectos viscosos y, por lo tanto, la fricción sobre el perfil. Si bien, estos efectos pueden ser medidos igualmente mediante instrumentación, no es muy común y en su lugar se dejan a la metodología de CFD.
Finalmente, una vez se obtienen los coeficientes aerodinámicos, estos suelen ser graficadas en función del ángulo de ataque, como se puede ver en la Figura 2.11. Estas curvas se usan además para medir el rendimiento de un cierto perfil alar o, proporcionar información útil sobre dicho perfil, como el ángulo de ataque máximo antes de entrar en pérdida, el ángulo de ataque en el que el perfil alcanzará su máximo rendimiento (mayor sustentación, menor arrastre), e incluso sirve como guía para elegir entre una geometría u otra según el objetivo de la aeronave.
Las curvas de rendimiento, además de cambiar entre distintas geometrías de perfiles, también lo hacen para distintos números de Reynolds, por lo que un solo perfil

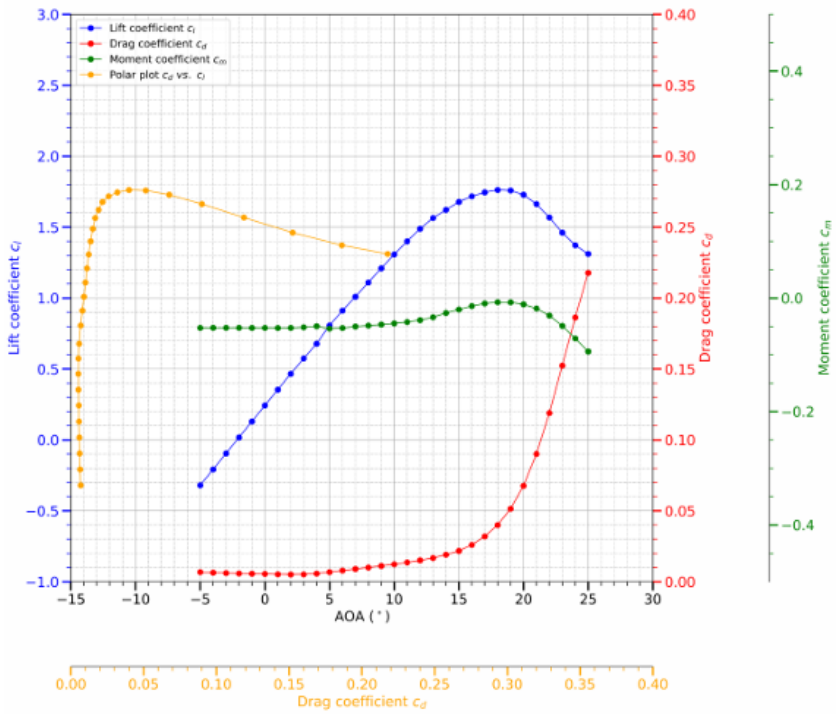


Figura 2.11: Resultados del perfil NACA2412, Re = 3,000,000, donde la curva azul, roja y verde representan el coeficiente de sustentación, arrastre y momento respectivamente, en función del ángulo de ataque y, la curva amarilla representa el $C_d$ en funcicón del $C_l$ [5].

puede tener distintas curvas para distintas velocidades, números de Mach o tamaños distintos de la geometría, esto se observar en la Figura 2.12, donde, para el perfil NACA 4415 se pueden observar distintas curvas de $C_L$ y $C_D$ según su Reynolds.
Como se puede ver, las pruebas en túnel de viento son muy usadas aún hoy en día, en las distintas industrias donde se estudia la aerodinámica, sin embargo, su implementación suele ser costosa al requerir instalaciones e instrumentación especializada, además de modelos en sí mismos con los que hacer pruebas, por lo que, otra de las grandes alternativas para ahorrar recurso, tiempo y dinero es la metodología de CFD.

### 2.2.2. Simulaciones de CFD

La Dinámica de Fluidos Computacionales (CFD), es el análisis de sistemas que involucran el flujo de fluidos, transferencia de calor y fenómenos asociados (como

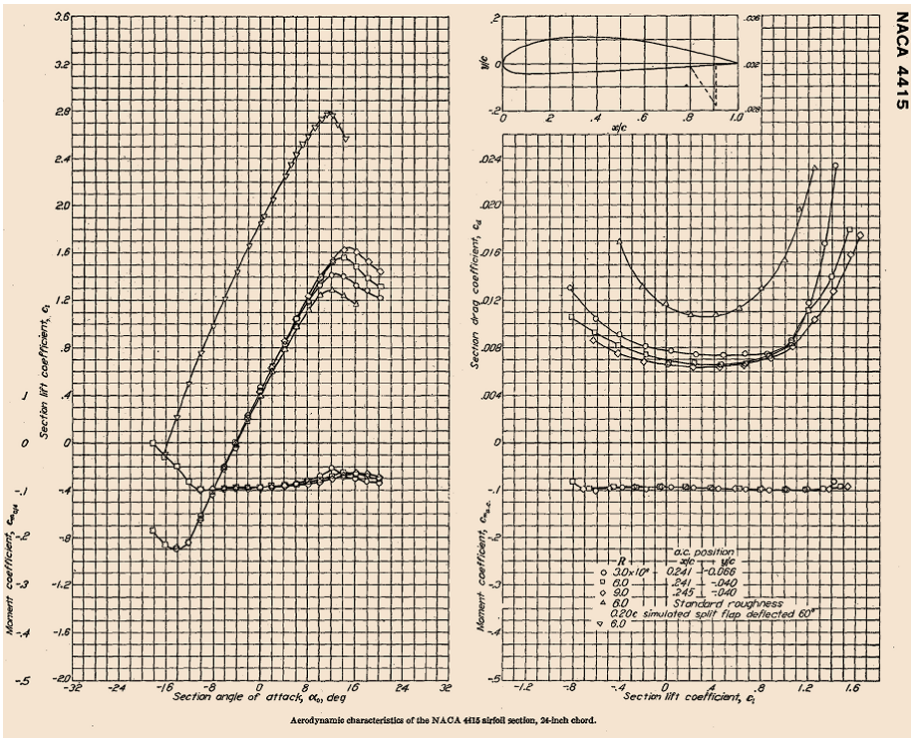


Figura 2.12: Curvas de Cl y Cd en función del número de Reynolds para el perfil NACA 4415 [5].

reacciones químicas), mediante simulaciones basadas en computadora [9]. Aunque esta técnica tiene una gran variedad de aplicaciones en distintas industrias, ha tenido un especial auge en la industria aeroespacial desde la década de 1960, tanto en el diseño, investigación y fabricación de aeronaves, motores y componentes de estos mismos.

Los resultados de la dinámica de fluidos computacional son directamente análogos a los resultados del túnel de viento, esto debido a que ambos representan conjunto de datos para una configuración de flujo dada a diferentes números de Mach y número de Reynolds [8], la principal diferencia consiste en la reducción significativa de las instalaciones necesarias para la realización del método, pues, el gran aumento de velocidad computacional en estos últimos años hace posible que no se requiera un hardware especialmente costoso para diferentes tipos de simulaciones. Sin embargo, sí es necesario mencionar que para simulaciones más complejas y, con un gran número de elementos, el costo se eleva de manera sustancial, además, se debe considerar que, en la mayoría de los softwares especializados en CFD, se debe incluir el costo

de una licencia anual para las opciones más avanzadas.
Aún con esto, el CFD tiene claras ventajas sobre la experimentación tradicional, pues ofrece una reducción significativa tanto en el tiempo como en los costos de desarrollo en nuevos diseños, además de capacidad para realizar experimentos en cualquier tipo de variedad de condiciones o, incluso en condiciones peligrosas sin riesgo alguno, y, además, un nivel prácticamente ilimitado de detalle en los resultados [8].
Ahora bien, al igual que es análogo a un túnel de viento, se puede mencionar que para obtener los resultados necesarios, como presión y velocidad, su metodología es análoga a los principios de la aerodinámica, pues este igualmente se rige por los principios fundamentales de la conservación de la masa, la segunda ley de Newton y la conservación de la energía. Estos principios se expresan en términos de ecuaciones matemáticas, que en su forma más general son ecuaciones integrales o diferenciales parciales, como las ecuaciones de Navier Stokes, por lo que, para su resolución, se remplazan por ecuaciones algebraicas discretizadas, que a su vez, se resuelven para valores de campo en puntos discretos de tiempo y o el espacio [8].
Con el fin de explicar paso a paso la metodología de CFD, es posible dividirla en tres elementos principales que suelen estar presentes en todos los paquetes de CFD comerciales: el procesador, el solucionador y el postprocesador. Aunque, a continuación se explicara en que consiste cada uno de ellos de forma general, se dará un especial énfasis en Ansys Fluent para la ejemplificación, al ser la paquetería utilizada en este trabajo.
El preprocesamiento consiste en introducir un problema de flujo en el programa de CFD a través de una interfaz y, en trasformar esta entrada en la forma adecuada para el solucionador [9]. Dentro de esta tapa se incluye:

- Definición de la geometría. Dominio computacional.
- Generación de malla. Subdivisión del dominio en pequeños subdominios (volumen de control finito para Ansys Fluent).
- Especificación de condiciones de frontera.

Como se mencionó anteriormente, es aquí donde los parámetros geométricos son definidos, de igual forma que el dominio, sin embargo, para resolver los principios físicos, en lugar de considerar todo el campo de flujo a la vez, se utiliza el volumen de control finito, en el cual, la atención se limita a una región finita del propio volumen y a la interacción del fluido con su superficie, esto se ve ejemplificado en la Figura 2.13, donde un elemento infinitesimal es lo que permite la resolución de las ecuaciones diferenciales parciales [8].
Es mediante el uso de este método con el que es posible generar el mallado, pues, al

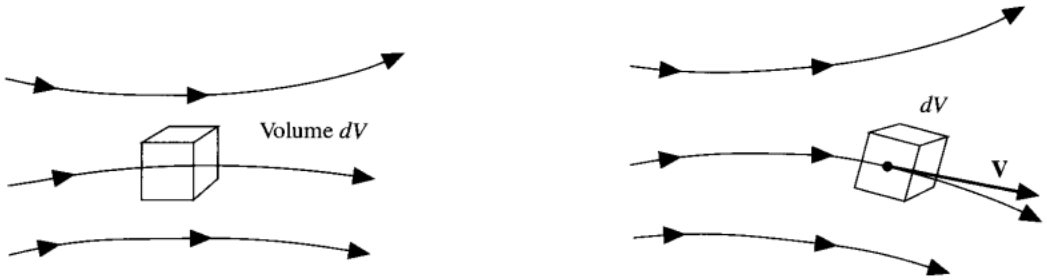


Figura 2.13: Elementos del fluido infinitesimal fijado en el espacio y el fluido mindoniense a través de el y el mismo elemento pero moviéndose junto con el fluido con una velocidad V igual a la del flujo. [8].

aplicarlo a todo el dominio, se crea una red en la cuál cada nodo es en sí mismo es un volumen infinitesimal, con el cuál se obtienen los resultados esperados. Aunque, existen diferentes tipos de malla, un ejemplo común se encuentra en la Figura 2.14, donde se tiende a una malla estructurada que capture con mayor precisión la curvatura del perfil y sea más fina en los puntos cercanos de la geometría.

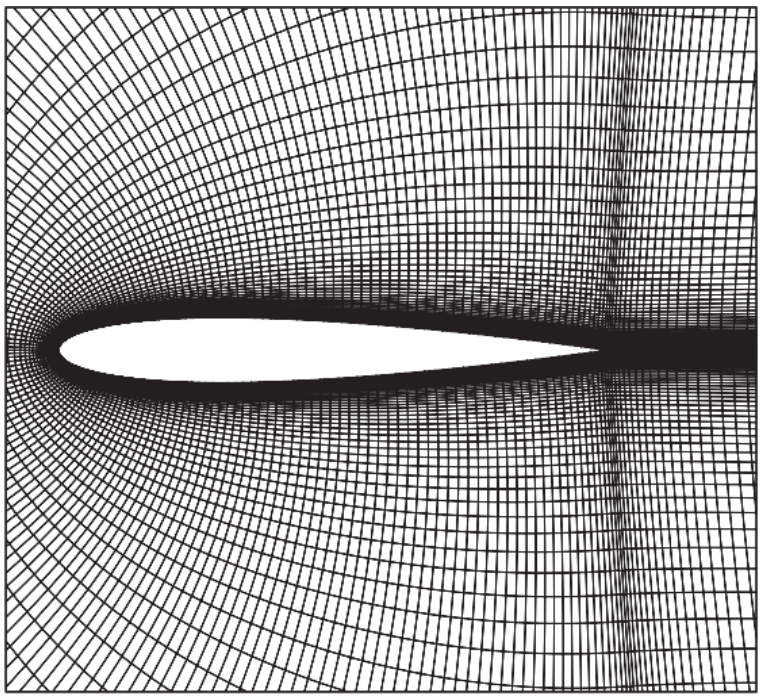

Figura 2.14: Ejemplo de mallado en un perfil aerodinámico. [9].

Continuando con la metodología de CFD, el siguiente paso es el solucionador, es aquí donde se ejecutan los cálculos para la solución numérica utilizando el ya mencionado método de volúmenes finitos. De forma resumida, los pasos para el algoritmo numérico son:

1. Integración de las ecuaciones gobernantes del flujo sobre cada volumen de control.

2. Discretización: Conversión de las ecuaciones integrales en un sistema de ecuaciones algebraicas.

3. Solución del sistema algebraico mediante un método iterativo.

Para Ansys fluent el solucionador se encuentra en el apartado de *setup*, por lo que, es igualmente en este apartado donde se definen las condiciones para la simulación, algunos ejemplos de estas son: la velocidad, presión, condiciones de deslizamiento, resultados a obtener o el modelo de turbulencia a usar. Este último es importante tenerlo en cuenta, pues, con él es posible cambiar radicalmente el tipo de simulación que se ejecute, dado que, al definir un flujo turbulento como el instante donde la velocidad y todas las demás propiedades de este varían de forma aleatoria [9], es imperativo un modelo con el que simular dichas variaciones de forma eficiente.
El modelo más comúnmente utilizado en aeroespacial, y, que además fue diseñado específicamente para ello, es el Spalart-Allmaras, que consta de una sola ecuación que resuelve una función de transporte modelada para la viscosidad turbulenta cinemática (*eddy viscocity*), este modelo ha demostrado ofrecer buenos resultados en capa límite sometida a gradientes de presión y para aplicaciones en turbomáquinas [24].
Finalmente, en el último paso de la metodología, se tiene al posprocesamiento. Este suele ofrecer herramientas para la visualización de datos, lo cual permite, entre algunas otras cosas:

- Mostrar la geometría del dominio y la malla.
- Graficar vectores de velocidad o flujo.
- Realizar gráficos de contornos en líneas y sombreados.
- Realizar seguimiento de partículas.

Ejemplos de visualización, para contornos de presión y velocidad en Ansys Fluent, se observan en las Figuras 2.15 y 2.16.

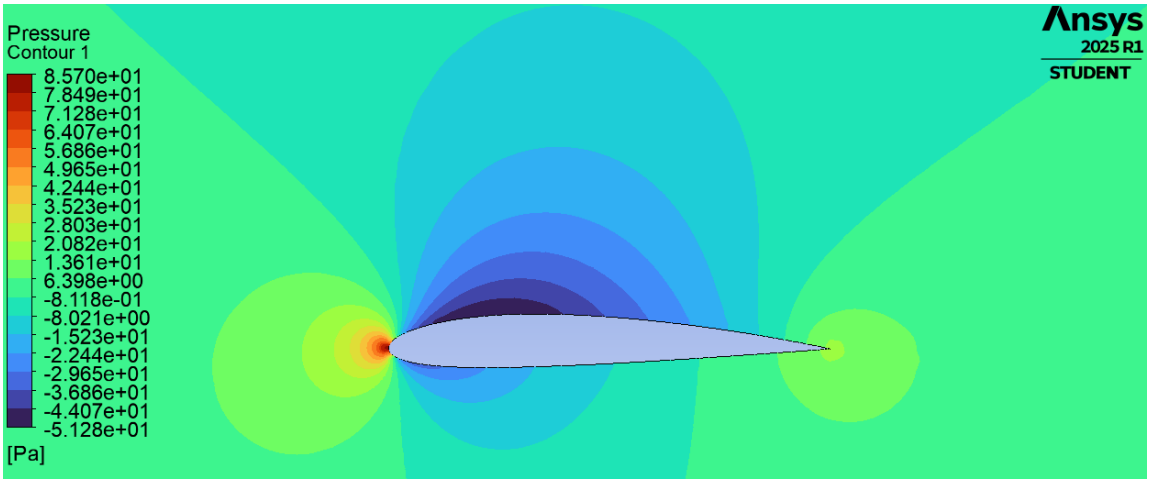


Figura 2.15: Ejemplo de contorno de presión en Ansys Fluent para NACA2412.

