Revista de Ciencias Tecnológicas (RECIT). Universidad Autónoma de Baja California ISSN 2594-1925
Volumen 9 (3): e460. Julio-Septiembre, 2026. https://doi.org/10.37636/recit.v9n3e460

Artículo de investigación

Implementación y verificación de un solucionador CFD bidimensional en Python para problemas difusivos y flujos forzados laminares

Implementation and verification of a two-dimensional CFD solver in Python for diffusive problems and laminar forced flows

Edgard Elohim Canche-Cauich[1], Felipe Noh-Pat[1*], Miguel Ángel Gijón-Rivera[2], Manuel Jesús Rodríguez-Pérez[1], Mauricio Iván Huchin-Miss[1]

[1]Universidad Autónoma de Campeche, Facultad de Ingeniería, Campus V., Av. Ing. Humberto Lanz Cárdenas, Col. Ex Hacienda Kalá, San Francisco de Campeche 24085, México
[2] Tecnológico Tecnológico de Monterrey, Escuela de Ingeniería y Ciencias, Vía Atlixcáyotl 5718, Reserva Territorial Atlixcáyotl, Puebla, Pue. 72453, México

Autor de correspondencia: Felipe Noh-Pat, Universidad Autónoma de Campeche, Facultad de Ingeniería, Campus V., Av. Ing. Humberto Lanz Cárdenas, Col. Ex Hacienda Kalá, San Francisco de Campeche 24085, México. Correo electrónico: felipnoh@uacam.mx. ORCID: 0000-0003-1981-8323.

Recibido: 13 de abril del 2026 **Aceptado:** 13 de Junio del 2026 **Publicado:** 3 de julio del 2026

Resumen. - *La Dinámica de Fluidos Computacional (CFD) permite predecir el comportamiento de fenómenos físicos en diversos sistemas de índole académico e industrial. En México, su aplicación ha tomado relevancia en la investigación de edificaciones energéticamente eficientes ante las consecuencias del calentamiento global. El objetivo de este trabajo es desarrollar y verificar un solucionador CFD bidimensional implementado en el lenguaje de programación Python, basado en el método de volumen finito, capaz de resolver ecuaciones difusivas y las ecuaciones de Navier-Stokes en régimen laminar, así como evaluar su desempeño computacional. La solución numérica de las ecuaciones gobernantes discretizadas se realiza mediante un método de direcciones alternantes (ADI) basado en Gauss-Seidel en línea (LGS-ADI). El primer caso de estudio corresponde a la solución de la ecuación de conducción de calor en estado estacionario en un bloque sólido. El segundo caso consiste en la simulación de un flujo laminar en una cavidad cuadrada llena de aire, impulsado por una velocidad impuesta en la tapa superior, con un número de Reynolds de 1000. En este caso se implementa el algoritmo SIMPLE para el acoplamiento presión-velocidad y se emplean mallas desplazadas para el campo de velocidades. Las simulaciones se realizan sobre una malla uniforme de 150x150 celdas, utilizando inicialmente un esquema upwind de primer orden y posteriormente un esquema híbrido para reducir la difusión numérica. Los resultados reproducen las estructuras características del flujo, como la formación de dos vórtices en las esquinas inferiores de la cavidad, y muestran concordancia con la solución de referencia. Los errores relativos porcentuales obtenidos en los casos de verificación son menores a 4.70%. En comparación con un solucionador basado en el método clásico de Gauss-Seidel, el algoritmo ADI con solución mediante el método de Thomas reduce el tiempo de procesamiento en 61.92% para el problema difusivo y en un máximo de 15.04% para el problema de flujo forzado, diferencia atribuible a la naturaleza no lineal de este último. Este trabajo contribuye al desarrollo de código propio para el modelado numérico mediante CFD.*

Palabras clave: Dinámica de fluidos computacional; Método de volumen finito; Algoritmos SIMPLE-ADI.

Abstract.- *Computational Fluid Dynamics (CFD) enables the prediction of the behavior of physical phenomena in a wide range of systems. In Mexico, its application has gained relevance in the study of energy-efficient buildings in response to the impacts of global warming. The objective of this work is to develop and verify a two-dimensional CFD solver implemented in the Python programming language, based on the finite volume method, capable of solving diffusive equations and the Navier-Stokes equations under laminar flow conditions, as well as to evaluate its computational performance. The numerical solution of the discretized governing equations is carried out using an Alternating Direction Implicit (ADI) method based on line Gauss-Seidel (LGS-ADI). The first case study corresponds to the solution of the steady-state heat conduction equation in a solid block. The second case consists of the simulation of laminar airflow in a square cavity filled with air, driven by a prescribed velocity at the top lid, with a Reynolds number of 1000. In this case, the SIMPLE algorithm is implemented for pressure-velocity coupling, and staggered grids are employed for the velocity field. The simulations are performed on a uniform 150x150 cell grid, initially using a first-order upwind scheme and subsequently a hybrid differencing scheme to reduce numerical diffusion. The results reproduce the characteristic flow structures, such as the formation of two vortices in the lower corners of the cavity, and show good agreement with the reference solution. The relative errors obtained in the verification cases are below 4.70%. Compared with a solver based on the classical Gauss-Seidel method, the ADI algorithm combined with the Thomas method reduces the computational time by 61.92% for the diffusive problem and by 15.04% for the forced flow problem, a difference attributed to the nonlinear nature of the latter. This work contributes to the development of an in-house code for numerical modeling using CFD.*

Keywords: Computational fluid dynamics; Finite volume method; ADI-SIMPLE algorithms.

