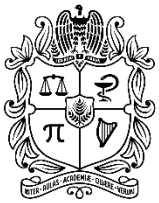


UNIVERSIDAD
NACIONAL
DE COLOMBIA

Diseño de bombas centrífugas utilizando el método de optimización topológica y computación paralela en la nube.

Xavier Andrés Arcentales Bastidas

Universidad Nacional de Colombia

Facultad de Minas, Departamento de Ingeniería Mecánica

Medellín, Colombia

2021

Diseño de bombas centrífugas utilizando el método de optimización topológica y computación paralela en la nube.

Xavier Andrés Arcentales Bastidas

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título

Magister en Ingeniería Mecánica

Director (a):

Wilfredo Montealegre Rubio Ph.D.

Línea de Investigación:

Optimización topológica aplicada al diseño de sistemas mecánicos

Grupo de Investigación:

Diseño y Optimización Aplicada

Universidad Nacional de Colombia

Facultad de Minas, Departamento de Ingeniería Mecánica

Medellín, Colombia

2021

“En los momentos de crisis, solo la imaginación es más importante que el conocimiento”

Albert Einstein

“No hay enseñanza sin investigación ni investigación sin enseñanza”

Paulo Freire

“En principio la investigación necesita más cabezas que medios”

Severo Ochoa

Agradecimientos

Agradezco a Dios por guiarme en este proceso de aprendizaje.

Agradezco a mi familia por su constante apoyo.

Agradezco a mi novia Estefanía Bucheli por su apoyo incondicional.

Agradezco a mi profesor Wilfredo Montealegre por su paciencia en la elaboración de mi tesis de maestría y por sus enseñanzas en todo este proceso de investigación.

Agradezco a cada uno de mis compañeros del grupo de investigación en Diseño y Optimización Aplicada (DOA), especialmente a Francisco Javier Ramírez y Esteban Foronda Obando por sus oportunas explicaciones y apoyo a lo largo de esta investigación.

Agradezco a la Universidad Nacional de Colombia Sede Medellín por haberme acogido como estudiante extranjero y ser parte de este proceso de aprendizaje.

Resumen

Este trabajo se focaliza específicamente en bombas centrífugas de flujo radial, cuya componente de velocidad axial no es considerada en comparación con las velocidades tangenciales y radiales, dando lugar al diseño de álabes en dos dimensiones. En consecuencia, como el diseño de álabe es una tarea compleja, debido a la cantidad de parámetros geométricos libres involucrados (radio de curvatura, ángulo del alabe, etc.), se usa el Método de Optimización Topológica (MOT). La selección del MOT en comparación con otro método de optimización (paramétrica, material o de forma) se justifica por el simple hecho de que este método combina todos los métodos anteriores (optimización más robusta).

Para el diseño topológico de los álabes, se solucionan las ecuaciones de estado de Navier-Stokes por medio del método de los elementos finitos (MEF), para generar los campos de velocidad y presión, que son los campos de distribución que simulan el comportamiento fluidodinámico dentro del impeler, para posteriormente minimizar dos fenómenos físicos (funciones objetivo) que son la energía de disipación viscosa y la vorticidad. Estas funciones objetivo se combinan en una función bi-objetivo mediante el método de la suma ponderada, dando así mayor minimización a una función con respecto a la otra. Adicionalmente se fórmula el problema de optimización agregando la fuerza de fricción artificial de Darcy en las ecuaciones de Navier-Stokes para flujo incompresible, el método adjunto discreto se utiliza para hallar las sensibilidades y se usa el método de las asíntotas móviles para actualizar la variable de diseño gamma.

Para la solución de las ecuaciones de Navier-Stokes en conjunto al problema de optimización, se desarrolla un algoritmo computacional en MATLAB. Adicionalmente se paraleliza el algoritmo desarrollado y se ejecuta el código con la utilización de varios Cores (núcleos CPU) en la nube con dos proveedores diferentes: **a)** Amazon Web Services (máquina virtual) y **b)** Equinix (máquina bare-metal), con el objetivo de acelerar el proceso de diseño de los álabes. El resultado obtenido de la topología cuando se considera únicamente la minimización de la energía de disipación $(w_d = 1 \text{ y } w_r = 0)$ es 5.88 [Watts]. Adicionalmente, el desempeño que se obtiene cuando se considera la minimización de la energía de disipación y vorticidad $(w_d = 0.8 \text{ y } w_r = 0.2)$ es de 5.94 [Watts]. Estas topologías son extendidas en un diseño de dominio completo en un modelo 3D usando ANSYS FLUENT, con el objetivo de validar la minimización de las funciones objetivos obtenidas por el Método de Optimización Topológica (MOT).

Palabras claves: Energía de disipación, Vorticidad, Optimización topológica, Computación Paralela, Nube, bombas centrífugas, método de los elementos finitos.

Abstract

Design of centrifugal pump rotors by using the topology optimization method and parallel cloud computing

This work focuses specifically on radial flow centrifugal pumps, whose axial velocity component is neglected compared to tangential and radials velocities, giving rise to the analysis of the blades in two dimensions. In addition, due to the design of flow machines is still a difficult task, mainly due to the large number of free geometrical parameters involved (radius of curvature, blade angle, etc.), the Topological Optimization Method (TOM) is used in this work. The selection of the TOM compared to another optimization method (parametric, material or shape) is justified by the simple fact that this method combines all the previous methods (more robust optimization).

For the topological design of the blades, the Navier-Stokes equations of state are solved by means fo the finite element method (FEM) to generate the velocity and pressure fields, which are the distribution fields that simulate the fluid-dynamic behavior within the impeller, for later minimize two physical phenomena (objective functions) which are viscous dissipation energy and vorticity. These objective functions are combined into a bi-objective function using the weighted sum method, thus giving greater minimization to one function concerning the other. Additionally, the optimization problem is formulated by adding the artificial Darcy friction force in the Navier-Stokes equations for incompressible flow, the discrete adjoint method is used to find the sensitivities and the moving asymptotes method is used to update the design variable gamma.

For the solution of the Navier-Stokes equations in conjunction with the optimization problem, a computational algorithm is developed in MATLAB. Additionally, the developed algorithm is parallelized, and the code is executed with the use of several cores (CPU cores) in the cloud on two different platforms: **a)** Amazon Web Services (virtual machine) and **b)** Equinix (bare-metal machine), to speed up the blade design process. The performance obtained from the topology result when only the minimization of the energy dissipation is considered ($w_d = 1 \text{ y } w_r = 0$) is 5.88 [Watts]. Additionally, the performance obtained when considering the minimization of the energy dissipation and vorticity ($w_d = 0.8 \text{ y } w_r = 0.2$) is 5.94 [Watts]. After these topologies results, they are extended in an entire domain design in a 3D model using ANSYS FLUENT, with the objective to validate the minimization of the objective functions obtained by the Topology Optimization Method (TOM).

Keywords: Energy dissipation, Vorticity, Topology Optimization, Parallel Computing, Cloud, centrifugal pumps, finite element method.

CONTENIDO

PÁG

LISTA DE FIGURAS

PÁG

LISTA DE TABLAS

PÁG

LISTA DE PSEUDO-CÓDIGOS

PÁG

CAPÍTULO 1: INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

Las turbomáquinas están ampliamente empleadas en la industria, y se utilizan en varias aplicaciones a diferentes escalas. Pueden ser no solo máquinas de gran tamaño, como turbinas utilizadas en plantas termoeléctricas e hidroeléctricas, sino también bombas de pequeño tamaño utilizadas en aplicaciones médicas, como dispositivos de asistencia ventricular [1]. Sin embargo, el diseño y la predicción del rendimiento de estas máquinas sigue siendo una tarea difícil, principalmente debido a la gran cantidad de parámetros geométricos involucrados. Además, el costo y el tiempo significativo de los métodos de prueba y error para construir los prototipos reducen los márgenes de beneficio de los fabricantes. Las simulaciones numéricas pueden proporcionar información muy precisa sobre el comportamiento del flujo de fluido en estas máquinas, ayudando así a los ingenieros a obtener una evaluación integral del desempeño de un diseño particular [2,3], es por eso que en este trabajo de investigación se estudia el comportamiento fluidodinámico de un fluido en particular dentro de un impeler con diferentes topologías de álabes.

Para el diseño de los álabes del impeler en turbomáquinas, es necesario solucionar las ecuaciones de estado del campo de velocidad y del campo de presión, ya que conociendo el comportamiento del flujo dentro de las bombas centrífugas se pueden conocer fenómenos físicos como la vorticidad, energía de disipación viscosa, etc., que son factores que tienen incidencia sobre el rendimiento hidráulico de las mismas y que son de estudio en este trabajo de investigación. En la Figura 1.1 se puede observar como la configuración geométrica (trabajo realizado por J.S. Romero) [4] de los álabes de los rotores de bombas centrífugas cambia el campo de velocidad considerablemente, dando lugar a un incremento o decremento de la eficiencia hidráulica (η), ya que la potencia hidráulica está en función de la velocidad y la presión. La eficiencia está dada por la (Ecc. 1.1), donde Q es el caudal del fluido, Δ_P el cambio de presión, T el torque en el eje y Ω la velocidad angular del rotor [5].

$$\eta = \frac{Pot._{hidráulica}}{Pot._{freno}} = \frac{Q*\Delta_P}{T*\Omega} \qquad \text{(Ecc. 1.1)}$$

En la Figura 1.1a-c se observa que la velocidad relativa no es uniforme a lo largo de la superficie exterior, y los valores más altos (4.5 m/seg) de distribución de la velocidad relativa se dan en la esquina superior debido al efecto de movimiento de rotación; en cambio, en la Figura 1.1d se observa la influencia de un contorno irregular que afecta naturalmente al flujo. Aparte de la cantidad de parámetros geométricos que ya se mencionaron, otro problema que se observa en el diseño de rotores es el tiempo que le toma al diseñador realizar varias simulaciones para cada configuración de álabe, ya que cada configuración requiere un bosquejo previo, para posteriormente realizar la simulación y su respectivo análisis de resultados. Por lo tanto, una solución para este problema es aplicar técnicas de optimización topológica [6] que mediante funciones objetivos específicas (ver CAPITULO 3: energía de disipación y vorticidad), permita obtener diseños adecuados con base a los requerimientos y al mismo tiempo evitar realizar un bosquejo y análisis de resultados para cada configuración.

A raíz de la solución de la configuración del alabe, al aplicar el Método de Optimización Topológica (MOT) como técnica de optimización [6], surge otro problema: el incremento considerable de los recursos computacionales, ya que existe la necesidad de la implementación de algoritmos con técnicas de programación matemática avanzada para la solución de dicho problema [7]. El tiempo de ejecución de los cálculos incrementa debido a la naturaleza de la ecuación a ser resuelta, método de discretización, técnica de solución empleada y debido a que el MOT es una técnica iterativa, los cálculos pueden ir de 60 a algunos miles de iteraciones [8].

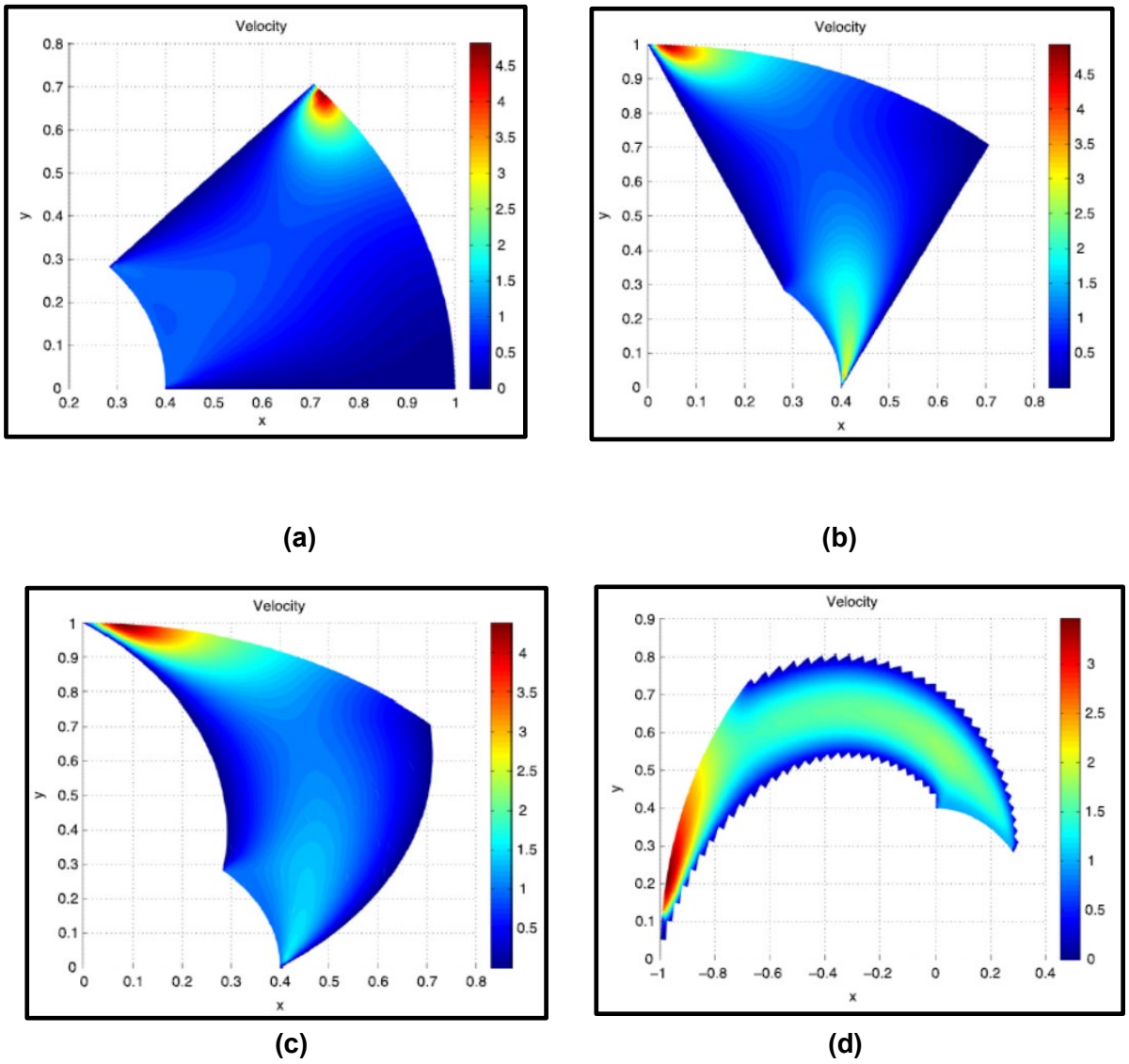


Figura 1.1- Campo de velocidades relativas de las siguientes topologías: **(a)** Alabe recto, **(b)** Alabe recto inclinada, (**c)** Alabe curvo y **(d)** Alabe curvo involuta [4].

El aumento del tiempo computacional también se ve afectado en el estudio de independencia de mallas para problemas de flujo, En la Figura 1.2 se observa que la precisión y la convergencia de los resultados es dependiente al número de malla a partir de 1.65 millones de elementos[9]. Adicionalmente el tiempo computacional puede aumentar aún más si se aborda el análisis de flujo en bombas centrífugas de forma tridimensional (3D).

Con la ayuda de técnicas de dinámica de fluidos computacionales (CFD), investigadores han realizado diferentes estudios sobre bombas centrífugas [2,9,10], como el trabajo realizado por Sun-Sheng Yang [9], que por medio de un método teórico de análisis de velocidades obtuvo correlaciones entre bomba y bomba como turbina (BCT), prediciendo la eficiencia hidráulica de la BCT. Esta validación de la eficiencia hidráulica la realizó mediante una simulación numérica en ANSYS CFX en 3D y mediante un banco de prueba experimental, variando los caudales hidráulicos. Él obtuvo resultados muy aproximados comparando la eficiencia hidráulica obtenida por método teórico vs. la simulación numérica en ANSYS CFX y el banco de prueba experimental, con un error relativo máximo de 3.33 % para un caudal Q/Q_{BEP}= 1.27 (Q_{BEP} es el caudal en el mejor punto de eficiencia). En esta investigación no se menciona el tiempo computacional para la simulación en ANSYS, pero el dominio de la bomba se dividió en 5 partes (tubería de entrada de la bomba, cámara delantera, cámara trasera, impulsor y voluta) dando un total de 1'297.308 elementos discretizados para la obtención de los resultados. Otra investigación, con el uso de CFD en 3D para bombas centrífugas, fue el trabajo realizado por K.W. Cheah [10], quién simuló el flujo en una bomba centrífuga en condiciones de diseño (punto de funcionamiento en el que la bomba tiene su máxima eficiencia) y fuera de diseño (punto de máxima eficiencia desplazado) usando ANSYS-CFX en 3D, para observar el comportamiento del flujo en estas dos condiciones. El impulsor de la bomba constaba de 6 álabes y la simulación fue realizada con un modelo de turbulencia $\mathrm{k}-\epsilon$ estándar, con diferentes tasas de flujo $\left(Q = Q_{design}\,, Q = 1.45\ Q_{\mathrm{design}}\ y\ Q = 0.43\ Q_{\mathrm{design}}\right)$. Los resultados obtenidos para el impulsor, en el punto de diseño (Figura 1.3 b), son flujos bastantes suaves que sigue la curvatura del álabe; sin embargo, obtiene una separación de flujo (separación de capa límite) en el borde de ataque, debido a la condición de entrada no tangencial. Por el contrario, cuando la bomba centrífuga está operando bajo condiciones de velocidad de flujo fuera de diseño, se desarrolla flujo inestable en el paso del impulsor, tal como se muestra en la Figura 1.3a y c. En este trabajo solo el dominio del impulsor consta de 1´004.139 de elementos discretizados, lo que genera también un tiempo computacional significativo. Por último, en el trabajo de B. Jafarzadeh [2] en la simulación de flujo de una bomba centrífuga en ANSYS FLUENT, se utilizan 5'839.589 elementos discretizados, tiempo que no fue cuantificado en esta investigación pero que el autor si menciona la falta de memoria y velocidad de procesamiento en dicho trabajo al ir resolviendo el problema de simulación con diferentes modelos de turbulencia. Podemos ver que estas simulaciones no son prácticas para el diseño de rotores en bombas centrífugas usando técnicas de CFD y más aún si se usa el MOT, que incrementaría mucho más el tiempo de cómputo en donde se requiere constantemente estar cambiando variables y simulando nuevos valores.

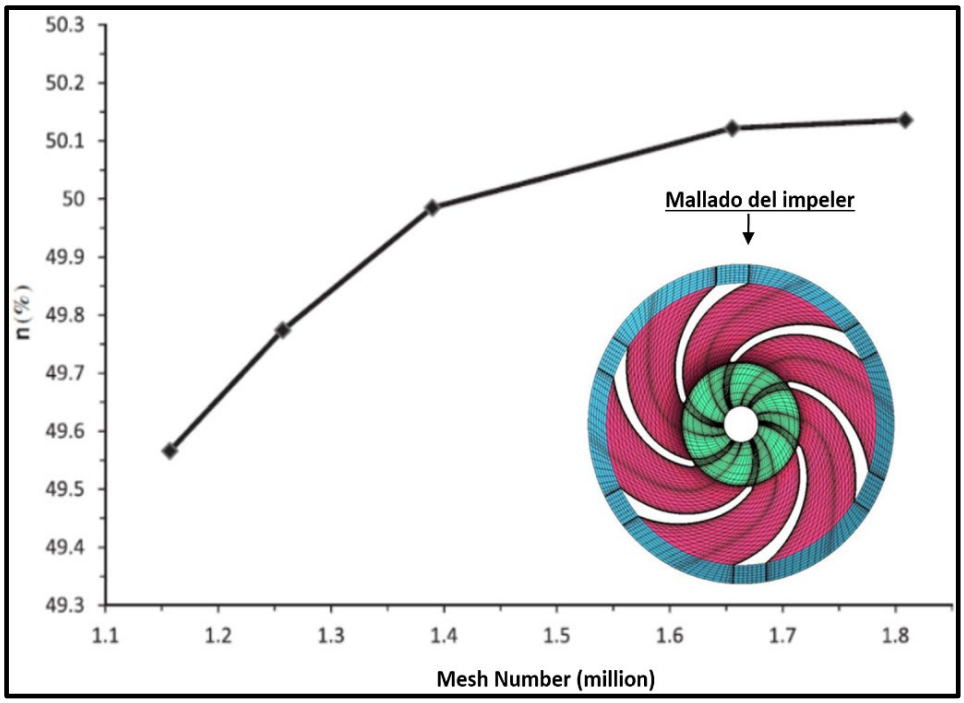


Figura 1.2.- Eficiencia hidráulica, ver (Ecc. 1.1)) vs. Número de Malla para un dominio computacional completo [9].

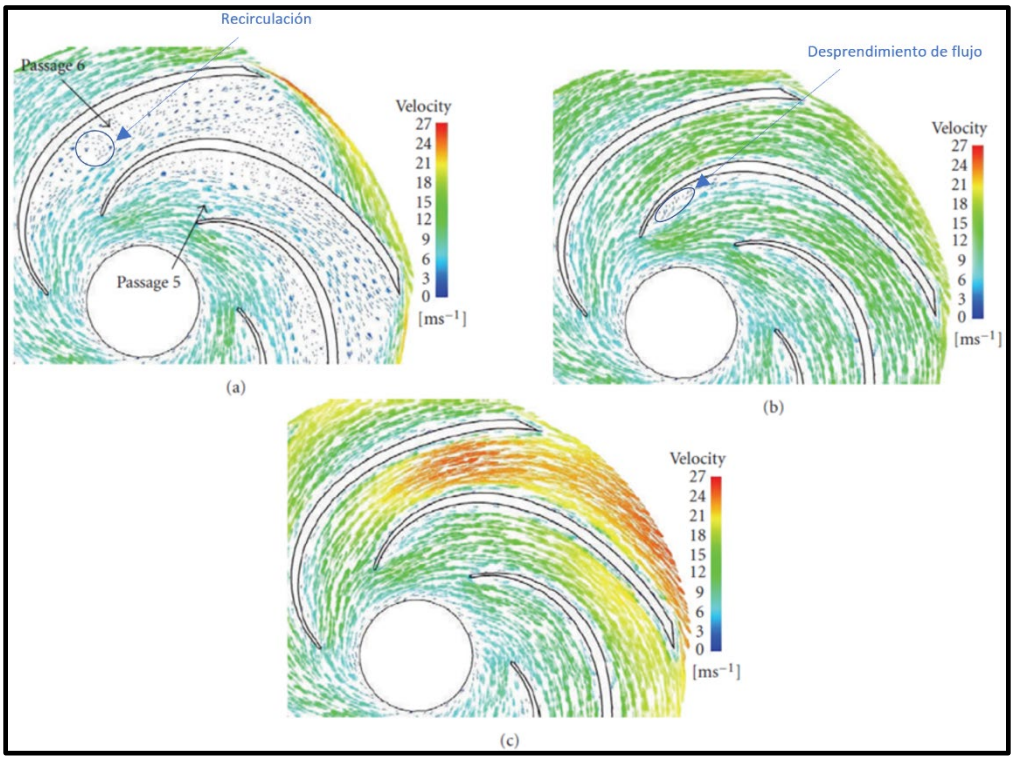


Figura 1.3.- Velocidad relativa dentro del paso del impulsor: **(a)** Punto fuera de diseño, $Q = 0.43\ Q_{design}$, **(b)** Punto de diseño $Q = Q_{design}$, **(c)** Punto fuera de diseño, $Q = 1.45\ Q_{design}$. Imagen adaptada de [10].

Por las razones anteriores, se observa que los requerimientos computacionales para el diseño de rotores en bombas centrífugas son altos para que estos sean abordados en computadores de uso general (estándar). Es por eso que en la actualidad algunas empresas y universidades optan por computación en la nube, ya que se puede realizar un sin número de tareas, desde almacenar información hasta desarrollar aplicaciones, análisis de datos e incluso realizar simulaciones en ingenierías con el uso de computación paralela para disminuir el tiempo computacional.

De todas estas tareas, una ventaja de las más atractivas al usar computación en la nube son los bajos costos. Al dejar la responsabilidad de la implementación de la infraestructura al proveedor, el cliente no tiene que preocuparse por capacitar al personal para la configuración y mantenimiento de éstos, por el desarrollo de softwares en algunos casos y por la compra de equipos de cómputo. En la actualidad, softwares y plataformas como ANSYS, MATLAB, Rescale, Equinix han venido implementando computación paralela (HPC) en la nube; y con esto se ha permitido realizar simulaciones de toda índole, con la ventaja de reducir el tiempo y costo computacional. Con respecto a lo previamente expuesto, se pretende usar computación paralela en la nube para afrontar el problema de diseño de rotores en bombas centrífugas usando el MOT por ser una opción novedosa, económica y eficiente.

Con base a todo este planteamiento del diseño de rotores en bombas centrífugas, surge entonces la pregunta de investigación: ¿Cómo determinar de manera óptima, la ubicación, forma y cantidad de los álabes minimizando la energía de disipación viscosa y vorticidad para el diseño de bombas centrífugas usando computación paralela en la nube? Para resolver este problema, se propone responder la pregunta anterior por medio del método de optimización topológica (MOT) usando como funciones objetivo la minimización de la energía de disipación viscosa y la vorticidad, y como variables de diseño la pseudo-densidad γ; la cual toma valores de $\gamma = 1$ cuando es completamente fluido y valores de $\gamma = 0$ cuando es completamente sólido. La aceleración de los cálculos de optimización se realiza con dos proveedores en la nube ya mencionados anteriormente.

Adicionalmente para clarificar el problema propuesto se muestra un esquema del método de optimización topológica aplicado a la búsqueda de posición, forma y cantidad óptima de álabes para el diseño de bombas centrífugas, minimizando la energía de disipación viscosa y vorticidad, ver Figura 1.4.

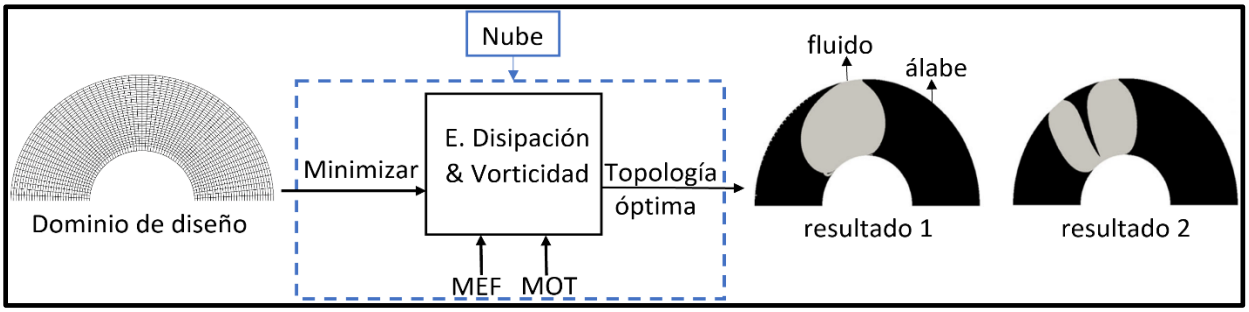


Figura 1.4.- Dominio computacional previo al diseño de alabes en bombas centrífugas (izquierda) y ubicación de álabes dentro del dominio computacional usando el método de optimización topológica (MOT).

1.2.- JUSTIFICACIÓN

1.2.1- JUSTIFICACIÓN RSL

Una argumentación elemental que motiva este proyecto de investigación es la ausencia de una herramienta de cómputo que permita diseñar rotores de bombas usando el MOT, realizando los cálculos numéricos mediante computación paralela en la nube, permitiendo trabajar tanto en la unidad central de procesamiento (CPU) como en la unidad de procesamiento gráfico (GPU) y dando la opción al usuario que trabaje con diferentes arquitecturas como: máquina RISC avanzada (ARM), elastic MapReduce por sus siglas en inglés (EMR), nube virtual privada (VPC), etc. Estas arquitecturas son administradas por lo general por proveedores como Amazon Web Services, Azure, Google Cloud, Equinix, etc. Esta ausencia de herramientas, que combine el MOT y computación paralela en la nube en el diseño de bombas centrífugas se la comprueba realizando una revisión sistemática de la literatura (RSL), siguiendo la metodología propuesta por Kitchenman 2004 [11], en la base de datos de Scopus. Particularmente, se realiza una combinación de palabras claves al tema de investigación, ver Figura 1.5.

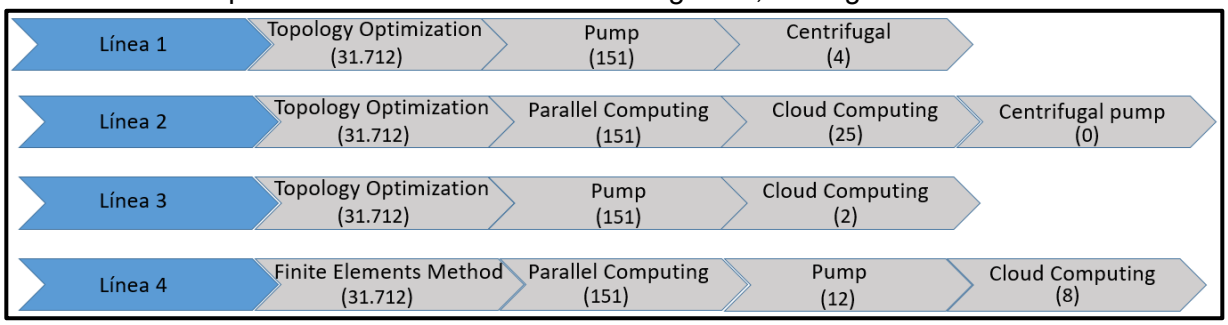


Figura 1.5.- Resultados de la RSL. Los números entre paréntesis representan el total de publicaciones por palabra clave y combinaciones. **Fuente:** www.scopus.com.

En la línea 1 de búsqueda de la RSL, se encuentra un trabajo de importancia en la aplicación de optimización topológica aplicado al diseño de rotores de turbomáquinas, considerando el comportamiento del fluido (disipación de energía, vorticidad y potencia mecánica) para la mejora en el rendimiento de las bombas centrífugas (Romero & Silva, 2014) [4]. En este trabajo de investigación, se desarrolla un algoritmo computacional con el fin de solucionar las ecuaciones de Navier-Stokes, por medio del método de los elementos finitos (MEF), para analizar el comportamiento del flujo dentro de la turbomáquina. Los resultados topológicos se obtienen por medio de la minimización de una función multiobjetivo que involucra funciones de energía de disipación, vorticidad y potencia mecánica. En este trabajo no se menciona el tiempo computacional en la ejecución del algoritmo para la solución del problema, pero cabe recalcar que, al ser un problema iterativo, incrementa el tiempo en la solución de las ecuaciones. En esta misma línea de búsqueda, las otras 3 investigaciones tratan sobre conferencias internacionales en avances de materiales y fabricación, sobre diseño mecánico e ingeniería en potencia y ciencias de los materiales e ingeniería y, no particularmente, en el diseño de bombas centrífugas por medio de optimización topológica. En la línea 3 y 4 de búsqueda se da el mismo caso y

tampoco se encuentra un trabajo que relacione optimización topológica con bombas y computación en la nube.

Por otro lado, realizando una RSL (en el diseño de bombas centrífugas), en otras bases de datos como Science Direct, Research Gate y Springer, se observa que hay un sin número de estudios en el abordaje de las bombas centrífugas como: predicción en el desempeño en el punto diseño y fuera de este [10,12-15], análisis de la cavitación [16-18], rendimiento del funcionamiento de una bomba como turbina [9,19-23], estudio paramétrico en bombas, análisis de la bomba del difusor, manejo de la bomba en fluidos no newtonianos, etc. Estas investigaciones son abordadas por medio de 3 técnicas notoriamente vistas, tal como se muestra en la Figura 1.6 [4,10,11,24, 25,26-38], por medio de códigos in-house, por medio de programas comerciales que resuelven las ecuaciones de Navier-Stokes como por ejemplo, ANSYS, y por medio de experimentación.

El desarrollo de algoritmos (códigos in-house) en la aplicación de optimización topológica en bombas centrífugas, ha sido más orientado a la mejora de la forma del álabe del impulsor para mejorar la eficiencia de la turbomáquina. Estos modelos son utilizados para determinar la eficiencia de un proceso en el que interviene una turbomáquina, particularmente, dependen de un gran número de variables, que describen el comportamiento del fluido (por ejemplo: turbulencia, caída de presión y vorticidad) y, la minimización de estos fenómenos en el desarrollo del algoritmo junto con la solución los campos de velocidades y presiones, vuelven a la solución del problema extensa en estos problemas iterativos usando el MOT, como el trabajo de investigación de David Berrío que no realizó precisamente un estudio en bombas centrífugas pero que sí uso el MOT para un instrumento musical idiófono, lo cual demoró 40 horas con tan solo 13.000 elementos para una simulación en estado transiente, en una Workstation Intel Xeon CPU E5-2687W v2. [39]. Otro ejemplo a lo anterior, es el estudio realizado por Shahram Derakhshan [30], el cual, por medio de un método de optimización global basado en redes neuronales artificiales (ANN) y el algoritmo artificial de colonia de abejas (ABC), logra solucionar las ecuaciones de Navier-Stokes para rediseñar la geometría de un impulsor y mejorar el rendimiento de una bomba marca Berkeh 32-160. El tiempo de cómputo fue de 5.8 horas para el proceso de optimización al usar una PC Intel Core i5, con 2.4 GHz de velocidad y 4 Gb de memoria RAM.

Otra técnica en el estudio de las bombas centrífugas, mencionado anteriormente, es la de programas comerciales que resuelven las ecuaciones de Navier-Stokes como, por ejemplo, ANSYS. La aplicación de CFD en el diseño de bombas de agua y turbinas comenzó hace 30 años. El primer paso se dio con la introducción de elementos finitos en el CFD y se caracterizó por soluciones de Euler cuasi-tridimensionales simplificadas [40]. En la Figura 1.6, podemos observar que el uso de estos programas comerciales en la investigación del flujo en bombas centrífugas ha sido fundamental en el estudio de casos de varias índole, y ha sido aplicado desde bombas centrífugas usadas como turbinas hasta la variación paramétrica del impulsor variando el ángulo de salida del álabe para mejora del desempeño [31]. Es así como la simulación numérica se convierte en una herramienta robusta, permitiendo evaluar diferentes diseños y considerando simultáneamente la influencia de diversas variables[10], dando respuesta a las necesidades de modelación del comportamiento de turbomáquinas en múltiples aplicaciones. [11].

Otras de las técnicas empleadas en la investigación de bombas centrífugas son las experimentales [35-38]. La instalación de banco de pruebas, sensores de presiones, sensores de caudal e instalación y puesta en marcha de los equipos para el estudio de las bombas centrífugas genera mucho tiempo y costo. El estudio con esta técnica también ha abordado diversos temas, desde banco de pruebas de bombas como turbina, hasta su empleo en hemodinámica para reducir la trombosis y la hemólisis en una bomba impulsora [38]. Este último estudio fue realizado en 1990, reduciendo la turbulencia y el estancamiento en un dispositivo ventricular probado en cuatro perros, donde; todos los parámetros hematológicos, medidos cada 2 horas, permanecieron dentro del rango normal.

De todas estas técnicas mencionadas, vemos que cada una de ellas tiene sus ventajas y sus desventajas, tal como se muestra en la Tabla 1.1, es por eso que este proyecto de investigación se justifica con el empleo de la simulación numérica y el desarrollo de un algoritmo usando optimización topológica pero en la nube, con la aplicación de HPC (High Performance Computing), que con el uso de computación paralela nos facilita la aceleración de cálculos y evitamos un alto costo computacional.

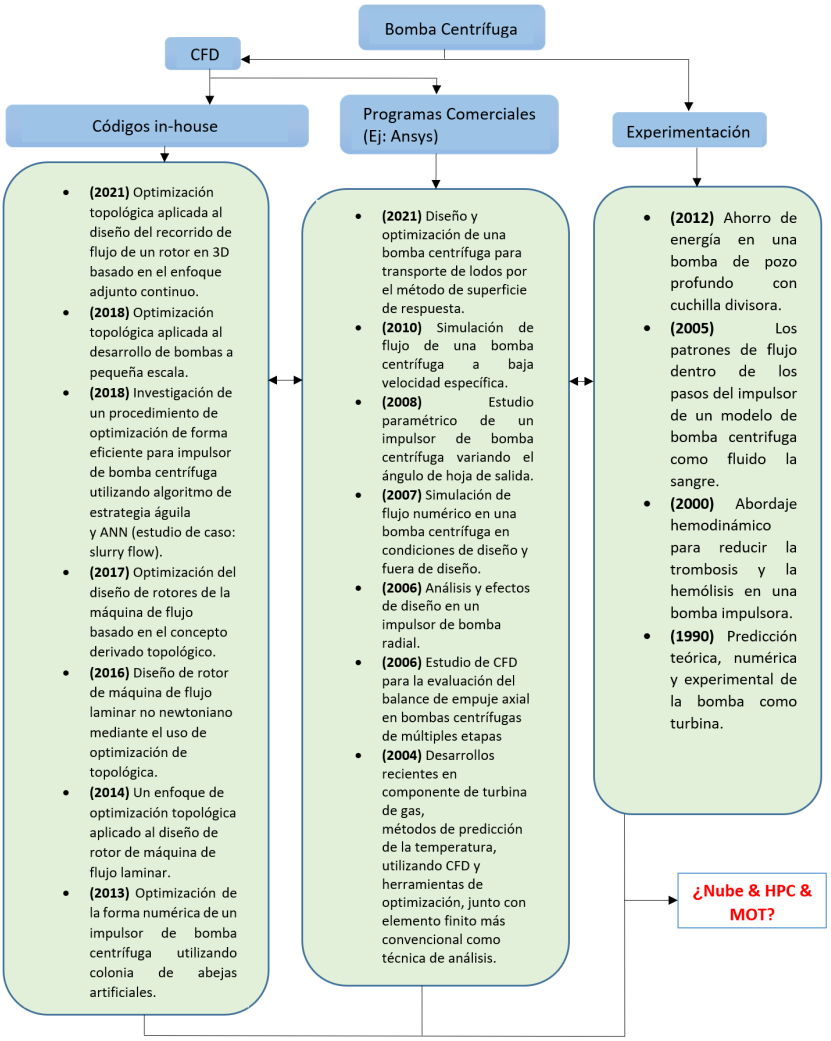


Figura 1.6.- Investigaciones de bombas centrífugas en diferentes abordajes. **a)** Códigos in-house **b)** Programas Comerciales y **c)** Experimentación.

1.2.2.- JUSTIFICACIÓN: APLICABILIDAD

Por otro lado, otra razón que justifica este proyecto de investigación es la aplicabilidad que se le da a las turbomáquinas. Estas máquinas rotativas son usadas en diferentes áreas de la industria como: agricultura, gestión de residuos, en la industria farmacéutica, química, alimentaria, aeroespacial, etc. Una aplicabilidad de importancia en el área medicinal es el uso de bombas centrífugas como dispositivos de asistencia ventricular (VAD). El instituto de Texas Heart [41], constantemente realiza investigaciones cardiovasculares con el uso de dispositivos ventriculares, que son máquinas rotativas a escala reducida que han sido de gran ayuda para personas con insuficiencia cardiaca. Para tener una idea de la necesidad de estos dispositivos, anualmente en Estados Unidos 50.000 personas necesitan de un trasplante de corazón, pero solo 2.500 realmente lo reciben y debido a esta escasez de donantes de corazón, los investigadores han pasado años desarrollando estos dispositivos. Por lo mencionado anteriormente, la necesidad de mejoras de estas máquinas rotativas es de importancia en diferentes áreas de investigación y, en consecuencia, en este trabajo se pretende acelerar la etapa de diseño en computación en la nube, permitiendo crear rotores de bombas con mejor eficiencia y reduciendo costos de pruebas experimentales que generaría grandes inversiones para los empresarios y sistemas de salud.

En conclusión, en la Figura 1.7, se puede observar el número de investigaciones que han venido creciendo de manera exponencial sobre bombas centrífugas considerando optimización topológica. Lo anterior demuestra que es un tema de investigación que ha venido desarrollándose y que todavía falta mucho por hacer en el desempeño y estudio de estas máquinas rotativas.

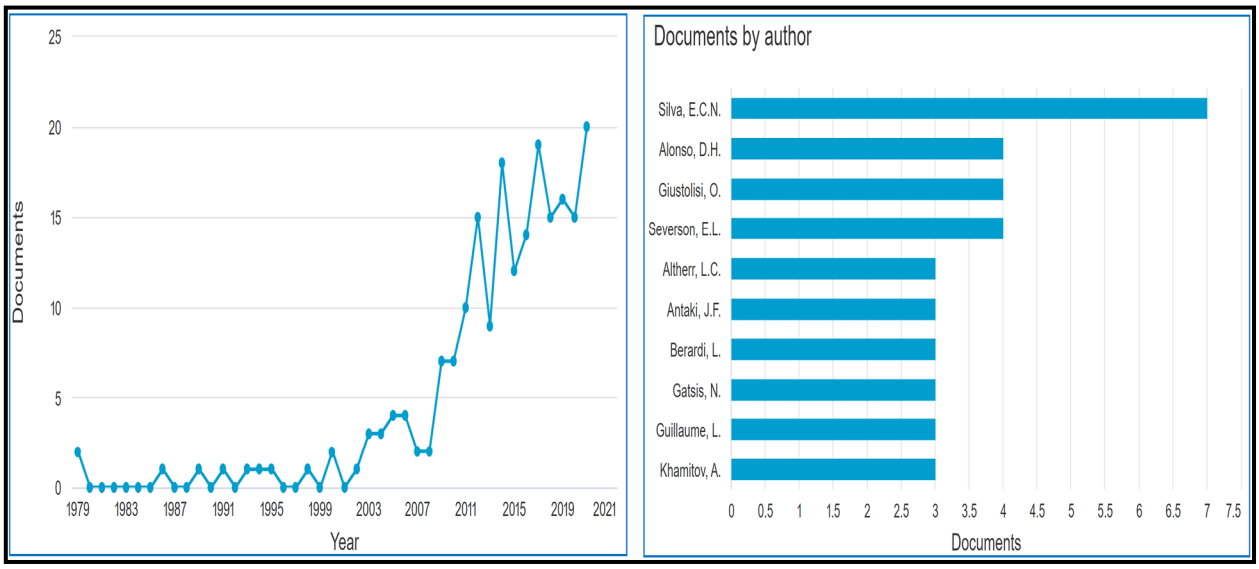


Figura 1.7.- Investigaciones relacionadas con Bombas Centrífugas. Palabras claves: “Pump Topology Optimization” Fuente: www.scopus.com.

Tabla 1.1- Ventajas y desventajas de las técnicas empleadas para los diferentes abordajes en bombas centrífugas.

	Ventajas	**Desventajas**
Código In-house **&** **Programas Comerciales**	• No hay necesidad de compra de equipos. • Implementación de varios casos de estudio en bombas centrífugas, sin la necesidad de inversión de equipos para cada caso (cavitación, bomba como turbina, desempeño en el punto diseño y fuera de este, estudio paramétrico, etc).	• Tiempo y costo computacional en problemas iterativos de MOT. • Se necesita de validación y verificación numérica de los resultados. • Costo de licencias en algunos softwares.
Experimentación	• Control de las variables.	• Alta inversión en compra de equipos. • Limitación a varios casos de estudio.

1.3.- BREVE RESEÑA DEL MOT EN FLUIDOS

La optimización topológica puede determinar la topología, la forma y el tamaño de las estructuras, simultáneamente. Este método ha sido considerado uno de los más poderosos para el diseño inverso de estructuras en varias áreas científicas, que incluyen elasticidad, dinámica de fluidos computacional, termodinámica, electromagnetismo, etc. [42]. En el área de la mecánica de fluidos, esta se fue desarrollando primero con optimización de forma, aplicada a problemas aerodinámicos e hidrodinámicos, como el diseño de las capotas de automóviles, alas de aviones y formas de entrada para motores a reacción. Uno de los primeros estudios se encuentra en las investigaciones realizadas por Pironneau (1974), donde se determina un perfil de arrastre mínimo, sumergido en una capa homogénea, estable y viscosa [43]. Luego de la optimización de forma aplicada a los fluidos, se dio enfoque a la optimización topológica, que fue introducida por Borrvall y Peterson (2003), donde implementa el enfoque de distribución de material para minimizar la energía de disipación en flujos de Stokes [44]. Esta distribución de material aplicada a las ecuaciones de Navier-Stokes se da por medio de la permeabilidad gamma γ como variable de diseño. De esta manera las zonas altamente permeables se interpretan como fluido y las zonas no permeables como sólidos.

En adición, la aplicabilidad del MOT en fluidos se ha venido desarrollando continuamente en muchas aplicaciones, un ejemplo es en el uso de flujos en tuberías (Figura 1.8). En su formulación original, el sólido corresponde a la región con menor permeabilidad (un valor pequeño distinto de cero), pero dicha situación fue llevada a sus condiciones extremas (esto es con permeabilidad igual a cero) por Evgrafov (2005).

Otras de las aplicaciones del MOT a menor escala y de gran importancia en el área de la biotecnología, física o química es el estudio de los micro-fluidos, donde se estudia el comportamiento de los fluidos en microescala y meso-escala. En la literatura se puede encontrar que cuando se trabaja con este tipo de fluidos en canales fluídicos, lo que se busca es la maximización de las fuerzas de cuerpo como función objetivo, es decir buscar la topología adecuada de estos canales que maximicen el trabajo hecho por la fuerza de Coriolis y la fuerza centrífuga [42].

Por último, en este trabajo en particular, se introduce la fuerza de Darcy $\boldsymbol{f}$ (Ecc. 1.2) a las ecuaciones de Navier-Stokes y se usa un modelo de material (α) similar al SIMP (Solid Isotropic Microstructure with Penalization for intermediate densities) para la distribución de material en regiones sólidas y regiones fluídicas.

$$\boldsymbol{f} = -\alpha\ \boldsymbol{u} \qquad \text{(Ecc. 1.2)}$$

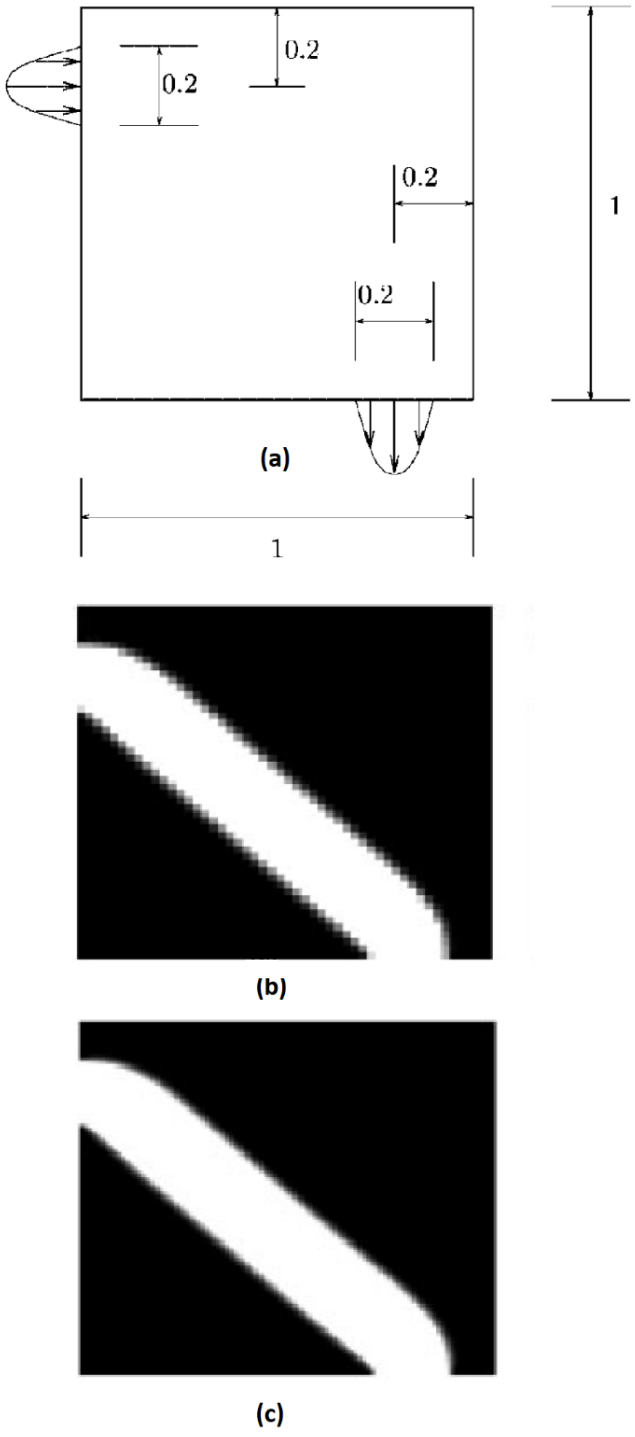


Figura 1.8.- (a) Dominio de diseño para la tubería. **(b)** Tubería óptima en una malla de elementos gruesos. **(c)** Tubería óptima en una malla de elementos finos.

1.4.- ESTADO DEL ARTE DE ELEMENTOS FINITOS APLICANDO COMPUTACIÓN PARALELA EN LA NUBE

En este tema de investigación se pretende hacer uso de computación paralela en la nube, ya que se ha comprobado con otros trabajos de investigación su alto desempeño al resolver problemas con grandes requerimientos de cálculos en simulaciones. Prueba de lo anterior se puede observar en la RSL (Figura 1.5), que existe una gran cantidad de publicaciones en las que se usa computación en la nube para acelerar los cálculos computacionales en aplicaciones científicas [45-48].

También se puede comprobar que, mediante esta RSL, que mayormente la computación en la nube se ha venido implementando para acelerar los cálculos en investigaciones donde se han tenido que resolver rutinas del Método de Elementos Finitos (MEF) [49-55]. Tal es el caso del estudio realizado por Dazhong Wu [50], donde el objetivo de su investigación fue evaluar el rendimiento de la ejecución de una simulación de elementos finitos en una nube pública. Los resultados mostraron que el rendimiento de HPC en la nube es suficiente para la aplicación de un análisis de elementos finitos, y que el tiempo de optimización se puede acelerar, seleccionando correctamente una configuración de CPU, memoria e interconexión. El ejemplo usado en el experimento es el análisis de deformación termo-mecánica de un troquel apilado en 3D.

Otro caso de estudio de elementos finitos en la nube fue el realizado por Haiming Lin [52], donde se propone un algoritmo paralelo de ensamblaje global de la matriz de rigidez del método lineal elástico de elementos finitos y se resuelven las ecuaciones lineales utilizando el método basado en el patrón arquitectónico de MapReduce, luego lo implementa en un cluster hadoop en la nube de 6 nodos. En los resultados se logra una tasa de aceleración de 3 veces el montaje global de la matriz de rigidez. La plataforma hadoop fue utilizada para resolver los elementos finitos en la nube. Un trabajo relevante en la computación mixta en la nube fue el realizado por Jin [46], donde se utilizó el método de elementos finitos y el algoritmo de red neuronal en la nube, para reducir el tiempo de consumo en los cálculos del par axial electromagnético de un motor sincrónico de imán permanente. La utilización de computación en la nube reduce aproximadamente 4 horas el calculó para este caso. La computación mixta se realiza en la nube, entre Windows y Linux; ejecutando MATLAB en Windows y ejecutando COMSOL en Linux en la forma de no GUI (interfaz gráfica de usuario). Los resultados mostraron que la optimización del motor combina la computación en la nube con el algoritmo BP, tiene viabilidad y alta eficiencia.

Como se puede observar en la RSL, existen un sin número de trabajos para el desarrollo de elementos finitos en la nube. En la sección 3.5 se detalla el estado del arte de optimización topológica aplicada a bombas centrífugas, pero en sí no se encuentran investigaciones de esta índole con el uso de computación paralela en la nube.

1.5.- HPC

1.5.1.- DEFINICIÓN

El término de HPC (High Performance Computing) es un término que hay que considerar en este trabajo de investigación, este término se refiere al uso de supercomputadoras (en paralelo) y agrupaciones de computadoras (clúster), es decir, sistemas de computación compuestos de múltiples procesadores (generalmente producidos en masa) unidos entre sí en un solo sistema con interconexiones disponibles comercialmente. Un clúster HPC utiliza una arquitectura de varias computadoras que presenta un sistema de cómputo paralelo que consta de uno o más nodos maestros y uno o más nodos de cómputo interconectados por un sistema de red privado, Figura 1.9 [56].

Hay que recalcar que el término “nodo” que se lo menciona más adelante en la sección 4.5, representa el “nodo de un cluster”; es decir, representa un equipo donde se ejecuta el sistema Operativo Solaris y el Software Sun Cluster. En la Figura 1.9 se puede observar que el nodo maestro tiene interconectado entre sí 8 nodos, como alternativa a una solución de aplicaciones paralelas en la nube.

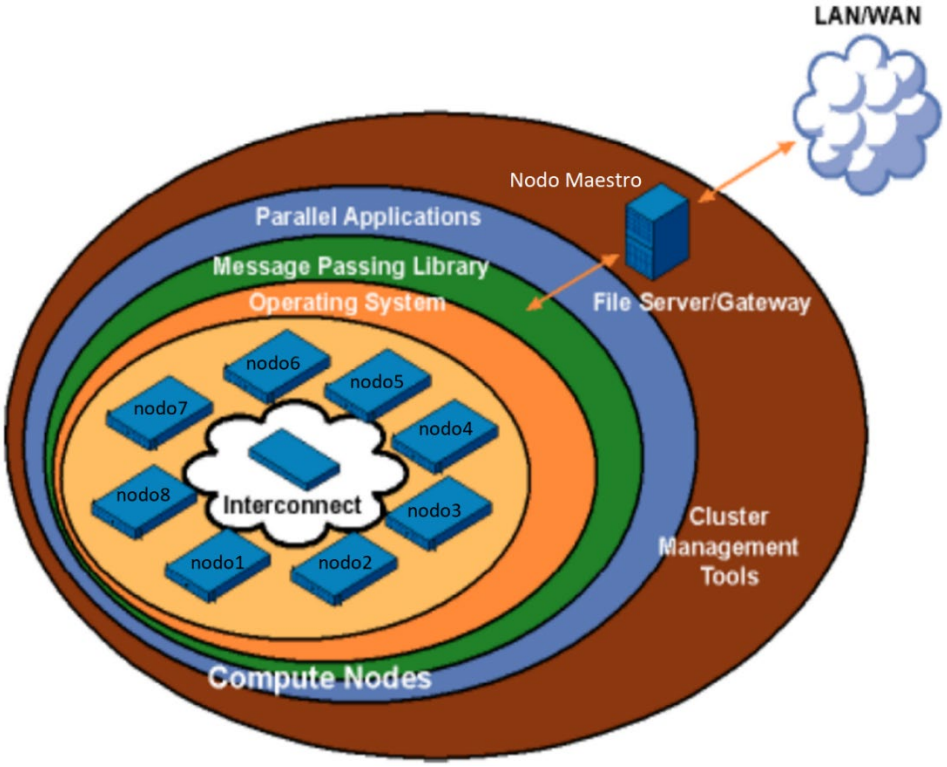


Figura 1.9.- Arquitectura Clúster de varias computadoras paralelas, HPC. Imagen adapta de [56].

1.5.2.- BREVE RESEÑA HPC

Las aplicaciones de computación de alto rendimiento (HPC en inglés) son de varias índoles, que van desde algoritmos aplicados para mejorar la eficiencia del tráfico de carros [57-58], aplicaciones biomédicas [59], aplicaciones en geociencia [60], simulaciones en reactores [61], para CFD [62], aplicaciones en la nube [63], etc.

En aplicaciones de CFD, códigos in-house, existe el desafío de descomponer nuestro algoritmo en un conjunto de tareas que deben ejecutarse: **a)** secuencialmente y en **b)** paralelo, esto se realiza para poder comparar las aceleraciones y rendimiento de un código secuencial vs. un código en paralelo. Estas terminologías son detalladas en el CAPÍTULO 4 sección 4.4 de este documento.

En este trabajo de investigación de optimización topológica en bombas centrífugas aparte de resolver el MOT y el MEF, se debe analizar las líneas del código que demandan mayor tiempo computacional para posteriormente reescribir estas líneas de manera que al momento de ejecutar MATLAB, estas puedan ser ejecutadas de forma paralela con el objetivo de reducir tiempos de ejecución. En la sección 4.7 se analizan las líneas que demandan mayor tiempo computacional en 1 iteración del MOT para una malla de 450 elementos.

1.6.- LA NUBE

1.6.1.- DEFINICIÓN

La nube no es un lugar, sino un método de gestión de recursos de información tecnológica que reemplaza a las máquinas locales y los centros de datos privados con infraestructura virtual. En este modelo, los usuarios acceden a los recursos virtuales de computación, red y almacenamiento que están disponibles en línea a través de un proveedor remoto. Dentro de esta nube virtual podemos identificar tres capas: SaaS, IaaS y PaaS.

SaaS, la capa de "Software como servicio", él proveedor del servicio pone a disposición de los clientes su propio software, desligándolos de tener que mantenerlos actualizados o comprar licencias, ejemplo de esta capa son Salesforce y Basecamp. IaaS, infraestructura como servicio, podría decirse que es la parte física de la nube. En vez de tener el equipamiento en su propio lugar de trabajo, los clientes pagan a un proveedor para que éste sea quien tenga todo ese equipamiento (llámese discos duros o equipamiento de redes) y se encargue de toda la mantención y optimización de dicho equipamiento. Empresas que ofrecen servicios más enfocado a esta capa son Amazon Web Services EC2 y Google Cloud Compute Engine (instancias de vm). Y por último se tiene la última capa conocida como PaaS, plataforma como servicio, que está muy ligada a la capa SaaS, ya que es la plataforma donde se envuelve el software que pone a disposición el proveedor y es el medio de virtualización para el hardware que el cliente arrienda [64]. Ejemplo de esta capa es Google App Engine.

1.6.2.- CFD, COMPUTACIÓN EN LA NUBE RSL

Las aplicaciones de computación en la nube son varias, desde simulación de eventos discretos [65], programación de tareas en computación en la nube [66], computación en la nube vehícular [67], hasta simulación en CFD en la nube basado en OpenFoam para el diseño de intercambiadores de calor [68]. Por mencionar algunos de los trabajos realizados en la nube destacaremos el de intercambiadores de calor realizado por Antonio Gómez et al. [68]. Ellos presentan una herramienta CFD, implementada en OpenFoam y ejecutada en un entorno Cloud que posteriormente fue integrada en una aplicación en la nube para simular el rendimiento de los intercambiadores de calor de carcasa y tubos que puedan permitir a las pequeñas y medianas empresas (PYMES) utilizar estos métodos de cálculo.

Para detallar el trabajo realizado para el diseño de los intercambiadores, en la Figura 1.10 se muestra un flujo de trabajo de la herramienta CFD. Esta herramienta ejecutada en la nube da la facilidad al usuario a que solo proporcione la geometría del intercambiador de calor, a través de un archivo STL generado por el mismo usuario que emplea su propio software de CAD, y el funcionamiento de los parámetros (caudales másicos de los gases de combustión y agua, temperatura de entrada, contenido de vapor en el gas de combustión y resistencia de las incrustaciones). Además, esta aplicación CFD genera automáticamente la malla computacional (usando SnappyHexMesh, un generador de mallas), resuelve las ecuaciones de conservación (implementadas en OpenFoam) y post-procesa los resultados en paraview, para proporcionar un informe con los parámetros más relevantes de interés para el diseñador.

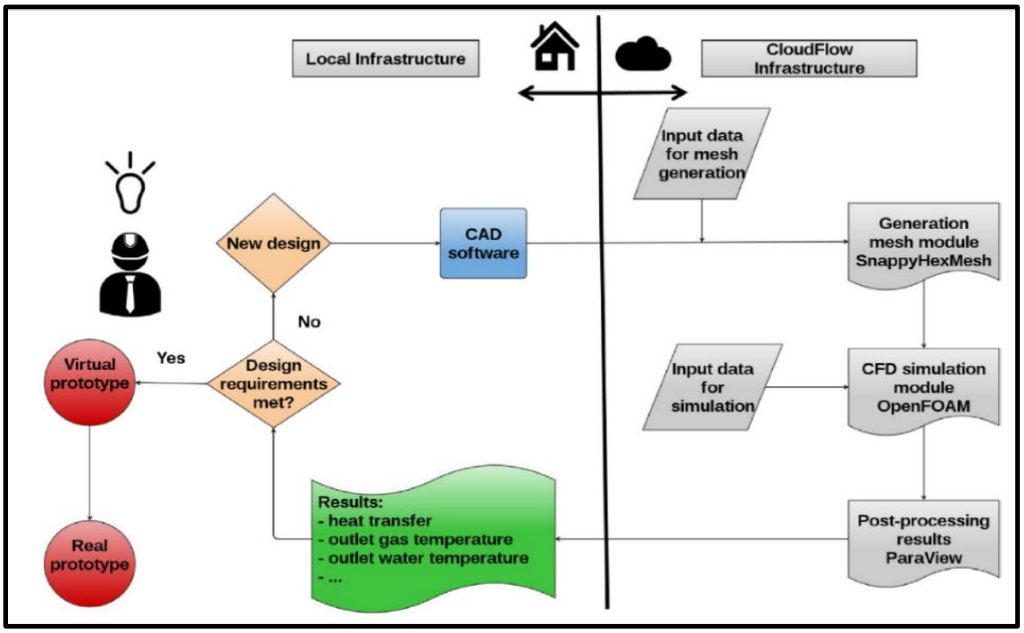


Figura 1.10.- Flujo de trabajo de la herramienta CFD en la nube [68].

Adicionalmente al desarrollo de esta herramienta, se realiza validaciones de esta con un modelo experimental donde se comparan los datos medidos y calculados para las pruebas de validación. Las pruebas consisten en medir la temperatura de la salida de los gases de combustión y la temperatura de salida del agua a diferentes cargas del intercambiador de calor para garantizar que el modelo desarrollado en OpenFoam este bien simulado. Una vez validado el modelo, se

estudian nuevos diseños (pequeños cambios al intercambiador de calor actual) obteniendo como resultado una reducción de volumen del 30% manteniendo su rendimiento térmico, dando beneficio a la reducción de costos debido a la reducción de material para futuras fabricaciones.

Otro trabajo destacable en la nube es el propuesto por Simon Taylor et.al [69] donde desarrolla una plataforma de simulación CFD descrita como CloudSME que tiene como objetivo proporcionar valga la redundancia una plataforma-como-servicio (Paas) basada en una nube flexible y fácil de usar, permitiendo así a las pymes obtener beneficios de computación de alto rendimiento (HPC). Esta plataforma desarrollada en la nube con simulación de alto rendimiento permite aplicaciones paralelizadas ejecutadas en múltiples CPUs, acopladas a una única instancia de nube (servidor virtual en la nube) y también permite ejecutar múltiples simulaciones en paralelo en una combinación de múltiples instancias en una única nube. Esta plataforma consta de un software en la nube como solucionador de CFD denominado TransAT. TransAT implementa la ejecución de cálculo de alto rendimiento mediante una paralelización híbrida Open-MPI [70]. Desde la interfaz de TransAT, un usuario puede especificar cuantas CPU de un clúster de cálculo utilizará en la nube.

Para mostrar el buen rendimiento de la plataforma como solución a casos de CFD, en este trabajo se presentan tres casos de estudios que muestran el uso y rendimiento de TransAT basado en la nube. Para detallar uno de estos casos, se menciona el estudio realizado a un separador trifásico horizontal basado en gravedad, dispositivo que es familiar para aquellos profesionales en la industria de petróleo y gas, ver Figura 1.11. Este dispositivo simulado tiene una entrada para el producto de campo y tres salidas para el petróleo, agua y producto de gases separados, que, dado un tiempo de permanencia suficiente, permite la separación de estos tres productos con la ayuda de la gravedad.

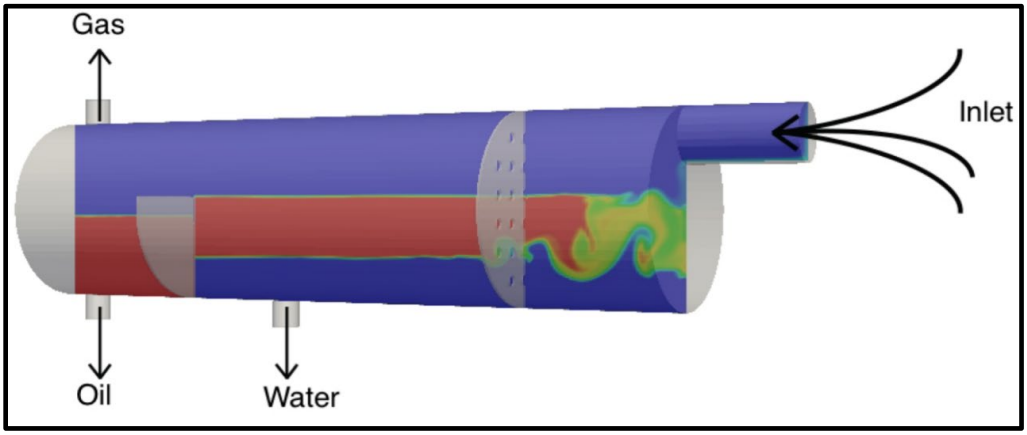


Figura 1.11.- Visualización en TransAT de un separador trifásico por gravedad horizontal [69].

Esta simulación con una malla de 300.000 celdas es comparada en términos de tiempos computacionales entre un clúster local ASCOMP y con recursos de la nube (AWS) usando la plataforma de simulación CloudSME. Las corridas se realizaron usando hasta 32 procesadores en el clúster local (procesadores Intel Xeon E5-649 con velocidad en los núcleos de 2.53 GHz con 32 Gb de memoria RAM). En la nube, para experimentos hasta 8 CPU, se utilizaron instancias en Amazon Web Services de propósito general a través de la plataforma de simulación

CloudSME (instancias tipo large, con CPU Intel Xeon E5-2670 con velocidad en los núcleos de 2.5 GHz y con 7.5 GB de memoria RAM), xlarge (4 CPU con 15 GB de memoria RAM) y 2xlarge (8 CPU con 30 GB de memoria RAM). En la Tabla 1.2 se muestran los tiempos de ejecución en segundos del modelo CFD del separador por gravedad ejecutando 3000 iteraciones para diferentes números de procesadores.

Tabla 1.2.- Duración en segundos (3000 iteraciones) [69].

Número de procesadores	2	4	8	16	32
Clúster local [seg]	412,319	153,687	121,791	78,805	38,667
Amazon EC2 [seg]	678,000	358,500	198,000	99,000	57,000

Los resultados de la Tabla 1.2 se visualizan de mejor manera en la Figura 1.12, donde se muestran los tiempos de ejecución del clúster local frente a las instancias de Amazon EC2. Al comparar los tiempos entre 2 y 8 CPU, el clúster mejora por un factor de 3.39 y la nube (AWS) por un factor de 3.42 en cuestión de hardware. Se puede observar en la Figura 1.12 que la simulación funcionó más lenta en las instancias de la nube que el clúster local por un factor de 0.60, 0.42, 0.60, 0.80 y 0.68 para los núcleos de 2,4,8,16 y 32 de la CPU respectivamente. Esto se debe principalmente a la velocidad de comunicación es mucho más rápida en el clúster local que en las instancias de la nube. Esto se debe principalmente a que la comunicación entre servidor y núcleos en una instancia en la nube es por medio de un hipervisor, a diferencia de un clúster local donde la comunicación es directa.

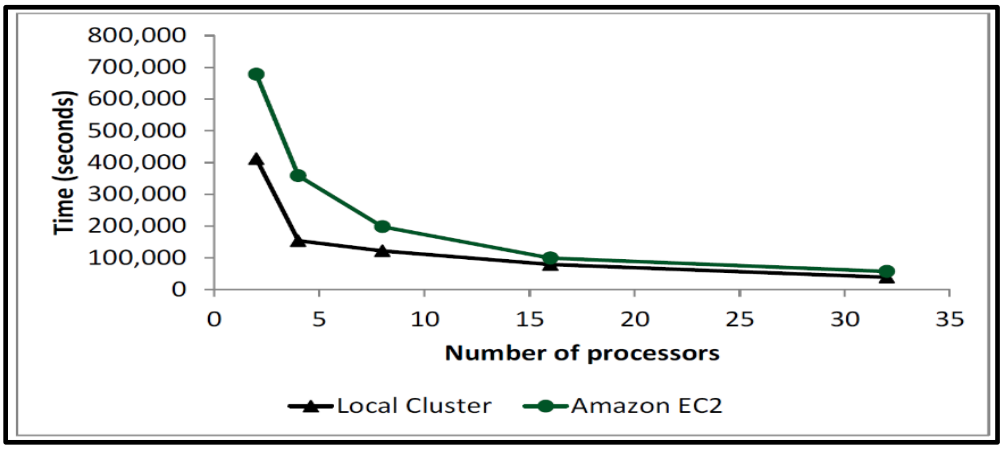


Figura 1.12.- Tiempo de ejecución de la simulación del separador de gravedad (3.000 iteraciones) [69].

En general, a pesar del hecho de que la plataforma basada en la nube parece ser más lento que el clúster local, el rendimiento de la nube es realmente muy bueno y más aún cuando el objetivo principal de este trabajo es el de ofrecer la posibilidad de que las PYMES reproduzcan el mismo alto rendimiento de computación (HPC) esperado al de un clúster local (caro de comprar y mantener) en una plataforma basada en la nube. Adicionalmente, para fortalecer aún más el buen desempeño de las instancias en AWS y su bajo costo, en este trabajo se muestran también el costo por hora de estas instancias en Amazon EC2, ver Tabla 1.3.

Tabla 1.3.- Costo de las instancias por hora en Amazon Web Services [69].

	Número de CPU	**Costo por hora ($)**	**Tiempo de ejecución [seg]**	**Costo Total ($)**
Amazon large	2	0.14	678,000	26.37
Amazon xlarge	4	0.28	358,500	27.88
Amazon 2xlarge	8	0.56	198,000	30.80
Amazon 4x large	16	0.84	99,000	23.10
Amazon 8xlarge	32	1.68	57,000	26.60

Estos costos totales de los tiempos de ejecución en paralelo de las instancias en Amazon que ilustra el autor en su documento de investigación [69], son mostrados para destacar el hecho de que la computación en la nube es una solución muy atractiva partiendo del hecho de que se necesitaría de la compra de un supercomputador para poder realizar simulaciones de esta índole.

En este trabajo el autor menciona que el costo del clúster utilizado en la experimentación (rack de 64 procesadores con interconexión de alta velocidad) es aproximadamente de 24.000 $ dólares americanos (sin incluir los costos de mantenimiento y el considerable costo de consumo de energía y refrigeración).

1.7.- OBJETIVOS

1.7.1.- OBJETIVO GENERAL

Formular una metodología de diseño conceptual de la configuración de la forma y ubicación de los álabes en una bomba centrífuga para mejorar la eficiencia de la bomba mediante la técnica de optimización topológica y considerando procesamiento en la nube.

1.7.2.- OBJETIVOS ESPECÍFICOS

1. Simular mediante el uso de un software comercial el comportamiento hidrodinámico de flujo Newtoniano en bombas centrífugas, resolviendo las ecuaciones de Navier-Stokes (ver Figura 5.1 y Figura 5.2).
2. Desarrollar un código para el modelado de bombas centrífugas usando el método de los elementos finitos, que permita simular los contornos de velocidad y presión (ver sección 2.4.3, formulación MEF).
3. Implementar un algoritmo de optimización topológica que permita diseñar la configuración óptima para el diseño de alabes considerando computación paralela en la nube (ver sección 3.6, formulación MOT).
4. Verificar numéricamente el código desarrollado de elementos finitos que describe el comportamiento hidrodinámico de la bomba centrífuga contra un software comercial de dinámica de fluidos computacional (ver Figura 5.1 y Figura 5.2).
5. Comparar exactitud de resultados, tiempos y costos computacionales usados en la resolución de resultados, usando Computación en la nube usando máquina virtual vs. Computación en la nube usando máquina baremetal (ver sección 5.3 y 5.4).

1.8.- ORGANIZACIÓN DE LA TESIS

Esta tesis está organizada de la siguiente manera: El **CAPÍTULO 2** detalla el método de los elementos finitos aplicado al modelo de máquina de flujo radial. El **CAPÍTULO 3** trata los temas relacionados a la implementación del MOT sobre el modelo de máquina de flujo radial. En el **CAPÍTULO 4** se presentan los temas relacionados a la computación paralela en la nube y se detalla los pseudocódigos paralelizados en este trabajo de investigación. En el **CAPÍTULO 5** se presentan los resultados de las topologías obtenidas tanto para la minimización de la energía de disipación y la vorticidad; y, se presentan la comparación de tiempos computacionales de la ejecución del algoritmo MOT entre la máquina virtual vs. máquina baremetal, en serial y en paralelo. En el **CAPÍTULO 6** se presentan las conclusiones y los trabajos futuros. Finalmente se presenta el **ANEXO A** donde se deja expresado en forma explícita los coeficientes de matriz $\boldsymbol{k}^E$ por elemento, se detalla la función de interpolación y se describe la técnica del elemento de referencia para un elemento rectangular de cuatro nodos, se detalla la formulación isoparamétrica del elemento finito con el objetivo de obtener un jacobiano de transformación.

CAPÍTULO 2: MODELO DE MÁQUINA DE FLUJO RADIAL

2.1 TURBOMÁQUINAS: ESQUEMA Y DESCRIPCIÓN DEL MODELO DEL FLUJO RADIAL EN BOMBAS CENTRÍFUGAS

Las turbomáquinas como las bombas centrífugas, también conocidas como bombas rotodinámicas, se componen de elementos para bombear líquidos en general que transforman la energía mecánica de un impulsor en energía de presión de un fluido incomprensible.

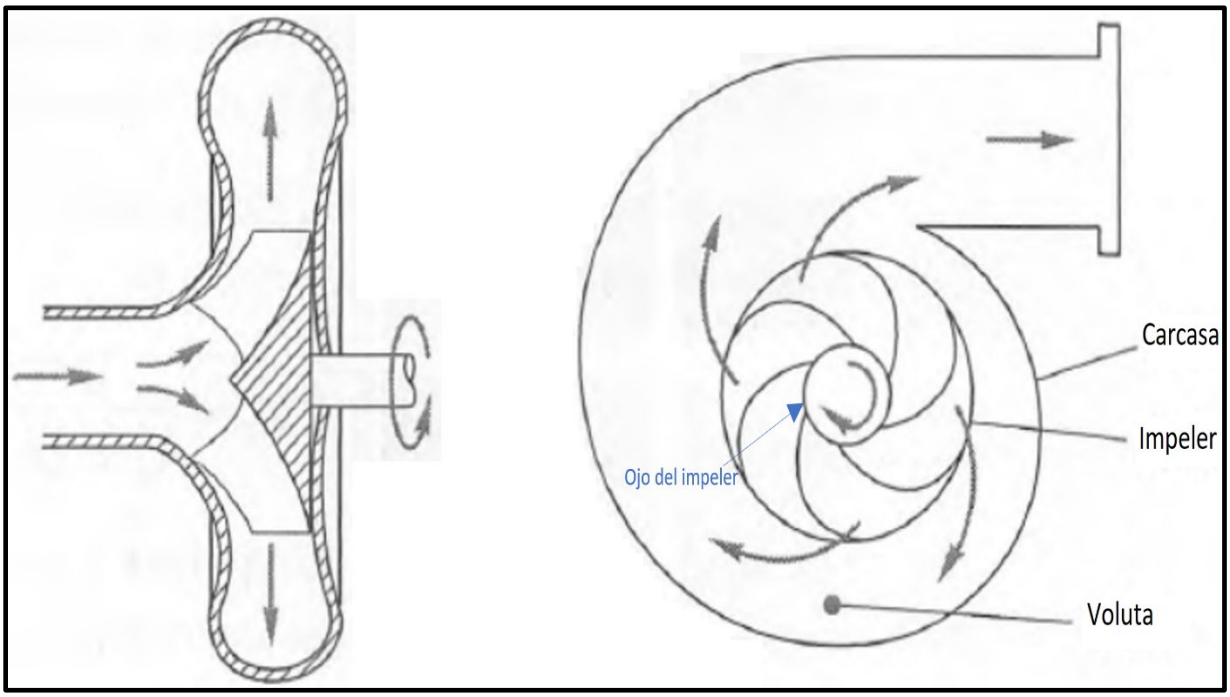


Figura 2.1.- Esquema de una bomba radial típica [5].

Estas bombas centrífugas están constituidas por una parte móvil, conformada por el impulsor y una parte estacionaria que es la voluta o carcasa. El rotor y la carcasa se muestran en la Figura 2.1. Este rotor comprende una serie de álabes, dispuestos de manera radial. El principio de funcionamiento de estas máquinas con el fluido es que este entra axialmente a través del impeler, en el eje de la carcasa, los álabes del rotor lo fuerzan a tomar un movimiento tangencial y radial hacia el exterior del rotor, donde es recogido por una carcasa que hace de difusor. El fluido aumenta su velocidad y presión cuando pasa a través del rotor. La parte de la carcasa, de forma toroidal, o voluta, desacelera el flujo y aumenta más la presión [5]. El incremento de la presión se debe a las fuerzas centrífugas y a la disminución de la velocidad relativa del fluido en su paso por el impulsor.

Adicional a lo anterior, existen otras partes mecánicas que no se muestran en esta figura pero que son elementos que cabe mencionarlos y que son pertenecientes a estas máquinas rotativas, como es el caso de cojinetes, soporte de los cojinetes, sellos mecánicos, tuercas de sujeción, etc.

Para describir de una mejor manera el modelo de flujo simplificado que se usa en este trabajo de investigación en el flujo de las máquinas radiales, a manera de ejemplo se muestra la Figura 2.2, donde se observa que se puede modelar al flujo de fluido de una bomba centrífuga en un canal recto, que gira alrededor de un eje de marco fijo (marco inercial).

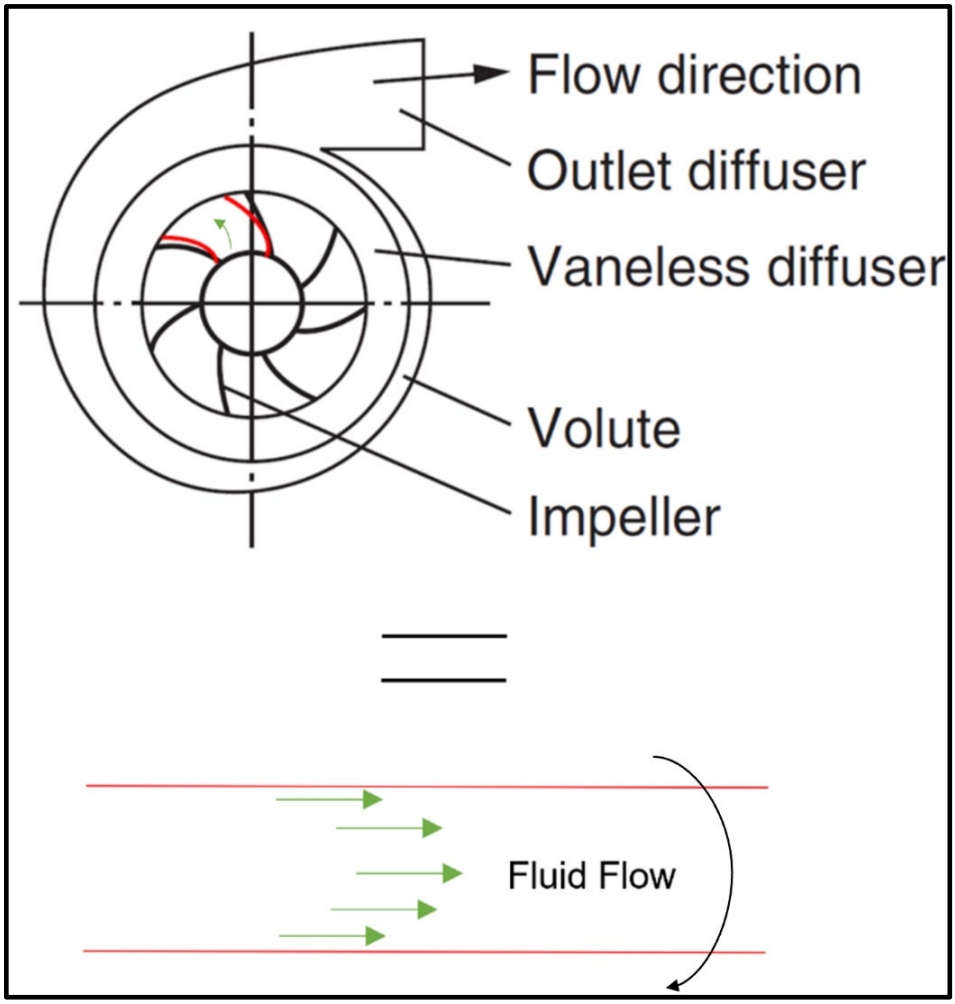


Figura 2.2.- Descripción del modelo de flujo radial aproximado al modelo de rotor en bombas centrífugas (imagen adaptada [71]).

Por lo tanto, las ecuaciones de Navier-Stokes se utilizan con la adición de fuerzas de cuerpo que representan los términos de rotación impuestos por el rotor. Estos términos son relativos a la inercia del fluido. Es así, que un fluido rotativo en movimiento da lugar a la aparición de dos fuerzas de cuerpo (fuerza de Coriolis y fuerza centrífuga) en las ecuaciones de gobierno.

El sistema de ecuaciones diferenciales parciales y su solución mediante los elementos finitos MEF a las ecuaciones de Navier-Stokes donde se introducen estas fuerzas de cuerpo son detalladas en la sección 2.2 y sección 2.3 respectivamente.