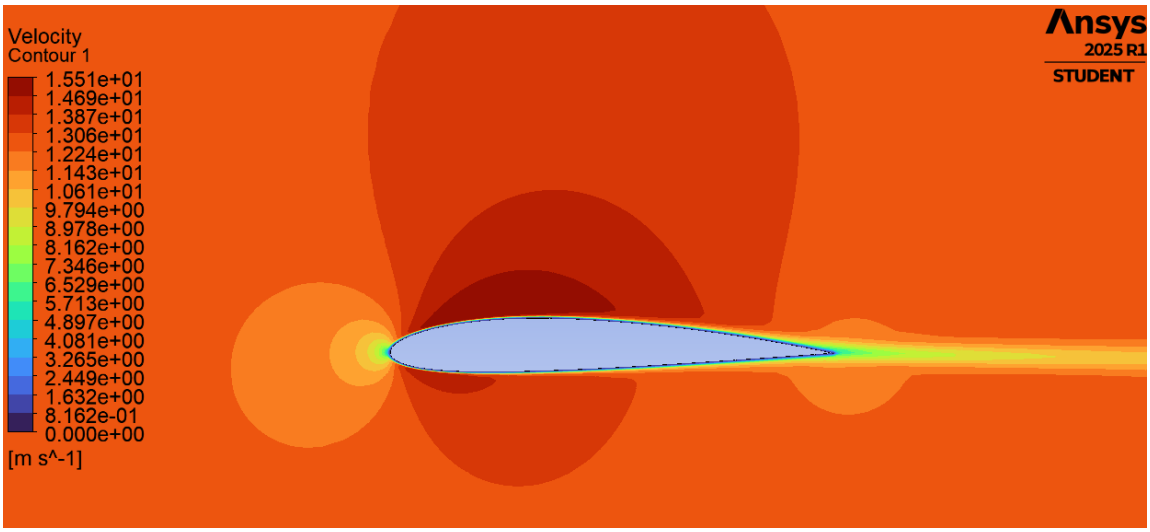


Figura 2.16: Ejemplo de contorno de velocidad en Ansys Fluent para NACA2412.

Además de las herramientas visuales, igualmente es posible obtener los datos numéricos de presión, velocidad o cualquier otro parámetro que se halla definido en el solucionador, esto en términos de cada coordenada en los nodos de la malla.
Como se pudo notar, existen similitudes muy claras en la experimentación con túnel de viento y las simulaciones con CFD, al ambos regirse por los principios fundamentales de la aerodinámica, sin embargo, en el CFD se encuentran ventajas claras con respecto a los costos y tiempo para el diseño, este se suele usar como una primera

iteración en todo un proceso de producción. Aún con esto, existen los problemas ya mencionados para las simulaciones, al requerir aún un hardware competente, la licencia de software y, una gran cantidad de tiempo para simulaciones de gran complejidad, es por ello que hoy en día existen otro tipo de aproximaciones con el uso de la creciente investigación en las redes neuronales artificiales.

## 2.3. Redes neuronales artificiales

Con el fin de poder entender qué es y cómo funciona una red neuronal, primero se debe ampliar el panorama y explicar conceptos anteriores que involucran a las técnicas de la inteligencia artificial. Para ello, el diagrama de la Figura 2.17 sirve como guía, en el se ejemplifica cómo el aprendizaje profundo (*deep learning*) no es más que un subconjunto de métodos dentro de las herramientas del aprendizaje automático (*machine learning*), donde, dichos métodos se enfocan principalmente en el uso de redes neuronales artificiales y, al mismo tiempo, técnicas de ambos conjuntos son utilizadas para la inteligencia artificial. Sin embargo, no toda la inteligencia artificial se basa en el aprendizaje automático y no todo el aprendizaje automático se utiliza para la inteligencia artificial.

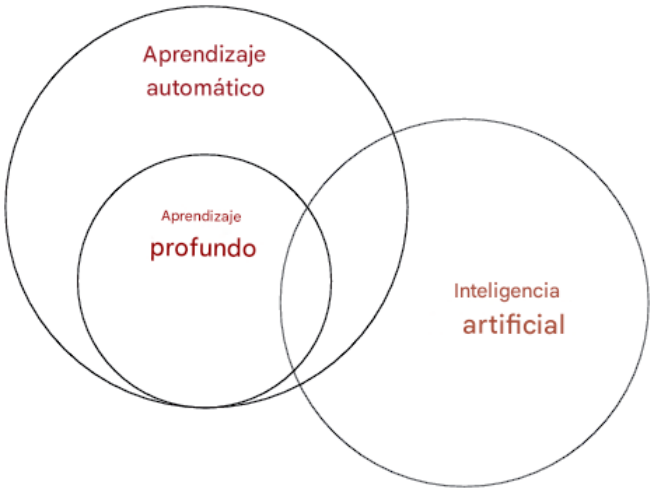


Figura 2.17: Diagrama de conjuntos del aprendizaje automático [10].

Con lo anterior dicho, el aprendizaje automático se define como un campo dentro de las ciencias de la computación, donde la máquina o computadora aprende a realizar

tareas para las que no fue explícitamente programada [10], es decir, la máquina observa un patrón y aprende a imitarlo de forma directa o indirecta.
Esta última diferenciación, se refiere a la forma de aprendizaje que el algoritmo emplea, es decir, aprendizaje supervisado o no supervisada. En la primera, la máquina aprende directamente de datos ya etiquetados o ya clasificados, por lo cual, en los datos con los que se alimenta al algoritmo, ya se incluye la solución deseada, de esta manera el modelo se autorregula para llegar a dicha solución. Esta es la metodología más utilizada y la gran mayoría de tareas son realizadas de esta forma. Sin embargo, para problemas en los que no se cuenta con muchos datos, o estos mismos no están etiquetados, se usa la no supervisada, pues etiquetarlos suele ser un proceso tardado y costoso. Además, la no supervisada es útil para encontrar patrones no detectados, pues, al dejar a la máquina aprender por sí sola sin indicarle cual es el resultado final, puede formar sus propias agrupaciones o encontrar características compartidas que no se visualizan a simple vista.
Ya se comentó un poco de la metodología detrás de la forma de abordar problemas con *machine learning*, sin embargo, con el fin de visualizarla de mejor forma, se puede observar la Figura 2.18. En esta Figura se muestra la estructura a seguir del algoritmo para resolver cualquier tarea utilizando el aprendizaje automático, donde el estudio del problema y la recopilación datos son pasos esenciales, además, tal como se mencionó anteriormente, este es un algoritmo que se evalúa y se corrige de forma constante durante la implementación. Aunque, cada uno de los puntos se abordarán en mayor profundidad a lo largo de este trabajo, es de gran utilidad visualizar la secuencia completa para una mejor referencia.
Obviando el primer punto de la Figura 2.18, a continuación se describe el segundo paso del algoritmo, los modelos de ML (*machine learning*). Aunque existe una gran diversidad en ellos, tales como regresión lineal, polinomial, logística, arboles de decisiones, bosques aleatorios, entre otros, este trabajo se centrará en el ya mencionado, y, el probablemente más poderoso, aprendizaje profundo y el uso de las redes neuronales.
Las redes neuronales artificiales (RNA) tienen su nacimiento en 1943 por el neurofisiólogo Warren McCulloh y el matemático Walter Pitts. Ellos presentaron por primera vez un modelo computacional simplificado que se asemejaba a las neuronas biológicas y cómo estas trabajan juntas en el cerebro animal para realizar cálculos complejos usando lógica proposicional [11], desde entonces ha habido mayor o menor velocidad de avance en las investigaciones debido en gran parte al limitado poder computacional, cosa que hoy en día ya no es un impedimento, por lo que las limitaciones prácticas desaparecieron para dar un gran paso al progreso en la investigación de esta rama.

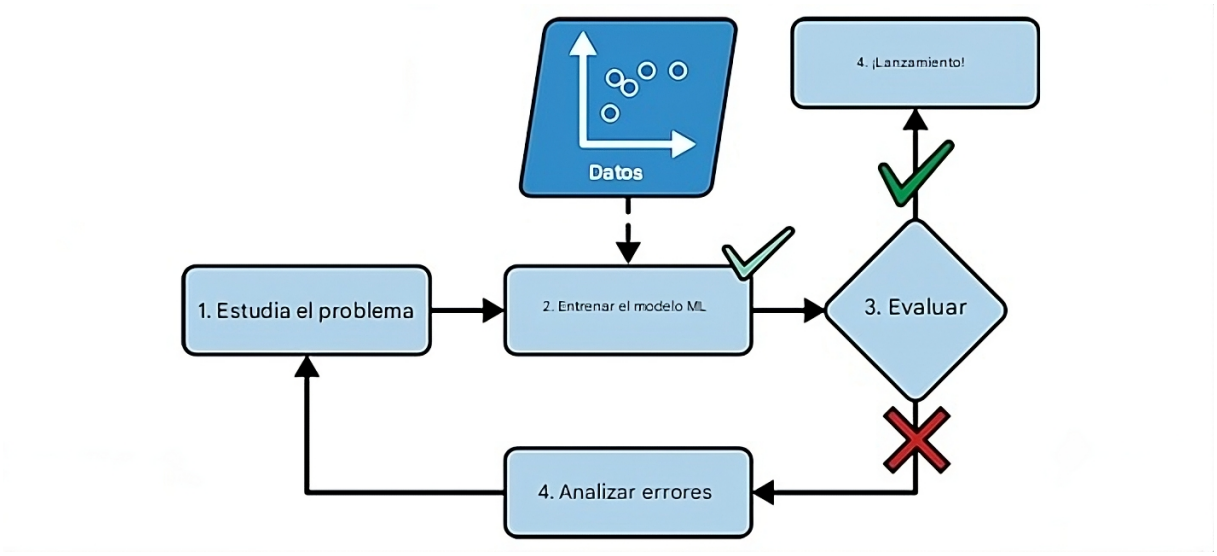


Figura 2.18: Metodología del aprendizaje automático [11].

Si bien, ya se mencionó la inspiración en sus homónimas biológicas, las RNA actuales distan mucho en su funcionamiento real de estas mismas, sin embargo, sí comparten características similares como que ambas parten de una unidad fundamental, una sola neurona, que se conecta a miles o millones de ellas en secuencia y en capas para poder realizar cálculos a una gran velocidad. Aunque, posteriormente se definirá con mayor profundidad la arquitectura tradicional de una RNA, es posible hablar del que se considera su unidad básica, el perceptrón.
Pese a que, el perceptrón sea considerado una forma básica, esta compuesto de neuronas artificiales llamadas TLU (*threshold logic unit*), en las cuales, las entradas y salida son números y cada conexión esta asociada a un peso; de tal forma que, estas realizan una suma ponderada de los valores en la entrada, más un término de sesgo, para finalmente aplicar una función escalón [11], esta operación se observa en la ecuación 2.17 y el esquema de una TLU se muestra en la Figura 2.19. La función escalón tiene como nombre general 'función de activación' y, esta ha sido remplazada por mejores alternativas para un mejor funcionamiento.

$$w_1 x_1 + w_2 x_2 + ... + w_n x_n + b = W^T X + b \tag{2.17}$$

donde $w_n$ es cada uno de los pesos asociados a las entradas $x_n$, $b$ es el término de sesgo y $W$ y $X$ son las matrices de pesos y entradas.
Con lo anterior dicho, el perceptrón se define como la combinación de una o más TLUs organizadas en una sola capa, donde cada TLU esta conectada a cada una

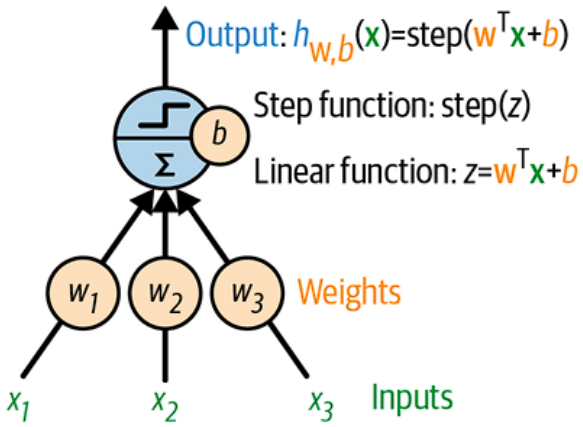


Figura 2.19: Esquema de una TLU [11].

de las entradas. El primer nivel se le conoce como capa de entrada y, dado que la capa única de TLUs produce en sí misma la salida final, se conoce como capa de salida. Un ejemplo de perceptrón con dos entradas y tres salidas esta representado en la Figura 2.20. Con ello finalmente se define la arquitectura más simple de red neuronal, el perceptrón multicapa.

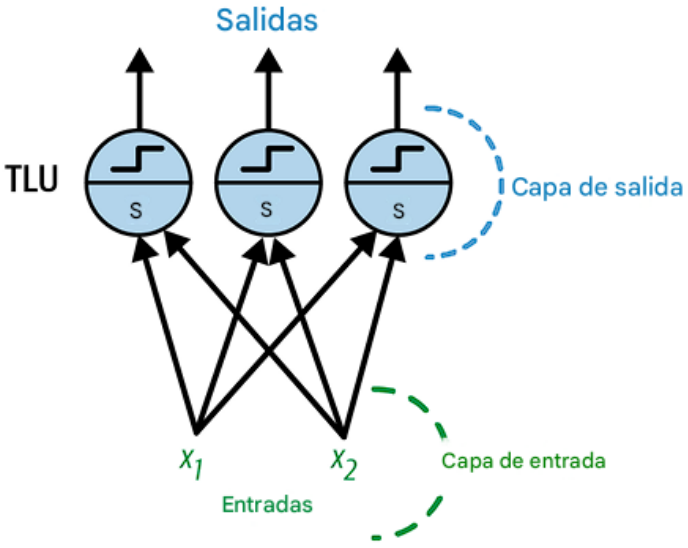


Figura 2.20: Arquitectura de un perceptrón con dos entradas y tres neuronas de salida [11].

### 2.3.1. Arquitectura de una red

El perceptrón multicapa (*Multilayer Perceptron*, MLP, por sus siglas en inglés) esta compuesto por una capa de entrada, una o más capas intermedias de TLUs, llamadas capas ocultas, y una capa de salida, también formada por TLUs. El esquema de dicha arquitectura se muestra en la Figura 2.21.

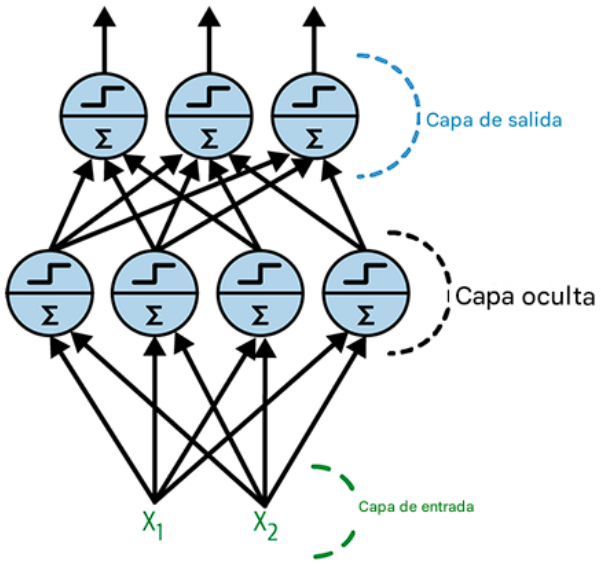


Figura 2.21: Arquitectura MLP [11].

Desde esta primera configuración, es donde se desprenden algunas otras variaciones que pueden cambiar la forma en que las redes se conectan entre ellas, sin embargo, el uso de estas variaciones suelen ser para tareas muy específicas y, aunque sus especificaciones se escapan del alance de este trabajo, si es posible mencionar algunos ejemplos populares de estas arquitecturas. Tales como las llamadas redes neuronales convolucionales (CNN) cuya utilidad radica en el reconocimiento y procesamiento de imágenes. Igualmente, se destacan las redes neuronales recurrentes (RNN), las cuales son capaces de realizar diversas tareas donde se involucre detectar patrones de datos históricos para predecir datos del futuro [11], como las tareas de el procesamiento del lenguaje natural, es por ello que uno de sus usos se halla en los chatbots de inteligencia artificial actuales.
Continuando con las MLP, aunque su arquitectura es siempre constante, si existe una gran variedad de configuraciones con las que se puede alterar su función según el arreglo presentado. El nombre que se utiliza para cada uno de los valores configurables de la RNA es el de hiperparámetros. La función de estos es controlar aspectos

como la velocidad de aprendizaje, el número de datos procesados por cada ciclo de tiempo, el rendimiento de la red, las métricas, entre muchos otros.
Los primeros hiperparámetros que se suelen determinar es el número de capas ocultas y de neuronas por cada una de las capas. Cuando una RNA tiene una pila profunda de capas ocultas se le llama red neuronal profunda [11], es precisamente por esto que se le llama aprendizaje profundo cuando se trata de redes neuronales, incluso si la red no es profunda en sí misma. No existe un número determinado de capas o neuronas a utilizar, pues esto depende en gran medida de la tarea, aunque normalmente, si es una tarea compleja, se utiliza una red con un gran número de capas ocultas y neuronas. Sin embargo, para la primera capa (capa de entrada) si existe un número determinado de neuronas, estas dependen completamente del número de características de entrada que se utilicen. Lo mismo ocurre con la capa de salida, cuyo número de neuronas está condicionado por la naturaleza del problema y el tipo de salida esperada. Por ejemplo, si se trata de una red cuyo numero de neuronas por capa oculta es de 8 y se tienen 4 características de entrada, su forma se define como $(4 \times 8)$, mientras que, si tiene una sola salida, su forma sería $(8 \times 1)$. Esto se ilustra con la Figura 2.22, donde se muestra un esquema de RNA habitual.

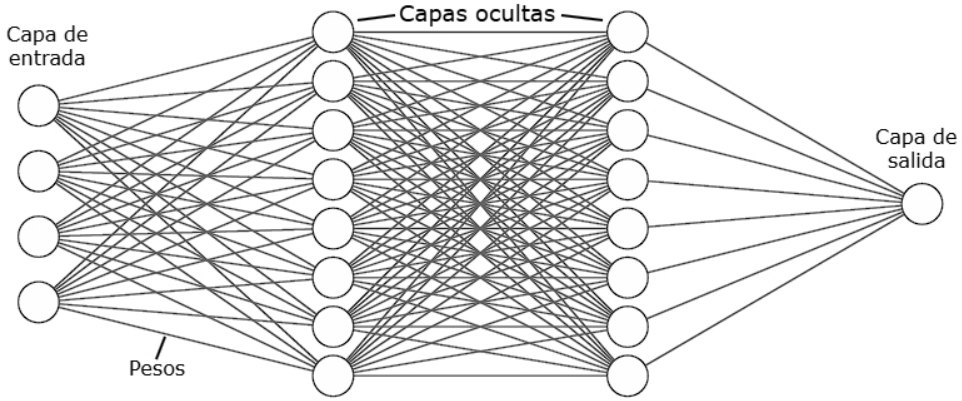


Figura 2.22: Red neuronal artificial [12].