ISSN: 2594-1925

1. Introducción

Las simulaciones numéricas mediante la Dinámica de Fluidos Computacional (CFD, por sus siglas en inglés) representan una herramienta esencial en ciencias e ingeniería para analizar los procesos de transferencia de calor y distribución del flujo en diversos sistemas de índole académico e industrial. Estas herramientas, sustentadas principalmente en el método numérico de volúmenes finitos, permiten encontrar soluciones aproximadas de modelos matemáticos complejos, así como evaluar distintos escenarios desde una computadora que en un experimento real resultaría demasiado costoso o imposible de efectuar. El método de volumen finito es el enfoque numérico más utilizado en códigos CFD, debido a que garantiza la conservación de las leyes físicas fundamentales y ofrece gran robustez para el tratamiento de geometrías y fenómenos físicos complejos [1].

Las simulaciones mediante CFD se pueden realizar mediante softwares comerciales, libres y códigos propios. En cuanto al primer grupo, la desventaja es que podrían resultar costosos y limitados para fines académicos. Ante esta situación, existen alternativas como el software libre y de código abierto, dentro de esta clasificación se encuentra OpenFOAM, el cual es un compendio de librerías escritas en C++ [2]. Su carácter de código abierto ha impulsado la modificación de sus códigos internos, siendo valioso en áreas de investigación. Por ejemplo, se han desarrollado librerías de propiedades térmicas no lineales para difusión térmica [3], condiciones de frontera personalizadas [4], nuevos modelos de turbulencia [5] y de radiación térmica [6].

El desarrollo de código propio se realiza de manera colaborativa o individual, suele implicar una demanda cognitiva alta, pero a su vez, aporta al avance tecnológico de una región y al fortalecimiento de conocimientos internos [7]. En la actualidad, se han desarrollado algoritmos para fenómenos complejos con cambio de fase, donde la transición entre una fase y otra comúnmente representa un desafío. Carlier y Papalexandris [8] implementaron un algoritmo numérico de seguimiento para interfases libres evaporativas, siendo verificado a través de una solución analítica para posteriormente simular casos 2D y 3D. Labihi et al. [9] validaron un código para modelar el cambio de fase sólido-líquido con una ecuación no lineal que se aproxima a la densidad variable real del material de cambio de fase. La importancia de estos trabajos radica en que se pueden aplicar en sistemas fototérmicos y enfriamiento pasivo de edificaciones. Particularmente, en la literatura reciente se reporta el modelado mediante volúmenes finitos de la conducción de calor en el suelo con el fin de evaluar el potencial de enfriamiento geotérmico [10], [11]. Estos modelos numéricos son capaces de incorporar propiedades térmicas variables, condiciones de frontera dependientes del clima y capturar detalles como la saturación térmica del suelo, siendo más precisos y relativamente sencillos comparados con las técnicas de solución analíticas y otros modelos numéricos simplificados.

México es uno de los principales países afectados por el calentamiento global, esto ha ocasionado el incremento del consumo de energía eléctrica en edificaciones del norte y sur del país [12]. Por lo tanto, distintos autores mexicanos han empleado códigos CFD propios para el modelado de los fenómenos de transferencia de calor que acontecen en habitaciones y en sistemas constructivos, permitiendo un diseño óptimo acorde al tipo de clima. Olazo et al. [13] emplearon un código propio escrito en Fortran que resuelve las ecuaciones gobernantes de la transferencia de calor conjugada en una habitación 2D. Los autores modelaron la convección natural turbulenta del aire en el interior de la habitación, así como la

ISSN: 2594-1925

conducción de calor en una ventana de doble vidrio y en el techo, analizando diferentes tipos de impermeabilizantes. En el estudio, los datos climáticos de la ciudad de Mérida, Yucatán, se incorporaron al modelo matemático como condiciones de frontera. Uriarte [14] desarrolló solucionadores de conducción de calor en coordenadas curvilíneas no ortogonales para techos con geometrías irregulares. Xamán et al. [15] implementaron métodos de malla fija para materiales de cambio de fase sólido-líquido. También se reporta la programación de solucionadores iterativos LGS-ADI en Fortran [16] y LBL-ADI de direcciones alternantes en C++ [17] aplicados a cavidades cuadradas llenas de aire. Aunque en la práctica los flujos turbulentos son dominantes, los flujos laminares también resultan bastante comunes, un ejemplo en sistemas constructivos mexicanos se presenta al interior de los huecos de bovedillas de concreto [18] y en los espacios de aire de las ventanas de triple vidrio [17].

La dificultad de escribir los códigos CFD mediante el método de volúmenes finitos recae en resolver y acoplar ecuaciones altamente no lineales, como las ecuaciones de Navier-Stokes que gobiernan el movimiento de los fluidos interactuando con otros mecanismos de transferencia de calor. Además, el modelo puede involucrar geometrías irregulares o complejas que requieren un sistema de coordenadas no ortogonal, y considerar el efecto 3D usualmente implica un incremento en el tiempo de cómputo, por lo que se recomienda en el análisis detallado de ventilación interna [19] y de la turbulencia [8]. En el caso de los materiales de cambio de fase, lo típico es modelar únicamente la conducción debido a que un cambio considerable en el volumen puede resultar perjudicial, como lo sería en sistemas constructivos.