2.2 ECUACIONES DE GOBIERNO

2.2.1 ECUACIONES DE NAVIER-STOKES PARA FLUIDOS INCOMPRESIBLES

El flujo que se considera en este trabajo es de un fluido newtoniano, incompresible y estable para bajos números de Reynolds, es decir, se trabaja con flujo laminar. La ecuación de movimiento para un flujo newtoniano son las ecuaciones de Navier-Stokes y en su notación vectorial se expresa de la siguiente manera [5]:

$$\rho \frac{D\boldsymbol{u}}{Dt} = -\nabla p + \mu \nabla \cdot (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) + \rho \boldsymbol{f} \qquad \text{(Ecc. 2.1)}$$

Donde ρ es la densidad del fluido, p es la presión estática, μ es la viscosidad dinámica del fluido, $\boldsymbol{u}$ es la velocidad absoluta del fluido y $\boldsymbol{f}$ representa a las fuerzas de cuerpo.

La ecuación de continuidad se expresa de la siguiente manera:

$$\nabla \cdot \boldsymbol{u} = 0 \qquad \text{(Ecc. 2.2)}$$

2.2.2 SISTEMA DE REFERENCIA ROTATIVO

Para el modelado de las ecuaciones de Navier-Stokes aplicado al problema del modelado de flujo radial, se debe lidiar con sistemas en rotación para formular la ecuación de momento y la ecuación de continuidad para flujos rotativos.

En este trabajo se considera que el sistema está en movimiento rotacional estable con velocidad angular constante alrededor de un eje que coincide con el eje de la coordenada x_3 (eje perpendicular al plano x_1- x_2) del marco fijo como se muestra en la Figura 2.3.

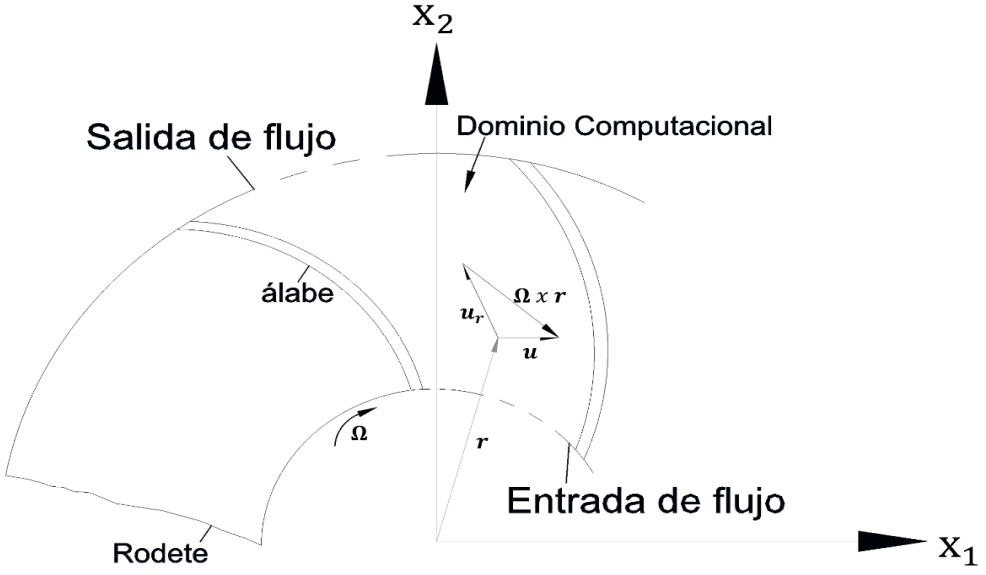


Figura 2.3.- Perfil de geometría de alabe del impulsor radial. $\boldsymbol{u}$ es la velocidad absoluta del fluido, $\boldsymbol{u_r}$ es la velocidad relativa y $\boldsymbol{\Omega}\ x\ \boldsymbol{r}$ es la velocidad de arrastre o tangencial debido a la rotación del rodete.

Si analizamos en esta figura, una partícula de fluido que entra o sale del rodete, esta vendrá definida por un vector posición $\boldsymbol{r}$ y tendrá una determinada velocidad absoluta como $\boldsymbol{u}$. Esta velocidad absoluta está formada por una componente tangencial $\boldsymbol{u_t}$ debida a la rotación del rodete y una velocidad relativa $\boldsymbol{u_r}$. Si analizamos estos tres vectores obtenemos los triángulos de velocidad tanto en la entrada como en la salida del rodete y esta expresión vectorial está dada por la (Ecc. 2.3) como [72]:

$$\boldsymbol{u} = \boldsymbol{u_r} + \boldsymbol{u_t} = \boldsymbol{u_r} + \boldsymbol{\Omega}\, x\, \boldsymbol{r} \qquad \text{(Ecc. 2.3)}$$

Entonces teniendo un modelo de fluido en rotación con respecto a un marco inercial de referencia, se puede reescribir la ecuación de movimiento en términos de velocidades relativas e introduciendo las dos fuerzas de cuerpo (centrífuga y Coriolis) de la siguiente manera:

$$\rho\nabla\, \boldsymbol{u_r}\cdot\boldsymbol{u_r} = -\nabla p + \mu\nabla\cdot(\nabla\boldsymbol{u_r} + \nabla\boldsymbol{u_r}^T) + \rho\boldsymbol{f} - 2\rho\boldsymbol{\Omega}\, x\, \boldsymbol{u_r} - \rho\boldsymbol{\Omega}\, x\, \boldsymbol{\Omega}\, x\, \boldsymbol{r} + \boldsymbol{f}_r \quad \text{(Ecc. 2.4)}$$

$$\boldsymbol{f}_r = -\alpha\left(\gamma_E\right)\boldsymbol{u_r} \qquad \text{(Ecc. 2.5)}$$

Donde el término $\rho\nabla\, \boldsymbol{u_r}\cdot\boldsymbol{u_r}$ representa la aceleración convectiva, $-\nabla p$ representa el gradiente de presión, $\mu\nabla\cdot(\nabla\boldsymbol{u_r} + \nabla\boldsymbol{u_r}^T)$ representa el término difusivo, $\boldsymbol{f}$ representa la fuerza de cuerpo como la gravedad, $-2\rho\boldsymbol{\Omega}\, x\, \boldsymbol{u_r}$ representa a la fuerza de Coriolis, $-\rho\boldsymbol{\Omega}\, x\, \boldsymbol{\Omega}\, x\, \boldsymbol{r}$ representa la fuerza centrífuga y $\boldsymbol{f}_r$ representa la fuerza debido a la fricción entre el fluido y la matriz solida fija, α y γ_E representan el coeficiente de absorción o campo de porosidad y la permeabilidad del fluido por elemento, respectivamente [4,26]. Podemos observar en la (Ecc. 2.4) que el término transiente queda eliminado de la ecuación porque estamos trabajando en estado permanente debido a que la velocidad de entrada y la velocidad de rotación son constantes.

Escribiendo también la ecuación de continuidad en términos de velocidades relativas se tiene:

$$\nabla\cdot\boldsymbol{u_r} = 0 \qquad \text{(Ecc. 2.6)}$$

2.3 METODO DE ELEMENTOS FINITOS APLICADO A MÁQUINAS DE FLUJO RADIAL

2.3.1 INTRODUCCIÓN A LOS ELEMENTOS FINITOS

Las ecuaciones de Navier-Stokes son un conjunto de ecuaciones diferenciales parciales no lineales que describen el flujo de fluidos, que representa la conservación de momentum lineal tal como se describió en la (Ecc. 2.1).

Estas ecuaciones representan la piedra angular de la mecánica de fluidos como señala Cengel et al. 2010 [73] y se resuelven conjuntamente con la ecuación de continuidad. Estas soluciones de ecuaciones se resuelven en conjunto por medio de métodos numéricos (volúmenes finitos, diferencias finitas, elementos finitos, etc.), ya que no pueden resolverse de manera exacta (no tiene una solución analítica).

El método de elementos finitos es el método que se elige para resolver las ecuaciones de Navier-Stokes y de continuidad en este proyecto de investigación, ya que además de ser una técnica

poderosa para la dinámica de fluidos computacionales que se aplica fácilmente a dominios de forma geométricas complejas, es una técnica que va muy de la mano con el método de optimización topológica para fluidos. Este método de los elementos finitos reduce el sistema de ecuaciones diferenciales parciales a un sistema de ecuaciones algebraicas que se pueden resolver con técnicas tradicionales de algebra lineal.

En el método de los elementos finitos, el dominio de interés se subdivide en pequeños subdominios llamados elementos finitos. Sobre cada elemento finito, las variables desconocidas (presión y velocidades) se aproximan mediante funciones de aproximación llamadas funciones de forma que están asociadas con el nodo del elemento que caracteriza al elemento. Estas aproximaciones para cada elemento finito del dominio de diseño se ensamblan juntas para obtener un sistema global y posteriormente resolver el sistema. En la siguiente sección 2.3.2, se detalla la implementación del método de elementos finitos aplicado a máquinas de flujo radial.

2.3.2 IMPLEMENTACIÓN DEL MÉTODO DE LOS ELEMENTO FINITOS A MÁQUINAS DE FLUJO RADIAL

Antes de implementar el método de los elementos finitos, es necesario escribir las ecuaciones de Navier-Stokes (Ecc. 2.4) y de continuidad (Ecc. 2.6), en su forma diferencial y en dos dimensiones:

Ecuación de Navier-Stokes en x_1:

$$\rho\left(u_{r1}\frac{\partial u_{r1}}{\partial \mathrm{x}_1}+u_{r2}\frac{\partial u_{r1}}{\partial \mathrm{x}_2}\right)=-\frac{\partial p}{\partial \mathrm{x}_1}+\mu\left(\frac{\partial^2 u_{r1}}{\partial^2 \mathrm{x}_1}+\frac{\partial^2 u_{r1}}{\partial^2 \mathrm{x}_2}\right)-2\rho\Omega u_{r2}-\Omega^2 r_{\mathrm{x}1}-\alpha\, u_{r1} \qquad (\text{Ecc. } 2.7)$$

Ecuación de Navier-Stokes en x_2:

$$\rho\left(u_{r1}\frac{\partial u_{r2}}{\partial \mathrm{x}_1}+u_{r2}\frac{\partial u_{r2}}{\partial \mathrm{x}_2}\right)=-\frac{\partial p}{\partial \mathrm{x}_2}+\mu\left(\frac{\partial^2 u_{r2}}{\partial^2 \mathrm{x}_1}+\frac{\partial^2 u_{r2}}{\partial^2 \mathrm{x}_2}\right)+\rho g_v-2\rho\Omega u_{r2}-\Omega^2 r_{\mathrm{x}2}-\alpha\, u_{r2} \quad (\text{Ecc. } 2.8)$$

Ecuación de Continuidad:

$$\frac{\partial u_{r1}}{\partial \mathrm{x}_1}+\frac{\partial u_{r2}}{\partial \mathrm{x}_2} \quad (\text{Ecc. } 2.9)$$

Donde u_{r1} y u_{r2} son las componentes de velocidad relativa; y, los términos $r_{\mathrm{x}1}$ y $r_{\mathrm{x}2}$ representan los radios del elemento en los ejes x_1 y x_2 respectivamente. El término g_v representa la gravedad, que es considerada en la ecuación de Navier-Stokes en x_2.

Una vez detalladas las ecuaciones de gobierno en su forma diferencial y en dos dimensiones, la estructura de la programación de elementos finitos aplicado a máquinas de flujo radial de una solución de estado estable de las ecuaciones de Navier-Stokes se describe a continuación.

Se considera el caso del alabe recto con un radio inicial de r_i = 0.4 [m] y un radio final r_f = 1, tal como se muestra en la Figura 2.4.

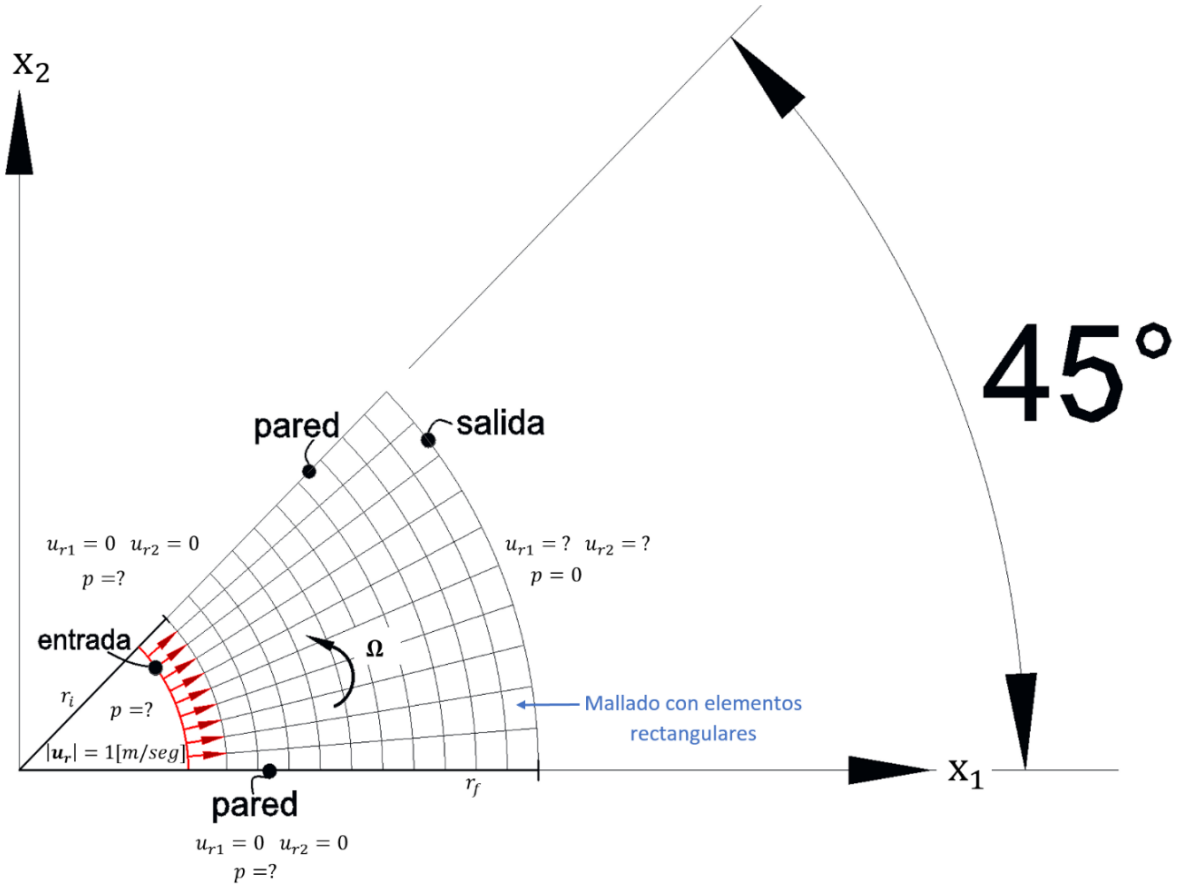


Figura 2.4.- Condiciones de frontera Alabe recto y geometría.

Trabajaremos con un flujo newtoniano que ingresa por la condición de frontera de entrada con una velocidad uniforme de una magnitud de 1 [m/seg]. Las condiciones de frontera restantes son las paredes (separadas entre si 45°) y la condición de salida con presión manométrica de 0 [pa].

Para modelar el flujo se seleccionan elementos rectangulares. Se considera un elemento rectangular por elemento de 8 nodos para modelar las componentes de la velocidad u_{r1} y u_{r2}, y un elemento rectangular de 4 nodos en las esquinas para modelar la presión p. Esta selección de elementos finitos en particular (Q_8 para la velocidad y Q_4 para la presión) se justifica debido a que las variables de estado que uno quiere resolver en las ecuaciones de Navier-Stokes poseen términos diferenciales de diferente orden. Debido a esto se seleccionan elementos mixtos conocidos como elementos de Taylor-Hood.

Entonces, si denotamos un elemento de la Figura 2.4 por E, los grados de libertad de la velocidad y presión con sus nodos respectivos pueden ser visualizados de mejor manera en la Figura 2.5. Las funciones de forma para los elementos rectangulares se expresan en términos de coordenadas locales ξ y η donde [74]:

$$\xi = \frac{2(\mathrm{x_1} - \mathrm{x_{1c}})}{l_{\mathrm{x1}}} \qquad \text{(Ecc. 2.10)}$$

$$\eta = \frac{2(x_2 - x_{2c})}{l_{x2}} \qquad \text{(Ecc. 2.11)}$$

Donde (x_{1c}, x_{2c}) es el centroide del elemento, y l_{x1}, l_{x2} representa su longitud en la dirección x_1 y x_2. respectivamente.

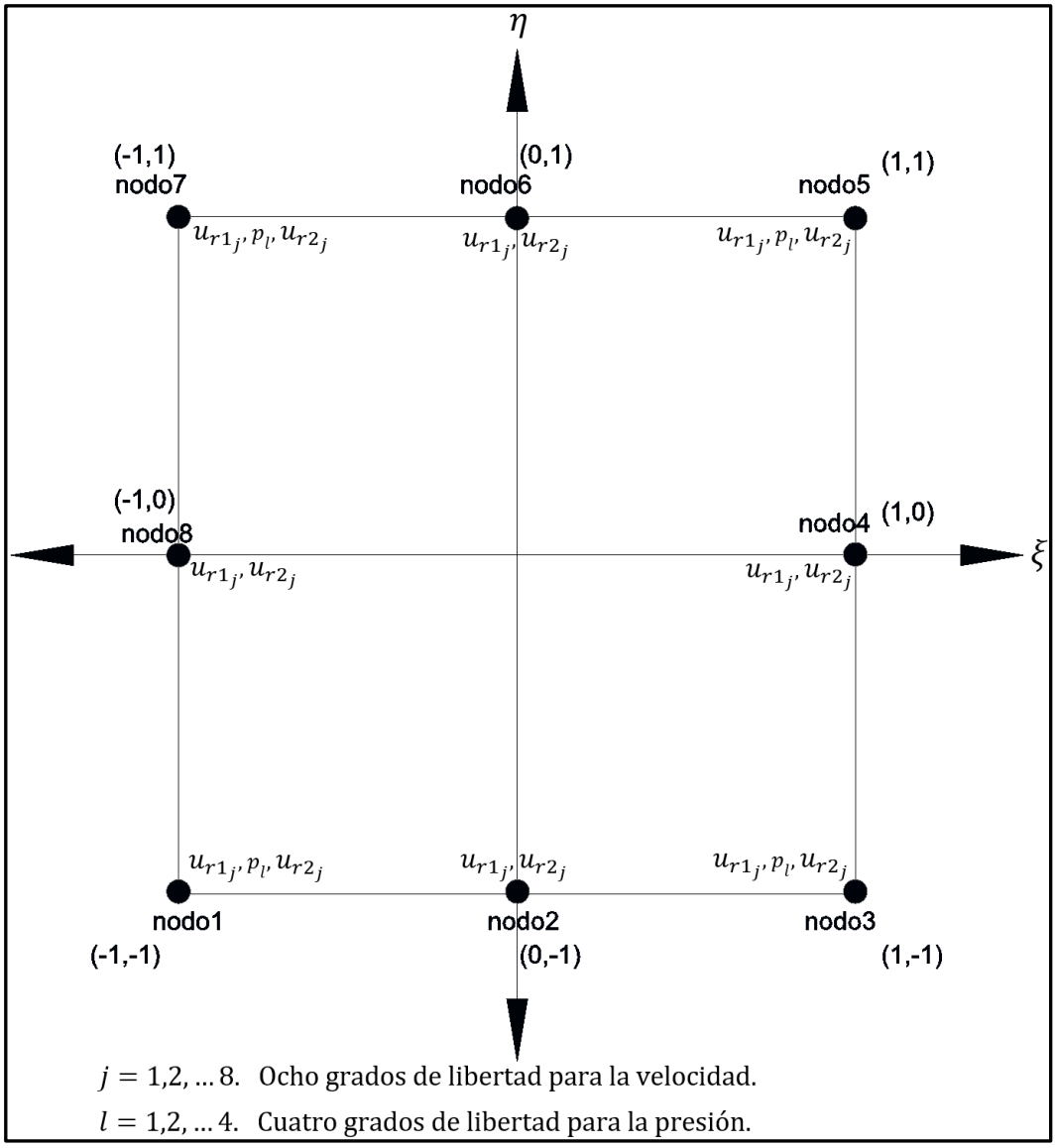


Figura 2.5.- Elemento rectangular de 8 y 4 nodos con grados de libertad locales para la velocidad y presión respectivamente.

Suponga que los nodos 1, 2, 3, 4, 5, 6, 7, 8 tienen coordenadas (-1,1), (0,-1), (1,-1), (1,0), (1,1), (0,1), (-1,1), (-1,0) en el sistema de coordenadas locales. Entonces la forma general de las funciones de forma para la presión usando el elemento rectangular bilineal de 4 nodos (considerando los nodos en las esquinas) en coordenadas locales es:

$$M_l = a_0 + a_1\xi + a_2\eta + a_3\xi\eta \qquad \text{(Ecc. 2.12)}$$

$$l = 1,2, \ldots 4.$$

Y la forma general de las funciones de forma para la presión considerando un elemento rectangular de 8 nodos (considerando todos los nodos) usando coordenadas locales es:

$$N_j = a_0 + a_1\xi + a_2\eta + a_3\xi^2 + a_4\xi\eta + a_5\eta^2 + a_6\xi^2\eta + a_7\xi\eta^2 \quad \text{(Ecc. 2.13)}$$

$$j = 1,2,\ldots 8.$$

Las funciones de forma asociadas a cada nodo para elementos rectangulares son:

<u>funciones de forma para la presión:</u>

$$M_1 = \frac{1}{4}(1 - \xi - \eta + \xi\eta)$$

$$M_2 = \tfrac{1}{4}(1 + \xi - \eta - \xi\eta) \quad \text{(Ecc. 2.14)}$$

$$M_3 = \tfrac{1}{4}(1 + \xi + \eta + \xi\eta)$$

$$M_4 = \frac{1}{4}(1 - \xi + \eta - \xi\eta)$$

<u>funciones de forma para las velocidades:</u>

$$N_1 = -\frac{1}{4}(1 - \xi)(1 - \eta)(1 + \xi + \eta)$$

$$N_2 = \frac{1}{2}(1 - \xi^2)(1 - \eta)$$

$$N_3 = -\frac{1}{4}(1 + \xi)(1 - \eta)(1 - \xi + \eta)$$

$$N_4 = \tfrac{1}{2}(1 + \xi)(1 - \eta^2) \quad \text{(Ecc. 2.15)}$$

$$N_5 = -\frac{1}{4}(1 + \xi)(1 + \eta)(1 - \xi - \eta)$$

$$N_6 = \frac{1}{2}(1 - \xi^2)(1 + \eta)$$

$$N_7 = -\frac{1}{4}(1 - \xi)(1 + \eta)(1 + \xi - \eta)$$

$$N_8 = \frac{1}{2}(1 - \xi)(1 - \eta^2)$$

En el ANEXO A se detalla el procedimiento para obtener las cuatro funciones de forma para el elemento rectangular de 4 nodos para la presión (Ecc. 2.14).

Por lo tanto, las funciones de interpolación cuadrática se utilizan para componentes de velocidad mientras funciones bilineales de interpolación para la presión.

Como resultados, las variables desconocidas para velocidades y presión son 20 por cada elemento. Por lo tanto, las variables dependientes ur_1, ur_2, y p se expresan como:

$$u_{r1} = \sum_{j=1}^{8} N_j \, u_{r1_j}$$

$$u_{r2} = \sum_{j=1}^{8} N_j \, u_{r2_j} \qquad (\text{Ecc. } 2.16)$$

$$p = \sum_{l=1}^{4} M_l \, p_l$$

Donde u_{r1_j}, u_{r2_j} y p_l son los valores nodales de la velocidad y la presión; y, N_j y M_l representan las funciones de forma para la velocidad y presión respectivamente.

2.3.3 DISCRETIZACIÓN DE ECUACIÓN DE MOVIMIENTO Y CONTINUIDAD

Ahora expresando las ecuaciones $(\text{Ecc. } 2.7)$, $(\text{Ecc. } 2.8)$ y $(\text{Ecc. } 2.9)$ en términos de las funciones de forma se obtiene [74]:

Para la ecuación de momento en x_1:

$$\rho \sum_{k=1}^{8} N_k u_{r1_k} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_1} u_{r1_j} + \rho \sum_{k=1}^{8} \mathrm{N}_k u_{r2_k} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_2} u_{r1_j}$$
$$= -\sum_{l=1}^{4} \frac{\partial M_l}{\partial \mathrm{x}_1} p_l + \mu \left(\sum_{j=1}^{8} \frac{\partial^2 N_j}{\partial \mathrm{x}_1^2} u_{r1_j} + \sum_{\mathrm{j}=1}^{8} \frac{\partial^2 N_j}{\partial \mathrm{x}_2^2} u_{r1_j} \right) - 2\rho\Omega \sum_{j=1}^{8} N_k u_{r2_j} - \rho\Omega^2 r_{\mathrm{x1}}$$
$$+ \alpha(\gamma_E) \sum_{j=1}^{8} N_j ur1_j \qquad (\text{Ecc. } 2.17)$$

Para la ecuación de momento en x_2:

$$\rho \sum_{k=1}^{8} N_k u_{r1_k} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_1} u_{r2_j} + \rho \sum_{k=1}^{8} N_k u_{r2_k} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_2} u_{r2_j}$$
$$= -\sum_{l=1}^{4} \frac{\partial M_l}{\partial \mathrm{x}_2} p_l + \mu \left(\sum_{j=1}^{8} \frac{\partial^2 N_j}{\partial \mathrm{x}_1^2} u_{r2_j} + \sum_{j=1}^{8} \frac{\partial^2 N_j}{\partial \mathrm{x}_2^2} u_{r2_j} \right) + \rho g_v - 2\rho\Omega \sum_{j=1}^{8} N_k u_{r1_j}$$
$$- \rho\Omega^2 r_{\mathrm{x2}}$$
$$+ \alpha(\gamma_E) \sum_{j=1}^{8} N_j ur2_j \qquad (\text{Ecc. } 2.18)$$

Para la ecuación de continuidad se tiene:

$$\sum_{j=1}^{8} \frac{\partial N_j}{\partial x_1} u_{r1_j} + \sum_{j=1}^{8} \frac{\partial N_j}{\partial x_2} u_{r2_j} = 0$$

(Ecc.2.19)

Una vez discretizadas las (Ecc. 2.17) y (Ecc. 2.18), se aplica el residuo ponderado de Galerkin:

$$\iint_E N_i R(x_1)\, dA = 0$$

(Ecc. 2.20)

$$\iint_E N_i R(x_2)\, dA = 0$$

Empleando el enfoque de los residuos ponderados de Galerkin a la (Ecc. 2.17) con las funciones de peso N_i, se tiene:

$$\iint_E N_i \left[\rho \sum_{k=1}^{8} N_k u_{r1_k} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x_1} u_{r1_j} + \rho \sum_{k=1}^{8} N_k u_{r2_k} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x_2} u_{r1_j} + \sum_{l=1}^{4} \frac{\partial M_l}{\partial x_1} p_l \right.$$
$$- \mu \left(\sum_{j=1}^{8} \frac{\partial^2 N_j}{\partial x_1^2} u_{r1_j} + \sum_{j=1}^{8} \frac{\partial^2 N_j}{\partial x_2^2} u_{r1_j} \right) + 2\rho\Omega \sum_{j=1}^{8} N_j u_{r2_j} + \rho\Omega^2 r_{x1}$$
$$\left. - \alpha(\gamma_E) \sum_{j=1}^{8} N_j u_{r1_j} \right] dA = 0$$

(Ecc. 2.21)

Usando el teorema de la divergencia de Gauss para el termino difusivo en la (Ecc. 2.21) se tiene:

$$\iint_E \mu \left(N_i \sum_{j=1}^{8} \frac{\partial^2 N_j}{\partial x_1^2} u_{r1_j} + N_i \sum_{j=1}^{8} \frac{\partial^2 N_j}{\partial x_2^2} u_{r1_j} \right) dA$$
$$= -\mu \left(\iint_E \frac{\partial N_i}{\partial x_1} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x_1} u_{r1_j}\, dA + \iint_E \frac{\partial N_i}{\partial x_2} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x_2} u_{r1_j}\, dA \right) + \mu \oint_\Gamma N_i \sum_{j=1}^{8} \frac{\partial N_j}{\partial n} u_{r1_j}\, ds$$

(Ecc. 2.22)

Donde Γ es la frontera del elemento E, $n = (n_{x1}, n_{x2})$ es el vector normal unitario hacia afuera del elemento y $\frac{\partial N_j}{\partial n} = \frac{\partial N_j}{\partial \mathrm{x}_1} n_{\mathrm{x}1} + \frac{\partial N_j}{\partial \mathrm{x}_2} n_{\mathrm{x}2}$ es la derivada direccional de N_j en la dirección normal a la frontera.

Por lo tanto, de la (Ecc. 2.21) ya aplicado el teorema de Green se tiene:

$$\iint_E \left[N_i \rho \sum_{k=1}^{8} N_k u_{r1_k} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_1} u_{r1_j} + N_i \rho \sum_{k=1}^{8} N_k u_{r2_k} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_2} u_{r1_j} + N_i \sum_{l=1}^{4} \frac{\partial M_l}{\partial \mathrm{x}_1} p_l \right.$$
$$+ \mu \left(\frac{\partial N_i}{\partial \mathrm{x}} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_1} u_{r1_j} + \frac{\partial N_i}{\partial \mathrm{x}_2} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_2} u_{r1_j} \right) + 2\rho\Omega \sum_{j=1}^{8} N_j u_{r2_j} + \rho\Omega^2 r_{\mathrm{x}1}$$
$$\left. - \alpha(\gamma_E) \sum_{j=1}^{8} N_j u_{r1_j} \right] dA - \mu \oint_{\Gamma} N_i \sum_{j=1}^{8} \frac{\partial N_j}{\partial n} u_{r1_j} \, ds = 0$$

(Ecc. 2.23)

Para la condición de frontera Dirichlet de la ecuación anterior se tiene:

Ecuación de momento en x_1, discretizada y con aproximación de los residuos ponderados de Galerkin.

$$\iint_E \left[N_i \rho \sum_{k=1}^{8} N_k u_{r1_k} \sum_{\mathrm{j}=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_1} u_{r1_j} + N_i \rho \sum_{k=1}^{8} N_k u_{r2_k} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_2} u_{r1_j} + N_i \sum_{l=1}^{4} \frac{\partial M_l}{\partial \mathrm{x}_1} p_l \right.$$
$$+ \mu \left(\frac{\partial N_i}{\partial \mathrm{x}_1} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_1} u_{r1_j} + \frac{\partial N_i}{\partial \mathrm{x}_2} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_2} u_{r1_j} \right) + N_i 2\rho\Omega \sum_{j=1}^{8} N_j u_{r2_j} + N_i \rho\Omega^2 r_{\mathrm{x}1}$$
$$\left. - N_i \alpha(\gamma_E) \sum_{j=1}^{8} N_j u_{r1_j} \right] dA = 0$$

(Ecc. 2.24)

Aplicando lo mismo para la ecuación (Ecc. 2.18), se obtiene la ecuación de momento en x_2:

$$\iint_E \left[N_i \rho \sum_{k=1}^{8} N_k u_{r1_k} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_1} u_{r2_j} + N_i \rho \sum_{k=1}^{8} N_k u_{r2_k} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_2} u_{r2_j} + N_i \sum_{l=1}^{4} \frac{\partial M_l}{\partial \mathrm{x}_2} p_l \right.$$
$$+ \mu \left(\frac{\partial N_i}{\partial \mathrm{x}_1} \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_1} u_{r2_j} + \frac{\partial N_i}{\partial \mathrm{x}_2} \sum_{\mathrm{j}=1}^{8} \frac{\partial N_j}{\partial x_2} u_{r2_j} \right) + N_i \rho g_v + N_i 2\rho\Omega \sum_{j=1}^{8} N_j u_{r1_j} + N_i \rho\Omega^2 r_{\mathrm{x}2}$$
$$\left. - N_i \alpha(\gamma_E) \sum_{j=1}^{8} N_j u_{r2_j} \right] dA = 0$$

(Ecc. 2.25)

De la misma manera, empleando el enfoque de los residuos ponderados de Galerkin a la (Ecc.2.19), usando las funciones de peso M_l, se tiene:

$$\iint_E M_l \left[\sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_1} u_{r1_j} + \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_2} u_{r2_j} \right] dA = 0$$

o (Ecc. 2.26)

$$\iint_E \left[M_l \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_1} u_{r1_j} + M_l \sum_{j=1}^{8} \frac{\partial N_j}{\partial \mathrm{x}_2} u_{r2_j} \right] dA = 0$$

$$l = 1,2,3,4.$$

Debido a la no linealidad en las ecuaciones de momentum (Ecc. 2.24) y (Ecc. 2.25) (término advectivo), el conjunto de ecuaciones algebraicas que se obtienen aquí no podrá ser resuelto de una vez, pero una solución iterativa es necesaria. En esa solución iterativa los términos no lineales se pueden linealizar de varias maneras diferentes. El método de linealización que se utiliza en este trabajo es la linealización de Picard [75].

La linealización de Picard consiste en:

- Asumir un valor nodal para u_{r1_k} y u_{r2_k} para cada elemento.
- Solucionar el sistema lineal global $\boldsymbol{k}\,\boldsymbol{z} = \boldsymbol{b}$ para obtener los valores nodales de velocidad u_{r1_j} y u_{r2_j} para todos los elementos.
- Se calcula el máximo error relativo nodal de las velocidades entre la solución final y el valor nodal asumido.
- El proceso iterativo MEF continúa hasta que se alcanza la convergencia. La tolerancia de la convergencia utilizada en este trabajo fue de $1x10^{-6}$.
- Para las siguientes iteraciones, u_{r1_k} y u_{r2_k} representan el valor promedio entre el valor nodal asumido y el valor nodal de solución de la iteración anterior.

Cabe destacar que de las (Ecc. 2.24), (Ecc. 2.25) y (Ecc. 2.26) se obtiene un sistema de ecuaciones por elemento en forma matricial:

$$\boldsymbol{k}^E \boldsymbol{z}^E = \boldsymbol{b}^E \quad \text{(Ecc. 2.27)}$$

Donde $\boldsymbol{k}^E$ representa la matriz de coeficientes de las (Ecc. 2.24), (Ecc. 2.25) y (Ecc. 2.26), $\boldsymbol{z}^E$ representa los valores nodales de la presión y velocidades en x_1 y en x_2, y $\boldsymbol{b}^E$ representa el vector de constantes de las ecuaciones de Navier-Stokes y continuidad por elemento. En el ANEXO A, se detalla las dimensiones de estos vectores y matrices; y, se deja expresado de forma explícita los coeficientes de la matriz $\boldsymbol{k}^E$ y el vector de constantes $\boldsymbol{b}^E$. Se detalla también la evaluación de las integrales de estos coeficientes por medio de la cuadratura de Gauss.

Ensamble global del sistema $\boldsymbol{k}\,\boldsymbol{z} = \boldsymbol{b}$

El ensamble de la matriz global $\boldsymbol{k}$ se realiza por medio de la acumulación de todos los términos presentes en las matrices locales $\boldsymbol{k}^E$ de los elementos formados a partir de las ecuaciones de momentum y de continuidad. El procedimiento de ensamble de los términos pertenecientes a la matriz de un elemento arbitrario $\boldsymbol{k}^E$ y para el caso específico de los 64 términos asociados a las 8 ecuaciones de momentum e incógnitas asociadas a la dirección x_1, se realiza por medio del cálculo de las filas/columnas que ocuparán dichos términos en la matriz global, usando para ello las ecuaciones recursivas (ver Figura 2.6.) que dependen del elemento *E* en el que se este parado y de la variable N_{x_1} (número de elementos en dirección x_1).

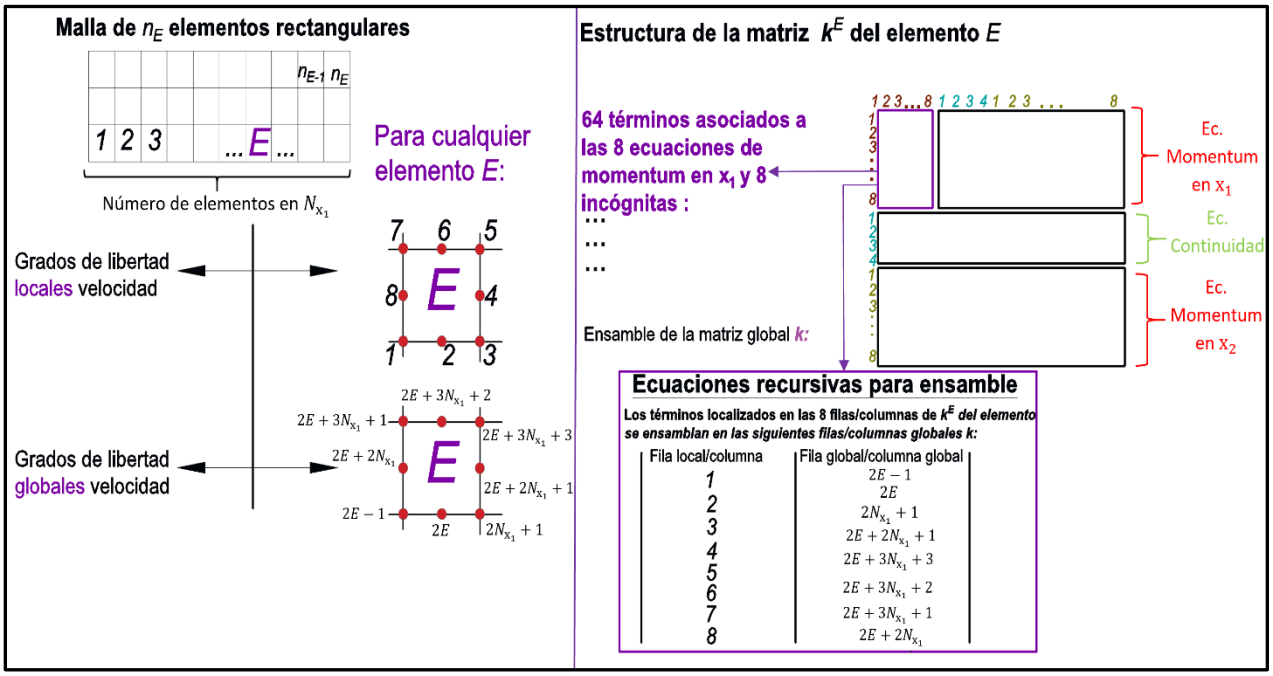


Figura 2.6.- Ecuaciones recursivas para el ensamble de la matriz global $\boldsymbol{k}$.

Nota: El ensamble del vector global $\boldsymbol{b}$ requiere el mismo procedimiento de la matriz de ensamble $\boldsymbol{k}$.

Una vez realizado el ensamble global de la matriz $\boldsymbol{k}$ y el vector de constantes $\boldsymbol{b}$, se procede a realizar la solución del sistema lineal $\boldsymbol{z} = \boldsymbol{k}^{-1} * \boldsymbol{b}$, donde $\boldsymbol{z}$ representa el vector solución de los grados de libertad globales de la velocidad en x_1, x_2 y presión, $\boldsymbol{z} = \left[u_{r1_j}\ p_l\ u_{r2_j}\right]^T$. Cabe destacar que la matriz global $\boldsymbol{k}$ y el vector de constantes $\boldsymbol{b}$ son matrices y vectores dispersos [1], con el objetivo de no sobrecargar la memoria RAM al momento de ensamblar esta matriz y este vector respectivamente. El solucionador del sistema lineal $\boldsymbol{z} = \boldsymbol{k}^{-1} * \mathbf{b}$ que se utiliza en este trabajo es el solucionador de factorización de Lu de MATLAB. Se utiliza este solucionador debido a las propiedades de nuestra matriz $\boldsymbol{k}$ (matriz cuadrada, no simétrica y no diagonal).

[1] *La función dispersa convierte una matriz completa en forma dispersa, excluyendo cualquier elemento que sea cero.*

CAPÍTULO 3: OPTIMIZACIÓN TOPOLÓGICA E IMPLEMENTACIÓN

3.1 RESUMEN INTRODUCTORIO

La estrategia metodológica aplicada en este capítulo se fundamenta en el uso de herramientas matemáticas de optimización en el contexto de la mejora continua e inherente del desempeño de turbomáquinas, a través de la modificación del diseño topológico del álabe. Se detalla la formulación matemática de un problema general de optimización como base del planteamiento que se usa en la implementación específica de un conjunto de ecuaciones asociadas al fenómeno físico propio del diseño a optimizar con relación al espacio geométrico que sirve de escenario en el proceso numérico. Se describe de manera general la metodología comúnmente usada en el método de optimización topológica (MOT), se realiza una derivación hacia el entorno físico sobre el cual se plantea e interactúa la metodología numérica (medio de flujo poroso), se describe el proceso de implementación del esquema numérico correspondiente, se deja expresado en forma explícita y discreta las ecuaciones de las funciones objetivos a minimizar (energía de disipación viscosa y vorticidad); y, se sintetiza en una función bi-objetivo (por medio del método de los pesos de suma ponderada) estas dos funciones, de manera que al tener una sola ecuación se pueda jugar con los pesos para dar priorización de minimización de una función sobre la otra. Se deja expresado de forma explícita y discreta, el análisis de sensibilidades para la energía de disipación viscosa y la vorticidad. También se expresa en términos de función bi-objetivo por medio del método de la suma ponderada el análisis de sensibilidad para estas dos funciones objetivos. Finalmente se describe algunos filtros que se incluyen en los algoritmos de optimización topológica y se detalla la formulación matemática del filtro usado en este trabajo de investigación (filtro AWS).

3.2 OPTIMIZACIÓN EN EL PROCESO DE DISEÑO

Independientemente del área de estudio, el proceso de diseño de productos siempre está ligado al cambio dinámico de los requerimientos dados por los usuarios (rendimiento, desempeño) y a un diseño base que mantiene en el tiempo muchos de sus atributos (color, forma, tamaño, etc.), ya que representan características inherentes al producto y que nacieron en el momento de resurgir el producto como respuesta a una necesidad o problemática dada. La creatividad de los diseñadores en etapas tempranas (diseño conceptual) tiene la capacidad de generar cambios positivos en los requerimientos dados para un diseño, sin embargo la prematura toma de decisiones puede frustrar el proceso de diseño, ya que limita y restringe el diseño bajo el criterio de los diseñadores, el cual puede no ser el más óptimo en diseños no intuitivos, debido a la gran cantidad de variables que intervienen y con ello la complejidad global del sistema bajo análisis [76]. Inclusive existen problemas de diseños tan complejos que el conocimiento empírico y la experiencia del diseñador no es suficiente para proponer alternativas de diseño que se traduzcan a una mejora de desempeño [77].

A pesar de toda esta búsqueda por mejorar los diseños de los productos, surge también la necesidad de mejorar los componentes estructurales, dando así el origen a la optimización estructural. Esta optimización en estructuras se fundamenta básicamente en una herramienta matemática que permite hallar un diseño óptimo valga la redundancia en una estructura bajo las diferentes restricciones que intervienen (funcionales u operacionales). El proceso de diseño mediante la metodología de optimización estructural inicia mediante la traducción directa de los requerimientos demandados por los usuarios, hacia un modelo matemático cuyo valor representa el objetivo demandado y sus parámetros las herramientas que usa la metodología para buscar la combinación que genere la mejora esperada.

Dados los elementos anteriores, una vez que se tiene un modelo matemático para el problema de diseño, es decir, una función que permita obtener una variable de respuesta a partir de la relación de diferentes factores, se puede plantear en términos matemáticos una formulación de un problema de optimización que está dada por (comúnmente minimización) [78] :

$$\text{minimizar } f(\boldsymbol{x}) \qquad \text{(Ecc.3.1)}$$

$$sujeto\ a\text{: } g(\boldsymbol{x}) \leq 0 \text{ y } h(\boldsymbol{x}) = 0$$

Donde:

$$\boldsymbol{x} \ \rightarrow \ \text{vector de parámetros (factores de diseño).}$$

$$f(\boldsymbol{x}) \rightarrow \text{función objetivo (variable de respuesta).}$$

$$g(\boldsymbol{x}) \rightarrow \text{restricciones de desigualdad.}$$

$$h(\boldsymbol{x}) \rightarrow \text{restricciones de igualdad.}$$

Este problema de optimización puede ser resuelto por algoritmos que se utilizan para encontrar un diseño que optimiza las variables de respuesta de cierto problema dado. En este problema de optimización $f(\boldsymbol{x})$ representa la función a minimizar; y, $h(\boldsymbol{x}) = 0$ y $g(\boldsymbol{x})$ representan funciones que restringen el espacio de solución factible tanto de igualdad y desigualdad respectivamente.

La solución a este problema de optimización en sí consiste en encontrar los valores de $\boldsymbol{x}^*$ tal que minimice la función $f(\boldsymbol{x})$, satisfaciendo la siguiente ecuación [79]:

$$\boldsymbol{x}^* = arg\left(\min_{\boldsymbol{x}} f(\boldsymbol{x})\right)$$

3.3 OPTIMIZACION ESTRUCTURAL

El método de optimización estructural comprende dos categorías (paramétricas y no paramétricas) y a su vez se subdividen en cuatro categorías [78]:

- Optimización de tamaño
- Optimización material
- Optimización de forma
- Optimización Topológica

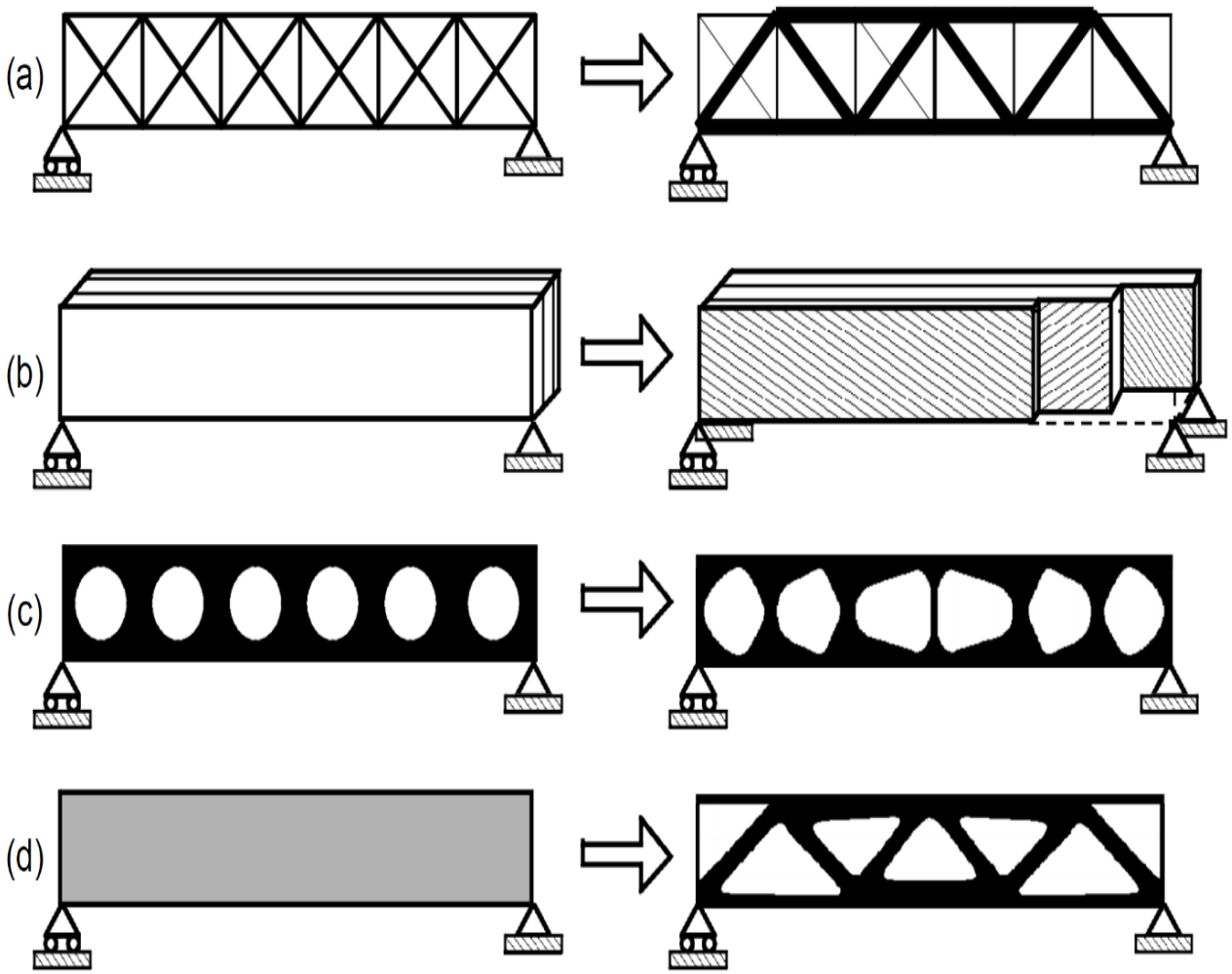


Figura 3.1.- Cuatro categorías de optimización estructural: **(a)** optimización de tamaño, **(b)** optimización de material, **(c)** optimización de forma y **(d)** optimización topológica [77].

La visualización y entendimiento de los cuatro tipos de optimización se llevan a cabo según la Figura 3.1 que corresponde al diseño de una viga con la restricción de conservación de la rigidez. La optimización de tamaño (Figura 3.1a) se concentra en hallar el área óptima de los elementos

que conforman la cercha base (algunos elementos pueden tener como resultante área aproximadamente cero, por lo cual pueden ser descartados del diseño); por otro lado, la optimización de material (Figura 3.1b) (en función de las condiciones de carga) obtiene el espesor y orientación de las capas requeridas que optimice la estructura (pueden usarse diferentes materiales); la optimización de forma (Figura 3.1c) parte de una geometría inicial y opera sobre las fronteras a partir de funciones de contorno con el objetivo de buscar la forma óptima; y la optimización topológica (Figura 3.1d) combina todos los métodos anteriores partiendo de un dominio de diseño continuo en el cual juega con la distribución de material hasta obtener el valor óptimo del requerimiento de interés bajo restricciones impuestas, determina las características como número, ubicación y forma de los agujeros o contornos, así como la conectividad del dominio.

En procesos de diseño convencionales es frecuente llevar a cabo un proceso secuencial donde se usan los cuatro tipos de optimización o algunas de estas para obtener la topología que será entregada al proceso de manufactura, sin embargo, en algunos casos es común pasar directamente de la optimización topológica para el proceso constructivo.

La esencia del modelo de optimización estructural puede ser usada en otros campos de diseño, tal como el área de fluidos donde el objetivo es encontrar la distribución de fluido en el dominio de diseño, tal como es el caso del diseño de la optimización topológica del rotor de turbomáquinas de flujo radial dadas por Romero y Silva (2014) [4].

3.4 RESEÑA HISTÓRICA

El método de optimización topológica tiene una historia por detrás, esta, surgió recién a finales del siglo XX; pero, hasta llegar a ese punto, investigadores como Newton, Lagrange, Cauchy, Leibnitz, etc., han intervenido con sus contribuciones en el desarrollo de los métodos de optimización, contribuciones que se han venido aplicando al cálculo variacional y otras herramientas aplicadas como solución de problemas de minimización de funcionales. Los primeros conceptos de buscar las formas óptimas de elementos estructurales están contenidos en las investigaciones realizadas por Galileo Galilei (1564-1642) en el libro Discorsi, en donde fue el primero en realizar investigaciones sistemáticas sobre el proceso de fractura de cuerpo frágiles [80]. Adicional a estas aportaciones que se vinieron dando, podemos destacar que el primer trabajo que sirvió como base para la optimización topológica fue el realizado por Michell en 1904 [81], quién estableció las condiciones de optimalidad sobre estructuras cargadas.

Fue entonces que en 1988, Bendsϕe y Kikuchi [82], por primera vez establecieron un procedimiento computacional para la optimización topológica, donde se desarrolló un método de distribución microestructural de material conocido como el método de homogenización para la minimización del peso en estructuras sujetas a restricciones de rigidez, que consistía en representar el espacio finito de diseño como un cuerpo celular con una microestructura periódica (elementos de tamaño unitario con cavidades tal como se observa en la Figura 3.2, donde el proceso de optimización consiste en encontrar el espesor y orientación de las paredes que forman las cavidades a tal punto de optimizar la respuesta de la estructura en términos de los desplazamientos y sujeto a restricciones de rigidez. En 1990, Martin Bendsϕe propuso una estrategia de solución al problema de optimización bajo los términos de "Densidades artificiales", que posteriormente esta solución iba a ser introducida como el modelo de densidades SIMP, por sus siglas en inglés "*Solid Isotropic Microstructure with Penalization*" [83].

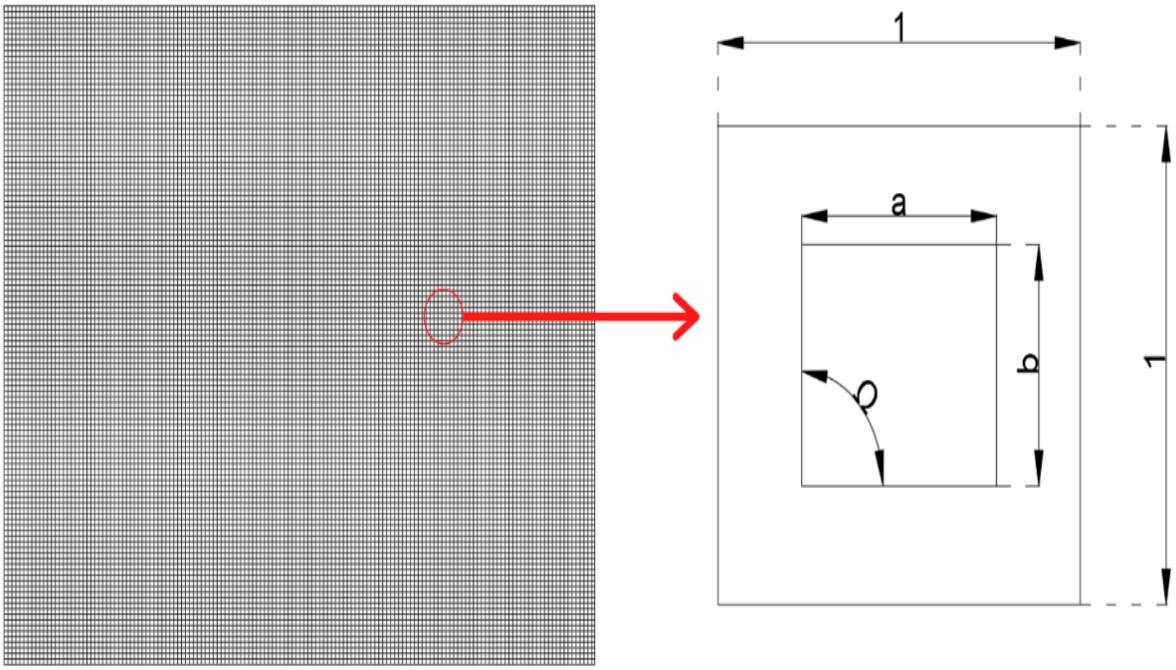


Figura 3.2. Dominio representado por microestructura perforada y composición de una celda. Imagen adaptada de [84].

El modelo de material SIMP se diferencia del método de homogenización debido a que este modelo introduce la posibilidad de utilizar materiales ficticios con pseudo-densidades intermedias, las cuales progresivamente pueden ser penalizadas hasta obtener un diseño únicamente de materiales sólidos y vacíos [84].

Con base a todo lo descrito anteriormente, podemos destacar que la optimización topológica es una técnica relativamente nueva que ha venido desarrollándose continuamente y que las aplicaciones del método han empezado a aparecer en diferentes campos de la ingeniería. A

pesar de que la optimización topológica tuvo su origen en el diseño estructural; debido a la maduración del método en el siglo XX se han incluido aplicaciones de combinación de estructuras, transferencia de calor, acústica, fluidos, aeroelasticidad, diseño de materiales, entre otros problemas multi-físicos [85].

En la Figura 3.3, se observa la evolución cronológica del método de optimización topológica y su derivación de esta hacia las ciencias de la mecánica de fluidos que fue introducida por Borrvall y Petersson [44]. Es a partir de este punto, que se han desarrollado de manera exponencial trabajos relacionados a problemas con la mecánica de fluidos y, específicamente, trabajos relacionados al diseño de máquinas de flujo radial, Romero & Silva (2014) [4]. Adicionalmente de la inclusión del método de optimización topológica en el área de fluidos, se puede encontrar en la literatura investigaciones del MOT aplicado a las turbomáquinas. Algunos de estos trabajos serán presentados en la siguiente sección 3.5 .

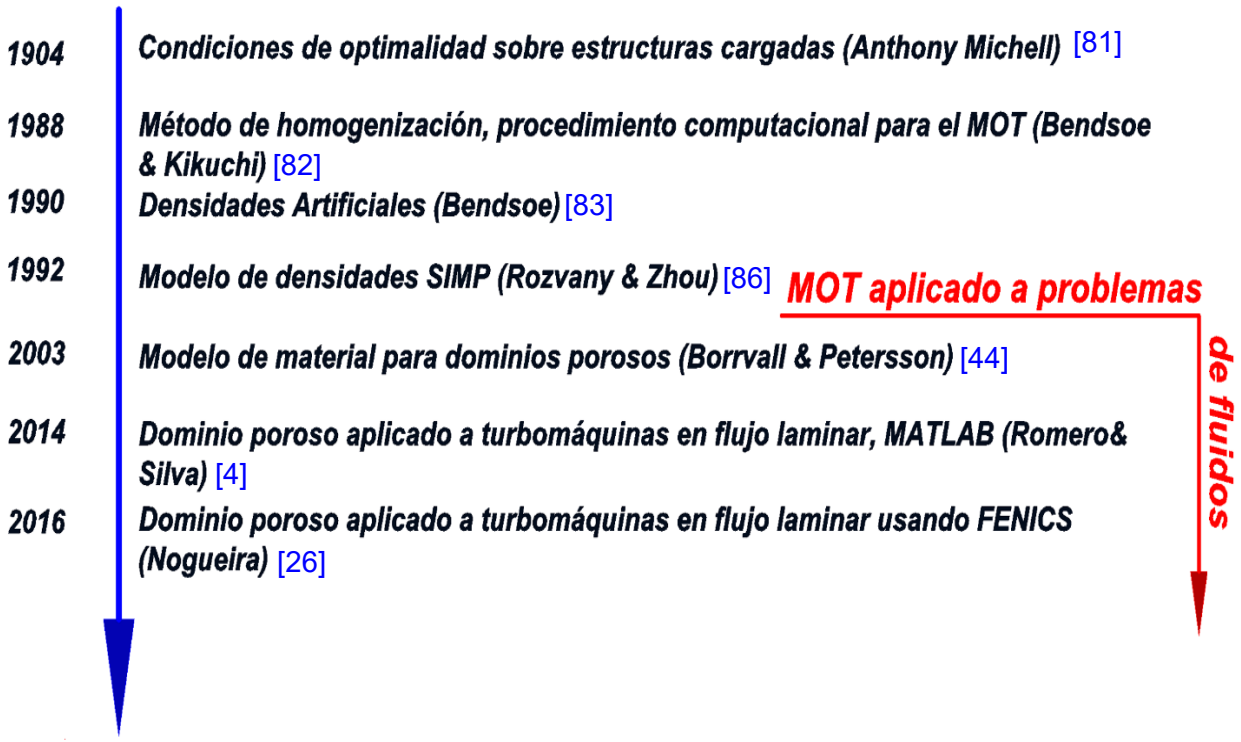


Figura 3.3.- Evolución cronológica del método de optimización topológica y su derivación hacia las ciencias de la mecánica de fluidos.

3.5 APLICACIONES DEL MOT EN BOMBAS CENTRÍFUGAS

A partir del primer trabajo en fluidos realizado por Borrvall y Petterson en el 2003 [44] (tal como se mostró en la Figura 3.3, donde se fundamentan las bases matemáticas en profundidad para la optimización topológica de flujos de Stokes), se han venido desarrollando un sin número de trabajos en el área de la mecánica de fluidos y en especial en la física de flujos de máquinas rotativas como las bombas centrífugas.

Para mencionar algunos de los trabajos de optimización topológica aplicada al diseño de bombas centrífugas, se divide esta sección en dos (sección 3.5.1 y 3.5.2). La primera sección trata de trabajos de turbomáquinas enfocados a flujos laminar estable y la siguiente sección de trabajos de turbomáquinas enfocados a flujos no newtonianos usando el MOT como herramienta de diseño.

3.5.1 FLUJO LAMINAR ESTABLE

El trabajo pionero que trata sobre el diseño de un álabe aplicado a turbomáquinas usando el MOT fue el publicado por Romero y Silva (2014) [4], en el cual realizan una optimización de rotores de máquinas de flujo radial (bombas y turbinas), donde se propone una metodología para el diseño de los álabes usando el método de optimización topológica basada en densidades para flujos laminares, incluyendo las fuerzas centrífugas y la de Coriolis. Una de las topologías obtenidas en este trabajo de investigación es la que se muestra en la Figura 3.4, considerando como variable de diseño inicial a la topología de un alabe recto y minimizando la energía de disipación como función objetivo. Se puede observar que se obtiene una topología de doble canal, similar a la configuración propuesta por Gölcü et al. [35] donde esencialmente, propone considerar splitters (separadores) entre dos álabes, obteniendo así canales dobles que aumentan la eficiencia.

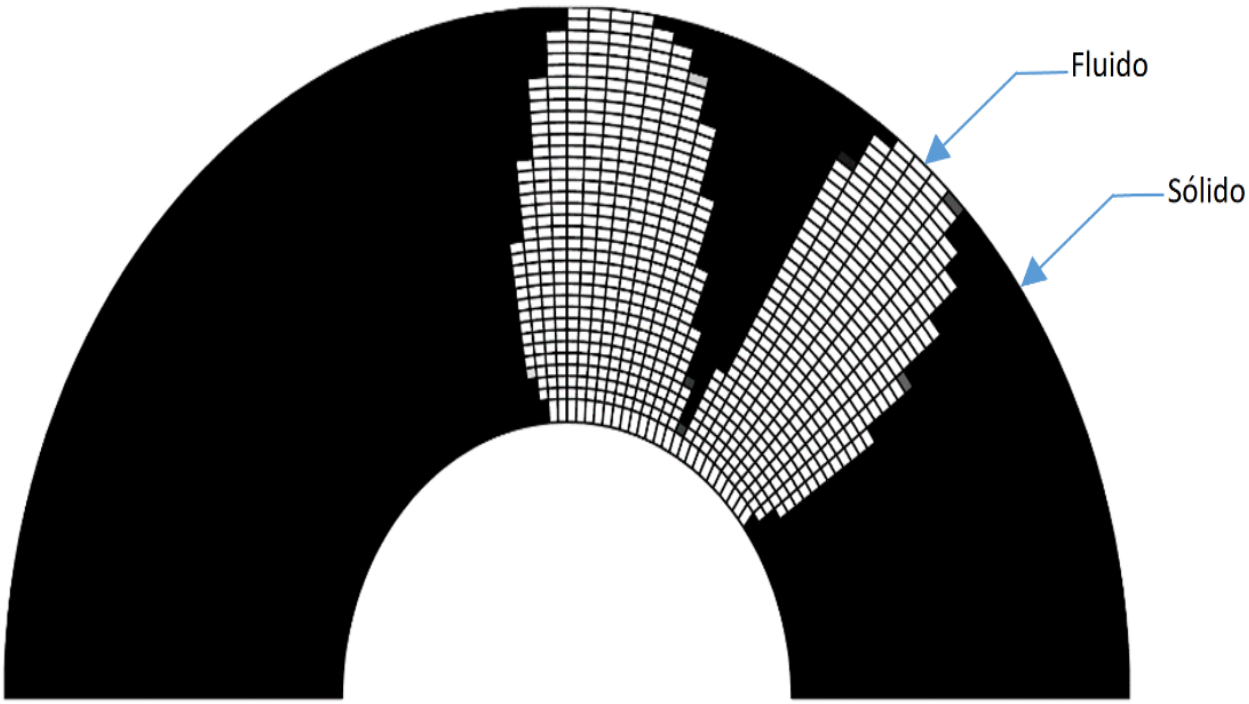


Figura 3.4.- Topología optima obtenida considerando la variable de diseño inicial a la topología de un alabe recto y minimizando la energía de disipación $w_d = 1$ [4].

Luego, L.F.N.Sá [26] extendió el trabajo realizado por Romero y Silva aplicando optimización topológica basada en densidades al diseño de bombas centrífugas a pequeña escala, considerando como funciones objetivos tanto la energía de disipación y vorticidad. Los resultados optimizados obtenidos son procesados posteriormente y se fabrican mediante el uso de una impresora 3D, construyendo un prototipo con un motor eléctrico. El proceso de desarrollo del rotor de la bomba de flujo radial impreso en 3D se muestra en la Figura 3.5, donde posteriormente a su caracterización experimental se mide el flujo del fluido (caudal) y la altura de presión dada por la bomba. Estos resultados experimentales se verifican con los computacionales, donde se obtienen mejoras en la eficiencia.

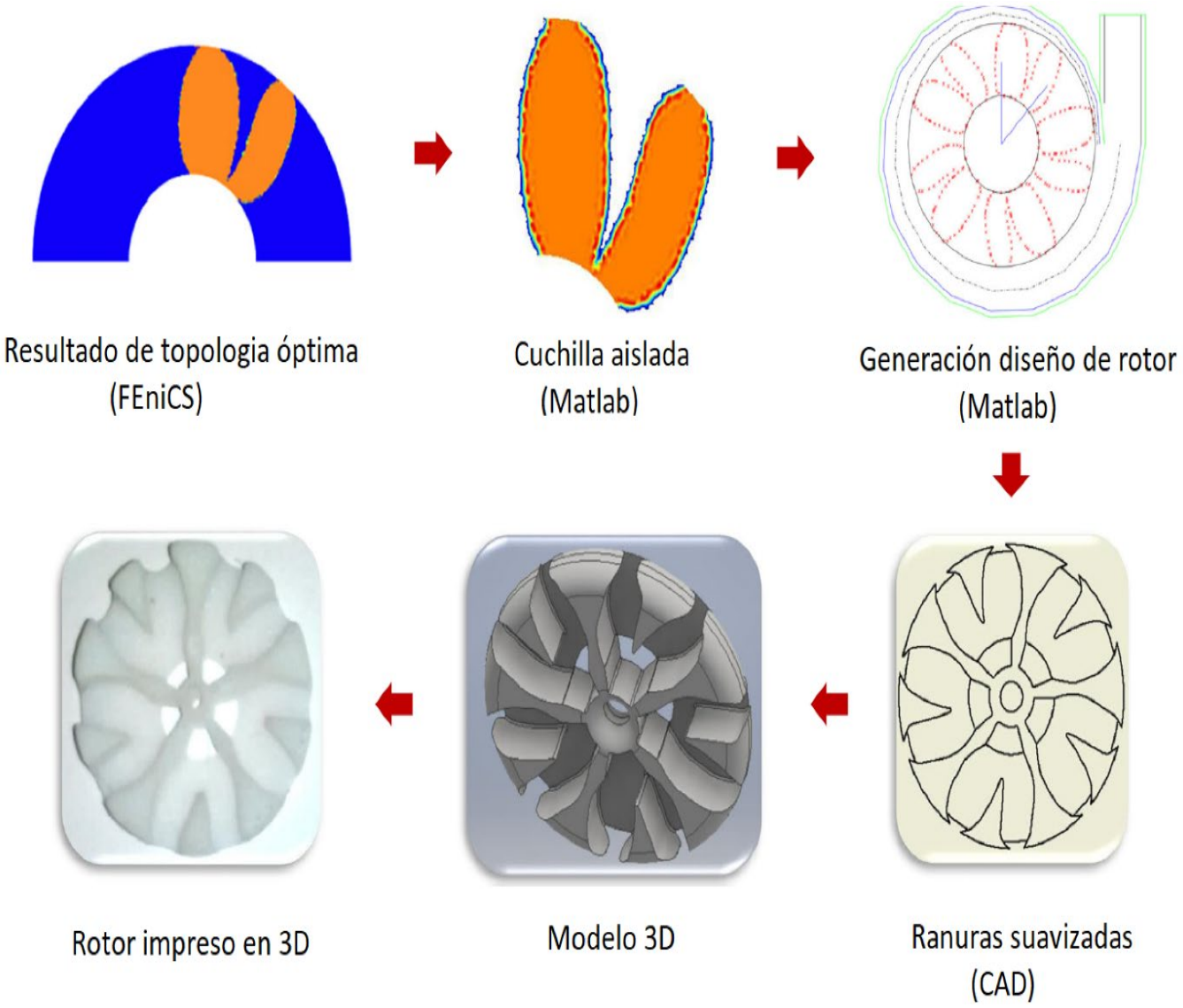


Figura 3.5.- Proceso de desarrollo del rotor de la bomba de flujo radial [26].

Por último, para mencionar otro trabajo de turbomáquinas usando el MOT, Alonso et al. [87] aplica la optimización topológica basada en densidades al diseño de bombas centrífugas de tipo Tesla. La característica de este tipo de bombas en particular es que es una bomba centrífuga que no posee álabes y la manera de generar momentum a un fluido es por medio de canales rotatorios. Uno de los resultados de la optimización topológica de flujo de remolino en dos dimensiones se muestra en la Figura 3.6, para un canal de 500 r.p.m y un flujo de 0.5 $[\mathrm{L/min}]$. Se puede observar en esta figura que las velocidades cerca de la salida son de magnitud de mayor valor en comparación con las velocidades en la entrada, lo cual es causado por el efecto viscoso de la capa límite, consistente con el principio de Tesla.

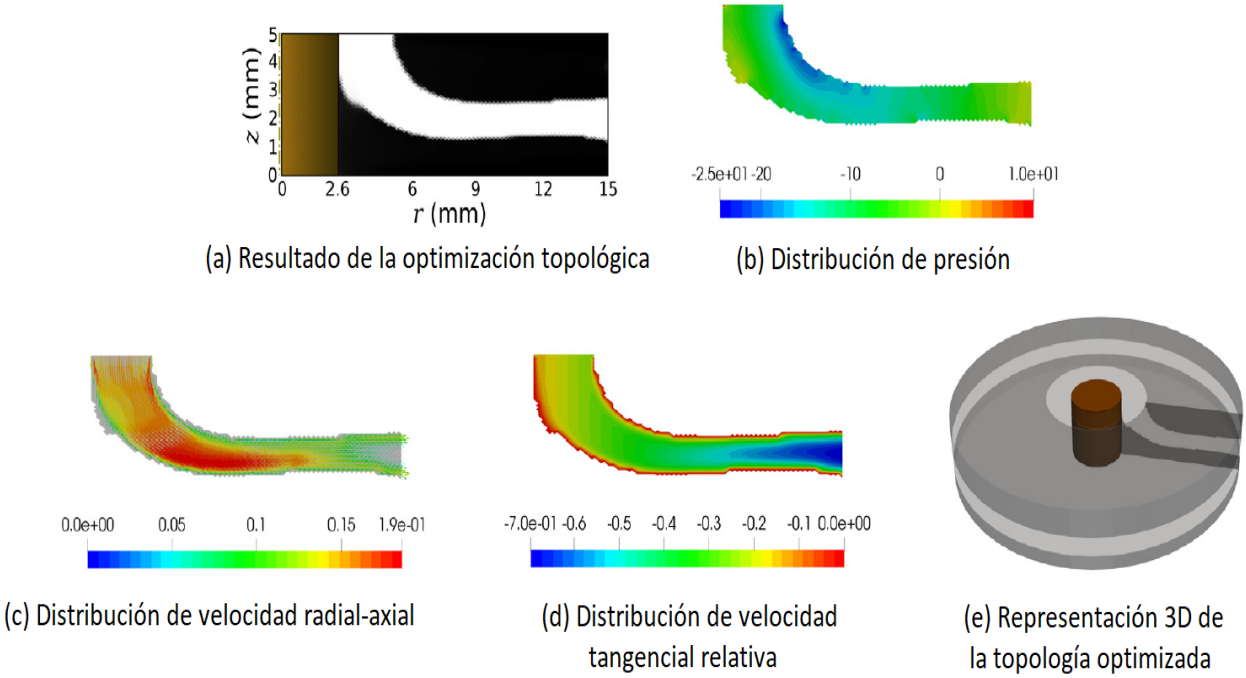


Figura 3.6.- Resultado de la optimización topológica de una bomba Tesla para un canal de 500 r.p.m y 0.5 $[\mathrm{L/min}]$ [87].

3.5.2 FLUIDOS NO NEWTONIANOS

Los fluidos no newtonianos son fluidos donde la viscosidad no se mantiene constante y esta varía en función de la temperatura y fuerza cortante a la que esté sometida. Un ejemplo de estos fluidos no newtonianos representa a la sangre, cuya viscosidad varía dependiendo de la temperatura a la que esté sometida.

Entonces basándonos en trabajos de la literatura donde la optimización topológica es aplicada al diseño de álabes en turbomáquinas, se encuentra el trabajo realizado por Romero y Silva (2016) [29], donde extendieron el trabajo de L.F.N.Sá [26] usando el MOT para encontrar topologías óptimas en el diseño de álabes pero esta vez aplicado a flujos no newtonianos como la sangre, para el diseño de bombas centrífugas a pequeña escala simulando el funcionamiento de un dispositivo de asistencia ventricular. Estos resultados fueron comparados con las topologías obtenidas para el caso de flujos newtonianos tal como se muestra en la Figura 3.7.

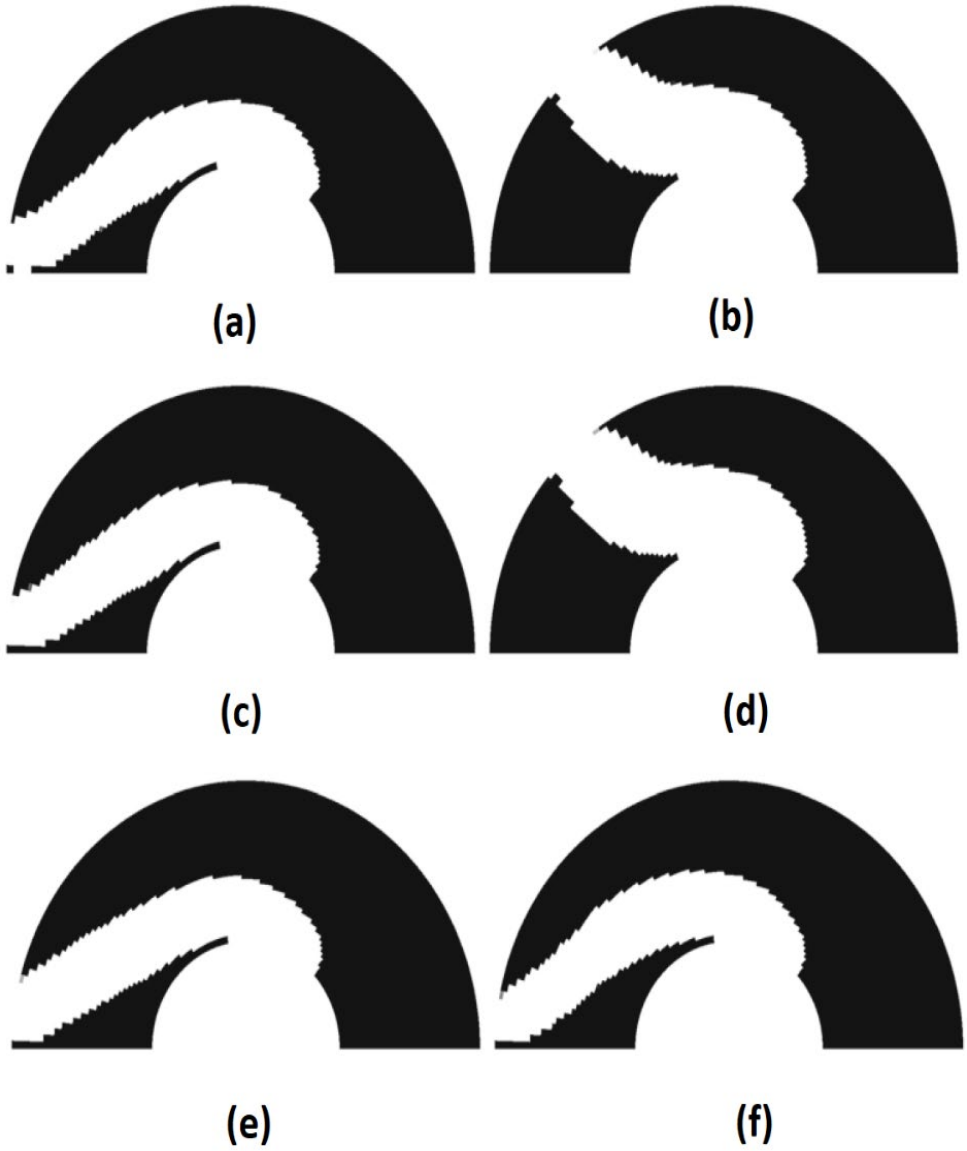


Figura 3.7.- Comparación de topologías optimizadas obtenidas considerando solo la minimización de la disipación de energía para flujos newtonianos y flujos no newtonianos para diferentes r.p.m del rotor: 500 r.p.m **a)** Newtoniano **b)** No newtoniano; 600 r.p.m **c)** newtoniano; **d)** no newtoniano; 800 r.p.m **e)** newtoniano; **f)** no newtoniano [29].

Los resultados topológicos obtenidos por Romero y Silva en este trabajo de investigación concluyen que para bajas r.p.m en el impeler del rotor $(< 700\, r.p.m)$, muestran que las topologías optimizadas para flujo newtonianos y no newtonianos difieren significativamente, considerando la forma y la posición del canal de salida; en cambio, para r.p.m altos $(> 700\, r.p.m)$ las topologías optimizadas para flujo newtoniano y no newtoniano no difieren significativamente, obteniendo así topologías muy similares.

3.6 IMPLEMENTACIÓN DEL MOT EN FLUJOS POROSOS (DESCRIPCIÓN DEL MÉTODO)

3.6.1. INTRODUCCIÓN

El método de optimización topológica se origina en el campo de la mecánica de sólidos, donde surgió la optimización de tamaño y de forma a finales de la década de 1980. Estas bases de optimización estructural han sido aplicadas para el caso de fluidos y por lo tanto es común encontrar investigaciones para cada uno de los tipos de optimización en los que se busca distribuir un fluido específico en el dominio de diseño, con el objetivo de maximizar o minimizar alguna variable de desempeño de flujo. Por ejemplo, Wang (2012) [88] para la optimización paramétrica de las características geométricas de un intercambiador de calor mejorando así la eficiencia de recuperación de calor, T.Lehnhäuser (2005) [89] desarrolló un método numérico para la optimización de forma en dominios de flujos de fluidos y, así, otras investigaciones han tenido su relevancia con estas metodologías de optimización.

Adicionalmente a estas metodologías de optimización, la optimización topológica ha venido desempeñando un rol importante en todas las áreas de la mecánica computacional, termo-mecánica e incluso multifísica [6]. Es así como en la dinámica de fluidos computacional, el uso de la optimización topológica fue iniciada por Borrvall y Petersson (2003), considerando regiones óptimas en flujo de Stokes, traduciendo así los conceptos de optimización estructural a una representación de dominio de diseño como un medio poroso y usando a la permeabilidad como la variable de diseño.

3.6.2. MODELO DE MATERIAL APLICADO A LAS ECUACIONES DE NAVIER-STOKES

El dominio de diseño poroso en optimización topológica aplicado a fluidos viene definido matemáticamente por la fuerza de Darcy, cuya (Ecc. 2.5) es detallada en la sección 2.2. Esta fuerza de Darcy es introducida en las ecuaciones de Navier-Stokes (ver (Ecc. 2.4)), para posteriormente proceder con su discretización espacial en elementos finitos y permitir así encontrar los campos de velocidades y de presión.

Esta fuerza de Darcy viene formulada por la función de absorción alpha (α), que físicamente representa la impermeabilidad de un flujo poroso; es decir, dividiendo las regiones del dominio entre alta permeabilidad de material, interpretada como fluido, y baja permeabilidad del material representada como sólido, es decir, la porosidad virtualmente separa las regiones que son fluido de las que son sólidos [90]. Esta función de impermeabilidad (modelo tipo Brinkman [91]) controla la distribución del material en todo el dominio; y, su valor depende de la variable de diseño de optimización gamma γ_E. Esta función de absorción viene dada por la siguiente expresión [44]:

$$\alpha(\gamma_E) = \alpha_{max} + (\alpha_{min} - \alpha_{max})\,\gamma_E \frac{1+q}{\gamma_E + q} \qquad \text{(Ecc. 3.2)}$$

donde α_{min} y α_{max} son los mínimos y máximos valores de α, respectivamente; y q es un parámetro real y positivo utilizado para ajustar la convexidad de la función de interpolación $\alpha(\gamma_E)$. Entonces, cuando q → infinito, $\alpha \rightarrow \alpha_{max} + (\alpha_{min} - \alpha_{max})\,\gamma_E$ es una función lineal.

En la Figura 3.8, se muestra el comportamiento de la función de interpolación de material considerando un $\alpha_{max} = 10.000$ y un $\alpha_{min} = 0$. El proceso de optimización ayuda a obtener valores de variable de diseños entre 0 y 1 γ_E ($\gamma_E \approx 0$ y $\gamma_E \approx 1$), dado que un valor intermedio no tendría un significado físico. Por lo tanto, cuando $\gamma_E \approx 1$ $\alpha = \alpha_{min}$, lo cual representa un flujo de puro fluido, mientras que cuando $\gamma_E \approx 0$ $\alpha = \alpha_{max}$, representa un flujo restringido dentro de un medio poroso.