Con el fin de describir algunos otros de los hiperparámetros más importantes, primero se explicará con mayor detalle el algoritmo usado para el entrenamiento de redes neuronales en la siguiente sección.

### 2.3.2. Funcionamiento de una red neuronal

La técnica que permite a una red neuronal aprender y realizar tareas de manera eficiente se le conoce como retropropagación, con dicha técnica se puede determinar la forma en que se debe ajustar cada peso y cada sesgo para reducir el error de la red neuronal. De esta forma, con solo dos pasadas (una hacia adelante y una hacia atrás), es capaz de calcular los gradientes del error neuronal con respecto a cada uno de los parámetros del modelo [11].
A continuación, se explicará con mayor detalle el proceso de retropropagación, así como el funcionamiento general de una red neuronal artificial (RNA):

1. Las entradas deben ser procesadas para adaptarse a la estructura de la red neuronal. Cualquier tipo de entrada debe convertirse en valores numéricos y someterse a adecuaciones pertinentes, como la normalización, limpieza y escalamiento de los datos, entre otras. Igualmente, los pesos son inicializados, aunque pueden asignarse aleatoriamente, existen algoritmos especializados que permiten una inicialización más eficiente que evitan problemas durante el entrenamiento, que se configuran como hiperparámetros, por ejemplo, los inicializadores He, Glorot y LeCun.

2. El entrenamiento se realiza procesando un mini-lote de entradas por iteración, recorriendo el conjunto completo de datos múltiples veces. Cada recorrido completo se le llama época. El número de instancias por mini-lote puede configurarse como un hiperparámetro. El tamaño afecta directamente en la velocidad de entrenamiento y suele ajustarse en función de la capacidad de la memoria RAM disponible.

3. Cada mini-lote entra a la red a través de la capa de entrada. Para cada instancia del mini lote, el algoritmo calcula la salida de todas las neuronas en la primera capa oculta. El resultado se transmite a la siguiente capa, donde nuevamente se calcula su salida, y así sucesivamente hasta que se obtiene la salida de la capa de salida. Entre capa y capa, se aplica una operación intermedia llamada función de activación, este hiperparámetro es útil para poder resolver problemas no lineales. Ya se ha mencionado la función escalón como la más básica; sin embargo, diversos estudios han identificado alternativas más eficaces, como tanh, ReLU o Sigmoid. Estas y sus derivadas se ilustran en la Figura 2.23.

4. El algoritmo evalúa el error en la salida, mediante una función de pérdida la cual compara el valor objetivo (*Target value*) con la salida de la red, devolviendo una medida del error. La función de pérdida es otro hiperparámetro a tomar

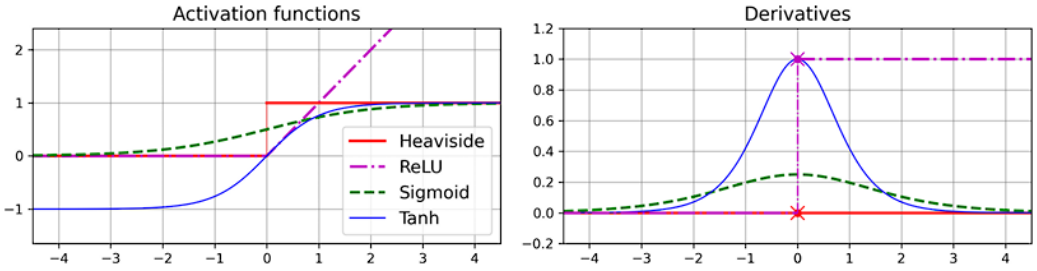


Figura 2.23: Funciones de activación (izquierda) y sus derivadas (derecha) [11].

en cuenta, algunas de las más comunes son el error cuadrático medio (MSE) o el error absoluto medio (MAE), ecuaciones 2.18 y 2.19, repectivamente.

$$MSE = \frac{1}{n}\sum_{i=1}^{N}(y_i - \bar{y}_i)^2 \tag{2.18}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{N}|y_i - \bar{y}_i| \tag{2.19}$$

donde $n$ es el número de instancias, $y_i$ es el resultado esperado y $\bar{y}_i$ es la salida de la red.

5. A continuación, el algoritmo calcula la contribución de cada sesgo de salida y de cada conexión hacia la capa de salida al error total. Este análisis se realiza de forma analítica mediante la aplicación de la regla de la cadena, lo que permite propagar el error hacia atrás a través de la red.

6. Posteriormente, el algoritmo determina qué proporción del error total se atribuye a cada conexión en las capas inferiores, aplicando nuevamente la regla de la cadena para propagar el error hacia atrás, hasta alcanzar la capa de entrada. Como se explicó anteriormente, este proceso inverso permite calcular de manera eficiente el gradiente del error en todos los pesos y sesgos conectados en la red, propagando el gradiente del error hacia atrás a través de la red (de ahí el nombre del algoritmo: retropropagación del error).

7. Finalmente, el algoritmo ejecuta un paso de *gradient descent*, explicado más adelante, con el fin de ajustar todos los pesos de conexión en la red, usando los gradientes de error previamente calculados durante la retropropagación

Para una explicación más profunda de todos los hiperparámetros mencionados, se puede consultar [11].
Con la explicación anterior, ya se mencionaron la mayoría de los hiperparámetros que se pueden considerar en una RNA, sin embargo, aún hay algunos otros más que definir, pues estos igualmente suelen mejorar el rendimiento general de la red.
El primero de ellos es el optimizador. Anteriormente se mencionó el optimizador por defecto, el descenso por gradiente *Gradient Descent*, el cual es capaz de encontrar soluciones óptimas para una amplia gama de problemas. La idea general del *Gradient Descent* es ajustar los pesos de manera iterativa para minimizar una función de costo, dando pequeños pasos uno a uno, esto lo hace al medir el gradiente local de la función con respecto al vector de parámetros (pesos en nuestro caso) $\theta$, moviéndose en la dirección del gradiente [11], una representación gráfica de este proceso se encuentra en la Figura 2.24. El tamaño de los pasos esta definido por otro hiperparámetro, llamado *learning rate* o taza de aprendizaje.
La manera en que se calcula el gradiente en cada paso es mediante el uso de la derivada parcial de la función de costo, por ejemplo, en el caso donde se use el MSE, es posible calcular todas las derivadas de cada parámetro utilizando la siguiente ecuación:

$$\nabla_\theta MSE(\theta) = \begin{pmatrix} \frac{\partial}{\partial\theta_0} MSE(\theta) \\ \frac{\partial}{\partial\theta_1} MSE(\theta) \\ \vdots \\ \frac{\partial}{\partial\theta_n} MSE(\theta) \end{pmatrix} = \frac{2}{m} X^\top (X\theta - y) \tag{2.20}$$

donde $\nabla_\theta MSE$ es el vector gradiente, $\theta$ es el vector de parámetros (pesos), $X$ es la matriz de entradas, $m$ el número total de instancias y $y$ es el resultado objetivo.
Con el gradiente calculado, el paso de *gradient descent* se realiza de la siguiente manera:

$$\theta^{(\text{next step})} = \theta - \eta \nabla_\theta MSE(\theta) \tag{2.21}$$

donde $\theta^{(\text{next step})}$ es el paso con el que se ajustan los parámetros y $\eta$ es la taza de aprendizaje.
Si bien, el *Gradient Descent* es un algoritmo rápido y eficiente en su tarea, existen modificaciones de este mismo que pueden acelerar en gran medida la velocidad de la arquitectura, estas modificaciones se basan principalmente en el ajuste de la tasa de aprendizaje según el comportamiento de las neuronas en cada mini-lote. Algunos de los más usados son el Nesterov, Adam y las variantes de estos mismos.
Otro hiperparámetro es la regularización a usar, esto hace referencia a varios métodos que intervienen en el aprendizaje de la RNA con el fin de que aprenda a generalizar

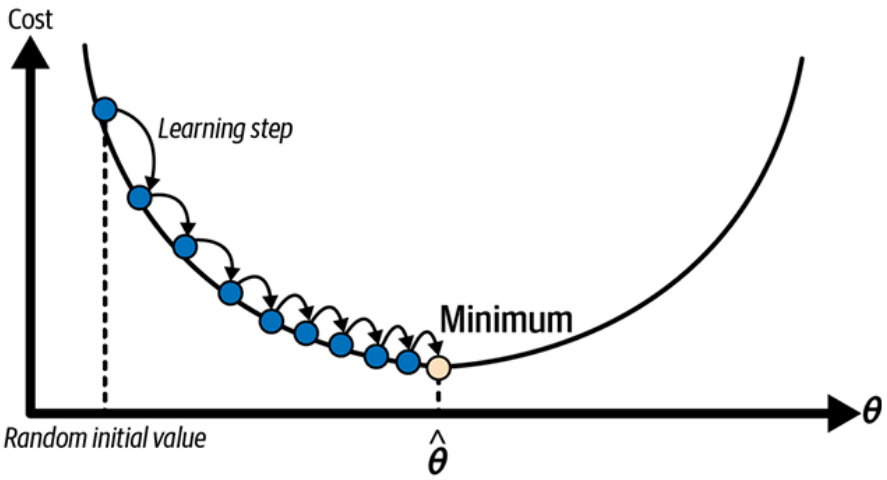


Figura 2.24: Representación del *Gradient Descent*, donde los parámetros se inicializan aleatoriamente y se modifican repetidamente para minimizar la función de costo; el tamaño del paso es proporcional a la pendiente de la función de costo, por lo que el paso es cada vez más pequeño mientras se aproxima al mínimo [11].

lo mejor posible, es decir, que aunque use los datos de entrenamiento como base para aprender patrones, no se quede solo con los presentes en el set de datos, si no que pueda predecir de forma eficiente para cualquier nueva instancia con la que trabaje. El error producido por no generalizar bien se le conoce como sobreajuste y, algunas de las técnicas que se utilizan para evitar este error son el *early stopping* o el *dropout*. Finalmente, el último de los hiperparámetros a definir es la métrica, como su nombre lo dice, se utiliza para medir el desempeño de la RNA, es decir, que tan cerca está de los valores objetivo. Comparte técnicas con la función de pérdida, pues puede ser usado de igual forma el MSE, el MAE, entre otros. Algo que mencionar es que, por lo general, el set de datos original se divide en set de entrenamiento y set de validación; la función de este último es medir el desempeño general al usar instancias en las que no haya entrenado, por lo que, usualmente se tiene la métrica de validación y la de entrenamiento, esto se muestra con más detalle en la metodología.
En suma, con lo expuesto hasta ahora, se cuenta con una visión integral del funcionamiento de una red neuronal, así como de los principales ajustes que deben considerarse para su implementación efectiva. Este enfoque permite abordar una amplia variedad de problemas en diversas industrias, destacando especialmente su aplicabilidad en el sector aeroespacial, donde el análisis predictivo, la optimización de procesos y la detección de patrones complejos son de particular relevancia.

### 2.3.3. Uso de redes neuronales en aerodinámica y aeroespacial

Las redes neuronales artificiales han tenido dos principales propósitos en los campos de la aerodinámica y aeroespacial, el primero es la parte estructural y la optimización del diseño, el segundo es para suplir el uso de simulaciones y experimentos en la parte aerodinámica, aunque, en esta segundo, el propósito último de sustituir dichos métodos suele ser igualmente el de optimizar la parte geométrica. Es por ello que en la gran mayoría de investigaciones en este rubro se tienen estos dos objetivos entrelazados.

Un primer gran ejemplo se muestra en [15], donde hace una revisión del estado del arte en el uso de la inteligencia artificial para la optimización aerodinámica de perfiles alares, revisando el proceso para llegar a ello al parametrizar la geometría, mostrando, además, que tipo de modelos de aprendizaje automático hacen posible dicha mejora estructural e, igualmente, que modelos se utilizan para la predicción de coeficientes, flujos y turbulencia. Esto para, finalmente, mostrar un ejemplo de su uso al optimizar la estructura de un ala.

Siguiendo esta rama se tiene a [25], donde, de igual forma, con el fin de optimizar la forma aerodinámica de un perfil, hace uso de las CNN, primeramente para filtrar formas geométricas no convencionales en este ámbito para seguidamente aplicar una mejora en su rendimiento aerodinámico.

De manera similar a la aplicación anterior, tenemos a [26] y [12], en las que, se utilizan como datos de entrenamiento las coordenadas geométricas del perfil y los coeficientes aerodinámicos, de forma que, como resultado, se puede hacer un ajuste a su diseño y encontrar una geometría no explorada anteriormente que resulte beneficiosa. De forma más simple, [13] y [27] hacen uso de datos recolectados de perfiles NACA para la predicción rápida de los mismos coeficientes en una arquitectura RNA más sencilla.

Para finalizar con estos ejemplos. Un caso especial se encuentra en [16], donde hace uso de una arquitectura especial de red llamada: aprendizaje automático basado en la física (Physics-Informed Neural Networks, PINN, por sus siglas en inglés). Esta arquitectura funciona de forma muy similar a la simulación CFD convencional, pues aprovecha la diferenciación automática encontrada en el algoritmo de la retropropagación para resolver directamente las ecuaciones de Navier-Stokes, en cada coordenada de una malla dada. Con ello, puede obtener contornos de velocidad y presión para un perfil aerodinámico muy simple a diferentes ángulos de ataque, con la diferencia que lo hace de una forma mucho más rápida que la simulación.

Como se puede ver, existe un gran potencial en el uso del aprendizaje profundo para el campo de la aerodinámica y el campo aeroespacial, en la cual, hay una gran diver-

sidad en las metodologías que se utilizan, pues, la variación de datos y parámetros en cada una de ellas, permiten encontrar resultados y patrones distintos cada vez. El uso de las RNA permiten, adicionalmente, una reducción de costos y un aumento de velocidad en la experimentación que con los métodos convencionales no son posibles al día de hoy, logrando acceder a una mejora continua y rápida en el diseño y optimización de formas aerodinámicas y estructurales.

# 3 Metodología

Dado que, el fin último de este trabajo es la implementación y entrenamiento de una red neuronal, se usará la guía para proyectos de aprendizaje automático presentada en [11], pues esta contiene una lista ordenada y detallada de actividades que se deben realizar para cualquier tipo de tarea en este campo. Aunque cada uno de los pasos principales contiene a su vez pasos secundarios igual de detallados, no se realizará uno por uno necesariamente, pues como lo menciona el mismo autor, esta guía se sigue solo con lo que se necesite en el proyecto.
Con lo anterior dicho, los ocho pasos principales considerados son los siguientes:

1. Delimitar el problema y ver el panorama completo.
2. Obtener los datos.
3. Explorar los datos para obtener información.
4. Preparar los datos para exponer mejor los patrones principales en los algoritmos de *machine learning*.
5. Explorar varios modelos diferentes y destacar los mejores.
6. Ajustar los modelos y combinarlos en una gran solución.
7. Presentar la solución.
8. Lanzar, monitorizar y darle mantenimiento al sistema.

## 3.1. Modelos de referencia (perfil de referencia)

Siguiendo el primer punto, se estudió el panorama completo para caracterizar un perfil aerodinámico, en donde se determinó que es necesario obtener la presión alrededor del mismo para calcular los coeficientes de las fuerzas aerodinámicas, y,

así mismo, dichos coeficientes están definidos por la forma del perfil, la velocidad del flujo y el ángulo de ataque.
Es por lo anterior que, con el fin de delimitar el problema, se propuso utilizar un solo perfil NACA a distintas velocidades y a distintos ángulos de ataque utilizando la metodología de la dinámica de fluidos computacionales en un perfil 2D, de esta forma, la obtención de los resultados numéricos se realiza de una manera eficiente y rápida, además de que el software a utilizar, Ansys® Fluent®, permite obtener datos ya etiquetados (con el valor objetivo en cada punto) sin costo alguno utilizando la licencia estudiantil. Igualmente, utilizar un solo perfil permite una gama más amplia de velocidades sin apenas aumentar el tiempo computacional requerido.
El perfil aerodinámico utilizado es el NACA 4412 , justificando su uso al contar con resultados experimentales del mismo para validar la simulación con CFD, además de que, si bien, no es un perfil geométricamente muy distinto a la media, el no ser un perfil simétrico permite a la red aprender mejor los distintos patrones geométricos que se puede encontrar en esta gama de perfiles. Así, siguiendo la misma metodología para la obtención de la Figura 2.6, se obtienen las coordenadas para graficar el perfil NACA 4412, mostrado en la Figura 3.1.

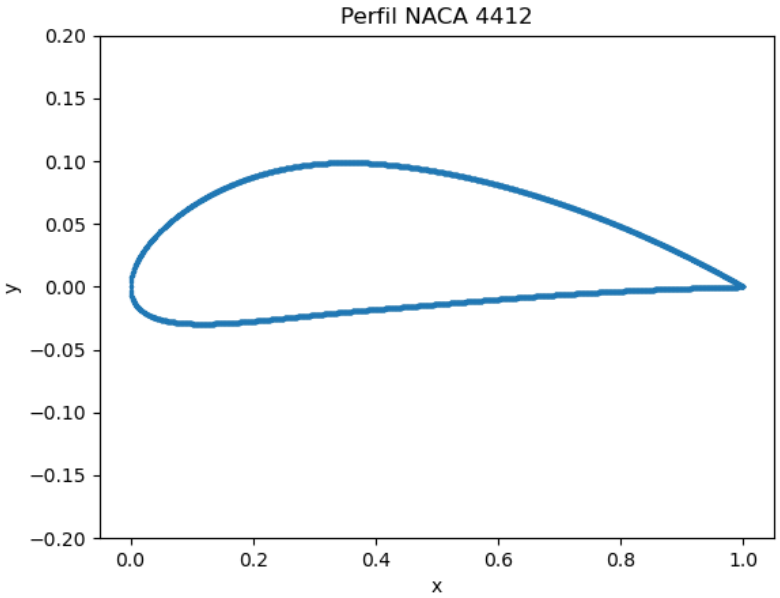


Figura 3.1: Perfil aerodinámico NACA 4412.