La mayoría de los trabajos que emplean códigos propios son resultado de años de investigación, y como se evidenció, deben atravesar por una serie de verificaciones rigurosas [8], [9], [13], [15], [16], [17]. En ausencia de datos experimentales para comparar, la verificación es un proceso que implica contrastar los resultados del código numérico con el de otros autores, lo que permite determinar su correcta configuración por parte del usuario. También se puede efectuar la verificación con una solución analítica o un modelo numérico simplificado, esto en parte porque al modelar casos aplicados con códigos que emplean volúmenes finitos, el fenómeno se complejiza al involucrar distintos mecanismos de transferencia de calor, geometrías complejas o condiciones de frontera dependientes del clima.

La documentación de acceso libre de códigos CFD en español que emplean volúmenes finitos es desde la perspectiva de los autores limitada, existen algunos libros que explican los fundamentos teóricos [1] y otros que enseñan la programación de métodos numéricos en lenguajes como Python [20]. En la literatura se documentan módulos de aprendizaje para resolver las ecuaciones de Navier-Stokes 2D mediante el método de diferencias finitas en Python [21] y en Julia [22]. En Suecia, Kumar et al. [23] emplearon el lenguaje Python y optimizaron sus códigos con volúmenes finitos para cómputo de alto desempeño mediante paralelización y uso de GPUs. Python es popular debido a que su sintaxis resulta amigable con el usuario y es de libre acceso.

De igual manera, en la literatura reciente se reportan técnicas numéricas clásicas, como el uso de mallas desplazadas durante la solución de las ecuaciones de Navier-Stokes [24], y el solucionador clásico Gauss-Seidel [9]. Aunque el algoritmo LGS-ADI de direcciones alternantes es más eficiente que Gauss-Seidel clásico, su aplicación en problemas de verificación es poco documentado. Por lo tanto, debido a la importancia mencionada del desarrollo de código CFD, el objetivo de este trabajo es implementar y verificar un solucionador CFD bidimensional en Python, basado en el método de volumen finito, capaz de resolver ecuaciones difusivas y las ecuaciones de Navier-Stokes en régimen laminar en una cavidad

ISSN: 2594-1925

cuadrada con pared deslizante [24], así como evaluar el desempeño computacional de los solucionadores iterativos Gauss Seidel y LGS-ADI de direcciones alternantes.

2. Modelos físicos

La **figura 1** muestra los modelos físicos de los dos problemas a resolver en el presente trabajo, ambos se consideran en dos dimensiones, implican dominios rectangulares, y cuentan con dimensiones de 1 x 1 m. El primer problema consiste en modelar el fenómeno de conducción de calor que acontece en un bloque sólido sujeto a condiciones de frontera de temperaturas (T) de primera clase (T_B y T_A), la conductividad térmica (k) del material es 35 W/mK. En el segundo problema se contempla el caso clásico de una cavidad llena de aire a la que se le impone una velocidad horizontal (u) en la tapa superior. Con este valor de velocidad y las propiedades físicas de viscosidad (μ) y densidad (ρ), el número de Reynolds (Re= ρuL/μ) correspondiente a este caso es de 1000, siendo un flujo laminar. Las condiciones de frontera de las componentes de velocidad (u, v) están dadas por las condiciones de no deslizamiento, esto significa que el fluido se adhiere en las paredes debido a su viscosidad.

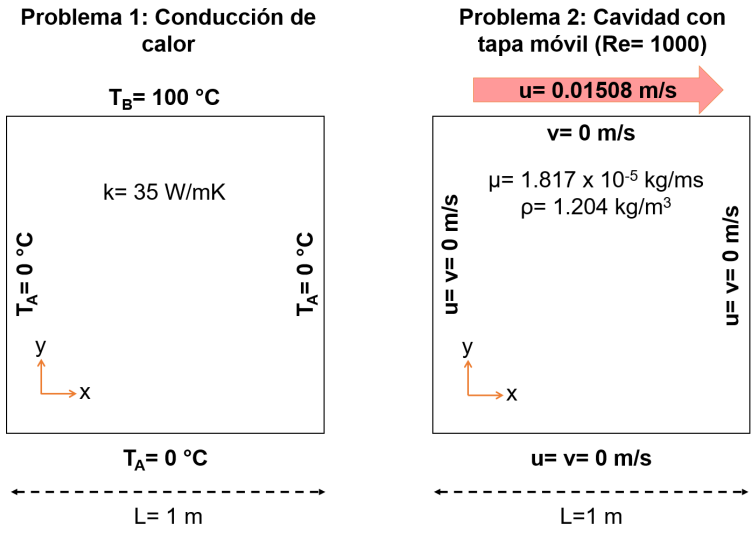


Figura 1. Modelos físicos de los problemas de referencia: a) conducción de calor y b) flujo forzado laminar.

3. Modelos matemáticos

La ecuación gobernante de la conducción de calor en estado estacionario y en dos dimensiones se presenta en la **ecuación 1**, donde los dos términos se denominan difusivos [1].