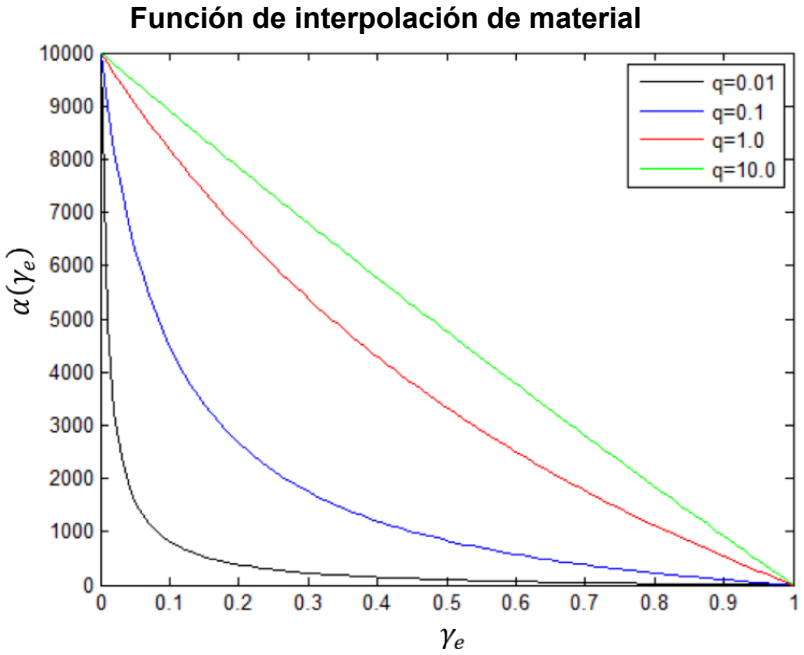


Figura 3.8.- $\alpha(\gamma_E)$ como función de los parámetros γ_E y q. Figura tomada de L.F.N.Sá [92].

Entonces el problema de optimización se formula de la siguiente manera:

$$\min_{\boldsymbol{\gamma}} c(\boldsymbol{z}(\boldsymbol{\gamma}), \boldsymbol{\gamma})$$

$$\text{Sujeto a: } \begin{cases} \boldsymbol{g}(\boldsymbol{z}(\boldsymbol{\gamma}), \boldsymbol{\gamma}) \leq 0 \\ \boldsymbol{h}(\boldsymbol{z}(\boldsymbol{\gamma}), \boldsymbol{\gamma}) = 0 \\ \gamma_{min} \leq \gamma_E \leq \gamma_{max} \end{cases} \qquad \text{(Ecc. 3.3)}$$

Donde:

$c(\boldsymbol{z}(\boldsymbol{\gamma}), \boldsymbol{\gamma})$ → función bi-objetivo.

$\boldsymbol{g}(\boldsymbol{z}(\boldsymbol{\gamma}), \boldsymbol{\gamma})$ → restricciones de desigualdad.

$\boldsymbol{h}(\boldsymbol{z}(\boldsymbol{\gamma}), \boldsymbol{\gamma})$ → restricciones de igualdad.

$\boldsymbol{z} = \left[u_{r1_j}\ p_l\ u_{r2_j}\right]^T$ → vector con los grados de libertad de fluidos.

La función $c(\boldsymbol{z}(\boldsymbol{\gamma}), \boldsymbol{\gamma})$ es una función bi-objetivo que incluye la minimización de la energía de disipación y la vorticidad en todo el rotor. Las restricciones de desigualdad $\boldsymbol{g}(\boldsymbol{z}(\boldsymbol{\gamma}), \boldsymbol{\gamma}) \leq 0$, corresponde a la restricción de volumen para nuestro problema en específico. Finalmente, las restricciones de igualdad $\boldsymbol{h}(\boldsymbol{z}(\boldsymbol{\gamma}), \boldsymbol{\gamma}) = 0$ corresponden a las ecuaciones de modelación de los fluidos presentadas en el MEF para flujo viscoso en forma residual (igualadas a cero).

3.6.3 ENERGÍA DE DISIPACIÓN VISCOSA

La energía de disipación viscosa es una de las funciones objetivos en este trabajo. Cuando el fluido entra por el impeler (ver Figura 2.1), el fluido adquiere energía interna representada en forma cinética, de presión y potencial. Cabe destacar que esta energía interna del fluido que adquiere en la entrada del rotor no va a ser la misma a la salida de este debido a que van a existir perdidas de energía generadas por los esfuerzos viscosos en la ecuación de energía. Estos esfuerzos viscosos surgen por la interacción que existe entre el rotor y el fluido en cuestión, que básicamente se encarga de cambiar el momentum del fluido cambiando así su dirección y su energía interna. Es así como la energía de disipación viscosa representa la energía, valga la redundancia, que pierde el fluido desde la entrada de la bomba hasta la salida de la misma; y es por eso que el objetivo de los diseñadores en estas turbomáquinas es siempre buscar la manera más optima de diseñar los alabes con el fin de que se pierda la menos cantidad de energía interna del fluido posible, desde la entrada hasta la salida del rotor.

Adicional a lo anterior, la minimización de la pérdida de energía es importante para una gran cantidad de aplicaciones, ya sea para sistemas hidrosanitarios, en sistemas contraincendios hasta en aplicaciones de pequeña escala como es el caso de las bombas cardíacas, justificando así la selección del uso de esta función en este trabajo de investigación.

Esta ecuación de energía de disipación está dada de forma discreta por la siguiente expresión [4]:

$$c_d(\gamma_E) = \frac{1}{2}\left[u_{r1_j},\, u_{r2_j}\right]\ \boldsymbol{k}^E\ \left[u_{r1_j},\, u_{r2_j}\right]^T \qquad \text{(Ecc. 3.4)}$$

Donde

- c_d → disipación de energía, escalar.
- γ_E → variable de diseño por elemento finito.
- $ur1_j, ur2_j$ → valores nodales de velocidad relativa por elemento.
- $\boldsymbol{k}^E$ → matriz de coeficientes de grados de libertad por elemento.

3.6.4 VORTICIDAD

Fuertes flujos secundarios circulatorios (flujos de vórtice) se observan en impulsores de flujo mixto como en bombas axiales/centrífugas, turbinas y compresores. Estos flujos de vórtices son indeseables ya que son los responsables de las pérdidas de carga (pérdidas de presión), flujos con falta de uniformidad y deslizamiento. El contraflujo causado por la recirculación puede provocar cavitación en las palas del impulsor [93].

Los diseñadores de turbomáquinas a menudo emplean elementos disruptivos/guías de flujo como paletas divisoras y otras modificaciones (afectando negativamente la eficiencia de la máquina) en lugar de centrarse en las causas reales y mecanismos físicos intrínsecos que generan flujos secundarios de vórtices [94]. Es así como en este trabajo también se optimiza la topología del flujo de la máquina para minimizar la vorticidad reduciendo así el efecto negativo que provoca.

En este trabajo, la vorticidad relativa será considerada como función objetivo. Para el caso en dos dimensiones, la vorticidad relativa en su forma diferencial viene dada por la siguiente expresión [5]:

$$\nabla x \boldsymbol{u_r} = \left(\frac{\partial u_{r2}}{\partial \mathrm{x}_1} - \frac{\partial u_{r1}}{\partial \mathrm{x}_2}\right) \quad \text{(Ecc. 3.5)}$$

Donde u_{r1}, u_{r2} son las componentes de las velocidades relativas en la dirección x_1, x_2. La rotación en el plano x_1, x_2 es de fundamental importancia cuando se estudia la cinemática del fluido. La función de vorticidad (Ecc. 3.5), puede ser escrita en su forma discreta de la siguiente manera [4]:

$$c_r(\gamma_E) = \left[u_{r1_j},\, u_{r2_j}\right] \boldsymbol{M_r} \left[u_{r1_j},\, u_{r2_j}\right]^T \qquad \text{(Ecc. 3.6)}$$

$$\boldsymbol{M_r} = \begin{bmatrix} \boldsymbol{Mr_{22}} & \boldsymbol{-Mr_{21}} \\ \boldsymbol{-Mr_{12}} & \boldsymbol{Mr_{11}} \end{bmatrix} \quad \text{(Ecc. 3.7)}$$

Donde:

C_r → vorticidad, escalar.

$\boldsymbol{M_r}$ → matriz de vorticidad por elemento finito.

Esta matriz de vorticidad $\boldsymbol{M_r}$ se obtienen por elemento, mediante las derivadas cruzadas de la velocidad en cada dirección (similar al término difusivo en la ecuación de Navier-Stokes) donde los coeficientes se detallan de forma explícita en las siguientes expresiones:

$$\boldsymbol{M}_{r_{22}} = \iint_E \mu \frac{\partial N_i}{\partial x_2} \frac{\partial N_j}{\partial x_2} dA \qquad \boldsymbol{M}_{r_{21}} = \iint_E \mu \frac{\partial N_i}{\partial x_2} \frac{\partial N_j}{\partial x_1} dA$$

$$\boldsymbol{M}_{r_{12}} = \iint_E \mu \frac{\partial N_i}{\partial x_1} \frac{\partial N_j}{\partial x_2} dA \qquad \boldsymbol{M}_{r_{11}} = \iint_E \mu \frac{\partial N_i}{\partial x_1} \frac{\partial N_j}{\partial x_1} dA$$

$i = 1,2 \ldots 8.$ (i representa la función de prueba que estemos evaluando).
$j = 1,2 \ldots 8.$ (j representa el grado de libertad que estemos evaluando).

Para detallar de mejor manera la matriz $\boldsymbol{Mr}$ de la (Ecc. 3.7), en la Figura 3.9 se ilustra las dimensiones de esta matriz por elemento de $16\ x\ 16$. Esta matriz esta compuesta por las matrices $\boldsymbol{M}_{r_{22}}$, $\boldsymbol{M}_{r_{21}}$, $\boldsymbol{M}_{r_{12}}$, and $\boldsymbol{M}_{r_{11}}$, cuyas dimensiones son de $8\ x\ 8$ respectivamente.

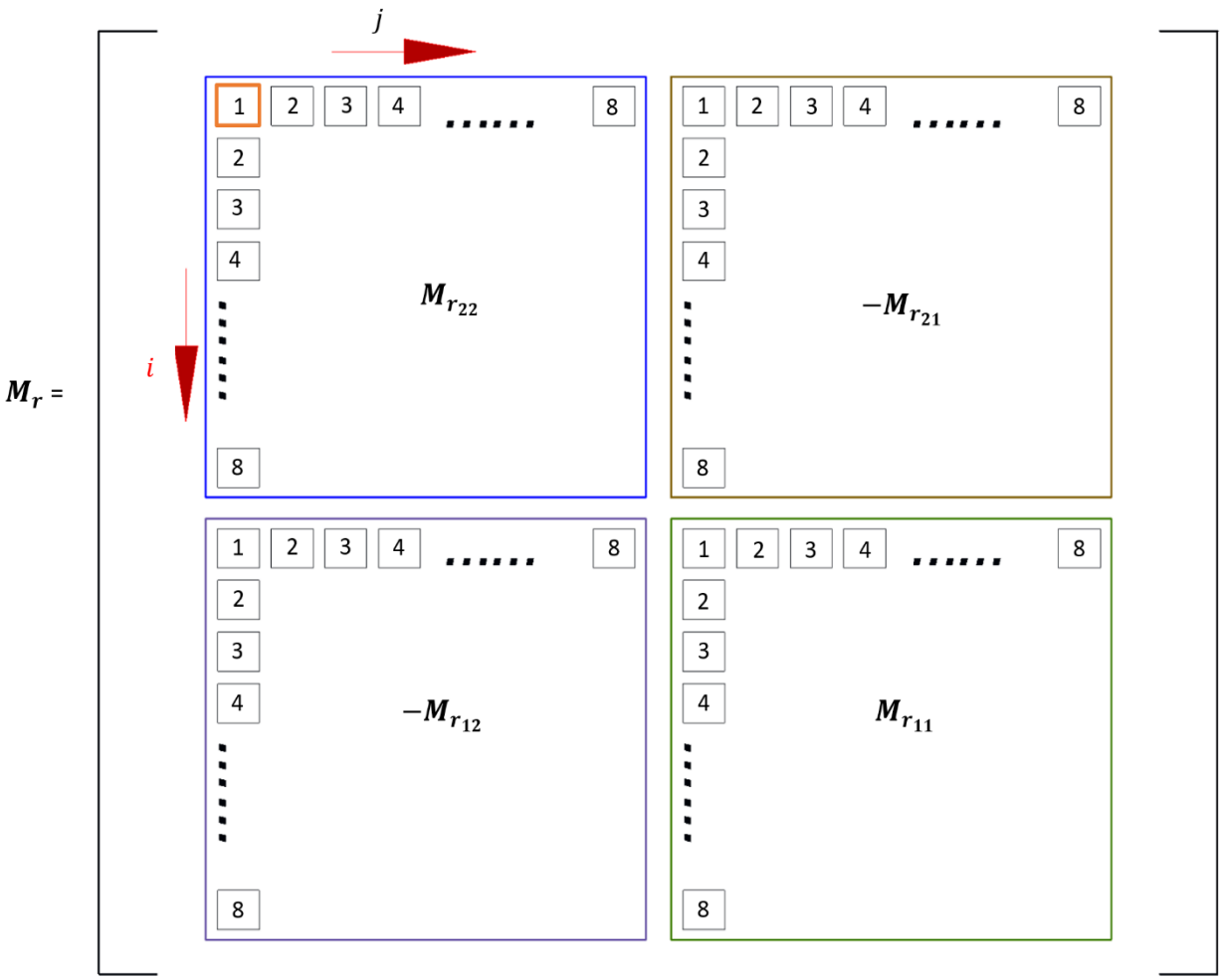


Figura 3.9.- Estructura de la matriz de vorticidad $\boldsymbol{M}_r$ por elemento finito.

3.7 FUNCIÓN BI-OBJETIVO

Una vez definido el modelo de material en la (Ecc. 3.2), las funciones objetivos de la energía de disipación viscosa (Ecc. 3.4) y la vorticidad (Ecc. 3.6), que son de importancia para el diseño del rotor, se combina estas dos ecuaciones con el fin de obtener una función bi-objetivo $c(\gamma_E)$, basada en el método de la suma ponderada. Cabe destacar que estas funciones objetivos del problema MOT imperativamente tienen que ser compatibles con la formulación MEF planteada, usando las variables de respuestas como parte de la formulación para la obtención de estas funciones que son evaluadas para cada elemento finito y cuya sumatoria represente el escalar a optimizar. Adicionalmente se destaca que la energía transmitida al fluido dada por la potencia mecánica que transmite el eje del rotor no se considera como función objetivo en este trabajo de investigación ya que esta función tiene relación directa con la disipación de energía; y, por lo tanto, las topologías generadas para la minimización de la disipación de energía son muy similares a las optimizadas por la potencia mecánica [92]. Esta relación directa de la energía de disipación viscosa con la potencia mecánica resulta debido a que la definición de la potencia está relacionada en términos con la energía interna del fluido.

Por lo tanto, la función bi-objetivo definida en esta sección para el estudio de los diferentes tipos de topologías que se pueden obtener del rotor se expresa de la siguiente manera:

$$c(\gamma_E) = w_d c_d(\gamma_E) + w_r H_{dr} c_r(\gamma_E) \quad \text{(Ecc. 3.9)}$$

Donde:

c → función bi-objetivo, escalar.

c_d → disipación de energía, escalar.

c_r → vorticidad, escalar.

w_d → coeficiente de ponderación asociado con el término de disipación de energía, escalar.

w_r → coeficiente de ponderación asociado con el término de vorticidad, escalar

H_{dr} → factor de normalización para la vorticidad.

Es así, como la función bi-objetivo de suma ponderada queda definida en la (Ecc. 3.9), y que la función de energía de disipación es el término principal en esta función, dando el formato principal a la topología, mientras que el término de vorticidad cambia esencialmente el contorno.

Además, debemos notar que se introdujo un factor de normalización de vorticidad H_{dr} en esta función bi-objetivo, que es un coeficiente que cambia en cada iteración n del algoritmo de optimización debido a las diferentes órdenes de magnitud entre la energía de disipación y la vorticidad. Este coeficiente de normalización utiliza la relación entre los valores de cada función objetivo de la iteración anterior $n-1$, así como se detalla en la siguiente expresión:

$$(H_{dr})_n = \frac{(c_d)_{n-1}}{(c_r)_{n-1}} \qquad \text{(Ecc. 3.10)}$$

Adicionalmente, cabe destacar que estos coeficientes deben cumplir con la suma $w_d + w_r = 1$ y es necesario la exploración de estos coeficientes de ponderación para determinar las diferentes relevancias que tendrían estas funciones objetivos sobre las topologías finales.

3.8 PLANTEAMIENTO Y TÉCNICAS DE SOLUCIÓN DEL PROBLEMA DE OPTIMIZACIÓN

Una vez definido la ecuación bi-objetivo en la (Ecc. 3.9), se procede en esta sección a plantear el problema de optimización de la función bi-objetivo sujeto a la restricción lineal de volumen en su forma continua $(\text{Ecc. } 3.11)$ y en su forma discreta $(\text{Ecc. } 3.12)$. El objetivo final del planteamiento del problema de optimización es encontrar las variables de diseño γ_E para cada elemento finito, de tal manera que se minimice la función bi-objetivo $c(\gamma_E)$ (disipación de energía y vorticidad) en todo el dominio de diseño sujeto a una restricción de volumen que debe ser inferior o igual a un volumen objetivo V^*. Esta variable de diseño por elemento puede tomar valores en un rango de $\gamma_E \ \epsilon \ [0,1]$ como solución óptima para la minimización de nuestra función bi-objetivo, permitiéndose valores intermedios mediante la modelación de un flujo poroso. Cabe destacar que las variables de estado ($ur1_j$, $ur2_j$ y p) siguen las ecuaciones MEF (Ecc. 2.24), (Ecc. 2.25) y (Ecc. 2.26), cuyas matrices formadas por la (Ecc. 2.27) por elemento, tienen dependencia por la variable de diseño regida en la (Ecc. 3.2).

$$\min_{\gamma_E} c(\gamma_E)$$

$$Sujeto\ a: \begin{cases} \int_{\Omega_s} \alpha d\Omega - V^* \leq 0 \\ 0 \leq \gamma_E \leq 1 \\ \text{Ec. de equilibrio MEF} \end{cases} \qquad \text{(Ecc. 3.11)}$$

$$\min_{\gamma_E} c(\gamma_E)$$

$$Sujeto\ a: \begin{cases} \sum_{E=1}^{nel} \gamma_E V_E - V^* \leq 0 \\ 0 \leq \gamma_E \leq 1 \\ \text{Ec. de equilibrio MEF} \end{cases} \qquad \text{(Ecc. 3.12)}$$

Cabe destacar que el problema de optimización definido en las $(\text{Ecc. } 3.11)$ y $(\text{Ecc. } 3.12)$ es un problema de optimización no lineal (debido a la función objetivo c). Para resolver este tipo de problemas, existen diferentes metodologías que requieren de algoritmos computacionales eficientes para lograr su solución [95]. Estas metodologías que resuelven estos tipos de problemas no lineales comprenden a los métodos analíticos o métodos numéricos, que pueden

ser probabilísticos o determinísticos. En la Figura 3.10, se presenta un diagrama con algunas técnicas existentes que pueden ser aplicadas para darle solución al problema de optimización planteado.

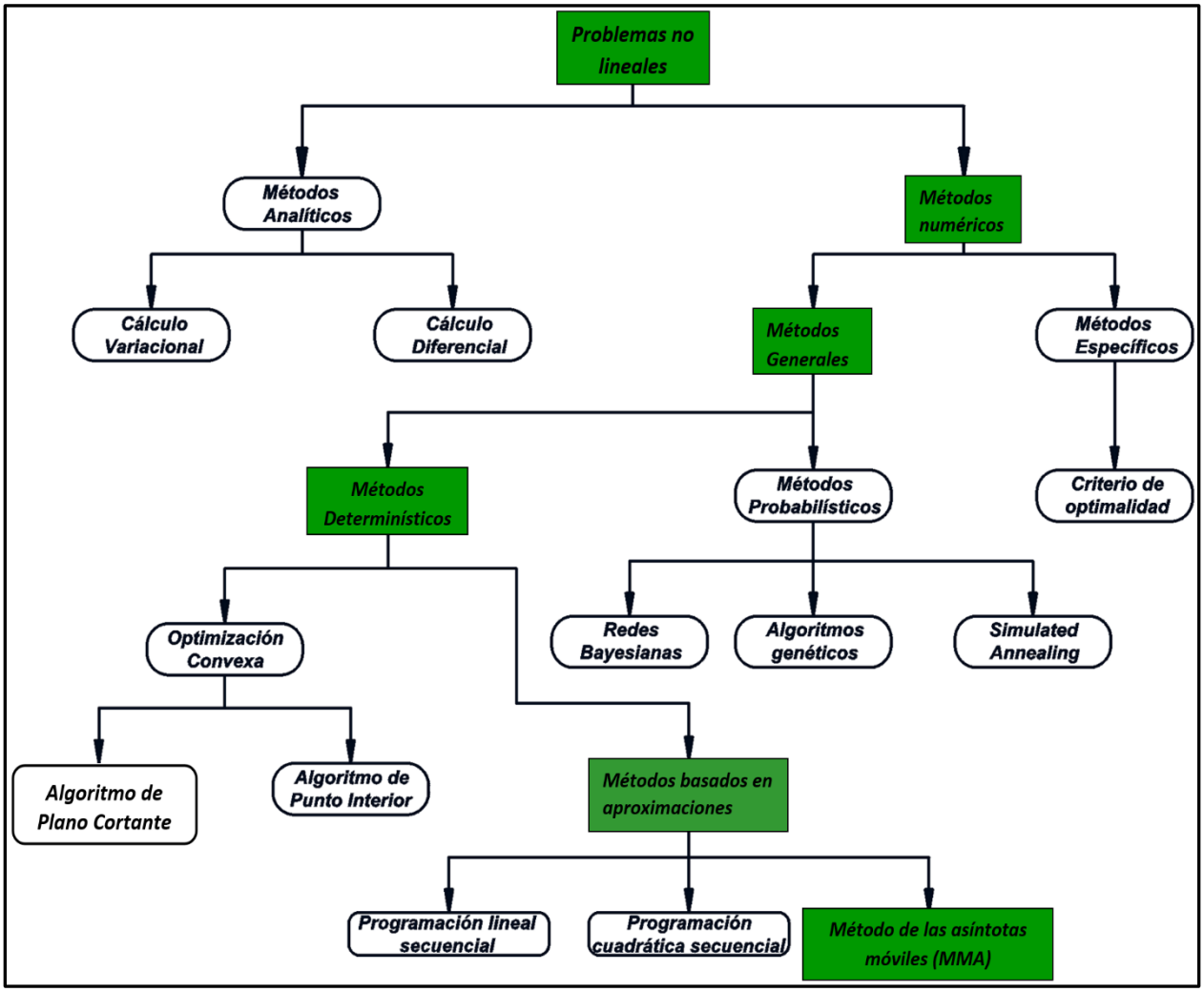


Figura 3.10.- Métodos de solución al problema de optimización planteado [78].

En este diagrama cabe destacar que los métodos numéricos admiten trabajar con problemas de optimización más generales y grandes en términos del número de variables de diseño a diferencia de los métodos analíticos que admiten la solución a problemas simples de optimización [78]. Una opción eficiente en términos de costo computacional para encontrar la solución de las variables que optimicen el problema son los métodos específicos como el criterio de optimalidad, pero son pocos versátiles ya que requieren de una formulación específica para cada problema [95].

En este trabajo se ha escogido al MMA (Method of moving asymptotes) como método de solución al problema de optimización ya que es el más usado para problemas de optimización en flujos porosos con no linealidades [42].

3.9 MÉTODO DE LAS ASÍNTOTAS MÓVILES

El método de las asíntotas móviles (MMA por sus siglas en inglés) opera sobre el problema de optimización topológica planteado en la $(\text{Ecc. } 3.12)$ donde el proceder inicial está constituido por la transformación del mismo en un problema convexo regular equivalente, donde el hallazgo de la solución óptima se garantiza bajo el cumplimiento de ciertas restricciones de optimalidad (condiciones de Kuhn Tucker, KKT) [96].

Para explicar la metodología del MMA en su formulación genérica, cabe destacar que este método se basa específicamente en la transformación de un problema con función no lineal sujeta a restricciones, a un problema convexo [2] con la adición de variables dependientes (y, z) y parámetros artificiales (a, c, d) tal como se muestra en la Figura 3.11.

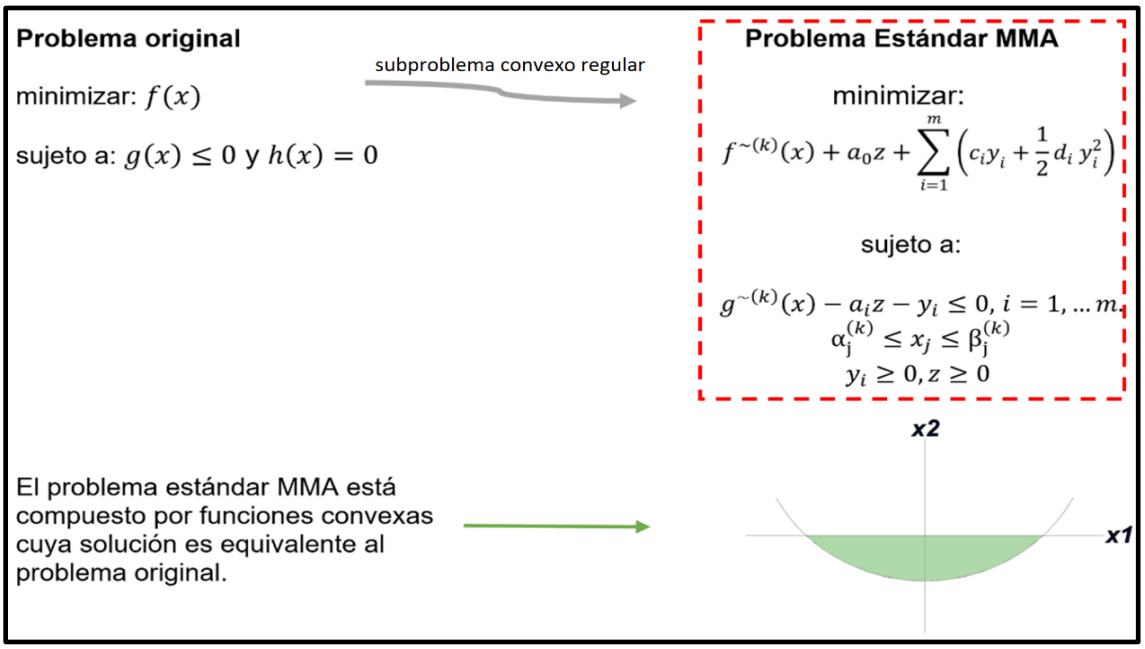


Figura 3.11.- Esquema metodológico del MMA.

Entonces el MMA es un método para resolver problemas cuya formulación genérica se encuentra descrita en el recuadro rojo. Cabe destacar que la función inicial $f(x)$ fue reemplazada por una función convexa $f^{\sim(k)}(x)$. La elección de estas funciones de aproximación se basa principalmente en información de gradiente en el punto de iteración actual, pero también en algunos parámetros $u_j^{(k)}$ y $l_j^{(k)}$ (asíntotas móviles) que se actualizan en cada iteración en función de la información de la iteración anterior [96].

[2] *Problema cuya función objetivo como las funciones que describen las restricciones de desigualdad son convexas.*

Una vez planteado el subproblema convexo regular (Figura 3.11), las condiciones de optimalidad KKT son necesarias y suficientes para una solución óptima. Cabe destacar que, para enunciar estas condiciones, primero se deja expresada la función de Lagrange (Ecc. 3.13) correspondiente a la formulación descrita en el recuadro en rojo [96]:

$$L = f^{\sim(k)}(x) + a_0 z + \sum_{i=1}^{m}\left(c_i y_i + \frac{1}{2} d_i y_i^2\right) + \sum_{i=1}^{m} \lambda_i (g_i(x) - a_i z - y_i - b_i) + \sum_{j=1}^{n}\left(\xi_j(\alpha_j - x_j) + \eta_j(x_j - \beta_j)\right) - \sum_{i=1}^{m} \mu_i y_i - \zeta z$$

(Ecc. 3.13)

donde $\lambda = (\lambda_1, \dots., \lambda_m)^T$, $\xi = (\xi_1, \dots., \xi_m)^T$, $\eta = (\eta_1, \dots., \eta_n)^T$, $\mu = (\mu_1, \dots., \mu_n)^T$ y ζ son multiplicadores de lagrange no negativos para las diferentes restricciones de la formulación del recuadro en rojo [96].

Una vez expresada la función de Lagrange (Ecc. 3.13), las condiciones de optimalidad son enunciadas, las cuales están constituidas por las derivadas parciales respecto a la variable de diseño x (Ecc.3.14) y a las variables artificiales (y, z) (Ecc.3.15) y (Ecc. 3.16), restricción primaria (Ecc. 3.17), restricciones artificiales de holgura (Ecc.3.18) – (Ecc. 3.22) y los requerimientos inherentes al problema base (Ecc. 3.23) - (Ecc. 3.24) y el problema dual (problema transformado) (Ecc. 3.25) - (Ecc. 3.27).

$$\frac{\partial \psi}{\partial x_j} - \xi_j + \eta_j = 0, \ \ j = 1, \dots, n \ \ (\partial L/\partial x_j = 0) \qquad \text{(Ecc.3.14)}$$

$$c_i + d_i y_i - \lambda_i - \mu_i = 0, i = 1, \dots, m \ (\partial L/\partial y_i = 0) \qquad \text{(Ecc.3.15)}$$

$$a_0 - \zeta - \lambda^T a = 0, (\partial L/\partial z = 0) \qquad \text{(Ecc. 3.16)}$$

$$g_i(x) - a_i z - y_i - b_i \leq 0, i = 1, \dots, m \ \ (restricción\ primaria) \qquad \text{(Ecc. 3.17)}$$

$$\lambda_i (g_i(x) - a_i z - y_i - b_i) = 0, i = 1, \dots m \ (holgura\ complementaria) \qquad \text{(Ecc.3.18)}$$

$$\xi_j(\alpha_j - x_j) = 0, j = 1, \dots, \eta \ (holgura\ complementaria) \qquad \text{(Ecc. 3.19)}$$

$$\eta_j(x_j - \beta_j) = 0, j = 1, \dots, \eta \ (holgura\ complementaria) \qquad \text{(Ecc. 3.20)}$$

$$-\mu_i y_i = 0, i = 1, \dots., m \ (holgura\ complementaria) \qquad \text{(Ecc. 3.21)}$$

$$-\zeta z = 0, (holgura\ complementaria) \qquad \text{(Ecc. 3.22)}$$

$$\alpha_j \leq x_j \leq \beta_j, j = 1, \dots, n \ (restricción\ primaria) \qquad \text{(Ecc. 3.23)}$$

$$-z \leq 0 \; y \; -y_i \leq 0, i = 1, \dots, m \; (restricción \; primaria) \quad \text{(Ecc. 3.24)}$$

$$\xi_j \geq 0 \; y \; \eta_j \geq 0, \;\; j = 1, \dots., n \; (restricción \; dual) \quad \text{(Ecc. 3.25)}$$

$$\zeta \geq 0 \; y \; \mu_i \geq 0, \;\; i = 1, \dots., m \; (restricción \; dual) \quad \text{(Ecc. 3.26)}$$

$$\lambda_i \geq 0, i = 1, \dots., m \; (restricción \; dual) \quad \text{(Ecc. 3.27)}$$

La solución del problema de optimización acoplado al conjunto de restricciones que garantiza el hallazgo del punto óptimo se ejecuta aplicando el método de Newton a las (Ecc.3.14) - (Ecc. 3.22), cuya solución obtenida indica la dirección en la cual nos debemos mover y por medio de recomendaciones específicas dadas por Svanberg (2007) [96] seleccionamos el tamaño del paso a ejecutar sobre la dirección dada (ver Figura 3.12).

Figura 3.12.- Esquema explicativo dirección de Newton.

El procedimiento algorítmico descrito en la Figura 3.12, se aplica de manera directa sobre el problema de optimización planteado en $(\text{Ecc. } 3.12)$, donde las variables de diseño son actualizadas en cada iteración. La función objetivo base corresponde a la energía de disipación y vorticidad, sujeto a la restricción lineal de volumen que limita el porcentaje que puede ser ocupado por fluido en el dominio de diseño. Para más detalles de la metodología del MMA, esta puede ser consultada en Svanberg (1987,2002).

Este algoritmo iterativo se implementa mediante el método de optimización topológica basada en gradientes. El procedimiento para la optimización iterativa incluye los siguientes pasos: **a)** se parte de un dominio de diseño fijo, con unas condiciones de frontera determinadas y con una distribución de material inicial (variable γ_E inicial en todo el dominio), **b)** luego se aplica el filtro espacial de pesos promediados que se detalla en la sección 3.11.2, **c)** luego las ecuaciones de Navier-Stokes se resuelven considerando la variable de diseño inicial, donde se obtienen las variables de estado $\boldsymbol{z} = \left[u_{r1_j} \; p_l \; u_{r2_j}\right]^T$, **d)** se calculan las sensibilidades de la función objetivo y

las restricciones, y **d)** finalmente se actualiza la variable de diseño por el método de las asíntotas móviles (MMA). Estos pasos se implementan de forma iterativa hasta que no se logra encontrar una mejora significativa en la función objetivo sin violar la restricción de volumen, momento en el cual se dice que se alcanza la convergencia y, por lo tanto, dicho resultado corresponderá a la topología final como solución al problema optimización planteado.

3.10 ANÁLISIS DE SENSIBILIDAD

El análisis de sensibilidad consiste en la evaluación de los gradientes de las funciones objetivos con el fin de proporcionar información sobre las funciones y las restricciones para guiar la optimización. Estas derivadas de las funciones objetivos se realizan respecto a la variable de diseño gamma γ_E, es decir, con la variable gamma de un elemento con el fin de evaluar el cambio parcial de nuestra función objetivo en todo el dominio de estudio, y así, proporcionar al algoritmo MOT el camino más rápido para encontrar el mínimo de nuestra función en el dominio de diseño.

Dado que la función bi-objetivo (Ecc. 3.9) proviene de una suma ponderada, la ecuación para el análisis de sensibilidad que involucra la vorticidad y energía de disipación viene dada también de una suma pondera y se expresa de la siguiente manera:

$$\frac{d_c(\gamma_E)}{d\gamma_E} = w_d \frac{d_{cd}(\gamma_E)}{d\gamma_E} + w_r H_{dr} \frac{d_{cr}(\gamma_E)}{d\gamma_E} \qquad \text{(Ecc. 3.28)}$$

Esta interpretación del análisis de sensibilidad se la puede ejecutar ya sea sobre ecuaciones continuas (modelo matemático) o sobre las ecuaciones discretas del MEF (método numérico), siendo la metodología mayormente aplicada para el caso de las ecuaciones discretas debido a que en los tipos de problema de optimización se trabaja con un sin número de variables de diseño. En este trabajo las sensibilidades fueron implementadas sobre las ecuaciones discretas del MEF, calculadas por el método adjunto (Haftka & Gurdal 1992) [78].

Cabe mencionar que adicionalmente se realiza la verificación numérica del análisis de sensibilidad del método adjunto versus el método diferencias finitas hacia adelante tanto para la energía de disipación viscosa (ver Tabla 5.2) y para la vorticidad (ver Tabla 5.3), dando errores relativos no mayores a 6%. El paso que se usa en este proyecto de investigación en el método por diferencias finitas hacia adelante es de $\Delta\gamma_E = 1\,x10^{-4}$.

Este análisis de sensibilidades es de importancia para el método de las asíntotas móviles (MMA) debido a que este método requiere del valor de las derivadas de todos los elementos finitos $\frac{d_c(\gamma_E)}{d\gamma_E}$ con el objetivo de minimizar la función bi-objetivo $c(\gamma_E)$ en todo el dominio de diseño. Para un mayor entendimiento, en la Figura 3.13 se muestra el algoritmo de optimización topológica (MOT) implementado en este trabajo de investigación.

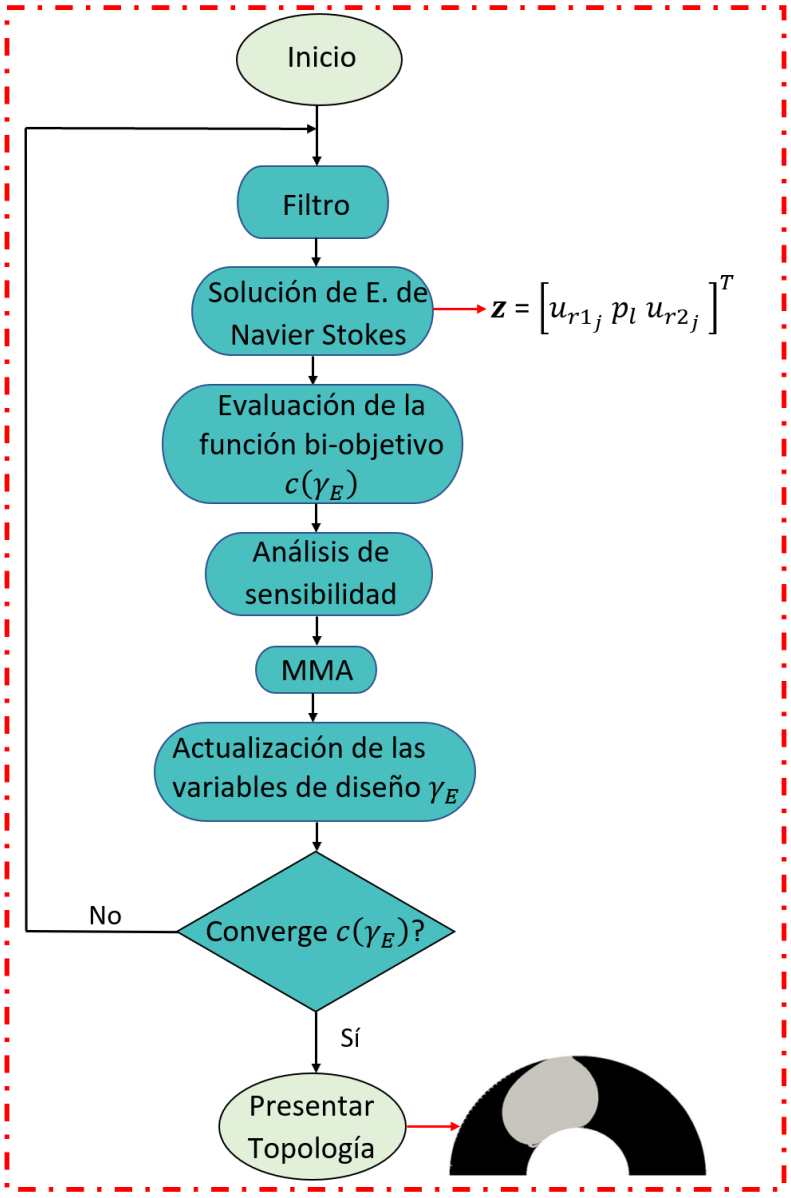


Figura 3.13.- Algoritmo de solución de optimización topológica.

3.10.1 FORMULACIÓN GENÉRICA DEL MÉTODO ADJUNTO

Una vez expresada el análisis de sensibilidad de nuestra función bi-objetivo (Ecc. 3.28), en esta sección se deja expresado la formulación genérica del método adjunto para la posterior solución individual de las sensibilidades de la energía de disipación y vorticidad respectivamente.

Esta formulación genérica viene expresada de la siguiente manera [78]:

$$\frac{d_c(\boldsymbol{u_r}(\boldsymbol{\gamma}))}{d\boldsymbol{\gamma}} = \frac{\partial c}{\partial \boldsymbol{\gamma}} - \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\gamma}} \qquad \text{(Ecc. 3.29)}$$

$$\frac{\partial \boldsymbol{R}^T}{\partial \boldsymbol{u_r}} \boldsymbol{\lambda} = \frac{\partial c^T}{\partial \boldsymbol{u_r}} \qquad \text{(Ecc. 3.30)}$$

Donde:

$\frac{dc(\boldsymbol{u_r}(\boldsymbol{\gamma}))}{d\boldsymbol{\gamma}}$ → vector de derivadas totales.

$\frac{\partial c}{\partial \boldsymbol{\gamma}}$ → vector de derivadas explícitas.

$\boldsymbol{\gamma}$ → vector de variables de diseño.

$\boldsymbol{u_r}$ → vector de variables de estado.

$\boldsymbol{\lambda}$ → vector adjunto.

$\boldsymbol{R}$ → vector residual de las ecuaciones MEF.

Se puede observar que para la ecuación genérica (Ecc. 3.30), el vector adjunto es la solución del sistema de ecuaciones formadas por las derivadas implícitas (respecto a la variable de estado $\boldsymbol{u_r}$) del residual $\boldsymbol{R}$ de las restricciones de igualdad (ecuaciones MEF igualadas a cero) y la función objetivo c. Cabe destacar que cuando existen no linealidades en el residual (término advectivo en el caso específico de nuestras ecuaciones de Navier-Stokes), la derivada parcial $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u_r}}$ no se calcula de manera directa, sino que representa el jacobiano $\boldsymbol{J}$ de las ecuaciones MEF. Esta exigencia adquiere gran importancia cuando el método de solución no lineal involucra alguna versión del método de Newton (linealización de Picard en nuestras ecuaciones de Navier-Stokes), puesto que el cálculo de $\boldsymbol{J}$ está incorporado dentro de dicho algoritmo, por lo que no se debe implementar un algoritmo adicional para el cálculo de $\boldsymbol{J}$.

3.10.2 ANÁLISIS DE SENSIBILIDAD ENERGIA DE DISIPACIÓN

De la función objetivo de la energía de disipación el único término que depende de manera explícita de la variable de diseño es el asociado a la fuerza de penalización de Brinkman. Por lo tanto, las sensibilidades siguen las (Ecc. 3.31) y (Ecc. 3.32). En este punto se destaca que la función de minimización de energía de disipación es expandida al incluir la variable de diseño de estado de la presión, mediante $\boldsymbol{z} = \left[u_{r1_j}\ p_l\ u_{r2_j}\right]^T$ para lograr la compatibilidad con la ecuación resultante del residual.

$$\frac{dc_d(\gamma_E)}{d\gamma_E} = \frac{1}{2}\boldsymbol{z}^T \frac{\partial\left(\boldsymbol{k}^E(\gamma_E)\right)}{\partial \gamma_E}\boldsymbol{z} - \boldsymbol{\lambda}_d^T \frac{\partial \boldsymbol{R}}{\partial \gamma_E} \qquad \text{(Ecc. 3.31)}$$

$$\boldsymbol{J}\boldsymbol{\lambda}_d = [(\boldsymbol{k}^E(\gamma_E)\boldsymbol{z})]^T \qquad \text{(Ecc. 3.32)}$$

Dejando la derivada $\frac{\partial\left(\boldsymbol{k}^{E}(\gamma_E)\right)}{\partial\gamma_E}$ de forma explícita para cualquier elemento en el que estemos parados se tiene que:

Para los residuales R_1 y R_3 la derivada de forma explícita es:

$$\frac{\partial \boldsymbol{k}^{E}(\gamma_E)}{\partial\gamma_E} = \iint_{E}\left[N_i\left(\left((\alpha_{min}-\alpha_{max})\frac{(1+q)q}{(\gamma_E+q)^2}\right)\right)\right]dA \qquad \text{(Ecc. 3.33)}$$

$$i = 1,2,\dots 8.$$

Para el residual R_2 la derivada de forma explícita es:

$$\frac{\partial \boldsymbol{k}^{E}(\gamma_E)}{\partial\gamma_E} = 0 \qquad \text{(Ecc. 3.34)}$$

Dejando expresado los términos del vector $\frac{\partial \boldsymbol{R}}{\partial\gamma_E}$ explícitamente se tiene:

Para el residual R_1:

$$\frac{\partial R_1}{\partial\gamma_E} = \iint_{E}\left[N_i\left(\left((\alpha_{min}-\alpha_{max})\frac{(1+q)q}{(\gamma_E+q)^2}\right)*u_{r1_j}\right)\right]dA \qquad \text{(Ecc. 3.35)}$$

$$i = 1,2,\dots 8. \quad j = 1,2,\dots 8.$$

Para el residual R_2:

$$\frac{\partial R_2}{\partial\gamma_E} = 0 \qquad \text{(Ecc. 3.36)}$$

Para el residual R_3:

$$\frac{\partial R_3}{\partial\gamma_E} = \iint_{E}\left[N_i\left(\left((\alpha_{min}-\alpha_{max})\frac{(1+q)q}{(\gamma_E+q)^2}\right)*u_{r2_j}\right)\right]dA \qquad \text{(Ecc. 3.37)}$$

$$i = 1,2,\dots 8. \quad j = 1,2,\dots 8.$$

Para el cálculo del Jacobiano $\boldsymbol{J}$ se siguen las ecuaciones explícitas presentadas por Endalew [74], las cuales tienen una similitud directa con las (Ecc. 2.24), (Ecc. 2.25) y (Ecc. 2.26), con la única diferencia que los términos adicionales surgen debido a la derivada del término advectivo (no lineal) (Ecc. 3.38).

$$\boldsymbol{J} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u_r}} = \begin{bmatrix} \boldsymbol{J}_{11} & \boldsymbol{J}_{12} & \boldsymbol{J}_{13} \\ \boldsymbol{J}_{21} & \boldsymbol{J}_{22} & \boldsymbol{J}_{23} \\ \boldsymbol{J}_{31} & \boldsymbol{J}_{32} & \boldsymbol{J}_{33} \end{bmatrix} \qquad \text{(Ecc. 3.38)}$$

Donde:

$$\boldsymbol{J}_{11} = \frac{\partial R_1}{\partial u_{r1_j}} = \iint_E \left\{ N_i \rho \left[\left(N_j \frac{\partial N_j}{\partial \mathrm{x}_1} \left(2u_{r1_j} \right) + N_j \overbrace{\sum_{k=1}^{8}}^{j \neq k} u_{r1_k} \frac{\partial N_k}{\partial \mathrm{x}_1} \right) + \left(\frac{\partial N_j}{\partial \mathrm{x}_1} \overbrace{\sum_{k=1}^{8}}^{j \neq k} u_{r1_k} N_k \right) \right] + N_i \rho \left[\frac{\partial N_j}{\partial \mathrm{x}_1} \sum_{k=1}^{8} u_{r2_k} N_k \right] + \mu \left(\frac{\partial N_i}{\partial \mathrm{x}_1} \frac{\partial N_j}{\partial \mathrm{x}_1} + \frac{\partial N_i}{\partial \mathrm{x}_2} \frac{\partial N_j}{\partial \mathrm{x}_2} \right) + N_i \alpha \left(\gamma_E \right) \right\} dA$$

$i = 1,2 \ldots 8.$ (i representa la función de prueba que estemos evaluando).
$j = 1,2 \ldots 8.$ (j representa el grado de libertad de la velocidad que estemos evaluando).

$$\boldsymbol{J}_{12} = \frac{\partial R_1}{\partial p_l} = \iint_E \left\{ N_i \frac{\partial M_l}{\partial \mathrm{x}_1} \right\} dA$$

$i = 1,2 \ldots .8.$ (i representa la función de prueba que estemos evaluando).
$l = 1,2 \ldots 4.$ (j representa el grado de libertad de la presión que estemos evaluando).

$$\boldsymbol{J}_{13} = \frac{\partial R_1}{\partial u_{r2_j}} = \iint_E N_i \left[\rho N_j \sum_{k=1}^{8} u_{r1_k} \frac{\partial N_k}{\partial \mathrm{x}_2} + 2\rho \Omega N_j \right] dA$$

$$i = 1,2, \ldots 8. \quad j = 1,2, \ldots 8.$$

$$\boldsymbol{J_{21}} = \frac{\partial R_2}{\partial u_{r1_j}} = \iint_E \left\{ M_l \left[\frac{\partial N_j}{\partial \mathrm{x}_1} \right] \right\} \mathrm{dA}$$

$l = 1,2 \ldots .4. \quad j = 1,2, \ldots 8.$

$$\boldsymbol{J_{22}} = \frac{\partial R_2}{\partial p_l} = 0.$$

$$\boldsymbol{J_{23}} = \frac{\partial R_2}{\partial u_{r2_j}} = \iint_E \left\{ M_l \left[\frac{\partial N_j}{\partial \mathrm{x}_2} \right] \right\} \mathrm{dA}$$

$l = 1,2 \ldots .4. . \quad j = 1,2, \ldots 8$

$$\boldsymbol{J_{31}} = \frac{\partial R_3}{\partial u_{r1_j}} = \iint_E \left\{ N_i \rho \left[N_j \sum_{k=1}^{8} u_{r2_k} \frac{\partial N_k}{\partial \mathrm{x}_1} + 2\,\rho \Omega N_j \right] dA \right\}$$

$i = 1,2, \ldots 8. \quad j = 1,2, \ldots 8$

$$\boldsymbol{J_{32}} = \frac{\partial R_3}{\partial p_l} = \iint_E \left\{ N_i \left[\frac{\partial M_l}{\partial \mathrm{x}_2} \right] dA \right\}$$

$i = 1,2 \ldots .8. \quad l = 1,2 \ldots 4.$

$$\boldsymbol{J_{33}} = \frac{\partial R_3}{\partial u_{r2_j}} = \iint_E \left[N_i \rho \left(\frac{\partial N_j}{\partial \mathrm{x}_1} \right) \overbrace{\sum_{k=1}^{8}}^{j \neq k} u_{r1_k} N_k \right] + N_i \rho \left[N_j \left(\frac{\partial N_j}{\partial \mathrm{x}_2} \right) 2 u_{r2_j} + \left(N_j \right) \overbrace{\sum_{k=1}^{8}}^{j \neq k} u_{r2_k} \frac{\partial N_k}{\partial \mathrm{x}_2} \right]$$
$$+ N_i \rho \left[\left(\frac{\partial N_j}{\partial \mathrm{x}_2} \right) \sum_{k=1}^{8} u_{r2_k} \frac{\partial N_k}{\partial \mathrm{x}_2} \right] + \mu \left[\left(\frac{\partial N_i}{\partial \mathrm{x}_1} \right) \left(\frac{\partial N_j}{\partial \mathrm{x}_1} \right) + \left(\frac{\partial N_i}{\partial \mathrm{x}_2} \right) \left(\frac{\partial N_j}{\partial \mathrm{x}_2} \right) \right]$$
$$+ \left[N_j \alpha \left(\gamma_E \right) N_j \right] dA$$

$i = 1,2, \ldots 8. \quad j = 1,2, \ldots 8.$

3.10.3 ANÁLISIS DE SENSIBILIDAD VORTICIDAD

Usando el mismo método de la sección anterior 3.10.2, el gradiente de la función de vorticidad con respecto a la variable de diseño γ_E es también calculado y estas sensibilidades siguen las (Ecc. 3.39) y (Ecc. 3.40):

$$\frac{dc_r(\gamma_E)}{d\gamma_E} = -\boldsymbol{\lambda}_r^T \frac{\partial \boldsymbol{R}}{\partial \gamma_E} \qquad \text{(Ecc. 3.39)}$$

$$\boldsymbol{J}\boldsymbol{\lambda}_r = \left(\boldsymbol{M_r} + \boldsymbol{M_r}^T\right)[\boldsymbol{u_r}, \boldsymbol{0}]^T \qquad \text{(Ecc. 3.40)}$$

Cabe destacar en la (Ecc. 3.39) que la derivada explícita es cero, debido a que únicamente depende de los términos difusivos de la ecuación de Navier-Stokes, los cuales no dependen de la variable de diseño. Adicionalmente, cabe mencionar que para garantizar la compatibilidad del lado izquierdo y el lado derecho en la (Ecc. 3.39), mediante selección directa se realiza el cálculo del vector adjunto, es decir, a manera de ejemplo, para el vector columna $[\boldsymbol{u_r}, \boldsymbol{0}]^{\boldsymbol{T}}$ de un elemento, se escogen las soluciones nodales de las velocidades en x_1 y en x_2, omitiendo así los resultados nodales de las presiones, generando así un vector columna de dimensiones $16x1$ para el vector adjunto. De igual modo la selección directa es aplicada al vector columna $\frac{\partial \boldsymbol{R}}{\partial \gamma_E}$ cuyas ecuaciones fueron detalladas en $(\mathrm{Ecc.\,3.35})$, $(\mathrm{Ecc.\,3.36})$ y $(\mathrm{Ecc.\,3.37})$.

3.11 FILTROS

El objetivo de los filtros en optimización topológica es controlar la complejidad de las topologías obtenidas al encontrar los valores óptimos, ya sea que minimicen o maximicen la función objetivo, dependiendo de la aplicabilidad del problema en sí. Este control de la complejidad de las topologías es atractivo en el enfoque de la manufactura final del producto, porque el costo de producción va a depender directamente de las facilidades que se tengan dependiendo de la topología final obtenida.

Adicional a lo anterior, cabe destacar que este control de complejidad en las topologías se puede llevar a cabo ya sea limitando las variables de diseño o las sensibilidades con funciones que ponderan el espacio de solución [6]. Por lo tanto, los filtros en sí pueden ser entendidos como operadores matemáticos que modifican un parámetro deseado para suavizar su comportamiento con funciones de regularización [78]. Es así, que los filtros pueden ser clasificados en dos categorías [9]: filtros de vecindad y filtros espaciales.

3.11.1 FILTROS DE VENCIDAD FIJA

Es el filtro más simple puesto que solo tiene en cuenta los elementos vecinos más próximos al elemento central que va a ser filtrado, compartiendo nodos y/o líneas. Aunque es el filtro más simple, su definición de vecindad implica dependencia a la discretización de la malla, ya que el área de influencia está determinada por el tamaño de la malla [97]. Cabe destacar que, debido a este problema de dependencia de malla, surgen los filtros espaciales que se describen en la siguiente sección. En la Figura 3.14, se presenta una comparación entre dos refinamientos de mallas empleadas para resolver un mismo problema usando filtros de vecindad fija.

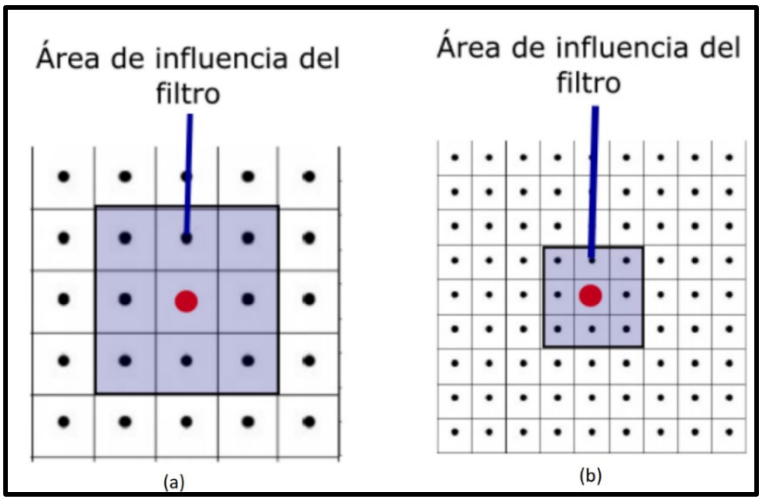


Figura 3.14.- Concepto de filtro de vecindad fija: **(a)** con malla gruesa, **(b)** con malla fina. Imagen tomada de Berrío (2017) [40].

En esta figura se puede destacar como el área de influencia del filtro cambia en función al tamaño de los elementos finitos ya que el filtro solo toma en cuenta cierta cantidad específica de elementos alrededor del elemento que está siendo filtrado; y, en consecuencia, generando así la dependencia de malla en el resultado final, obteniendo adicionalmente topologías más complejas.

3.11.2 FILTROS ESPACIALES

El problema de la dependencia de malla se reduce con la aplicación de los filtros espaciales, ya que controlan el área de influencia a partir de un radio definido (R_{max}), dando lugar al control de la complejidad de la topología [97]. La Figura 3.15, ilustra el concepto de un filtro espacial en una malla bidimensional donde los elementos vecinos son aquellos donde el centroide de estos elementos se encuentra dentro del área de influencia del filtro.

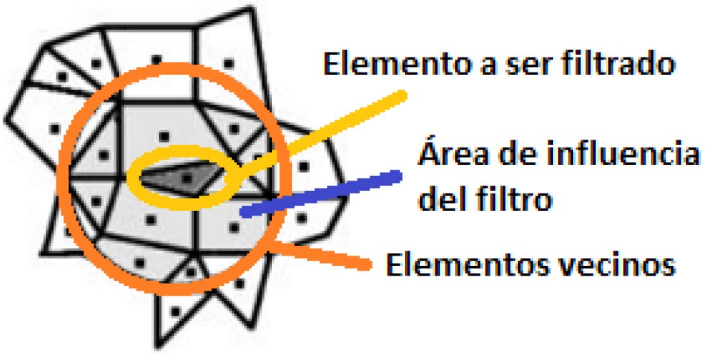


Figura 3.15.- Concepto de filtro espacial en una malla bidimensional. Imagen tomada de Ramírez Gil (2013) [98].

Cabe destacar que con el objetivo de usar un filtro espacial con cualquier tipo de malla (estructurada o no), Cardoso y Fonseca [97] propusieron dos alternativas, usando un peso promedio o individual para cada vecino. La formulación de estos filtros se detalla en la (Ecc.3.41) y (Ecc.3.42), donde se denominan AWSF (Average Weight Spatial Filter) y el filtro IWSF (Individual Weight Spatial Filter), respectivamente.

$$\hat{a}_i = \frac{a_i V_i + \bar{W}_i \sum_{j=1}^{nv} a_j V_j}{V_i + \bar{W}_i \sum_{j=1}^{nv} V_j}, \qquad \bar{W}_i = \sum_{j=1}^{nv} \frac{R_{max} - R_{ij}}{nv * R_{max}} \qquad \text{(Ecc.3.41)}$$

$$\hat{a}_i = \frac{a_i V_i + \sum_{j=1}^{nv} W_{ij} a_j V_j}{V_i + \sum_{j=1}^{nv} W_{ij} V_j}, \qquad W_{ij} = \sum_{j=1}^{nv} \frac{R_{max} - R_{ij}}{R_{max}} \qquad \text{(Ecc.3.42)}$$

Donde a_i es el valor de la variable sin filtro, $\hat{a}$ es la variable filtrada, V_i es el volumen del elemento i, W es un factor de peso, $\bar{W}$ es un valor promedio del factor de peso, R_{max} es el radio del filtro, R_{ij} es la distancia entre el centro del elemento i y el centro del elemento j y nv es el número de elementos vecinos.

Para el filtro AWSF, el peso viene dado por el promedio de la fracción de todos los elementos vecinos que existe entre la diferencia del radio del filtro R_{max} y la distancia entre el centro del elemento i (elemento a ser filtrado) y el centro del elemento j (elemento vecino) respecto al radio del filtro. Los pesos de mayor influencia al elemento filtrado serán los elementos más cercanos a este, reduciendo así la dispersión en la variable de diseño en el radio de filtro seleccionado. Una de las ventajas de este tipo de filtro es que puede ser aplicado a cualquier variable, como la pseudo-densidad o la sensibilidad.

En este trabajo de investigación se usa este tipo de filtro (AWSF), aplicado a la pseudo-densidad o variables de diseño de cada elemento, evitando tener zonas dispersas entre sólido o fluido y poder suavizar la solución del problema de optimización que se verá reflejado en la topología final.

CAPÍTULO 4: COMPUTACIÓN PARALELA EN LA NUBE

4.1 INTRODUCCIÓN

La ingeniería moderna requiere de capacidades computacionales significativas, en diferentes áreas de especialización ya sea en resistencia de materiales, mecánica de sólidos, flujo de fluidos, diseño de productos y mucho más. Estos temas tienen en común que se han vuelto cada vez más exigentes en la demanda de capacidades computacionales a medida que los productos industriales se han vuelto más complejos y/o necesitan mejoras. Estas nuevas capacidades computacionales permiten diseños y simulaciones más adecuada para su aplicación inmediata en un entorno comercial [99].

El análisis de flujo de fluidos es un ejemplo de un tema computacionalmente exigente, más aún en el caso de implementar ecuaciones de Navier-Stokes completas y llegar a una representación del flujo de fluido con características como turbulencia, fase de cambios y comportamientos no newtonianos [99].

Existen otros ejemplos donde es necesario predecir o modelar lo más cerca posible el fenómeno de interés como se ve en la realidad; esos ejemplos incluyen diseños de aviones, válvulas cardíacas, dispositivos protésicos entre muchos otros.

Para este proyecto de investigación, se plantea a la computación paralela en la nube como una solución de plataforma para resolver el problema MOT y MEF aplicado al problema de flujo en bombas centrífugas.

4.2 COMPUTACIÓN PARALELA

Los computadores han ido evolucionando a través de sus inicios, aumentando su capacidad y velocidad de cálculo siguiendo la ley de Moore que predice un crecimiento exponencial en la velocidad de procesamiento, sin embargo, el consumo de energía y disipación de calor pusieron un límite al aumento de la velocidad del reloj (frecuencia) en las CPU (Central Processing Unit por sus siglas en inglés) [100]. Para este problema, una de las soluciones fue la creación de los procesadores multinúcleos (actualmente AMD cuenta con uno de los procesadores más potentes de 64 núcleos en un solo procesador, AMD Ryzen Threadripper 3990X), aumentando el paralelismo ofrecido por estos, manteniendo baja la frecuencia de las mismas. Adicionalmente a estos avances, se han venido dando otros desarrollos tecnológicos, como es el caso de procesadores escalables de 2, 4 y 8 zócalos, empleados para cargas de trabajo exigentes, así como los procesadores Xeon propuestos por Intel. Adicional a estos avances tecnológicos en la CPU, se puede sumar a la computación paralela en la nube, que a diferencia de comprar un clúster local e incurrir en gastos significativos, se puede alquilar por hora, cierta máquina específica con determinadas características de hardware dependiendo de la necesidad del proyecto a ejecutar.

4.2.1 PARALELISMO

El aumento en el rendimiento en serie se ha estancado a medida que los diseños de los procesadores han alcanzado los límites de miniaturización, frecuencia de reloj, potencia y calor. La Figura 4.1, muestra la tendencia de evolución (1970-2018) del número de transistores, frecuencia de reloj, el consumo de energía, el número de procesadores lógicos y el rendimiento del hardware a lo largo del tiempo para los procesadores. Es así que, en el 2005, el número de núcleos comenzó abruptamente a aumentar de un solo núcleo a múltiples núcleos, es a partir de aquí que comienza el conteo de los núcleos en los chips de la CPU (comienzo de la era del paralelismo), mientras que la frecuencia de reloj y el consumo de energía se estabilizaron, sin embargo el rendimiento aumentó constantemente [101].

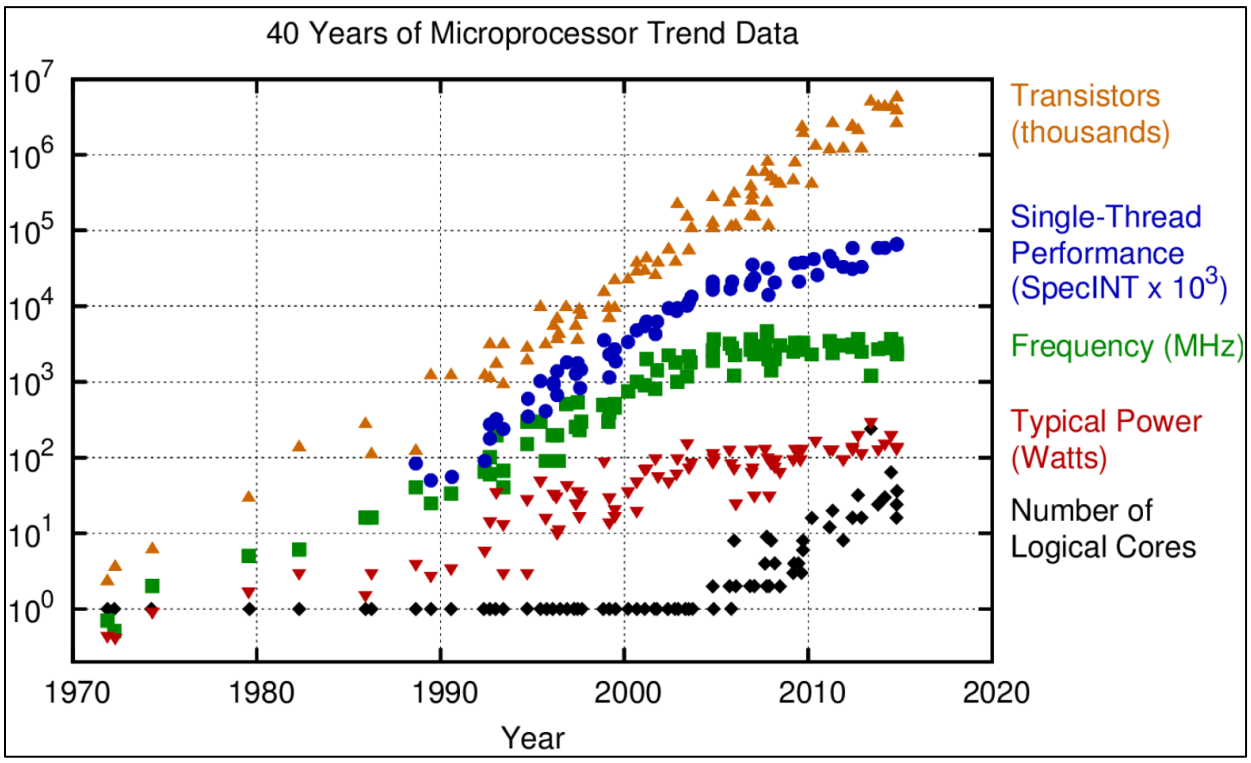
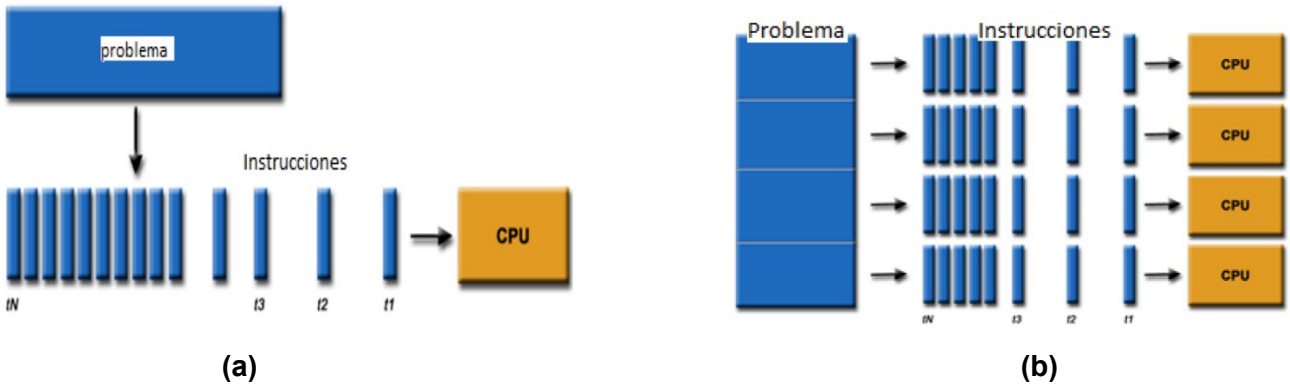


Figura 4.1.- El número de transistores, el rendimiento de un solo hilo, la frecuencia de reloj de la CPU [MHz], el consumo de energía de la CPU [vatios] y el número de procesadores lógicos en los chips de la CPU se muestran desde 1970 hasta 2018 [102].

Este rendimiento teórico aumentó constantemente debido al hecho de que el rendimiento es proporcional al producto de la frecuencia del reloj y al número de núcleos. Este cambio hacia el aumento del conteo de núcleos sobre la velocidad del reloj indica que lograr el rendimiento más ideal de una unidad central de procesamiento (CPU) solo está disponible a través de la computación paralela [102].

4.2.2. TIPO DE PARALELISMOS

La computación paralela es una forma de computo que se ha venido desarrollando de manera continua y que ha sido usada para simular problemas complejos de gran envergadura que se presentan en la ingeniería. Algunas de las razones por las cuales se justifica el uso de computación paralela es el ahorro en tiempo y/o dinero, puesto que, en teoría, entre más recursos se invierta en una tarea, esta tomará menor tiempo para completarse, con un ahorro potencial en costos [103]. La computación paralela es una evolución de la computación serial, en computación serial un problema es dividido en un número infinito de instrucciones y cada instrucción es ejecutada una tras otra en un solo CPU (ver Figura 4.2a). En cambio, la computación paralela es el uso de múltiples recursos computacionales para resolver un problema, el cual puede ser dividido en un sin número de partes y estas partes a su vez se dividen en una serie de instrucciones que son ejecutadas simultáneamente en diferentes procesadores (ver Figura 4.2b).



(a) **(b)**

Figura 4.2.- Modelo de computación: **a)** Serial **b)** Paralela. Adaptado de [103].

Cabe puntualizar que existen diferentes tipos de paralelismo según la taxonomía de Flynn [104]. Cuando hablamos de taxonomía nos referimos al número de instrucciones y flujo de datos en una arquitectura, y, estas instrucciones y datos son clasificados en individuales o único (single) y múltiple (multiple).

Por lo tanto, una máquina puede tener uno o múltiples flujos de datos y puede tener uno o varios procesadores trabajando en datos, por lo que esto nos conlleva a tener cuatro combinaciones de arquitecturas:

- **SISD:** Single Instruction, Single Data (única instrucción, únicos datos)
- **SIMD:** Single instruction, Multiple Data (única instrucción, múltiples datos)
- **MISD:** Multiple instruction, Single Data (múltiples instrucciones, únicos datos)
- **MIMD:** Multiple instruction, Multiple Data (múltiples instrucciones, múltiples datos).

ÚNICA INSTRUCCIÓN (SISD)

Esta arquitectura es la más antigua y común de todas. Posee un único procesador y no contiene paralelismo, solo una instrucción es ejecutada en el CPU y solo un dato es usado en la ejecución.

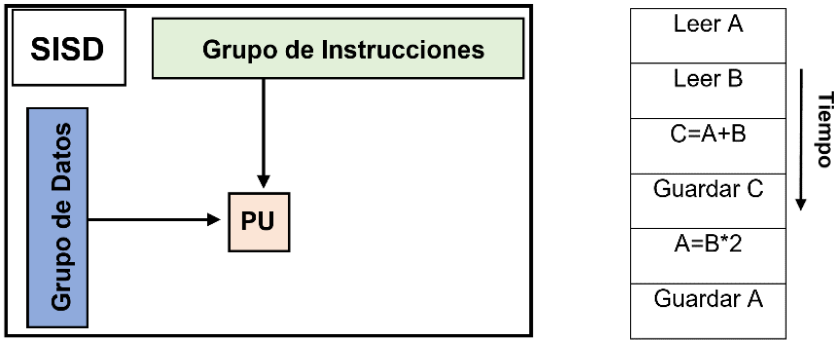


Figura 4.3.- Computadores Tipo SISD.

ÚNICA INSTRUCCIÓN, MÚLTIPLES DATOS (SIMD)

Esta arquitectura es empleada para conseguir paralelismo a nivel de datos. Todas las unidades de procesamiento ejecutan la misma instrucción y cada procesador opera en elementos distintos de los datos. En la Figura 4.4 se muestra este tipo de computador.

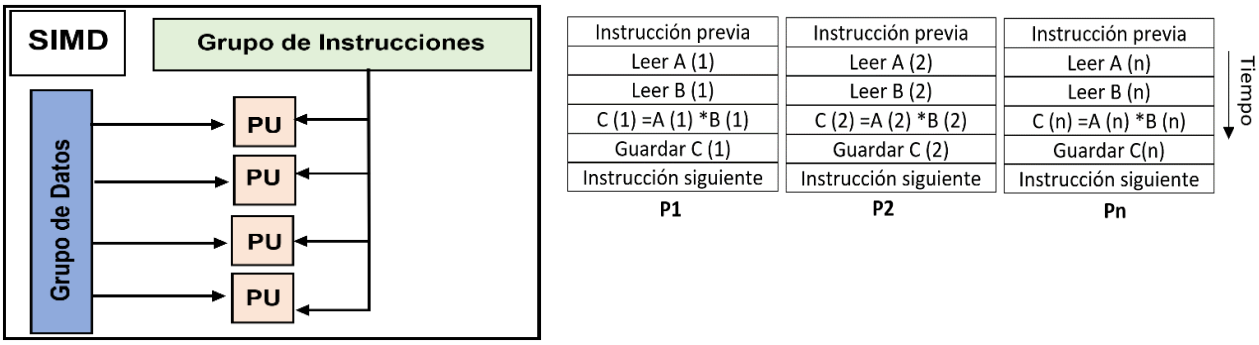


Figura 4.4.- Computadores Tipo SIMD.

MÚLTIPLES INSTRUCCIONES, ÚNICOS DATOS (MISD)

Es una arquitectura que posee múltiples flujos de instrucciones que ejecutan un solo tipo de datos. Actualmente este tipo de arquitectura no se emplea porque es muy cara y poco rentable al usarla. En la Figura 4.5 se muestra este tipo de computador.

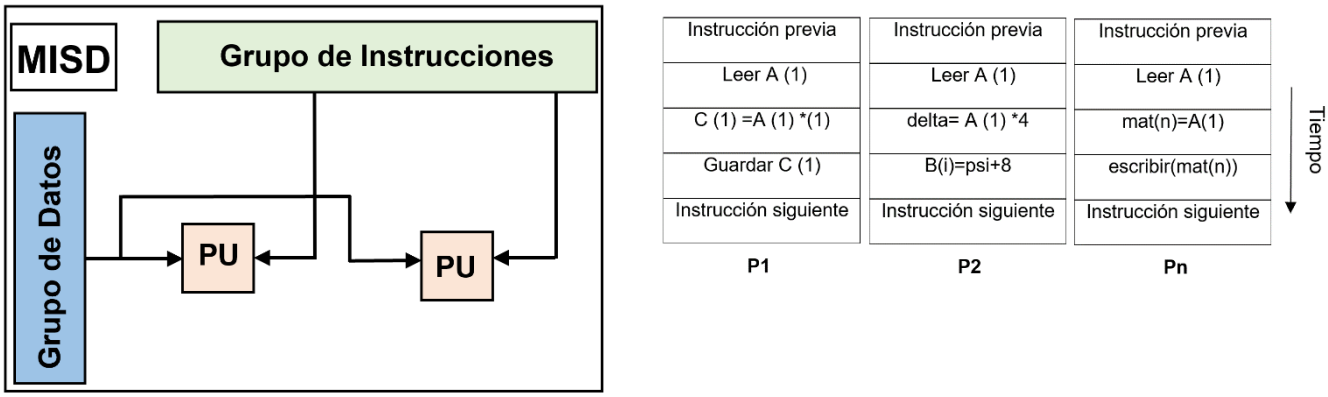


Figura 4.5.- Computadores Tipo MISD.

MÚLTIPLES INSTRUCCIONES, MÚLTIPLES DATOS (MIMD)

Este tipo de arquitectura corresponde a los computadores más modernos y sus características son: cada procesador puede ejecutar una instrucción diferente y cada procesador puede trabajar con datos diferentes. Por ejemplo, computadores tipo clúster y "grids", computadores multiprocesadores y/o multinúcleos [104]. En la Figura 4.6 se muestra este tipo de computador.

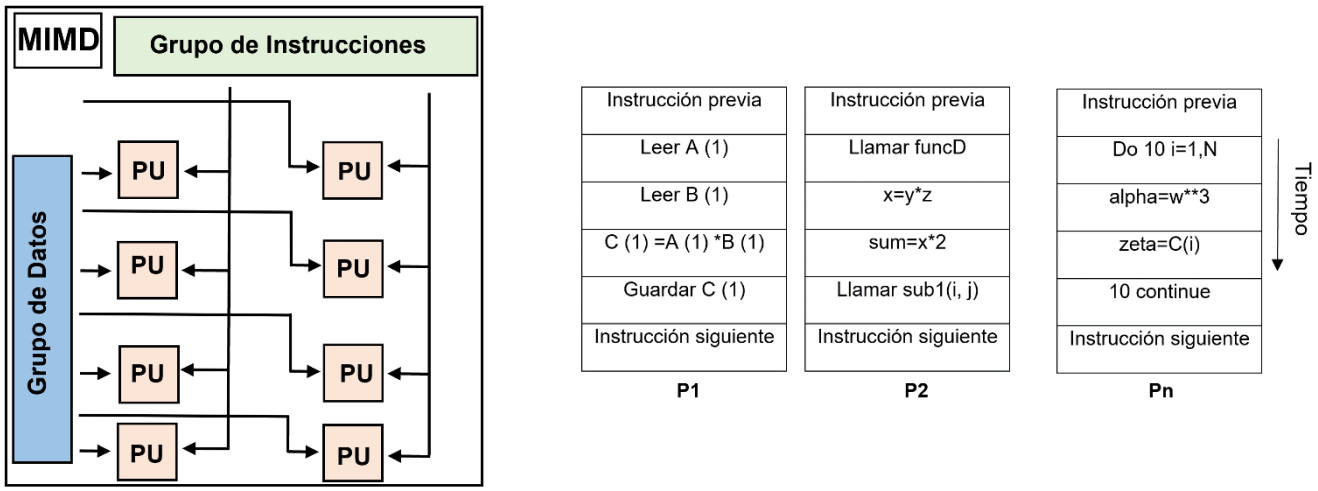


Figura 4.6.- Computadores Tipo MIMD.

4.3 MÉTODOS DE SIMULACIÓN CFD ACELERADOS

Investigadores en la dinámica de fluidos computacionales CFD han desarrollado métodos numéricos, desde décadas atrás, con el fin de formular y solucionar problemas matemáticos relacionados a la física del problema en cuestión. Estos métodos numéricos pueden clasificarse en dos grupos principalmente, como: “métodos convencionales” y “métodos acelerados” tal como se muestra en la Figura 4.7 [105].

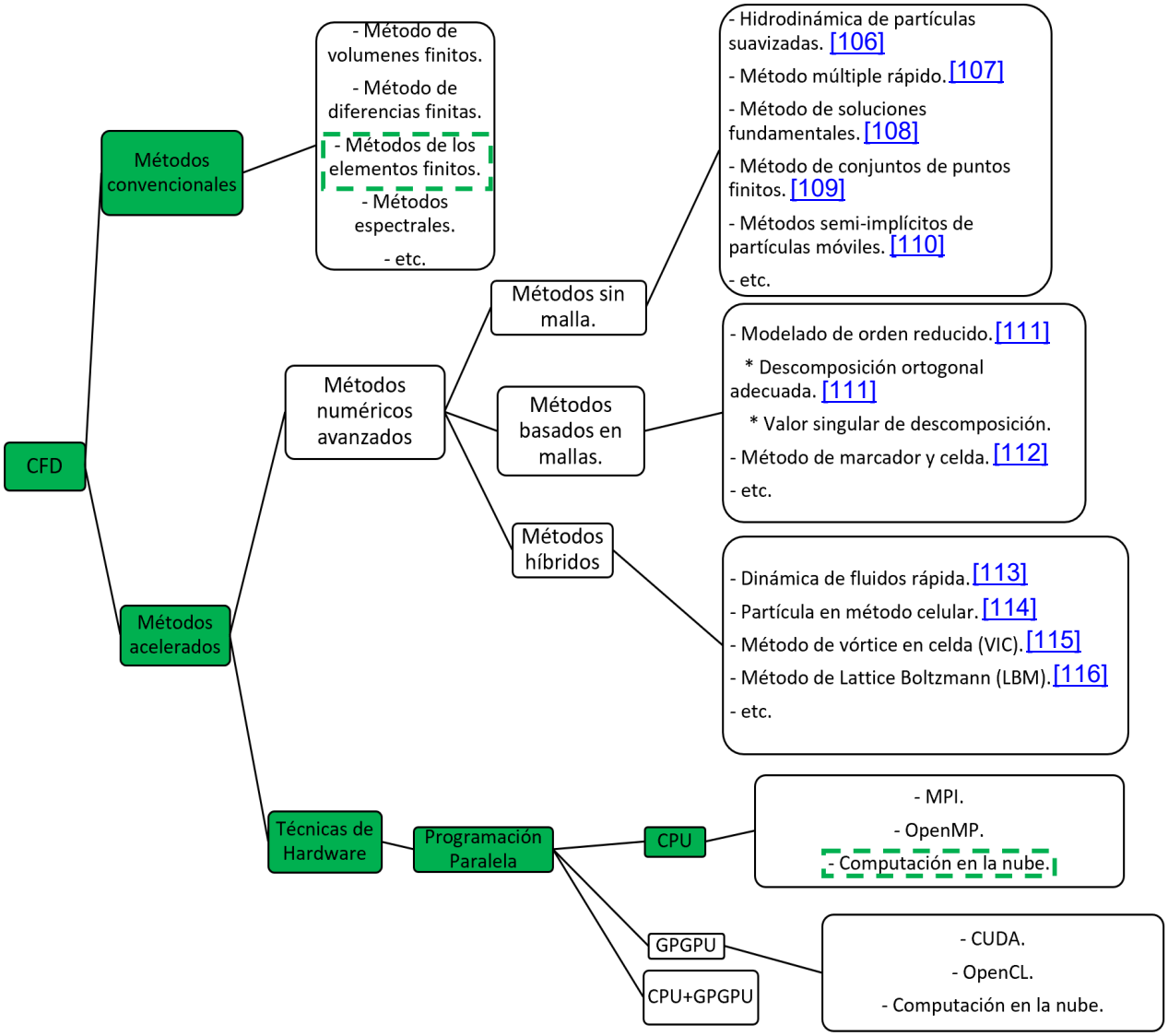


Figura 4.7.- Clasificación jerárquica de varios métodos CFD acelerados. Imagen adaptada de Md Lokman (2015) [105].