Con este mismo perfil, se rota su ángulo de ataque en un rango de entre -15 a 15

grados para cubrir todos los ángulos de operación posibles en su caracterización. Este procedimiento se realiza directamente en la sección *Geometry* de *Ansys Fluent* (Figura 3.2), teniendo que remallar en cada caso, sin embargo, igualmente se cuenta con una función para rotar las coordenadas del perfil en Python que se requirió más adelante, pues al preprocesar los datos igualmente se requiere remallar en cada caso.

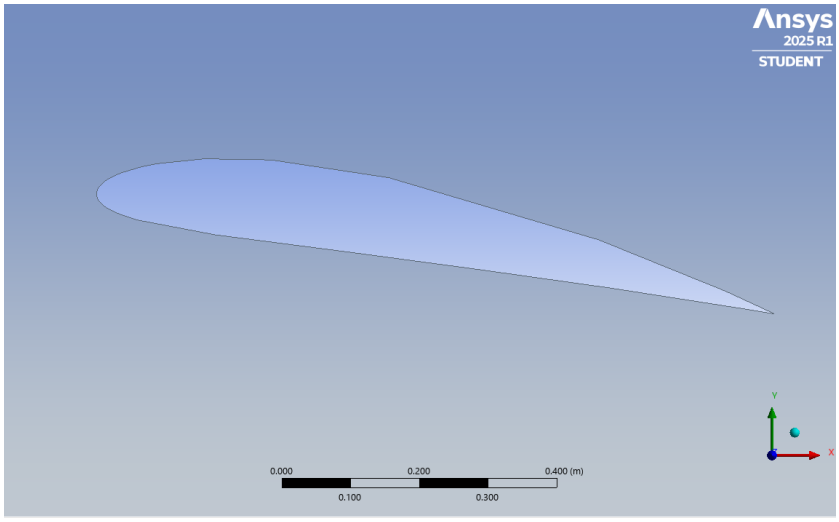


Figura 3.2: NACA 4412 rotado 10 grados en la pestaña de *Geometry*.

## 3.2. Simulación numérica con CFD de el perfil de referencia

Antes de proceder con a la obtención de datos y, por ende, con la simulación propiamente dicha, una última delimitación del problema: El régimen de Reynolds en el que se trabajará; Considerando que un régimen de flujo turbulento externo comienza con un $Re = 500000$ [4], se ha optado por un número de Reynolds menor a este con el fin de evitar valores atípicos que puedan comprometer la precisión del modelo. De esta forma, utilizando la ecuación 2.12 para cuerda de $1[m]$, $\rho = 1.225[\frac{kg}{m^3}]$ y $\upsilon = 1.79 \times 10^{-05}[\frac{m^2}{2}]$, el valor de las velocidades utilizadas se reflejan en la Tabla 3.1.

Ahora bien, la geometría y el dominio sobre el que trabajará la simulación fue definido de tal forma que se tenga una malla estructurada lejos del perfil, una no estructurada

Tabla 3.1: Tabla de Reynolds con sus velocidades para cuerda de 1 [m]

| **Reynolds** | **Velocidad [m/s]** |
|---|---|
| 10000 | 0.15 |
| 25000 | 0.37 |
| 50000 | 0.73 |
| 75000 | 1.10 |
| 100000 | 1.46 |
| 150000 | 2.19 |
| 200000 | 2.92 |
| 250000 | 3.65 |

cerca del mismo y, una capa tipo *inflation* en la pared del perfil, de esta forma, la malla externa no se modifica, la interna se adapta al perfil rotado en cualquier ángulo y, es posible captar la capa límite en caso de ser requerido gracias a la capa *inflation*. La geometría y el mallado mencionado se pueden observar en las las Figuras 3.3, 3.4 y 3.5.

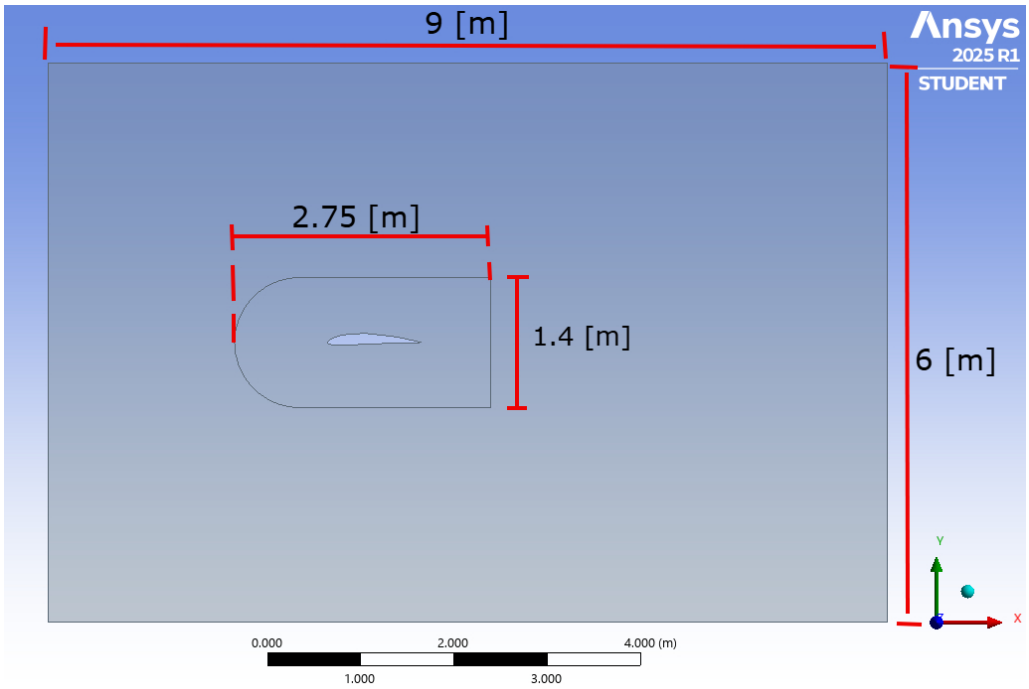


Figura 3.3: Geometría en Ansys Fluent.

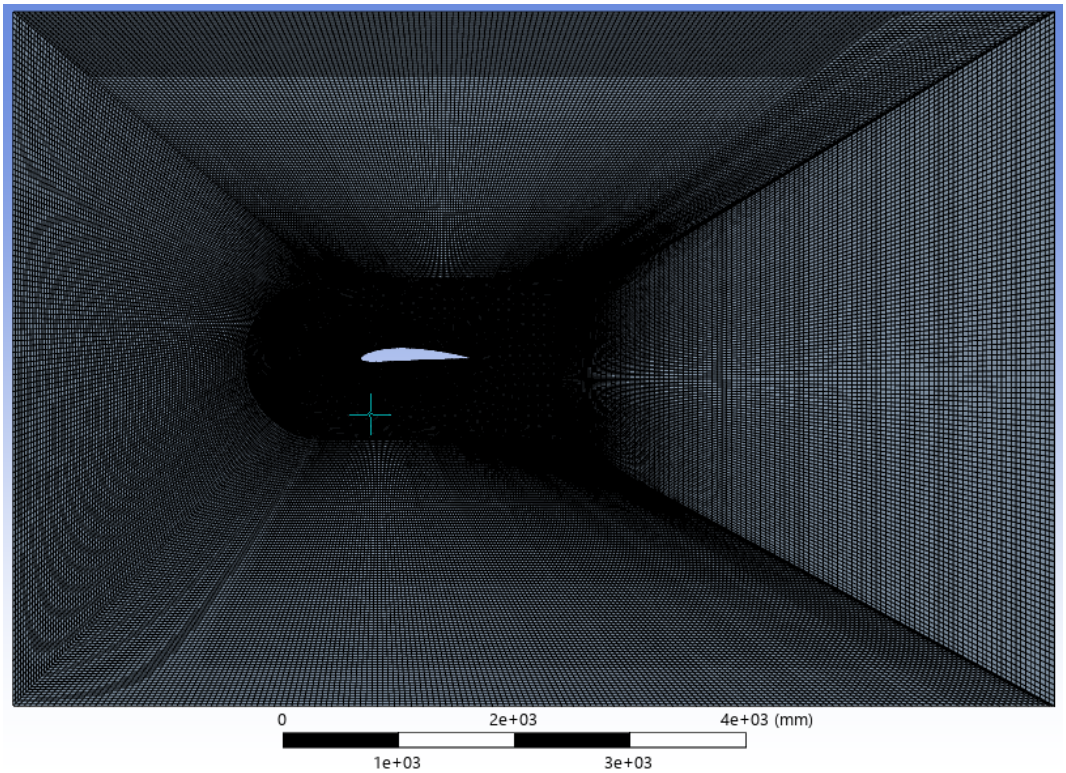


Figura 3.4: Mallado completo de la geometría.

Una vez definido lo anterior, se configura el solucionador del Fluent. Se configura la solución del tipo *steady*, el modelo de la turbulencia es el ya mencionado Spalart-Allmaras, el fluido es aire con los parámetros ya definidos para el Reynolds de la Tabla 3.1 y la velocidad esta dada por la misma tabla.
Al realizar la simulación con estas condiciones, se obtuvieron los residuales de la Figura 3.6 para 500 iteraciones, en este caso para una velocidad de $3.65[m/s]$ y ángulo de ataque de cero grados.
Tras completar la simulación, se determinan los campos de presión y velocidad con los cuales se puede observar de manera visual la distribución de estas variables sobre el perfil, sin embargo, para la obtención de datos, se exporta directamente los valores numéricos de las variables para cada par de coordenadas $x$ y $y$ de los nodos en la malla.
El procedimiento anterior se realizó para cada uno de los ángulos de ataque mencionados, para cada una de las velocidades de la Tabla 3.1. Con ello, se efectuaron un total de 248 simulaciones de las cuales se obtienen los datos de presión y velocidad.

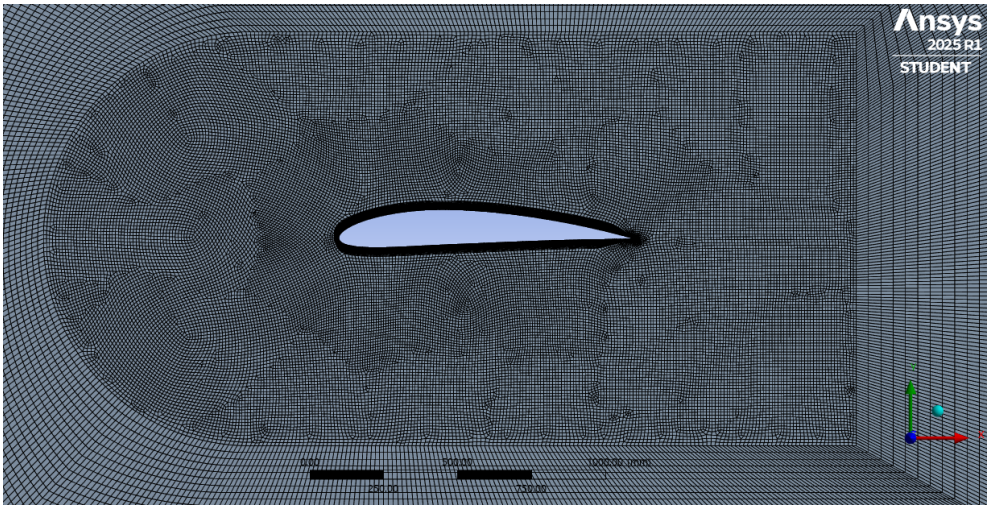


Figura 3.5: Mallado interno de la geometría.

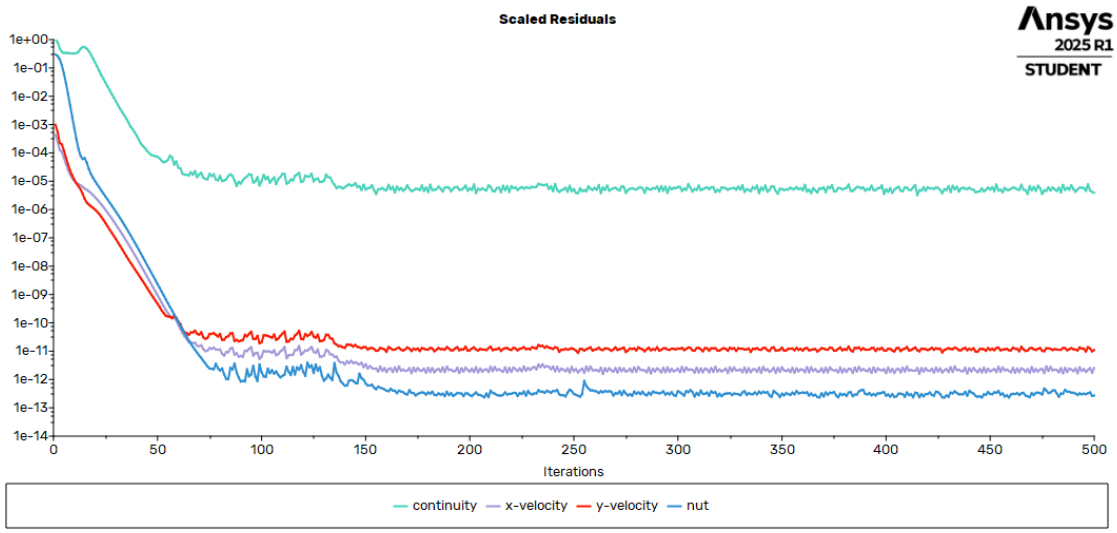


Figura 3.6: Residuales para simulación con velocidad de 3.65$[m/s]$ y ángulo de ataque de cero grados.

## 3.3. Preparación de los datos

Al seguir con el tercer paso, la exploración de datos, se observó que en estos esta presente la coordenada $z$, esto debido a que el método usado por el software Ansys es

el de volumen finito, por lo que siempre estará esta coordenada en los datos aún si la simulación es de tipo bidimensional, sin embargo, esta es despreciable y en todos los casos su valor es prácticamente cero. Además, se encontró que cada simulación cuenta con alrededor de 218000 puntos, aunque, esto se debe principalmente al tamaño del dominio que se requiere para realizar la simulación y, la mayoría de estos no son necesarios al no revelar información importante para el entrenamiento, pues su distancia con el perfil puede provocar un sesgo al entrenar la red neuronal si se cuenta con un mismo valor de velocidad en casi todos los casos. Igualmente, se observó que el número de puntos y su posición varía según el ángulo de ataque de el perfil a causa a la región no estructurada de la malla, esto podría suponer un problema considerable para la red al no poder detectar los patrones de posición adecuadamente.
Con base en lo anterior, se procedió a preparar los datos tras identificar los problemas que el formato original podría representar para la red neuronal. Para ello, se desarrolló una función en Python encargada de preprocesar y cargar todos los archivos, la cual realizaba las siguientes tareas:

1. Se eliminó la columna de la coordenada $z$.

2. Se delimitó la región de datos a solo un rectángulo cercano alrededor del perfil, con medidas de 1.75[m] de ancho y 0.8[m] de alto, descartando el resto de datos.

3. Una vez delimitado el dominio, se realizó una nueva malla directamente en el código, esto con el fin de hacerla estructurada y eliminar la diferencia encontrada en el número y la posición de los puntos. La nueva malla se muestra en la Figura 3.7, igualmente se puede observar el tamaño ya delimitado de la misma.

4. Se uso la función *griddata* de python para interpolar los valores de presión y velocidad obtenidos de las simulaciones de CFD a los puntos de la nueva malla.

5. Finalmente, además de los datos de coordenadas, se añadieron como parámetros el ángulo de ataque correspondiente, el número de Reynolds como representación de la velocidad y longitud, la distancia de cada punto hasta el borde más cercano del perfil y, un parámetro extra para dividir entre lo que es *wall* (el borde del perfil) y lo que es fluido, pues, como se explicó en el marco teórico, la cercanía con la geometría es un punto importante que considerar en la dinámica de fluidos.

6. Como último punto, los datos se añadieron todos juntos en un nuevo archivo de tipo *dataframe* de la librería pandas de python, pero ordenándolos aleatoriamente, esto para generar variedad y evitar un sesgo en el entrenamiento.

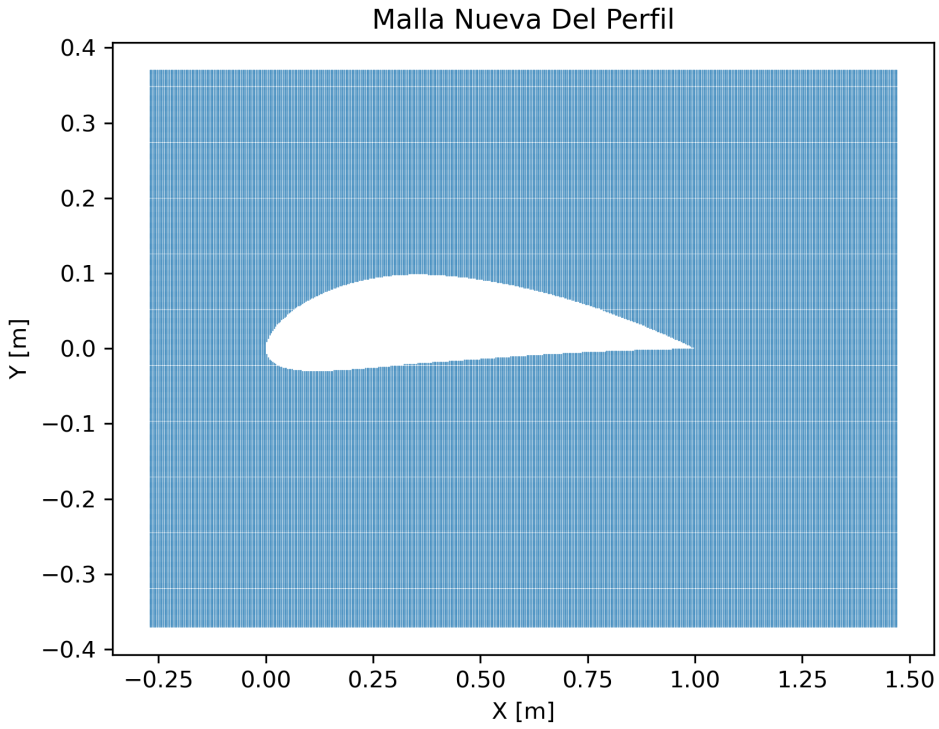


Figura 3.7: Malla reestructurada del perfil aerodinámico.