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)=0 \qquad (1)$$

La solución analítica de la ecuación 1 con las condiciones de frontera del primer problema se reporta por Incropera et al. [25] según la **ecuación 2**:

ISSN: 2594-1925

$$T(x,y) = T_A + \frac{2(T_B - T_A)}{\pi} \sum_{n=1}^{\infty} \frac{1-(-1)^n}{n} \frac{\operatorname{senh}\left(\frac{n\pi}{L}y\right)}{\operatorname{senh}(n\pi)} \operatorname{sen}\left(\frac{n\pi}{L}x\right) \quad (2)$$

Por otro lado, las ecuaciones que gobiernan el comportamiento físico de los fluidos se conocen en la literatura como ecuaciones de Navier-Stokes, los cuales expresan principios físicos de conservación [26]. Considerando el enfoque euleriano, la **ecuación 3** expresa el principio de conservación de masa.

$$\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \quad (3)$$

Las **ecuaciones 4** y **5** representan la conservación de *momentum*, donde P es la presión y F representa alguna fuerza adicional presente en el sistema, como pudiera ser la fuerza de flotabilidad en convección natural o la electromagnética, por mencionar algunos [27].

$$\frac{\partial(\rho u \cdot u)}{\partial x} + \frac{\partial(\rho v \cdot u)}{\partial y} = \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial u}{\partial y}\right) - \frac{\partial P}{\partial x} + F_x \quad (4)$$

$$\frac{\partial(\rho u \cdot v)}{\partial x} + \frac{\partial(\rho v \cdot v)}{\partial y} = \frac{\partial}{\partial x}\left(\mu \frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial v}{\partial y}\right) - \frac{\partial P}{\partial y} + F_y \quad (5)$$

En las ecuaciones de *momentum*, los primeros dos términos se identifican como términos convectivos, mientras que los dos primeros términos del lado derecho de la igualdad corresponden a los términos difusivos, y el penúltimo término es el gradiente de presión. La solución de las ecuaciones de Navier-Stokes es complicada por dos factores: los términos convectivos son altamente no lineales y no hay una ecuación explícita para la presión, sino que se formula con base en la conservación de masa [28]. Esto ocasiona que las soluciones analíticas se limiten a casos simplificados, por lo que en las aplicaciones prácticas se necesitan técnicas numéricas para sobrellevar las no linealidades y encontrar soluciones aproximadas.

4. Metodología de solución numérica

4.1. Discretización por volúmenes finitos

En el método de volumen finito las ecuaciones gobernantes se integran en cada volumen de control de un espacio discretizado, lo que permite el cumplimiento de los principios físicos de conservación. En esta sección y en las posteriores se dará énfasis al proceso de discretización de las ecuaciones de Navier-Stokes, teniendo en cuenta que la discretización de la ecuación de conducción de calor sigue la misma filosofía.

ISSN: 2594-1925

En la estrategia de solución numérica se consideró la linealización de los términos convectivos de las ecuaciones de *momentum*, esto significa que las componentes de velocidad (u, v) se suponen inicialmente como valores conocidos. Simultáneamente, se define ϕ como incógnita, la cual representaría los valores de u y v actualizados después de una iteración con los valores supuestos [29]. La **ecuación 6** expresa esta linealización.

$$\frac{\partial(\rho u\cdot\phi)}{\partial x}+\frac{\partial(\rho v\cdot\phi)}{\partial y}=\frac{\partial}{\partial x}\left(\mu\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial\phi}{\partial y}\right)-\frac{\partial P}{\partial x} \tag{6}$$

En la **figura 2** se presenta el espacio discreto 2D, también conocido como malla. Esta malla posee volúmenes de control o celdas de igual espesor. Se denota al nodo principal con la letra P, y a los vecinos con las letras E, W, N y S (este, oeste, norte y sur). De igual manera, las interfases o caras en común entre la celda principal y las vecinas se representan con las mismas letras en minúsculas (e, w, n y s).

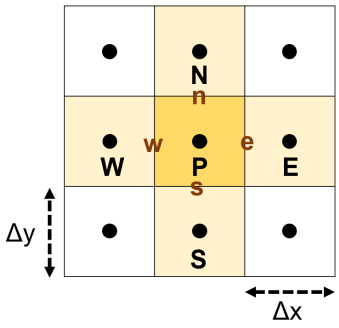


Figura 2. Dominio 2D discretizado.

Al efectuar la integración de los términos convectivos de las ecuaciones de *momentum* resultan las **ecuaciones** 7 y 8.

$$\int_w^e\int_s^n\frac{\partial(\rho u\cdot\phi)}{\partial x}dx\cdot dy=\int_s^n[(\rho u\cdot\phi)_e-(\rho u\cdot\phi)_w]\cdot dy=[(\rho u\cdot\phi)_e-(\rho u\cdot\phi)_w]\cdot \Delta y \tag{7}$$

$$\int_w^e\int_s^n\frac{\partial(\rho v\cdot\phi)}{\partial y}dx\cdot dy=\int_w^e[(\rho v\cdot\phi)_n-(\rho v\cdot\phi)_s]\cdot dx=[(\rho v\cdot\phi)_n-(\rho v\cdot\phi)_s]\cdot \Delta x \tag{8}$$

Al efectuar la integración de los términos difusivos de las ecuaciones de *momentum* resultan las **ecuaciones** 9 y 10.