Los métodos convencionales son los más utilizados, altamente precisos y normalmente tienden a utilizarse en la mayoría de los paquetes comerciales (ANSYS FLUENT, Abaqus, etc). Sin embargo, los métodos convencionales pueden llegar a ocupar mayor memoria de cómputo cuando resuelven el campo de fluidos en determinado dominio de diseño, por ejemplo, en el trabajo D. Kandhai et al. [117], el método Lattice-Boltzman (LBM por sus siglas en inglés) utiliza 10 veces menos memoria en comparación que el MEF cuando resuelven el flujo en reactores de mezcladores estáticos, lo que hace que sea casi imposible resolver grandes problemas en un tiempo razonable y más aún si hay que resolver el problema MOT. En cambio, los métodos de aceleración son entonces categorizados en dos grupos principales: métodos numéricos avanzados y técnicas de hardware. Las técnicas de hardware de aceleración se utilizan generalmente junto con los métodos numéricos convencionales y avanzados [105]. Cabe destacar que la programación paralela según las técnicas de hardware puede realizarse en la unidad central de procesamiento (CPU por sus siglas en inglés), computación de propósito general en unidades de procesamiento gráfico (GPGPU por sus silgas en inglés) o en una combinación de CPU y GPGPU.

Para el caso de la programación paralela en la CPU, que es la técnica de hardware que se usa en este trabajo de investigación; esta puede ser ejecutada ya sea por MPI, OpenMP o mediante computación paralela en la nube. Cabe destacar que MPI “Message Passing Interface” y OpenMP “Open Multiprocessing” son dos tipos de programación tradicionales que llegaron a dominar la computación paralela en las décadas de 1990 y son estándares definidos comúnmente para Fortran y C/C++ [118]. La diferencia de estas dos técnicas de programación paralela radica en la arquitectura del sistema paralelo que utilizan. MPI se utiliza para sistemas de memoria distribuida mientras que OpenMP se utiliza en sistemas de memoria compartida tal como se muestra en la Figura 4.8; donde, la comunicación de procesos en MPI es a través del paso de mensajes mientras que en OpenMP no es necesaria la comunicación entre procesos ya que todos los hilos comparten la memoria.

Cabe mencionar que a pesar de que MPI y OpenMP son técnicas de programación en paralelo tradicionales y todavía tienen un lugar en el mundo multinúcleo, el proceso de aprendizaje de dichos lenguajes es demasiado fuerte para la mayoría de los programadores (integración de bibliotecas externas, sintaxis más compleja, necesidad de compilar códigos, etc.) por lo que en este trabajo se adicionan bucles paralelos “parfor” en las líneas que demandan mayor tiempo computacional del código MOT en conjunto con la interfaz de herramientas que presenta MATLAB como medio de paralelización. MATLAB posee dos herramientas de paralelización: **a)** la herramienta Parallel Computing Toolbox como paradigma similar al de memoria compartida habilitada con OpenMP, pero en este caso desarrollado de una forma más sencilla e intuitiva, y **b)** MATLAB Distributed Computing Server donde se permite escalar la computación paralela en clúster pudiendo distribuir tareas en los distintos nodos, similar al paradigma de memoria distribuida MPI. La herramienta de interfaz escogida en este proyecto como técnica de paralelización es la de Parallel Computing Toolbox, que permite al usuario seleccionar la cantidad “workers” o “núcleos” que se requieren para ejecutar el algoritmo. Adicional a esta herramienta de paralelización que se usa en MATLAB, el algoritmo MOT es ejecutado en la nube debido a que los proveedores por medio de las máquinas virtuales van a proporcionar los núcleos de CPU como hardware para llevar a cabo la paralelización.

Los recuadros de color verde de la Figura 4.8 ilustran la técnica de paralelización usada en este trabajo, que es el método convencional de los elementos finitos para solución del sistema de ecuaciones MEF $\boldsymbol{z} = \boldsymbol{k}^{-1} * \mathbf{b}$ en conjunto con la técnica de hardware de computación en la nube para la selección de recursos computacionales, usando a la CPU como medio de paralelización.

La herramienta Parallel Computing Toolbox de MATLAB permite el acceso a la paralelización en una computadora multi-procesos similar a la arquitectura usada en OpenMP de memoria compartida, tal como se muestra en la Figura 4.8.

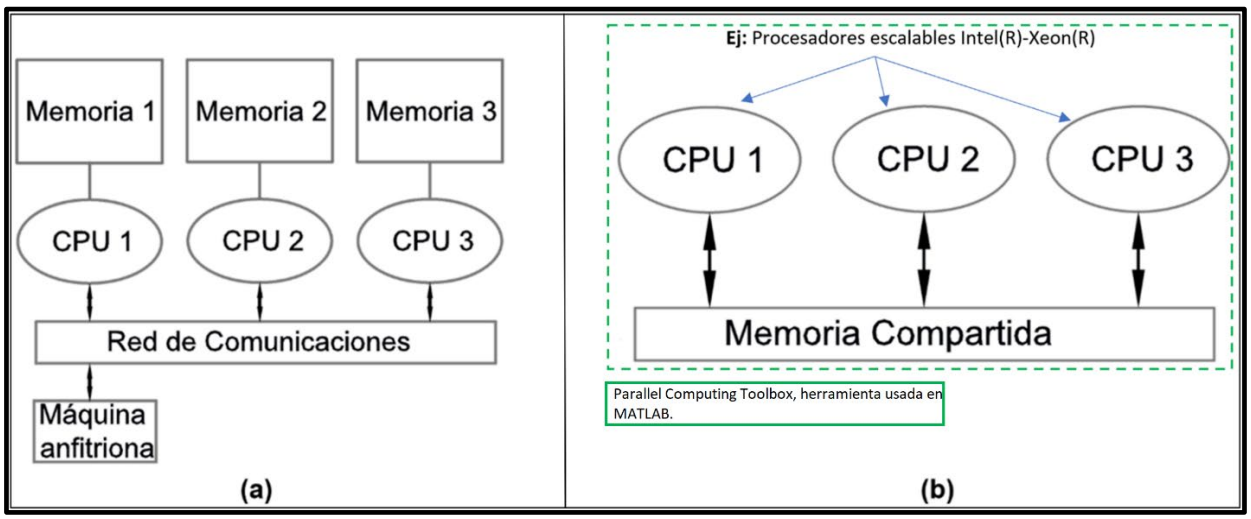


Figura 4.8.- a) Arquitectura del sistema paralelo MPI de memoria distribuida para un clúster de computadoras y **b)** Arquitectura del sistema paralelo OpenMP de memoria compartida para una computadora multi-procesador.

4.4 TERMINOLOGÍA BÁSICA

En esta sección se presenta las terminologías básicas comúnmente usadas en computación paralela, con el objetivo de poder medir el desempeño del algoritmo MOT paralelizado en el CAPÍTULO 5.

4.4.1 ACELERACIÓN (speedup)

El término de aceleración se usa cuando un código ha sido paralelizado. Es uno de los indicadores más simples y más ampliamente usados para medir el rendimiento de un programa desarrollado en computación paralela, y se define como [119]:

$$Speedup = \frac{Tiempo\ de\ ejecución\ del\ código\ serial}{Tiempo\ de\ ejecución\ del\ código\ paralelo} = \frac{T_s}{T_p} \qquad \text{(Ecc. 4.1)}$$

4.4.2 ESCALABILIDAD

Un aspecto importante al escribir un código en optimización topológica es asegurar la escalabilidad del programa. Es decir, un software es escalable si al aumentar el número de procesos debería resultar en un tiempo de solución más rápido. Caso contrario, un software no es escalable si su rendimiento no "escala" o crece con el incremento del número de procesadores. Para cuantificar la escalabilidad del programa en paralelo se ha utilizado la Ley de Amdahl (1967).

4.4.3 LEY DE AMDAHL

Realmente no es una ley, sino una aproximación que modela la aceleración ideal que puede ocurrir cuando un programa serial es modificado para correr en paralelo [120]. Esta ley muestra que, para un problema de tamaño fijo, la aceleración está limitada por el porcentaje del código que es inherentemente secuencial tal como se muestra en la (Ecc. 4.2) [120]. Podemos notar varios aspectos: si $P = 0$, es decir, nada del código puede ser paralelizado, entonces $Speedup = 1$, en otras palabras, el código no sufre aceleración cuando se usan varios recursos (varios procesadores). Si $P = 1$, entonces todo el código es factible de ser paralelizado y como consecuencia $Speedup$ = ∞, idealmente. En cambio, si $P = 0.5$, quiere decir que el 50% del código puede ser paralelizado, entonces $Speedup$ = 2, es decir, el código paralelizado puede correr dos veces más rápido que la versión en serial.

$$Speedup = \frac{1}{1-P} \qquad \text{(Ecc. 4.2)}$$

Esta (Ecc. 4.2) (ley de Amdahl) puede ser escrita de otra manera teniendo en cuenta el número de procesadores usados en la paralelización: la aceleración esperada por un código paralelo sobre uno serial cuando se usan n procesadores está dictada por la fracción del programa que puede ser paralelizado (P) y la fracción de éste que no puede ser paralelizado $(1 - P)$. Esta relación se presenta en la (Ecc. 4.3) [120]:

$$Speedup\ (n) = \frac{1}{(1-P)+P/n} \quad \text{(Ecc. 4.3)}$$

Para esta (Ecc. 4.3) podemos notar que si $P = 0$, la $Speedup$ = 1 igual que en la (Ecc. 4.2); es decir, el código no sufre aceleración. Ahora si $P = 1$ entonces el $Speedup = n$, lo que significa que la aceleración alcanzada al paralelizar el código es linealmente proporcional al número de procesadores usados y no infinita como en la (Ecc. 4.2). En la práctica lo ideal sería obtener un $P = 1$, pero no siempre sucede esto debido a que a medida que se va aumentando el número de procesadores, el código tiende a ser menos escalable; es decir, que la (Ecc. 4.3) tiene límites tal como se ilustra en la Figura 4.9, y esto depende de la fracción de código que puede ser paralelizado.

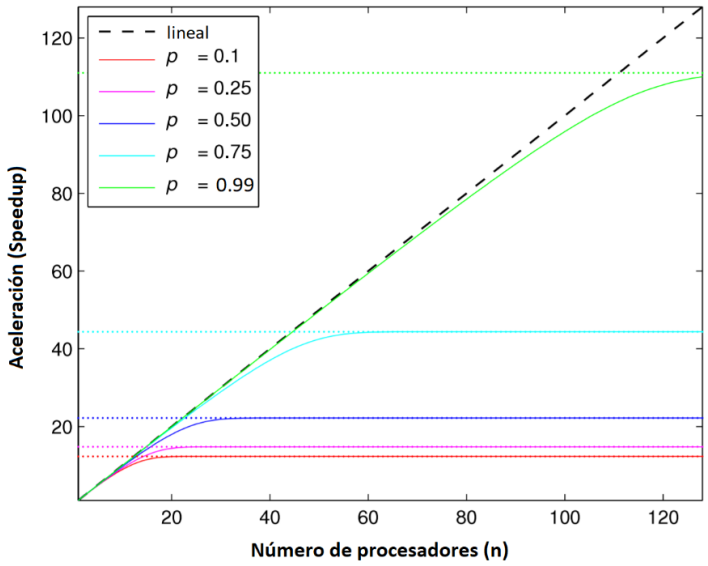


Figura 4.9.- Límites para la escalabilidad según la ley de Amdahl. Imagen adaptada de [121].

4.4.4 EFICIENCIA PARALELA

La eficiencia paralela $Efic.$ está dada por la normalización de la (Ecc.4.1) por el número de procesadores usados en el código desarrollado en paralelo y se define en la (Ecc. 4.4) [119]. Según la (Ecc. 4.4), la eficiencia $Efic.$ decrece al aumentar el número de procesadores n. Sin embargo, hay que observar que la eficiencia se puede mantener incrementando n y también aumentando el trabajo realizado por éstos [119].

$$Efic. = \frac{T_s}{n * T_p} \qquad \text{(Ecc. 4.4)}$$

4.4.5 NOTACIÓN O GRANDE (BIG-O)

Esta notación describe como el tamaño de un problema puede afectar el consumo de recursos computacionales (memoria y/o procesador) usado en un algoritmo. Algunas tasas de crecimiento más comunes son [120]:

- $O(1)$: Esto significa que el algoritmo siempre consume el mismo tiempo, independiente del tamaño de conjunto de datos de entrada.
- $O(N)$: El consumo de recursos computacionales por este tipo de algoritmos crece linealmente con el tamaño de los datos de entrada.
- $O(N^2)$: El desempeño del algoritmo es directamente proporcional al cuadrado del tamaño de los datos de entrada.

Para más detalle sobre la notación O grande, se pueden consultar textos sobre análisis de algoritmos, por ejemplo, "*Introduction to Algorithms* por Cormen et al" [122].

4.5 RESEÑA DE TRABAJOS MOT PARALELIZADOS APLICADO A FLUIDOS

El MOT aplicado a los fluidos es un método que se ha ido expandiendo ya sea para estudios de varios diseños aplicativos a nivel académico o industrial. Algunos de los trabajos encontrados en la literatura referente al uso aplicativo de técnicas de paralelización es el de Niels Aage (2007)[123] donde desarrolló un algoritmo computacional para la resolución de problemas de flujo de Stokes en 2D y 3D a gran escala utilizando cálculos paralelos en una computadora de memoria compartida “shared memory computer”. El esquema de optimización topológica se implementa en fortran90 usando OpenMP y Sun Performance Library (SPL) para la paralelización. El programa esta compilado y ejecutado en un sistema de multiprocesamiento simétrico (SMP) que consta de 24 Sun Ultrasparc IV de 1.35 GHz ejecutado en un sistema operativo Sun Studio 9 y un total de grupo de memoria de 48 GB. La sección paralelizada del algoritmo desarrollado por Niels Aage fue aplicada al solucionador lineal usando la factorización de Cholesky en paralelo. El diseño planteado en este estudio fue realizado para un problema en 2D y 3D como se muestra en la Figura 4.10.

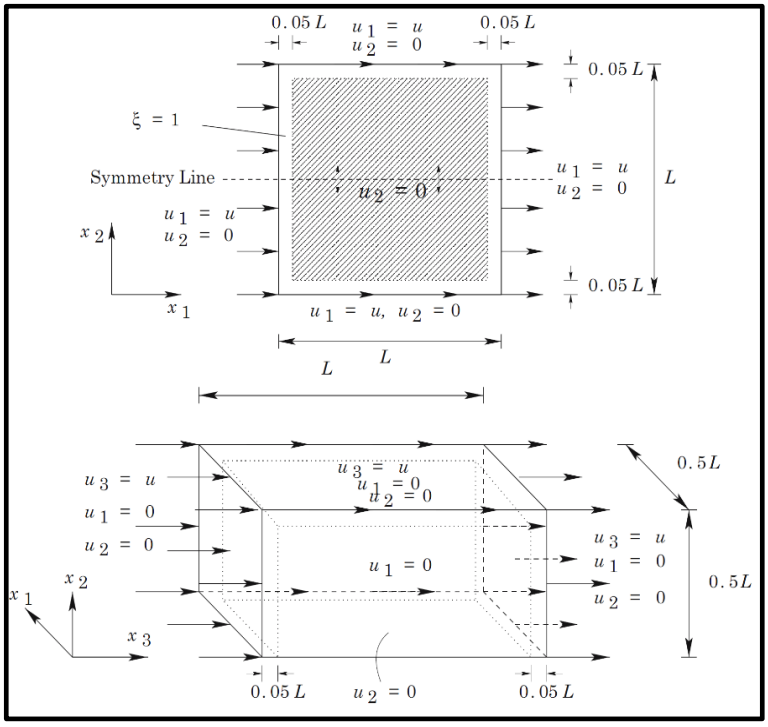


Figura 4.10.- Esquema del problema en 2D (arriba) y 3D (abajo) con condiciones de frontera. El boceto en 3D muestra $1/4$ de simetría de todo el dominio de diseño [123].

Para el esquema del problema en 2D, aprovechando la simetría, se utilizaron seis CPU para obtener un diseño optimizado de ceros y unos, con $2'250.000$ elementos después de 384 iteraciones y 38.5 horas de cálculo tal como se muestra en la Figura 4.11 [123].

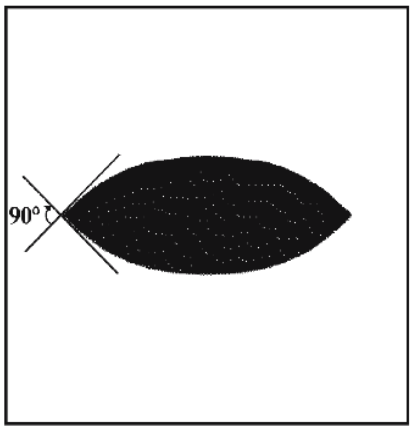


Figura 4.11.- Diseño optimizado minimizando la energía de disipación para un ejemplo en 2D con una fracción volumétrica $frac_{.vol} = 0.85$ y un total de $2'250.000$ elementos. Se muestra el campo de la variable de diseño γ [123].

En este trabajo también se realizaron simulaciones para el problema en 3D usando las mismas condiciones de fronteras que son planteadas para el caso en 2D. Para este problema tridimensional Niels usa 6 CPU, obteniendo un diseño optimizado para la energía de disipación con un total de 128.000 elementos finitos, con 349 iteraciones y un costo computacional de 197 horas. El valor funcional de la energía de disipación para el caso en 3D fue de un valor de $C_d = 4.97$ [Watts].

Otros de los trabajos que se encuentran en la literatura usando técnicas de paralelización aplicando el MOT a problemas de fluidos es el realizado por Kentaro Yaji (2017) [124]. En este trabajo se desarrolla un algoritmo computacional en paralelo de optimización topológica desarrollado en Fortran aplicado al diseño de un problema termo-fluídico inestable a gran escala que incorpora el método de Lattice Boltzman (LBM por sus siglas en inglés) para la solución de las ecuaciones de estado, un método adjunto local en el tiempo donde se divide el problema de optimización por subintervalos con el objetivo de dividirlo en subproblemas razonables para reducir el costo de la memoria y computación paralela usando una supercomputadora Fujitsu PRIMEHPC FX100 en la universidad de Nagoya, Japón. Cada nodo computacional fue equipado con dos procesadores “Fujitsu SPARC64 XIfx” de 16 núcleos a 2.2 GHz y 32 GB de memoria.

La configuración de diseño que ellos plantean para la solución al problema de optimización topológica se muestra en la Figura 4.12 [124], en donde la velocidad de entrada se define como un perfil parabólico y la expresión está dada por la siguiente ecuación:

$$u_{x1} = U_{in} + \Delta U_{in} sin(\omega t),\ u_{x2} = u_{x3} = 0\ \ t\ \epsilon\ [0,2\pi] \quad \text{(Ecc. 4.5)}$$

Donde $\omega = 1$, U_{in} se define como un perfil parabólico, cuyo valor máximo se estableció en $9.0x10^{-2}$ $[m/seg]$, y $\Delta U_{in} = 1.0x10^{-2}$ es la amplitud de la oscilación.

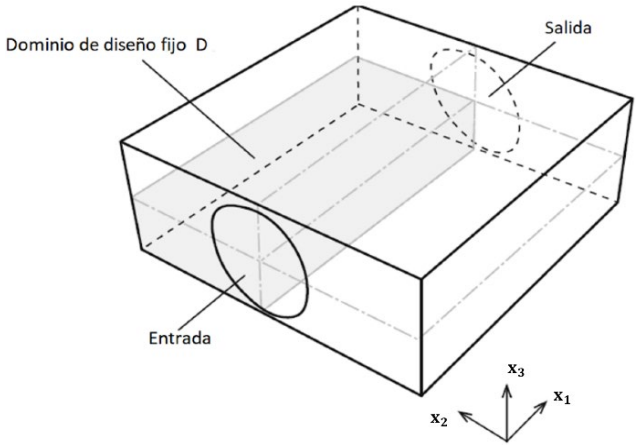


Figura 4.12.- Configuración de diseño, en las que una condición de contorno simétrica se impone, la velocidad de entrada se hace oscilar (velocidad varía en el tiempo), y las condiciones de salida y pared son las mismas que las de Chen (2017) [125]. Imagen tomada de [124].

El objetivo en el problema de optimización topológica realizado en este trabajo se enfoca en la maximización del intercambio de calor sujeto a baja pérdidas de presión. El método de optimización se resuelve utilizando el método de las asíntotas móviles (MMA). Los parámetros para el sistema termo-fluido que se usan son: $R_e = 40$, el coeficiente de generación de calor del sólido es de $\beta = 1.0x10^{-3}$ y un número de Prandtl de $P_r = 7$. La variable de diseño inicial en todo el dominio de diseño se establece en $\gamma = 0.99$ y el factor de penalización q se establece en un valor de $q = 0.01$. El diseño optimizado se muestra en la Figura 4.13, utilizando una malla discretizada de $200x100x40 = 800.000$ elementos finitos.

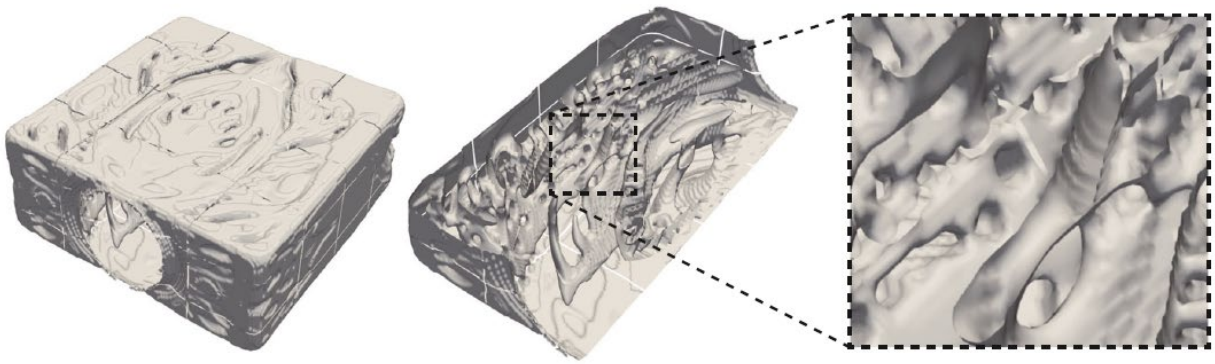


Figura 4.13.- Configuración optimizada. **Izquierda:** Dominio completo. **Derecha:** cuarto de dominio correspondiente al dominio de diseño fijo D. Imagen tomada de [124].

Para mostrar el desempeño paralelo de la supercomputadora Fujitsu PRIMEHPC FX100, se muestra en la Tabla 4.1 el tiempo de solución y el escalado computacional, aumentando progresivamente el número de nodos. Para estos tres casos, la reducción de memoria en el supercomputador se la realiza dividiendo el problema de optimización con un valor de subintervalos de $K = 49$.

Tabla 4.1.- Escalado del algoritmo computacional para una optimización con subintervalos $K = 49$. Se asignaron dos procesos en cada nodo y 16 subprocesos se utilizan para cada proceso.

# de elementos finitos	Nodos	Procesos	Tiempo [seg]	Escalado [%]
$200x100x40$	8	16	234	100
$400x200x80$	64	128	240	97.5
$800x400x160$	512	1.024	251	93.2

Otro trabajo que se puede encontrar en la literatura usando técnicas de paralelización aplicadas al MOT en fluidos es el realizado por Joe Alexandersen (2016) [126]. En este trabajo el desarrolla un algoritmo computacional paralelizado para optimizar disipadores de calor refrigerados por convección natural tridimensional a gran escala. Las ecuaciones que gobiernan el desarrollo de este algoritmo son las ecuaciones incompresibles de estado estacionario de Navier-Stokes acopladas a la ecuación de difusión-convección térmica mediante la aproximación de Boussinesq. El problema multifísico se resuelve utilizando elementos finitos tri-lineales en un marco paralelo que permite la optimización de grandes problemas con un orden de 20 a 330 millones de grados de libertad. El flujo es considerado laminar y el estudio topológico a gran escala se realiza variando el número de Grashof entre 10^3 y 10^6, donde se concluye que el número de ramificaciones en el diseño optimizado es mayor a medida que aumenta este número.

En este trabajo la solución de los problemas de transferencia de calor conjugada a gran escala se resuelve con la implementación del MEF utilizando el software PETSc [126] y el marco de optimización topológica es resuelta por medio del método de las asíntotas móviles. El software PETSc se utiliza debido a la escalabilidad paralela, disponibilidad de solucionadores lineales y no lineales, pre-acondicionadores y la facilidad de uso de mallas estructuradas.

El problema de optimización considerado en este estudio es un ejemplo (académico) de un disipador de calor en una cavidad cúbica cerrada. La Figura 4.14 [126], muestra la configuración del problema con sus dimensiones y sus condiciones de frontera específicas. La fuente de calor (negro en la figura), ejemplifica un chip electrónico, se coloca en la mitad de la cavidad inferior y se modela usando un pequeño material sólido con generación de calor volumétrico, $s_o = 10^4$. El dominio de diseño (gris en la figura) se coloca encima de la fuente de calor. Las paredes exteriores verticales y superiores de la cavidad se suponen que se mantienen a una temperatura fría constante, $\mathrm{T} = 0$, mientras que el fondo esta aislado. Las dimensiones de la cavidad son de $1x1x1$, las dimensiones del dominio de diseño son $0.75x0.75x0.75$ y las dimensiones de la fuente de calor son $0.1x0.05x0.1$. Para este ejemplo en específico el diseño y las soluciones de estado se solucionan manteniendo un cuarto de simetría durante la optimización debido a la simetría de las condiciones de dominio y de contorno.

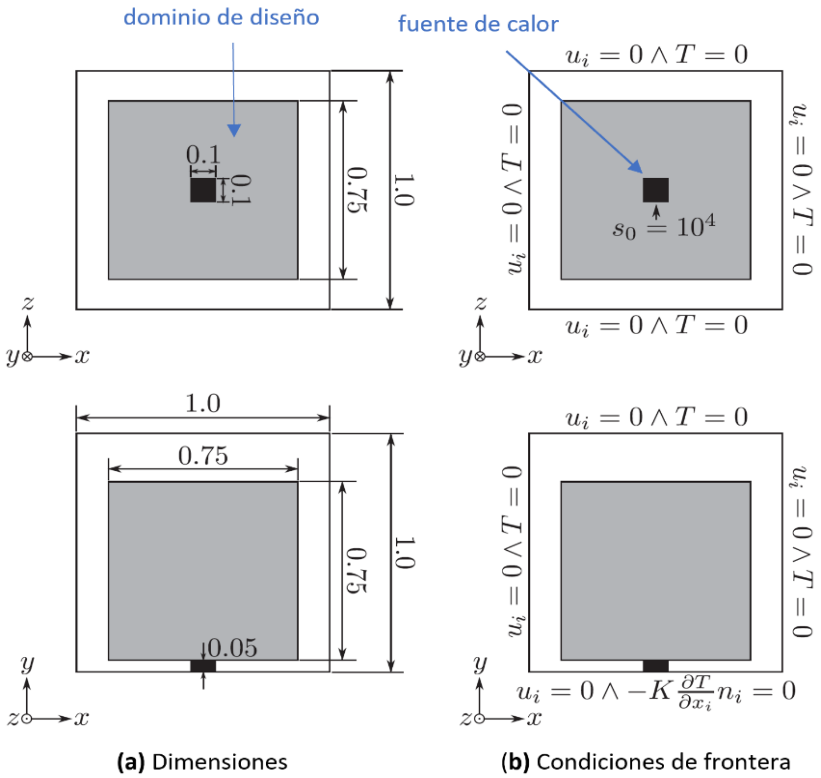


Figura 4.14.- Configuración del problema del disipador de calor en una cavidad cúbica cerrada. Imagen tomada y adaptada de Joe Alexander (2016) [126].

Para demostrar el rendimiento paralelo en el solucionador de estado, el problema de optimización se resuelve en una malla fija para diferentes procesos. Los cálculos en este trabajo se realizan en un clúster donde cada nodo está equipado con dos procesadores Intel-Xeon E5-2680v de 10 núcleos a 2.8 GHz. Los resultados que se muestran en la Tabla 4.2 y Tabla 4.3 tienen un promedio de más de 250 iteraciones de diseño y muestran el rendimiento para números de Grashof de $G_r = 10^3$ y $G_r = 10^6$, respectivamente.

Tabla 4.2.- Tiempo promedio que toma resolver el problema de estado para más de 250 iteraciones de diseño para un número de Grashof de $G_r = 10^3$ en una resolución de malla de $80x160x80$.

Procesos	Tiempo [seg]	Escalado
160	53.2	1.00
320	28.9	0.54
640	14.1	0.26

Tabla 4.3.- Tiempo promedio que toma resolver el problema de estado para más de 250 iteraciones de diseño para un número de Grashof de $G_r = 10^6$ en una resolución de malla de $80x160x80$.

Procesos	Tiempo [seg]	Escalado
160	62.6	1.00
320	31.9	0.51
640	16.5	0.26

Se puede ver en esta reseña que las técnicas de paralelización aplicadas al MOT en fluidos han venido desarrollándose continuamente, y estas técnicas han sido implementadas con una mayor tendencia a la solución de las ecuaciones de estado a diferencia de ser aplicadas al algoritmo optimizador, esto se debe porque el cálculo de solución MEF es el que conlleva el mayor tiempo de ejecución en un algoritmo de optimización topológica. Otros trabajos referentes al uso de técnicas de paralelización aplicadas al MOT en fluidos se pueden encontrar en [127-132].

4.6 CPU Y VIRTUALIZACIÓN (MÁQUINA VIRTUAL) / NO VIRTUALIZACIÓN (BARE METAL) EN LA NUBE

4.6.1 CPU, BREVE RESEÑA DE SU EVOLUCIÓN

Hace 50 años atrás la unidad central de procesamiento (CPU por sus siglas en inglés) estaba compuesta por un single-core y la comunicación de datos entre el núcleo y la RAM era a través del bus de datos/direcciones tal como se muestra en la Figura 4.15a. En aquel tiempo la velocidad de los procesadores y la memoria RAM eran equivalentes.

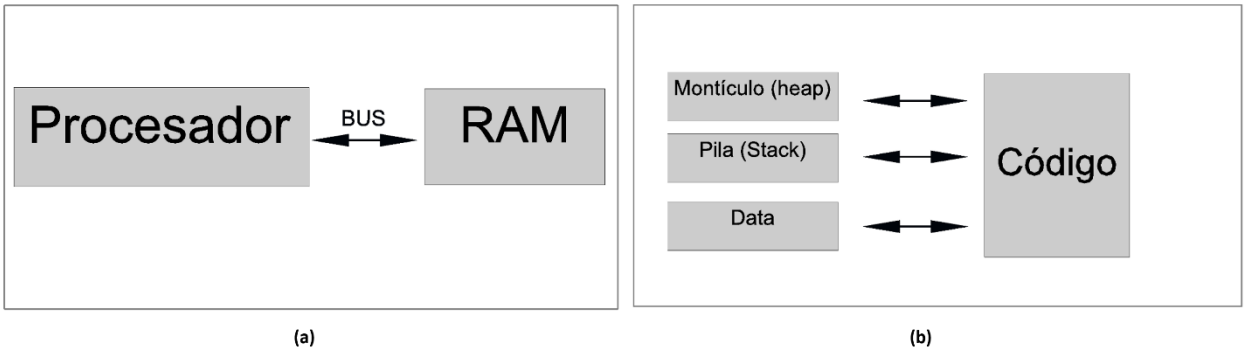


Figura 4.15.- a) Computador de un solo núcleo (single-core computer) y **b)** programa en ejecución.

En este single Core (CPU de un solo núcleo), el programa de ejecución de un algoritmo computacional estaba dado por el esquema de la Figura 4.15b. La memoria RAM se dividía en memoria stack “pila” y memoria heap “montículo” (sectores que se encontraban dentro de la memoria) y es aquí donde los datos de una variable iban a ser alocadas al momento de ejecución de un código computacional. Cabe destacar que la memoria RAM no tiene una división física que indique que variables son destinadas a la memoria stack o a la memoria heap, sino que es un concepto solo lógico que es importante tenerlo en cuenta.

Luego años después, la velocidad de los procesadores fue siendo mayor al de la memoria RAM. En aquel tiempo se introdujo en el procesador una memoria intermedia o conocida también como memoria caché de alta velocidad (ver Figura 4.16a). Esta memoria caché de CPU fue introducida por Intel en el modelo de microprocesador 80486 en 1989 [133]. Solo tenía 8 kb de memoria y un nivel denominado caché de nivel 1 o L1. La función de esta “memoria caché” fue introducida con el objetivo de leer de forma anticipada la memoria RAM mientras que el Core continuaba trabajando y así poder modificar la información leída de forma local. Cuando entra nueva información al caché, la información ya procesada es almacenada en la memoria RAM. En otras palabras, la inclusión de esta memoria caché fue de mejorar el rendimiento en el que eran procesados los datos en la CPU. Adicional a esta incorporación de la memoria caché en el procesador, la ejecución de un algoritmo (ver Figura 4.16b) estaba dada por la ejecución de varios hilos o *threads* en un pseudo-paralelismo. Los threads son una secuencia de instrucciones que se ejecutan de forma independiente. Los *threads* comparten regiones de memoria heap y data, y además trabajan con un stack independiente para cada uno [133].

Luego alrededor del año 2000, surge otro cambio importante en la arquitectura de los procesadores. Este nuevo cambio implicaba la ejecución de los threads en los cores, en contraste a la arquitectura mostrada en la Figura 4.16 donde los threads no eran ejecutados dentro de estos.

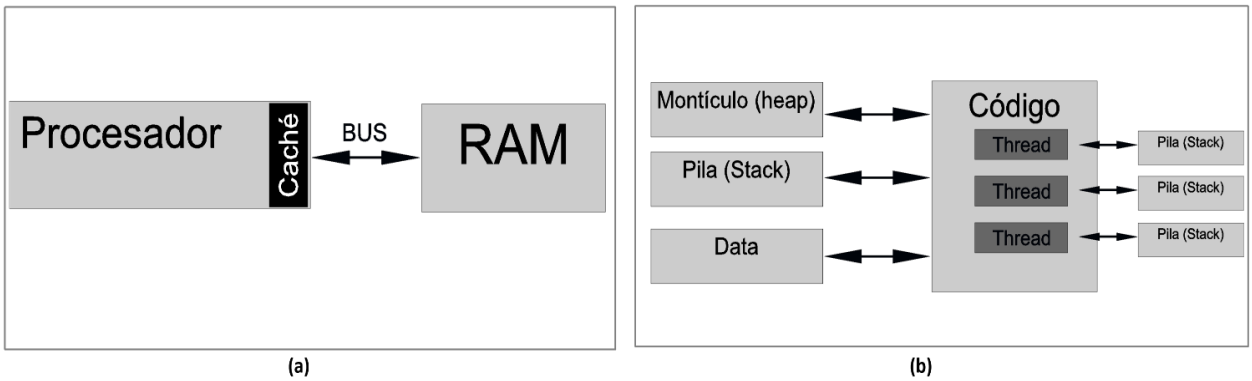


Figura 4.16.- a) Computador de un solo núcleo con memoria caché y **b)** programa en ejecución.

Hoy por hoy la tecnología de los procesadores ha tenido cambios significativos; no solo a nivel de CPU donde actualmente cada núcleo posee hasta dos threads, sino que también se ha incorporado a estas máquinas una tarjeta GPU (unidad de procesador gráficos) como medio adicional de computación acelerada; innovación de NVIDIA en el año 2007, en donde se posee una mayor cantidad de núcleos (miles) a ser paralelizados, pero no es alcance en este trabajo de investigación. Intel y AMD año tras año compiten por tener cada vez más núcleos en la CPU con el fin de proporcionar al usuario final más recursos computacionales en una misma máquina, conocido actualmente con el nombre de supercomputadoras o Workstations.

En la siguiente sección 4.6.2, se presentan las arquitecturas de CPU con las que se trabaja en este proyecto de investigación.

4.6.2 ARQUITECTURAS DE CPU SELECCIONADAS EN LA NUBE PARA EJECUCIÓN DEL ALGORITMO EN PARALELO

Existen algunos proveedores (Microsoft Azure, IBM, Google Cloud Platform, Oracle, Big Step, Scaleway, etc.) que ofrecen recursos computacionales en la nube; ya sea como baremetal (metal desnudo [3]), máquina virtual o híbridos; es decir, un mismo proveedor ofrece el alquiler de su equipo ya sea como máquina virtual o baremetal dependiendo de la exigencia del usuario, un ejemplo claro es el de AWS que en sí es un proveedor que se encarga de proporcionar instancias para el alquiler de máquinas virtuales; pero, hace pocos años atrás comenzó a proporcionar algunas pocas instancias como baremetal. El objetivo en sí de estos proveedores es de ofrecer un alquiler de cómputo (supercomputadoras) por hora a un usuario final, que remotamente acceden al alquiler de estos equipos por medio de un computador personal y acceso a internet a cambio de recibir un pago que será dependiente del número de horas total utilizadas.

[3] *Servidor dedicado exclusivamente a un usuario en especifico, con el fin de que utilice todos los recursos computacionales.*

Para tener un concepto más claro entre los términos baremetal y máquina virtual, el primer concepto que se debe abordar es que los dos tipos de servicios poseen un servidor físico que va a estar ubicado en las instalaciones del proveedor. La diferencia de estos dos términos radica netamente en la manera de como los proveedores gestionan este servidor para el alquiler de estos recursos al usuario (s) final (es); es decir, en baremetal todo el servidor físico está a la dependencia de un solo usuario, por lo tanto, será este usuario que tenga el control total de todo el servidor para el manejo propio de los recursos computacionales que este ofrece. En cambio, en una máquina virtual esto no ocurre de la misma manera, la diferencia radica en que el proveedor virtualizará este servidor físico (por medio de un hipervisor) con el objetivo de crear varias máquinas virtuales y que varios usuarios accedan a estos recursos compartidos (RAM, núcleos, GPU, etc.) de un mismo servidor.

Así, una vez definido los términos del párrafo anterior, es imperativo pensar que un servidor baremetal obtiene mejores desempeños en términos de tiempos computacionales en la ejecución de un algoritmo y costos más elevados en alquiler por hora en comparación con un servidor virtualizado (máquina virtual), debido a que en este último se comparten recursos derivando a un costo menos elevado por alquiler de la hora. Este análisis de tiempos y costos lo dejaremos para el análisis en el CAPÍTULO 5, una vez que se tomen los tiempos de ejecución del código MOT en serie y en paralelo.

Entonces, una vez mencionado lo anterior, para poder realizar una comparación “equitativa” en términos de tiempos y costos por hora entre estos dos tipos de servidores en la nube, se decide seleccionar en este proyecto de investigación instancias en donde la “velocidad de reloj” del procesador sea la misma tanto para máquina virtual como para la máquina baremetal. Así, el proveedor seleccionado para la máquina virtual es AWS y el proveedor seleccionado para la máquina baremetal es Equinix (ver Tabla 4.4).

Tabla 4.4.- Proveedores de máquinas baremetal/virtual para ejecución del código MOT en paralelo en la nube.

Proveedores	Nombre instancia/servidor	Procesador	Procesador Escalable	Vcpus/ Núcleos	Velocidad de reloj [Ghz]
AWS	m5ad.12xlarge	AMD EPYC 7571	No	48 vcpus	2.2
Equinix	s3.xlarge.x86	Intel Xeon Silver 4214	Sí	24 núcleos	2.2

En la Tabla 4.4 se especifica los tipos de procesadores que se usan en la nube para poder medir el desempeño del algoritmo MOT paralelizado en la nube. Cabe destacar que la instancia que se escoge en AWS es la m5ad.12xlarge. Este tipo de instancia cuenta con 1 procesador AMD-EPYC

7571 con 48 vcpus virtuales y 48 núcleos físicos. En cambio, la instancia escogida en la máquina baremetal proporcionada por Equinix es la instancia s3.xlarge.x86. Este tipo de instancia cuenta con 2 procesadores escalables interconectados entre sí. Este procesador consta con 12 núcleos físicos por procesador; es decir, un total de 24 núcleos de recursos que se pueden utilizar en este supercomputador.

Las arquitecturas de estos procesadores son mostradas en la Figura 4.17 y Figura 4.18 respectivamente. La arquitectura del procesador AMD-EPYC 7571 (Figura 4.17) ofrece una velocidad en cada núcleo (core) de 2.2 Ghz, similar a la velocidad de núcleo seleccionada en Equinix. En esta arquitectura existe un canal de comunicación (bus de datos/direcciones) con la RAM y para reducir la latencia de comunicación, el cache L2 es incorporado en este tipo de arquitectura para la comunicación de datos entre los núcleos y la RAM. En cambio, en la CPU de Intel Xeon (Figura 4.18), se muestra procesadores con tecnología escalables que trabajan con más de un procesador multi-core, es decir, cada procesador está colocado en su propio socket. Estas computadoras están diseñadas para que cada procesador acceda a un banco de memoria independiente de forma rápida (NUMA, non-uniform memory Access) debido a que sería lento acceder al banco de memoria del otro procesador, pero, la coherencia del cache hace de comunicación entre estos 2 procesadores, simulando así una de memoria compartida.

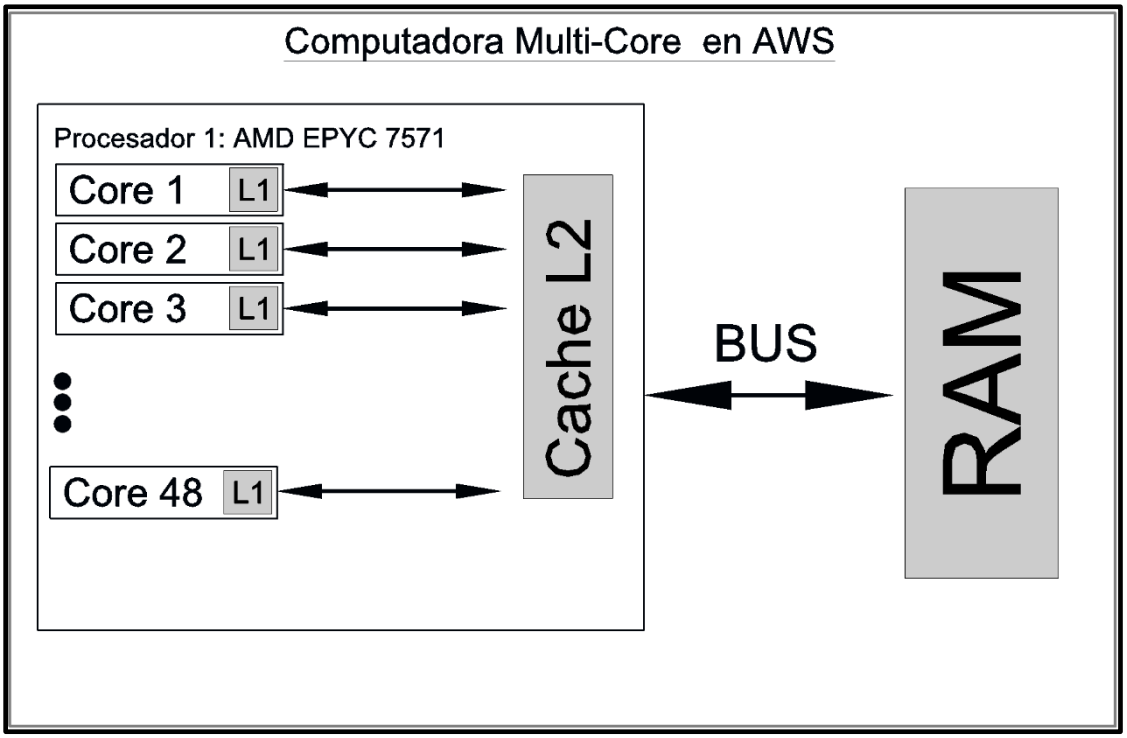


Figura 4.17.- Arquitectura CPU usada en AWS. Velocidad en los núcleos de 2.2 GHz.

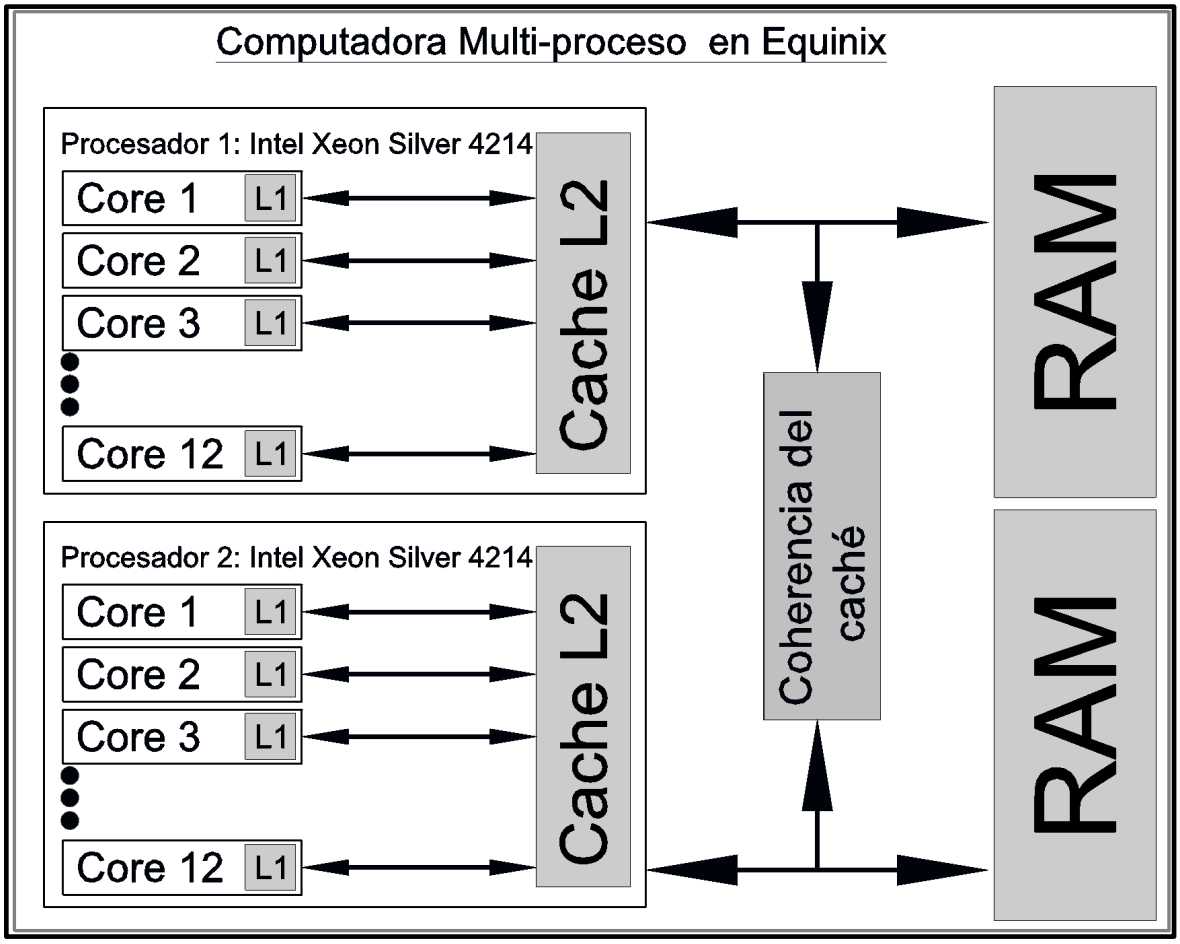


Figura 4.18.- Arquitectura CPU usada en Equinix. Velocidad en los núcleos de 2.2 GHz.

4.6.3 VIRTUALIZACIÓN (MAQUINA VIRTUAL) / NO VIRTUALIZACIÓN (BARE METAL) EN LA NUBE

Una vez mencionado el concepto de baremetal y máquina virtual en la nube, es importante tener presente otro término fundamental en el uso de este tipo de máquinas; que es el concepto de virtualización y no virtualización de estos servidores físicos.

El concepto de virtualización y no virtualización se relaciona directamente con el tipo de arquitectura de una máquina virtual y una máquina baremetal respectivamente (ver Figura 4.19 y Figura 4.20). Es necesario tener presente de que para que un equipo sea máquina virtual o máquina baremetal es fundamental conocer el concepto de virtualización y no virtualización. Este concepto es de importancia debido a que cualquier servidor físico podría generar diferentes máquinas virtuales, es decir, se podría gestionar en una máquina baremetal o incluso en un computador personal varias máquinas virtuales (virtualización del servidor físico) dependiendo de cómo el proveedor de la nube gestiones sus equipos.

Para que un servidor físico sea virtualizado, este debe tener instalado un hipervisor. Este hipervisor conocido también como monitor de máquina virtual (VMM), es un software que crea y ejecuta máquinas virtuales (VM) y que, además, aísla el sistema operativo y administra los recursos del servidor físico para generar varias máquinas virtuales que van a ser alquiladas por horas por los diferentes usuarios.

Hay dos ventajas notoriamente visibles que encontramos en el uso de estas máquinas virtuales por medio del hipervisor:

- Una de las ventajas es que el proveedor es el encargado de crear cuantas máquinas virtuales sean posibles de un mismo servidor físico, distribuyendo así las cantidades de vcpu necesarias para cada máquina virtual.

- Al tener varias máquinas virtuales de un mismo servidor físico, cada máquina virtual maneja su propio sistema operativo, independiente al sistema operativo del servidor físico.

En la primera ventaja mencionada, cabe destacar que el número de VCPU no puede exceder el número de Cores físicos del host (servidor físico). Por ejemplo, si un servidor físico tiene 1 CPU Quadcore (procesador de 4 núcleos), el número máximo de VCPU a asignar es de 4.

Entonces, teniendo estos conceptos de virtualización claros, en la Figura 4.19 y Figura 4.20 se puede destacar que la diferencia más notoria entre estas dos arquitecturas de máquinas radica en el hipervisor. Una máquina baremetal no tiene hipervisor debido a que este servidor va a ser asignado a un solo usuario en específico. En cambio, en la arquitectura de una máquina virtual se requiere el uso obligatorio de un hipervisor para la creación de estas máquinas virtuales.

Es así, que cuando se crea una máquina virtual en AWS, un guest#2 podría estar usando los otros recursos de este mismo servidor físico para otra tarea en particular, es decir, el algoritmo MOT utiliza los recursos computacionales compartidos entre varios usuarios, ver Figura 4.19. En cambio, cuando se crea una máquina baremetal para ejecución de un algoritmo en la nube de Equinix, todos los recursos computacionales están a la disposición de un solo usuario, tal como se muestra en la Figura 4.20. A pesar de esta limitación de recursos que existe en las máquinas virtuales, un factor que podría afectar a la ralentización de ejecución de datos es la comunicación que debe existir entre los núcleos físicos, el hipervisor y la máquina virtual, hasta la obtención final de resultados, debido a una intercomunicación no directa ya que el hipervisor actúa como intermediario.

Cabe mencionar que el alquiler de estas máquinas virtuales y máquinas baremetal se lo realiza remotamente por medio de un ordenador personal, donde el usuario aprovisiona una instancia para ser manejada remotamente. Esta conexión remota se la realiza por medio de establecer un secure Shell (SSH) o una conexión remota de escritorio (RDP) para así poder administrar el servidor físico. Para las instancias que se trabajan en este proyecto se usa una conexión remota de escritorio (RDP) tanto para la instancia en AWS como para la instancia en Equinix.

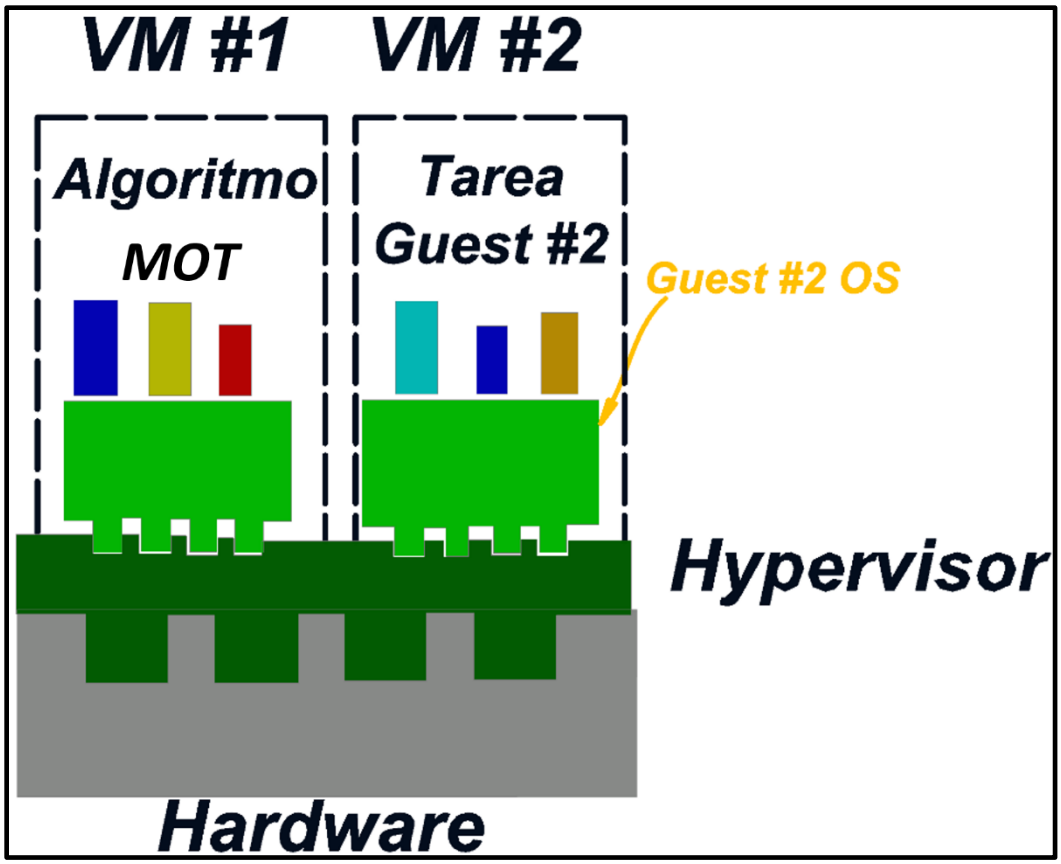


Figura 4.19.- Arquitectura de una máquina virtual. Imagen adaptada de [133].

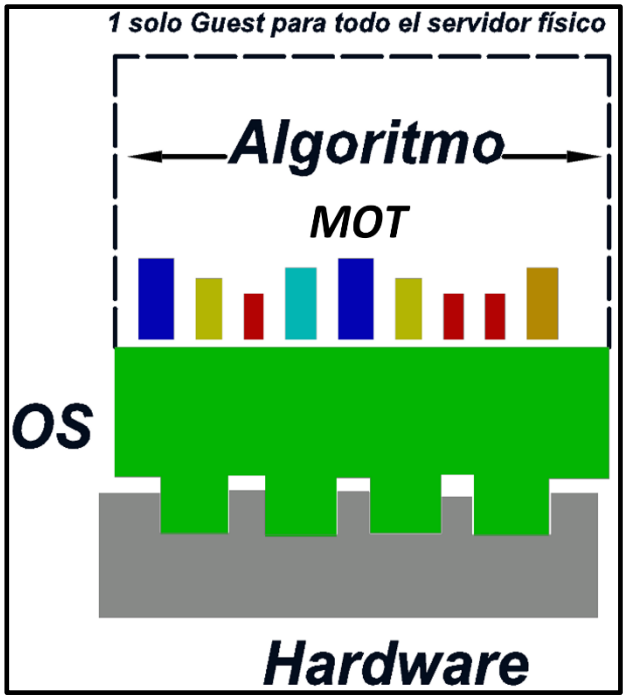


Figura 4.20.- Arquitectura de una maquina bare-metal. Imagen adaptada de [134].

4.7 PSEUDOCÓDIGOS PARALELIZABLES

Tal como se describió en la sección 4.6.2., en donde las arquitecturas de CPU en Equinix y en AWS muestran múltiples procesadores (o núcleos) en un solo computador, es obligatorio el uso de la herramienta Parallel Computing, herramienta que MATLAB proporciona para la paralelización, siguiendo el modelo de memoria compartida tal como se muestra en la Figura 4.8b.

Para un análisis en MATLAB sobre que líneas del código deben ser paralelizadas, en esta sección se decide mostrar en 1 iteración del algoritmo MOT, las líneas que demandan mayor tiempo computacional. Estos tiempos se muestran en la Tabla 4.5, donde se ejecutó el código de una iteración MOT para el caso de la Figura 5.3 (esquema del problema de diseño de rotores de turbomáquinas), usando una malla de 450 elementos finitos en serial. El registro de este tiempo se lo realizó por medio del "profiler" que MATLAB usa por defecto.

Tabla 4.5.- Detalle de líneas que demandan mayor tiempo computacional en 1 iteración del MOT. Profiler que ilustra MATLAB.

Número de Línea	Código	Llamadas	Tiempo Total [seg]	% Tiempo
788	for e=1:ne	22	86.130	37.2%
270	for e=1:ne	22	81.427	35.2%
811	for i=:ne	22	12.466	5.4%
300	for i=1:ne	22	11.965	5.2%
439	[Jelem(:,:,e),Aelemg(:,:,e),drgamae(:,e)]=Jadjoint(coord8f,coord4f...	450	11.075	4.8%
Todas las demás líneas			28.508	12.3%
Total			231.570	100%

Podemos observar en la Tabla 4.5, que las líneas que demandan mayor tiempo computacional son bucles for. Es así que se decide paralelizar estos bucles "for" del algoritmo MOT, y para esto es necesario escribir "parfor" en vez de "for" en las líneas 788, 270, 811 y 300 pero hay que tomar en cuenta que el "parfor" debe respetar las siguientes condiciones para una paralelización eficiente:

- El índice de un bucle "parfor" debe tomar valores enteros.

- Los bucles "parfor" no se pueden anidar.

- Se prohíbe todo comportamiento de dependencia de la iteración anterior.

Otra recomendación que hay que considerar al momento de crear un bucle "parfor" es que este ciclo debe estar en el script principal donde se ejecuta el algoritmo MOT, es decir, no es recomendable que un bucle "parfor" se encuentre dentro de una función que va a ser llamada debido a que esto generaría un fallo en la comunicación de datos entre los núcleos (llamados "workers" en MATLAB).

Una vez conocido las condiciones que debe respetar un bucle “parfor”, procedemos a escribir en términos de pseudocódigos la paralelización del algoritmo MOT. De la Tabla 4.5, podemos ver que las líneas que demandan mayor tiempo computacional son las líneas 788 y 270, con tiempos de ejecución de 86.130 [seg] y 81.427 [seg], representando un porcentaje del 37.2% y 35.2% respectivamente del tiempo total. El pseudocódigo paralelizado de estas líneas está representado en el Cuadro 4.1., en donde se realiza el cálculo de la matriz de rigidez ($\boldsymbol{k}^E$), matriz de vorticidad ($\boldsymbol{M_r}$) y el vector de constantes ***b*** por elemento.

Cuadro 4.1.- Cálculo de la matriz de rigidez ($\boldsymbol{k}^E$), Matriz de Vorticidad ($\boldsymbol{M_r}$) y vector ***b*** por elemento.

```
parfor e=1:ne
Cálculo de las coordenadas por elemento (Q8) y (Q4)
Cálculo de la velocidad promedio de las 2 últimas iteraciones por elemento en x_1 y en x_2
Cálculo de la matriz k^E por elemento
Cálculo de la matriz de M_r de vorticidad por elemento
Cálculo del vector b por elemento.
Almacenamiento de la matriz k^E de todos los elementos.
Almacenamiento de la matriz de vorticidad M_r de todos los elementos.
Almacenamiento del vector b de todos los elementos.
fin
```

Otras líneas representativas en términos de tiempos computacionales son las líneas 811 y 300, con tiempos de 12.466 [seg] y 11.965 [seg], representando un porcentaje del 5.4% y 5.2% del tiempo total respectivamente. El pseudocódigo paralelizado de estas líneas está representado en el Cuadro 4.2, en donde se cálcula pares de permutaciones con repetición de los grados de libertad totales, en donde posteriormente estos pares de permutaciones se utilizan para ensamblar los coeficientes de la matriz $\boldsymbol{k}^E$ por elemento en la matriz global $\boldsymbol{k}$.

Cuadro 4.2.- Permutaciones a los grados de libertad globales por elemento para el ensamble de la matriz de coeficientes global $\boldsymbol{k}$.

```
parfor i=1:ne
Permutaciones a los grados de libertad globales por elemento para ubicar los coeficientes de
las matrices locales k^E en la matriz global k
fin
```

Adicional a estos tiempos computacionales que son mostrados en el recuadro rojo (86.130 [seg] y 81.427 [seg]) y verde (12.466 [seg] y 11.965 [seg]) de la Tabla 4.5, también se representa en forma de pseudocódigo paralelizado las dos líneas adicionales que se encuentran dentro de "todas las demás líneas" de la Tabla 4.5. Estas líneas son la 371 y 877, con tiempos de ejecución de 0.0396 [seg] y 0.0373 [seg] respectivamente. La representación de estas líneas en forma de pseudocódigos paralelizados se detalla en el Cuadro 4.3, en donde se calcula las funciones objetivos de energía de disipación y vorticidad por elemento.

Cuadro 4.3.- Cálculo de las funciones objetivos de energía de disipación c_d y vorticidad c_r por elemento.
parfor i=1:ne Cálculo de la energía de disipación c_d Cálculo de la vorticidad c_r fin

Cabe destacar que los resultados de aceleración (speedup), escalabilidad y eficiencia que son mostrados en el CAPÍTULO 5, están en función de estos tres pseudocódigos paralelizados que son ejecutados en los dos tipos de máquinas ya mencionadas anteriormente (Tabla 4.4): **a)** máquina baremetal proporcionada por Equinix; y, **b)** máquina virtual proporcionada por AWS.

CAPÍTULO 5: RESULTADOS

Una vez detallada toda la base matemática del problema MEF, las funciones objetivos (energía de disipación viscosa y vorticidad) en su forma discreta, el análisis de sensibilidades, el solucionador MMA para el proceso de optimización y las líneas de pseudocódigos paralelizables aplicados tanto en la ejecución de una máquina baremetal como una máquina virtual; en este capítulo se presenta en primera instancia la verificación del código MEF desarrollado en MATLAB contra un software comercial de CFD, ANSYS FLUENT y trabajos de la literatura. Adicionalmente, se realiza la verificación de sensibilidades del método adjunto con el método de las diferencias finitas, y esta se realiza tanto para la energía de disipación viscosa como para la vorticidad, debido a que es necesario para el optimizador del MMA. Luego de realizar estas verificaciones numéricas, se presentan los resultados de las topologías obtenidas en la minimización de la función de la energía de disipación $c_d(\gamma_E)$ en todo el dominio de diseño aplicado a media circunferencia de rotor como se muestra más adelante. En esta media circunferencia de rotor se diseña el canal por donde se conduce el fluido en el interior de una turbomáquina (específicamente una bomba) y a su vez se define la forma de los álabes del rotor. Luego se presentan otras topologías en la minimización de esta función con la variación de parámetros del algoritmo MOT sobre topologías resultantes. Además, se analiza la influencia de la no-linealidad del Jacobiano de la función objetivo, el punto de diseño inicial $(\gamma_E)_o$, la influencia del factor de penalización q y la máxima penalización Brinkman α_{max}. Luego se muestra el efecto en el resultado final topológico que se obtiene al incluir la función de vorticidad $c_r(\gamma_E)$ y la función de energía de disipación $c_d(\gamma_E)$ en una función bi-objetivo, de manera de poder evaluar la priorización de una función sobre otra por medio del método de la suma ponderada de pesos. Posterior a los resultados de optimización topológica obtenidos con el mejor desempeño, se realiza un post-procesamiento de estos rotores de media circunferencia en una circunferencia completa, en el software comercial de ANSYS FLUENT con el objetivo de verificar que las minimizaciones de las funciones objetivos cumplen con los requerimientos exigidos por el algoritmo MOT. Como último, se evalúa el desempeño del código en términos de aceleración, escalabilidad y eficiencia tanto en la máquina baremetal proporcionada por Equinix como en la máquina virtual proporcionada por AWS, y se realiza un breve análisis de costos por horas de alquiler de estas máquinas.

5.1.- VERIFICACIÓN DEL CÓDIGO MEF IMPLEMENTADO Y ANÁLISIS DE SENSIBILIDADES

El código MEF que se desarrolla en MATLAB y sigue la (Ecc. 2.24), (Ecc. 2.25) y (Ecc. 2.26) en la solución de los valores nodales, tiene que ser verificado debido a que por cada iteración del algoritmo MOT se requiere la solución de los campos de velocidad y presión. Por lo tanto, para llevar a cabo esta verificación, se considera el álabe recto de la Figura 2.4. Adicionalmente se realiza la verificación de las sensibilidades de las funciones objetivos previo a la obtención de los resultados del MOT aplicados a turbomáquinas.

5.1.1- VERIFICACIÓN DEL CÓDIGO MEF APLICADO AL FLUJO EN UN ÁLABE RECTO

La verificación de este álabe recto es realizada con ANSYS FLUENT y con los trabajos desarrollados por Romero y Silva (2014) [4] y posteriormente con el trabajo de Esteban Foronda (2020) [76] debido a la cercanía de los problemas de diseños aquí abordados.

Las condiciones de fronteras planteadas para este problema de alabe recto se muestran en la Figura 2.4, con una velocidad de flujo uniforme en la entrada de 1 [m/seg], con condiciones de no deslizamiento en las paredes y con una presión en la salida de 0 [Pa].

Esta simulación se realiza para un flujo viscoso estacionario, laminar, incompresible y en un marco no inercial de referencia (marco rotacional). Los parámetros de esta simulación se detallan en la Tabla 5.1.

En las Figura 5.1 y Figura 5.2 se presentan los resultados que se obtienen para el campo de velocidad y presión para el flujo viscoso utilizando el código implementado al resolver el sistema de ecuaciones $\boldsymbol{z} = \boldsymbol{k}^{-1} * \boldsymbol{b}$, al igual que su contraparte en software comercial y los provenientes de la literatura. Es así que estos campos globalmente son considerados como verificados, y a pesar de que se puede construir métricas numéricas para cuantificar de una mejor manera la exactitud de estos resultados, esto no se considera necesario debido a que estas diferencias mínimas en los resultados en comparación contra el software comercial y la literatura se pueden haber originado por la selección de diferentes criterios de convergencia, diferentes elementos finitos y diferentes algoritmos de solución al problema numérico. Adicionalmente se puede destacar que ANSYS FLUENT usa el método de los volúmenes finitos, un método diferente al usado en este proyecto.

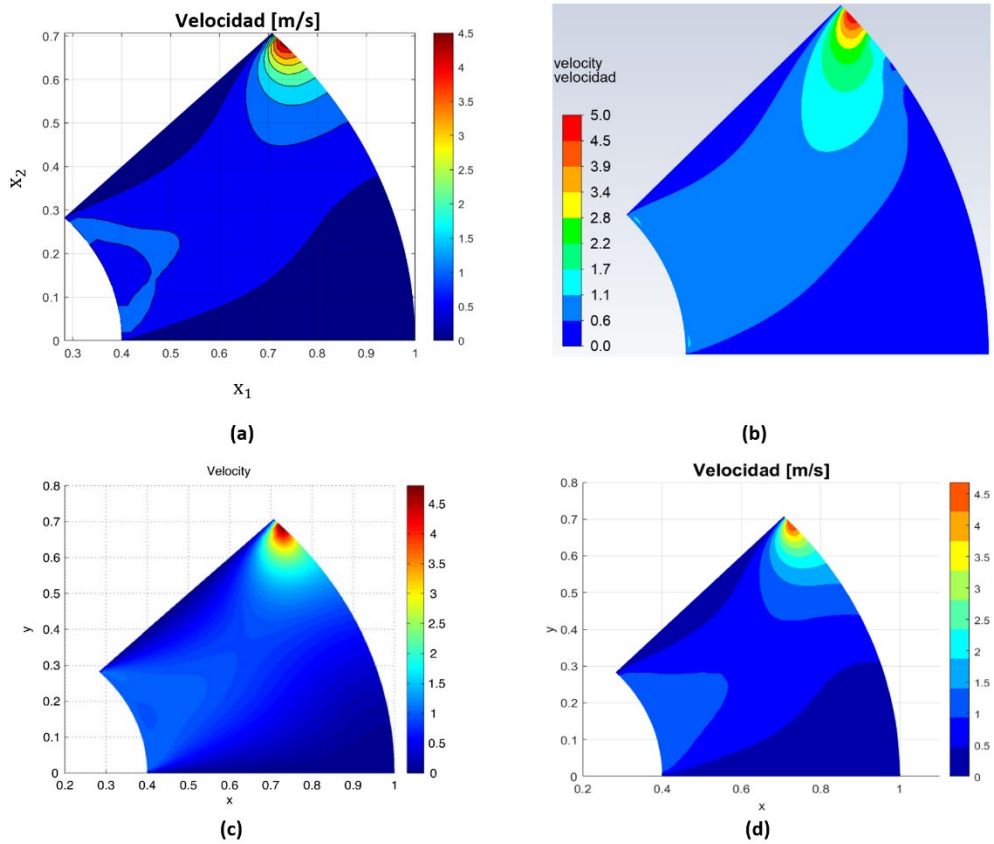


Figura 5.1.- Campo de velocidad para el problema del álabe recto de flujo viscoso para verificación del código MEF, según Tabla 5.1. **(a)** código implementado, **(b)** Software comercial, **(c)** Romero y Silva (2014) [4], **(d)** Esteban Foronda (2020) [76].

Tabla 5.1.- Parámetros de simulación del álabe recto.

Parámetros	Valor
Fluido	Genérico
Viscosidad cinemática	0.1 [kg/m.s]
Densidad del fluido	1 [kg/m^3]
Velocidad de entrada	1[m/seg]
Presión de salida	0 [Pa]
Velocidad angular Ω	1000 rpm

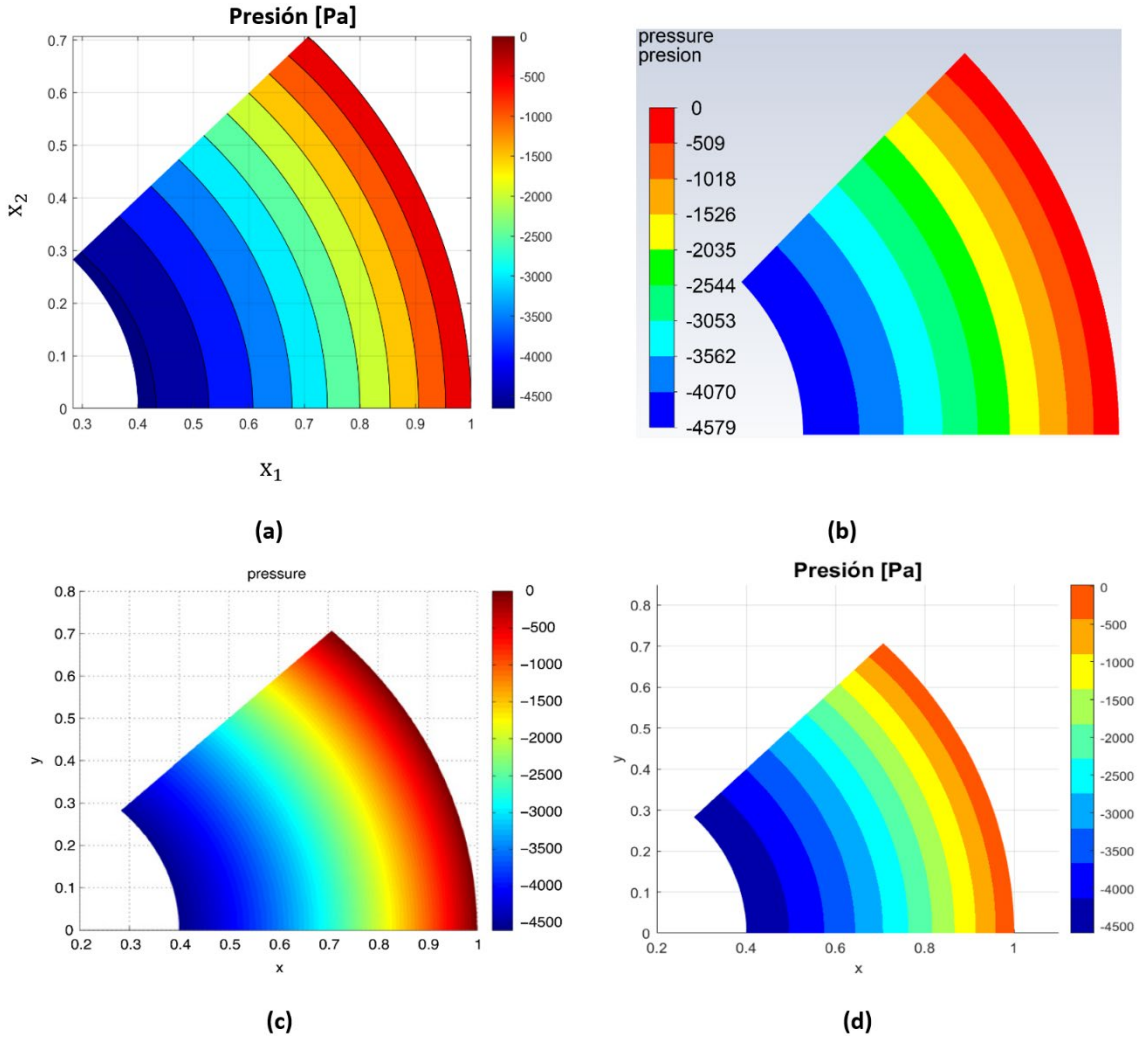


Figura 5.2.- Campo de presión para el problema del álabe recto de flujo viscoso para verificación del código MEF, según la tabla 5.1. **(a)** código implementado, **(b)** Software comercial, **(c)** Romero y Silva (2014), (d) Esteban Foronda (2020).

5.2.- VERIFICACIÓN DE ANÁLISIS DE SENSIBILIDADES

Debido a que el MMA requiere la derivada de la función objetivo con respecto a la variable de diseño, en esta sección se realiza la comparación de los resultados del cálculo de las sensibilidades realizadas por el método adjunto frente a los obtenidos mediante el método de las diferencias finitas. Existen tres formas básicas para calcular la derivada de la función objetivo con diferencias finitas: **a)** diferencia hacia adelante, **b)** diferencia hacia atrás y, **c)** diferencia central. Cabe destacar que el mejor esquema de diferencias finitas es el esquema de diferencia central, puesto que aproxima mejor la derivada (menor error), sin embargo, conlleva mayor tiempo computacional. Los otros dos esquemas son similares en cuanto costo computacional y por lo tanto la verificación del cálculo de sensibilidades del método adjunto se la hará con el esquema de diferencia hacia adelante.

5.2.1.- SENSIBILIDAD DE LA ENERGÍA DE DISIPACIÓN

Para los cálculos de sensibilidad de la energía de disipación se utiliza el esquema presentado en la Figura 5.3 (esquema del problema de diseño de rotores de turbomáquinas), con los parámetros de fluidos que se presentan en la Tabla 5.4. Para verificar la sensibilidad del método adjunto de la energía de disipación (Ecc. 3.31), se compara estos resultados obtenidos a través del método adjunto con los resultados obtenidos a través de diferencias finitas hacia adelante, el cual está descrito por la (Ecc. 5.1):

$$\frac{\partial c_d(\gamma_E)}{\partial(\gamma_E)} \cong \frac{c_d(\gamma_E+\Delta\gamma_E)-c_d(\gamma_E)}{\Delta\gamma_E} \qquad \text{(Ecc. 5.1)}$$

$$\gamma_E = 1,2,3, \ldots n_E.$$

Donde c_d es la función objetivo de la energía de disipación que se puede observar en la (Ecc. 3.4), γ_E son las variables de diseño, n_E es el número de elementos finitos usados para discretizar el dominio de diseño y $\Delta\gamma_E$ es una variación pequeña sobre la variable de diseño.

Se discretiza el dominio de diseño con 450 elementos finitos y un $\Delta\gamma_E$ de $1x10^{-4}$, para calcular las sensibilidades del problema tanto por el método adjunto como por el método de las diferencias finitas. Los resultados se presentan en la Tabla 5.2.

Tabla 5.2.- Comparación de las sensibilidades calculadas a través del método adjunto y el método de las diferencias finitas para la energía de disipación.

Variable de diseño (γ_E)	Sensibilidad a través del método de diferencias finitas	Sensibilidad a través del método adjunto	Error relativo (%)
1	-8,1543e − 02	-8,1653 e − 02	0,14
2	-3,3518e − 01	-3,3619e − 01	0,30
3	-7,2243e − 01	-7,3126e − 01	1,22
4	-1,1247	-1,1565	2,82
5	-1,5356	-1,5467	0,73
6	-1,8765	-1,8774	0,04
7	-2,1287	-2,1373	0,40
8	-2,3343	-2,3344	0,003
9	-2,464	-2,4742	0,41
10	-2,569	-2,5759	0,27
11	-2,6398	-2,6480	0,31
12	-2,7266	-2,7267	0,004
13	-2,8325	-2,8358	0.12
14	-3,1346	-3,1475	0,41
15	-4,2456	-4,2584	0,30
16	-0,4733	-0,4733	0,0

5.2.2.- SENSIBILIDAD DE LA VORTICIDAD

Para el cálculo de sensibilidad de la vorticidad también se utiliza el esquema de la Figura 5.3, con los mismos parámetros de fluidos que se presentan en la Tabla 5.4. De la misma manera que la sensibilidad de la energía de disipación, se verifica la sensibilidad de la vorticidad del método adjunto (Ecc. 3.39) y se compara los resultados obtenidos con el método de las diferencias finitas hacia adelante, el cual esta descrito por la (Ecc. 5.2):

$$\frac{\partial c_r(\gamma_E)}{\partial(\gamma_E)} \cong \frac{c_r(\gamma_E+\Delta\gamma_E)-c_r(\gamma_E)}{\Delta\gamma_E} \qquad \text{(Ecc. 5.2)}$$

$$\gamma_E = 1,2,3, \ldots n_E.$$

Donde c_r es la función objetivo de vorticidad que se puede observar en la (Ecc. 3.6). γ_E, n_E y $\Delta\gamma_E$ tienen la misma representación que la sensibilidad de la energía de disipación. En este punto se destaca que el vector de velocidades $\boldsymbol{u_r}$ de la (Ecc. 3.6) tiene dependencia de manera indirecta de variable de diseño $\boldsymbol{u_r}(\gamma_E)$.

Se utiliza una discretización de 450 elementos finitos y un $\Delta\gamma_E$ de $1e-4$, para calcular las sensibilidades del problema tanto por el método adjunto como por el método de las diferencias finitas. Los resultados se presentan en la Tabla 5.3.

Tabla 5.3.- Comparación de las sensibilidades calculadas a través del método adjunto y el método de las diferencias finitas para la vorticidad.

Variable de diseño (γ_E)	**Sensibilidad a través del método de diferencias finitas**	**Sensibilidad a través del método adjunto**	**Error relativo (%)**
1	-1,1800e − 03	-1,1900e − 03	0,85
2	-2,9800e − 03	-3,0300e − 03	1,66
3	-6,120e − 03	-6,120e − 03	1,71
4	-8,545e − 03	-8,560e − 03	0,19
5	-1,001e − 02	-1,013e − 02	1,22
6	-1,164e − 02	-1,089e − 02	6,46
7	-1,1103e − 02	-1,1105e − 02	0,20
8	-1,054e − 02	-1,078e − 02	2,24
9	-9,89e − 03	-1,032e − 02	4,32
10	-9,43e − 03	-9,67e − 03	2.57
11	-8,91 e − 03	-9,17e − 03	2,95
12	-8,54 e − 03	-8,65 e − 03	1,32
13	-8,84 e − 03	-8,95 e − 03	1,24
14	-1,089e − 02	-1,042e − 02	4,31
15	-1,9895e − 02	-1,9995e − 02	0,50
16	-6,54 e − 03	-6,847 e − 03	4,69

5.3.- RESULTADOS PARA TURBOMÁQUINAS

Una vez verificado el MEF y el análisis de sensibilidades, en esta sección se decide presentar los resultados topológicos de la minimización de las funciones objetivo evaluando así el efecto de estas en el diseño del alabe.

El esquema que se toma como referencia para la obtención de los resultados, es el que se presenta en la Figura 5.3, esquema similar al trabajo de Romero y Silva (2014) [4]. La selección de este esquema en dos dimensiones se justifica debido a que las velocidades radiales son las que predominan a diferencia de las velocidades axiales donde el efecto de estas no es significativo en el diseño de bombas centrífugas [4]. Este modelo es una simplificación de asumir un flujo entre dos alabes consecutivos en el rotor de una bomba centrífuga. Los parámetros de diseño de este esquema considerados en esta tesis se encuentran sintetizados en la Tabla 5.4, donde se observa que se tiene una velocidad en la entrada uniforme y normal a la curva del radio interno de 1 [m/seg], una presión en la salida de 0 [Pa] y condición de no deslizamiento en las paredes.