Tras aplicar esta función a todos los archivos, se obtiene el formato datos que se puede observar en la Figura 3.8, donde, las características de salida unicamente corresponden a la columna de *Pressure [ Pa ]* y *Velocity [ m s∧-1]*, mientras el resto se usan como entradas. De esta manera, cada simulación terminó con 234194 instancias, para un set completo de 58080112 de instancias totales.
Igualmente, como es habitual en el aprendizaje automático, el set completo de datos se dividió en set de entrenamiento y set de prueba usando la función *train_test_split* de la librería *sklearn* con un tamaño para el set de pruebas del 20 %, donde, además de dividir los datos de esta forma, nuevamente reordena aleatoriamente los mismos siguiendo la lógica ya mencionada previamente. De la misma forma, usando esta misma función, se dividió con el mismo porcentaje el set de entrenamiento para obtener un set de validación, ambos con datos de entradas y salidas.
Una última preparación de datos, es la estandarización de las características usando la función *StandardScaler()*, la cual, a partir de la media y la desviación estándar

| | X [ m ] | Y [ m ] | Pressure [ Pa ] | Velocity [ m s^-1 ] | Distancia_min | AOA | Reynolds | Wall |
|---|---|---|---|---|---|---|---|---|
| 0 | -0.270000 | -0.37 | -0.009943 | 1.128912 | 0.454852 | -15 | 75000 | 0 |
| 1 | -0.266513 | -0.37 | -0.010841 | 1.129566 | 0.452747 | -15 | 75000 | 0 |
| 2 | -0.263026 | -0.37 | -0.011755 | 1.130231 | 0.450659 | -15 | 75000 | 0 |
| 3 | -0.259539 | -0.37 | -0.012685 | 1.130905 | 0.448589 | -15 | 75000 | 0 |
| 4 | -0.256052 | -0.37 | -0.013630 | 1.131589 | 0.446537 | -15 | 75000 | 0 |
| 5 | -0.252565 | -0.37 | -0.014588 | 1.132282 | 0.444499 | -15 | 75000 | 0 |
| 6 | -0.249078 | -0.37 | -0.015559 | 1.132985 | 0.442479 | -15 | 75000 | 0 |
| 7 | -0.245591 | -0.37 | -0.016545 | 1.133698 | 0.440478 | -15 | 75000 | 0 |
| 8 | -0.242104 | -0.37 | -0.017544 | 1.134422 | 0.438495 | -15 | 75000 | 0 |
| 9 | -0.238617 | -0.37 | -0.018557 | 1.135157 | 0.436529 | -15 | 75000 | 0 |

Figura 3.8: *Dataframe* de las primeras 10 instancias.

del set de entrenamiento, escala los datos numéricos con el fin de que características como el número de Reynolds no destaquen sobre otras al momento de entrenar la red debido a su magnitud, mientras que los datos categóricos como el ángulo de ataque (AOA) y el *wall* se dejan sin transformación.

## 3.4. Descripción y configuración de la red neuronal

El siguiente paso para el proyecto es la exploración de modelos, esto se refiere sobre todo a los diferentes modelos de aprendizaje automático, como *decision trees* o *random forest*. En este caso la opción definida desde el principio es la de una red neuronal con una arquitectura de MLPs de regresión con capas densas debido a su gran potencial y competencia ya demostrada en una gran variedad de tareas (como se pudo ver en [15], [25], [26], [12], [13] y [27]), es por esta razón que se continuó directamente con el siguiente paso: el ajuste del modelo para encontrar la mejor configuración de hiperparámetros en el mismo.
Para el ajuste del modelo, se inició con una configuración estándar recomendada para la mayoría de modelos de acuerdo con [11], a partir de esta primera iteración se exploraron diferentes hiperparámetros extras que podrían mejorar el rendimiento de la red, acelerar la velocidad de entrenamiento de la misma o mejorar su generalización. Para ello, se utilizó el llamado *hyperband tuner* de la librería *Tensor Flow*,

la cual pseudoaleatoriamente explora diferentes configuraciones del modelo en unas pocas épocas cada vez (entre 2 y 5), seleccionando los que arrojaban un menor error de validación y entrenando nuevamente estos mismos unas pocas épocas más, para al final seleccionar el modelo que arrojara el mejor resultado. La función que construía el modelo completo se puede encontrar en el Apéndice A, sin embargo, para una lista rápida de los hiperparámetros probados y finales se puede observar la Tabla 3.2. Nuevamente, para una mejor descripción de cada uno se puede consultar [11].

Tabla 3.2: Valores probados y finales en el ajuste del modelo.

| **Hiperparámetro** | **Valores probados** | **Valor final** |
|---|---|---|
| Capas ocultas | De 5 a 24 | 9 |
| Neuronas por capa | De 50 a 252 | 147 |
| Taza de aprendizaje | De 1e-5 a 1e-3 | 1.432e-4 |
| Optimizador | Nesterov o Adam | Adam |
| Función de pérdida | Huber o MSE | MSE |
| Ratio de dropout | De 0.1 a 0.3 | 0.15 |
| Valor de Beta para Swish | De 0.5 a 2 | 1.9 |
| Uso de capa batchorm | True o False | True |
| Tamaño del lote | 32, 64, 128, 256 o 512 | 128 |
| Función de activación | Swish | Swish |
| Métrica | MAE | MAE |
| Callbacks | Early Stopping y ReduceLROnPlateu | Early Stopping y ReduceLROnPlateu |

Como se observa, la mejor configuración del modelo cuenta con nueve capas, de 147 neuronas en cada una, donde la función de activación es *Swish*. Entre cada capa hay una capa de *batchnorm* que normaliza las salidas de las capas directamente durante el entrenamiento y actúa como regularización junto con la capa de Dropout, la cual, en términos simples, apaga algunas neuronas aleatoriamente durante cada lote de datos. La representación de este modelo se encuentra en la Figura 3.9. Con ello, ahora es posible entrenar la red neuronal en todo el conjunto de datos.

## 3.5. Entrenamiento de la red neuronal

El entrenamiento se planteó para ser realizado por cien épocas, es decir, cien iteraciones con todos los datos, sin embargo, con el fin de aplicar regularización se usaron los *callbacks* nombrados en la Tabla 3.2, donde el *Early Stopping* detenía el

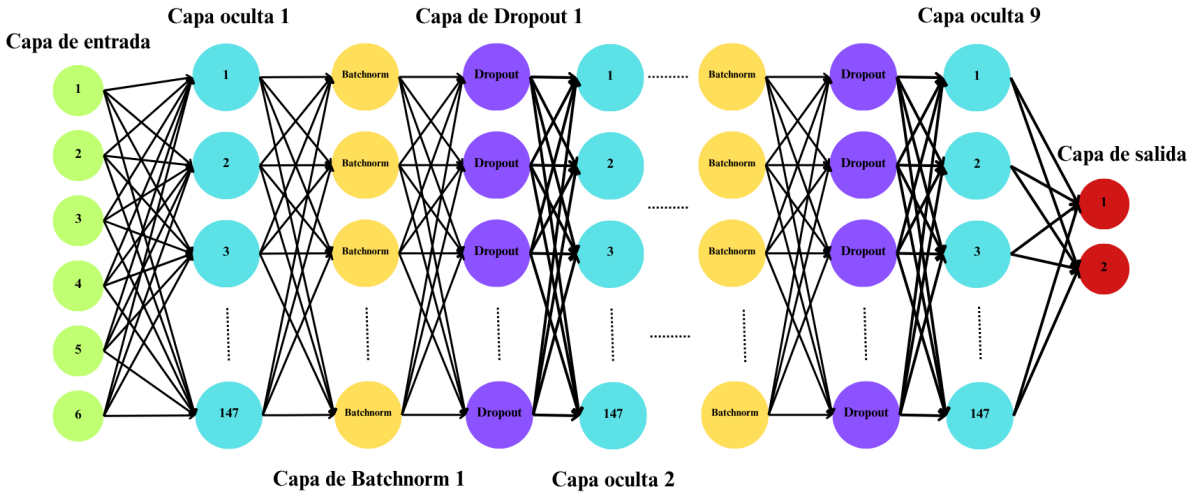


Figura 3.9: Diagrama de la red neuronal utilizada.

entrenamiento en caso de que no hubiera mejoría en la métrica del *validation loss* tras un número definido de épocas. De esta manera, bastaron únicamente 29 iteraciones y un total de 925 minutos para alcanzar el resultado final
El código para el entrenamiento, así como el historial de entrenamiento se puede observar en el Apéndice A. La gráfica de la métrica (MAE) y la de la función de pérdida (MSE), tanto del entrenamiento como de la validación, se encuentran en las Figuras 3.10 y 3.11.
En la Figura 3.10 es posible observar que la pérdida se detuvo en la iteración número 29 debido a que en la número 14 alcanzó el mínimo global, la razón para esto ya fue mencionada y se puede observar directamente en la gráfica, pues generalmente una pérdida de validación mayor a la pérdida de entrenamiento indica que el modelo está sobre ajustado a los datos de entrenamiento y, por lo tanto, no generalizará bien, por esta razón se utilizó el *Early Stopping*, ya que al detenerse, este vuelve a los valores de la red con mejor rendimiento.
En la Figura 3.11 se muestra un comportamiento típico, pues, como se observa, el valor del error es considerablemente mayor en el set de entrenamiento que en el de validación, esto es resultado de utilizar la técnica de *Dropout*, pues, como ya se mencionó, este apaga aleatoriamente algunas neuronas, haciendo que la red al momento de evaluar la métrica de error en el entrenamiento lo haga con un modelo que no está al cien por ciento de su capacidad, sin embargo, esta técnica solo se activa durante el entrenamiento y tanto en la validación como en cualquier otro momento, como en la predicción, este trabaja sin esas neuronas de menos.

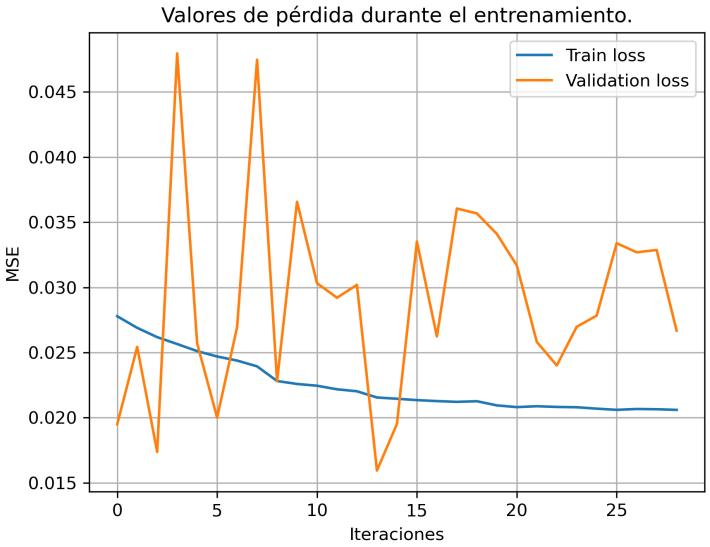


Figura 3.10: Valores de pérdida de el set de entrenamiento y el set de validación.

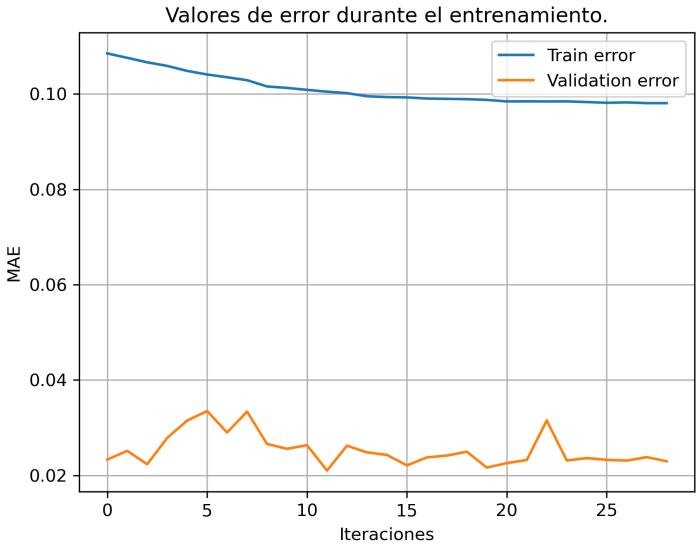


Figura 3.11: Valores de error de el set de entrenamiento y el set de validación.

# 4 Resultados

Tras el capítulo anterior, la red neuronal ha sido completamente entrenada y ajustada, por lo que el siguiente paso consiste en presentar los resultados obtenidos. En esta sección se muestran los valores de error, las pérdidas durante el entrenamiento, así como los campos de presión y velocidad generados por la red para una velocidad dada. Estos resultados se analizarán tanto para datos entrenados como no entrenados y se comparan con los obtenidos mediante una simulación numérica convencional.

## 4.1. Valores de pérdida y error

Los valores correspondientes a la pérdida y al error ya han sido presentados previamente en las Figuras 3.10 y 3.11, sin embargo, como se mencionó, los únicos valores importantes para este caso es el de la iteración número 14, pues es en este punto donde se obtuvo el mejor resultado y, por lo tanto, es el punto al que los valores de la red regresaron tras el entrenamiento.
Con base en lo anterior, el valor final para evaluar el desempeño de la red —tanto para el set de entrenamiento, el set de validación y añadiendo la evaluación en el set de prueba— se presenta en la Tabla 4.1.

Tabla 4.1: Valores de error y pérdida en set de entrenamiento, validación y prueba.

| | **Entrenamiento** | **Validación** | **Prueba** |
|---|---|---|---|
| **Error** | 0.0995 | 0.0248 | 0.0253 |
| **Pérdida** | 0.0215 | 0.0159 | 0.0164 |

Algo a tomar en cuenta es que los valores mostrados en la Tabla 4.1 es el resultado del conjunto completo de datos, con valores de distintas simulaciones a la vez, por lo que, se realizaron igualmente algunas otras pruebas de desempeño y precisión para un solo

tipo de simulación, tanto para un conjunto de datos con velocidad ya entrenada, como uno con velocidad no entrenada y, uno con un perfil NACA distinto. El resultado de dichas pruebas se muestra en la Tabla 4.2. Recordando que para obtener estos valores igualmente se realizaron simulaciones con el software Ansys Fluent, pues la evaluación se realiza comparando el valor real con el precedido por el modelo.

Tabla 4.2: Valores de error y pérdida para distintas pruebas.

| | **Datos con Re = 100,000 AOA = 0 (Entrenados)** | **Datos con Re = 100,000 AOA = 10 (Entrenados)** | **Datos con Re = 68,500 AOA = 0 (No entrenados)** | **Datos con Re = 68,500 AOA = 10 (No entrenados)** | **Datos con Re = 400,000 AOA = 0 (No entrenados)** | **Datos con NACA 2412 AOA = 0 Re = 100,000 (No entrenados)** |
|---|---|---|---|---|---|---|
| **Error** | 0.0100 | 0.0177 | 0.0468 | 0.0528 | 1.6217 | 0.0363 |
| **Pérdida** | 0.0006 | 0.0012 | 0.0034 | 0.0040 | 4.0319 | 0.0030 |

Una última consideración es que los valores de error vistos en ambas tablas es el resultado tanto de la presión como de la velocidad combinados, sin embargo, puede ser igualmente útil ver ambos valores por separado con el fin de observar si existe una mayor o menor precisión en alguno de ellos. Es por lo anterior que se obtuvo el valor de error individual de las salidas en algunas de las pruebas ya realizadas, en la Tabla 4.3 se muestran estos valores.

Tabla 4.3: Valores de error para presión y velocidad por separado.

| | **Datos con Re = 100,000 AOA = 0 (Entrenados)** | **Datos con Re = 68,500 AOA = 0 (No entrenados)** | **Datos con Re = 400,000 AOA = 0 (No entrenados)** | **Datos con NACA 2412 AOA = 0 Re = 100,000 (No entrenados)** |
|---|---|---|---|---|
| **Error Presión** | 0.01 | 0.0144 | 1.2795 | 0.0472 |
| **Error Velocidad** | 0.0101 | 0.0793 | 1.9638 | 0.0254 |

## 4.2. Valores de presión y velocidad precedidos por el modelo

Una vez obtenidas las métricas derivadas del entrenamiento de la red para distintos casos, es importante señalar que, si bien estas reflejan el posible margen de error, no permiten dimensionar adecuadamente su magnitud sin considerar el orden y los valores específicos de los parámetros involucrados.
Por ello, para lograr una visualización completa, los resultados se presentan gráficamente en forma de contornos de presión y velocidad, similares a los generados por Ansys Fluent. Asimismo, se incluyen tablas con valores numéricos representativos en las zonas más relevantes, con el fin de facilitar la interpretación cuantitativa de los resultados.
En primer lugar, en las Figuras 4.1, 4.2, 4.3, 4.4, 4.5 y 4.6 se pueden observar los contornos de presión y velocidad para el que podemos considerar como el caso objetivo (mismo perfil y distinta velocidad), con un Reynolds de 68,500, a distintos ángulos de ataque y con el fin de observar los distintos patrones de los parámetros aún en datos no entrenados Se observa que, de acuerdo con la teoría ya citada, la distribución de la presión y la velocidad coinciden con el de cualquier otro perfil aerodinámico convencional.