$$\int_w^e\int_s^n\frac{\partial}{\partial x}\left(\mu\frac{\partial\phi}{\partial x}\right)dx\cdot dy=\int_s^n\left[\left(\mu\frac{\partial\phi}{\partial x}\right)_e-\left(\mu\frac{\partial\phi}{\partial x}\right)_w\right]\cdot dy=\left[\left(\mu\frac{\Delta\phi}{\Delta x}\right)_e-\left(\mu\frac{\Delta\phi}{\Delta x}\right)_w\right]\cdot \Delta y \tag{9}$$

ISSN: 2594-1925

$$\int_w^e \int_s^n \frac{\partial}{\partial y}\left(\mu \frac{\partial \phi}{\partial y}\right) dx \cdot dy = \int_w^e \left[\left(\mu \frac{\partial \phi}{\partial y}\right)_n - \left(\mu \frac{\partial \phi}{\partial y}\right)_s\right] \cdot dx = \left[\left(\mu \frac{\Delta \phi}{\Delta y}\right)_e - \left(\mu \frac{\Delta \phi}{\Delta y}\right)_s\right] \cdot \Delta x \quad (10)$$

Si se definen los flujos másicos convectivos como F= ρuΔy, pvΔx (dependiendo de la dirección); y los aportes de conductancia por difusión como D= μ/Δx, μ/Δy, la ecuación discreta sin considerar el gradiente de presión es la **ecuación 11**:

$$[F_e\phi_e - F_w\phi_w] + [F_n\phi_n - F_s\phi_s] = [D_e\phi_e - D_w\phi_w] + [D_n\phi_n - D_s\phi_s)] \quad (11)$$

Análogamente, la ecuación discreta de conservación de masa es la **ecuación 12**.

$$[F_e - F_w] + [F_n - F_s] = 0 \quad (12)$$

Los términos F y D pueden evaluarse en cada iteración. Sin embargo, las incógnitas ϕ en las caras no son conocidas como si lo son en los nodos de la malla, por lo que debe seleccionarse un esquema de interpolación o esquema numérico.

4.2. Esquemas numéricos

En este trabajo inicialmente se configuraron los esquemas *upwind* de primer orden para estabilizar la solución debido a que es incondicionalmente acotada [1]. Una vez que se alcanzó el criterio de convergencia, se efectuó el cambio a esquemas híbridos con el fin de reducir la difusión numérica inducida [30].

La aproximación de las incógnitas con el esquema atrasado para los términos convectivos ($\phi_e = \phi_P$, $\phi_w = \phi_W$, $\phi_n = \phi_P$, $\phi_s = \phi_S$), y centrado para los difusivos, resultaría en la **ecuación 13**.

$$[F_e\phi_P - F_w\phi_W] + [F_n\phi_P - F_s\phi_S] = [D_e(\phi_E - \phi_P) - D_w(\phi_P - \phi_W)] + [D_n(\phi_N - \phi_P) - D_s(\phi_P - \phi_S)] \quad (13)$$

De manera análoga, con el esquema adelantado para los términos convectivos ($\phi_e = \phi_E$, $\phi_w = \phi_P$, $\phi_n = \phi_N$, $\phi_s = \phi_P$) resulta la **ecuación 14**. La diferencia entre el esquema adelantado y atrasado radica evidentemente en la dirección del flujo.

$$[F_e\phi_E - F_w\phi_P] + [F_n\phi_N - F_s\phi_P] = [D_e(\phi_E - \phi_P) - D_w(\phi_P - \phi_W)] + [D_n(\phi_N - \phi_P) - D_s(\phi_P - \phi_S)] \quad (14)$$

ISSN: 2594-1925

Reordenando y factorizando los términos se obtiene la ecuación discreta, sin considerar el gradiente de presión, y sus coeficientes (a_P, a_E, a_W, a_N, a_S) en la **ecuación 15**.

$$a_P\phi_P = a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S \quad (15)$$

De acuerdo con Xamán y Gijón [1], la formulación de los esquemas numéricos puede expresarse de forma general mediante las **ecuaciones** 16, 17, 18 y 19. La función máximo toma en consideración la dirección según los esquemas adelantado y atrasado. El valor de S depende del tipo de esquema, en la **tabla 1** se presentan sus ecuaciones, las cuales permiten el cambio de esquemas *upwind* a esquemas de segundo orden e híbridos.

$$a_E = D_e \cdot S + \max(-F_e, 0) \quad (16)$$

$$a_W = D_w \cdot S + \max(F_w, 0) \quad (17)$$

$$a_N = D_n \cdot S + \max(-F_n, 0) \quad (18)$$

$$a_S = D_s \cdot S + \max(F_s, 0) \quad (19)$$

Tabla 1. Esquemas numéricos.

Esquema numérico	**S**
Híbrido	Max $[0, (1 - 0.5\left\|\frac{F}{D}\right\|)]$
Upwind	1

Nota: La división F/D se conoce como número de Peclet (Pe) [1].

4.3. Malla desplazada

En este trabajo se empleó la técnica de la malla desplazada que acopla las ecuaciones de continuidad y *momentum*, a su vez, evita inconsistencias durante el cálculo del gradiente de presión. En la **figura 3** se ilustra la malla principal, donde se almacenan las variables escalares en las coordenadas I, J, como la presión, temperatura y viscosidad. Las mallas desplazadas almacenan información de las variables vectoriales, en este caso, las velocidades. Se puede apreciar de la figura que existe una malla desplazada por cada componente de velocidad, de tal manera que las coordenadas de la malla desplazada en x son i, J; y las de la malla desplazada en y son I, j. En resumen, en la malla principal se cuenta con Nx +2 y Ny

ISSN: 2594-1925

+ 2 nodos, contabilizando los nodos en las fronteras, mientras que a las desplazadas se les resta un nodo en la dirección desplazada.