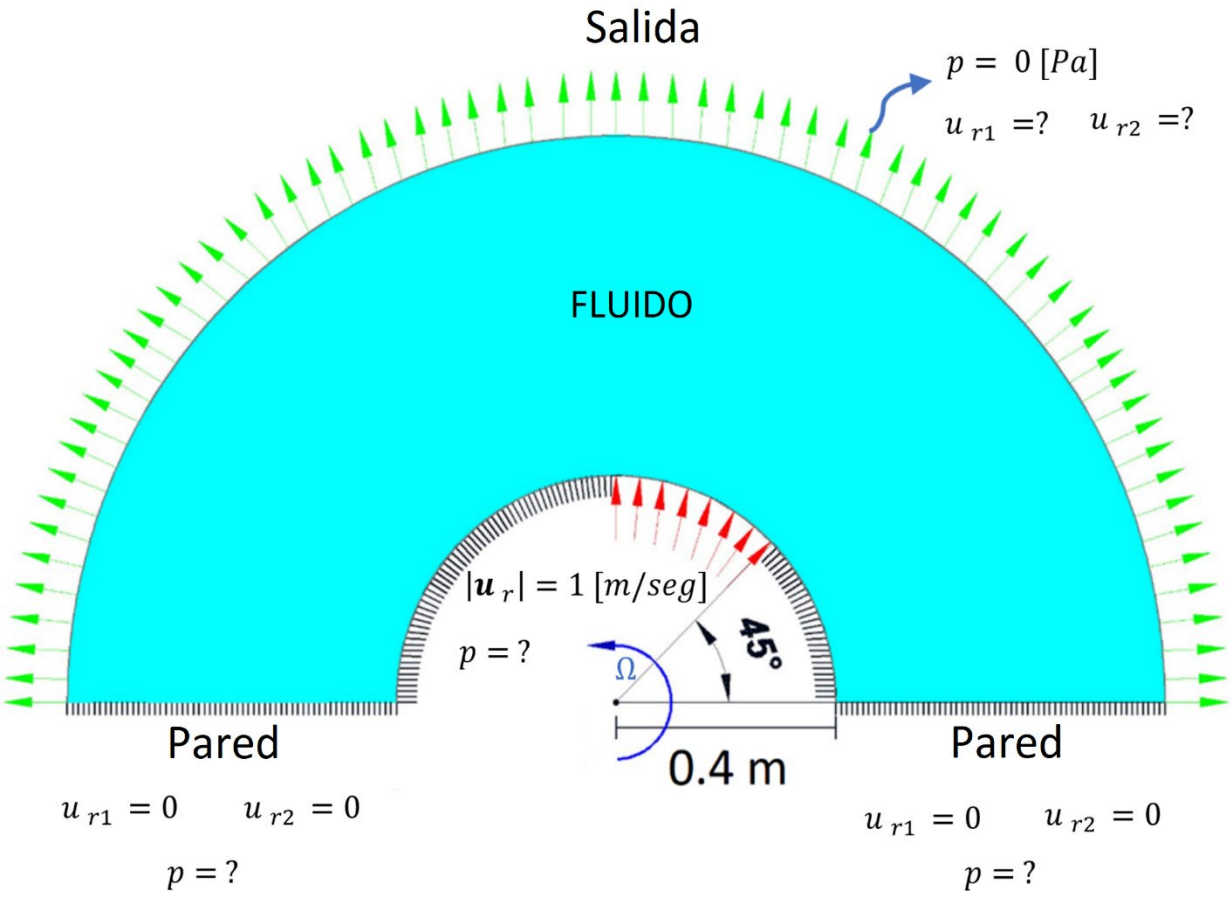


Figura 5.3.- Esquema del problema de diseño de rotores de turbomáquinas usando MOT-Campo de fluidos para una bomba centrífuga.

Tabla 5.4.- Parámetros del problema de diseño de rotores de turbomáquinas para el campo de fluido y MOT.

Parámetros de campo de fluido	
Parámetro	**Valor**
fluido	Genérico
Viscosidad dinámica μ	0.1 [kg/m.seg]
Densidad de fluido ρ	1 [kg/m^3]
Velocidad entrada $\|\boldsymbol{u_r}\|$	1 [m/seg]
Presión en la salida	0 [Pa]
Penalización permeabilidad q	1
Máxima penalización Brinkman α_{max}	$1x10^5$
Mínima penalización Brinkman α_{min}	0
Parámetros MOT	
Fracción Volumétrica objetivo	0.25
Método de optimización	MMA
Número de elementos finitos	1.800

Estos parámetros presentados en la Tabla 5.4 corresponden a la configuración de referencia en que se basan todos los resultados topológicos en este capítulo, puesto que en las próximas secciones se modificarán algunos de estos para evaluar la incidencia sobre la topología final.

5.3.1- MOT MONO-OBJETIVO: MINIMIZACIÓN DE ENERGÍA DISIPADA

La minimización de la energía de disipación viscosa sigue el problema MOT que se presenta en la (Ecc. 3.9), con la diferencia que los pesos de suma ponderada en esta ecuación siguen la configuración de $w_d = 1$ y $w_r = 0$, dando la no consideración al efecto de la función de vorticidad ya que primero se evalúa la influencia de la minimización de la energía disipada sobre el resultado de la topología final del álabe con el objetivo de estudiar de manera escalonada a cada una de las funciones objetivos. Así, en la Figura 5.4a se encuentra la topología final cuando se da priorización a la minimización de la energía de disipación viscosa. Los parámetros usados en este problema de optimización se muestran en la Tabla 5.5 con condiciones de frontera del campo de fluido del esquema de la Figura 5.3.

Tabla 5.5.- Parámetros MOT mono-objetivo $(w_d = 1 \text{ y } w_r = 0)$.

Parámetro	**Valor**
MOT	
Fracción de volumen objetivo	< 0.25
Variable de diseño inicial $(\gamma_E)_0$	0.25
Máxima penalización Brinkman α_{max}	$1x10^5$
Penalización de permeabilidad q	1
Radio máximo del R_{max}	0.03 [m]
Número máximo de iteraciones	21
Desempeño	
Fracción de volumen final	0.2493
Energía disipada	5.88 [Watts]

En esta topología final en la minimización de la energía de disipación viscosa, podemos notar que el resultado corresponde a una geometría característica de rotores en bombas centrífugas, donde se muestra un perfil de rotor hidrodinámico con cierto radio de curvatura más pronunciado en la salida del fluido, favoreciendo así la transferencia de momentum en esta región. Cabe destacar que el significado físico de esta minimización de energía recae en la minimización del trayecto del fluido desde la entrada del impeler hasta su salida de este; generando así las mínimas perdidas posibles de energía del fluido.

En la Figura 5.4a se observa que el modelo de material implementado en la (Ecc. 3.2) es lo suficientemente fuerte como para representar el sólido como un fluido impermeable. Se puede notar que esta topología obtuvo ligeras escalas de grises en la parte superior izquierda del álabe. Cabe destacar que estas escalas de grises (variables de diseño intermedias), no representan ni una región sólida ni fluida, es decir, no tiene una representación física. Estas escalas de grises pueden ser atribuidas al uso de un radio de filtro muy grande R_{max}. La influencia de estas escalas de grises sobre el campo de velocidades de la Figura 5.4b es que se obtienen velocidades de menor magnitud en estas secciones; es decir, dificultad del fluido a fluir en estas zonas de grises debido a que estas variables de diseño darían cierta magnitud significativa a la fuerza de Darcy (Ecc. 2.5).

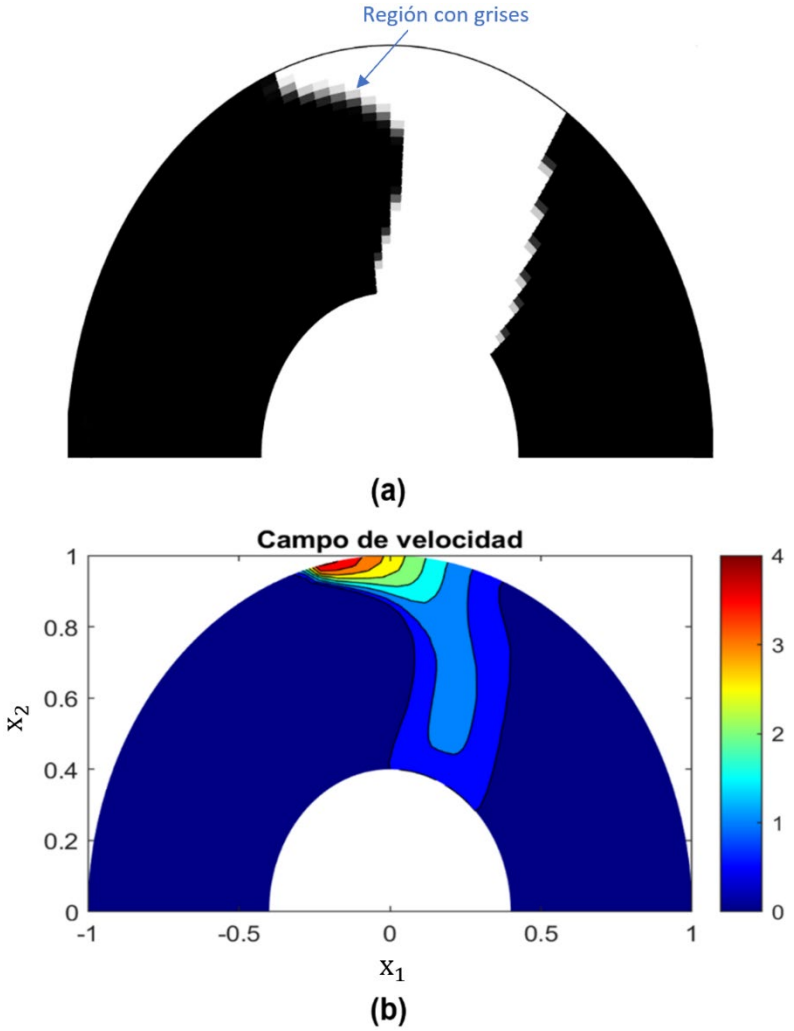


Figura 5.4.- Resultados MOT para el diseño de rotores de turbomáquinas minimizando la energía de disipación ($w_d = 1$ y $w_r = 0$). **a)** Topología final, **b)** Campo de velocidad.

En la Figura 5.5 se muestra la curva convergencia del algoritmo MOT. Se observa que esta curva alcanza rápidamente su convergencia en la minimización de la energía de disipación, es decir, con apenas 22 iteraciones. Esta rápida convergencia se debe al hecho de que las sensibilidades de la energía de disipación alcanzan un mínimo local rápidamente de manera que los gradientes se comienzan a estabilizar, lo que resulta a la vez en una estabilización de la energía de disipación. En esta misma figura podemos observar que la restricción de volumen objetivo de 0.25 es violada en las primeras iteraciones debido a que el campo de fluidos no puede ser resuelto apropiadamente cuando se tiene una distribución inicial homogénea de la variable de diseño, puesto que el dominio de diseño corresponde a un medio poroso que restringe completamente el flujo, generando una elevada disipación de energía y originando así una violación de la fracción de volumen. Después de la iteración número 5 podemos observar que la curva de fracción de volumen comienza a disminuir para luego estabilizarse y cumplir con esta restricción a la vez que se minimiza la función objetivo de energía. Es así como se observa que el MMA alcanza la convergencia del valor de la función objetivo con un desempeño de 5.88 [Watts] y una fracción de volumen de 0.2493 en todo el dominio de diseño (ver Tabla 5.5).

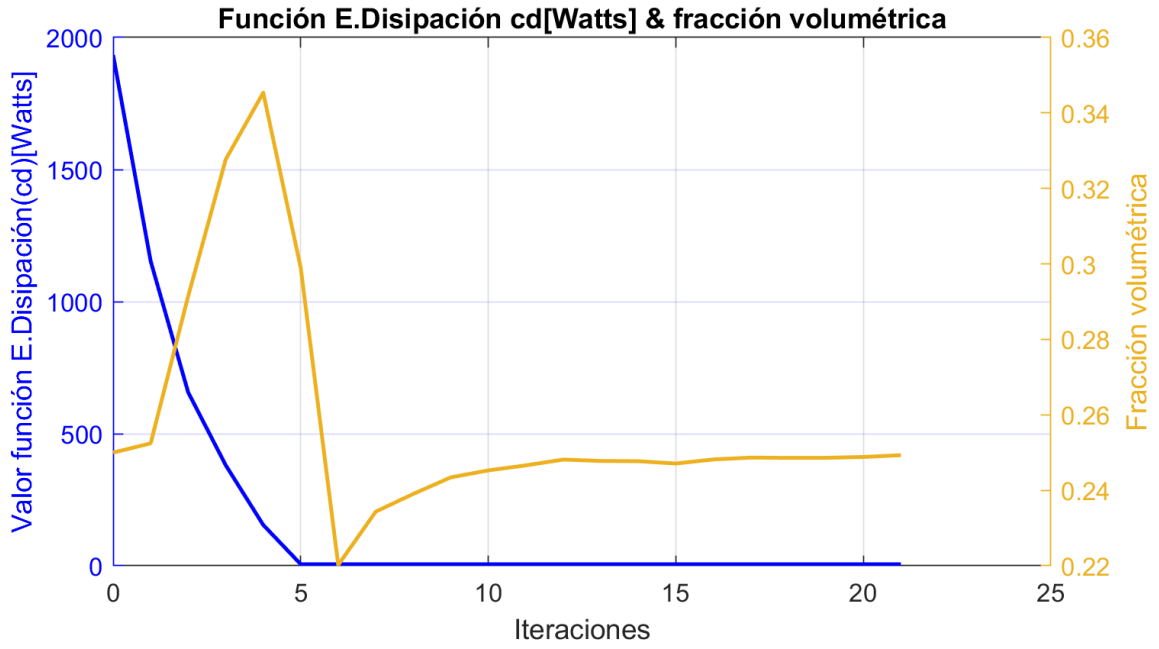


Figura 5.5.- Curva de convergencia minimizando la energía de disipación ($w_d = 1$ y $w_r = 0$) para el diseño de rotores en turbomáquinas.

5.3.2- EVALUACIÓN DE PARÁMETROS MOT MONO-OBJETIVO: MINIMIZACIÓN DE ENERGIA DISIPADA

Considerando el resultado de referencia de la topología anterior, en esta sección se decide evaluar el efecto de modificar los parámetros del algoritmo MOT sobre la topología final obtenida. Los parámetros que se estudian en esta sección son: la influencia de la no-linealidad del Jacobiano de la función objetivo, el punto de diseño inicial, el factor de penalización q y la máxima penalización de Brinkman α_{max}. En esta sección se realiza el estudio de la variación de los parámetros para la obtención de nuevas topologías en el diseño de turbomáquinas. La configuración de parámetros referenciales bases de esta sección son los mostrados en la Tabla 5.5.

- NO-LINEALIDAD DEL JACOBIANO DEL RESIDUAL

En el cálculo de sensibilidades de las funciones objetivos es necesario el cálculo del Jacobiano del residual de las ecuaciones de Navier-Stokes (Ecc. 3.38), en el cual incluye términos no lineales en los términos advectivos en las ecuaciones de momentum en x_1 (Ecc. 2.24) y en x_2 (Ecc. 2.25), pero que usualmente son reducidos a una cuantificación lineal. Al no considerar estos términos no-lineales para el cálculo de la matriz Jacobiana, se considera entonces a u_{r1_k} y u_{r2_k} (provenientes de los términos advectivos), como valores de velocidades promedios de las dos últimas iteraciones (linealización de Picard); en cambio, al considerar estos términos no-lineales en el cálculo de la matriz del Jacobiano, las velocidades u_{r1_k} y u_{r2_k} son consideradas como variables para el cálculo de las derivadas $\partial R_1/u_{r1_j}$ y $\partial R_3/u_{r2_j}$ respectivamente. Así, la decisión de considerar los términos no lineales depende mucho del problema en específico en el cual se esté trabajando, pero para el caso de las ecuaciones desarrolladas para el diseño de álabes en turbomáquinas debemos considerar estos términos no-lineales debido a que tienen incidencia significativa en la topología final.

En la Figura 5.6 se muestra el resultado obtenido al simular el mismo problema de la Tabla 5.5 con la única modificación de linealizar el Jacobiano del residual (no consideración de los términos no lineales) de la ecuación de Navier-Stokes. Esta topología alcanza su convergencia en la iteración 26, con una energía de disipación de 5.98 [Watts] y una fracción de volumen de 0.2492 en todo el dominio de diseño. Podemos notar en esta topología obtenida tiene un ligero incremento en el área de salida del fluido en comparación con la topología de la Figura 5.4a (consideración de los términos no-lineales), lo cual conlleva a una ligera mayor pérdida de la energía del fluido.

- PUNTO DE DISEÑO INICIAL

La influencia del punto de diseño inicial sobre las topologías finales fue demostrada por Romero y Silva (2014) [4] y Sá (2016) [92], donde consideraron puntos iniciales de la variable de diseño con distribución homogénea. Es así, que en esta sección se decide evaluar la incidencia de dicha variable sobre la topología final del álabe, cambiando el valor de esta a un mismo valor para todos los elementos finitos del dominio de diseño (distribución homogénea).

El desempeño de las topologías obtenidas cambiando el valor de la variable de diseño con esta distribución homogénea para la energía de disipación se muestran en la Tabla 5.6, los cuales los resultados son mostrados en las Figura 5.7 y Figura 5.8. A pesar de que los resultados topológicos de la Figura 5.7 son muy similares entre sí, el radio de curvatura no es el mismo, sobre todo en el lado izquierdo del álabe. En la última topología de la Figura 5.7d, se aprecia un radio de curvatura menos marcado, dando un diseño de un álabe más recto en comparación con las otras topologías.

Debido a los diferentes resultados topológicos que se obtienen, se puede asegurar que el problema MOT para turbomáquinas es propenso a obtener varios desempeños que minimicen la función de energía de disipación, de manera que esto se traduce a la obtención de diferentes óptimos locales para un solo objetivo.

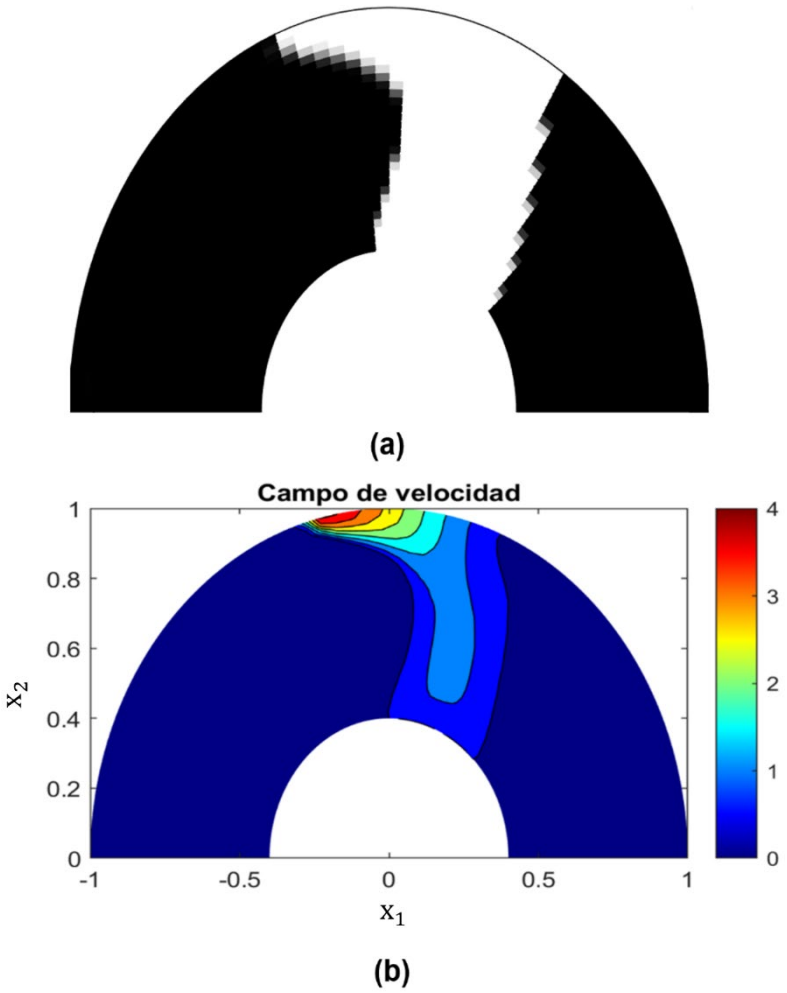


Figura 5.6.- Resultados MOT para el diseño de rotores de turbomáquinas minimizando la energía de disipación ($w_d = 1$ y $w_r = 0$). Linealización del Jacobiano. **a)** Topología final, **b)** Campo de velocidad.

Un aspecto que es importante tener en cuenta, es que cuando algoritmo el MOT parte de pseudodensidades homogéneas cercanas o iguales a $(\gamma_E)_0 = 0$ (completamente negro), se dificulta el hallazgo de una solución óptima, lo que se traduce en una convergencia más lenta del algoritmo de optimización. Por ejemplo, para el caso de $(\gamma_E)_0 = 0.05$, la curva de convergencia del MMA tiende a converger en un mayor número de iteraciones debido a que se parte de una distribución homogénea que es completamente "sólido" en todo el dominio de diseño, lo cual restringe la energía de disipación (valores altos de perdida de energía), incurriendo así en una demora en la minimización. El número de iteraciones que alcanza en converger el algoritmo para este caso en particular es de 35 iteraciones.

Tabla 5.6.- Desempeño de las topologías para diferentes variables de diseño inicial con distribución homogénea para la energía de la disipación como función objetivo.

Parámetro	**Pseudodensidad Inicial**			
Desempeño:	0.25	0.05	0.50	0.95
Fracción de volumen final	0.249	0.249	0.248	0.249
Energía disipada [Watts]	5.88	6.06	6.03	5.97

Adicionalmente, para validar lo que se mencionó anteriormente con respecto al incremento de la sección de salida del fluido que favorece a la perdida de energía, esto se puede constatar en la Tabla 5.6, que para el caso de $(\gamma_E)_0 = 0.05$ y $(\gamma_E)_o$= 0.50 la energía disipada es de 6.06 [Watts] y 6.03 [Watts] respectivamente, y si observamos las topologías de la Figura 5.7b y Figura 5.7c, se puede notar una ligera sección más ancha en comparación con las demás topologías.

Todo este análisis que se realiza permite concluir que las topologías en el diseño final de álabes para turbomáquinas deben garantizar cierta curvatura, y que el parámetro de la pseudodensidad inicial tiene incidencia sobre el resultado final topológico. El autor concluye y recomienda valores de pseudodensidad iniciales y homogéneas menores a 0.50, debido a que se garantiza una curvatura aceptable en el diseño de álabes en estas turbomáquinas.

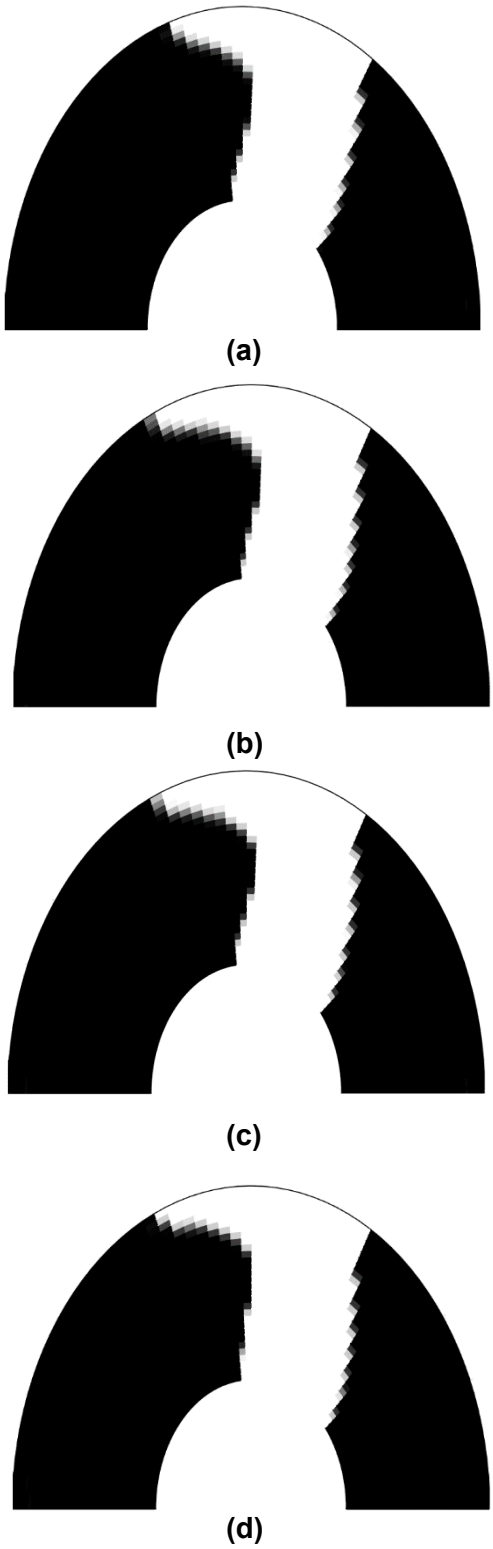


Figura 5.7.- Topologías finales para diferentes variables de diseños iniciales con distribución homogénea para la energía de disipación como función objetivo: **a)** $(\gamma_E)_o$= 0.25, **b)** $(\gamma_E)_o$= 0.05, **c)** $(\gamma_E)_o$= 0.50, **d)** $(\gamma_E)_o = 0.95$.

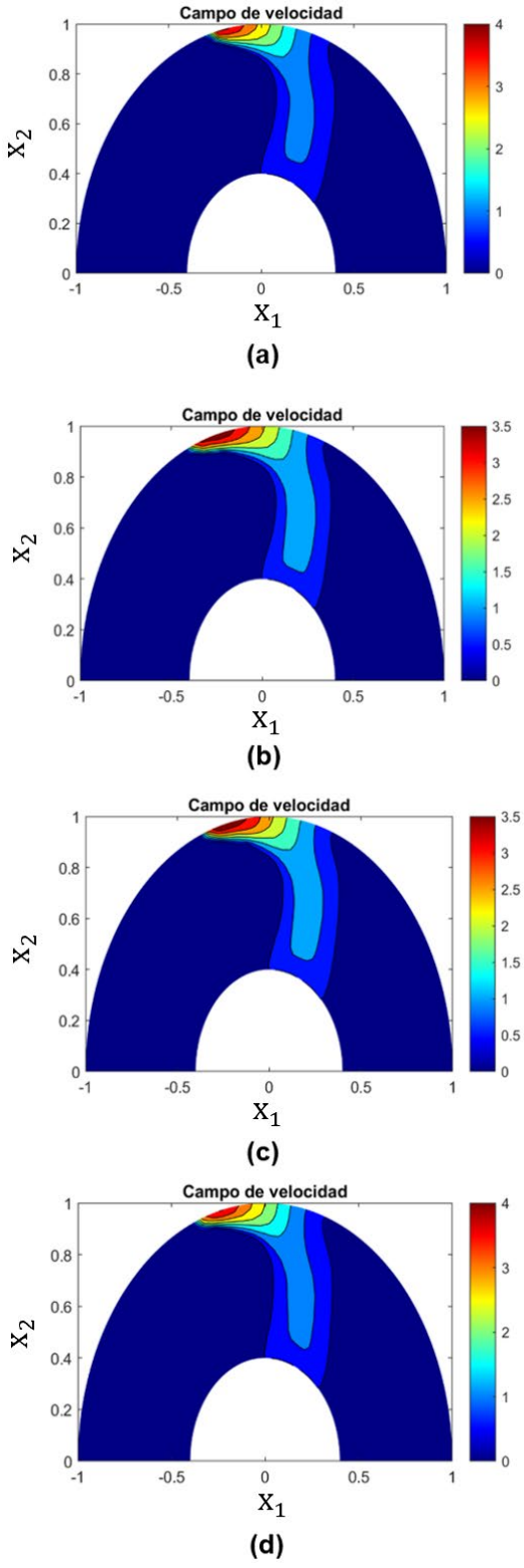


Figura 5.8.- Campo de velocidades de las topologías para diferentes variables de diseños iniciales con distribución homogénea para la energía de disipación como función objetivo: **a)** $(\gamma_E)_o$= 0.25, **b)** $(\gamma_E)_o$= 0.05, **c)** $(\gamma_E)_o$= 0.50, **d)** $(\gamma_E)_o$= 0.95.

- RADIO DEL FILTRO R_{max}

Las variaciones del radio R_{max} del filtro de la (Ecc.3.41), usadas en esta sección son: **a)** R_{max} = 0.03 [m], **b)** R_{max} = 0.04 [m] y **c)** R_{max} = 0.05 [m]. Cabe mencionar que no se muestran las topologías ni los campos de velocidades debido a que no hay que destacar algún resultado de importancia, debido a que no difieren en mucho a las topologías anteriormente mostradas. Los desempeños para estos tres radios son: **a)** 5.88 [Watts] con $frac_{vol}$= 0.249, **b)** 5.90 [Watts] con $frac_{vol}$= 0.249 y **c)** 6.29 [Watts] con $frac_{vol}$= 0.248 respectivamente. Cabe destacar que un valor alto de R_{max} (más de 8 elementos finitos dentro del radio del filtro) incrementa la escala de grises y disminuye la tasa de convergencia del algoritmo MOT en la obtención de la topología final del álabe, en cambio radios menores a esta cantidad de elementos finitos favorece la curvatura de diseño.

- FACTOR DE PENALIZACIÓN q Y MÁXIMA PENALIZACIÓN DE BRINKMAN α_{max}

La selección del modelo de material de la (Ecc. 3.2), es un factor de importancia en la minimización de las funciones objetivos. Este modelo de material depende o está en función de dos parámetros fundamentales: **a)** máxima penalización de Brinkman α_{max} y **b)** factor de penalización q. El estudio de la variación de estos parámetros es necesario en la obtención de nuevos diseños que pueden no ser intuitivos; y, que pueden ser considerados en la manufactura final. En este apartado, se analiza solo la variación del parámetro q, debido a que el autor realizó la variación de la penalización de Brinkman α_{max}, y las topologías obtenidas fueron similares a las del parámetro q. Cabe mencionar que el factor de penalización q para este estudio, debe elegirse entre un valor de 0.01 y 1, debido a que penalizaciones más altas restringiría el espacio de búsqueda en la minimización de nuestra función.

En las Figura 5.9 y Figura 5.10, se observan los resultados obtenidos de las topologías finales y el campo de velocidad respectivamente. En la Tabla 5.7 se muestran los desempeños de estas topologías. Se puede notar que la variación de este parámetro de penalización q, tiene incidencia significativa en las topologías finales obtenidas. Se puede ver que el radio de curvatura del álabe se ve afectado, es decir, una penalización baja de $(q = 0.01)$ induce a topologías finales con mayor radio de curvatura, en cambio cuando la penalización es igual a $(q = 1)$, se reduce significativamente el radio de curvatura. Un aspecto que es importante destacar en la topología de la Figura 5.8a $(q = 0.01)$, es que este tipo de impulsor es similar a los impulsores “tipo Vortex”, impulsores que se emplean generalmente para el manejo de fluidos con ciertas fibras o elementos abrasivos. El funcionamiento de este tipo de impulsores se basa en la creación de un torbellino en el interior, para así aumentar la presión del fluido. Este tipo de impeler se lo puede encontrar en bombas de achique o también conocidas como bombas sumergibles. Adicionalmente, cabe mencionar que radios de curvaturas muy pronunciados no siempre favorecen a la conservación de energía del fluido, ya que secciones bruscas pueden a inducir a la formación de vórtices en el fluido, siendo perjudicial para el rendimiento de la bomba.

Tabla 5.7.- Desempeño de las topologías para diferentes factores de penalización q, para la energía de disipación como función objetivo: **a)** q = 0.01, **b)** q = 0.1, **c)** q = 0.5, **d)** q = 1.

Parámetro	**Factor de penalización** q			
Desempeño:	0.01	0.1	0.5	1
Fracción de volumen final	0.250	0.249	0.250	0.249
Energía disipada [Watts]	10.89	7.2746	5.96	5.88

Cabe destacar que el no análisis de estos parámetros del radio de filtro R_{max} y el factor de penalización q pueden repercutir en los diseños finales debido a las cantidades de soluciones que se pueden generar, y no solo esto, sino que la calibración de estos parámetros es fundamental debido a que siempre se buscan parámetros donde se eliminen las escalas de grises, debido a que esto favorece a la convergencia del algoritmo MOT.

Una vez garantizada la calibración y convergencia en los resultados MOT, en la próxima sección se procede a evaluar a la inclusión de la vorticidad como función objetivo, obteniendo así una función bi-objetivo con el fin de modificar los pesos w_d y w_r de la (Ecc. 3.9) con el objetivo de dar mayor priorización de minimización de una función sobre la otra.

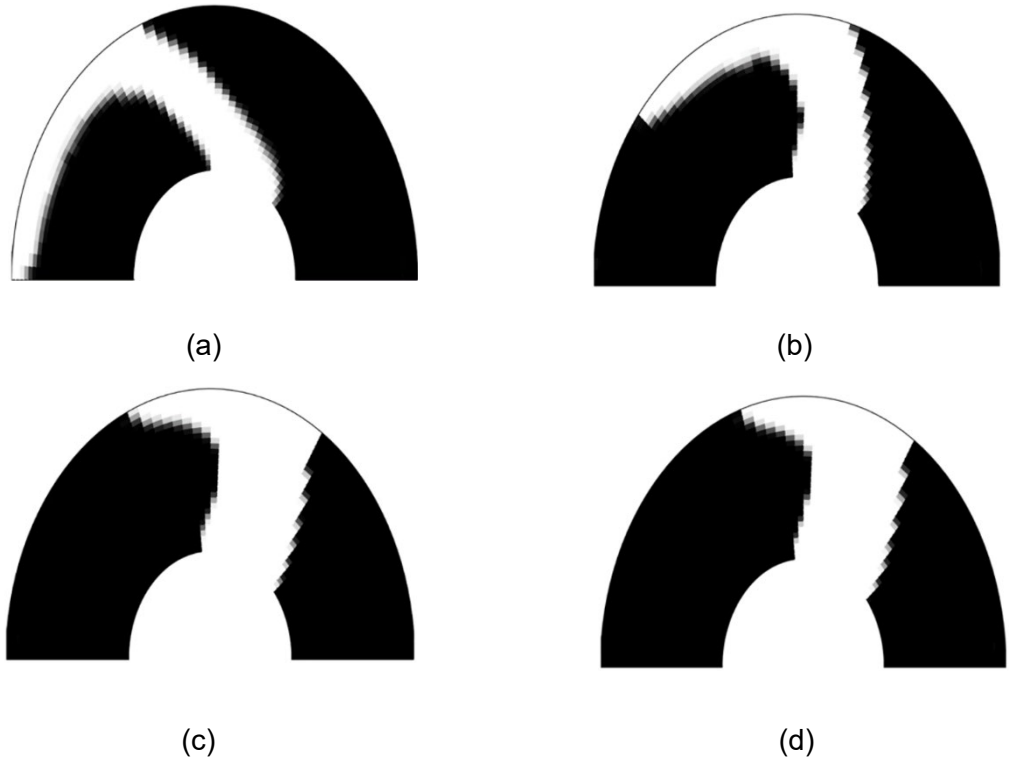


Figura 5.9.- Topologías finales para diferentes factores de penalización q, para la energía de disipación como función objetivo: **a)** q = 0.01, **b)** q = 0.1, **c)** q = 0.5, **d)** q = 1.

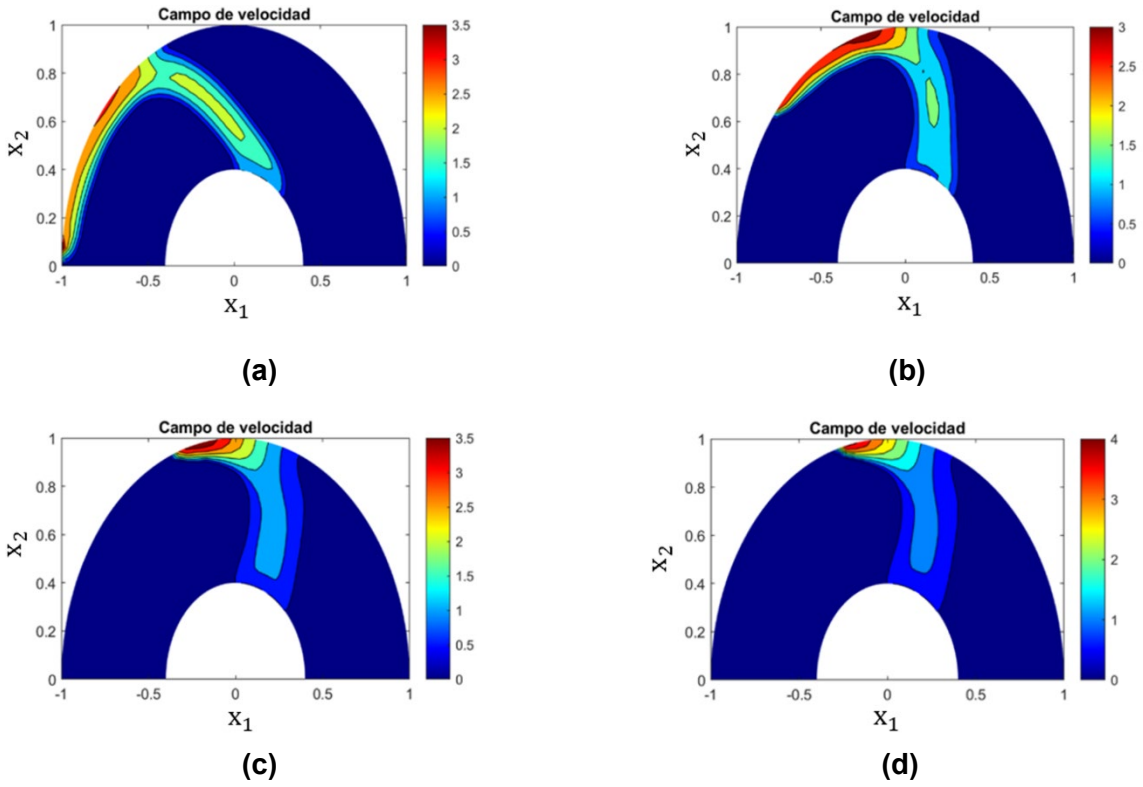


Figura 5.10.- Campo de velocidades de las topologías para diferentes factores de penalización q, para la energía de disipación como función objetivo: **a)** q = 0.01, **b)** q = 0.1, **c)** q = 0.5, **d)** q = 1.

5.3.3- MOT BI-OBJETIVO: CONSIDERANDO A LA E. DE DISIPACIÓN Y VORTICIDAD

En la sección anterior se presentaron diferentes resultados para el problema MOT aplicado al diseño de álabes en turbomáquinas considerando únicamente la minimización de la energía de disipación viscosa como única función objetivo. En esta sección dicho análisis es ampliado con la inclusión de la función de vorticidad en una función bi-objetivo. De esta manera, en la Figura 5.11 se presenta la evolución de la vorticidad en el problema de referencia en el que se minimiza solo la disipación de energía (curva de convergencia de la topología de la Figura 5.4). Así, resulta evidente que el comportamiento de la vorticidad presenta muchas fluctuaciones. Esto claramente se traduce de manera que la minimización de una función no implica la minimización de la otra. Adicionalmente, un análisis claro del porqué de estas fluctuaciones de la vorticidad en toda la curva de convergencia del algoritmo MOT puede deberse específicamente por dos situaciones principalmente: **a)** en las primeras iteraciones se observa que la vorticidad alcanza un pico para posteriormente ir disminuyendo. Esto puede originarse debido a que el flujo es poroso (elementos con variables de diseño intermedias) y no se ha alcanzado la completa definición del canal entre los álabes que conecta la entrada y la salida del fluido; y, **b)** la tendencia de incremento en la curva de convergencia de la función de vorticidad en las últimas iteraciones se da (a pesar de que no es un valor de vorticidad significativo) debido a que cuando el álabe termina de formarse,

este presenta cierto radio de curvatura, pudiendo así ocasionar cambios bruscos en la conducción del fluido y favoreciendo así la generación de vórtices en el dominio de fluido.

Este análisis de la curva de convergencia del algoritmo MOT tiene su importancia debido a que en esta sección se estudia la influencia que tiene de incluir la vorticidad en la topología final del álabe en el diseño de turbomáquinas. Es así, que la minimización de esta función de vorticidad implica la reducción de los gradientes de velocidad en el dominio del fluido, esto es, la minimización de cada uno de los gradientes cruzados de la velocidad puede implicar la reducción en su diferencia, lo que se traduce a una limitación del fluido a rotar.

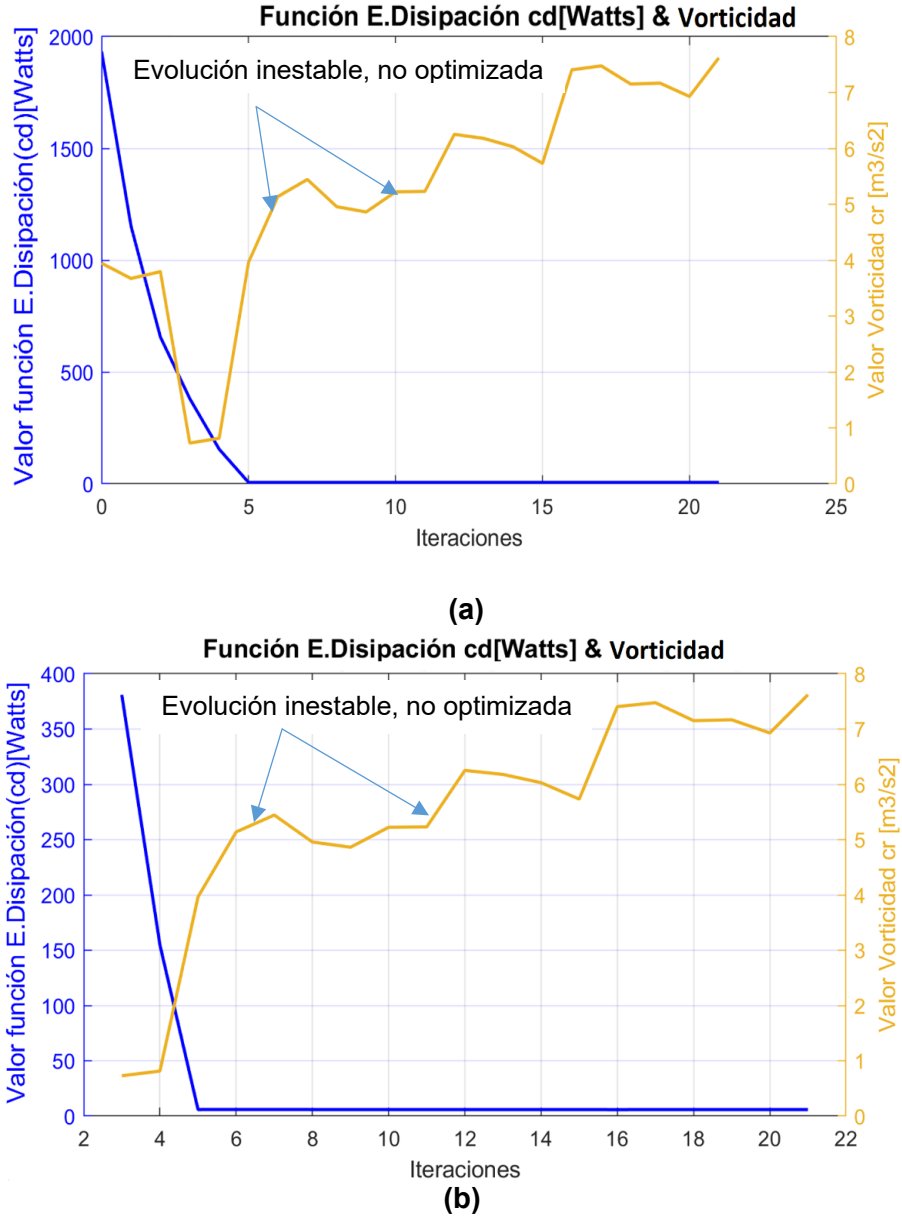


Figura 5.11.- Evolución de la vorticidad del algoritmo MOT en la minimización de la energía de disipación en el diseño de álabes en turbomáquinas ($w_d = 1$ y $w_r = 0$): **a)** Total de iteraciones, **b)** A partir de la iteración 3 (mejor escala).

Es así, que en la Figura 5.12 se obtiene la topología del álabe con el efecto de incluir la vorticidad en la función objetivo, siguiendo la ponderación $w_d = 0.8$ y $w_r = 0.2$. Se puede ver que este diseño final de topología es un poco similar al diseño topológico cuando se considera la minimización de la disipación de energía Figura 5.4a, con la variación más destacada en que el canal de flujo tiene un menor radio de curvatura en el lado izquierdo y una sección más recta en el alabe en la esquina superior derecha.

La curva de convergencia de la función bi-objetivo para este problema se presenta en la Figura 5.13, con un valor final en la función bi-objetivo de 5.948 [Watts] y una fracción volumétrica de 0.2491. Asimismo, al observar la evolución de cada una de las variables de respuesta (ver Figura 5.14 y Figura 5.15), resulta evidente que la vorticidad comienza a tener un papel importante en el resultado final de la topología del álabe, a pesar de que el mayor peso ponderado lo tiene la energía de disipación. Aun así, a pesar de que el peso de ponderación de la vorticidad es de w_r = 0.2, se puede observar en la curva de convergencia de la Figura 5.15b que la vorticidad comienza a tener una tendencia similar de evolución a la curva de la función bi-objetivo.

Adicionalmente, un punto que se debe tomar en cuenta es que si se observa la curva de convergencia bi-objetivo de la Figura 5.13 y se lo compara con la curva de convergencia en donde solo se minimiza la energía de disipación (Figura 5.5), se observa que la curva bi-objetivo tiene una tasa de convergencia mucho más lenta, esto se debe porque para garantizar la convergencia de la función bi-objetivo se debe garantizar la estabilidad de dos funciones objetivos y ya no de una.

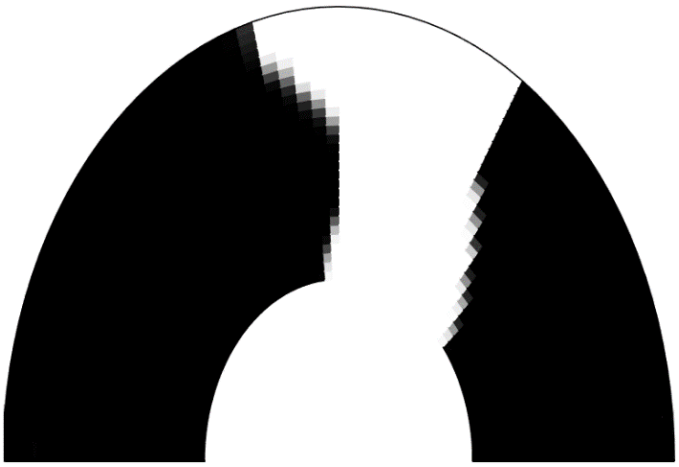


Figura 5.12.- Topología final MOT bi-objetivo incluyendo el efecto de la vorticidad. Ponderación utilizada $w_d = 0.8$ y $w_r = 0.2$.

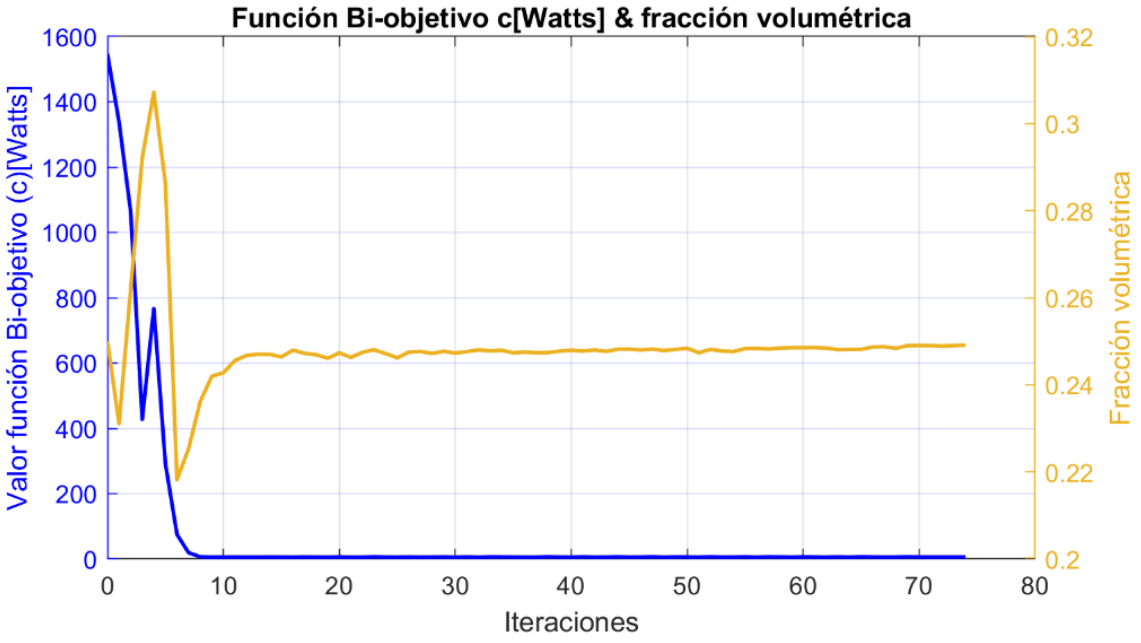


Figura 5.13.- Curva de convergencia para MOT bi-objetivo, incluyendo el efecto de la vorticidad. Ponderación utilizada $w_d = 0.8$ y $w_r = 0.2$.

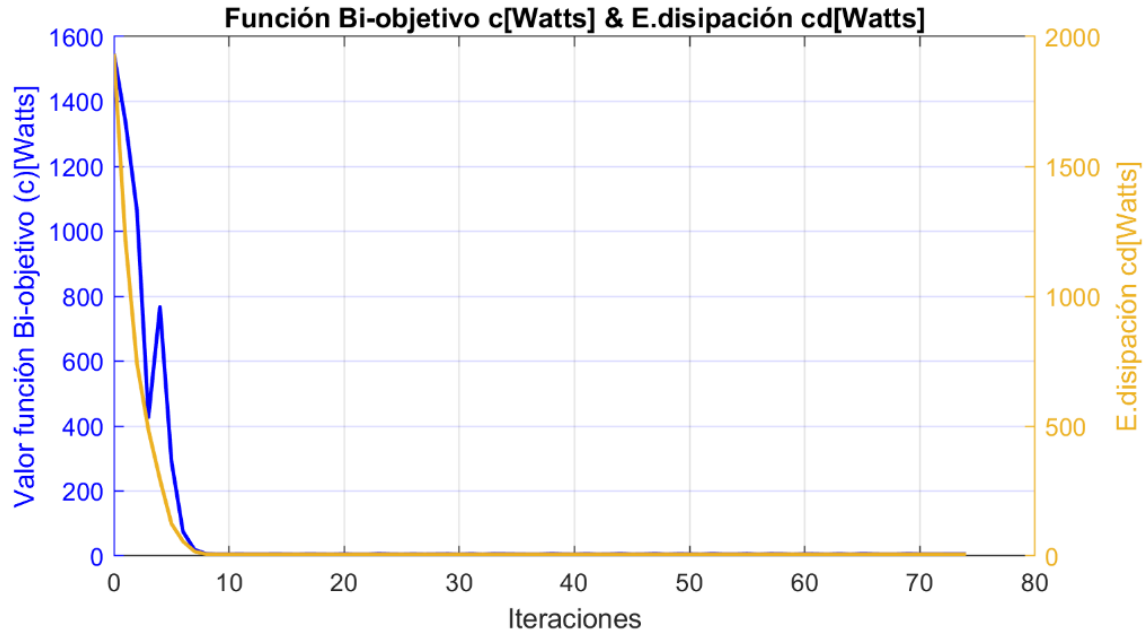
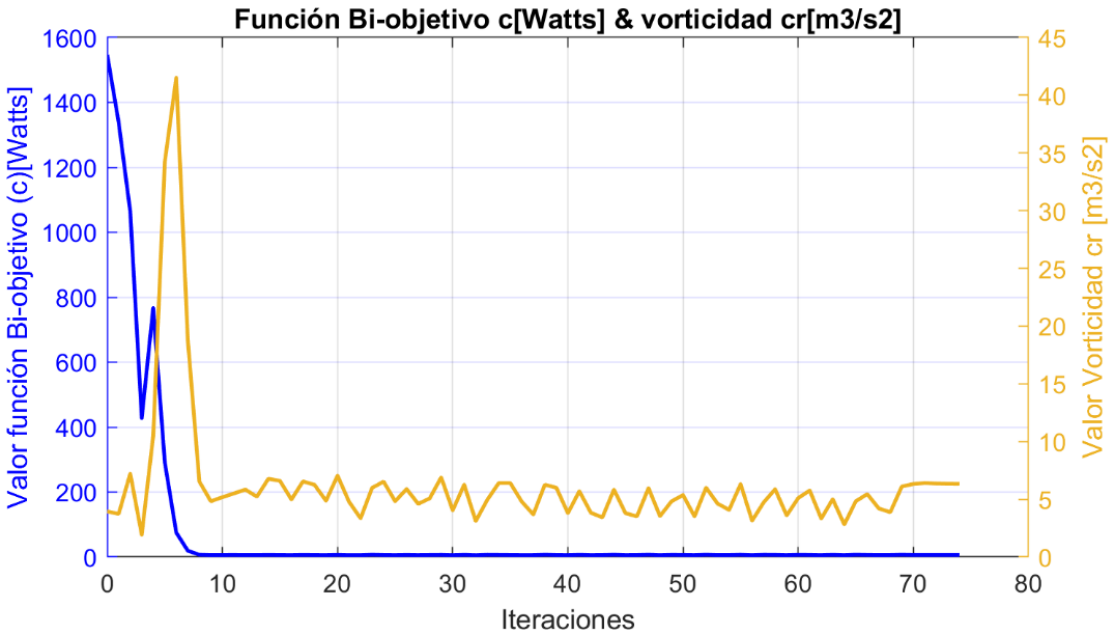
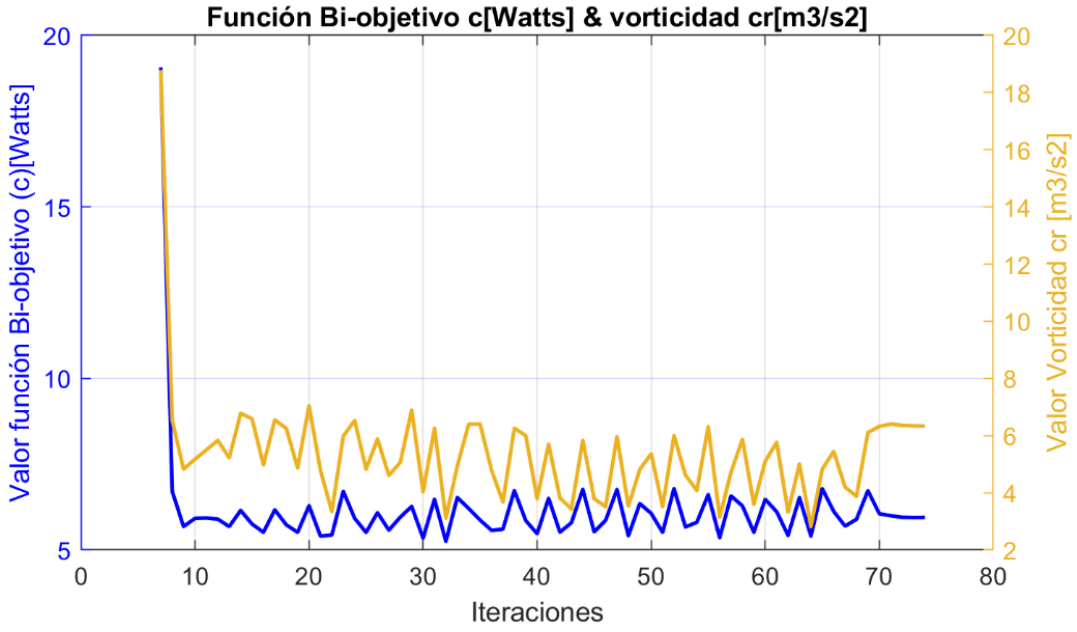


Figura 5.14.- Evolución de la disipación de energía para MOT bi-objetivo, incluyendo el efecto de la vorticidad. Ponderación utilizada $w_d = 0.8$ y $w_r = 0.2$.



(a)



(b)

Figura 5.15.- Evolución de la vorticidad para MOT bi-objetivo. Ponderación utilizada $w_d = 0.8$ y $w_r = 0.2$. **a)** Total de iteraciones, **b)** A partir de la iteración 9 (mejor escala).

De manera similar al problema anteriormente descrito donde se incluyó la función de vorticidad ($w_r = 0.2$) a la función bi-objetivo, se decide ahora aumentar el peso de ponderación de la vorticidad para observar la incidencia de esta función sobre la topología final. En la Figura 5.16, se muestra la topología final del álabe siguiendo la ponderación $w_d = 0.6$ y $w_r = 0.4$, la cual converge a un valor de en la función bi-objetivo de 5.89 [Watts] y una fracción volumétrica de 0.2489. En esta topología se puede apreciar un álabe incluso más recto si lo comparamos con el resultado anterior. Asimismo, al observar las curvas de convergencia de la función bi-objetivo (ver Figura 5.17) y la evolución de la energía de disipación (ver Figura 5.18) y vorticidad (ver Figura 5.19), se observa que la vorticidad comienza a ser predominante, debido a que la ponderación es muy similar al de la energía de disipación.

Un aspecto que se destaca en el diseño final de estas topologías cuando se incluye la vorticidad en la función bi-objetivo es que se reduce de cierto modo la curvatura por donde conducirá el fluido. Sin embargo, a pesar de esta reducción siempre se debe garantizar de cierto modo un radio de curvatura en el álabe. Este radio de curvatura en el álabe está garantizado matemáticamente por las ecuaciones del MEF, donde las ecuaciones de Navier-Stokes están regidas por términos como la fuerzas de Coriolis y fuerza Centrífuga, fuerzas que expresan físicamente la rotación del fluido.

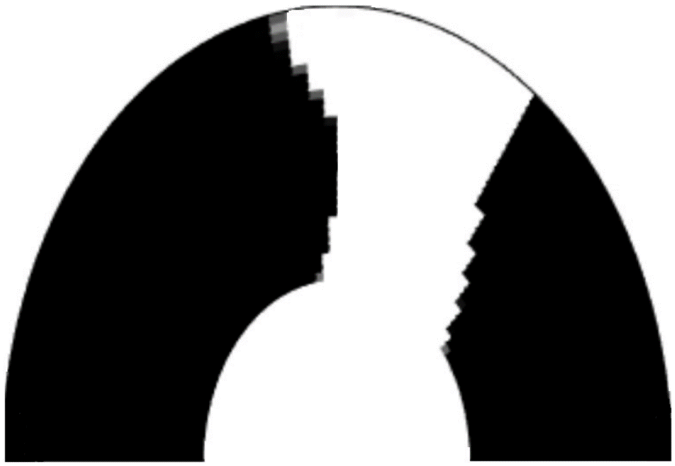


Figura 5.16.- Topología final MOT bi-objetivo incluyendo el efecto de la vorticidad. Ponderación utilizada $w_d = 0.6$ y $w_r = 0.4$.

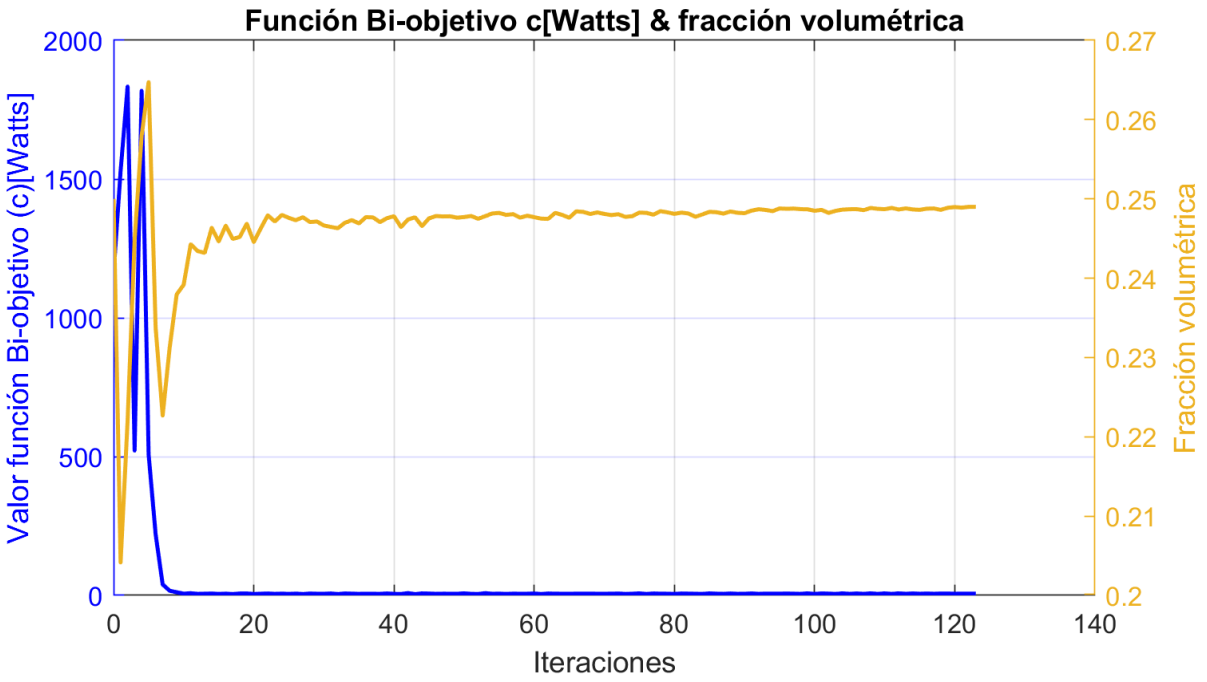


Figura 5.17.- Curva de convergencia para MOT bi-objetivo, incluyendo el efecto de la vorticidad. Ponderación utilizada $w_d = 0.6$ y $w_r = 0.4$.

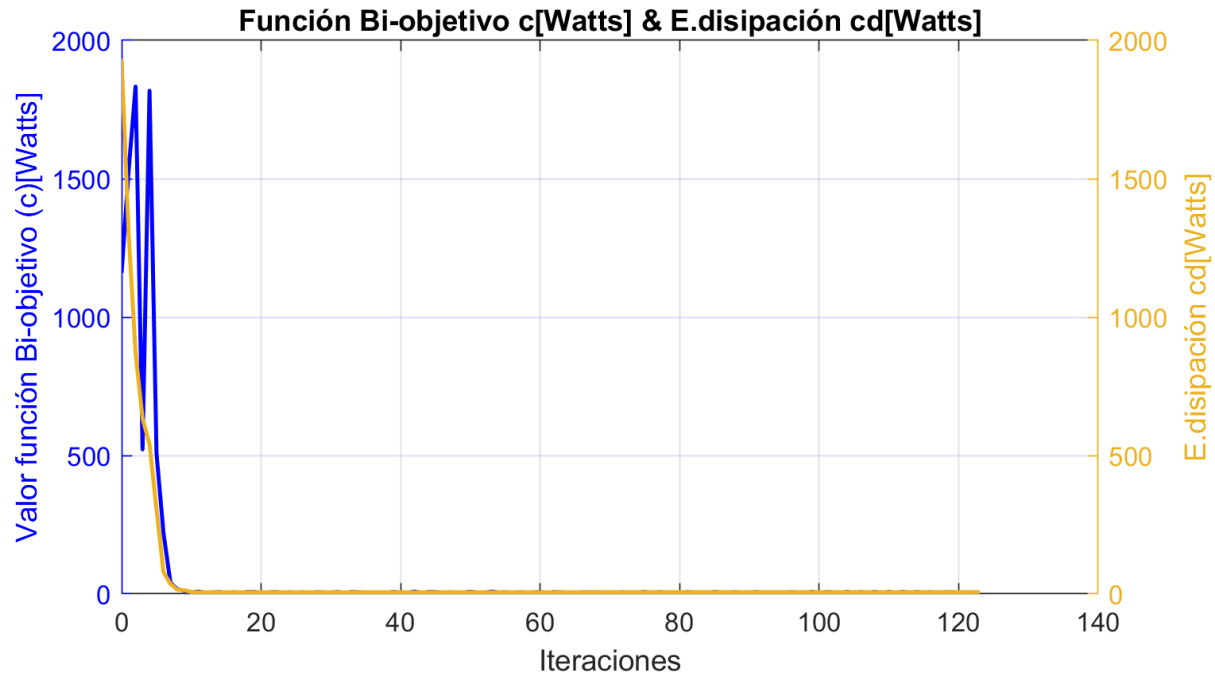
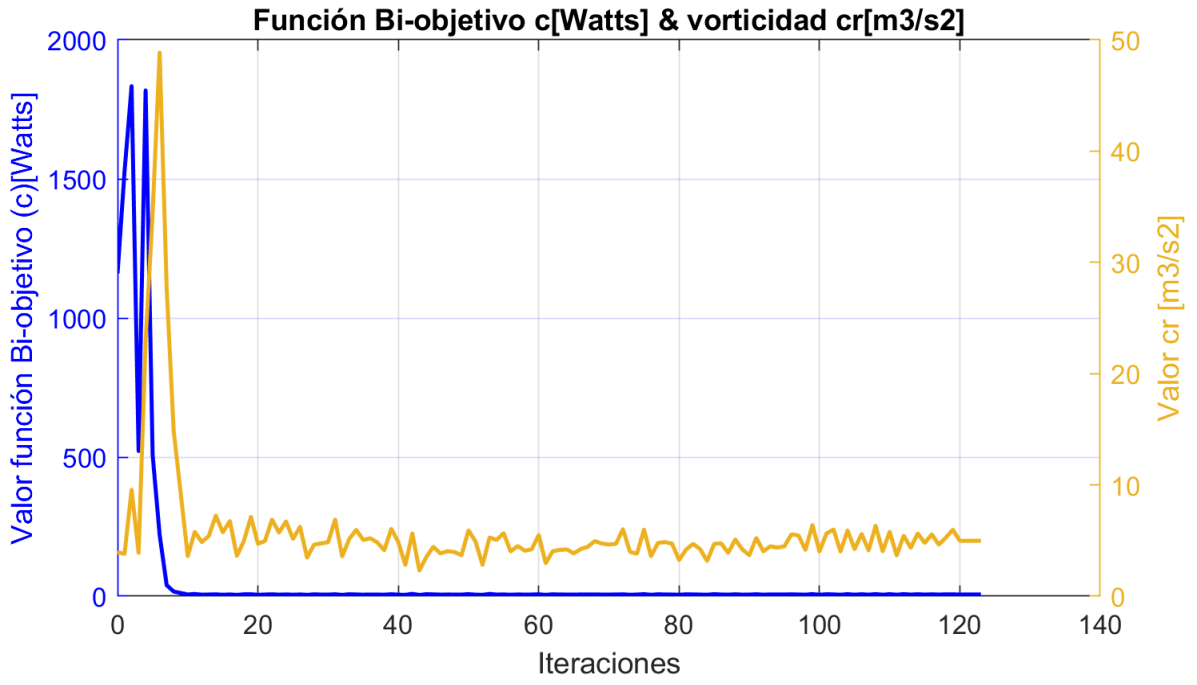
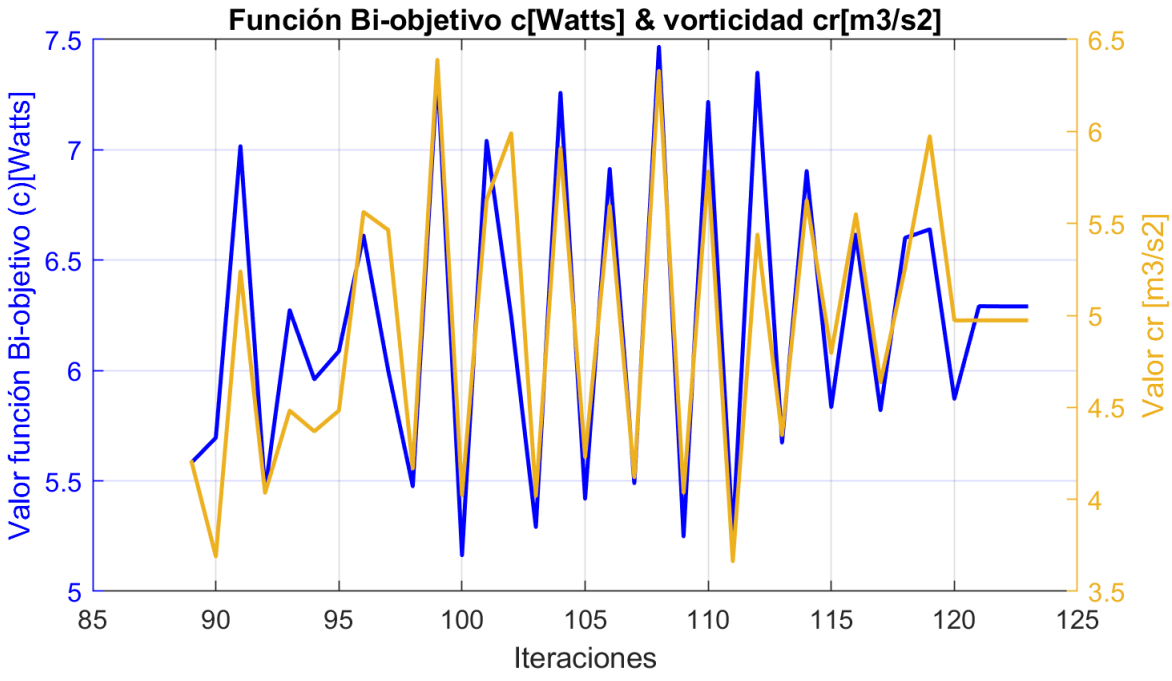


Figura 5.18.- Evolución de la disipación de energía para MOT bi-objetivo, incluyendo el efecto de la vorticidad. Ponderación utilizada $w_d = 0.6$ y $w_r = 0.4$.



(a)



(b)

Figura 5.19.- Evolución de la vorticidad para MOT bi-objetivo. Ponderación utilizada $w_d = 0.6$ y $w_r = 0.4$. **a)** Total de iteraciones, **b)** A partir de la iteración 89 (mejor escala).

Finalmente, en la Tabla 5.8 se presentan los desempeños para las topologías correspondientes a cada una de las ponderaciones bi-objetivo evaluadas. Así, se puede observar que el algoritmo MOT incorpora adecuadamente a cada una de las variables de respuesta y las optimiza de acuerdo con su ponderación. En este punto se destaca que no es posible obtener resultados adecuados cuando se favorece completamente a la vorticidad (por ejemplo: $w_d = 0$ y $w_r = 1$). Esto se debe principalmente porque el modelo implementado en este trabajo parte de una configuración de flujo en un canal donde la función principal es la energía de disipación viscosa, y en base a la formulación original de Borrvall y Peterson (2003), la (Ecc. 3.4) depende de forma directa de la variable de diseño asociada a la matriz $\boldsymbol{k}^E$ de cada elemento finito. Es así, que en la minimización de la función bi-objetivo, la minimización de la energía de disipación viscosa tiene relación directa con la formación del canal, es decir, la conexión entre la entrada del fluido y la salida, en cambio la vorticidad de cierta manera solo modifica el contorno del álabe.

Tabla 5.8.- Desempeño de topologías finales para MOT bi-objetivo.

Desempeño topología final	**Ponderación**	
	w_d	w_r
Función bi-objetivo [W]:		
5.88	1.0	0.0
5.95	0.8	0.2
6.29	0.6	0.4
6.54	0.5	0.5
6.76	0.2	0.8
Energía disipada [W]:		
5.88	1.0	0.0
5.95	0.8	0.2
6.27	0.6	0.4
6.26	0.5	0.5
5.79	0.2	0.8
Vorticidad $[m^3/s^2]$		
7.61	1.0	0.0
6.33	0.8	0.2
4.99	0.6	0.4
5.30	0.5	0.5
4.91	0.2	0.8

5.3.4- POSPROCESAMIENTO DE ROTORES DE TURBOMÁQUINAS

Los resultados anteriores representan los diseños de álabes y a la vez la geometría del canal base por donde se conduce el fluido en media circunferencia de rotor para diferentes exigencias de minimización en la función objetivo. Por lo tanto, para evaluar el desempeño completo de estas topologías, en esta sección se decide extender los resultados del álabe en una circunferencia completa, simulando el impeler de una bomba centrífuga. Es así como en la Figura 5.20 se presentan los rotores abiertos construidos con un diseño de 5 álabes, para topologías de los resultados de cada uno de los siguientes casos: Caso 1: Función mono-objetivo en la minimización de energía de disipación ($w_d = 1$ y $w_r = 0$) (Figura 5.4a); Caso 2: función bi-objetivo en la minimización de energía de disipación y vorticidad ($w_d = 0.8$ y $w_r = 0.2$) (Figura 5.12); y Caso 3: función bi-objetivo en la minimización de energía de disipación y vorticidad ($w_d = 0.6$ y $w_r = 0.4$) (Figura 5.16). La elección de cinco álabes en el rotor se da debido a que es la cantidad máxima que ocupa en una circunferencia completa.

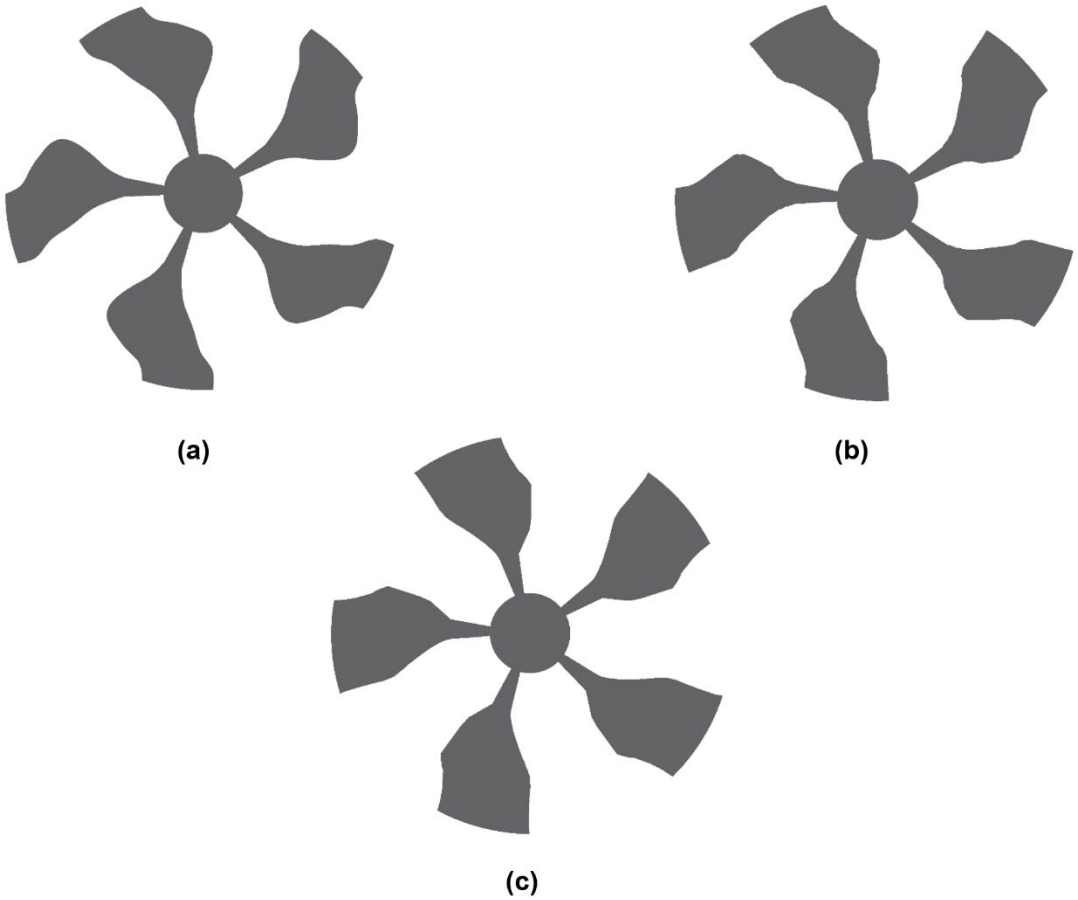


Figura 5.20.- Diseño de impulsores para el modelo de bomba centrífuga de pequeña escala considerando 5 álabes: **a)** Caso 1: Función mono-objetivo (minimización de energía de disipación $w_d = 1$ y $w_r = 0$), **b)** Caso 2: función bi-objetivo (minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) y **c)** Caso 3: función bi-objetivo (minimización de energía de disipación y vorticidad, $w_d = 0.6$ y $w_r = 0.4$).

Una vez obtenida la sección transversal de los impulsores, el diseño puede ser llevado a características constructivas más complejas, como el diseño de un impeler cerrado o semiabierto con la inclusión de todas las partes mecánicas que conforman en sí a este equipo rotativo (rodamientos axiales, diseño del eje, sellos mecánicos, venteos, etc.), con el fin de poder realizar una evaluación final de desempeño para cada parte en específico. Pero en este trabajo se toma el problema de diseño de rotor (impulsor) de una bomba a pequeña escala donde el diámetro externo del rotor es de 4 cm, modelo similar al trabajo realizado por Sá (2016) [26], en el prototipado de bombas para asistencia ventricular.

Es así, que el diseño del rotor es complementado con la inclusión de una carcasa (voluta) donde las porciones de fluido que corresponden al interior de la bomba se pueden observar en la Figura 5.21, donde el impulsor seleccionado para este modelo corresponde al Caso 1. Para este problema en particular, se considera agua como fluido, donde en la sección de succión se tiene una velocidad de entrada de 1 [m/seg] (equivalente a un flujo másico en la entrada de 0.12 [kg/seg]) y en la sección de descarga se tiene una presión de salida de 0 [Pa]. Se considera la sección periférica de la voluta como pared y se asigna una velocidad de rotación angular en el rotor de 1.815 r.p.m. En la Figura 5.22, se muestra la discretización de malla para el modelo planteado, la cual está construida por 97.165 celdas previo al análisis de simulación de CFD en el software comercial ANSYS FLUENT. Cabe mencionar que en estos resultados no es necesario ilustrar la selección de los criterios de los esquemas de solución del método numérico debido a que el objetivo principal de este análisis es de caracterizar las diferentes opciones de impulsores que se pueden tener en el diseño de estas máquinas rotativas.

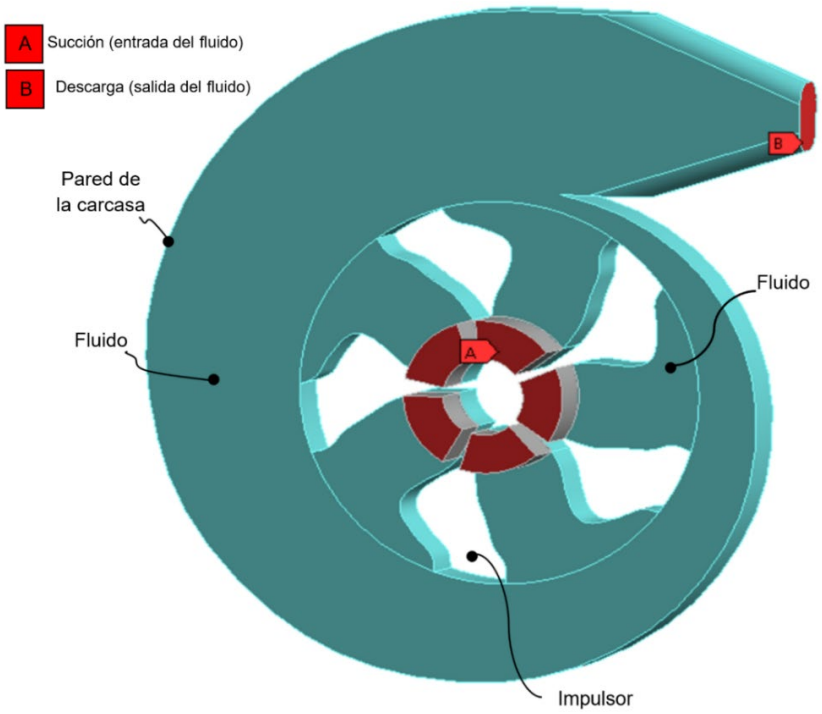


Figura 5.21.- Modelo de bomba centrífuga a pequeña escala para el impulsor del Caso 1: Función mono-objetivo (minimización de energía de disipación $w_d = 1$ y $w_r = 0$).

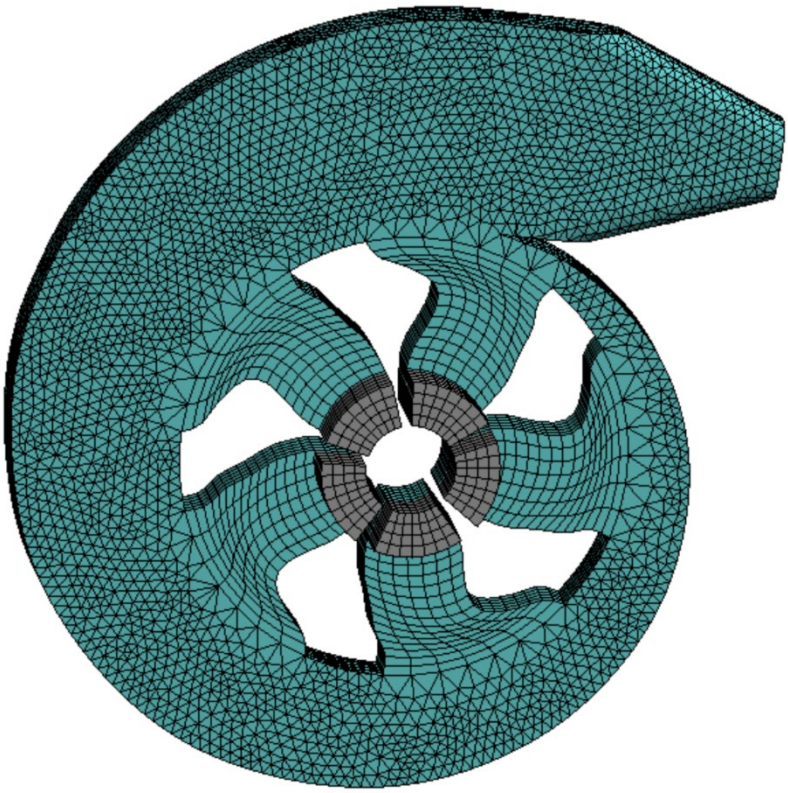


Figura 5.22.- Malla de discretización de volúmenes finitos para el modelo de bomba centrífuga a pequeña escala para el impulsor del Caso 1: Función mono-objetivo (minimización de energía de disipación $w_d = 1$ y $w_r = 0$).