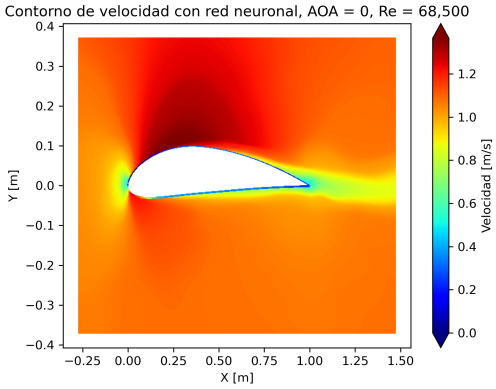


Figura 4.1: Contorno de velocidad de la RNA con AOA = 0 y Re = 68,500

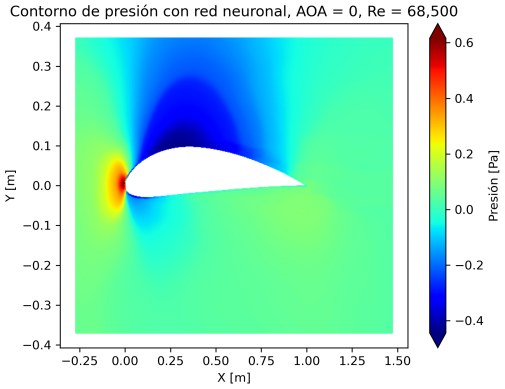


Figura 4.2: Contorno de presión de la RNA con AOA = 0 y Re = 68,500

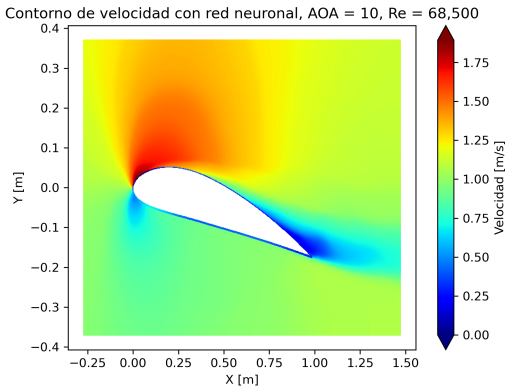


Figura 4.3: Contorno de velocidad de la RNA con AOA = 10 y Re = 68,500

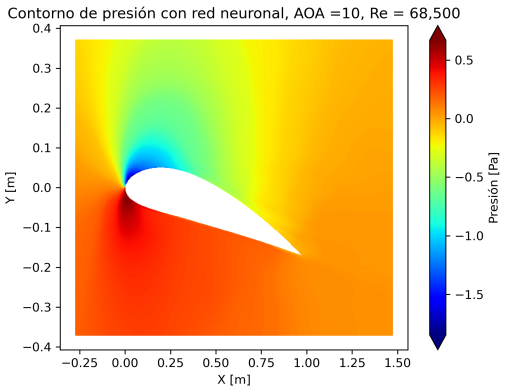


Figura 4.4: Contorno de presión de la RNA con AOA = 10 y Re = 68,500

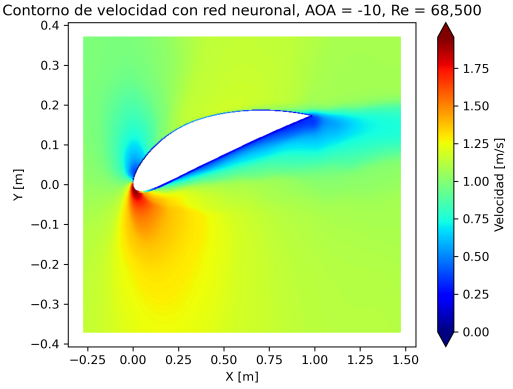


Figura 4.5: Contorno de velocidad de la RNA con AOA = -10 y Re = 68,500

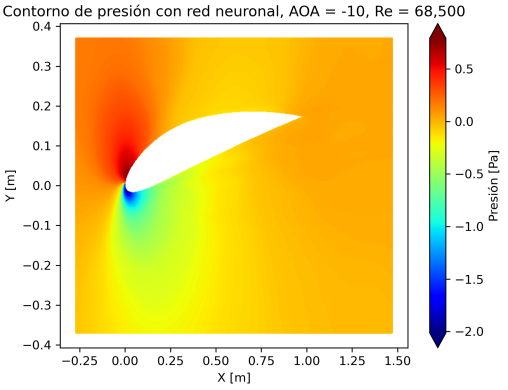


Figura 4.6: Contorno de presión de la RNA con AOA = -10 y Re = 68,500

En segundo lugar, aunque en los contornos es posible observar la escala de los datos y los valores aproximados de estos en cada punto del nuevo mallado, resulta igualmente útil contar con los valores numéricos en algunas zonas de interés, es por ello que en la Tabla 4.4 se observan los primeros 15 puntos de datos filtrados de presión, primero en donde se encuentra el *Wall*, el contorno del perfil, y segundo en los puntos marcados a una distancia de 0.1 [m] o menor de este contorno, con el fin de captar solo las zonas mas afectadas por el flujo.
Los últimos dos conjuntos de datos, que son interesantes para su análisis, son los contornos arrojados en las pruebas donde se utilizaron datos no entrenados, es decir,

Tabla 4.4: Tabla de valores numéricos de presión y velocidad en zonas de interés.

| **Punto** | **Presión [Pa] en el contorno del perfil** | **Presión [Pa] a una distancia <= 0.1 [m]** | **Velocidad [m/s] a una distancia <= 0.1[m]** |
|---|---|---|---|
| **0** | -0.2432 | -0.0179 | 0.8482 |
| **1** | -0.2394 | -0.0218 | 0.8215 |
| **2** | -0.2345 | -0.0253 | 0.7935 |
| **3** | -0.2293 | -0.0285 | 0.7635 |
| **4** | -0.2231 | -0.0313 | 0.7347 |
| **5** | -0.2153 | -0.0338 | 0.7091 |
| **6** | -0.2068 | -0.0360 | 0.6863 |
| **7** | -0.1992 | -0.0379 | 0.6652 |
| **8** | -0.2890 | 0.0395 | 1.0992 |
| **9** | -0.2853 | -0.0408 | 1.0763 |
| **10** | -0.2813 | -0.0419 | 1.0532 |
| **11** | -0.2772 | -0.0427 | 1.0302 |
| **12** | -0.2730 | -0.0434 | 1.0062 |
| **13** | -0.2688 | -0.0438 | 0.9801 |
| **14** | -0.2643 | -0.0441 | 0.9519 |

los precedidos para un Re = 400,000 y los precedidos para un perfil NACA 2412. Así, los contornos de presión de ambos se muestran en las Figuras 4.7 y 4.8

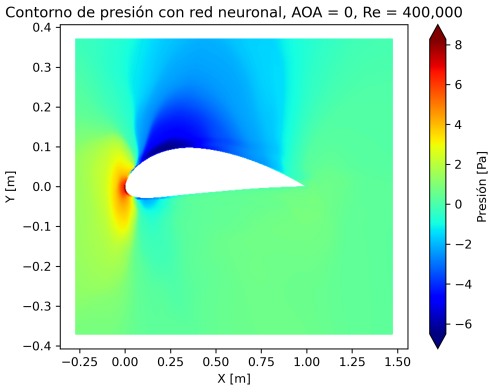


Figura 4.7: Contorno de presión de la RNA con AOA = 0 y Re = 400,000

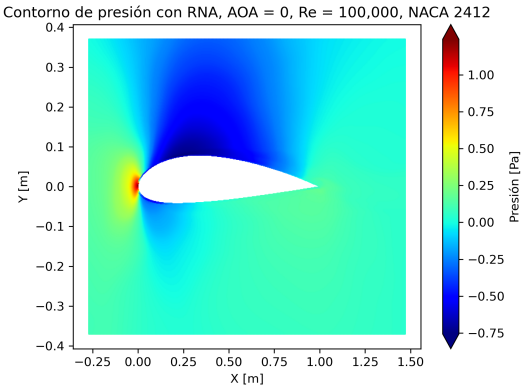


Figura 4.8: Contorno de presión de la RNA con AOA = 0, Re = 100,000 y NACA 2412

## 4.3. Comparación de resultados con CFD

Tras obtener los valores de error y las magnitudes de los parámetros precedidos es posible compararlos con los datos originales de la simulación CFD, con este fin y, utilizando los valores de error (MAE) de la Tabla 4.3, es posible obtener el porcentaje de error para las distintas pruebas, esto utilizando como valor teórico el promedio de la velocidad y presión en cada una de las mismas, obteniendo así los datos mostrados en la Tabla 4.5.

Tabla 4.5: Porcentajes de error para distintas pruebas de la RNA contra datos obtenidos de simulaciones CFD.

| | **Datos con Re = 100,000 AOA = 0 (Entrenados)** | **Datos con Re = 68,500 AOA = 0 (No entrenados)** | **Datos con Re = 400,000 AOA = 0 (No entrenados)** | **Datos con NACA 2412 AOA = 0 Re = 100,000 (No entrenados)** |
|---|---|---|---|---|
| **% Error de velocidad** | 12.9994 | 17.4260 | 55.7833 | 34.1194 |
| **% Error de presión** | 8.3567 | 10.2937 | 288.2933 | 40.4548 |

De forma muy similar, es posible obtener el error individualmente para cada uno de los puntos. Así, con los valores de la Tabla 4.4 se obtiene la Tabla 4.6 con el porcentaje de error individual para los primeros 15 puntos de la zonas ya mencionadas.
Ahora bien, aunque ya se presentó que la forma más común para medir la precisión de una red neuronal en una tarea de regresión es mediante el MAE o el MSE, estas sólo corresponden a la diferencia entre el valor absoluto punto a punto en el mallado, sin embargo, debido a la naturaleza de los datos de entrada y los resultados esperados, esta es solo una parte de lo que se debe considerar para el rendimiento del modelo, pues si se comparan cualitativamente los contornos de la simulación de CFD y los de la red neuronal, estos presentan claramente una gran similitud, para ello, en las Figuras 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15 y 4.16 se pueden observar los contornos originales de la simulaciones numéricas de CFD para los casos ya presentados en la sección 4.2.
Pese a que se a simple vista se observan claras similitudes, es preferible usar una medida clara para reflejar esta relación. Para ello, se calculó un parámetro útil denominado como el *Structural similarity index measure* (SSIM), el cual es usada principalmente en tratamiento de imágenes, donde, midiendo características como la luminiscencia, contraste y estructura, arroja un valor entre -1 y 1 de la similaridad entre dos imágenes, en el cual -1 corresponde a una perfecta anti correlación, 0 indica no similaridad en absoluto y 1 una perfecta similaridad.

Tabla 4.6: Porcentaje de error en los primeros quince puntos en las zonas de interés.

| **Punto** | **% Error de presión en el contorno del perfil** | **% Error de presión a una distancia <= 0.1 [m]** | **% Error de velocidad a una distancia <= 0.1 [m]** |
|---|---|---|---|
| **0** | 51.74 | 15.31 | 9.39 |
| **1** | 53.07 | 5.94 | 9.46 |
| **2** | 53.48 | 0.95 | 9.51 |
| **3** | 53.09 | 5.98 | 9.55 |
| **4** | 52.57 | 9.61 | 9.58 |
| **5** | 50.92 | 12.17 | 9.60 |
| **6** | 48.17 | 13.88 | 9.61 |
| **7** | 46.43 | 14.92 | 9.62 |
| **8** | 46.67 | 15.43 | 9.61 |
| **9** | 46.84 | 15.51 | 9.60 |
| **10** | 45.82 | 15.22 | 9.58 |
| **11** | 46.30 | 14.65 | 9.56 |
| **12** | 47.44 | 13.88 | 9.54 |
| **13** | 46.30 | 12.95 | 9.51 |
| **14** | 47.27 | 11.92 | 9.48 |

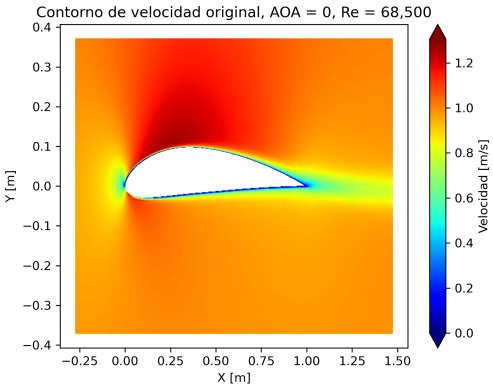


Figura 4.9: Contorno de velocidad original con AOA = 0 y Re = 68,500

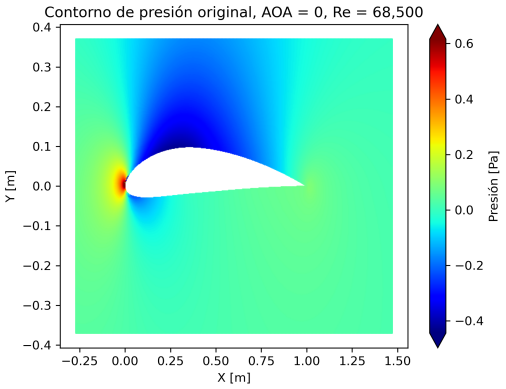


Figura 4.10: Contorno de presión original con AOA = 0 y Re = 68,500

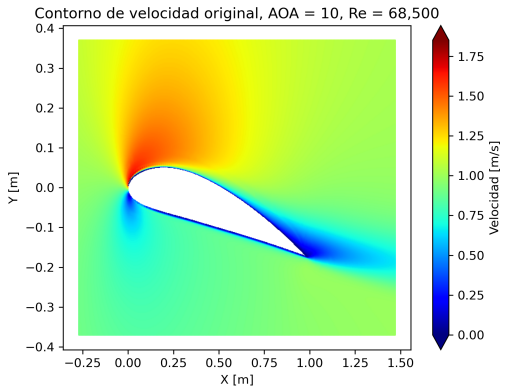


Figura 4.11: Contorno de velocidad original con AOA = 10 y Re = 68,500

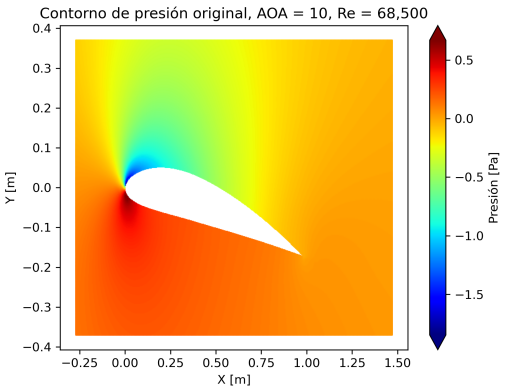


Figura 4.12: Contorno de presión original con AOA = 10 y Re = 68,500

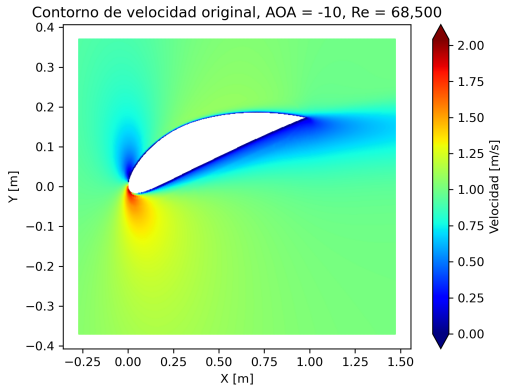


Figura 4.13: Contorno de velocidad original con AOA = -10 y Re = 68,500

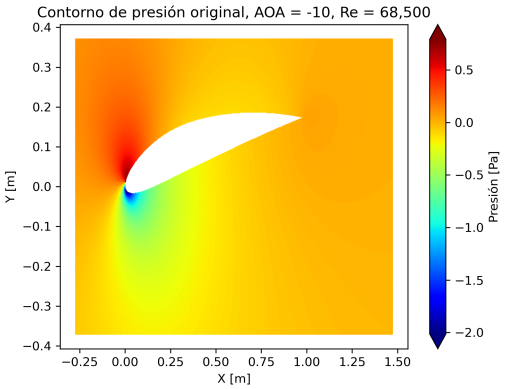


Figura 4.14: Contorno de presión original con AOA = -10 y Re = 68,500

Aunque, como ya se mencionó, esta fue pensada para el tratamiento de imágenes y fotografías, es posible utilizarlo individualmente para los datos de presión y velocidad, pues internamente cada parámetro medido corresponde a: Un nivel medio local de cada punto (si la predicción tiene en promedio las mismas presiones locales que la simulación), la escala de variación (si la predicción reproduce la variación local de presión en cada zona) y la forma del patrón local (si los cambios de presión ocurren en la misma dirección y patrón espacial). Con lo anterior, en la Tabla 4.7 se muestran cada uno de las medidas SSIM de los contornos presentados en las secciones 4.2 y 4.3.