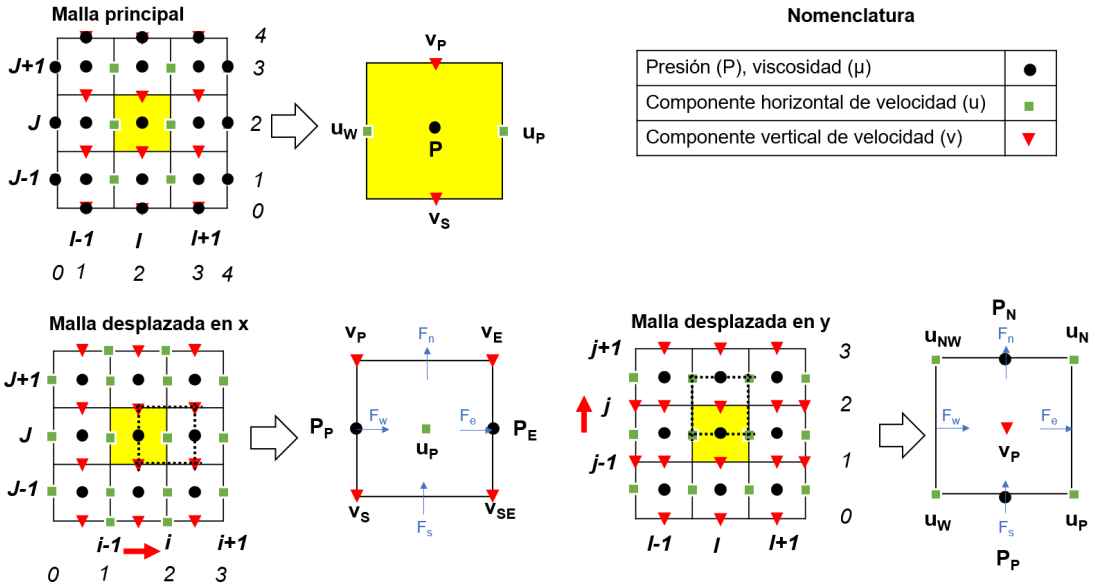


Figura 3. Mallas desplazadas.

4.4. Algoritmo SIMPLE

El algoritmo SIMPLE (método semi-implícito para el acople de presión, por sus siglas en inglés) es generalmente aplicado en flujos compresibles e incompresibles en estado estacionario. Retomando las ecuaciones de *momentum*, la discretización final agregando los términos de gradientes de presión quedarían como las **ecuaciones 20 y 21**.

$$a^{u}_{i,J} \cdot u_{i,J} = a^{u}_{P} \cdot u_{P} = \sum a^{u}_{vecinos} \cdot u_{vecinos} - (P_E - P_P) \cdot \Delta y + b \qquad (20)$$

$$a^{v}_{I,j} \cdot v_{I,j} = a^{v}_{P} \cdot v_{P} = \sum a^{v}_{vecinos} \cdot v_{vecinos} - (P_N - P_P) \cdot \Delta x + b \qquad (21)$$

Donde b es un término fuente. El primer paso inicia suponiendo un campo de presión y velocidades, por lo que se pueden obtener los valores supuestos de u* y v*.

Sin embargo, es necesario corregir estos valores considerando correcciones de presión (P') y velocidades (u', v') para obtener los resultados correctos (P, u, v). Esto se representa en la **ecuación 22**:

$$\phi = \phi^{*} + \phi' \qquad (22)$$

ISSN: 2594-1925

Las ecuaciones de corrección mostradas en las **ecuaciones** 23 y 24 se generan al despejar los términos correspondientes en la ecuación 22. Si se considera que el término fuente es el mismo en las ecuaciones correctas y supuestas, simplemente se anula.

$$u' = u - u^* = a_P^u \cdot u'_P = \sum a^u_{vecinos} \cdot u'_{vecinos} - (P'_E - P'_P) \cdot \Delta y \tag{23}$$

$$v' = v - v^* = a_P^v \cdot v'_P = \sum a^v_{vecinos} \cdot v'_{vecinos} - (P'_N - P'_P) \cdot \Delta x \tag{24}$$

La principal característica del algoritmo SIMPLE es que la sumatoria de los coeficientes y velocidades vecinas se aproxima a cero, lo cual es válido cuando la solución numérica progresa debido a que en flujos estacionarios el balance neto de masa y *momentum* debe ser cercano a cero. Por lo tanto, las ecuaciones finales de corrección de velocidades se presentan en las **ecuaciones** 25 y 26. No obstante, la dependencia directa de las velocidades a las presiones origina que la solución numérica tienda a divergir, por esta razón, deben aplicarse factores de subrelajación.