Una vez que se tienen los tres modelos de bombas con su respectiva malla, se decide ahora obtener los campos de presiones y de velocidades, con el objetivo de poder definir métricas de desempeño del fluido de forma que podamos validar las minimizaciones de las funciones objetivos que se consideraron en la ponderación de los pesos. Es así, que en la Figura 5.23 y Figura 5.24 se muestran los campos de presiones y velocidades respectivamente para cada uno de los casos propuestos. En la Figura 5.23 se presentan los valores de la presión en la succión de la bomba y la suma total de la presión relativa en el fluido, lo cual se evidencia que la bomba del Caso I es la que posee el valor más alto de presión y efectivamente es en este caso topológico donde se dio priorizo la minimización de la energía de disipación viscosa. Adicionalmente cabe destacar el efecto que tiene la voluta en el diseño de estas bombas centrífugas, siendo notorio que la función principal de esta carcasa en forma de espiral es de recoger el fluido propulsado radialmente por el impeler, dirigiéndolo hacia la tubería de descarga. Otra función de esta carcasa es la de contribuir al incremento de presión del fluido dentro de la bomba, lo cual aparte de poder ser evidenciado esta alza de energía del fluido en estas figuras desde la succión hasta su descarga, Nogueira SÁ (2016) [26] incluyó en sus simulaciones de rotores, bombas con voluta y sin voluta, obteniendo menores perdidas de energía de disipación cuando incluyó esta carcasa en forma de espiral.

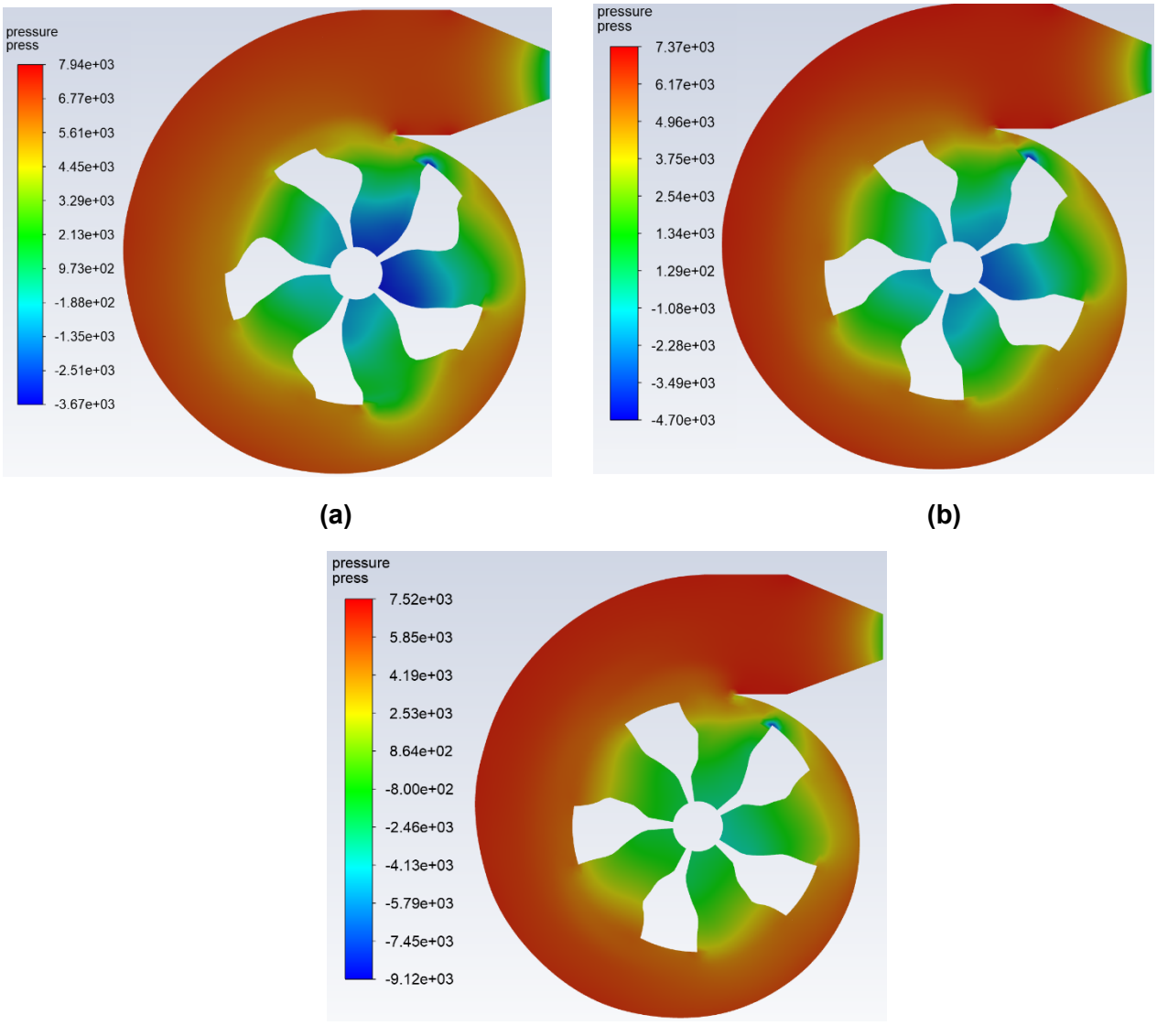


Figura 5.23.- Campo de presiones para el problema de los rotores de bombas a pequeña escala. Donde: **a)** Caso I, Presión succión: -2297 [Pa] Presión total: 3.90 [GPa], **b)** Caso II, Presión succión: -2787 [Pa] Presión total: 3.75 [GPa] y **c)** Caso III, Presión succión: -1944 [Pa] Presión total: 3.56 [GPa].

De la misma manera, en la Figura 5.24 se muestran los valores de vorticidad total en todo el dominio de diseño, con el fin de poder evidenciar que efectivamente el menor valor de vorticidad se obtiene para el caso III, donde se dio un mayor peso de ponderación en comparación con los otros dos casos.

Adicionalmente un aspecto que se puede incorporar en estos modelos es el análisis estructural para el rotor del impeler, seleccionando así un material adecuado para este tipo de dispositivos con el objetivo de poder así evaluar los esfuerzos que se generan. Actualmente las empresas que se dedican a la fabricación de estas máquinas rotativas incorporan en sus diseños no solo este tipo de análisis, sino que también incorporan en ellos los análisis de frecuencias naturales de vibración donde se miden las amplitudes de estas vibraciones con el objetivo de realizar una evaluación técnica de estas máquinas y poder así reducir estas frecuencias.

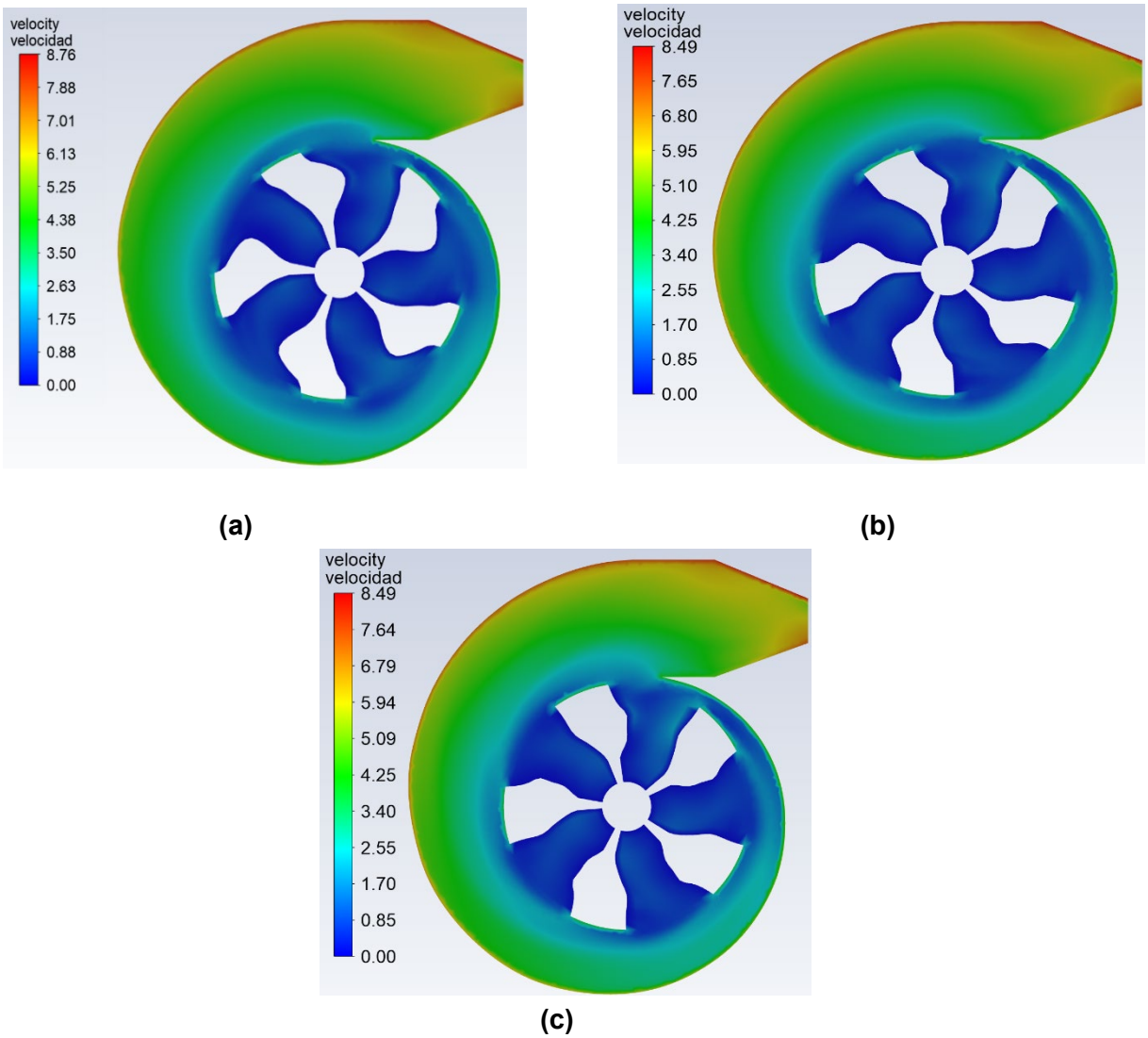


Figura 5.24.- Campo de velocidades para el problema de los rotores de bombas a pequeña escala. Donde: **a)** Caso I, Vorticidad Total: $7.82x10^7$ [1/seg], **b)** Caso II, Vorticidad Total: $6.56x10^7$ [1/seg] y **c)** Caso III, Vorticidad Total: $5.01x10^7$ [1/seg].

5.4.- DESEMPEÑO DE LOS ALGORITMOS COMPUTACIONALES EN SERIAL Y EN PARALELO USANDO COMPUTACION EN LA NUBE

Una vez obtenido los resultados topológicos y de posprocesamiento, en esta sección se decide evaluar en términos escalabilidad y eficiencia, el algoritmo computacional para el diseño de álabes en turbomáquinas. Para realizar este análisis, el autor decide dividir esta sección en dos partes: **a)** en la sección 5.4.1 donde se evalúa el desempeño del algoritmo computacional solo del MEF en la ejecución del álabe recto (ver Figura 2.4) incrementando la cantidad de elementos finitos e incrementado la cantidad de núcleos de la CPU en la nube y **b)** en la sección 5.4.2 donde se evalúa el algoritmo computacional MEF y MOT para el Caso I y Caso II (ver Figura 5.4a y Figura 5.12) para el diseño del álabe en turbomáquinas en media circunferencia del rotor, manteniendo una malla fija de 1.800 elementos y el incremento de la cantidad de núcleos de la CPU en la nube. Cabe destacar que para estos análisis de desempeño de los algoritmos computacionales que se realizan en esta sección, se utilizan las arquitecturas de CPU tanto para la máquina virtual proporcionada por AWS y máquina baremetal proporcionada por Equinix que se detallaron en la sección 4.6.2.

5.4.1- DESEMPEÑO DEL ALGORITMO MEF APLICADO AL ÁLABE RECTO

En esta sección se presenta el desempeño del algoritmo MEF del tiempo total en serial y en paralelo ejecutado en la nube. Cabe mencionar que son tres “for” que se paralelizan para el modelo del alabe recto de la Figura 2.4: **a)** el primer “for” es similar al del Cuadro 4.1., con la variante de que para este modelo de alabe recto solo es necesario el cálculo de la matriz $\boldsymbol{k}^E$ y el vector de constantes $\boldsymbol{b}^E$, excluyendo así el cálculo y el almacenamiento de la matriz de vorticidad $\boldsymbol{M}_r$ para todos los elementos, debido a que en este algoritmo no son necesarias las funciones objetivos por la razón de que el MOT no es considerado en este modelo, **b)** el segundo “for” paralelizado es el que se encuentra en el Cuadro 4.2 y **c)** y el tercer “for” es el que se muestra en el Cuadro 5.1. Este último “for” paralelizado guarda en vectores filas los resultados de los valores nodales de las presiones, debido a que esto es necesario para graficar el contorno de presión que se muestra en la Figura 5.2.

Cuadro 5.1.- Almacenamiento de los resultados de los valores nodales de las presiones en vectores filas previo a la graficación del contorno de presión en el dominio de diseño del modelo del álabe recto
parfor i=1:Neta+1 Almacenamiento de los valores nodales de las presiones en vectores filas fin

En la Tabla 5.9 y Tabla 5.10 se presentan los tiempos computacionales totales y el desempeño del algoritmo MEF a medida que se va incrementando la cantidad de elementos finitos tanto para las arquitecturas de CPU de la máquina virtual proporcionada por AWS y de la máquina baremetal proporcionada por Equinix respectivamente. Cabe destacar que un buen desempeño del algoritmo computacional paralelizado radica esencialmente en la aceleración (speed-up), es decir, que lo ideal sería obtener una aceleración igual o mayor a la cantidad de número de procesadores utilizados, para así obtener una escalabilidad lineal o exponencial respectivamente. Este concepto anterior se puede ver reflejado si se toma como ejemplo el caso de 6.400 elementos finitos para la Tabla 5.9, donde se puede observar que a partir de utilizar 16 núcleos en paralelo lo ideal sería obtener una aceleración de 16x o mayor, pero la aceleración que se obtiene es de 12.30x, lo cual se puede decir que es una aceleración no deseada. En cambio, si se observa el caso de 78.400 elementos finitos para esta misma tabla, se puede notar que comienza a existir una ligera desaceleración cuando se utiliza 24 núcleos, dando un valor de aceleración de 22.26x. Estos dos ejemplos clarifican el hecho de que la cantidad de datos que se tienen que procesar tiene influencia directa sobre la eficiencia del algoritmo paralelizado, es decir, que para cierta cantidad fija de elementos finitos siempre va a existir una desaceleración del algoritmo con cierta cantidad de núcleos. Para visualizar de mejor manera estos desempeños, se decide graficar las aceleraciones vs. el número de procesadores para los diferentes números de elementos finitos (6.400, 19.600, 40.000 y 78.400). Es así, que en la Figura 5.25 se observa la escalabilidad del algoritmo para los diferentes números de elementos finitos tanto para las arquitecturas de CPU en la máquina virtual de AWS y la máquina baremetal de Equinix.

Para los casos de 6.400, 19.600 y 40.000 elementos finitos para AWS, se observa que la desaceleración comienza a partir de utilizar 8 núcleos en paralelo en cambio que para el caso de 78.400 elementos finitos se puede notar que existe una ligera desaceleración recién cuando se utilizan 16 núcleos en paralelo. En cambio, para las gráficas en Equinix se puede observar que solo para el caso de 6.400 elementos finitos la desaceleración comienza a partir de utilizar 8 núcleos en paralelo.

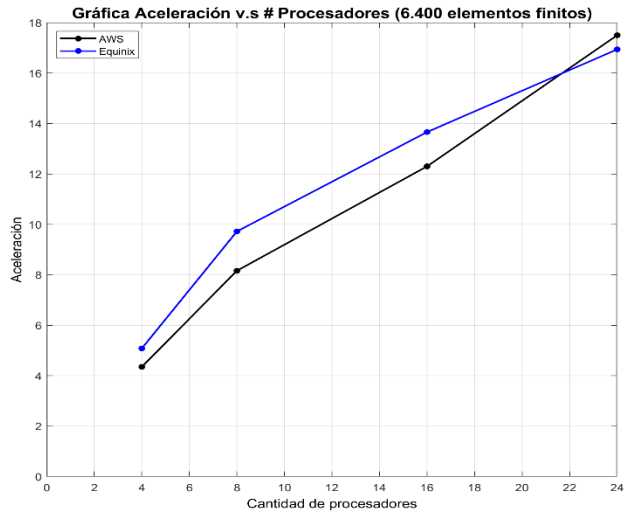
En todo este análisis se destaca que se alcanza una desaceleración más rápida en la máquina de AWS que en la máquina de Equinix, esto se da principalmente por el hipervisor que posee la máquina virtual de AWS, lo cuál de cierta manera desmejora el desempeño de los núcleos en paralelo debido a esta virtualización.

Tabla 5.9.- Tiempo de cómputo total del modelo del álabe recto y desempeño del algoritmo incrementando la cantidad de elementos finitos usando la arquitectura de CPU de la Figura 4.17 (Máquina Virtual proporcionada por AWS en la nube).

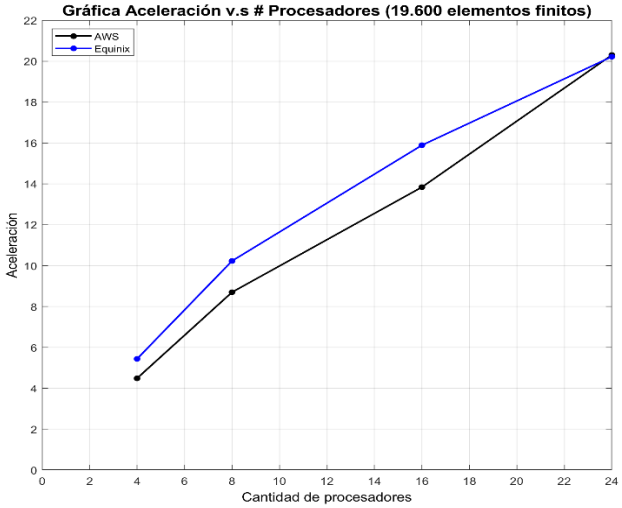
Nr	Nteta	# de elementos finitos	Tiempo total del algoritmo MEF [seg]					Desempeño del algoritmo							
			Serial	Paralelo				Speed-up				Eficiencia			
				Cantidad de núcleos				Cantidad de núcleos				Cantidad de núcleos			
				4	8	16	24	4	8	16	24	4	8	16	24
80	80	6.400	6.769,11	1.555,36	829,95	550,47	386,86	4.35	8.16	12.30	17.50	1.08	1.02	0.77	0.73
140	140	19.600	63.135,32	14.047,88	7.260,22	4.562,01	3.109,62	4.49	8.70	13.84	20.30	1.12	1.08	0.87	0.85
200	200	40.000	245.478	59.980,00	29.801	18.102	12.527,1	4.09	8.24	13.56	19.60	1.02	1.03	0.85	0.82
280	280	78.400	1'058.947	243.957	126.435	67.761	47.571	4.34	8.38	15.63	22.26	1.04	1.05	0.98	0.93

Tabla 5.10.- Tiempo de cómputo total del modelo del álabe recto y desempeño del algoritmo incrementando la cantidad de elementos finitos usando la arquitectura de CPU de la Figura 4.18 (Máquina Baremetal proporcionada por Equinix en la nube).

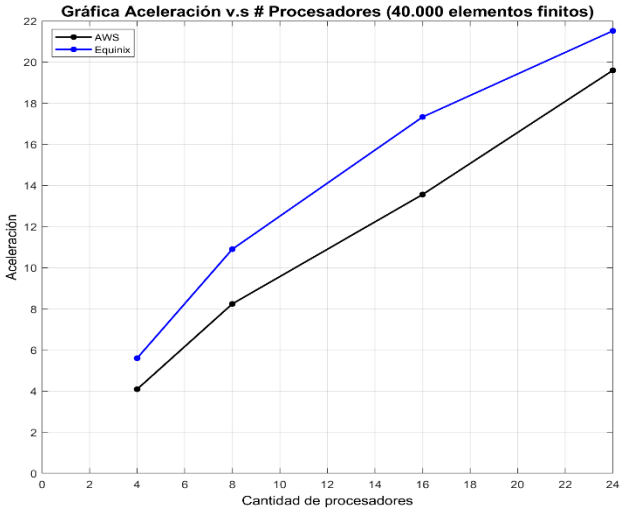
Nr	Nteta	# de elementos finitos	Tiempo total del algoritmo MEF [seg]					Desempeño del algoritmo							
			Serial	Paralelo				Speed-up				Eficiencia			
				Cantidad de núcleos				Cantidad de núcleos				Cantidad de núcleos			
				4	8	16	24	4	8	16	24	4	8	16	24
80	80	6.400	5.646,44	1.109,82	580,99	413,21	333,39	5,08	9,72	13,66	16,94	1,27	1,21	0,85	0,71
140	140	19.600	53.027,7	9.756,61	5.181,24	3.337,60	2.622,67	5,44	10,23	15,88	20,22	1,36	1,28	0,99	0,84
200	200	40.000	228.636	40.828,24	20.969,96	13.196	10.620,2	5,60	10,90	17,33	21,53	1,40	1,36	1,08	0,90
280	280	78.400	882.456	164.836,3	80.149,30	50.545,28	39.755	5,35	11,01	17,46	22,20	1,27	1,38	1,09	0,92



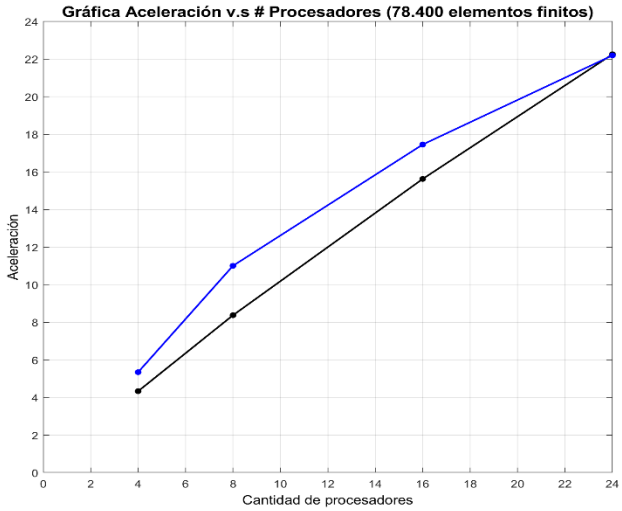
(a)



(b)



(c)



(d)

Figura 5.25.- Gráficas de escalabilidad, aceleración vs. número de procesadores para diferentes números de elementos finitos **a)** 6.400, **b)** 19.600, **c)** 40.000 y **d)** 78.400 utilizando la arquitectura de CPU de AWS y Equinix en la nube.

Adicionalmente, luego de evaluar la escalabilidad del algoritmo computacional para diferentes números de elementos finitos, se decide graficar la eficiencia del modelo del alabe recto de la Tabla 5.9 y Tabla 5.10. En la Figura 5.26, se presentan las gráficas de eficiencia tanto para la arquitectura de CPU proporcionada por AWS y Equinix respectivamente. Cabe destacar que la eficiencia cuya formulación se basa en la (Ecc. 4.4), en sí divide la aceleración total para el número de núcleos utilizados en la paralelización del algoritmo; es decir, es una métrica que determina la mejora que se obtiene por unidad contribuyente (por cada núcleo). En adición, un algoritmo deja de ser eficiente cuando esta métrica se encuentra por debajo de 1 y cuando está por encima de 1 es eficiente. Cabe destacar que esta métrica siempre va a tender a decrecer a medida que se aumente la cantidad de número de procesadores, pero, en el caso que se decida mantener o aumentar en cierto grado la eficiencia, una solución para que esto sea posible sería de buscar alguna otra línea en el algoritmo del MEF que se pueda paralelizar con el fin de mejorar el tiempo en paralelo con respecto al serial y a su vez mejorar la eficiencia del código.

Para los casos de 6.400, 19.600 y 40.000 en AWS se puede observar que el algoritmo es eficiente con 8 núcleos en paralelo y para el caso de 78.400 elementos se puede mencionar que el algoritmo se aproxima a ser eficiente cuando se utiliza 16 núcleos en paralelo. A manera de ejemplificar lo del párrafo anterior con respecto a que la eficiencia es una métrica que determina la mejora por unidad contribuyente, si se toma como ejemplo el caso donde se utiliza 6.400 elementos finitos para 4 núcleos en paralelo, la eficiencia que se obtiene para este caso en específico es de 1.27; es decir, que estaríamos obteniendo una aceleración aproximadamente de 1.27x por cada núcleo.

Adicionalmente, si se analiza el desempeño de estas gráficas con la arquitectura de CPU en Equinix, se puede observar que para el caso de 6.400 elementos finitos el algoritmo es eficiente con 8 núcleos en paralelo, pero si se quisiera utilizar una mayor cantidad de núcleos de manera eficiente para esta misma cantidad de elementos finitos, podemos inferir que una cantidad estimada de núcleos en paralelo sería de 13 (debido a que la eficiencia sigue siendo mayor a 1). Para los casos de 19.600, 40.000 y 78.400 elementos finitos la cantidad de núcleos paralelos estimada que podrían utilizarse de manera eficiente sería de 15, 19 y 20 respectivamente.

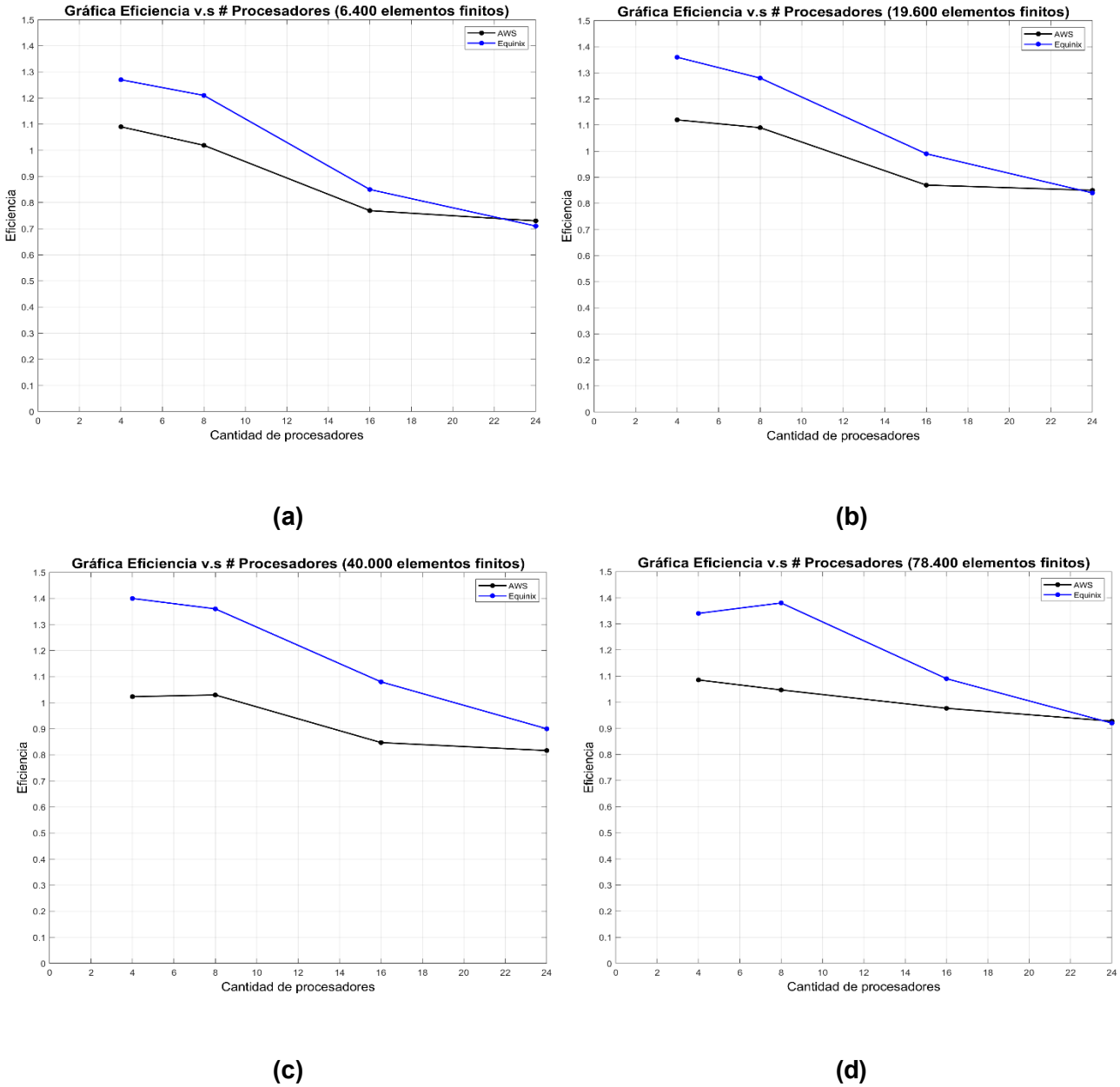


Figura 5.26.- Gráficas de eficiencia para el algoritmo del modelo del álabe recto para diferentes números de elementos finitos: **a)** 6.400, **b)** 19.600, **c)** 40.000 y **d)** 78.400 utilizando la arquitectura de CPU de AWS y Equinix en la nube.

5.4.1.1.- DESEMPEÑO DEL BUCLE “FOR” DEL CUADRO 4.1 DEL MODELO DE ÁLABE RECTO

Una vez obtenido los tiempos totales del algoritmo computacional del modelo del álabe recto, se decide ahora mostrar en la Tabla 5.11 y Tabla 5.12 el desempeño del “for” del Cuadro 4.1 tanto en serial y en paralelo tanto para la máquina virtual en AWS y para la máquina baremetal en Equinix respectivamente, con el objetivo de poder evidenciar la influencia que tiene este “for” sobre el tiempo total del algoritmo computacional. De igual manera se decide mostrar las gráficas de escalabilidad y de eficiencia de este bucle con el objetivo de poder realizar una mejor evaluación de desempeño tanto en serial y en paralelo.

En la Figura 5.27 se muestran las gráficas de escalabilidad para el “for” paralelizable del Cuadro 4.1. del algoritmo del modelo del álabe recto para diferentes números de elementos finitos para la arquitectura de CPU de AWS y Equinix. Para las gráficas de AWS para todos los casos (6.400, 19.600, 40.000 y 78.400), se obtienen aceleraciones por debajo 4x cuando se utiliza 4 núcleos en paralelo. Además, si se analiza el caso de 78.400 elementos finitos con 24 núcleos en paralelo, se observa que la aceleración que se alcanza para este caso es de 18.75x, todavía muy por debajo de 24x. En cambio, si se analiza las gráficas de Equinix para todos los casos, se observa que cuando se utilizan 4 núcleos en paralelo, se alcanza una aceleración por encima de 4x. Lo mismo ocurre cuando se utiliza 8 núcleos en paralelo que la aceleración se encuentra por encima de 8x, a excepción para el caso de 6.400 elementos finitos cuya aceleración es de 7.85x.

Se podría llegar a tener una mejor expectativa de estos resultados de aceleración del bucle del Cuadro 4.1., pero en este punto el autor destaca que en sí los resultados de aceleración son bastantes buenos debido al hecho de que este es el bucle que mayor tiempo computacional le conlleva al algoritmo MEF en ejecutarse. Adicionalmente, la razón del porque este bucle contiene la mayor cantidad de cálculos se debe principalmente porque este ciclo tiene que calcular todos los coeficientes de la matriz $\boldsymbol{k}^E$ y el vector de constantes $\boldsymbol{b}^E$ para cada elemento finito del dominio de diseño; y, para realizar estos cálculos debe iterar n_E veces la cantidad de elementos finitos, lo cual de cierto modo es un proceso que conlleva tiempo y más aún si se incrementa la discretización.

Otro aspecto que se debe mencionar en este bucle es que para cada iteración de elemento finito se tiene que llamar a la función “fluidelementmatrix”, la cual es la función encargada de realizar estos cálculos y almacenarlos. En este punto el autor destaca que otra alternativa de paralelización de este bucle “for” que podría haber favorecido aún más los resultados de aceleración hubiera sido la eliminación de esta función “fluidelementmatrix” y todo el código que contenía esta función ubicarlo en el script general del algoritmo del MEF. Se menciona esto debido al hecho de que los bucles “for” a pesar de que deben cumplir ciertas condiciones específicas (mencionadas en la Sección 4.7) para poder ser ejecutados en paralelo, no siguen un formato trivial de paralelismo; y, es el programador que finalmente decide de qué manera organiza su algoritmo.

Tabla 5.11.- Tiempo de cómputo totales y desempeño del algoritmo del modelo del álabe recto para el "for" paralelizable del Cuadro 4.1. incrementando la cantidad de elementos finitos usando la arquitectura de CPU de la Figura 4.17 (Máquina Virtual proporcionada por AWS en la nube).

Nr	Ntet a	# de elementos finitos	Tiempo total del "for" paralelizable del Cuadro 4.1. [seg]					Desempeño del algoritmo							
			Serial	Paralelo				Speed-up				Eficiencia			
				Cantidad de núcleos				Cantidad de núcleos				Cantidad de núcleos			
				4	8	16	24	4	8	16	24	4	8	16	24
80	80	6.400	5.070	1.491,10	754,71	514,12	362,06	3,40	6,72	9,86	14	0,85	0,84	0,62	0,58
140	140	19.600	47.487	13.225,23	6.598,60	4.047,87	2.850,61	3,59	7,20	11,73	16,66	0,84	0,90	0,73	0,69
200	200	40.000	182.287,98	52.132	26.776	17.263,78	12.157,59	3,50	6,81	10,56	14,99	0,62	0,85	0,66	0,62
280	280	78.400	808.456,65	232.824	118.580	61.231,78	43.120,97	3,47	6,82	13,20	18,75	0,58	0,85	0,83	0,78

Tabla 5.12.- Tiempo de cómputo totales y desempeño del algoritmo del modelo del álabe recto para el "for" paralelizable del Cuadro 4.1. incrementando la cantidad de elementos finitos usando la arquitectura de CPU de la Figura 4.18 (máquina baremetal proporcionada por Equinix en la nube).

Nr	Ntet a	# de elementos finitos	Tiempo total del "for" paralelizable del Cuadro 4.1. [seg]					Desempeño del algoritmo							
			Serial	Paralelo				Speed-up				Eficiencia			
				Cantidad de núcleos				Cantidad de núcleos				Cantidad de núcleos			
				4	8	16	24	4	8	16	24	4	8	16	24
80	80	6.400	3.236,45	785,12	412,34	295,15	239,96	4,12	7,85	10,97	13,49	1,03	0,98	0,69	0,56
140	140	19.600	33.353,25	7.231,24	3.694,46	2.657,88	2.160,88	4,61	9,03	12,55	15,44	1,15	1,13	0,78	0,64
200	200	40.000	162.327,90	35.123,98	17.663	12.042,78	9.790,88	4,62	9,19	13,48	16,58	1,16	1,15	0,84	0,69
280	280	78.400	601.378,92	134.753	67.649	45.235,79	29.777,45	4,46	8,89	13,29	20.20	1,12	1,11	0,83	0,84

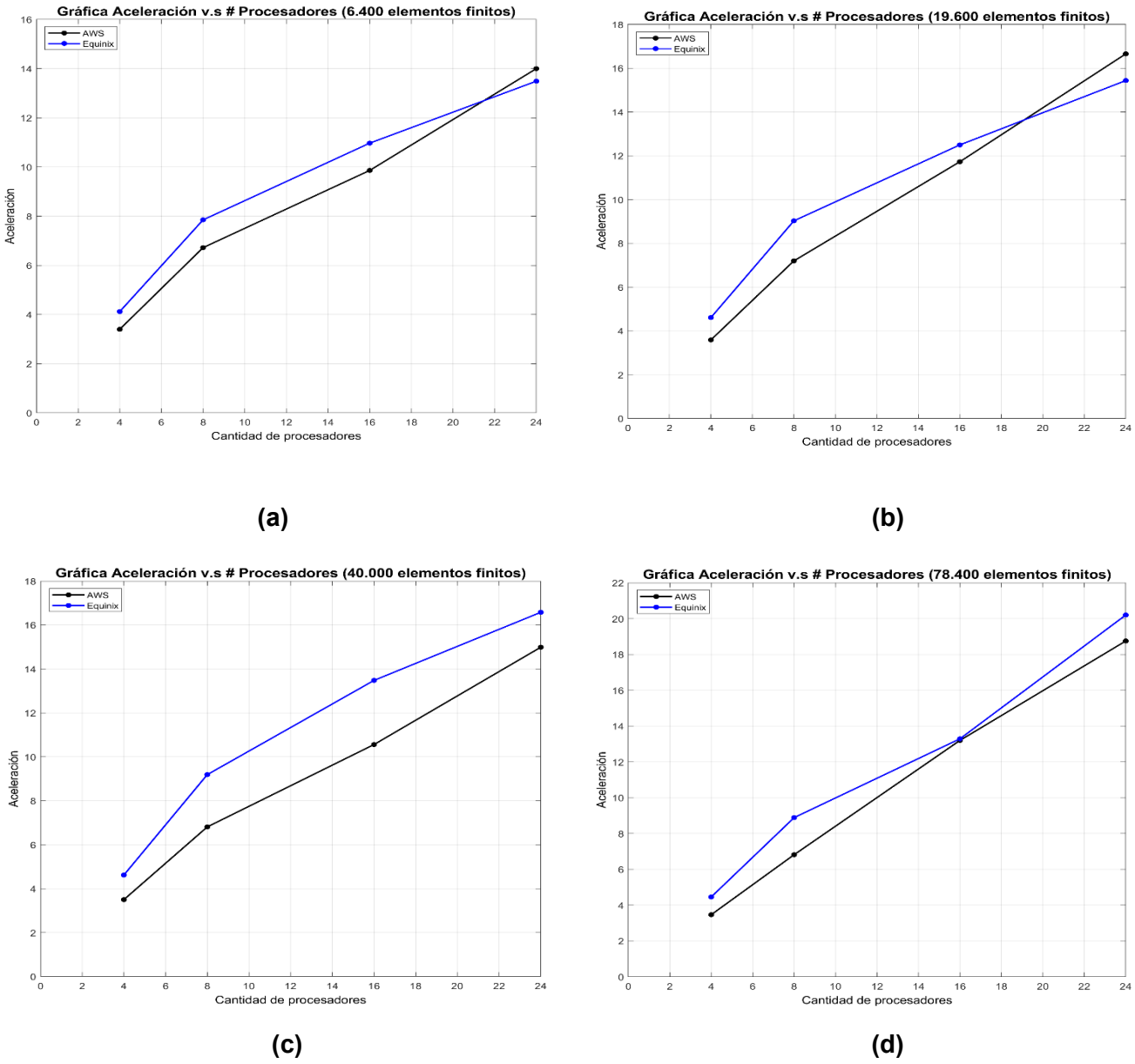


Figura 5.27.- Gráficas de escalabilidad para el “for” paralelizable del Cuadro 4.1. del algoritmo del modelo del álabe recto para diferentes números de elementos finitos: **a)** 6.400, **b)** 19.600, **c)** 40.000 y **d)** 78.400 utilizando la arquitectura de CPU de AWS y Equinix en la nube.

Así, una vez mostradas las gráficas de escalabilidad en la Figura 5.27, se decide ahora mostrar las gráficas de eficiencia para este bucle “for”. En la Figura 5.28 se muestran estas gráficas de eficiencia tanto para la estructura de CPU proporcionada por AWS y por Equinix.

Analizando las gráficas de AWS, se puede observar que para ningún caso de elementos finitos se obtiene una eficiencia mayor que 1. Ahora si se analiza el caso de 6.400 elementos finitos, se puede observar que las eficiencias que se obtienen son de 0.85, 0.84, 0.62 y 0.58 cuando se utilizan 4, 8,16 y 24 núcleos en paralelo respectivamente; y, a pesar de que no son eficiencias mayores a 1, concluimos que son aceleraciones significativas (por unidad contribuyente) que de cierta manera aceleran el tiempo en serial. De igual manera se emite el mismo análisis para los demás casos de elementos finitos.

Para las gráficas en Equinix podemos notar que para todos los casos de elementos finitos se observa una eficiencia por encima de 1 cuando se utilizan 4 y 8 núcleos en paralelo, a excepción para el caso de 6.400 elementos finitos que solo cuando se utiliza 4 núcleos en paralelo se da una eficiencia mayor a 1. Un aspecto adicional que hay que notar es que para los tres primeros casos (6.400, 19.600 y 40.000 elementos finitos) se aprecia una misma tendencia en el decremento de la eficiencia; en cambio para la gráfica de 78.400 elementos finitos se observa también una disminución de la eficiencia, pero con una tendencia más estabilizada en comparación con las gráficas anteriores. Esta tendencia más estabilizada de la eficiencia se debe al aumento de la cantidad de elementos finitos en el dominio de diseño.

Es así como de esta manera se termina el análisis del bucle “for” del Cuadro 4.1. En la siguiente sección 5.4.1.2. , se muestran los tiempos computacionales en serial y en paralelo del bucle “for” del Cuadro 4.2 en la Tabla 5.13 y Tabla 5.14 tanto para la arquitectura de CPU proporcionada por AWS y Equinix respectivamente, donde de igual manera se presentan las gráficas de escalabilidad y eficiencia para los diferentes números de elementos finitos.

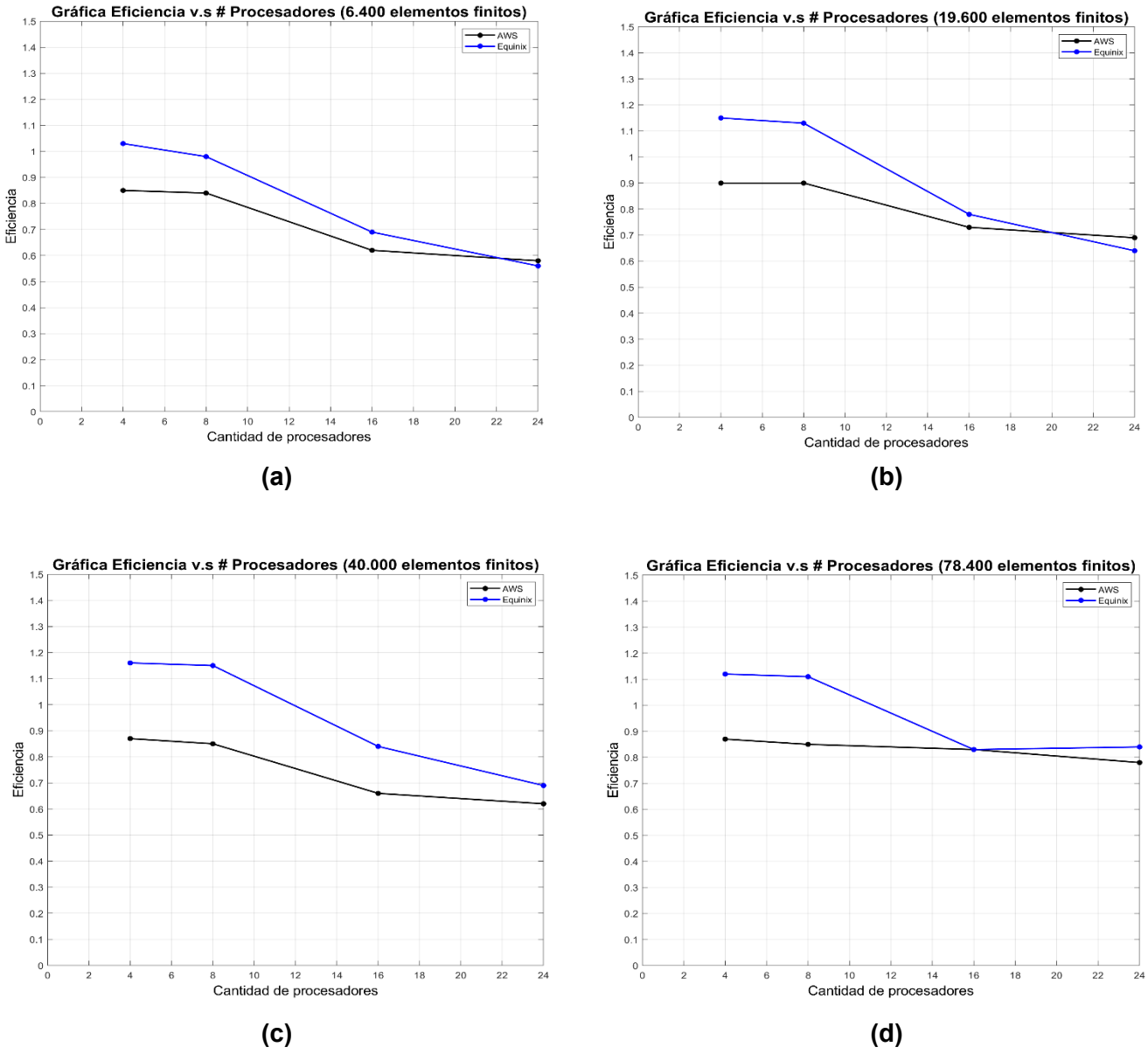


Figura 5.28.- Gráficas de eficiencia para el “for” paralelizable del Cuadro 4.1. del algoritmo del modelo del álabe recto para diferentes números de elementos finitos: **a)** 6.400, **b)** 19.600, **c)** 40.000 y **d)** 78.400 utilizando la arquitectura de CPU de AWS y Equinix en la nube.

5.4.1.2.- DESEMPEÑO DEL BUCLE “FOR” DEL CUADRO 4.2 DEL MODELO DE ÁLABE RECTO

En la Tabla 5.13 y Tabla 5.14 se presentan los tiempos computacionales para la arquitectura de CPU proporcionada por AWS y Equinix respectivamente. Se decide presentar las gráficas de escalabilidad y de eficiencia de este “for”, con el objetivo de obtener un análisis independiente de este ciclo paralelizado; y, la incidencia que este tiene sobre el tiempo total del algoritmo MEF del modelo del álabe recto.

Es así, que en la Figura 5.29 se muestran las gráficas de escalabilidad para el “for” paralelizable del Cuadro 4.2. del algoritmo del modelo del álabe recto para diferentes números de elementos finitos para la arquitectura de CPU de AWS y Equinix. Analizando las gráficas de AWS, se puede observar que para todos los casos (6.400, 19.600, 40.000 y 78.000 elementos finitos) se alcanzan aceleraciones muy por encima de la cantidad de núcleos utilizados en paralelo. Por ejemplo, para el caso de 6.400 elementos finitos, las aceleraciones que se alcanzan son de 33.43x, 60.46x, 94.92x y 134.79x cuando se utilizan 4, 8, 16 y 24 núcleos paralelos respectivamente.

De igual manera se obtienen aceleraciones muy por encima de la cantidad de núcleos cuando se utiliza la arquitectura de Equinix para los diferentes elementos finitos. Se puede observar también que para todos los casos se obtienen aceleraciones muy por encima a la cantidad de núcleos.

Se puede ver que estas aceleraciones de este bucle “for” son resultados muy favorables; y, si lo comparamos con el bucle anterior existe una diferencia muy notaria. Esto no quiere decir que un “for” está mejor paralelizado que el otro. La razón por la cual el bucle “for” del Cuadro 4.2. tiene mejores resultados que el bucle “for” del Cuadro 4.1. se debe principalmente por dos razones específicamente: **a)** en este punto se destaca que cuando el programador asigna un “parfor” a un “for”, debe tener en cuenta que MATLAB clasifica a las variables dentro de un “parfor-loop” dependiendo de la estructura que tengan estas variables (variables tipo: rodaja, difusión, reducción y temporales). Por lo tanto, el bucle del Cuadro 4.2. posee solo una variable clasificada cuando tiene que llamar a ejecutar a la “función permn” (función encargada de realizar permutaciones pares con repetición a los grados de libertad totales para cada elemento finito para el ensamblaje de la matriz global $\boldsymbol{k}$); en cambio, el bucle del Cuadro 4.1. posee siete variables clasificadas previo al llamado de la función “fluidelementmatrix”. El hecho de tener mayores variables clasificadas previo al llamado de una función ralentiza a la paralelización, debido a que se requiere mayor intercambio de información entre núcleos. **b)** La otra razón que es notoria, radica esencialmente en la cantidad de cálculos que se tienen que realizar en “fluidelementmatrix” en comparación con “permn”. Esto se evidencia sustancialmente en los tiempos en seriales si se comparan los de la Tabla 5.11 y Tabla 5.12 (pertenecientes al bucle del Cuadro 4.1.) con los tiempos en seriales de la Tabla 5.13 y Tabla 5.14 (pertenecientes al bucle del Cuadro 4.2.).

Tabla 5.13.- Tiempo de cómputo totales y desempeño del algoritmo del modelo del álabe recto para el "for" paralelizable del Cuadro 4.2 incrementando la cantidad de elementos finitos usando la arquitectura de CPU de la Figura 4.17 (Máquina Virtual proporcionada por AWS en la nube).

Nr	Ntet a	# de elementos finitos	Tiempo total del "for" paralelizable del Cuadro 4.2 [seg]					Desempeño del algoritmo							
			Serial	Paralelo				Speed-up				Eficiencia			
				Cantidad de núcleos				Cantidad de núcleos				Cantidad de núcleos			
				4	8	16	24	4	8	16	24	4	8	16	24
80	80	6.400	1.470,20	43,98	24,32	15,49	10,91	33,43	60,46	94,92	134,79	8,36	7,56	5,93	5,62
140	140	19.600	13.732	410,12	218,23	139	97,89	33,48	62,92	98,79	140,28	8,37	7,87	6,17	5,85
200	200	40.000	57.338	1.690	910,35	579,84	408,34	33,93	62,98	98,89	140,42	8,48	7,87	6,18	5,85
280	280	78.400	232.345,12	6.951	3.644	2.321	1.634,51	33,43	63,76	100,11	142,15	8,36	7,97	6.26	5,92

Tabla 5.14.- Tiempo de cómputo totales y desempeño del algoritmo del modelo del álabe recto para el "for" paralelizable del Cuadro 4.2 incrementando la cantidad de elementos finitos usando la arquitectura de CPU de la Figura 4.18 (Máquina Baremetal proporcionada por Equinix en la nube).

Nr	Ntet a	# de elementos finitos	Tiempo total del "for" paralelizable del Cuadro 4.2 [seg]					Desempeño del algoritmo							
			Serial	Paralelo				Speed-up				Eficiencia			
				Cantidad de núcleos				Cantidad de núcleos				Cantidad de núcleos			
				4	8	16	24	4	8	16	24	4	8	16	24
80	80	6.400	1.655,60	43,75	24,15	14,45	10,89	38,07	68,97	115,27	152,95	9,52	8,62	7,20	6,37
140	140	19.600	14.445,23	374,12	199,07	126,80	95,23	38,61	72,56	113,92	151,69	9,65	9,07	7,12	6,32
200	200	40.000	61.243	1.598	839,55	534,30	401,18	38,32	72,95	114,53	152,66	9,58	9,12	7,16	6,36
280	280	78.400	238.452,14	6.235	3.334,22	2.123,71	1.578,78	38,24	71,52	112,28	151,04	9,56	8,94	7,03	6,29

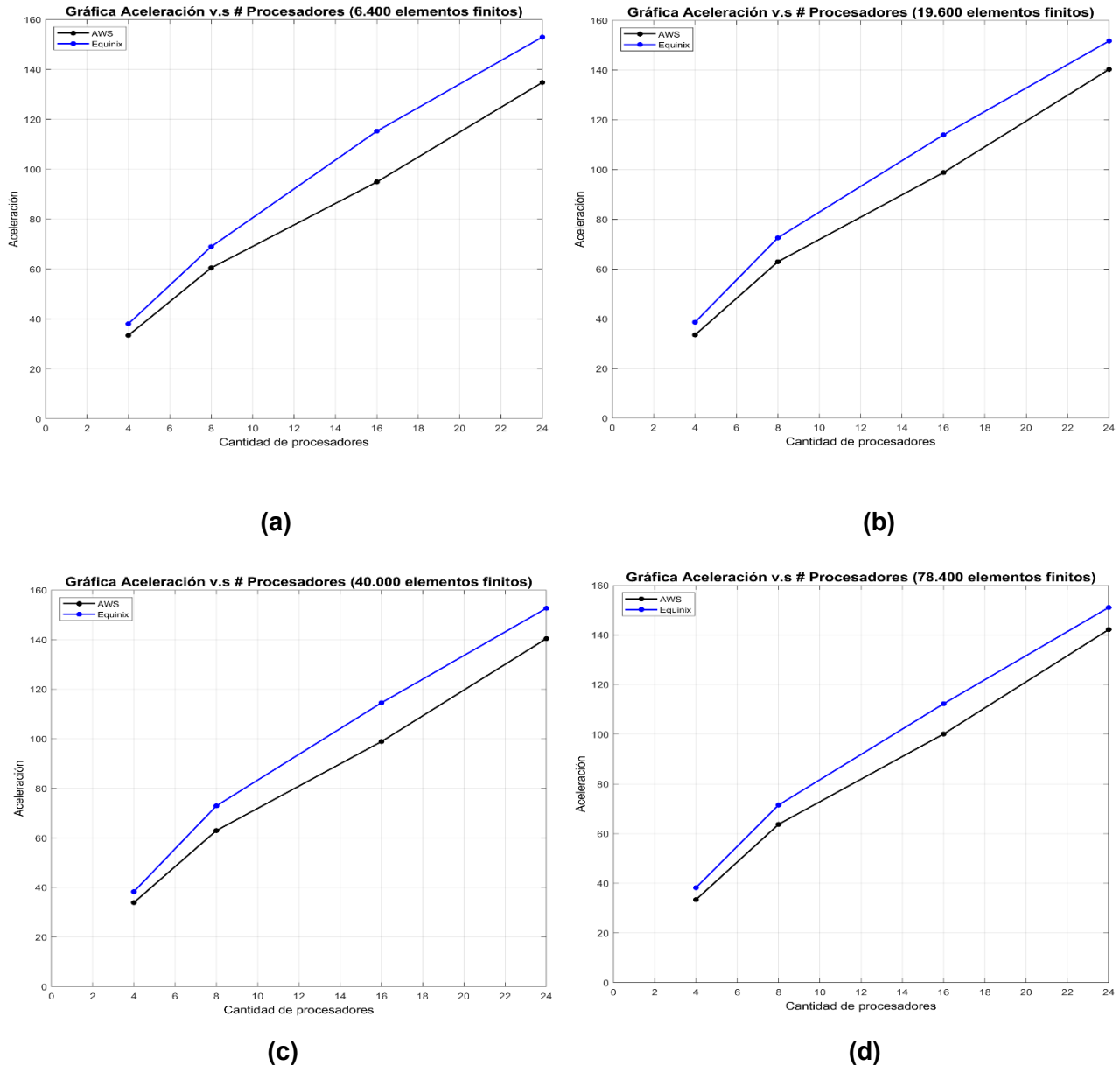


Figura 5.29.- Gráficas de escalabilidad para el “for” paralelizable del Cuadro 4.2 del algoritmo del modelo del álabe recto para diferentes números de elementos finitos: **a)** 6.400, **b)** 19.600, **c)** 40.000 y **d)** 78.400 utilizando la arquitectura de CPU de AWS y Equinix en la nube.

Así, una vez obtenido los resultados de escalabilidad para el bucle del Cuadro 4.2, de igual manera que el bucle anterior se decide presentar las gráficas de eficiencia. En la Figura 5.30 se presentan estas gráficas de eficiencia para la estructura de CPU proporcionada por AWS y por Equinix.

Para las gráficas de AWS se puede observar que para todos los casos (6.400, 19.600, 40.000 y 78.400 elementos finitos) la eficiencia es mayor a 1. Entonces, para el caso de 6.400 elementos finitos las aceleraciones que se alcanzan por núcleo son 8.36x, 7.56x, 5.93x y 5.62x cuando se utilizan 4, 8, 16 y 24 núcleos en paralelo respectivamente. Como era de esperarse, vemos la eficiencia va disminuyendo a medida que se aumentan la cantidad de núcleos en paralelo.

De igual manera, si se observan las gráficas de Equinix, podemos ver que para todos los casos la eficiencia está muy por encima de 1. Si se analiza el caso de 78.400 elementos finitos, las eficiencias que se obtienen son de 9,56, 8,94, 7,03 y 6,29 cuando se utilizan 4,8, 16 y 24 núcleos en paralelo respectivamente. Vemos también que para todos los casos se observa una tendencia de decremento en la eficiencia a medida que se aumenta la cantidad de núcleos utilizados en paralelo.

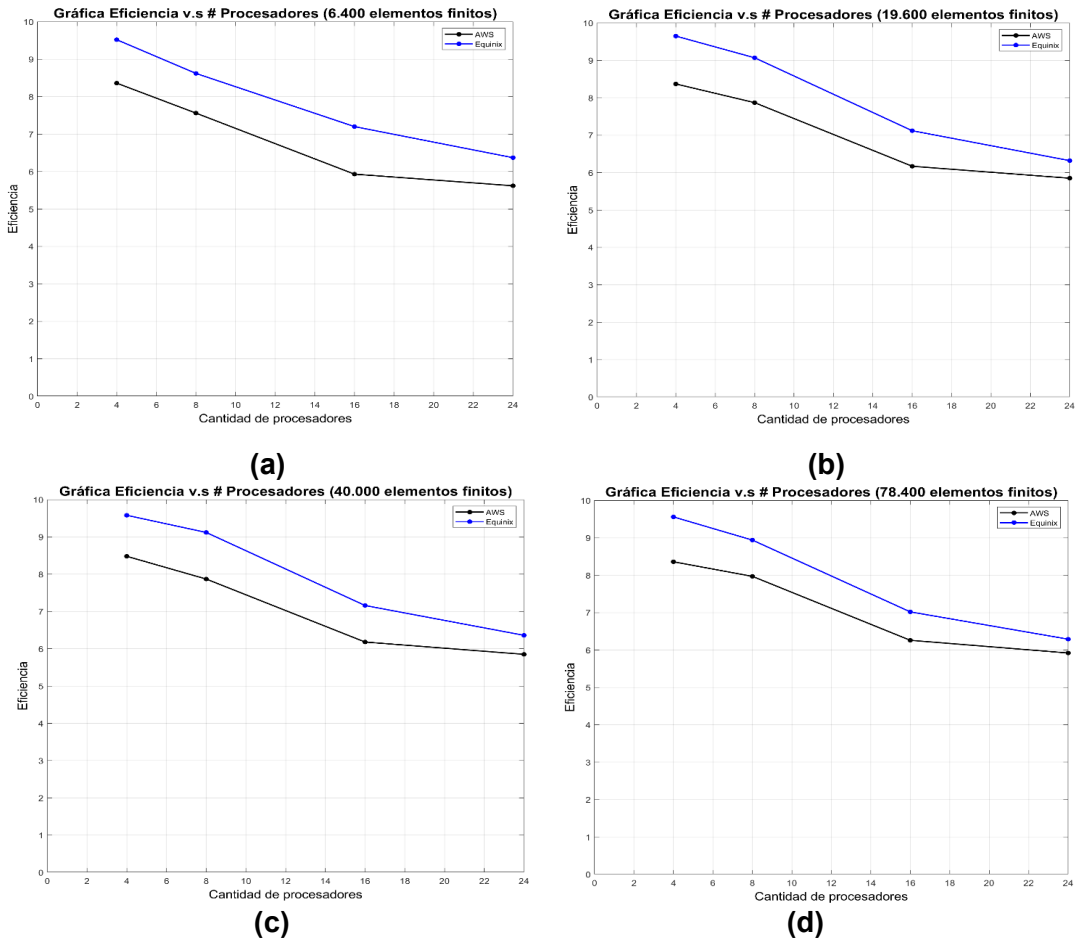


Figura 5.30.- Gráficas de eficiencia para el "for" paralelizable del Cuadro 4.2 del algoritmo del modelo del álabe recto para diferentes números de elementos finitos: **a)** 6.400, **b)** 19.600, **c)** 40.000 y **d)** 78.400 utilizando la arquitectura de CPU de AWS y Equinix en la nube.

5.4.1.3.- DESEMPEÑO DEL BUCLE “FOR” DEL CUADRO 5.1 DEL MODELO DE ÁLABE RECTO

Como se mencionó en la sección 5.4.1, este bucle paralelizado guarda en vectores filas los resultados de los valores nodales de las presiones para posteriormente graficar el contorno de presión. Los desempeños de este bucle tanto para la arquitectura de CPU proporcionada por AWS y Equinix se encuentran en la Tabla 5.15 y Tabla 5.16.

En estas tablas se puede notar que los tiempos en paralelo deberían reducirse, pero esto no ocurre de esta manera. En este bucle a medida que se aumenta la cantidad de núcleos en paralelo, pasa totalmente lo contrario, el tiempo aumenta. Es así, que en este punto se destaca que a pesar de que la paralelización de este bucle fue ejecutada en el algoritmo del MEF, la paralelización no es recomendable. Esto se debe principalmente por la poca cantidad de operaciones que se tienen que realizar en este ciclo, cuya única función es la de ordenar y almacenar los resultados de los valores nodales de la presión en vectores filas.

Es así que debido a la ralentización de este bucle cuando se corre en paralelo, el autor decide no presentar las gráficas de aceleración y escalabilidad para este ciclo “for” del Cuadro 5.1. Adicionalmente, en este punto se destaca la importancia de tener un criterio sobre que bucle puede o no ser paralelizado, es por eso que antes de paralelizar un ciclo, es importante correr casos con pocos elementos finitos o al menos correr 1 iteración del MEF por medio del profiler de MATLAB con el objetivo de tener una visualización general en base a tiempos de ejecución de todo el algoritmo computacional.

De esta manera se concluye el análisis del modelo MEF para el modelo del álabe recto. En la siguiente sección 5.4.2, se analizan los tiempos computacionales tanto en serie y en paralelo para el Caso I (función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) y Caso II (función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) en el modelo de diseño del álabe recto en media circunferencia de rotor utilizando el algoritmo MEF y MOT.

Tabla 5.15.- Tiempo de cómputo totales y desempeño del algoritmo del modelo del álabe recto para el “for” paralelizable del Cuadro 5.1 incrementando la cantidad de elementos finitos usando la arquitectura de CPU de la Figura 4.17 (Máquina Virtual proporcionada por AWS en la nube).

Nr	Ntet a	# de elementos finitos	Tiempo total del “for” paralelizable del Cuadro 5.1 [seg]					Desempeño del algoritmo							
			Serial	Paralelo				Speed-up				Eficiencia			
				Cantidad de núcleos				Cantidad de núcleos				Cantidad de núcleos			
				4	8	16	24	4	8	16	24	4	8	16	24
80	80	6.400	0,0075	0,00587	0,0910	0,182	0,271	0,13	0,08	0,04	0,03	0,032	0,010	0,026	0,0012
140	140	19.600	0,00820	0,0593	0,0922	0,184	0,275	0,14	0,09	0,04	0,03	0,035	0,011	0,028	0,0012
200	200	40.000	0,0090	0,0695	0,0940	0,188	0,280	0,13	0,10	0,05	0,03	0,032	0,012	0,030	0,0013
280	280	78.400	0,00940	0,0789	0,0990	0,198	0,295	0,12	0,09	0,05	0,03	0,030	0,012	0,030	0,0013

Tabla 5.16.- Tiempo de cómputo totales y desempeño del algoritmo del modelo del álabe recto para el “for” paralelizable del Cuadro 5.1 incrementando la cantidad de elementos finitos usando la arquitectura de CPU de la Figura 4.18 (Máquina Baremetal proporcionada por Equinix en la nube).

Nr	Ntet a	# de elementos finitos	Tiempo total del “for” paralelizable del Cuadro 5.1 [seg]					Desempeño del algoritmo							
			Serial	Paralelo				Speed-up				Eficiencia			
				Cantidad de núcleos				Cantidad de núcleos				Cantidad de núcleos			
				4	8	16	24	4	8	16	24	4	8	16	24
80	80	6.400	0,0062	0,0359	0,081	0,162	0,235	0,17	0,08	0,04	0,03	0,04	0,010	0,0024	0,0011
140	140	19.600	0,0064	0,0379	0,089	0,178	0,258	0,17	0,07	0,04	0,02	0,04	0,09	0,0022	0,0010
200	200	40.000	0,0065	0,0384	0,092	0,184	0,267	0,17	0,07	0,04	0,02	0,04	0,09	0,0022	0,0010
280	280	78.400	0,0068	0,0427	0,095	0,190	0,276	0,16	0,07	0,04	0,02	0,04	0,09	0,0022	0,0010

5.4.2- DESEMPEÑO DEL ALGORITMO MEF Y MOT APLICADO A MEDIA CIRCUNFERENCIA DEL ROTOR PARA EL DISEÑO DE ROTORES DE TURBOMÁQUINAS

En esta sección en cambio se analiza el desempeño del algoritmo MEF y MOT para el Caso I (Figura 5.4a) y Caso II (Figura 5.12) en el diseño de la topología del álabe en media circunferencia de rotor para turbomáquinas manteniendo una malla fija de 1.800 elementos finitos.

En este punto se destaca que a pesar de que se toman los tiempos totales del algoritmo de este modelo tanto en serial como en paralelo y que estos valores representan en sí el tiempo total que se demora en converger el algoritmo MEF y MOT, hay que mencionar que los tres bucles paralelizados que se presentan en esta sección provienen exclusivamente del MEF. En este punto se destaca que el optimizador del MMA no lleva ningún bucle paralelizado debido principalmente al hecho de que los tiempos de ejecución de este algoritmo están en el orden de centésimas de segundos, por lo que alguna paralelización de algún bucle podría ralentizar el proceso de ejecución en vez de acelerarlo.

Es así, que los bucles del MEF paralelizados que se presentan en esta sección son: **a) "**for" del pseudocódigo del Cuadro 4.1. donde se incluye el cálculo y el almacenamiento de la matriz de vorticidad $\boldsymbol{M_r}$ para todos los elementos debido a que el cálculo de las funciones objetivos son necesarias para el MOT, **b)** el segundo "for" paralelizado es el que se encuentra en el Cuadro 4.2 y **c)** el "for" del pseudocódigo del Cuadro 4.3. De igual manera que el modelo del alabe recto, se presentan los tiempos en seriales y en paralelo tanto para la arquitectura de CPU proporcionada por AWS y por Equinix.

5.4.2.1- DESEMPEÑO DEL ALGORITMO MEF Y MOT PARA EL CASO I: FUNCIÓN MONO-OBJETIVO, MINIMIZACIÓN DE ENERGÍA DE DISIPACIÓN, $w_d = 1$ Y $w_r = 0$.

En la Tabla 5.17. y Tabla 5.18. se muestran los tiempos computacionales en serial y en paralelo juntos con los desempeños del algoritmo MEF y MOT para el Caso I (función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) tanto para la arquitectura de CPU proporcionada por AWS y por Equinix respectivamente.

En la Figura 5.31 se presentan las gráficas de escalabilidad (aceleración vs. número de procesadores) tanto para la arquitectura de CPU de la máquina proporcionada por AWS y Equinix. Si se analiza las gráficas de AWS, las aceleraciones que se obtienen son de 3.88x, 5.99x, 6.25x, 6.69x, 7.18x y 7.48 cuando se utilizan 4, 8, 12, 16, 20 y 24 núcleos en paralelo respectivamente. En cambio, para las gráficas de Equinix las aceleraciones que se obtienen para este tipo de arquitectura son de 4.02x, 5.95x, 6.74x, 6.80x y 6.80x para la misma cantidad de núcleos.

Se podría llegar a pensar que estas aceleraciones no son muy favorables en la obtención del diseño final del álabe; pero, si comparamos el tiempo en serial para la máquina virtual en AWS (13.033 [seg] igual a 3.62 horas) y en Equinix (9.646 [seg] igual a 2.67 horas) con el tiempo en paralelo cuando se utiliza 24 núcleos (1.741 [seg] igual a 29 [min] en AWS y 1.418 [seg] igual a 23 min con 37 segundos en Equinix), se puede notar que se reduce mucho el tiempo de diseño, lo cual es un punto muy favorable que se tendría que tomar en cuenta no solo a nivel académico sino a nivel industrial donde el tiempo es un factor muy importante en la producción de diseños de álabes en turbomáquinas.

Adicional a lo anterior, se puede observar en estas gráficas que la cantidad máxima de núcleos recomendable a utilizar en paralelo es de 8 y 12 tanto para la máquina de AWS y Equinix respectivamente. La razón del porque se recomienda esta cantidad máxima de núcleos se debe a que posterior a esta cantidad de núcleos utilizados en paralelo, la aceleración no es significativa. Este análisis de recursos se realiza debido al hecho de que estos proveedores alquilan las máquinas virtuales por hora; y, entre más recursos se utilicen en ellas más cuesta el valor por hora (análisis que será abordado en la sección 5.5).

Tabla 5.17.- Tiempo de cómputo totales y desempeño del algoritmo para el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) utilizando la arquitectura de CPU proporcionada por AWS.

Nr	Nteta	# de elementos finitos	Tiempo total del algoritmo MEF y MOT [seg]							Desempeño del algoritmo												
			Serial	Paralelo						Speed-up						Eficiencia						
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos						
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	13.033	3.361	2.174	2.085	1.949	1.815	1.741	3,88	5,99	6,25	6,69	7,18	7,48	0,97	0,75	0,52	0,42	0,36	0,31

Tabla 5.18.- Tiempo de cómputo totales y desempeño del algoritmo para el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso II (función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) utilizando la arquitectura de CPU proporcionada por Equinix.

Nr	Nteta	# de elementos finitos	Tiempo total del algoritmo MEF y MOT [seg]							Desempeño del algoritmo												
			Serial	Paralelo						Speed-up						Eficiencia						
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos						
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	9.646	2.399	1.621	1.430	1.428	1.419	1.418	4,02	5,95	6,74	6,75	6,80	6,80	1,01	0,74	0,56	0,42	0,34	0,28

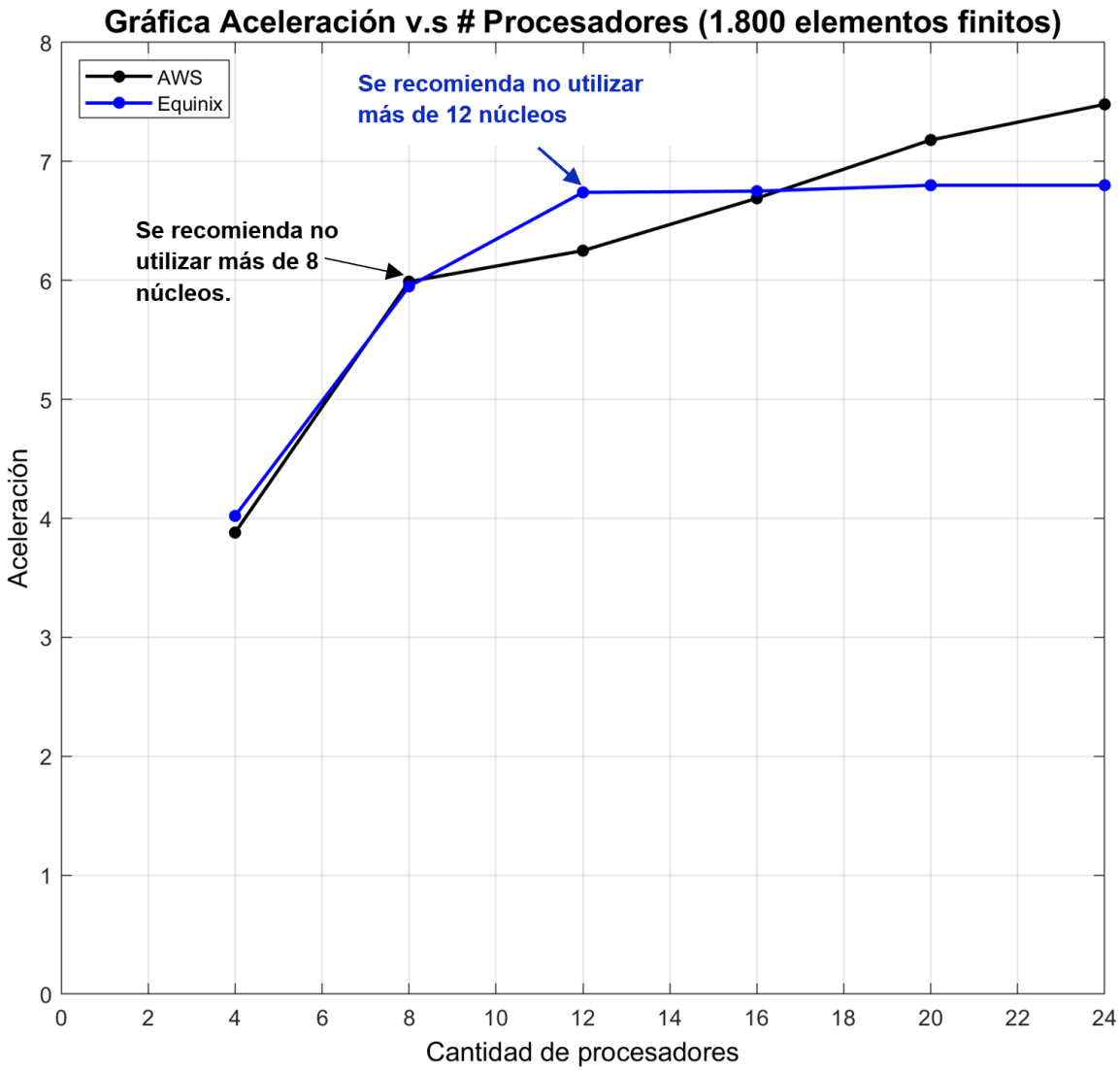


Figura 5.31.- Gráficas de escalabilidad, aceleración vs. número de procesadores para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación $w_d = 1$ y $w_r = 0$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

Una vez mostradas las gráficas de escalabilidad, en la Figura 5.32 se presentan las gráficas de eficiencia tanto para la arquitectura de CPU en la máquina de AWS y Equinix. Para las gráficas de AWS, las eficiencias que se obtienen son de 0,97, 0,75, 0,52, 0,42, 0,36 y 0,31 cuando se utilizan 4, 8, 16, 20 y 24 núcleos respectivamente. En cambio, para las gráficas en Equinix las eficiencias que se obtienen son de 1,01, 0,74, 0,56, 0,42 para la misma cantidad de núcleos.

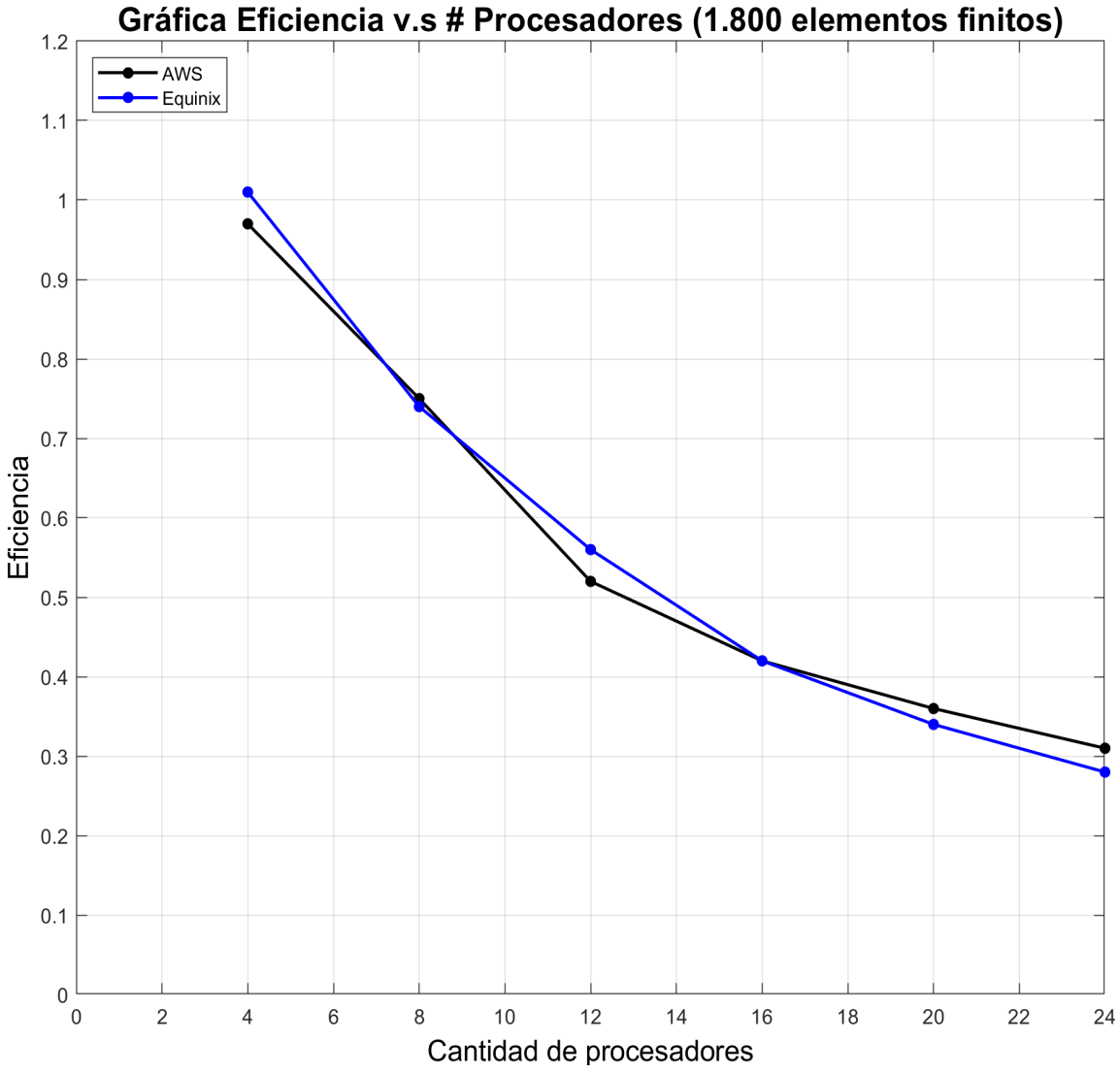


Figura 5.32.- Gráficas de Eficiencia vs. número de procesadores para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación $w_d = 1$ y $w_r = 0$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

5.4.2.1.1.- DESEMPEÑO DEL BUCLE “FOR” DEL CUADRO 4.1 PARA MODELO DEL DISEÑO DE ÁLABES EN TURBOMÁQUINAS PARA EL CASO I

Una vez presentadas las gráficas de escalabilidad y eficiencia del tiempo total del algoritmo MEF y MOT, se decide mostrar el desempeño del bucle “for” del cuadro 4.1. En la Tabla 5.19. y Tabla 5.20. se muestran los tiempos computacionales (serial y paralelo) y los desempeños para el bucle “for” del cuadro 4.1 tanto para la arquitectura de CPU utilizada en AWS y Equinix en la nube.

De la misma manera se presentan las gráficas de escalabilidad y de eficiencia para estas tablas, con el objetivo de poder tener un mejor análisis de la incidencia de este bucle “for” sobre todo algoritmo MEF y MOT en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor. Es así como en la Figura 5.33 se presenta el gráfico de escalabilidad para el bucle “for” del cuadro 4.1 tanto para la arquitectura de CPU proporcionada por AWS y Equinix.

Analizando las gráficas de AWS, se puede notar que las aceleraciones que se obtienen son de 3,95x, 7,40x, 8,43x, 10,65x, 13,38x y 15,93 cuando se utilizan 4,8,12,16 y 20 núcleos en paralelo respectivamente. En cambio, para las gráficas de Equinix las aceleraciones que se obtienen son de 3.51x, 6.50x, 8,27, 8,32, 9,30, 11.01 cuando se utilizan la misma cantidad de núcleos. Adicionalmente se observa que las aceleraciones de este bucle “for” tanto para AWS y Equinix no alcanzan el desempeño esperado debido a que ninguna aceleración se encuentra por encima de la cantidad de núcleos utilizados en paralelo.

Un punto que cabe destacar en este gráfico es el hecho de que se obtienen mejores aceleraciones en la máquina de AWS que en la máquina de Equinix. Es así como se podría llegar a pensar que se obtuvieron mejores resultados en AWS que en Equinix, pero el análisis que habría que tomar en cuenta en este punto es en el registro de datos en la diferencia de tiempos entre serial y paralelo. Si se observa la Tabla 5.19. y Tabla 5.20. los tiempos en seriales de este bucle son de 9.273,80 [seg] y 5.748,20 tanto para la máquina en AWS como en Equinix respectivamente, y; si observamos los tiempos en paralelo para 4,8,12,16,20 y 24 se observa de que los tiempos en Equinix siguen siendo mejores que en AWS. Sin embargo, estas aceleraciones de la Figura 5.33 destacan “un mejor desempeño de AWS” debido al hecho de que el rango entre serial y paralelo es muy amplio. En otras palabras, Equinix alcanza un desempeño óptimo mucho más rápido que AWS en la utilización temprano de núcleos en paralelo que impiden la escalabilidad del algoritmo en el modelo de diseño de alabes en media circunferencia de rotor.

Por otro lado, un objetivo adicional que puede ser considerado para posteriores tesis es la evaluación de datos que se envían y se ejecutan en cada núcleo, ya que MATLAB posee comandos como ticBytes(gcp) y tocBytes(gcp) que se ubican antes y después de cada “for”. La importancia de este análisis radica en que se pueden obtener gráficas en el tiempo que permitirían observar el comportamiento de respuesta de cada núcleo a medida que se aumentan la cantidad de núcleos paralelizados.