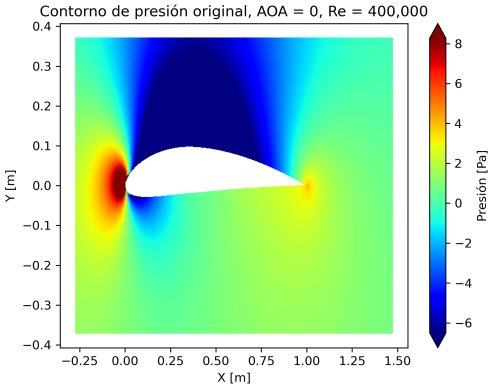


Figura 4.15: Contorno de velocidad original con AOA = 0 y Re = 400,000

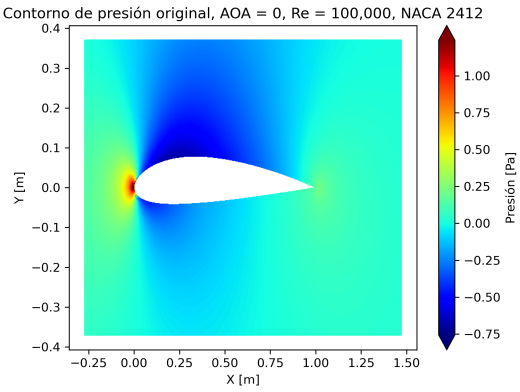


Figura 4.16: Contorno de presión original con AOA = 0, Re = 100,000 y NACA 2412

Tabla 4.7: SSIM de los contornos de la RNA y los de CFD.

| | **Datos con Re = 68,500 AOA = 0** | **Datos con Re = 68,500 AOA = 10** | **Datos con Re = 68,500 AOA = -10** | **Datos con Re = 400,000 AOA = 0** | **Datos con Re = 100, 000 AOA = 0 NACA 2412** |
|---|---|---|---|---|---|
| **SSIM de velocidad** | 0.9781 | 0.9798 | 0.9732 | 0.6279 | 0.8566 |
| **SSIM de presión** | 0.9678 | 0.9944 | 0.9765 | 0.8842 | 0.6982 |

## 4.4. Análisis de los datos

Finalmente, un último tipo de resultado a considerar es el análisis de los tiempos de computo y *hardware* que hizo posible la creación del modelo, pues la reproducción del mismo, así como muchas de las mejoras que pueden llevarse a cabo dependen de esto último.
En primer lugar, la Tabla 4.8 contiene la configuración del hardware de computo utilizada para el ajuste, entrenamiento y predicción del modelo, el cual fue hecho completamente con el procesador, sin una tarjeta de video dedicada para su aceleramiento.

Tabla 4.8: Configuración de la PC utilizada para el modelo.

| | **Procesador** | **Memoria RAM** | **Sistema operativo** | **Tarjeta gráfica dedicada** |
|---|---|---|---|---|
| **Componente** | Ryzen 5 5600X | 16GB x 2 DDR4 3200Mhz | Windows 10 | NA |

Tomando en cuenta lo anterior, en la Tabla 4.9 se presenta el tiempo de cómputo de las distintas etapas del modelo, así como el tiempo requerido para realizar solo el cálculo de la simulación de CFD (sin considerar el tiempo requerido en hacer le malla y configuración de parámetros del solucionador) esto último para compararlo posteriormente con el tiempo de predicción.

Tabla 4.9: Tiempos de cómputo.

| **Ajuste [Hrs]** | **Entrenamiento [Hrs]** | **Predicción [s]** | **Simulación CFD [s] (200 iteraciones)** |
|---|---|---|---|
| 68 | 15.4 | 9 | 35 |

# 5 Discusión de resultados

Con los resultados ya conocidos, es posible hacer una interpretación más allá del valor numérico y estimar de mejor forma el desempeño del modelo de la red neuronal. Para ello, se analizará cada una de las tablas presentadas en la sección de resultados y así concluir si los objetivos planteados desde el inicio fueron cumplidos.

## 5.1. Significado de los valores de error.

En primer lugar, la Tabla 4.1 presenta los valores de error y pérdida al final del entrenamiento, recordando que ambos valores se basan en el *Mean Absolute Error* y el *Mean Squared Error* respectivamente, esta primer distinción puede explicar la diferencia en ambos valores, pues el MSE suele ajustar mayormente los pesos en caso de encontrar errores grandes, sin embargo, dado que al analizar los datos desde el principio no se encontraron valores atípicos ni picos muy altos, es poco probable que existan esta magnitud de errores, dando como resultado que ambos, pese a su diferencia, sean parecidos.
En la misma tabla podemos observar algo igualmente esperado; que el error más grande se encuentre en el set de entrenamiento, debido a lo ya mencionado previamente, el uso de la técnica de *Dropout*, donde el modelo apaga neuronas y su rendimiento no está al máximo en esta etapa, mientras que tanto los valores en el set de validación y de prueba, muy parecidos entre sí, arrojan un resultado mucho más próximo al rendimiento real de la red neuronal. Con ello, el rendimiento basado en la métrica del MAE, es un error de apenas 0.0253, cifra que si bien, no es un valor significativo hasta no tener clara la magnitud de los valores reales, si es de ayuda para un primer análisis.
De la misma forma, en la Tabla 4.2 es posible observar las mismas métricas de rendimiento para distintas situaciones, principalmente con datos no entrenados, que, en general, es el objetivo de entrenar una red neuronal; poder utilizar información ya conocida para predecir resultados en nuevos conjuntos de datos. Así, primero se

observa para un conjunto con el que fue entrenado, con Reynolds de 100,000, y a dos distintos ángulos, teniendo un MAE de 0.01 para un AOA = 0 grados y un MAE de 0.0177 para un AOA = 10 grados, siendo ambos valores muy parecidos entre sí, pese a la leve mejora en cero grados. En seguida se tienen los resultados para el conjuntos de datos no entrenados, en este segundo caso para un Re = 68500 y con los mismos ángulos, se observa un claro aumento en el valor de error y pérdida, con valores de 0.0468 y 0.0528 respectivamente. Se observa el comportamiento de la red ante una velocidad de entrada diferente, pero que está en el mismo rango de las velocidades con que se entrenó la red.
Por último, se tienen dos casos especiales con datos que se consideran como entradas fuera del rango de velocidades consideradas. Primero para un Re = 400000, la magnitud de las métricas aumenta completamente, pasando a 1.6217 y 4.0319, destacando que la pérdida es mayor probablemente por el ya mencionado comportamiento del MSE con los errores grandes. El segundo caso especial es utilizando un perfil NACA diferente al usado para el entrenamiento, en este caso tratándose de un NACA 2412, encontrando interesante que los valores de error y pérdida son de 0.0363 y 0.0030.
Con todo lo anterior, se destaca que la RNA se comporta de mejor manera en una geometría distinta que en una velocidad diferente, esto se explica con mayor detalle más adelante.
Otra forma de observar el desempeño general en los resultados de la red, es determinar si en algunas de las dos salidas de la red existe un mayor o menor desempeño, pese a estar muy relacionadas entre sí como se explicó en el marco teórico, por ello, en la Tabla 4.3 se observa que, en casi todos los casos, la precisión es mayor en la magnitud de la presión que en el de la velocidad, con excepción de cuando se trata de otro perfil.

## 5.2. Interpretación de los valores numéricos.

En los contornos de presión y velocidad de las Figuras 4.1, 4.2, 4.3, 4.4, 4.5 y 4.6, así como en la Tabla 4.4 es posible conocer la magnitud de los resultados numéricos obtenidos mediante la regresión, dichos valores muestran una distribución esperable alrededor de los perfiles aerodinámicos, por lo cual es posible conocer algunos puntos o valores clave para su rápida interpretación; como los puntos de su máxima o mínima presión, el valor de dicha característica, si existe o no desprendimiento de flujo, el valor de la presión alrededor del perfil para obtener la magnitud de su sustentación, entre otros. Sin embargo, si el objetivo es, como en este caso, medir el nivel de precisión de la regresión, no es hasta compararlos con sus valores originales (obtenidos

mediante CFD) donde cobran un mayor significado al poder medir con exactitud sus diferencias y de esta forma obtener una interpretación más1 directa de los valores numéricos, con ello, una forma sencilla de compararlos sin analizar punto por punto es observar los contornos de velocidad, pues es bien sabido que en aquellas zonas donde el fluido no interactúa con el cuerpo (el perfil en este caso) la velocidad esta completamente desarrollada y su magnitud, por lo tanto, es la configurada en la simulación, con esto en cuenta, es posible confirmar que en este punto el desempeño del modelo, pues la velocidad con un $Re = 68500$ es de 1 m/s, cosa que se refleja en los contornos de las Figuras ya mencionadas.

## 5.3. Análisis de los porcentajes de error.

Como se observa, la Tabla 4.5 presenta similitudes con la Tabla 4.2, ya que el menor porcentaje de error corresponde a los datos empleados en el entrenamiento, mientras que el mayor se registra en el número de Reynolds fuera del rango considerado (400000). Es importante señalar que, al emplear esta métrica —la cual toma en cuenta la magnitud de los valores reales—, el segundo mayor error se presenta en el perfil NACA 2412. Aun así, se mantiene el mismo comportamiento general: el modelo logra una mejor generalización en geometrías distintas que en velocidades distintas. En el primer caso, los errores alcanzan valores de 34 % y 40 %, mientras que en el segundo llegan hasta 288 %. No obstante, para los casos en los que la red fue entrenada, se obtuvieron errores de 17 % y 10 %, que, si bien no son despreciables, representan un margen relativamente bajo considerando el tipo de datos empleados como entrada, además de ser valores que en general se consideran aceptables al predecir para aplicaciones en la manufactura o automovilista (como se menciona en [28] y [29]).

Por su parte, la Tabla 4.6—que presenta los errores punto a punto en zonas específicas— muestra que existe un error considerable en la región correspondiente al contorno del perfil, cercano al 50 %. Este comportamiento puede atribuirse, en primer lugar, a que en esta zona el método de interpolación empleado en la nueva malla (griddata) presenta dificultades al tomar valores cercanos. Además, dentro del perfil no existe información disponible, lo que genera un sesgo en esa región. Cabe destacar también que el número de puntos en esta zona, clasificada como wall, es ínfimo (alrededor de 950 puntos) en comparación con el número total de puntos en cada nueva malla (aproximadamente 230000).

## 5.4. Análisis de la similitud estructural.

Aunque los resultados obtenidos directamente a partir de la regresión numérica en algunas zonas deben ser considerados, es importante destacar que la similitud en los contornos es evidente. Este hecho, junto con la precisión alcanzada al emplear un perfil distinto, demuestra que la red neuronal presenta un comportamiento altamente eficiente para detectar patrones geométricos y distribuir de manera adecuada los parámetros en la malla. Este desempeño se explica nuevamente por el tipo de datos de entrada, ya que el número de puntos empleados es de un orden mucho mayor que el de las simulaciones: 58080112 puntos en todos los conjuntos frente a únicamente 248 de estas últimas. En consecuencia, cada velocidad contó con solo 31 simulaciones, una cantidad muy reducida para que la red neuronal pudiera aprender de manera eficaz el valor real de regresión al que debía converger.
Todo lo anterior se refleja en la Tabla 4.7, donde, al utilizar el índice SSIM (*Structural Similarity Index Measure*), se obtienen resultados bastante favorables. La similitud entre contornos —considerando valores promedio, distribución y estructura— resulta casi perfecta en las pruebas correspondientes a los datos para los que fue entrenada la red, con valores entre 0.97 y 0.99. Por otro lado, para los datos no utilizados en el entrenamiento también se observan resultados positivos, aunque en menor medida; nuevamente, para el perfil NACA 2412 se alcanzan valores superiores de entre 0.70 y 0.85

## 5.5. Mejora en los tiempos de computo.

Un último análisis se presenta en la Tabla 4.9, relacionado con uno de los aspectos más relevantes y que motivaron este trabajo: el tiempo de cómputo. Aunque las simulaciones mediante CFD contribuyen de manera significativa a su reducción, también están condicionadas por el *hardware* utilizado. Como se muestra en la Tabla 4.8, con un equipo de características relativamente estándar, el tiempo de cálculo se redujo casi cuatro veces, pasando de 35 segundos a solo 9 al emplear la red neuronal. Este resultado considera únicamente el tiempo de ejecución de la simulación con 200 iteraciones, sin incluir la definición de la geometría, el dominio, la generación de la malla, la especificación de las condiciones de frontera ni el manejo del software. Si bien el tiempo requerido para el ajuste y entrenamiento del modelo puede considerarse elevado, este proceso solo debe realizarse una vez, salvo en los casos en que se desee incorporar nuevos datos.

# 6 Conclusiones

Tomando en cuenta tanto los objetivos como la hipótesis con las que se inició este trabajo, en donde se planteaba la posibilidad de utilizar metodologías del aprendizaje automático como una alternativa a la simulación de fluidos, es posible concluir que dichas implementaciones son efectivamente una posible solución y mejora a la técnica de CFD numérica, pues mediante la red neuronal fue posible reducir aun más los tiempos y costos computacionales para la obtención de los parámetros necesarios en la caracterización de un perfil aerodinámico, demostrando además tener un gran margen de mejora en esta primer implementación de métodos provenientes del Aprendizaje Automático, la cual podría adecuarse para poder funcionar a cualquier velocidad del perfil e incluso a cualquier geometría sin perder precisión y, con el *hardware* adecuado, reducir el tiempo de cálculo en mayor medida.
Tal como se mencionó en la discusión de resultados, el grado de error en la regresión de la red neuronal varía según el tipo de prueba en que se presente, sin embargo, para el tipo de prueba objetivo de la red, es decir, el caso para el mismo perfil aerodinámico en cualquier velocidad y ángulo de ataque dentro del rango, el porcentaje de error esta entre el 10 y el 17 por ciento en casi todos los puntos, un error que podríamos considerar bajo al recalcar que los datos de entrada correspondían a una mayor precisión en la distribución geométrica de puntos, más que a un menor error en el valor de la regresión; esto debido a que la cantidad total de puntos que contribuyen a este primer caso asciende hasta los 58 millones, mientras que el aprender con precisión el valor exacto de los parámetros de salida le correspondía a unas pocas simulaciones por cada número de Reynolds. Un caso muy similar y con mayor repercusión se refleja en el porcentaje de error en el contorno del perfil, pues, como ya se mencionó, cada malla nueva contenía solo 950 de estos puntos de los 230000 puntos totales, demostrando el gran sesgo que la red tuvo al entrenar para estas regiones.
Reafirmando lo anterior, se encuentran las pruebas realizadas para determinar el nivel de generalización que alcanza la RNA, demostrando que el porcentaje de error es menor al utilizar perfil NACA diferente que el obtenido al utilizar una velocidad fuera del rango entrenado. Además, más allá del valor puntual de los parámetros, se

tiene el ya mencionado *Structural similarity index measure*, donde todas las pruebas alcanzan una similitud estructural muy alta con los datos originales, llegando a una precisión casi perfecta, entre 0.97 y 0.99, para el caso objetivo y una buena (sobre el 0.62) para todos los demás casos.
En cuanto al tiempo computacional, ya se mencionó una reducción de este mismo; netamente la reducción corresponde a casi una cuarta parte del tiempo original de la simulación, esto por supuesto tomando en cuenta unicamente el tiempo neto de cálculo, pues, si se incluye todo el procedimiento ya mencionado requerido para llevar a cabo este tipo de simulaciones, desde la generación de la geometría, dominio, malla y configuración, esta reducción se incrementa significativamente, pues inclusive es posible considerar el propio tiempo para dominar el *software* dentro de el total, demostrando las ventajas de la implementación usando RNAs.
Aunado a lo anterior, la implementación de aprendizaje automático tiene igualmente ventaja al no requerir gran capacidad de *hardware*, pues el uso de las redes neuronales es posible en la gran mayoría de configuraciones computacionales, siendo que esto únicamente repercutirá en reducir o aumentar el tiempo que tome el cálculo y no en la precisión de los resultados, mientras que en la gran mayoría de programas usados en CFD se requiere un mínimo de potencia con el fin de llevar a cabo las simulaciones, normalmente limitadas además por una licencia.

## 6.1. Limitaciones del estudio y recomendaciones.

Es posible añadir que esta primera implementación de red neuronal tiene una amplia gama mejoras que pueden llevarse a cabo con un mayor tiempo de desarrollo, empezando por el número de simulaciones y datos, pues, la poca variedad que se pudo conseguir en las simulaciones con el tiempo dedicado a la realización de este trabajo se refleja en los valores de error de algunos puntos, además de que, aunque esta red neuronal funcionó de buena forma, no se descartan otras alternativas que ajusten los hiperparámetros con el objetivo de ayudar a mejorar la precisión. Una propuesta es, por ejemplo, dar mayor peso a los puntos que conformen el contorno del perfil al momento de entrenar la red, así, aunque haya en menor cantidad, estos sean tomados más en cuenta para la regresión.
Siguiendo la misma línea, esta implementación puede expandirse para muchas más velocidades en un rango de Reynolds mucho mayor, además de incluir una mayor cantidad de perfiles alares distintos, con el fin de generalizar de una mejor manera en estos aspectos y, con ello, lograr una red que pueda funcionar perfectamente para cualquier geometría y velocidad de flujo, consiguiendo de una forma rápida una

primera iteración que caracterice al perfil aerodinámico en un tiempo reducido, y con ello, ser de ayuda en propuestas de mejora de perfiles en etapas tempranas de diseño, como ya se ha demostrado en otros trabajos.
Igualmente, retomar que todas estas mejoras es posible añadirlas de una forma más eficiente en un *hardware* de mayor potencia, sobre todo al incluir el uso de una tarjeta gráfica dedicada que facilite el cálculo de una gran cantidad de datos al mismo tiempo.
Finalmente, este caso se estudió usando una arquitectura de MLPs de regresión, siendo una de las muchas opciones que se pueden implementar para la solución de este tipo de problemas, otras propuestas ya se mencionaron en el marco teórico, tal como los modelos de PINN o las CNN. Sin embargo, el campo del aprendizaje automático sigue estudiándose y en constante actualización, por lo que, tanto mejoras de los modelos ya conocidos, así como nuevos modelos, pueden ayudar a facilitar diversas tareas en la ingeniería en general y, sobre todo, en la ingeniería aeroespacial.