$$u'_P = \frac{\Delta y}{a_P^u}(P'_P - P'_E) = d_e(P'_P - P'_E) \tag{25}$$

$$v'_P = \frac{\Delta x}{a_P^v}(P'_P - P'_N) = d_n(P'_P - P'_N) \tag{26}$$

El segundo paso consiste en determinar la ecuación de corrección de presión P' a partir de la ecuación de continuidad en la malla principal y de los valores calculados de u* y v*. En la **ecuación 27** se presenta la ecuación discreta final de presión, donde las velocidades se sustituyen por sus ecuaciones de corrección. Y la **ecuación 28** incluye el balance de masa con las velocidades supuestas.

$$\begin{gathered} b' + [\rho_e u_e \Delta y - \rho_w u_w \Delta y] + [\rho_n v_n \Delta x - \rho_s v_s \Delta x] = b' + [\rho_e \Delta y d_e(P'_P - P'_E) - \\ \rho_w \Delta y d_w(P'_W - P'_P)] + [\rho_n \Delta x d_n(P'_P - P'_N) - \rho_s \Delta x d_s(P'_S - P'_P)] = -a_P P'_P + \\ a_E P'_E + a_W P'_W + a_N P'_N + a_S P'_S + b' \end{gathered} \tag{27}$$

$$b' = [(\rho u^*)_w \Delta y - (\rho u^*)_e \Delta y] + [(\rho v^*)_s \Delta x - (\rho v^*)_n \Delta x] \tag{28}$$

Finalmente, las velocidades y presiones correctas se calculan en las **ecuaciones** 29, 30 y 31, donde α es el factor de subrelajación:

$$P_{I,J} = P_P = P_P^* + \alpha_P \cdot P'_P \tag{29}$$

ISSN: 2594-1925

$$u_{i,J} = u_P = u_P^* + \alpha_u \cdot \frac{\Delta y}{a_P^u}(P_P' - P_E') \tag{30}$$

$$v_{I,j} = v_P = v_P^* + \alpha_v \cdot \frac{\Delta x}{a_P^v}(P_P' - P_N') \tag{31}$$

La **figura 4** presenta el diagrama de flujo completo del código Python aplicando el algoritmo SIMPLE. Se configuraron 30 iteraciones en el lazo interno de la presión, esto permitió estabilizar el código. El criterio de paro del lazo interno se cumple cuando se alcanza el número máximo de iteraciones o el criterio de convergencia de presión. El criterio de convergencia global se asignó en residuales de presión y velocidades menores o iguales a $1x10^{-8}$. Se calcularon los residuales basados en la desviación cuadrática media [1]. Debido a que las condiciones de fronteras del campo de presiones son gradiente cero normal a la pared cavidad, se induce que la matriz de presiones sea singular, por lo que se debe asignar un valor de presión en algún nodo de la malla [29]. Esta estrategia es replicada en softwares más sofisticados como OpenFOAM [31].

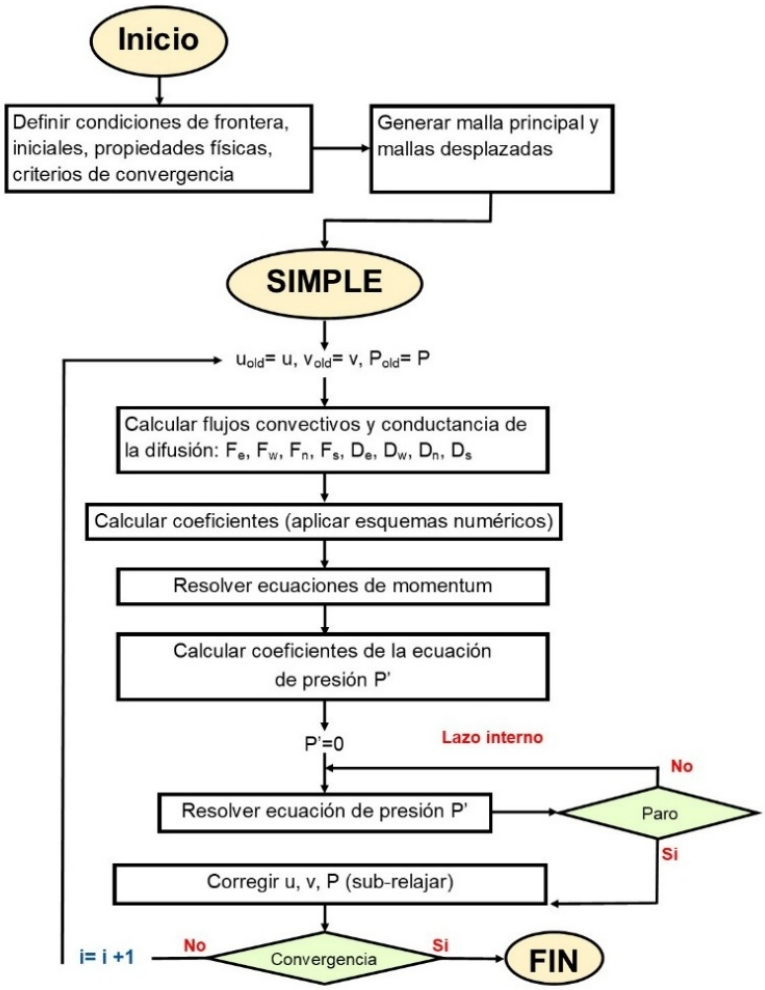


Figura 4. Diagrama de flujo del código en Python con el algoritmo SIMPLE.