Tabla 5.19.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) para el "for" paralelizable del Cuadro 4.1. utilizando la arquitectura de CPU en AWS.

Nr	Nteta	# de elementos finitos	Tiempo total del "for" paralelizable del Cuadro 4.1 [seg]							Desempeño del algoritmo										
			Serial	Paralelo						Speed-up						Eficiencia				
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos				
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	9.273,80	2.350,74	1.253,71	1.099,80	870,50	693,31	582,03	3,95	7,40	8,43	10,65	13,38	15,93	0,99	0,92	0,70	0,67	0,67	0,66

Tabla 5.20.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) para el "for" paralelizable del Cuadro 4.1. utilizando la arquitectura de CPU en Equinix.

Nr	Nteta	# de elementos finitos	Tiempo total del "for" paralelizable del Cuadro 4.1 [seg]							Desempeño del algoritmo										
			Serial	Paralelo						Speed-up						Eficiencia				
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos				
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	5.748,20	1.637,40	884,39	695,13	690,55	618,30	522,27	3,51	6,50	8,27	8,32	9,30	11,01	0,88	0,81	0,69	0,52	0,46	0,46

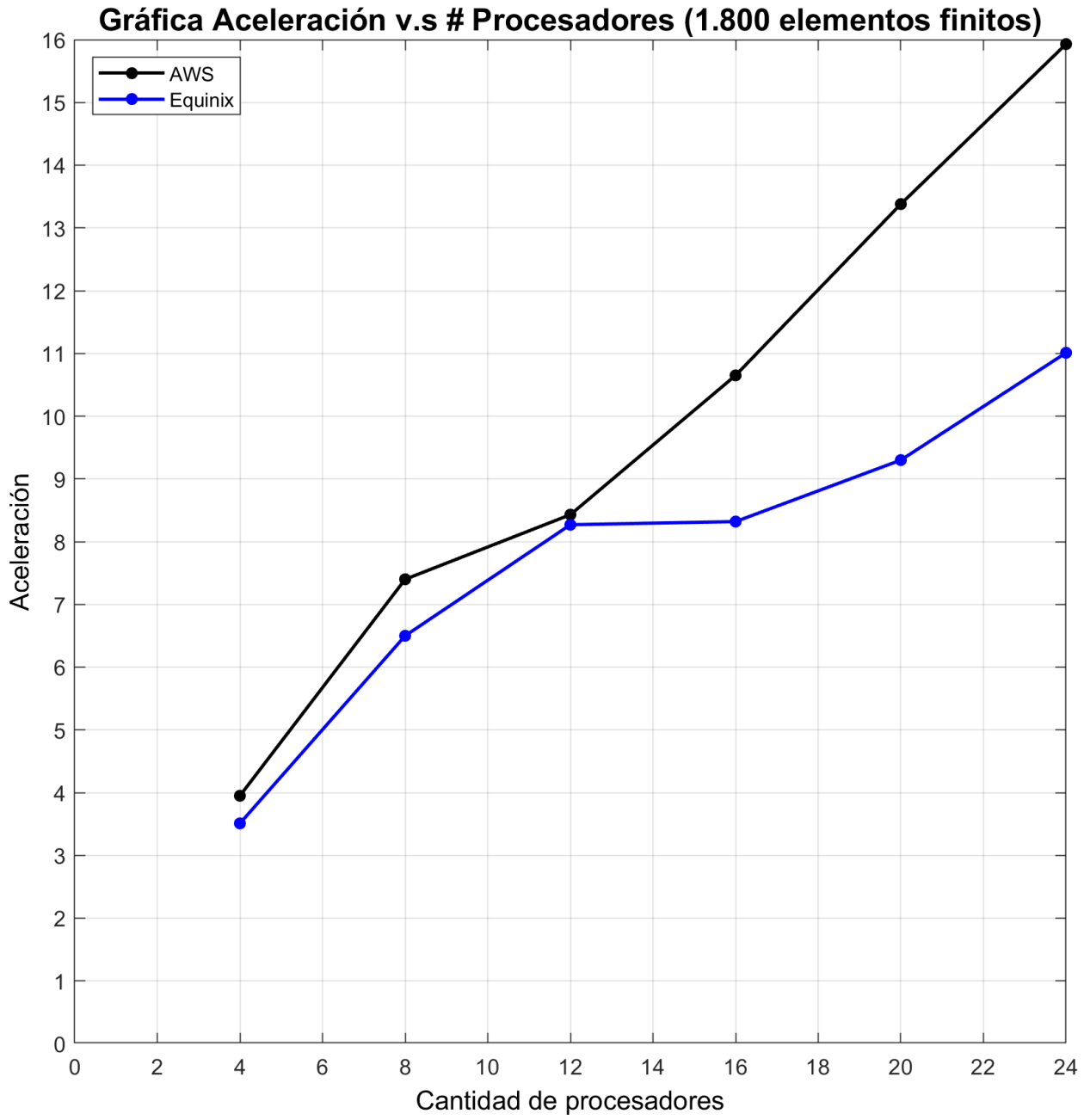


Figura 5.33.- Gráficas de escalabilidad para el bucle "for" del Cuadro 4.1., aceleración vs. número de procesadores para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

Una vez mostradas las gráficas de escalabilidad, en la Figura 5.34 se muestran las gráficas de eficiencia tanto para la arquitectura de CPU en la máquina de AWS y Equinix. Para las gráficas de AWS, las eficiencias que se obtienen son de 0,99, 0,92, 0,70, 0,67, 0,67 y 0,66 cuando se utilizan 4, 8, 16, 20 y 24 núcleos respectivamente. En cambio, para las gráficas en Equinix las eficiencias que se obtienen son de 0,88, 0,81, 0,69, 0,52, 0,46 y 0,46 para la misma cantidad de núcleos.

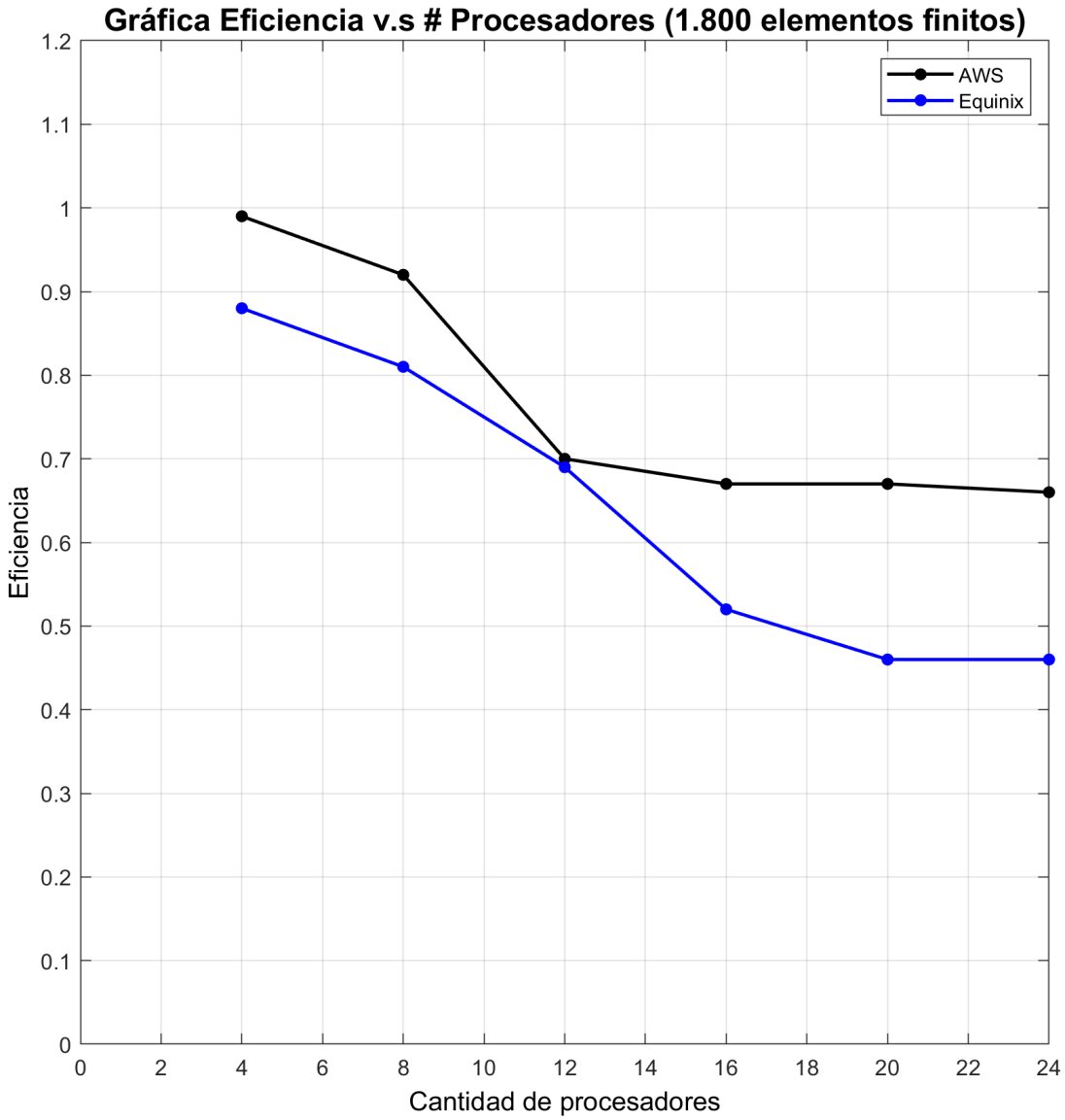


Figura 5.34.- Gráficas de eficiencia para el bucle "for" del Cuadro 4.1. para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

5.4.2.1.2.- DESEMPEÑO DEL BUCLE “FOR” DEL CUADRO 4.2 PARA MODELO DEL DISEÑO DE ÁLABES EN TURBOMÁQUINAS PARA EL CASO I.

En la Tabla 5.21. y Tabla 5.22. se muestran los tiempos computacionales (serial y paralelo) y los desempeños para el bucle “for” del Cuadro 4.2 tanto para la arquitectura de CPU utilizada en AWS y Equinix en la nube respectivamente.

De la misma manera que el bucle anterior, se presentan las gráficas de escalabilidad y eficiencia en base a los datos registrados en estas tablas tanto para la máquina virtual en AWS y Equinix con el fin de evaluar la incidencia de este “for” del Cuadro 4.2 sobre el algoritmo total del MEF y MOT en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor. Es así como en la Figura 5.35 se presenta el gráfico de escalabilidad para el bucle “for” del pseudocódigo del Cuadro 4.2 tanto para la arquitectura de CPU proporcionada por AWS y Equinix.

En esta figura las aceleraciones que se obtienen para la gráfica de AWS son de 20,81x, 28,67x, 20,70x,12,92x, 9,36x, 7,97x cuando se utilizan 4,8,12,16, 20 y 24 núcleos en paralelo. En cambio, para las gráficas de Equinix las aceleraciones que se obtienen son de 27,94x, 30,92x, 37,55x, 26,96x, 15,20x y 11,37x cuando se utilizan la misma cantidad de núcleos. En estas gráficas se puede observar que las aceleraciones se encuentran por debajo de la cantidad de núcleos a partir de utilizar 16 y 20 núcleos cuando se utiliza la máquina de AWS. En cambio, cuando se utiliza la máquina en Equinix para 20 núcleos en paralelo, se obtiene una aceleración de 15,20x, aceleración muy por debajo de la cantidad de 20 núcleos.

Adicionalmente, se destaca que los tiempos en seriales que se registraron son de 2.342,70 [seg] y 2.756,60 [seg] tanto para AWS y Equinix respectivamente. Es entonces que, si se analizan estos tiempos en seriales, se puede notar un mayor tiempo computacional en el registro cuando se utiliza la máquina baremetal de Equinix en comparación a la de AWS, pero si se analizan los tiempos en paralelo se evidencia que se obtienen mejores resultados en la máquina de Equinix, lo que se traduce a tener mejores aceleraciones que en AWS.

Por otro lado, otro punto de análisis que hay que destacar en esta gráfica, es el efecto de tener pocos números de elementos finitos (1.800) en el dominio de diseño en comparación con el modelo del álabe recto. Este efecto de tener pocos elementos finitos se traduce a un decremento temprano en la escalabilidad. Es así que a partir de 8 y 12 núcleos tanto para AWS y Equinix respectivamente, la aceleración comienza a disminuir drásticamente hasta llegar al punto de obtener aceleraciones por debajo de la cantidad de núcleos utilizados en paralelo.

Tabla 5.21.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) para el "for" paralelizable del Cuadro 4.2 utilizando la arquitectura de CPU en AWS.

Nr	Nteta	# de elementos finitos	Tiempo total del "for" paralelizable del Cuadro 4.2 [seg]							Desempeño del algoritmo											
			Serial	Paralelo						Speed-up						Eficiencia					
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos					
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	2.342,70	112,56	81,71	113,19	181,28	250,19	293,94	20,81	28,67	20,70	12,92	9,36	7,97	5,20	3,58	1,73	0,81	0,47	0,33

Tabla 5.22.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) para el "for" paralelizable del Cuadro 4.2 utilizando la arquitectura de CPU en Equinix.

Nr	Nteta	# de elementos finitos	Tiempo total del "for" paralelizable del Cuadro 4.2 [seg]							Desempeño del algoritmo											
			Serial	Paralelo						Speed-up						Eficiencia					
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos					
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	2.756,60	98,65	89,15	73,42	102,27	181,32	242,51	27,94	30,92	37,55	26,96	15,20	11,37	6,99	3,87	3,13	1,69	0,76	0,47

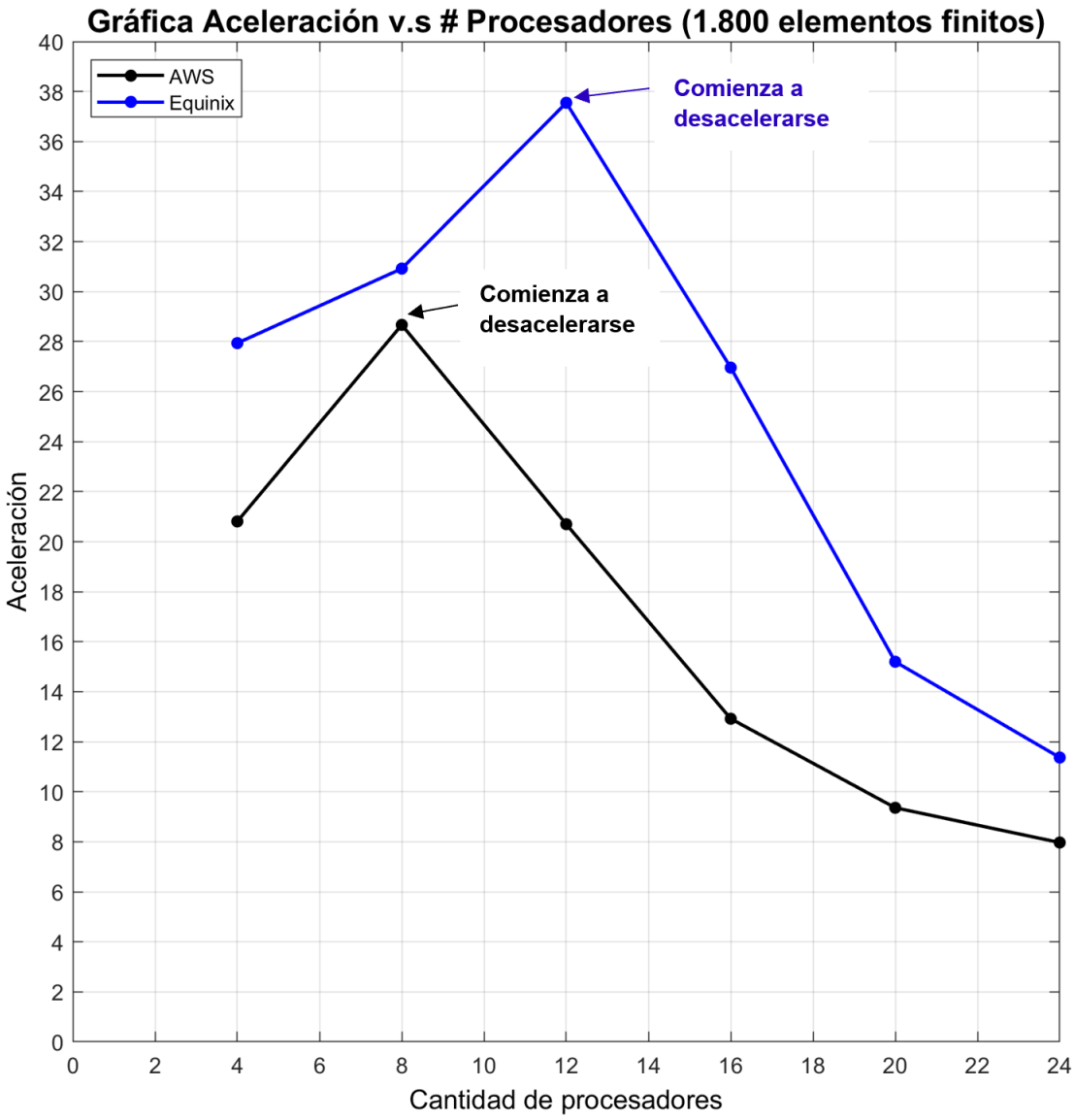


Figura 5.35.- Gráficas de escalabilidad para el bucle "for" del Cuadro 4.2, aceleración vs. número de procesadores para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

En la Figura 5.36 se muestran las gráficas de eficiencia tanto para la arquitectura de CPU en la máquina de AWS y Equinix. Para las gráficas de AWS, las eficiencias que se obtienen son de 5,20, 3,58, 1,73, 0,81, 0,47 y 0,33 cuando se utilizan 4,8,12,16, 20 y 24 núcleos respectivamente. En cambio, para las gráficas en Equinix las eficiencias que se obtienen son de 6,99, 3,87, 3,13, 1,69, 0,76 y 0,47 para la misma cantidad de núcleos.

Adicionalmente, se destaca que la paralelización del bucle "for" del Cuadro 4.2 es eficiente hasta 12 y 16 núcleos utilizados en paralelo para la máquina de Equinix debido a que la eficiencia se encuentra por encima de 1. En cambio, para la máquina de AWS la paralelización de este bucle es eficiente hasta 12 núcleos utilizados en paralelo.

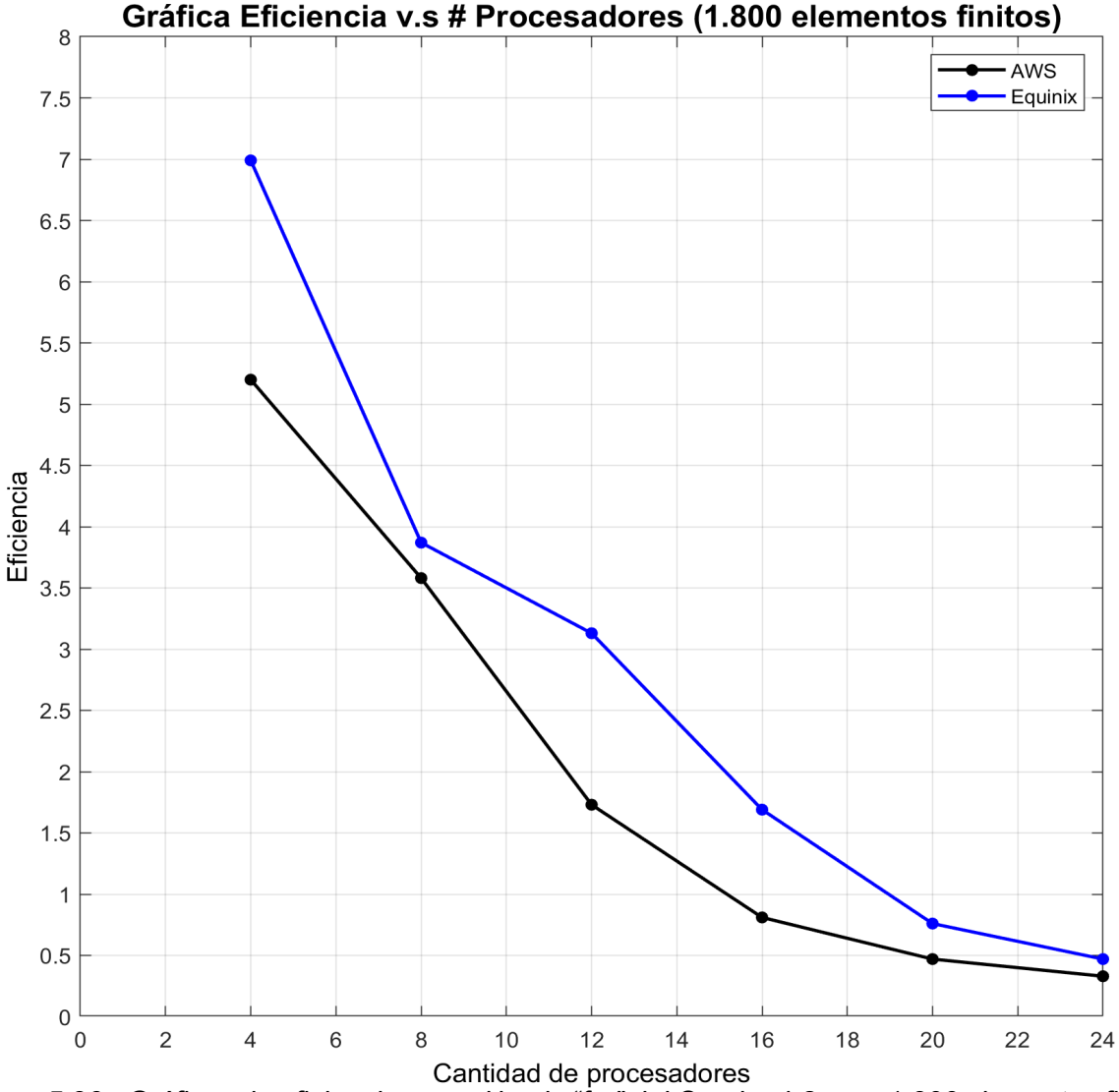


Figura 5.36.- Gráficas de eficiencia para el bucle "for" del Cuadro 4.2 para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación $w_d = 1$ y $w_r = 0$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

5.4.2.1.3.- DESEMPEÑO DEL BUCLE “FOR” DEL CUADRO 4.3 PARA MODELO DEL DISEÑO DE ÁLABES EN TURBOMÁQUINAS PARA EL CASO I.

En la Tabla 5.23. y Tabla 5.24. se muestran los tiempos computacionales en serial y en paralelo en conjunto con el desempeño (escalabilidad y eficiencia) del bucle “for” paralelizado del pseudocódigo del Cuadro 4.3. tanto para la arquitectura de CPU proporcionada por AWS y Equinix respectivamente.

De la misma manera que el pseudocódigo del Cuadro 5.1, se puede observar en estas tablas que en vez de reducirse el tiempo computacional de este bucle a medida que se aumenta la cantidad de núcleos en paralelo, pasa totalmente lo contrario, el tiempo aumenta. Es así como se puede observar que las aceleraciones en la Tabla 5.23. y Tabla 5.24. son menores que 1, lo que se traduce a una desaceleración debido al hecho de que el tiempo en paralelo es mucho mayor al tiempo en serial. Por lo tanto, se concluye que la paralelización de este bucle “for” no es paralelizable.

Tabla 5.23.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) para el “for” paralelizable del Cuadro 4.3 utilizando la arquitectura de CPU en AWS.

Nr	Nteta	# de elementos finitos	Tiempo total del “for” paralelizable del Cuadro 4.3 [seg]							Desempeño del algoritmo											
			Serial	Paralelo						Speed-up						Eficiencia					
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos					
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	0,43	2,02	2,43	2,82	3,46	4,14	5,04	0,21	0,18	0,15	0,13	0,10	0,086	0,053	0,022	0,013	0,0078	0,0052	0,0036

Tabla 5.24.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso I (Función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$) para el “for” paralelizable del Cuadro 4.3 utilizando la arquitectura de CPU en Equinix.

Nr	Nteta	# de elementos finitos	Tiempo total del “for” paralelizable del Cuadro 4.3 [seg]							Desempeño del algoritmo											
			Serial	Paralelo						Speed-up						Eficiencia					
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos					
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	0,48	1,81	1,98	2,12	2,50	2,78	3,08	0,26	0,24	0,23	0,19	0,17	0,16	0,066	0,030	0,019	0,012	0,0086	0,0065

5.4.2.2- DESEMPEÑO DEL ALGORITMO MEF Y MOT PARA EL CASO II: FUNCIÓN BI-OBJETIVO, MINIMIZACIÓN DE ENERGÍA DE DISIPACIÓN Y VORTICIDAD, $w_d = 0.8$ Y $w_r = 0.2$.

En esta sección se registran los tiempos totales en serial y en paralelo en conjunto con su desempeño del algoritmo computacional en el diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso II, ver Figura 5.12 (función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$.). De igual manera que los casos anteriores, se muestran las gráficas de escalabilidad y de eficiencia con el fin de poder comparar el desempeño entre la máquina virtual de AWS y máquina baremetal proporcionada por Equinix.

En la Tabla 5.25. y Tabla 5.26. se muestran los tiempos computacionales en serial y en paralelo juntos con los desempeños del algoritmo MEF y MOT tanto para la arquitectura de CPU proporcionada por AWS y por Equinix respectivamente.

En la Figura 5.37 se presenta la gráfica de escalabilidad (aceleración vs. número de procesadores) tanto para la arquitectura de CPU de la máquina proporcionada por AWS y Equinix. Si se analizan las gráficas de AWS, las aceleraciones que se obtienen son de 3,84x, 5,93x, 6,27x, 6,68x, 7,14x y 7,49x cuando se utilizan 4,8,12,16,20 y 24 núcleos en paralelo respectivamente. En cambio, para las gráficas de Equinix las aceleraciones que se obtienen para este tipo de arquitectura son de 4,02x, 5,83x, 6,61x, 6,74x, 6,72x y 6,88x para la misma cantidad de núcleos. Estas aceleraciones son muy similares a las del caso I tanto para AWS y Equinix. La razón del porque se obtienen aceleraciones muy similares se debe a que el algoritmo del Caso I es el mismo para el Caso II, con la única diferencia que los pesos de ponderación varían dando como resultado convergencia en diferentes números de iteraciones.

Adicionalmente, se puede observar en el caso de AWS que para ningún caso se obtiene una aceleración por encima de la cantidad de núcleos utilizados en paralelo, en cambio, en Equinix cuando se utilizan 4 núcleos en paralelos se puede observar que la aceleración es de 4,02x. Otro punto que se destaca en esta figura de escalabilidad es que como en la máquina de Equinix se obtienen menor diferencia entre el tiempo en serial y paralelo en el algoritmo MEF y MOT en comparación con la máquina virtual de AWS, en la gráfica de escalabilidad en Equinix se evidencia que a partir de utilizar 12 núcleos en paralelo la aceleración comienza a estabilizarse; es decir, deja de escalar. En cambio, para la gráfica de AWS a pesar de que no se obtienen aceleraciones por encima de la cantidad de núcleos, se observa escalabilidad cuando se utilizan 12 núcleos en paralelo. Como se mencionó anteriormente, este efecto de escalabilidad de AWS a diferencia de Equinix se debe a que en la máquina baremetal se obtienen mejores desempeños muchos más rápido que en las máquinas virtuales.

Tabla 5.25.- Tiempo de cómputo totales y desempeño del algoritmo para el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) utilizando la arquitectura de CPU proporcionada por AWS.

Nr	Nteta	# de elementos finitos	Tiempo total del algoritmo MEF y MOT [seg]							Desempeño del algoritmo											
			Serial	Paralelo						Speed-up						Eficiencia					
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos					
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	44.222,36	11.517,01	7.459,47	7.056,43	6.623,45	6.192	5.907,60	3,84	5,93	6,27	6,68	7,14	7,49	0,96	0,74	0,52	0,42	0,36	0,31

Tabla 5.26.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) utilizando la arquitectura de CPU proporcionada por Equinix.

Nr	Nteta	# de elementos finitos	Tiempo total del algoritmo MEF y MOT [seg]							Desempeño del algoritmo											
			Serial	Paralelo						Speed-up						Eficiencia					
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos					
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	32.623,81	8.107,12	5.599,63	4.934,97	4.843,14	4.853,82	4.742,85	4,02	5,83	6,61	6,74	6,72	6,88	1,01	0,73	0,55	0,42	0,34	0,29

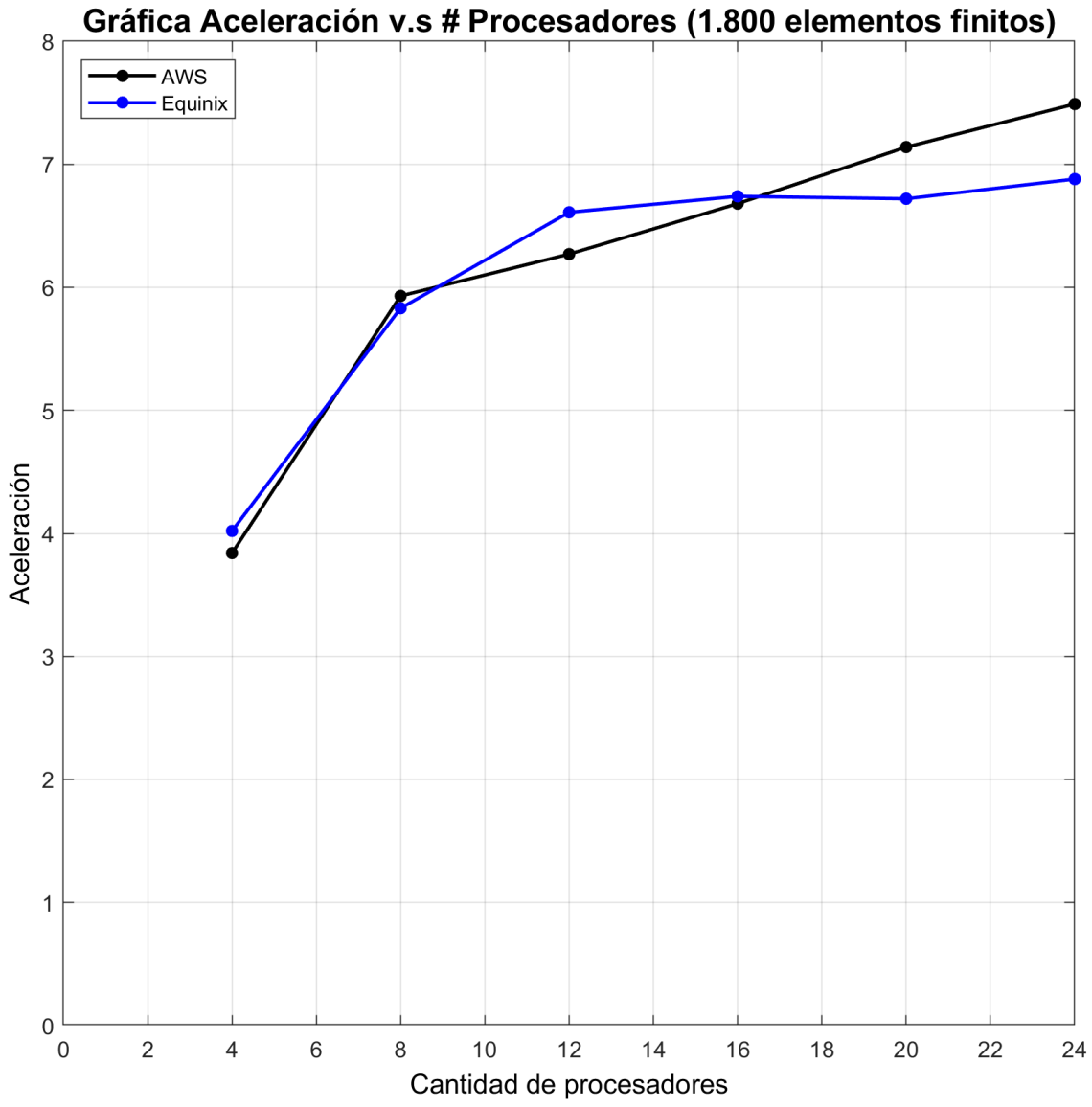


Figura 5.37.- Gráficas de escalabilidad, aceleración vs. número de procesadores para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

Una vez mostradas las gráficas de escalabilidad, en la Figura 5.38 se presentan las gráficas de eficiencia tanto para la arquitectura de CPU en la máquina de AWS y Equinix. Para las gráficas de AWS, las eficiencias que se obtienen son de 0,96, 0,74, 0,52, 0,42, 0,36 y 0,31 cuando se utilizan 4,8,16,20 y 24 núcleos respectivamente. En cambio, para las gráficas en Equinix las eficiencias que se obtienen son de 1,01, 0,73, 0,55, 0,42, 0,34 y 0,29 para la misma cantidad de núcleos.

Se destaca que el algoritmo es eficiente cuando se alcanzan eficiencias por encima de 1. Es así, que se observa que cuando se utilizan 4 núcleos en paralelo en la máquina de Equinix se obtiene un valor de 1,01 de eficiencia.

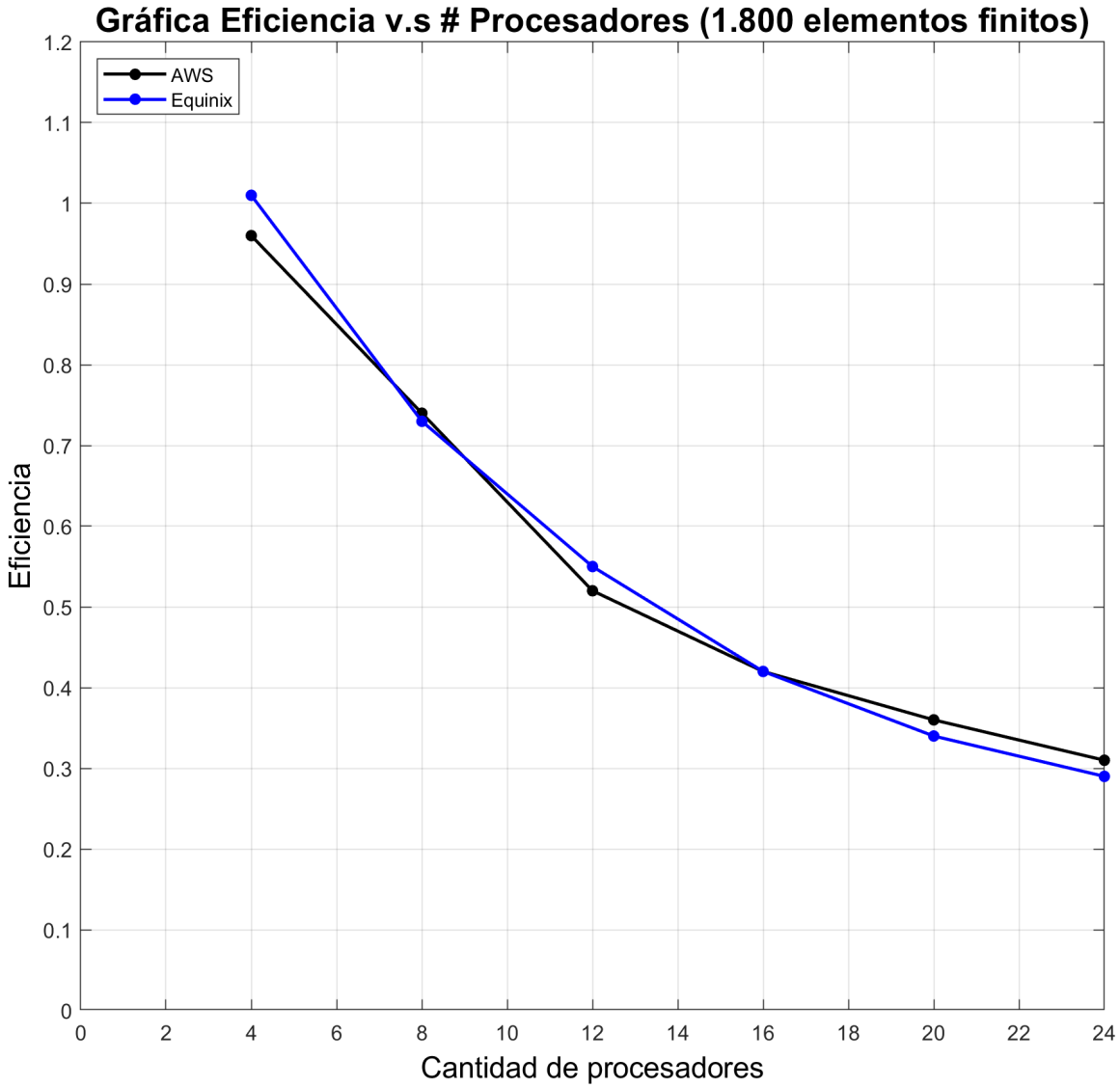


Figura 5.38.- Gráficas de Eficiencia vs. número de procesadores para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

5.4.2.2.1.- DESEMPEÑO DEL BUCLE “FOR” DEL CUADRO 4.1 PARA MODELO DEL DISEÑO DE ÁLABES EN TURBOMÁQUINAS PARA EL CASO II.

De igual manera que el Caso I, una vez presentadas las gráficas de escalabilidad y eficiencia del tiempo total del algoritmo MEF y MOT, se decide mostrar el desempeño del bucle “for” del cuadro 4.1. En la Tabla 5.27 y Tabla 5.28. se presentan los tiempos computacionales (serial y paralelo) y los desempeños para el bucle “for” del cuadro 4.1 tanto para la arquitectura de CPU utilizada en AWS y Equinix en la nube respectivamente.

De igual forma se presentan las gráficas de escalabilidad y de eficiencia para estas tablas, con el objetivo de poder tener un mejor análisis en la incidencia de este bucle “for” sobre todo el algoritmo MEF y MOT en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor. Es así como en la Figura 5.39 se presenta el gráfico de escalabilidad para el bucle “for” del cuadro 4.1 tanto para la arquitectura de CPU proporcionada por AWS y Equinix.

Si se analiza la gráfica de AWS, se puede notar que las aceleraciones que se obtienen son de 4,03x, 6,78x, 9,78x, 11,47x, 14,39x y 17,31x cuando se utilizan 4,8,12,16,20 y 24 núcleos en paralelo. En cambio, para las gráficas de Equinix las aceleraciones que se obtienen son de 3,87x, 7,04x, 9,23x, 9,19x, 10,28x, 12,11x cuando se utilizan la misma cantidad de núcleos. Adicionalmente se observa que solo en AWS cuando se utiliza 4 núcleos en paralelo se alcanza una aceleración por encima de la cantidad de núcleos, 4,03x. Para los demás casos las aceleraciones se encuentran por debajo de la cantidad de núcleos utilizados en paralelo.

Por otro lado, como era de esperarse se muestra un patrón de gráfica similar a la escalabilidad del Caso I (ver Figura 5.33). Cabe destacar que a pesar de que se obtienen mejores desempeños en AWS, los tiempos en seriales y en paralelo siguen siendo menores en la máquina baremeral de Equinix.

En esta gráfica de escalabilidad se puede notar que a partir de utilizar 12 núcleos en paralelo en la máquina baremetal de Equinix, la aceleración incrementa lentamente a medida que se aumentan la cantidad de núcleos, en cambio, en AWS esto no ocurre de la misma manera y a pesar de que no se llega a aceleraciones deseadas (por encima de la cantidad de núcleos) se observa que se puede seguir escalando.

Tabla 5.27.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) para el "for" paralelizable del Cuadro 4.1. utilizando la arquitectura de CPU en AWS.

Nr	Nteta	# de elementos finitos	Tiempo total del "for" paralelizarle del Cuadro 4.1 [seg]							Desempeño del algoritmo											
			Serial	Paralelo						Speed-up						Eficiencia					
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos					
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	31.795,89	7.896,80	4.689,80	3.252,80	2.771,90	2.209,60	1.836,70	4,03	6,78	9,78	11,47	14,39	17,31	1,01	0,85	0,81	0,72	0,72	0,72

Tabla 5.28.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) para el "for" paralelizable del Cuadro 4.1. utilizando la arquitectura de CPU en Equinix.

Nr	Nteta	# de elementos finitos	Tiempo total del "for" paralelizable del Cuadro 4.1 [seg]							Desempeño del algoritmo											
			Serial	Paralelo						Speed-up						Eficiencia					
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos					
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	20,388	5.267,40	2.895,70	2.210	2.218,90	1.982,70	1.684,30	3,87	7,04	9,23	9,19	10,28	12,11	0,97	0,88	0,77	0,57	0,51	0,50

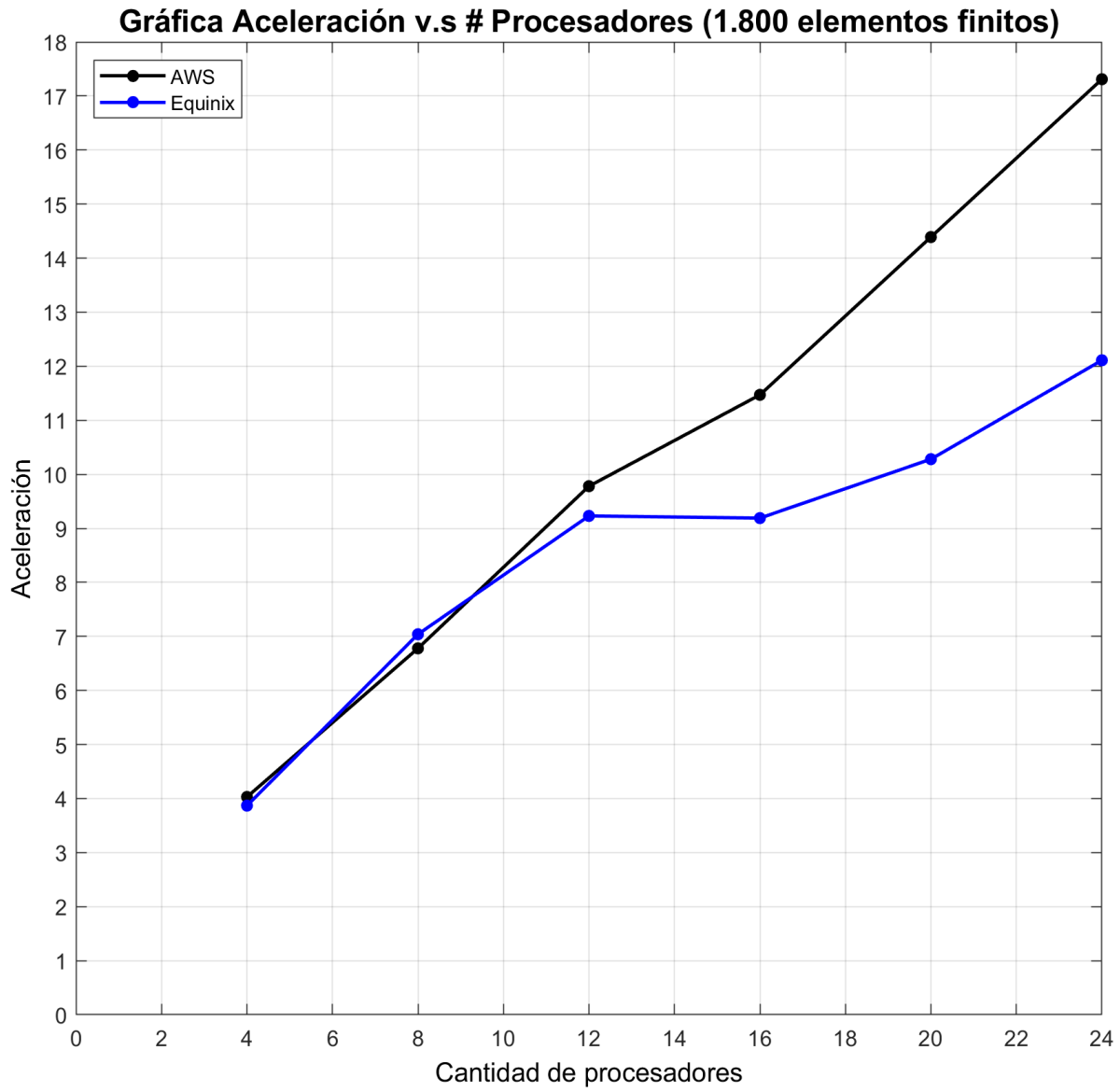


Figura 5.39.- Gráficas de escalabilidad para el bucle "for" del Cuadro 4.1., aceleración vs. número de procesadores para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

De igual manera, en la Figura 5.40 se presentan las gráficas de eficiencia tanto para la arquitectura de CPU en la máquina de AWS y Equinix. Para las gráficas de AWS, las eficiencias que se obtienen son de 1,01, 0,85, 0,81, 0,72, 0,72 y 0,72 cuando se utilizan 4, 8, 16, 20 y 24 núcleos respectivamente. En cambio, para las gráficas en Equinix las eficiencias que se obtienen son de 0,97, 0,88, 0,77, 0,57, 0,51 y 0,50 para la misma cantidad de núcleos.

Se destaca que el algoritmo es eficiente cuando se alcanzan eficiencias por encima de 1. Es así, que podemos observar que cuando se utilizan 4 núcleos en paralelo en la máquina de AWS se obtiene un valor de 1,01.

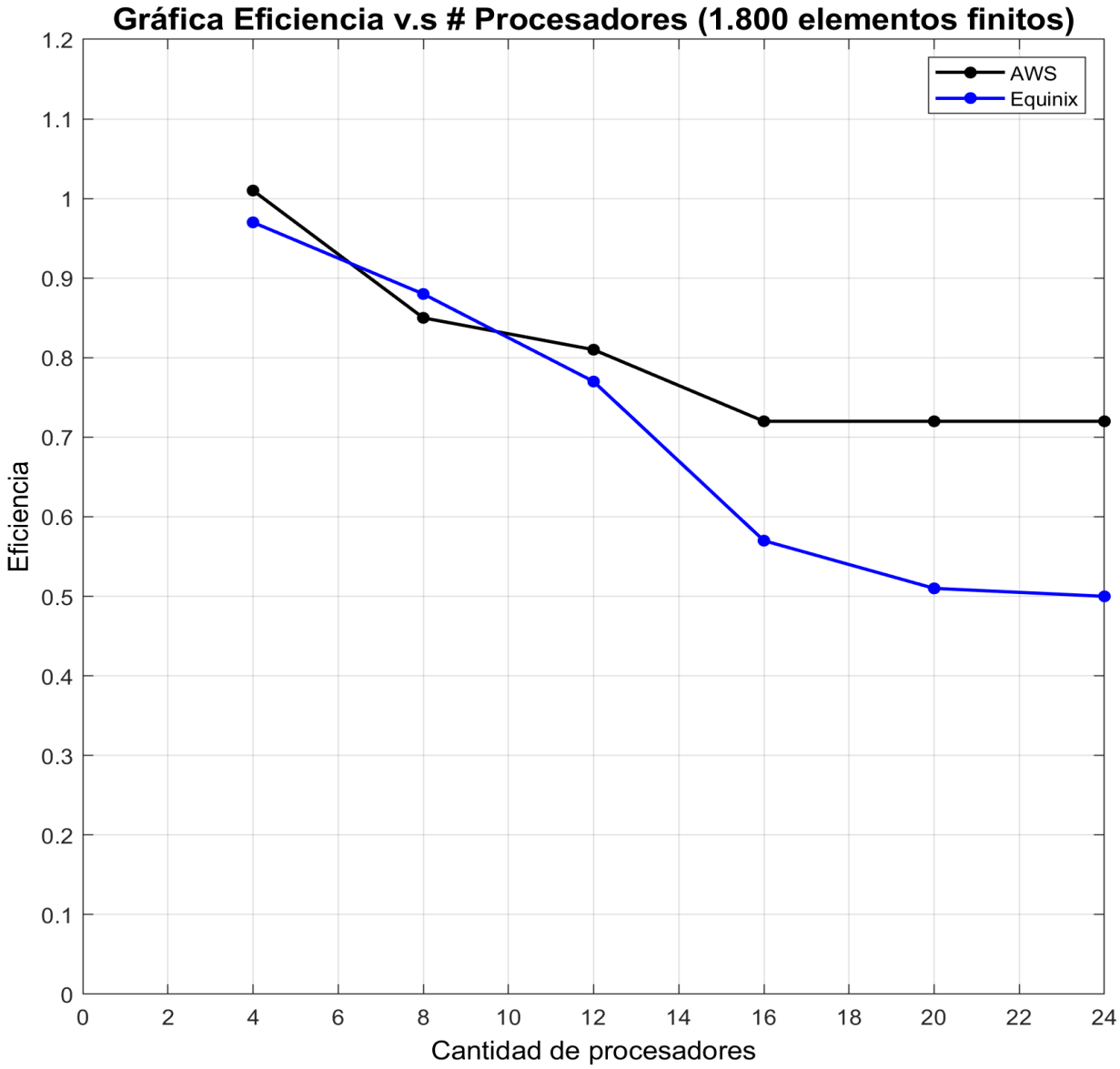


Figura 5.40.- Gráficas de eficiencia para el bucle “for” del Cuadro 4.1. para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

5.4.2.2.2.- DESEMPEÑO DEL BUCLE “FOR” DEL CUADRO 4.2 PARA MODELO DEL DISEÑO DE ÁLABES EN TURBOMÁQUINAS PARA EL CASO II.

En la Tabla 5.29. y Tabla 5.30 se presentan los tiempos computacionales (serial y paralelo) y los desempeños para el bucle “for” del Cuadro 4.2 tanto para la arquitectura de CPU utilizada en AWS y Equinix en la nube respectivamente.

De igual forma se presentan las gráficas de escalabilidad y de eficiencia para estas tablas, con el objetivo de poder tener un mejor análisis de la incidencia de este bucle “for” sobre todo el algoritmo MEF y MOT en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor. Es así como en la Figura 5.41 se presenta el gráfico de escalabilidad para el bucle “for” del Cuadro 4.2 tanto para la arquitectura de CPU proporcionada por AWS y Equinix.

En esta figura de escalabilidad, las aceleraciones que se obtienen para la gráfica de AWS son de 24,39x, 22,79x, 20,40x, 12,30x, 9,96x y 8,13x cuando se utilizan 4,8,12,16, 20 y 24 núcleos en paralelo. En cambio, para las gráficas de Equinix las aceleraciones que se obtienen son de 28,03x, 34,71x, 35,99x, 23,45x, 16,13x y 11,68x cuando se utilizan la misma cantidad de núcleos. De igual manera que la Figura 5.35 se puede observar que las aceleraciones se encuentran por debajo de la cantidad de núcleos a partir de utilizar 16 y 20, para la máquina virtual en AWS.

Adicionalmente, se destaca que los tiempos en seriales que se registraron son de 8.091,80 [seg] y 9.467,60 [seg] tanto para AWS y Equinix respectivamente. Es entonces que, si se analizan estos tiempos en seriales se puede notar un mayor tiempo computacional en el registro cuando se utiliza la máquina baremetal de Equinix en comparación a la de AWS, pero si se analizan los tiempos en paralelo se observa que se obtienen mejores resultados en la máquina de Equinix, lo que se traduce a tener mejores aceleraciones que en AWS.

Tabla 5.29.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) para el “for” paralelizable del Cuadro 4.2 utilizando la arquitectura de CPU en AWS.

Nr	Nteta	# de elementos finitos	Tiempo total del “for” paralelizarle del Cuadro 4.2 [seg]							Desempeño del algoritmo											
			Serial	Paralelo						Speed-up						Eficiencia					
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos					
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	8.091,80	331,80	355,11	396,76	657,63	812,83	995,26	24,39	22,79	20,40	12,30	9,96	8,13	6,09	2,85	1,7	0,77	0,50	0,34

Tabla 5.30.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) para el "for" paralelizable del Cuadro 4.2 utilizando la arquitectura de CPU en Equinix.

Nr	Nteta	# de elementos finitos	Tiempo total del "for" paralelizarle del Cuadro 4.2 [seg]							Desempeño del algoritmo											
			Serial	Paralelo						Speed-up						Eficiencia					
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos					
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	9,467,60	337,73	272,73	263,08	403,72	587,11	810,31	28,03	34,71	35,99	23,45	16,13	11,68	7,01	4,34	3,00	1,47	0,81	0,49

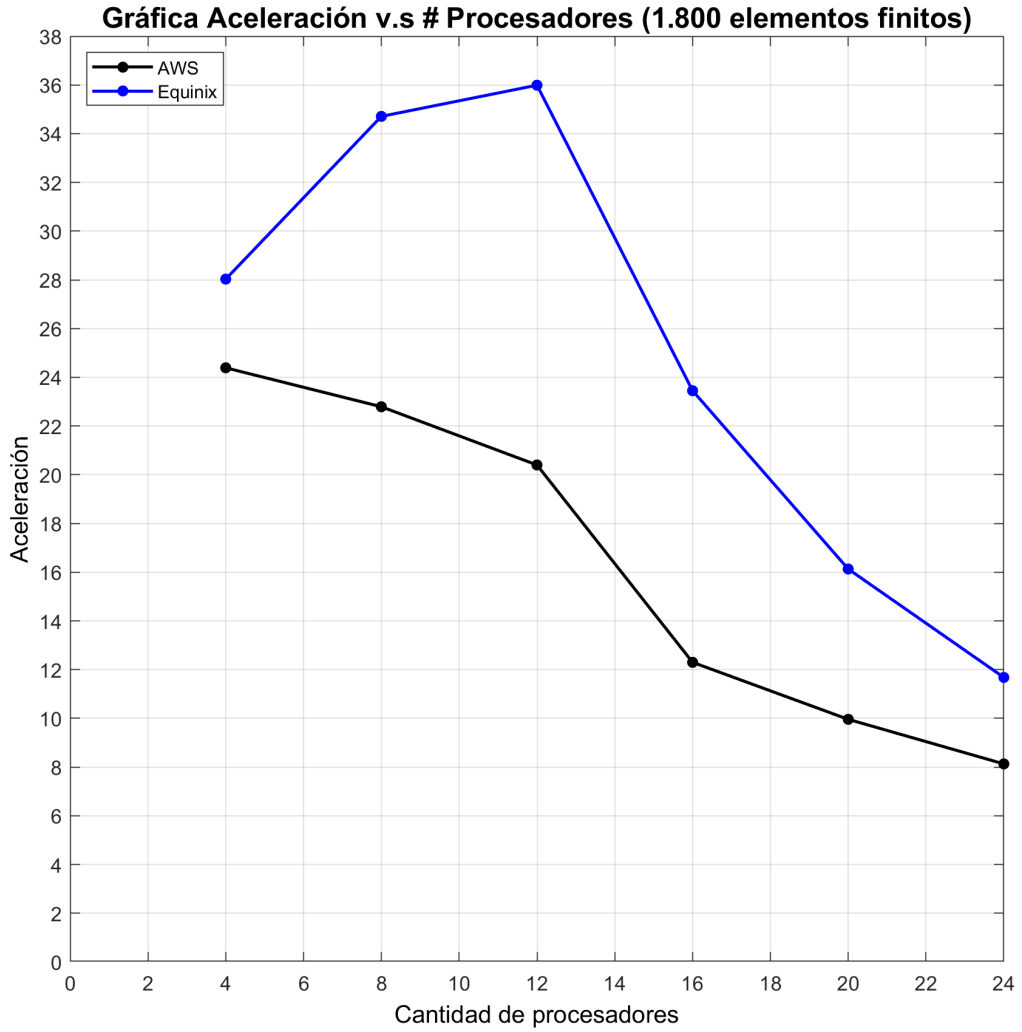


Figura 5.41.- Gráficas de escalabilidad para el bucle "for" del Cuadro 4.2, aceleración vs. número de procesadores para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

De igual manera, en la Figura 5.42 se presentan las gráficas de eficiencia tanto para la arquitectura de CPU en la máquina de AWS y Equinix. Para las gráficas de AWS, las eficiencias que se obtienen son de 6,09, 2,85, 1,7, 0,77, 0,50 y 0,34 cuando se utilizan 4, 8, 16, 20 y 24 núcleos respectivamente. En cambio, para las gráficas en Equinix las eficiencias que se obtienen son de 7,01, 4,34, 3,00, 1,47, 0,81 y 0,49 para la misma cantidad de núcleos.

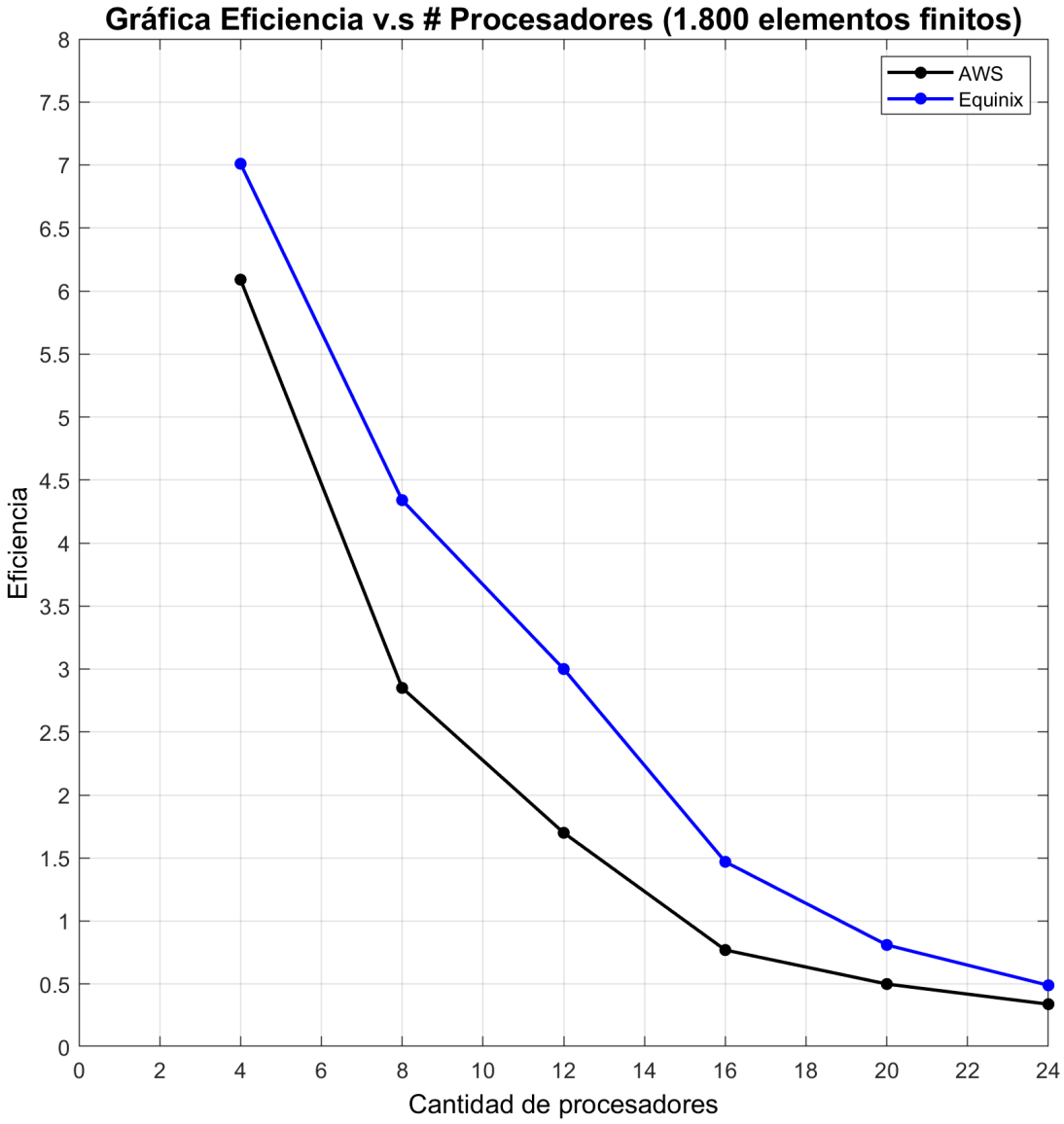


Figura 5.42.- Gráficas de eficiencia para el bucle “for” del Cuadro 4.2 para 1.800 elementos finitos en el modelo de diseño de álabes en turbomáquinas en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad $w_d = 0.8$ y $w_r = 0.2$) utilizando la arquitectura de CPU de AWS y Equinix en la nube.

5.4.2.2.3.- DESEMPEÑO DEL BUCLE “FOR” DEL CUADRO 4.3 PARA MODELO DEL DISEÑO DE ÁLABES EN TURBOMÁQUINAS PARA EL CASO II.

En la Tabla 5.31. y Tabla 5.32. se presentan los tiempos computacionales en serial y en paralelo en conjunto con el desempeño (escalabilidad y eficiencia) del bucle “for” paralelizado del pseudocódigo del Cuadro 4.3. tanto para la arquitectura de CPU proporcionada por AWS y Equinix respectivamente.

De la misma manera que el pseudocódigo del Cuadro 4.3. del Caso I, se puede observar en estas tablas que en vez de reducirse el tiempo computacional de este bucle a medida que se aumenta la cantidad de núcleos en paralelo pasa totalmente lo contrario, el tiempo aumenta. Es así como se observa que las aceleraciones en la Tabla 5.31. y Tabla 5.32. son menores que 1, lo que se traduce a una desaceleración debido al hecho de que el tiempo en paralelo es mucho mayor al tiempo en serial. Por lo tanto, se concluye que la paralelización de este bucle “for” no es paralelizable.

Tabla 5.31.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) para el “for” paralelizable del Cuadro 4.3 utilizando la arquitectura de CPU en AWS.

Nr	Nteta	# de elementos finitos	Tiempo total del “for” paralelizarle del Cuadro 4.3 [seg]							Desempeño del algoritmo												
			Serial	Paralelo						Speed-up						Eficiencia						
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos						
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
30	60	1.800	1,31	6,95	9,14	9,58	12,09	13,93	16,28	0,19	0,14	0,14	0,11	0,09	0,08	0,047	0,018	0,011	0,0068	0,0047	0,0034

Tabla 5.32.- Tiempo de cómputo totales y desempeño del algoritmo en el modelo del diseño de la topología del álabe en media circunferencia de rotor para el Caso II (Función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$) para el “for” paralelizable del Cuadro 4.3 utilizando la arquitectura de CPU en Equinix.

Nr	Nteta	# de elementos finitos	Tiempo total del “for” paralelizarle del Cuadro 4.3 [seg]							Desempeño del algoritmo												
			Serial	Paralelo						Speed-up						Eficiencia						
				Cantidad de núcleos						Cantidad de núcleos						Cantidad de núcleos						
				4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24	
30	60	1.800	1,37	6,29	6,65	7,28	8,71	9,47	10,38	0,22	0,21	0,19	0,16	0,14	0,13	0,054	0,026	0,016	0,0098	0,0072	0,0055	

5.5.- ANÁLISIS DE COSTOS DE ALQUILER DE MÁQUINAS EN LA NUBE

Una vez presentados los tiempos computacionales en serial y en paralelo junto con los desempeños de escalabilidad y eficiencia, en esta sección se decide evaluar el costo de la máquina virtual de AWS y la máquina baremetal de Equinix.

La forma de evaluar estos costos se realiza mediante el registro de tiempo en serial y en paralelo; es decir, se detallan los registros de tiempos totales en serie y en paralelo de los algoritmos computacionales que se describieron anteriormente: **a)** modelo de alabe recto, **b)** caso I: función mono-objetivo: minimización de energía de disipación, $w_d = 1$ y $w_r = 0$ y **c)** caso II: función bi-objetivo: minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$. El alquiler de las máquinas virtuales es evaluado por hora en base a los rubros que maneja AWS y Equinix por instancias/servidores para así poder obtener el costo total de ejecución de los algoritmos.

5.5.1.- MODELO DE ÁLABE RECTO

- 6.400 ELEMENTOS FINITOS

En la Tabla 5.33. y Tabla 5.34. se presentan los costos totales al ejecutar el MEF del modelo del álabe recto para 6.400 elementos finitos para la máquina de AWS y Equinix respectivamente. En este punto se destaca que el proveedor AWS maneja diferentes instancias con el fin de proveer diferentes recursos computacionales, es decir, cada instancia posee un número máximo de VCPUs con el objetivo de que el costo por hora sea el adecuado dependiendo de la cantidad de recursos computacionales que se utilice. En cambio, el proveedor Equinix no posee esta flexibilidad de proporcionar varios servidores, es decir solo posee una cantidad máxima de 24 núcleos en el servidor s3.xlarge.x86.

Para una mayor visualización de los costos totales de ejecución, en la Figura 5.43 se presenta en un diagrama de barras los costos entre la máquina virtual de AWS y baremetal de Equinix cuando se utilizan diferentes números de VCPUs/CPUs. En este diagrama de barras, para el modelo de álabe recto para 6.400 elementos finitos se concluye que cuando se ejecuta este algoritmo en serial y con 4 núcleos en paralelo, se generan menores costos en AWS (0,35$ y 0,33$) en comparación con los costos totales en Equinix (2,90$ y 0,57$), en cambio, a partir de 8 núcleos en adelante, resulta más conveniente ejecutar el algoritmo computacional del MEF en Equinix.

Tabla 5.33.- Costos Totales en AWS en la ejecución del modelo del álabe recto para 6.400 elementos finitos.

Instancia	# de VCPUs	Costo por hora $	Tiempo de ejecución [seg]	Costo Total $
m5ad.large	1	0,19	6.769,11	0,35
m5ad.2xlarge	4	0,78	1.555,36	0,33
m5ad.4xlarge	8	1,56	829,95	0,35
m5ad.8xlarge	16	3,12	550,47	0,47
m5ad.12xlarge	24	4,68	386,86	0,50

Tabla 5.34.- Costos Totales en Equinix en la ejecución del modelo del álabe recto para 6.400 elementos finitos.

Instancia	# de CPUs	Costo por hora $	Tiempo de ejecución	Costo Total $
s3.xlarge.x86	1	1,85	5.646,44	2,90
s3.xlarge.x86	4	1,85	1.109,82	0,57
s3.xlarge.x86	8	1,85	580,99	0,30
s3.xlarge.x86	16	1,85	413,21	0,21
s3.xlarge.x86	24	1,85	333,39	0,17

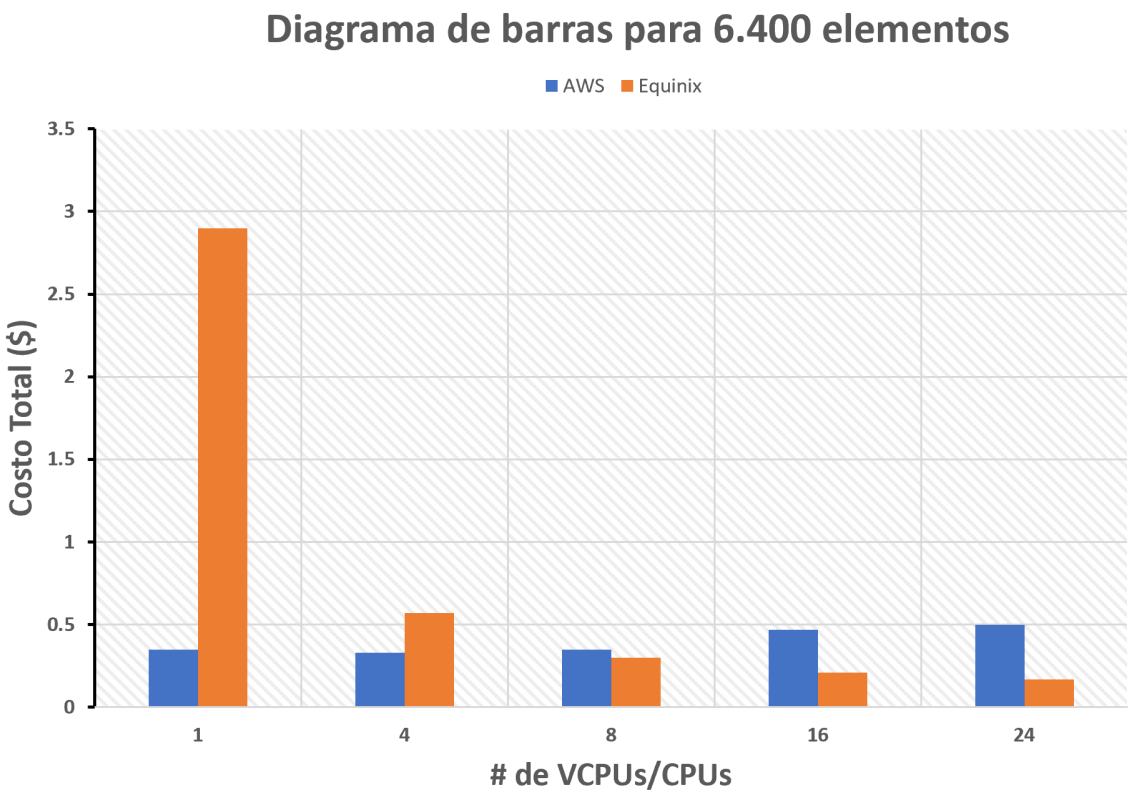


Figura 5.43.- Diagrama de barras para 6.400 elementos finitos para evaluación del costo total de ejecución del algoritmo computacional del modelo del álabe recto entre AWS y Equinix.

- 19.600 ELEMENTOS FINITOS

De la misma manera, en la Tabla 5.35. y Tabla 5.36. se muestran los costos totales al ejecutar el MEF del modelo del álabe recto para 19.600 elementos finitos para la máquina de AWS y Equinix respectivamente.

En la Figura 5.44 se presenta en un diagrama de barras los costos entre la máquina virtual de AWS y baremetal de Equinix cuando se utilizan diferentes números de VCPUs/CPUs cuando se ejecuta el algoritmo computacional del álabe recto para 19.600 elementos finitos. Como era de esperarse, igual que la gráfica anterior se puede observar en este diagrama de barras que cuando se ejecuta este algoritmo en serial y con 4 núcleos en paralelo se generan menores costos en AWS de 3,33$ y 3,04$ en el tiempo total de ejecución en comparación con los costos totales en Equinix de 27,25$ y 5,01$, en cambio, a partir de 8 núcleos en adelante, resulta más conveniente ejecutar el algoritmo computacional del MEF en Equinix.

Tabla 5.35.- Costos Totales en AWS en la ejecución del modelo del alabe recto para 19.600 elementos finitos.

Instancia	# de VCPUs	Costo por hora $	Tiempo de ejecución [seg]	Costo Total $
m5ad.large	1	0,19	63.135,32	3,33
m5ad.2xlarge	4	0,78	14.047,88	3,04
m5ad.4xlarge	8	1,56	7.260,22	3,15
m5ad.8xlarge	16	3,12	4.562,01	3,95
m5ad.12xlarge	24	4,68	3.109,62	4,04

Tabla 5.36.- Costos Totales en Equinix en la ejecución del modelo del alabe recto para 19.600 elementos finitos.

Instancia	# de CPUs	Costo por hora $	Tiempo de ejecución	Costo Total $
s3.xlarge.x86	1	1,85	53.027,7	27,25
s3.xlarge.x86	4	1,85	9.756,61	5,01
s3.xlarge.x86	8	1,85	5.181,24	2,66
s3.xlarge.x86	16	1,85	3.337,60	1,72
s3.xlarge.x86	24	1,85	2.622,67	1,35

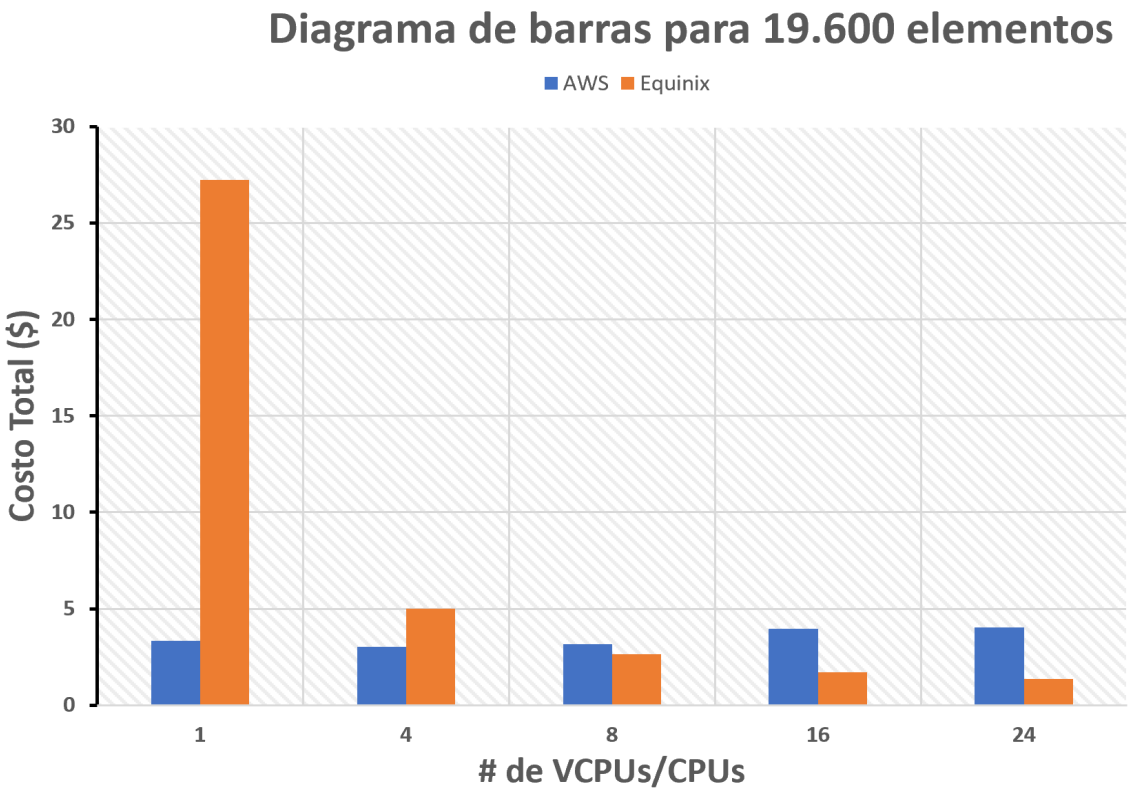


Figura 5.44.- Diagrama de barras para 19.600 elementos finitos para evaluación del costo total de ejecución del algoritmo computacional del modelo del álabe recto entre AWS y Equinix.

- 40.000 ELEMENTOS FINITOS

En la Tabla 5.37. y Tabla 5.38. se presentan los costos totales al ejecutar el MEF del modelo del álabe recto cuando se utilizan 40.000 elementos finitos tanto en la máquina de AWS y Equinix respectivamente.

En la Figura 5.45 se observa la misma tendencia que las gráficas anteriores, que cuando se corre el código en serial y en paralelo con 4 núcleos, el costo total en la máquina de Equinix es superior al costo en AWS, pero, a partir de 8 núcleos en adelante es favorable ejecutar el algoritmo en Equinix que en AWS debido a que se generan menores costos en la máquina baremetal. Cabe mencionar que el precio en serial para la máquina en Equinix es demasiado alto llegando a un valor de 117,49 $ dólares americanos en comparación con la máquina en AWS que el costo es solo de 12,96 $ dólares americanos.

Tabla 5.37.- Costos Totales en AWS en la ejecución del modelo del álabe recto para 40.000 elementos finitos.

Instancia	# de VCPUs	Costo por hora $	Tiempo de ejecución [seg]	Costo Total $
m5ad.large	1	0,19	245.478	12,96
m5ad.2xlarge	4	0,78	59.980	13,00
m5ad.4xlarge	8	1,56	29.801	12,91
m5ad.8xlarge	16	3,12	18.102	15,69
m5ad.12xlarge	24	4,68	12.527,1	16,29

Tabla 5.38.- Costos Totales en Equinix en la ejecución del modelo del álabe recto para 40.000 elementos finitos.

Instancia	# de CPUs	Costo por hora $	Tiempo de ejecución	Costo Total $
s3.xlarge.x86	1	1,85	228.636	117,49
s3.xlarge.x86	4	1,85	40.828,24	20,98
s3.xlarge.x86	8	1,85	20.969,96	10,78
s3.xlarge.x86	16	1,85	13.196	6,78
s3.xlarge.x86	24	1,85	10.620,2	5,46

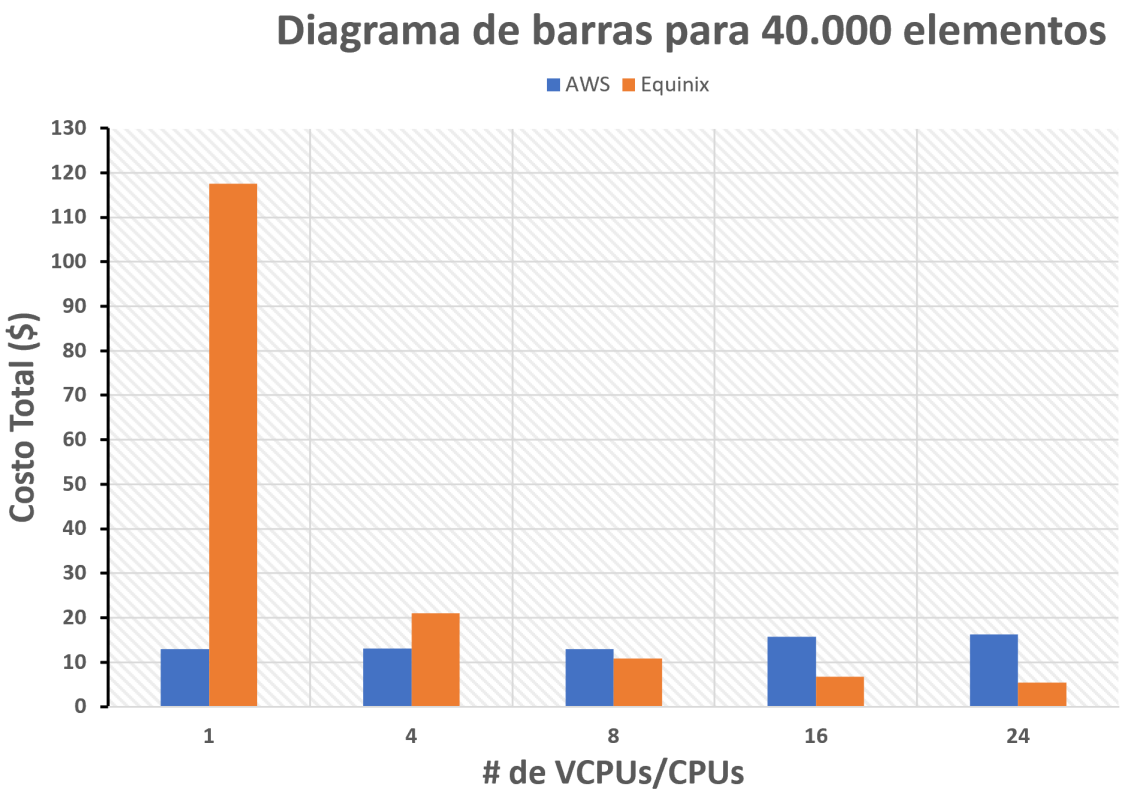


Figura 5.45.- Diagrama de barras para 40.000 elementos finitos para evaluación del costo total de ejecución del algoritmo computacional del modelo del álabe recto entre AWS y Equinix.

- 78.400 ELEMENTOS FINITOS

En la Tabla 5.39. y Tabla 5.40. se presentan los costos totales para este mismo modelo para 78.400 elementos finitos para la máquina proporcionada por AWS y Equinix respectivamente.

En la Figura 5.46 se presenta el diagrama de barras de los costos totales para este modelo de la máquina de AWS y Equinix. En este punto se destaca la importancia de tener un algoritmo paralelizado debido a los costos que se pudieran generar si se corriera solo en serial, ya que en AWS el costo total es de 55,89$ dólares y en Equinix 453,48$ dólares cuando se ejecuta el algoritmo MEF en serial. De igual forma se destaca que a partir de 8 núcleos en adelante, es favorable ejecutar el algoritmo MEF en la máquina de Equinix debido a un menor costo total en comparación con la máquina de AWS.

Cabe destacar que aumentar la cantidad de elementos finitos en el modelado de flujo de las ecuaciones de Navier-Stokes para diseño de álabes en turbomáquinas tiene su importancia debido al hecho de que se pueden captar fenómenos (vorticidad, trayectoria de las partículas, etc.) que no se captarían con profundidad si se usa poca cantidad de elementos finitos. Es así, que se deja en claro la importancia de tener un algoritmo paralelizado.

Tabla 5.39.- Costos Totales en AWS en la ejecución del modelo del álabe recto para 78.400 elementos finitos.

Instancia	# de VCPUs	Costo por hora $	Tiempo de ejecución [seg]	Costo Total $
m5ad.large	1	0,19	1’058.947	55,89
m5ad.2xlarge	4	0,78	243.957	52,86
m5ad.4xlarge	8	1,56	126.435	54,79
m5ad.8xlarge	16	3,12	67.761	58,73
m5ad.12xlarge	24	4,68	47.571	61,84

Tabla 5.40.- Costos Totales en AWS en la ejecución del modelo del álabe recto para 78.400 elementos finitos.