# Referencias


[1] M. Kaushik, *Theoretical and Experimental Aerodynamics*. 01 2019. 7

[2] F. White and M. Coello, *Mecánica de fluidos*. McGraw-Hill, 2004. 1, 4, 7, 8

[3] G. Navarro Diaz, *(Español) Simulación del efecto de la interacción de turbinas eólicas con su entorno - (English) Simulation of the effect of the wind turbines interaction with the environment*. PhD thesis, 12 2019. 9

[4] J. D. Anderson, *Introduction to Flight*. McGraw-Hill Education, 2014. 4, 8, 9, 10, 14, 18, 39

[5] Na, "Anatomy of airfoils and their performance," *Na*, Na. 11, 19, 20

[6] L. N. Cattafesta, C. J. Bahr, and J. Mathew, "Fundamentals of wind-tunnel design," 2010. 15, 16

[7] C. Kolstad, "¿qué es un transductor de presión?," 2025. 17

[8] J. Anderson, *Computational Fluid Dynamics: The Basics with Applications*. McGraw-Hill International Editions: Mechanical Engineering, McGraw-Hill, 1995. 20, 21, 22

[9] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics*. Harlow: Pearson Prentice Hall, 1995. 20, 21, 22, 23

[10] A. Trask, *Grokking Deep Learning*. Manning, 2019. 25, 26

[11] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2022. 26, 27, 28, 29, 30, 31, 32, 33, 34, 37, 45, 46

[12] M. Zanichelli, "Shape optimization of airfoils by machine learning-based surrogate models," Master's thesis, 12 2021. 1, 30, 35, 45

[13] H. Moin, H. Zeeshan Iqbal Khan, S. Mobeen, and J. Riaz, "Airfoil's aerodynamic coefficients prediction using artificial neural network," in *2022 19th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pp. 175–182, 2022. 1, 35, 45

[14] T. Zou and K. Sun, "Application and prospect of artificial intelligence in aircraft design," pp. 201–205, 11 2021. 1

[15] L. Wang, H. Zhang, C. Wang, J. Tao, X. Lan, G. Sun, and J. Feng, "A review of intelligent airfoil aerodynamic optimization methods based on data-driven advanced models," *Mathematics*, vol. 12, no. 10, 2024. 1, 35, 45

[16] C. Unlusoy, B. Maier, K. Al Handawi, T. Mathew, R. Srinivasan, M. Salz, and M. Kokkolaras, "Advancing airfoil design: A physics-inspired neural network model," 08 2024. 1, 35

[17] A. Franchini and O. Lopez, *Introducción a la Ingenieria Aeroespacial*. Alfaomega – Garceta, 2013. 4, 7, 9

[18] R. Sanchez García, *Simulación de la capa límite atmosférica neutralmente estable*. PhD thesis, 11 2020. 5, 6

[19] R. Stull, *An Introduction to Boundary Layer Meteorology*. Atmospheric and Oceanographic Sciences Library, Springer Netherlands, 1988. 5, 6

[20] I. Abbott and A. Von Doenhoff, *Theory of Wing Sections, Including a Summary of Airfoil Data*. Dover Books on Aeronautical Engineering Series, Dover Publications, 1959. 11

[21] D. Flores, "Diseño de perfiles aerodinámicos," Master's thesis, Instituto politécnico nacional, 2006. 11

[22] A. Pope and J. Harper, *Low-speed Wind Tunnel Testing*. Wiley, 1966. 14, 15, 16

[23] P. Lucerna, "Transductores de presión y de flujo," *XIII Seminario de Ingeniería Biomédica*, vol. 16, 2004. 17

[24] ANSYS, *Ansys Fluent Theory Guide*, 2021. 23

[25] J. Li and M. Zhang, "On deep-learning-based geometric filtering in aerodynamic shape optimization," *Aerospace Science and Technology*, vol. 112, p. 106603, 02 2021. 35, 45

[26] X. Song, L. Wang, and X. Luo, "Airfoil optimization using a machine learning-based optimization algorithm," *Journal of Physics: Conference Series*, vol. 2217, p. 012009, apr 2022. 35, 45

[27] D. Tyrrell, "Airfoil performance prediction through an artificial neural network," 12 2023. 35, 45

[28] IMPERIA, "Mape and supply chain forecasting: How to measure and enhance accuracy," Na. 61

[29] STARSOFT, "Mape: El secreto para predecir mejor," Na. 61

# Apéndice A

```
def preprocesar_archivos(lista_archivos, pos_minx = -0.3, pos_maxx =
    1.5, pos_miny = -0.4, pos_maxy = 0.4, codigo_naca = '4412',
   umbral_wall = 0.0025):

    dataframes = []
    #Preparar los puntos para la nueva malla.
    for archivo in lista_archivos:
        x_new = np.linspace(-0.27, 1.47, 500)
        y_new = np.linspace(-0.37, 0.37, 500)
        X_new, Y_new = np.meshgrid(x_new, y_new)
        x_flat = X_new.flatten()
        y_flat = Y_new.flatten()
        #Crear los puntos del perfil.
        puntos_perfil = np.linspace(0, 1, 3000)
        naca_coordenadas = naca_4d(puntos_perfil, codigo_naca)

        try:
            #Cargar los archivos
            ruta = Path(archivo)
            if ruta.suffix == ".csv":
                df = pd.read_csv(ruta, delimiter=",",skiprows=5)
                #Preparar la region que nos interesa
                df = df.drop(columns=[' Z [ m ]'])
                df = df[(df['X [ m ]'] >= pos_minx)
                & (df['X [ m ]'] <= pos_maxx)
                & (df[' Y [ m ]'] >= pos_miny)
                & (df[' Y [ m ]'] <= pos_maxy)]

                #Obtener valores para luego interpolar los datos en
                #la nueva malla
                x = df['X [ m ]'].copy().to_numpy()
                y = df[' Y [ m ]'].copy().to_numpy()
                p = df[' Pressure [ Pa ]'].copy().to_numpy()
                v = df[' Velocity [ m s^-1 ]'].copy().to_numpy()
```

```
33
34                  #Obtener valores del nombre del archivo
35                  AOA = re.search(r'_(\-?\d+)', ruta.stem)
36                  valor = int(AOA.group(1)) if AOA else None
37                  Re = re.search(r"(\d+)$", ruta.stem)
38                  valorRe = int(Re.group(1))
39
40                  naca_coordenadas = rotate_points_around_tip(
41                  naca_coordenadas, -valor)
42                  naca_coordenadas = np.vstack(
43                  [naca_coordenadas, naca_coordenadas[0]])
44
45                  path = Pathlib(naca_coordenadas)
46                  grid_points = np.column_stack(
47                  (X_new.ravel(), Y_new.ravel()))
48
49                  #Filtrar los puntos y excluir los que esten
50                  #dentro del contorno del perfil.
51                  inside = path.contains_points(grid_points)
52                  outside = ~inside
53                  indices_fuera = np.where(outside)[0]
54                  X_filtered = x_flat[indices_fuera]
55                  Y_filtered = y_flat[indices_fuera]
56
57                  #Interpolar los datos en la nueva malla
58                  #con los puntos filtrados previamente
59                  P_new = griddata(
60                  (x, y), p, (X_filtered, Y_filtered), method='cubic')
61                  V_new = griddata(
62                  (x, y), v, (X_filtered, Y_filtered), method='cubic')
63                  p_flat = P_new.flatten()
64                  v_flat = V_new.flatten()
65
66                  #Obtener atributo de la distancia minima de
67                  cualquier punto al contorno
68                  distancias = cdist(
69                  np.column_stack((X_filtered, Y_filtered))
70                  , naca_coordenadas)
71                  dist_minima = distancias.min(axis=1)
72
73                  #Combinar todos los atributos en un dataframe
74                  df = pd.DataFrame({
75                      'X [ m ]': X_filtered,
76                      ' Y [ m ]': Y_filtered,
77                      ' Pressure [ Pa ]': p_flat,
78                      ' Velocity [ m s^-1 ]': v_flat,
```

```
                    'Distancia_min': dist_minima,
                    })
                df['AOA'] = valor
                df['Reynolds'] = valorRe
                df['Wall'] = np.where(
                dist_minima < umbral_wall, 1, 0)
                dataframes.append(df)

                print(f"Cargado: {ruta.name}")
            else:
                print(f"{ruta.name} no es un archivo CSV. "
                "Se omitira.")
        except Exception as e:
            print(f"Error al leer {archivo}: {e}")
            print(e)

    # Unir todos los DataFrames en uno solo y mezclarlos
    aleatoriamente
    random.shuffle(dataframes)
    df_final = pd.concat(dataframes, ignore_index=True)
    return df_final
```

Listing 6.1: Función de Python para el preprocesamiento de archivos.

```
def build_model(hp):
    #Rango de hyperparametros de capas ocultas, numero de neuronas
    #y taza de aprendizaje.
    n_hidden = hp.Int(
    "n_hidden", min_value=5, max_value=24, default=2)
    n_neurons = hp.Int(
    "n_neurons", min_value=50, max_value=252)
    learning_rate = hp.Float(
    "learning_rate", min_value=1e-5, max_value=1e-3,
                              sampling="log")
    #Seleccion del optimizador.
    optimizer = hp.Choice("optimizer", values=["sgd", "adam"])
    if optimizer == "sgd":
        optimizer = tf.keras.optimizers.SGD(learning_rate=
    learning_rate, nesterov=True)
    else:
        optimizer = tf.keras.optimizers.Adam(learning_rate=
    learning_rate)

    #Seleccion de la funcion de perdida.
    loss_fn = hp.Choice("loss", ["huber", "mse"])
    if loss_fn == "huber":
        loss = tf.keras.losses.Huber(delta=1.0)
```

```
    else:
        loss = "mse"

    #Seleccion de la taza de dropout.
    dropout_rate = hp.Float(
    "dropout_rate", min_value=0.1, max_value=0.3, step=0.05)

    #Seleccion de Beta para la funcion de activacion swish.
    beta = hp.Float("beta", min_value=0.5, max_value=2, step=0.1)
    swish = swish_beta(beta)
    get_custom_objects().update({"swish_beta": Activation(swish)})

    #Creacion del modelo con las capas, neuronas y otros
    #hyperparametros seleccionados.
    model = tf.keras.Sequential()
    model.add(
    tf.keras.layers.Input(shape = x_train.shape[1:]))
    for _ in range(n_hidden):
        model.add(tf.keras.layers.Dense(
                n_neurons, activation = swish,
                kernel_initializer = "he_normal"))
        if hp.Boolean("use_batchnorm"):
            model.add(tf.keras.layers.BatchNormalization())
        model.add(tf.keras.layers.Dropout(dropout_rate))
    model.add(tf.keras.layers.Dense(2))
    model.compile(loss=loss, optimizer=optimizer,
                  metrics=["mae"])
    return model

    class MyClassificationHyperModel(kt.HyperModel):
    def build(self, hp):
        tf.keras.backend.clear_session()
        hp_batch_size = hp.Choice(
        "batch_size", [32, 64, 128, 256, 512])
        return build_model(hp)

    def fit(self, hp, model, X, y, **kwargs):
        return model.fit(
        X, y, batch_size=hp.get("batch_size"), **kwargs)
```

Listing 6.2: Función para crear y ajustar el modelo.

```
hyperband_tuner = kt.Hyperband(
    MyClassificationHyperModel(), objective="val_mae"
    ,max_epochs=5, factor=3, hyperband_iterations=2
    ,overwrite=True, directory="PresVel_6"
    ,project_name="hyperband_presvel")
early_stopping_cb = tf.keras.callbacks.EarlyStopping(patience=10)
lr_scheduling = tf.keras.callbacks.ReduceLROnPlateau(
    factor=0.5, patience=5)
hyperband_tuner.search(
    x_train, y_train,  epochs=10,
    validation_data=(x_valid, y_valid),
    callbacks=[early_stopping_cb, lr_scheduling])
```

Listing 6.3: Código para empezar el ajuste del modelo.

```
history = best_model.fit(
    x_train, y_train, epochs=100,
    validation_data=(x_valid, y_valid),batch_size=128,
    callbacks=[tensorboard_cb, early_stop, lr_scheduling])
```

Listing 6.4: Código para realizar el entrenamiento con el modelo ajustado.

```
Epoch 1/100
290405/290405 [==============================] - 1952s 7ms/step -
   loss: 0.0278 - mae: 0.1085 - val_loss: 0.0195 - val_mae: 0.0233 -
    lr: 1.4327e-04
Epoch 2/100
290405/290405 [==============================] - 1976s 7ms/step -
   loss: 0.0269 - mae: 0.1076 - val_loss: 0.0254 - val_mae: 0.0251 -
    lr: 1.4327e-04
Epoch 3/100
290405/290405 [==============================] - 1979s 7ms/step -
   loss: 0.0262 - mae: 0.1066 - val_loss: 0.0174 - val_mae: 0.0223 -
    lr: 1.4327e-04
Epoch 4/100
290405/290405 [==============================] - 1890s 7ms/step -
   loss: 0.0257 - mae: 0.1059 - val_loss: 0.0480 - val_mae: 0.0278 -
    lr: 1.4327e-04
Epoch 5/100
290405/290405 [==============================] - 1963s 7ms/step -
   loss: 0.0251 - mae: 0.1048 - val_loss: 0.0257 - val_mae: 0.0315 -
    lr: 1.4327e-04
Epoch 6/100
290405/290405 [==============================] - 1974s 7ms/step -
   loss: 0.0247 - mae: 0.1041 - val_loss: 0.0200 - val_mae: 0.0335 -
    lr: 1.4327e-04
Epoch 7/100
```

```
290405/290405 [==============================] - 1889s 7ms/step -
   loss: 0.0244 - mae: 0.1035 - val_loss: 0.0269 - val_mae: 0.0290 -
    lr: 1.4327e-04
Epoch 8/100
290405/290405 [==============================] - 1889s 7ms/step -
   loss: 0.0239 - mae: 0.1029 - val_loss: 0.0475 - val_mae: 0.0334 -
    lr: 1.4327e-04
Epoch 9/100
290405/290405 [==============================] - 1867s 6ms/step -
   loss: 0.0228 - mae: 0.1016 - val_loss: 0.0228 - val_mae: 0.0266 -
    lr: 7.1635e-05
Epoch 10/100
290405/290405 [==============================] - 1880s 6ms/step -
   loss: 0.0226 - mae: 0.1013 - val_loss: 0.0366 - val_mae: 0.0256 -
    lr: 7.1635e-05
Epoch 11/100
290405/290405 [==============================] - 1876s 6ms/step -
   loss: 0.0225 - mae: 0.1009 - val_loss: 0.0303 - val_mae: 0.0263 -
    lr: 7.1635e-05
Epoch 12/100
290405/290405 [==============================] - 1873s 6ms/step -
   loss: 0.0222 - mae: 0.1005 - val_loss: 0.0292 - val_mae: 0.0210 -
    lr: 7.1635e-05
Epoch 13/100
290405/290405 [==============================] - 1867s 6ms/step -
   loss: 0.0220 - mae: 0.1002 - val_loss: 0.0302 - val_mae: 0.0262 -
    lr: 7.1635e-05
Epoch 14/100
290405/290405 [==============================] - 1865s 6ms/step -
   loss: 0.0215 - mae: 0.0995 - val_loss: 0.0159 - val_mae: 0.0248 -
    lr: 3.5818e-05
Epoch 15/100
290405/290405 [==============================] - 1864s 6ms/step -
   loss: 0.0215 - mae: 0.0993 - val_loss: 0.0195 - val_mae: 0.0243 -
    lr: 3.5818e-05
Epoch 16/100
290405/290405 [==============================] - 1849s 6ms/step -
   loss: 0.0213 - mae: 0.0993 - val_loss: 0.0335 - val_mae: 0.0221 -
    lr: 3.5818e-05
Epoch 17/100
290405/290405 [==============================] - 1864s 6ms/step -
   loss: 0.0213 - mae: 0.0990 - val_loss: 0.0262 - val_mae: 0.0238 -
    lr: 3.5818e-05
Epoch 18/100
290405/290405 [==============================] - 1843s 6ms/step -
   loss: 0.0212 - mae: 0.0990 - val_loss: 0.0360 - val_mae: 0.0241 -
```

```
    lr: 3.5818e-05
Epoch 19/100
290405/290405 [==============================] - 1862s 6ms/step -
   loss: 0.0213 - mae: 0.0989 - val_loss: 0.0357 - val_mae: 0.0250 -
    lr: 3.5818e-05
Epoch 20/100
290405/290405 [==============================] - 1855s 6ms/step -
   loss: 0.0209 - mae: 0.0988 - val_loss: 0.0341 - val_mae: 0.0216 -
    lr: 1.7909e-05
Epoch 21/100
290405/290405 [==============================] - 1864s 6ms/step -
   loss: 0.0208 - mae: 0.0984 - val_loss: 0.0317 - val_mae: 0.0226 -
    lr: 1.7909e-05
Epoch 22/100
290405/290405 [==============================] - 1863s 6ms/step -
   loss: 0.0209 - mae: 0.0984 - val_loss: 0.0258 - val_mae: 0.0232 -
    lr: 1.7909e-05
Epoch 23/100
290405/290405 [==============================] - 1871s 6ms/step -
   loss: 0.0208 - mae: 0.0984 - val_loss: 0.0240 - val_mae: 0.0315 -
    lr: 1.7909e-05
Epoch 24/100
290405/290405 [==============================] - 1911s 7ms/step -
   loss: 0.0208 - mae: 0.0984 - val_loss: 0.0270 - val_mae: 0.0231 -
    lr: 1.7909e-05
Epoch 25/100
290405/290405 [==============================] - 2022s 7ms/step -
   loss: 0.0207 - mae: 0.0983 - val_loss: 0.0278 - val_mae: 0.0236 -
    lr: 8.9544e-06
Epoch 26/100
290405/290405 [==============================] - 2030s 7ms/step -
   loss: 0.0206 - mae: 0.0981 - val_loss: 0.0334 - val_mae: 0.0232 -
    lr: 8.9544e-06
Epoch 27/100
290405/290405 [==============================] - 2004s 7ms/step -
   loss: 0.0207 - mae: 0.0982 - val_loss: 0.0327 - val_mae: 0.0231 -
    lr: 8.9544e-06
Epoch 28/100
290405/290405 [==============================] - 2051s 7ms/step -
   loss: 0.0206 - mae: 0.0981 - val_loss: 0.0329 - val_mae: 0.0238 -
    lr: 8.9544e-06
Epoch 29/100
290405/290405 [==============================] - 2026s 7ms/step -
   loss: 0.0206 - mae: 0.0981 - val_loss: 0.0267 - val_mae: 0.0230
```

Listing 6.5: Historial de entrenamiento.