Instancia	# de CPUs	Costo por hora $	Tiempo de ejecución	Costo Total $
s3.xlarge.x86	1	1,85	882.456	453,48
s3.xlarge.x86	4	1,85	164.836,3	84,71
s3.xlarge.x86	8	1,85	80.149,30	41,19
s3.xlarge.x86	16	1,85	50.545,28	25,98
s3.xlarge.x86	24	1,85	39.755	20,43

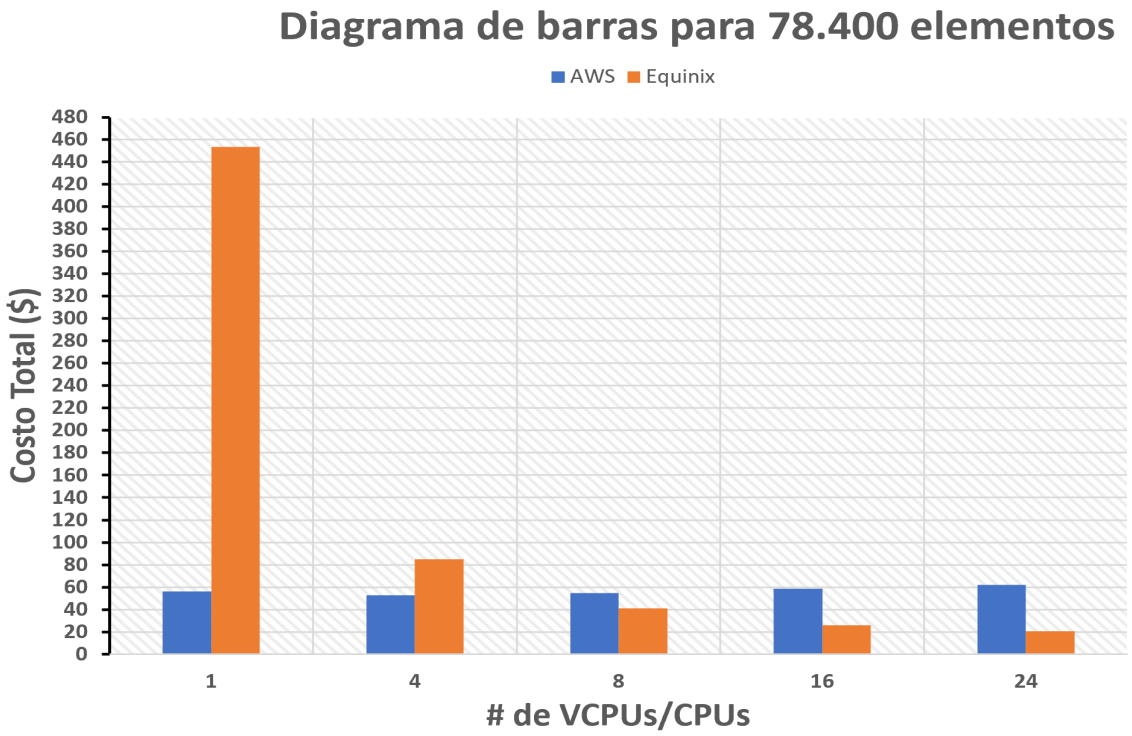


Figura 5.46.- Diagrama de barras para 78.400 elementos finitos para evaluación del costo total de ejecución del algoritmo computacional del modelo del álabe recto entre AWS y Equinix.

5.5.2.- MODELO MEDIA CIRCUNFERENCIA DE ROTOR EN EL DISEÑO DE ÁLABES EN TURBOMÁQUINAS

- CASO I: FUNCIÓN MONOBJETIVO EN LA MINIMIZACIÓN DE LA ENERGÍA DE DISIPACIÓN, $w_d = 1$ y $w_r = 0$

En la Tabla 5.41. y Tabla 5.42. se presentan los costos totales cuando se ejecuta el algoritmo MEF y MOT en el diseño de álabes para turbomáquinas en media circunferencia de rotor para la máquina AWS y Equinix respectivamente.

De igual manera que el modelo del álabe recto, en la Figura 5.47 se presenta el diagrama de barras de los costos totales para este modelo en la máquina de AWS y Equinix. A pesar de que los costos totales en AWS y en Equinix no son muy elevados para este modelo, cabe destacar que se cumple la misma tendencia que las gráficas anteriores, es decir, cuando se ejecuta en serial y en paralelo con la utilización de 4 núcleos, es más conveniente ejecutar el algoritmo en AWS. En cambio, a partir de 8 núcleos en adelante se recomienda la utilización de Equinix.

Adicionalmente, si se quisiera comparar una máquina de Equinix con AWS con las mismas características en la cantidad de núcleos y frecuencia, se puede tomar como ejemplo las instancias, m5ad.12xlarge de AWS y el servidor s3.xlarge.x86. Si comparamos el precio por hora, se puede observar que AWS posee un precio más elevado que Equinix. Este precio elevado de

AWS se debe al sistema operativo de la máquina virtual que se selecciona (Windows server 2019), ya que AWS eleva mucho sus costos por hora por requerir esta licencia. En cambio, a pesar de que se usa el mismo sistema operativo en Equinix, este proveedor no eleva sus precios sea cual sea el sistema operativo, se mantiene con el mismo precio estándar.

Una solución viable para obtener menores costos por hora en alquiler de las máquinas virtuales en AWS sería utilizar el sistema operativo de Linux. A manera de ejemplo, esta misma instancia m5ad.12xlarge en sistema operativo de Linux para AWS tiene un costo de alquiler por hora de 1,55$ dólares, un 16,22% más barato que la instancia de Equinix s3.xlarge.86, ya que con un sistema operativo también de Linux el costo sería el mismo, de 1,85$ por alquiler de hora. En este punto se destaca que el buen uso de los recursos en Equinix o en AWS influye notablemente en el costo final de alquiler. Es por eso que se sugiere siempre tener un conocimiento previo de los recursos que puede ofrecer cada proveedor debido a que esto podría generar ahorros antes de la ejecución de estas máquinas en la nube.

Tabla 5.41.- Costos Totales en AWS en la ejecución del modelo de media circunferencia de rotor para el Caso I: función mono-objetivo en la minimización de la energía de disipación, $w_d = 1$ y $w_r = 0$.

Instancia	# de VCPUs	Costo por hora $	Tiempo de ejecución [seg]	Costo Total $
m5ad.large	1	0,19	13.033	0,69
m5ad.2xlarge	4	0,78	3.361	0,73
m5ad.4xlarge	8	1,56	2.174	0,94
m5ad.8xlarge	12	3,12	2.085	1,81
m5ad.8xlarge	16	3,12	1.949	1,69
m5ad.12xlarge	20	4,68	1.815	2,36
m5ad.12xlarge	24	4,68	1.741	2,26

Tabla 5.42.- Costos Totales en Equinix en la ejecución del modelo de media circunferencia de rotor para el Caso I: función mono-objetivo en la minimización de la energía de disipación, $w_d = 1$ y $w_r = 0$.

Instancia	# de CPUs	Costo por hora $	Tiempo de ejecución	Costo Total $
s3.xlarge.x86	1	1,85	9.646	4,96
s3.xlarge.x86	4	1,85	2.399	1,23
s3.xlarge.x86	8	1,85	1.621	0,83
s3.xlarge.x86	12	1,85	1.430	0,74
s3.xlarge.x86	16	1,85	1.428	0,73
s3.xlarge.x86	20	1,85	1.419	0,73
s3.xlarge.x86	24	1,85	1.418	0,73

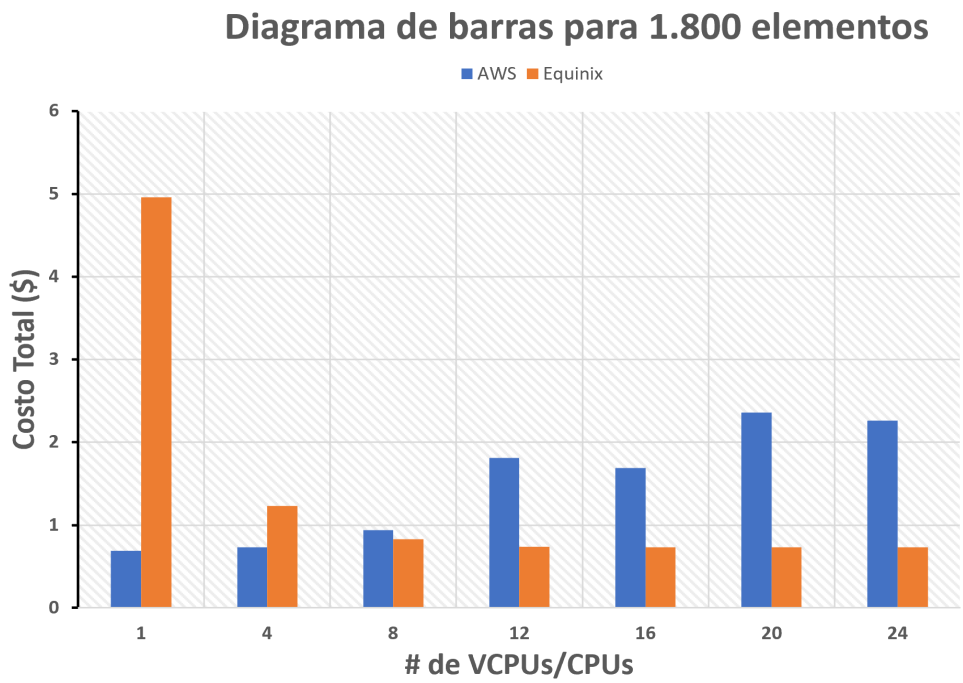


Figura 5.47.- Diagrama de barras para 1.800 elementos finitos para evaluación de costos totales en la ejecución del algoritmo computacional del modelo de media circunferencia de rotor en el diseño de álabes en turbomáquinas para el Caso I: función mono-objetivo en la minimización de energía de disipación, $w_d = 1$ y $w_r = 0$, para las máquinas AWS y Equinix.

- CASO II: FUNCIÓN BI-OBJETIVO EN LA MINIMIZACIÓN DE LA ENERGÍA DE DISIPACIÓN Y VORTICIDAD, $w_d = 0.8$ y $w_r = 0.2$.

En la Tabla 5.43. y Tabla 5.44. se muestran los costos totales cuando se ejecuta el algoritmo MEF y MOT en el diseño de álabes para turbomáquinas en media circunferencia de rotor para el Caso II en la máquina de AWS y de Equinix respectivamente.

En la Figura 5.48 se presenta el diagrama de barras de los costos totales para este modelo para la máquina de AWS y Equinix respectivamente. En este caso II, a pesar de que se obtiene la misma tendencia que el gráfico anterior (menores costos totales en Equinix a partir utilizar 8 núcleos en paralelo), el efecto de la escalabilidad (ver Figura 5.37) es una métrica importante a considerar en los costos totales de la máquina de AWS y Equinix, ya que por más VCPUs o núcleos de CPUS se consideren en estos servidores, el algoritmo tiene un límite de paralelización (no se puede escalar más), y en vez de abaratar los “costos totales” en estos servidores por el alquiler total de las horas, realmente lo que se está haciendo es encarecer los costos totales, esto se puede ver reflejado más notoriamente en el servidor de AWS de la Tabla 5.43.

Tabla 5.43.- Costos Totales en AWS en la ejecución del modelo de media circunferencia de rotor para el Caso II: función bi-objetivo en la minimización de la energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$.

Instancia	# de VCPUs	Costo por hora $	Tiempo de ejecución [seg]	Costo Total $
m5ad.large	1	0,19	44.222,36	2,33
m5ad.2xlarge	4	0,78	11.517,01	2,50
m5ad.4xlarge	8	1,56	7.459,47	3,23
m5ad.8xlarge	12	3,12	7.056,43	6,12
m5ad.8xlarge	16	3,12	6.623,45	5,74
m5ad.12xlarge	20	4,68	6.192	8,05
m5ad.12xlarge	24	4,68	5.907,60	7,68

Tabla 5.44.- Costos Totales en Equinix en la ejecución del modelo de media circunferencia de rotor para el Caso II: función bi-objetivo en la minimización de la energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$.

Instancia	# de CPUs	Costo por hora $	Tiempo de ejecución	Costo Total $
s3.xlarge.x86	1	1,85	32.623,81	16,77
s3.xlarge.x86	4	1,85	8.107,12	4,17
s3.xlarge.x86	8	1,85	5.599,63	2,88
s3.xlarge.x86	12	1,85	4.934,97	2,54
s3.xlarge.x86	16	1,85	4.843,14	2,49
s3.xlarge.x86	20	1,85	4.853,82	2,49
s3.xlarge.x86	24	1,85	4.742,85	2,44

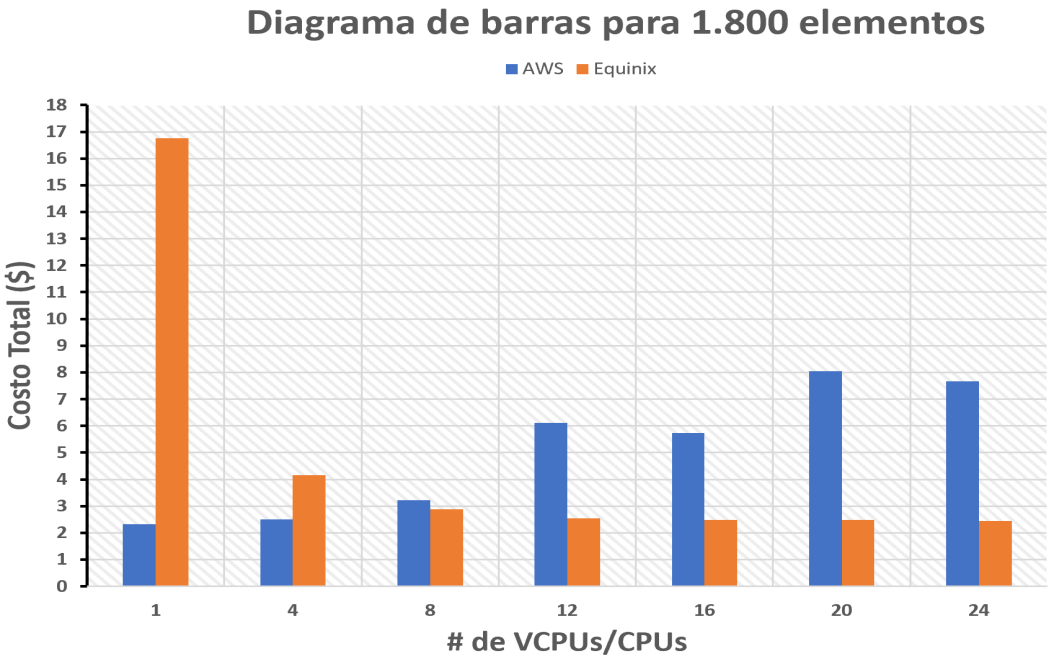


Figura 5.48.- Diagrama de barras para 1.800 elementos finitos para evaluación de los costos totales en la ejecución del algoritmo computacional del modelo de media circunferencia de rotor en el diseño de álabes en turbomáquinas para el Caso II: función bi-objetivo en la minimización de energía de disipación y vorticidad, $w_d = 0.8$ y $w_r = 0.2$, para las máquinas AWS y Equinix.

CAPÍTULO 6: CONCLUSIONES Y TRABAJOS FUTUROS

El diseño de rotores en turbomáquinas específicamente en bombas de flujo radial es un problema tan complejo que el conocimiento empírico y la experiencia del diseñador no es suficiente para proponer alternativas de diseño que se traduzcan en mejores prestaciones. Adicionalmente, la importancia de reducir costos y tiempo de simulación en el diseño de álabes en turbomáquinas es un factor que juega un papel muy importante en el diseño de estos dispositivos. Bajo este contexto, el método de optimización topológica utilizando el MMA en la minimización de una función objetivo ha demostrado ser un método exitoso en el diseño de álabes para estas máquinas rotativas.

Es así, que, mediante el desarrollo de un algoritmo de optimización topológica paralelizado en la nube, se abordó el diseño de rotores de turbomáquinas mediante el uso de una función bi-objetivo, donde se minimiza los factores del campo de fluidos (la disipación de energía y la vorticidad). De esta manera, se obtienen topologías que representan el requerimiento otorgado dependiendo de la priorización de minimización que se dé a estas funciones objetivos por medio de los pesos de ponderación, proporcionando así un balance entre estos dos fenómenos físicos.

De igual manera, la paralelización del algoritmo computacional en el modelo de diseño de álabes en media circunferencia de rotor se considera exitosa. El hecho de reducir el tiempo de ejecución a más de la mitad, aumenta la robustez del algoritmo desarrollado, lo que permite evaluar con mayor rapidez los diferentes parámetros del MOT (necesario para la obtención de topologías finales no intuitivas), obteniendo como resultado un desempeño optimizado en el diseño de álabes. Adicionalmente, se aprecia que la penalización del modelo de material Brinkman es lo suficientemente fuerte para representar zonas sólidas y de fluido, dando como resultado la obtención de topologías que representan la minimización física del problema.

Se deja claro que la metodología empleada permite extender el estudio de otras funciones objetivos del problema de diseño de turbomáquinas. Este estudio puede ir más allá del hidrodinámico y extenderse así a funciones objetivos estructurales que estudien las partes mecánicas que lo conforman, por ejemplo, el estudio de los esfuerzos que se generan en el rotor de una turbomáquina por medio de la interacción con el fluido en cuestión.

En sí, se deja plasmado una herramienta lo suficientemente robusta que genera diseños novedosos y que facilita a los diseñadores salir de métodos convencionales de cálculo.

Asimismo, se observa que la mejora de modelos matemáticos a través de los años en el modelamiento y optimización han venido ofreciendo diversas alternativas de solución cada vez más novedosas, que asemejan el modelamiento computacional a la física del problema en cuestión. Y a raíz del desarrollo de la paralelización en CPU, los aportes científicos han tenido su repunte hacia el estudio de fenómenos físicos que anteriormente no se abordaban. Es por eso que a continuación se presentan algunos puntos donde este trabajo puede ser extendido en trabajos futuros:

- Incrementar la complejidad del diseño de álabes en rotores de turbomáquinas mediante la incorporación de fluidos no-newtonianos, simulando fluidos como la sangre que asemejen el estudio a dispositivos como bombas de asistencia ventriculares.

- Extender el estudio en la obtención de nuevas topologías finales en el diseño de álabes, mediante la inclusión de otros regímenes de flujo, incluyendo flujos turbulentos y comportamientos transitorios en las ecuaciones de Navier-Stokes.

- Incorporar nuevas funciones objetivos con el fin de abordar estudios de otra índole, así como el análisis estructural de rotores donde se minimice la flexibilidad (maximización de rigidez) o funciones que minimicen las cargas térmicas generadas en las turbinas que provocan perdidas en la eficiencia y fatiga en los componentes mecánicos que lo conforman.

- Estudiar a mayor profundidad el filtro y proponer unos más avanzados como el filtro de proyección y verificar su comportamiento en el diseño de álabes en turbomáquinas.

- Evaluar otras metodologías de métodos acelerados, así como las que se mencionó en la Figura 4.7. Adicionalmente, se podría utilizar la herramienta de “DISTRIBUTED COMPUTING SERVER” que ofrece MATLAB, con el fin de evaluar el algoritmo paralelizado utilizando el paradigma similar al de memoria distribuida, MPI. AWS ofrece el servicio de clústeres donde la comunicación entre nodos sería ideal para la paralelización del algoritmo.

CAPITULO 7: BIBLIOGRAFIA

[1] AARONSON, K. D.; SLAUGHTER, M.S, MILLER, L., "Use of an intrapericardial, continuos-flow, centrifugal pump in patients awaiting heart transplantation", Circulation, v. 125, n. 25, p. 3191-3200, jun 2012. ISSN 0009-7322.

[2] B. Jafarzadeh, A. Hajari, M.M. Alishahi, M. Akbari, "The flow simulation of a low-specific- speed high-speed centrifugal pump", Appl. Math. Model. 35 (2011) 242–249.

[3] K. Cheah, T.S. Lee, S.H. Winoto, Z. Zhao, "Numerical flow simulation in a centrifugal pump at design and off-design conditions", Internat. J. Rotat. Mach. 2007 (2007) 1–8.

[4] J.S. Romero (2014), "A topology optimization approach applied to laminar flow machine rotor design", Comput. Methods Appl. Mech. Engrg. 279 (2014) 268–300, Department of Mechanical Engineering, Federal University of Espirito Santo, ES, Brazil.

[5] Frank White,Fluids Mechanics, McGraw-Hill, New York, Sext Edition (2008).

[6] M. P. Bendsoe y O. Sigmund, "Topology optimization: theory, methods, and applications". Springer, 2003.

[7] O. Sigmund, "Design of multiphysics actuators using topology optimization - part I: One-material structures", Computer Methods in Applied Mechanics and Engineering, vol. 190, no. 49-50, pags. 6577-6604, 2001, Department of Mechanical Engineering, Solid Mechanics, the Technical university of Denmark.

[8] D. Hutton, "Comparison of Finite Element and Exact Solutions", in Fundamentals of Finite Element Analysis, USA: McGraw−Hill, 2004, pags. 4-7.

[9] Sun-Sheng Yang, "Theoretical, numerical and experimental prediction of pump as turbine performance", Renewable Energy 48 (2012) 507-513, Research Center of fluid Machinery Engineering and Technology, Jiangsu University, China.

[10] K.W. Cheah, "Numerical flow simulation in a centrifugal pump at design and off-design conditions", Singapore, Hindawi Publishing Corporation International Journal of Rotating Machinery Volume 2007, Article ID 83641, 8 pages.

[11] B. Kitchenham, "Procedures for performing systematic reviews", Keele, UK, Keele University, vol. 33, 2004.

[12] Hedi, L., Hatem, K.,Ridha, Z., 2010."Numerical flow simulation in a centrifugal pump," International Renewable Energy Congress. Sousse, Tunisia. pp. 300-304.

[13] Mentzos, M.,Filios, A.,Margaris, P.,Papanikas, D.,2004. "A numerical simulation of the impeller-volute interaction in a centrifugal pump", Proceedings of International Conference from Scientific Computing to Computational Engineering. Athens, pp. 1-7.

[14] Mentzos, M., Filios, A., Margaris, P., Papanikas, D., 2005. "CFD predictions of flow through a centrifugal pump impeller," Proceedings of International Conf. Experiments/Process/System Modelling/ Simulation/Optimization. Athens, pp. 1-8.

[15] Shah, S., Jain, S., Lakhera, V., 2010. "CFD based flow analysis of centrifugal pump," Proceedings of International Conference on Fluid Mechanics and Fluid Power. Chennai, India, paper#TM08.

[16] Medvitz, R., Kunz, R., Boger, D., Adam, J.,Yocum, A., 2002. "Performance analysis of cavitating flow in centrifugal pumps using multiphase cfd", Journal of Fluid

Engineering 124, p. 377, Applied Research Laboratory, The Pennsylvania State University.

[17] Nohmi, M.,Goto, A.,Iga, Y.,Ikohagi, T., 2003.“Cavitation CFD in a centrifugal pump,” Proceedings of International Symposium on Cavitation. Osaka, Japan, pp. 1-7.

[18] Caridad, J., Asuaje, M.,Kenyery, F.,Tremante, A.,Aguillon, O., 2008. “Characterization of a centrifugal pump Impeller under two-phase flow conditions”, Journal of Petroleum Science and Engineering 63,p. 18.

[19] Williams, A., "Pumps as turbines for low-cost micro hydropower", Renew. Energy 1996, vol. 9, pp. 1227–1234, September-december 1996.

[20] S. Natanasabapathi and J. Kshirsagar, "Pump As Turbine-An Experience With CFX-5.6," Corporate research and engineering division. Pune, India, Kirloskar Bros. Ltd, 2004.

[21] Fernandez, J., Barrio, R., Blanco, E.,Parrondo, J., Marcos, A., 2010. Numerical Investigation of a Centrifugal Pump Running in Reverse Mode. Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy 224, p. 373.

[22] Barrio, R., Fernandez, J., Blanco, E., Parrondo, J., Marcos, A., 2011. Performance Characteristics and Internal Flow Patterns in a Reverse-Running Pump–Turbine. Proceedings of IMechE Vol. 226 Part C: J. Mechanical Engineering Science, p. 695.

[23] Jasmina B., Bogdanovi-Jovanovi, “Pumps used as a Turbines, Power Recovery, Energy Efficiency, CFD Analysis” Faculty of Mechanical Engineering, University of Nis, Nis, Serbia, Jaroslav Cerni Institute for the Development of Water Resources, Belgrade, Serbia.

[24] Carlos M. OkuboJr., César Y. Kiyono, Luís F.N. Sá, Emílio C.N. Silva, “Topology optimization applied to 3D rotor flow path design based on the continuous adjoint approach”, Computers and Mathematics with Applications, vol. 96, pp. 16-30, August 2021.

[25] Khaled Alawadhi, Bashar Alzuwayer, Tareq Ali Mohammad and Mohammad H. Buhemdi, “Design and optimization of a centrifugal pump for slurry transport using the response surface method”, MDPI, 9 (3):60, March 2021.

[26] L.F.N.Sá, J.S.Romero, “Topology optimization applied to the development of small scale pump” Structural and Multidisciplinary Optimization (2016) 57:2045–2059, Department of Mechatronics and Mechanical Systems, Engineering of Escola Politecnica, University of Sao Paulo.

[27] Shahram Derakhshan, “Investigation of an efficient shape optimization procedure for centrifugal pump impeller using eagle strategy algorithm and ANN (case study: slurry flow)” Structural and Multidisciplinary Optimization.

[28] L.F.N.Sá, A. A. Novotny, “Design optimization of laminar flow machine rotors based on the topological derivative concept”, Struct Multidisc Optim (2017) 56:1013–1026, Department of Mechatronics and Mechanical Systems Engineering of Escola Politecnica, University of Sao Paulo.

[29] J.S.Romero, E.C. N.Silva, “Non-newtonian laminar flow machine rotor design by using topology optimization” Structural and Multidisciplinary Optimization, Department of Mechatronics and Mechanical Systems Engineering of Escola Politecnica, University of Sao Paulo.

[30] Shahram Derakhshan, “Numerical shape optimization of a centrifugal pump impeller using artificial bee colony algorithm” Computers & Fluids 81 (2013) 145–151, School of Mechanical Engineering, Iran University of Science & Technology.

[31] Bacharoudis, E.,Filios, A.,Mentzos, M.,Margaris, “Parametric study of a centrifugal pump impeller by varying the outlet blade angle”, The Open Mechanical Engineering Journal 2, p. 75.

[32] Anagnostopoulos, “CFD Analysis and design effects in a radial pump impeller”, WSEAS Transactions on Fluid Mechanics, vol. 1 (7), pp. 763, January 2006.

[33] Gatta, S.,Salvadori, S., Adami, P.,Bertolazzi, “CFD study for assessment of axial thrust balance in centrifugal multistage pumps”, International Conferenceon Fluid Flow Technologies.

[34] Dixon, J. a, Verdicchio, J. a, Benito, D., Karl, A., & Tham, K. M. (2004), “Recent developments in gas turbine component temperature prediction methods, using computational fluid dynamics and optimization tools, in conjunction with more conventional finite element analysis techniques”, Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy, 218(4), 241–255.

[35] Mustafa Golcu, “Energy saving in a deep well pump with splitter blade” Energy Conversion and Management 47 (2006) 638–651, Department of Mechanical Education, Pamukkale University, 20017 Kinikli, Denizli, Turkey.

[36] Sun-Sheng Yang, “Theoretical, numerical and experimental prediction of pump as turbine performance” Renewable Energy 48 (2012) 507-513, Research Center of Fluid Machinery Engineering and Technology, Jiangsu University.

[37] S.C.M. Yu, “The flow patterns within the impeller passages of a centrifugal blood pump model” Medical Engineering & Physics 22 (2000) 381–393, Nanyang Technological University, Thermal and Fluids Engineering Division, School of Mechanical and Production Engineering, Singapore.

[38] Kun-Xi, “Haemodynamic approach to reducing thrombosis and haemolysis in an impeller pump” Scientific and Technical Record, Republic of China.

[39] Juan David Berrío, “Optimización topológica de un instrumento musical idiófono tipo metalófono” Tesis de Maestría de Investigación, Universidad Nacional de Medellín.

[40] SRShah, ≪CFD for centrifugal pumps: a review of the state-of-the-art, Ahmedabad, India, Procedia Engineering 51 (2013) 715 – 720.

[41] << Ventricular Assist devices> www.texasheart.org/heart-health/heart-information Center/topics/ventricular-assist-devices/ [Accesed: 3-Nov-2018].

[42] Yongbo Deng,Yihui Wu,Zhenyu Liu, Topology optimization Theory for Laminar Flow, Applications in inverse Design of Microfluidics.

[43] Maatoug Hassine ESSTH, Sousse University Tunisia, Topology Optimization of fluid Mechanics Problems.

[44] Thomas Borrvall and Joakim Petersson, <<Topology Optimization of fluids in Stokes flow>>, International Journal for Numerical Methods in fluids, 2003.

[45] NitelMuhtaroglu “Democratization of HPC cloud services with automated parallel solvers and applications containers”, Concurrency Computation Practice and Experience, 2018.

[46] Jin “Research on Optmization Algorithm of BP neural network for permanent magnet synchronous TOMor based on Cloud Computing”, Institute of Electrical and Electronics Engineers Inc, 2018.

[47] K. Konopka “Improving efficiency of CCS numerical simulations through use of parallel processing”, Archives of Metallurgy and Materials, Volume 60, Pages 235-238, 2015.

[48] X. Man “A high performance Computing Cloud Computing Environment for machining simulations”, Procedia CIRP 8 57 – 62, 2013.

[49] Ismail Ari ≪Design and implementation of a cloud computing service for finite element analysis≫, Advances in Engineering Software 60-61 (2013) 122‑135.

[50] Dazhong Wu ≪Performance Evaluation of cloud-based high-performance computing for finite element analysis≫ Proceedings of the ASME 2015 International Design Engineering Technical Conferences, 2015.

[51] Bartosz Balis ≪Leveraging workflows and clouds for a multi-frontal solver for finite element meshes≫ Procedia Computer Science, Volume 51, 2015, Pages 944–953.
[52] Haiming Lin, Xiahu Liu and Kangyu Jia, "A study on linear Elastic Fem by Cloud Computing", ACM International Conference Proceeding Series, 2013.
[53] Jeremy Cohen ≪Nekkloud a software Environment for high order finite element analysis on clusters and clouds≫, Proceedings - IEEE International Conference on Cluster Computing, ICCC 2013.
[54] Zou Xin≪Structural ≪Finite Element Method based on Cloud Computing≫, Proceedings - 2012 International Conference on Computer Science and Electronics Engineering.
[55] Bai Xiaoyong ≪High performance computing for finite element in cloud≫, Proceedings -2011 International Conference on Future Computer Sciences and Application.
[56] Book: High Performance Computing and Networking, by Wolfgang Gentzsch, Uwe Harms.
[57] Willem Himpe, "High Performance Computing applied to Dynamic Traffic" Procedia Computer Science 151 (2019) 409–416, Transport and Mobility Leuven, Diestesteenweg 57, 3010 Leuven, Belgium.
[58] Michal Janczykowski, "Large-scale urban traffic simulation with Scala and high-performance computing system" Journal of Computational Science 35 (2019) 91–101, AGH University of Science and Technology, Department of Computer Science, Faculty of Computer Science, Electronics and Telecommunications, Al.Mickiewicza 30, 30-059 Krakow, Poland.
[59] S.Bastrakov, "High performance computing in biomedical applications" Procedia Computer Science 18 (2013) 10 – 19, Lobachevski State University of Nizhni Novgorod, 23 Prospekt Gagarina, 603950, a Nizhni Novgorod, Russia.
[60] Francois-Xavier Le Dimet, "High Performance Computing in the Geoscience" Mathematical and Physical Sciences Vol. 462.
[61] Jun Fang, "Direct numerical simulation of reactor two-phase flows enabled by high-performance computing" Nuclear Engineering and Design 330 (2018) 409–419, Argonne Leadership Computing Facility, Argonne National Laboratory, Lemont, IL 60439, United States.
[62] Marc Garbey, "High Performance Computing for CFD" Department of Computer Science, University of Houston, Houston, TX, USA.
[63] Wolfgang Gentzsch, "High performance computing in the cloud" Future Generation Computer Systems 29 (2013) 1407, University of Calabria, Electronics, Informatics and Systems, Cosenza, Italy.
[64] Victor Fernandez, "Cloud Computing" Universidad Técnica Federico Santa María.
[65] James Byrne, "A Review of Cloud Computing Simulation Platforms and Related Environments" Irish Centre for Cloud Computing and Commerce, Dublin City University, Glasnevin, Dublin 9, Ireland.
[66] Sobhanayak Srichandan, "Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm" Future Computing and Informatics Journal 3 (2018) 210-230, Department of Computer Science, IIIT Bhubaneswar, Odisha, India.
[67] Azzedine Boukerche, "Vehicular Cloud Computing: Architectures, Applications, and Mobility" School of Electrical Engineering and Computer Science University of Ottawa Canada.
[68] Antonio Gómez, "An OpenFOAM-based model for heat-exchanger design in the Cloud" Applied Thermal Engineering 139 (2018) 239–255, Nabladot, S.L., WTCZ Torre Oeste, Planta 11, Avda. María Zambrano 31, 50018 Zaragoza, Spain.
[69] Simon Taylor et al, "Enabling Cloud-based Computational Fluid Dynamics with a Platform-as-a Service Solution.

[70] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David Daniel, Richard Graham, and Timothy Woodall, "Open mpi: goals, concept, and design of a next generation mpi implementation", Recent Advances in Parallel Virtual Machine and Message Passing Interface, pp. 97-104, January 2004.

[71] S.L. Dixon and C.A. Hall, "Fluids Mechanics and thermodynamics of turbomachinery", six edition.

[72] Kundu, P.K.;COHEN, I.M.;DOWLING, D.R. Fluid mechanics. 5. ed. [S.l.]: Academic Press, 2013. 920 p. ISBN 9780123821003. Citado 2 veces en página 26 y 27.

[73] Cengel, Yunus A. and Cimbala, John M. (2014). Fluid Mechanics: Fundamentals and Applications, Third edition, McGraw-Hill.

[74] Endalew Getnet Tsega and V.K. Katiyar, "Finite Element Solution of the Two-dimensional Incompressible Navier-Stokes Equations Using MATLAB", Applications and Applied Mathematics: An International Journal (AAM), Department of Mathematics, Indian Institute of Technology Roorkee.

[75] W. F. Ames, "Nonlinear partial differential equations in engineering", Mathematics in Science & Engineering, vol. 18, 527 pages.

[76] Esteban Foronda, "Optimización Topológica aplicada al diseño de turbomáquinas considerando restricciones estructurales y sobre el fluido" Universidad Nacional de Medellín, Colombia.

[77] Sigmund,Ole. (2000). Topology optimization: a tool for the tailoring of structures and materials. Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences, 358, 211-227.

[78] R. T. Haftka and Z. Gürdal, Elements of Structural Optimization, 3rd rev. and expanded Ed., vol. 11. Springer, 1992.

[79] Boom, T. Van Den, & Schutter, B. De. (2007). Optimization in Systems and Control. Delft Center for Systems and Control.

[80] Hans A Eschenauer and Niels Olhoff. Topology optimization of continuum structures: a review. Applied Mechanics Reviews, 54 (4):331-390,2001.

[81] A.G.M. Michell, "LVIII. The limits of economy of material in frame-structures, "Philosophical Magazine Series 6, vol 8, no. 47, pp. 589-597,1904.

[82] Martin Philip Bendsøe and Noboru Kikuchi. Generating optimal topologies in structural design using a homogenization method. Computer methods in applied mechanics and engineeting, 71 (2):197-224,1988.

[83] GIN Rozvany. Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics. Structural and Multidisciplinary optimization, 21(2):90-108,2001.

[84] Campelo, F., Ram, J. A., & Igarashi, H. (2010), "A survey of topology optimization in electromagnetics: considerations and current trends".

[85] Joshua D Deaton and Ramana V Grandhi. A survey of structural and multidisciplinary continuum topology optimization: post 2000. Structural and Multidisciplinary Optimization, 49 (1):1-38,2014.

[86] G.I.N. Rozvany, M. Zhou and T.Birker, "Generalized shape optimization without homogenization", Structural Optimization Vol 4, pp 250-252.

[87] Diego Hayashi Alonso et al., "Topology Optimization applied to the design of 2D swirl flow devices", Structural Multidisciplinary Optimization, Department of Mechatronics and Mechanical Systems Engineering, Polythechnic School of the University of Sao Paulo, 2018.

[88] Z.Q.Wang, “Fluid selection and parametric optimization of organic Rankine cycle using low temperature waste heat”, Energy 40 (2012) pg:107-115, Institute of Mechanical Engineering, China.

[89] T.Lehnhauser, “A numerical approach for shape optimization of fluid flow domains”, Compu. Methods Appl. Mech. Engrg. 194 (2005) pg: 5221-5241, Department of numerical Methods in Mechanical Engineering, Germany.

[90] Gerborg-Hansen, A., Sigmund, O., & Haber, R. B. (2005). Topology optimization of channel flow problems. Structural and Multidisciplinary Optimization, 30 (3), 181-192.

[91] Rozvany, G. I. N., & Lewinski, “Topology optimization in structural and continuum mechanics”, in CISM International Centre for Mechanical Sciences (vol. 549).

[92] L.F.N.Sá , “Topology optimization Method applied to laminar flow machine rotor design”, Department of Mechatronics and Mechanical Systems Engineering of Escola Politecnica, University of Sao Paulo, 2016.

[93] W.H.Fraser, Flow recirculation in centrifugal pumps, Proceedings of the Tenth Turbomachinery Symposium (1981) 95-100.

[94] B.Klaus, K. Rainer, Analysis of secondary flows in centrifugal impellers, Internat. J.Rotat. Mach. 1 (2005) 45-52.

[95] W. Montealegre Rubio, “Projeto de ‘MEMS’ eletrotermomecânicos usando o método de otimização topológica,” Master’s thesis, Universidade de São Paulo (USP). Escola Politécnica, Brasil, 2005.

[96] Krister Svanberg, “MMA and GCMMA- two methods for nonlinear optimization”, Optimization and System Theory, Stockholm, Sweden.

[97] Eduardo Lenz Cardoso and Jun Sergio Ono Fonseca. Complexity control in the topology optimization of continuum structures. Journal of the brazilian society of mechanical sciences and engineering, 25(3):293–301, 2003.

[98] Francisco Javier Ramírez Gil. Diseño óptimo de micromecanismos tridimensionales con actuación electrotérmica utilizando optimización topológica y unidades de procesamiento gráfico (gpu). M.Sc. Thesis, Facultad de Minas, 2013.

[99] Benito Stradi-Granados, Cloud Computing for Engineers Applications, Springer, School of material Science, Institute of technology of Costa Rica.

[100] D.B.Kirk and W.W.Hwu, Programming Massively Parallel Processors: A Hands-on Approach,1st ed. Morgan Kaufmann, 2010.

[101] Robert Robey, Yuliana Zamora, Parallalel and High Performance Computing.

[102] Horowitz et al. and Rupp, Computational Scientist, https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/.

[103] B. Barney, “Introduction to Parallel Computing,” 16-Jul-2012. [Online]. Available: https://computing.llnl.gov/tutorials/parallel_comp/. [Accessed: 23-Apr-2013].

[104] Michael J. Flynn and Kevin W. Rudd, “Parallel architectures”, ACM Computing Surveys, in press.

[105] Md Lokman Hosain ^{a,b}, Rebei Bel Fdhila ^{a,b}, “Literature review of accelerated CFD Simulation Methods towards Online Application”, Energy Procedia 75 (2015) 3307 – 3314 (2015). ^{a} ABB AB, Corporate Research, SE-72178, V$\ddot{a}$sterás, Sweden. ^{b} M$\ddot{a}$lardalen University, School of Business, Society / Engineering, P.O. Box 883, SE’72123, V$\ddot{a}$ster$\ddot{a}$s, Sweden.

[106] R. A. Gingold and J.J. Monaghan, “Smoothed particle hydrodynamics:theory and application to non-spherical stars”, Royal Astronomical Society, vol. 181, pp. 375-389, February 1977.

[107] L. Greengard and V. Rokhlin, “A fast algorithm for particle simulations”, Journal of Computational Physics, vol. 73, pp. 325-348, February 1987.

[108] V.D. Kupradze and M.A. Aleksidze**,** "The method of functional equations for the approximate solution of certain boundary value problems", USSR Comput. Math. Phys., vol. 4, pp. 82-126, June 1963.

[109] Sudarshan Tiwari and Jorg Kuhnert, "Finite poinset method based on the projection method for simulations of the incompressible navier-stokes equations", Meshfree Methods for Partial Differential Equations, pp. 373-387, July 2011.

[110] Koshizuka and Oka, "Moving particle semi-implicit method" A Meshfree Particle Method for fluid Dynamics, 1st edition, pp. count: 306, June 2018.

[111] Arstid, P., "Reduction of process simulation models: a proper orthogonal approach", P.h.D. Thesis, Technische Universiteit Eindhoven, January 2004.

[112] Francis H. Harlow and J. Eddie Welch, "Numerical calculation of time dependent viscous incompressible flow of fluid with free surface", The Physics of Fluids, vol. 8, number 12, December 1965.

[113] Thomas A. Brenner et al., "A reduced-order model for heat transfer in multiphase flow and practical aspects of the proper orthogonal decomposition", Computers & Chemical Engineering, vol. 43, pp. 68-80, April 2012.

[114] O. Buneman, "Dissipation of currents in ionized media", Physical Review, vol. 115, number 3, February 1959.

[115] J.P. Christiansen, "Numerical simulation of hydrodynamics by method of point vortices", Journal of Computational Physics, vol. 13, pp. 363-379, October 1971.

[116] Guo Z. and Shu., "Lattice Boltzman method and its application in engineering", World Scientific Publishing Company, pp. count: 420, March 2013.

[117] D. Kandhai et al., "A comparison between Lattice-Bolztmann and finite-element simulations of fluid flow in static mixer reactors", International Journal of Modern Physics C., vol. 9, pp. 1123-1128, September 1998.

[118] Juan Rodríguez Pérez, "Programación Matlab en paralelo sobre Clúster Computacional: Evaluación de prestaciones", Tesis de grado, Escueal Técnica Superior de Ingeniería de Telecomunicación,Universidad Politécnica de Cartagena, Febrero 2010.

[119] T. Borrvall and J. Petersson, "Large-scale topology optimization in 3D using parallel computing," Computer Methods in Applied Mechanics and Engineering, vol. 190, no. 46-47, pp. 6201-6229, Sep.2001.

[120] R. Farber, CUDA Application Design and Development, 1st ed. Morgan Kaufmann, 2011.

[121] Hao Che [1], Min Nguyen [1], "Amdahl's law for multithreaded multicore processors", J. Parallel Distrib. Compu. 74 (2014) 3056-3069, [1] Department of Computer Science and Engineering, University of Texas at Arlington, TX 76019, USA.

[122] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd Revised edition. MIT Press, 2009.

[123] Niels Aage [1] et al, "Topology optimization of large-scale Stokes flow problems", Struct. Multidisciplinary Optimization (2008) 35:175-180. [1]Department of Mechanical Engineering Technical University of Denmark, DK-2800 Lyngby, Denmark.

[124] Kentaro Yaji et al, "Large-scale topology optimization incorporating local-in-time adjoint-based method for unsteady thermal-fluid problem.

[125] Chen C, Yaji K, Yamada T, Izuki K, Nishiwaki S (2017) Local-in-time adjoint-based topology optimization of unsteady fluid flows using the lattice Boltzman method. Mechanical Engineering Journal 4(3):17-00120.

[126] Joe Alexandersen [1] et al, "Large scale three-dimensional topology optimization of heat sinks cooled by natural convection", International Journal of Heat and Mass Transfer 100 (2016) 876-891. [1]Department of Mechanical Engineering Solid Mechanics,

Technical University of Denmark, Nils Koppels, Allé, Building 404, DK-2800 Kgs. Lyngby, Denmark.

[127] Vivien J. Challis [1], James K. Guest [2], "Level set topology optimization of fluids in Stokes flow", International Journal for numerical methods in engineering (2009); 79:1284-1308. [1] Department of Mathematics, The University of Queensland, Brisbane, QLD 4072, Australia. [2]Department of Civil Engineering, Johns Hopkins University, 3400 N. Charles St. Baltimore, MD 21218, USA.

[128] Kazuo Yonekura [1], Yoshihiro Kanno [2], "Topology optimization method for interior flow based on transient information of the lattice Boltzmann method with a level-set function", Japan J. Indust. Appl. Math. (2016). [1]IHI Corporation, Shin-Nakahara-Cho, Isogo-Ku, Yokohama 2358501, Japan. [2]Tokyo Institute of Technology, 4259 Nagatsuta-cho,Midori-ku, Yokohama 2268503, Japan.

[129] Hernan Villanueva [1], Kurt Maute [2], "CutFEM topology optimization of 3D laminar incompressible flow problems", Comput. Methods Appl. Mech. Engrg. [1]Department of Mechanical Engineering, University of Colorado Boulder, CO 427 UCB, USA. [2]Department of Aerospace Engineering, University of Colorado Boulder, Boulder, CO 427 UCB, USA.

[130] Sumer B. Dilgen [1] et al, "Density based topology optimization of turbulent flow heat transfer systems", Structural and Multidisciplinary Optimization (2017). [1]Department of Electrical Engineering, Technical University of Denmark, Lyngby, Denmark.

[131] Cetin B. Dilgen [1] et al, "Topology optimization of turbulent flows", Comput. Methods Appl. Mech. Engrg (2017). [1]Department of Electrical Engineering, Technical University of Denmark, Denmark.

[132] Joe Alexandersen [1] et al, "Design of passive coolers for light-emitting diode lamps using topology optimization", International Journal of Heat and Mass Transfer 122 (2018) 138-149. [1]Department of Mechanical Engineering, Solid Mechanics, Technical University of Denmark, Nils Koppels Allé, Building 404, DK-2800, Denmark.

[133] Michat Tomasz Jakóbczyk, "Practical Oracle Cloud Infrastructure", pag. 16.- Bare Metal Machine v.s. Virtual Machine.

[134] Wu, C., & Buyya, R., "Cloud Infrastructure Servers. Cloud Data Centers and Cost Modeling, 371-424, (2015).

[135] M. G. Larson and F. Bengzon, The Finite Element Method: Theory, Implementation, and Practice. Springer, 2010.

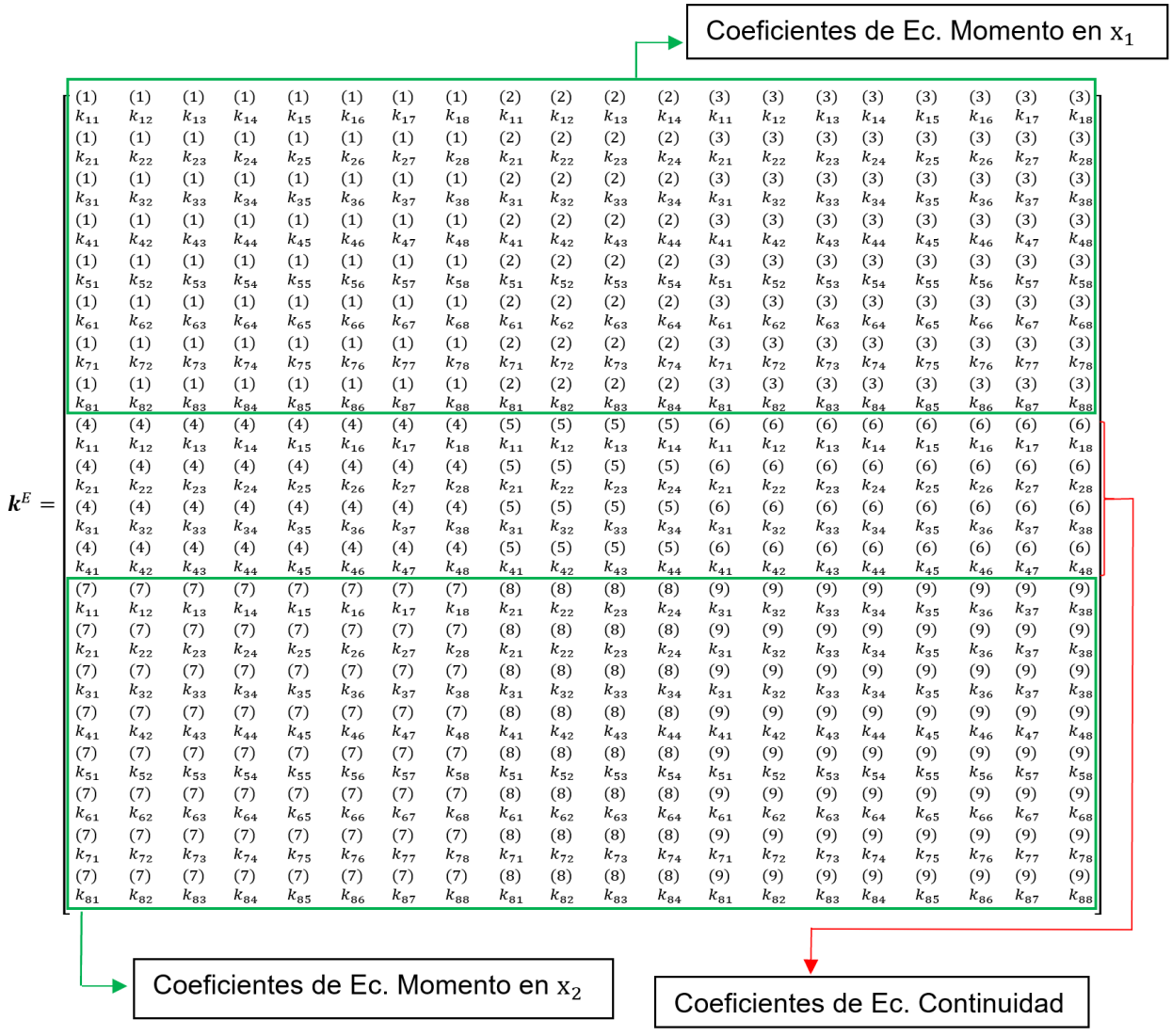
[136] Y. W. Kwon and H. Bang, The Finite Element Method Using Matlab. CRC Press, 2000.

[137] D.V.Hutton, Fundamentals of Finite Element Analysis,1st ed.McGraw-Hill,2003.

8 ANEXO A: MODELAMIENTO DE UN ELEMENTO

8.1 COEFICIENTES DE MATRIZ $\boldsymbol{k}^E$ Y VECTOR $\boldsymbol{b}^E$

La ecuación lineal (Ecc. 2.27) contiene las ecuaciones de gobierno de momentum de Navier-Stokes y de continuidad, por lo tanto, se deben calcular los coeficientes de la matriz $\boldsymbol{k}^E$ y las constantes del vector $\boldsymbol{b}^E$. Para expresar los coeficientes de la matriz $\boldsymbol{k}^E$ por elemento se tiene una matriz cuadrada de dimensiones $20x20$. Esta matriz tiene esta estructura debido a que se tiene 1 ecuación por cada grado de libertad; y, debido a que se tiene 8 grados de libertad para la velocidad en x_1, 4 grados de libertad para la presión y 8 grados de libertad para la velocidad en x_2, se obtienen 20 ecuaciones por elemento, resultando así 20 coeficientes por cada ecuación. A continuación, se detallan estos coeficientes de la matriz $\boldsymbol{k}^E$:



La obtención de estos coeficientes provienen de la (Ecc. 2.24), (Ecc. 2.25) y (Ecc. 2.26). Para los siguientes coeficientes de la matriz $\boldsymbol{k}^E$ que se detallan a continuación, j está relacionado con el grado de libertad del elemento mientras que i esta relacionado con la función de prueba que resulta de la metodología del método de Galerkin de residuos ponderados.

Los coeficientes que se obtienen de la ecuación de momento (Ecc. 2.24) en x_1 son los siguientes:

Coeficientes para la componente u_{r1}:

$$\overset{(1)}{k_{ij}} = \iint_E \left[N_i \rho N_j u_{r1_k} \frac{\partial N_j}{\partial \mathrm{x}_1} + N_i \rho N_j u_{r2_k} \frac{\partial N_j}{\partial \mathrm{x}_2} + \mu \left(\frac{\partial N_i}{\partial \mathrm{x}_1} \frac{\partial N_j}{\partial \mathrm{x}_1} + \frac{\partial N_i}{\partial \mathrm{x}_2} \frac{\partial N_j}{\partial \mathrm{x}_2} \right) + N_i \alpha(\gamma) \right] dA \qquad \text{(Ecc. 8.1)}$$

$i, j = 1,2, \dots 8$

Coeficientes para p:

$$\overset{(2)}{k_{ij}} = \iint_E \left[N_i \frac{\partial M_j}{\partial \mathrm{x}_1} \right] dA \qquad \text{(Ecc. 8.2)}$$

$i = 1,2, \dots 8 \qquad j = 1,2 \dots 4$

Coeficientes para la componente u_{r2}:

$$\overset{(3)}{k_{ij}} = \iint_E \left[N_i 2 \rho \Omega N_j \right] dA \qquad \text{(Ecc. 8.3)}$$

$i, j = 1,2, \dots 8$

Los coeficientes que se obtienen de la ecuación de continuidad (Ecc. 2.25) son los siguientes:

Coeficientes para la componente u_{r1}:

$$\overset{(4)}{k_{ij}} = \iint_E \left[M_l \frac{\partial N_j}{\partial \mathrm{x}_1} \right] dA \qquad \text{(Ecc. 8.4)}$$

$l = 1,2, \dots 4 \qquad j = 1,2 \dots 8$

Coeficientes para la componente u_{r2}:

$$\overset{(6)}{k_{ij}} = \iint_E \left[M_l \frac{\partial N_j}{\partial \mathrm{x}_2} \right] dA \qquad \text{(Ecc. 8.5)}$$

$l = 1,2, \dots 4 \qquad j = 1,2 \dots 8$

Los coeficientes que se obtienen de la ecuación de momento (Ecc. 2.26) en x_2 son los siguientes:

Coeficientes para la componente u_{r1}:

$$\overset{(7)}{k_{ij}} = \iint_E \left[N_i 2\rho\Omega N_j\right] dA \qquad (\text{Ecc. } 8.6)$$

$i, j = 1,2, \dots 8$

Coeficientes para p:

$$\overset{(8)}{k_{ij}} = \iint_E \left[N_i \frac{\partial M_j}{\partial \mathrm{x}_2}\right] dA \qquad (\text{Ecc. } 8.7)$$

$i = 1,2, \dots 8 \qquad j = 1,2 \dots 4$

Coeficientes para la componente u_{r2}:

$$\overset{(9)}{k_{ij}} = \overset{(1)}{k_{ij}} \qquad (\text{Ecc. } 8.8)$$

De la ecuación (Ecc. 2.27), $\boldsymbol{z}^E$ es un vector de dimensiones $(20x1)$ que contiene los valores nodales de velocidad y presión:

$$\boldsymbol{z}^E = \begin{Bmatrix} u_{r1_j} \\ p_l \\ u_{r2_j} \end{Bmatrix}$$

$$j = 1,2 \dots 8 \quad l = 1,2 \dots 4$$

De la (Ecc. 2.27), $\boldsymbol{b}^E$ contiene los términos constantes de las (Ecc. 2.24), (Ecc. 2.25) y (Ecc. 2.26):

Los coeficientes constantes que se obtienen de la ecuación de momento (Ecc. 2.24) en x_1 son los siguientes:

$$\overset{(10)}{b_i} = \iint_E \left[N_i \rho\Omega^2 r_{\mathrm{x}_1}\right] dA \qquad (\text{Ecc. } 8.9)$$

$$i = 1,2, \dots 8$$

Los coeficientes constantes que se obtienen de la ecuación de momento (Ecc. 2.26) en x_2 son los siguientes:

$$b_i^{(12)} = \iint_E \left[N_i \rho g_v + N_i \rho \Omega^2 r_{\mathrm{x}_2}\right] dA \qquad (\mathrm{Ecc.}\, 8.10)$$

$$i = 1,2, \ldots 8$$

Entonces, de la (Ecc. 2.27) se tiene que el vector $\boldsymbol{b}^E$ es de dimensiones de $(20x1)$ y es de la forma:

$$\boldsymbol{b}^E = \begin{Bmatrix} b_1^{(10)} \\ b_2^{(10)} \\ b_3^{(10)} \\ b_4^{(10)} \\ b_5^{(10)} \\ b_6^{(10)} \\ b_7^{(10)} \\ b_8^{(10)} \\ 0 \\ 0 \\ 0 \\ 0 \\ b_1^{(12)} \\ b_2^{(12)} \\ b_3^{(12)} \\ b_4^{(12)} \\ b_5^{(12)} \\ b_6^{(12)} \\ b_7^{(12)} \\ b_8^{(12)} \end{Bmatrix}$$

Los ceros en el vector $\boldsymbol{b}^E$ se debe a que la ecuación de continuidad (Ecc. 2.25) no posee términos constantes.

8.2 EVALUACIÓN DE LAS INTEGRALES DEL PROBLEMA HIDRODINÁMICO

Para el problema de elementos finitos, aparecen integrales de área para el cálculo de la matriz $\boldsymbol{k}^E$ y el vector de constantes $\boldsymbol{b}^E$. Dichas integrales dependen de las funciones de interpolación N_i y M_l y de sus derivadas. Estas funciones dependen exclusivamente del tipo de elemento finito seleccionado. En el presente estudio se selecciona un elemento de 8 nodos para la variable de la velocidad y un elemento rectangular de 4 nodos para la variable de la presión, tal como se detalla en la Figura 2.5.

8.2.1 ELEMENTOS RECTANGULARES

8.2.1.1 FUNCIÓN DE INTERPOLACIÓN PARA EL ELEMENTO RECTANGULAR DE 4 NODOS.

Al elemento rectangular de 4 nodos también se lo conoce como Q_4. La (Ecc. 8.11) representa la función de interpolación polinómica de menor grado para este elemento:

$$\phi^E(\mathrm{x}_1,\mathrm{x}_2)=a_1+a_2\mathrm{x}_1+a_3\mathrm{x}_2+a_4\mathrm{x}_1\mathrm{x}_2 \qquad \text{(Ecc. 8.11)}$$

Matricialmente, la (Ecc. 8.11) se puede escribir como $\phi^E(\mathrm{x}_1,\mathrm{x}_2)=[\mathrm{X}]\{a\}$ donde $[\mathrm{X}]$ es una matriz que contiene las variables x_1, x_2 y $\{a\}$ es el vector de coeficientes $a_1 \ldots a_4$ desconocidos. Como se requiere que la función de interpolación sea igual al valor nodal de la variable interpolada ϕ, entonces se cumple lo siguiente:

$$\phi^E\left(\mathrm{x}_{1_i},\mathrm{x}_{2_i}\right) \equiv \phi^E{}_i \qquad \text{con } i = 1,2,\ldots 4 \qquad \text{(Ecc. 8.12)}$$

Donde x_{1_i}, x_{2_i} $(i = 1,2,\ldots 4)$ representan las coordenadas globales de los nodos del elemento. Ahora se requiere hallar los valores de a_i $(i = 1 \ldots 4)$ de la (Ecc. 8.11), los cuales se obtiene usando la (Ecc. 8.12):

$$\begin{aligned}
\phi^E\left(\mathrm{x}_{1_1},\mathrm{x}_{2_1}\right) &= \phi^E{}_1 = a_1+a_2\mathrm{x}_{1_1}+a_3\mathrm{x}_{2_1}+a_4\mathrm{x}_{1_1}\mathrm{x}_{2_1} \\
\phi^E\left(\mathrm{x}_{1_2},\mathrm{x}_{2_2}\right) &= \phi^E{}_2 = a_1+a_2\mathrm{x}_{1_2}+a_3\mathrm{x}_{2_2}+a_4\mathrm{x}_{1_2}\mathrm{x}_{2_2} \\
\phi^E\left(\mathrm{x}_{1_3},\mathrm{x}_{2_3}\right) &= \phi^E{}_3 = a_1+a_2\mathrm{x}_{1_3}+a_3\mathrm{x}_{2_3}+a_4\mathrm{x}_{1_3}\mathrm{x}_{2_3} \\
\phi^E\left(\mathrm{x}_{1_4},\mathrm{x}_{2_4}\right) &= \phi^E{}_4 = a_1+a_2\mathrm{x}_{1_4}+a_3\mathrm{x}_{2_4}+a_4\mathrm{x}_{1_4}\mathrm{x}_{2_4}
\end{aligned} \qquad \text{(Ecc. 8.13)}$$

Dado que el vector $\{\phi^E\}$ y la matriz $[\overline{\mathrm{X}}]$ son conocidos, de la (Ecc. 8.13) se calcula el vector de coeficientes desconocidos $\{a\}$ como:

$$\begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{bmatrix} 1 & \mathrm{x}_{1_1} & \mathrm{x}_{2_1} & \mathrm{x}_{1_1} & \mathrm{x}_{2_1} \\ 1 & \mathrm{x}_{1_2} & \mathrm{x}_{2_2} & \mathrm{x}_{1_2} & \mathrm{x}_{2_2} \\ 1 & \mathrm{x}_{1_3} & \mathrm{x}_{2_3} & \mathrm{x}_{1_3} & \mathrm{x}_{2_3} \\ 1 & \mathrm{x}_{1_4} & \mathrm{x}_{2_4} & \mathrm{x}_{1_4} & \mathrm{x}_{2_4} \\ & & & & \end{bmatrix}^{-1} \begin{Bmatrix} \phi^E{}_1 \\ \phi^E{}_2 \\ \phi^E{}_3 \\ \phi^E{}_4 \end{Bmatrix} \qquad \text{(Ecc. 8.14)}$$

El vector $\{a\}$ es un vector que vendría a representar las funciones de forma del elemento rectangular de cuatro nodos. Haciendo análogo este vector con las funciones de forma para la presión se puede decir que $\{a\} = \{M_l\}$. Entonces reemplazando los valores de $\{a\}$ en la (Ecc. 8.12) se obtiene que:

$$\phi^E(\mathrm{x}_1, \mathrm{x}_2) = a_1(\mathrm{x}_1, \mathrm{x}_2)\Phi_1 + a_2(\mathrm{x}_1, \mathrm{x}_2)\Phi_2 + a_3(\mathrm{x}_1, \mathrm{x}_2)\Phi_3 + a_4(\mathrm{x}_1, \mathrm{x}_2)\Phi_4 \qquad \text{(Ecc. 8.15)}$$

Dejando en términos de la presión se obtiene que:

$$p(\mathrm{x}_1, \mathrm{x}_2) = M_1(\mathrm{x}_1, \mathrm{x}_2)p_1 + M_2(\mathrm{x}_1, \mathrm{x}_2)p_2 + M_3(\mathrm{x}_1, \mathrm{x}_2)p_3 + M_4(\mathrm{x}_1, \mathrm{x}_2)p_4 \qquad \text{(Ecc. 8.16)}$$

8.2.1.2 TÉCNICA DEL ELEMENTO DE REFERENCIA

La Figura 8.1.a presenta un elemento rectangular, el cual se conoce como elemento de referencia, esto se debe por su tamaño y sus coordenadas. Esta técnica del elemento de referencia es de importancia cuando se aplica optimización topológica, ya que esta técnica aplicada en conjunto se encarga de generar topologías optimas a partir de formas básicas como cuadrados o rectángulos para el caso bidimensional; además, si la malla consiste de formas simples (estructurada), las funciones de formas locales de cada elemento se pueden calcular una sola vez, puesto que esta será igual para todos los elementos [135].

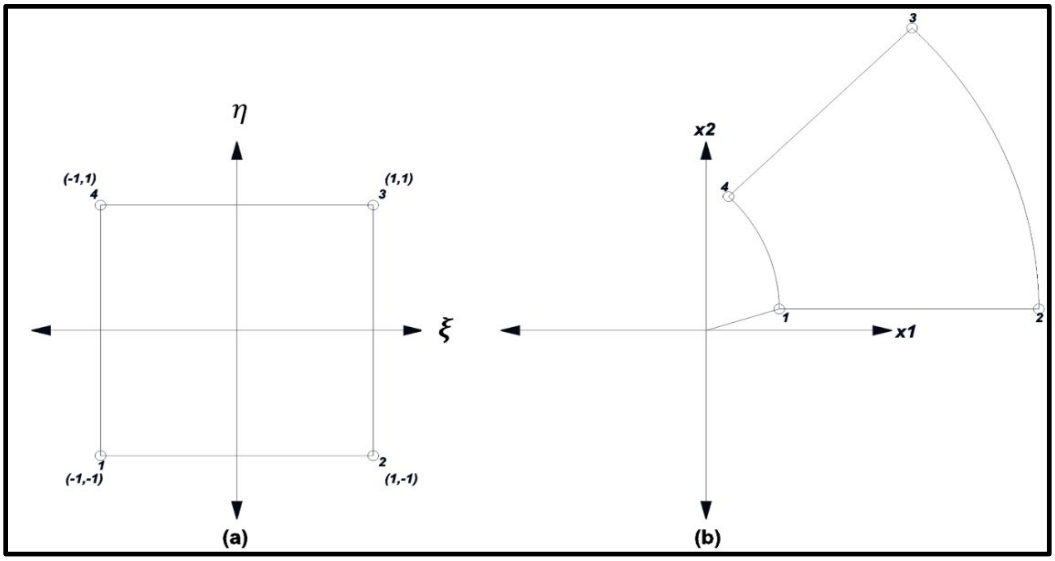


Figura 8.1.- Elemento rectangular de 4 nodos en sus coordenadas **a)** locales **b)** globales.

Ahora con el fin de hallar las ecuaciones de forma para el elemento rectangular de referencia, considere el dominio $\overline{\mathrm{D}} = \{(\xi, \eta): -1 \leq \xi, \eta \leq 1\}$ con vértices como se ilustra en la Figura 8.1.a. Sobre el elemento de referencia están definidas cuatro funciones de forma M_1, M_2, M_3, M_4. Estas ecuaciones de forma cumplen la propiedad de la (Ecc. 8.17).

$$M_l(\xi_i, \eta_j) = \delta_{il} = \begin{cases} 1 \; si \; i = l \\ 0 \; si \; i \neq l \end{cases} \qquad \text{(Ecc. 8.17)}$$

Entonces, estas funciones de forma para la presión (elemento rectangular de 4 nodos) puede ser escrita en su forma generalizada en coordenadas locales de la siguiente manera [74]:

$$M_l = a_0 + a_1\xi + a_2\eta + a_3\xi\eta \qquad \text{(Ecc. 8.18)}$$

Los coeficientes de la (Ecc. 8.18) se pueden hallar usando la propiedad de la (Ecc. 8.17). Usando la propiedad para la primera función de forma para la presión se tiene que:

$$M_1(\xi = -1, \eta = -1) = 1 = a_0 + a_1(-1) + a_2(-1) + a_3(-1)(-1)$$

$$M_1(\xi = 1, \eta = -1) = 0 = a_0 + a_1(1) + a_2(-1) + a_3(1)(-1)$$

$$M_1(\xi = 1, \eta = 1) = 0 = a_0 + a_1(1) + a_2(1) + a_3(1)(1)$$

$$M_1(\xi = -1, \eta = 1) = 0 = a_0 + a_1(-1) + a_2(1) + a_3(-1)(1)$$

Solucionando los coeficientes a_0, a_1, a_2 y a_3 y reemplazando en la función de forma M_1 se obtiene que:

$$M_1 = \frac{1}{4} - \frac{1}{4}\xi - \frac{1}{4}\eta + \frac{1}{4}\xi\eta$$

O (Ecc. 8.19)

$$M_1 = \frac{1}{4}(1 - \xi - \eta + \xi\eta)$$

Es así como se obtienen los coeficientes de la función de forma M_1 (Ecc. 2.14). De forma similar se obtienen las funciones de forma para M_2, M_3 y M_4.

8.2.1.3 FORMULACIÓN ISOPARAMÉTRICA DEL ELEMENTO FINITO

Una vez obtenido los coeficientes de la matriz $\boldsymbol{k}^E$ y el vector $\boldsymbol{b}^E$ de la sección anterior, es necesario ahora hallar las derivadas parciales de las funciones de forma con respecto a las coordenadas globales x_1 y x_2, y el cálculo de las integrales de área en el dominio de diseño por elemento. El primer problema se resuelve con la formulación isoparamétrica y el segundo con integración numérica [136].

Las funciones de forma del elemento de referencia están en términos de las coordenadas locales ξ y η mientras que las derivadas parciales en los coeficientes de la matriz $\boldsymbol{k}$ están en coordenadas globales x_1 y x_2. Para ir de un sistema de coordenadas a otro, se usa una transformación (mapeo) mediante la (Ecc. 8.20):

$$\mathrm{x}_1 = M_1(\xi,\eta)\mathrm{x}_{1_1} + M_2(\xi,\eta)\mathrm{x}_{1_2} + M_3(\xi,\eta)\mathrm{x}_{1_3} + M_4(\xi,\eta)\mathrm{x}_{1_4}$$

(Ecc. 8.20)

$$\mathrm{x}_2 = M_1(\xi,\eta)\mathrm{x}_{2_1} + M_2(\xi,\eta)\mathrm{x}_{2_2} + M_3(\xi,\eta)\mathrm{x}_{2_3} + M_4(\xi,\eta)\mathrm{x}_{2_4}$$

donde (x_{1_1}, x_{2_1}),..... $(x1_4, x_{2_4})$ son las coordenadas globales de los nodos del elemento y (x_1, x_2) son las coordenadas globales de un punto en el elemento.

Ahora con el objetivo de calcular las derivadas en términos de las variables ξ y η de las funciones dependientes de las coordenadas globales x_1 y x_2, se hace uso de la regla de la cadena presentada en la (Ecc. 8.21):

$$\frac{\partial x_1}{\partial \xi} = \frac{\partial M_1}{\partial \xi} x_{1_1} + \frac{\partial M_2}{\partial \xi} x_{1_2} + \frac{\partial M_3}{\partial \xi} x1_3 + \frac{\partial M_4}{\partial \xi} x1_4$$

$$\frac{\partial x_2}{\partial \xi} = \frac{\partial M_1}{\partial \xi} x_{2_1} + \frac{\partial M_2}{\partial \xi} x_{2_2} + \frac{\partial M_3}{\partial \xi} x_{2_3} + \frac{\partial M_4}{\partial \xi} x_{2_4} \quad \text{(Ecc. 8.21)}$$

$$\frac{\partial x_1}{\partial \eta} = \frac{\partial M_1}{\partial \eta} x_{1_1} + \frac{\partial M_2}{\partial \eta} x_{1_2} + \frac{\partial M_3}{\partial \eta} x_{1_3} + \frac{\partial M_4}{\partial \eta} x_{1_4}$$

$$\frac{\partial x_2}{\partial \eta} = \frac{\partial M_1}{\partial \eta} x_{2_1} + \frac{\partial M_2}{\partial \eta} x_{2_2} + \frac{\partial M_3}{\partial \eta} x_{2_3} + \frac{\partial M_4}{\partial \eta} x_{2_4}$$

Donde $M(\xi, \eta)$ es una función de forma en coordenadas locales. Si "x_1" y "x_2" son coordenadas globales, entonces:

$$\frac{\partial M}{\partial \xi} = \frac{\partial M}{\partial x_1}\frac{\partial x_1}{\partial \xi} + \frac{\partial M}{\partial x_2}\frac{\partial x_2}{\partial \xi}$$

(Ecc. 8.22)

$$\frac{\partial M}{\partial \eta} = \frac{\partial M}{\partial x_1}\frac{\partial x_1}{\partial \eta} + \frac{\partial M}{\partial x_2}\frac{\partial x_2}{\partial \eta}$$

Escribiendo la (Ecc. 8.22) en su forma matricial:

$$\underbrace{\begin{bmatrix} \frac{\partial M}{\partial \xi} \\ \frac{\partial M}{\partial \eta} \end{bmatrix}}_{\text{Derivadas locales}} = \underbrace{\begin{bmatrix} \frac{\partial x_1}{\partial \xi} & \frac{\partial x_2}{\partial \xi} \\ \frac{\partial x_1}{\partial \eta} & \frac{\partial x_2}{\partial \eta} \end{bmatrix}}_{\text{Jacobiano}} \underbrace{\begin{bmatrix} \frac{\partial M}{\partial x_1} \\ \frac{\partial M}{\partial x_2} \end{bmatrix}}_{\text{Derivadas globales}} \quad \text{(Ecc.8.23)}$$

El jacobiano es una matriz de transformación del sistema de coordenadas globales al sistema de coordenadas locales, entonces se tiene:

$$\begin{bmatrix} \frac{\partial M}{\partial x_1} \\ \frac{\partial M}{\partial x_2} \end{bmatrix} = \boldsymbol{J_{acob.}}^{-1} \begin{bmatrix} \frac{\partial M}{\partial \xi} \\ \frac{\partial M}{\partial \eta} \end{bmatrix} \quad \text{(Ecc. 8.24)}$$

Ahora escribiendo la ecuación (Ecc. 8.21) en forma matricial se obtiene:

$$\boldsymbol{J_{acob.}} = \begin{bmatrix} \frac{\partial x_1}{\partial \xi} & \frac{\partial x_2}{\partial \xi} \\ \frac{\partial x_1}{\eta} & \frac{\partial x_2}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial M_1}{\partial \xi} & \frac{\partial M_2}{\partial \xi} & \frac{\partial M_3}{\partial \xi} & \frac{\partial M_4}{\partial \xi} \\ \frac{\partial M_1}{\partial \eta} & \frac{\partial M_2}{\partial \eta} & \frac{\partial M_3}{\partial \eta} & \frac{\partial M_4}{\partial \eta} \end{bmatrix} \begin{bmatrix} x_{1\,1} & x_{2\,1} \\ x_{1\,2} & x_{2\,2} \\ x_{1\,3} & x_{2\,3} \\ x_{1\,4} & x_{2\,4} \end{bmatrix} \qquad \text{(Ecc. 8.25)}$$

Entonces el Jacobiano $\boldsymbol{J_{acob.}}$ es de la forma:

$$\boldsymbol{J_{acob.}} = \begin{bmatrix} \sum_{i=1}^{4} \frac{\partial M_i}{\partial \xi} x_{1\,i} & \sum_{i=1}^{4} \frac{\partial M_i}{\partial \xi} x_{2\,i} \\ \sum_{1}^{4} \frac{\partial M_i}{\partial \eta} x_{1\,i} & \sum_{1}^{4} \frac{\partial M_i}{\partial \eta} x_{2\,i} \end{bmatrix} \qquad \text{(Ecc. 8.26)}$$

8.2.1.4 INTEGRACIÓN NUMÉRICA

Para calcular los coeficientes de la matriz $\boldsymbol{k^E}$ y los del vector $\boldsymbol{b^E}$**,** es necesario hacer una integración doble sobre el dominio del elemento; sin embargo, dicha integración realizada por métodos analíticos puede ser muy complicada por el hecho de que en el contexto de los elementos finitos se requiere un gran número de integraciones donde los métodos analíticos pueden ser ineficientes, por lo que una solución a esto son las integraciones numéricas. La cuadratura Gaussiana es la mayormente empleada y es la que se usa en este proyecto de investigación. Esta integral puede ser exacta dependiendo del número de puntos de integración y el grado del polinomio integrado [137].

Esta integración es exacta si se usan "xp" puntos de integración y "w" pesos de integración. Esta integral es exacta para un polinomio de grado 2*xp-1 [136].Para la selección de los puntos de integración "xp" y "w" se tiene que tomar en cuenta el coeficiente de mayor grado de polinomio de la matriz $\boldsymbol{k^E}$ y del vector $\boldsymbol{b^E}$. Vemos que el coeficiente de mayor grado de polinomio es el $k_{ij}^{(1)}$ (Ecc. 8.1), con un grado de polinomio 5 para cada una de las variables ξ y η. Por lo tanto, el número de puntos de integración debe ser 2*xp-1=5 → xp=3, lo que significa que se deben usar 3 puntos de integración y pesos de integración por cada variable (ξ, η) al usar la integración numérica para obtener una integración exacta, ver Tabla 7.1 .

Tabla 7.1 Puntos y pesos de integración para los elementos rectangulares de 8 y 4 nodos.

Coordenada	**Localización**	**Pesos** $(\boldsymbol{w})$
ξ	$0, \pm\sqrt{3/5}$	5/9,8/9,5/9
η	$0, \pm\sqrt{3/5}$	5/9,8/9,5/9

Entonces usando el cálculo para la transformación de coordenadas, una integral típica para un elemento rectangular bidimensional se puede evaluar como:

$$\iint_E f(\mathrm{x},\mathrm{y})d\mathrm{x}_1 d\mathrm{x}_2 = \iint_E f\big(\mathrm{x}_1(\xi,\eta),\mathrm{x}_2(\xi,\eta)\big)|\det\,(J_{acob.})|\,d\xi\,d\eta$$

$$= \int_{-1}^{1}\int_{-1}^{1} f\big(\mathrm{x}_1(\xi,\eta),\mathrm{x}_2(\xi,\eta)\big)\,|\det\,(J_{acob.})|\,d\xi\,d\eta \quad \text{(Ecc. 8.27)}$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{n} w_i w_j \bar{f}(\xi_i,\eta_j)$$

donde,

$$\bar{f}(\xi,\eta) = f\big(\mathrm{x}_1(\xi,\eta),\mathrm{x}_2(\xi,\eta)\big)|\det\,(J_{acob.})| \quad \text{(Ecc. 8.28)}$$

Donde $J_{acob.}$ es la matriz Jacobiana de transformación, ξ_i y η_j son las abscisas y ordenadas de la cuadratura Gaussiana, y w_i, w_j son los correspondientes pesos. Con el fin de hallar valores estándares para los puntos de integración, es necesario que el dominio de integración este normalizado y que las variables usadas solo varíen entre -1 y 1, ver Figura 8.2 [136].

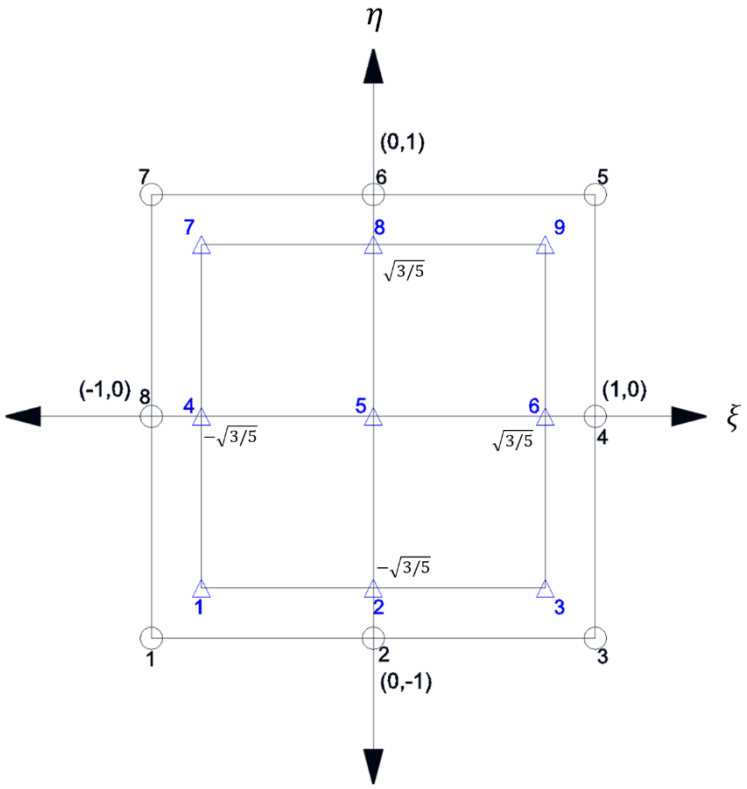


Figura 8.2.- Localización de los nodos y puntos de integración para el elemento rectangular de 8 nodos Q_8.

8.2.1.5 CÁLCULO DE LAS MATRICES Y VECTORES DEL PROBLEMA DE FLUJO

La matriz $\boldsymbol{k}^E$ por elemento, se expresa en términos de las coordenadas locales ξ y η usando la formulación isoparamétrica anteriormente desarrollada y considerando que $dx_1 dx_2 = |J_{acob.}| d\xi\, d\eta$, donde $|J_{acob.}|$ es el determinante de la matriz Jacobiana:

$$\iint_E f(x_1, x_2) dx_1 dx_2 = \sum_{i=1}^{m} \sum_{j=1}^{n} w_i w_j \bar{f}(\xi_i, \eta_j) \qquad (\text{Ecc. } 8.29)$$

A manera de ejemplo para la integración numérica aplicando la cuadratura Gaussiana se toma el coeficiente $\overset{(2)}{k_{ij}}$ (Ecc. 8.2)

$$f = \overset{(2)}{k_{ij}} = N_i \frac{\partial M_j}{\partial x_1}$$

Entonces dejando expresado la función f en términos de la $(\text{Ecc. } 8.29)$ se tiene:

$$\iint_E f(x_1, x_2) dx_1 dx_2 = \sum_{i=1}^{3} \sum_{j=1}^{3} w_i w_j \bar{f}(\xi_i, \eta_j)$$

$$\bar{f} = N_i \frac{\partial M_j}{\partial x_1} |det\ (J_{acob.})|$$

$$\iint_E f(x_1, x_2) dx_1 dx_2 = \sum_{i=1}^{3} \sum_{j=1}^{3} w_i w_j N_i \frac{\partial M_j}{\partial x_1} |\det\ (J_{acob.})| \qquad (\text{Ecc. } 8.30)$$

En el Cuadro 8.1 se presenta el pseudocódigo para el cálculo de los coeficientes $\overset{(2)}{k_{ij}}$ de la matriz $\boldsymbol{k}^E$.

Cuadro 8.1.- Cálculo de coeficiente $\overset{(2)}{k_{ij}}$ de la matriz $\boldsymbol{k}^E$.

```
Cálculo de las funciones de forma
Cálculo de las derivadas de las funciones de forma con respecto a ξ y a η
for P=1: puntos de integración
Calcular el jacobiano J_acob.
Calcular el determinante del Jacobiano y su inversa
Calcular las derivadas de las funciones de forma con respecto a x1 y a x2
for i=1:8
for j=1:4
k2(i,j)=k2(i,j)+w(P)*N(i,P)*gdm1(j)*detJ_acob.
fin
fin
fin
```

Finalmente, la matriz global $\boldsymbol{k}$ se construye a partir del ensamble de todas las matrices $\boldsymbol{k}^E$. A manera de ejemplo, en la Figura 2.6. se presentan ocho ecuaciones recursivas para el ensamble de los 64 términos asociados a las 8 ecuaciones de momentum e incógnitas asociadas a la dirección x_1.