ISSN: 2594-1925

4.5. Solucionadores numéricos

En la **figura 5** se observa la diferencia entre el solucionador numérico matricial de Jacobi y Gauss-Seidel, donde este último es más eficaz computacionalmente al emplear los valores calculados en la iteración actual, siendo un solucionador implícito. Sin embargo, en problemas estacionarios no lineales es recomendable subrelajar las ecuaciones para evitar la divergencia de la simulación.

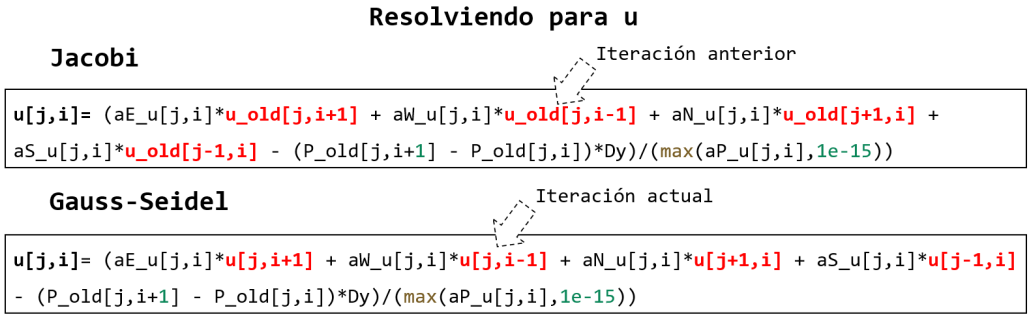
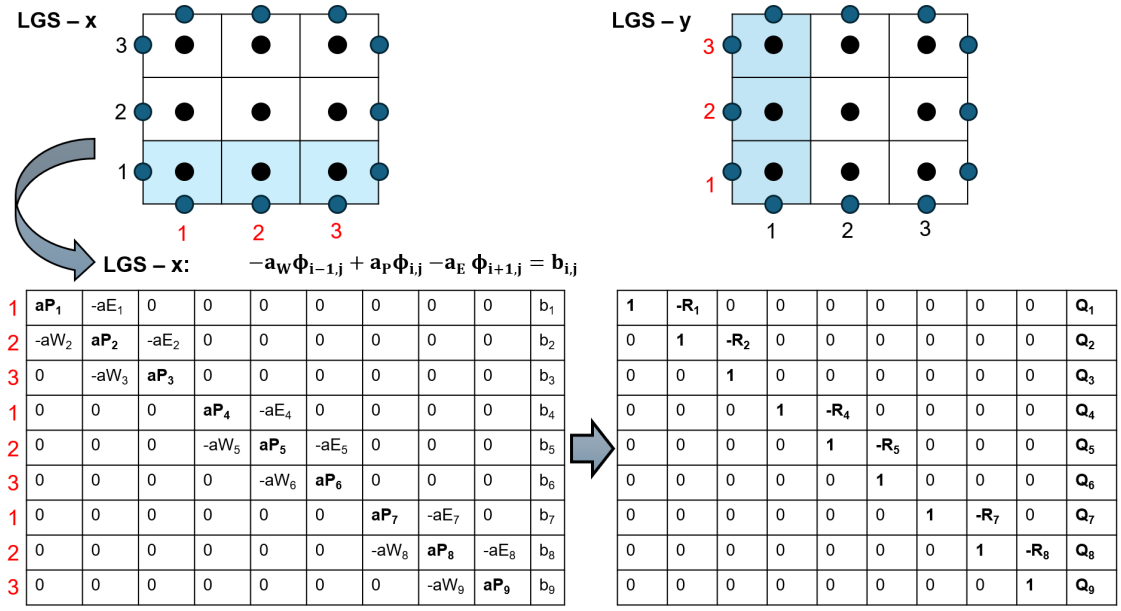


Figura 5. Solucionador numérico de Jacobi y Gauss-Seidel en Python.

El método de direcciones alternantes LGS-ADI se basa en el algoritmo de Thomas en una dimensión, también conocido como algoritmo para matrices tridiagonales (TDMA) [1]. Este método permite resolver sistemas de ecuaciones que surgen en problemas bidimensionales y tridimensionales. Se trata de un solucionador ampliamente utilizado debido a su eficiencia computacional, ya que descompone el problema multidimensional en una serie de problemas unidimensionales. Para ello, realiza barridos alternados por filas (LGS-x) y columnas (LGS-y), utilizando relaciones de recurrencia que incluyen pasos de resolución hacia adelante y hacia atrás.

La **figura 6** muestra el desarrollo de la matriz en la dirección x. Se muestran los coeficientes de las ecuaciones y términos fuente en cada celda de la malla. En el código, los coeficientes en las fronteras también se agregaron al término fuente, por ejemplo, en la ubicación (1, 1) de LGS-x, el coeficiente aW se contempla en el término b. Después de aplicar los pivotes correspondientes se forma la matriz tridiagonal superior característica, donde surgen las relaciones de recurrencia R y Q.

ISSN: 2594-1925



1	aP_1	$-aE_1$	0	0	0	0	0	0	0	b_1
2	$-aW_2$	aP_2	$-aE_2$	0	0	0	0	0	0	b_2
3	0	$-aW_3$	aP_3	0	0	0	0	0	0	b_3
1	0	0	0	aP_4	$-aE_4$	0	0	0	0	b_4
2	0	0	0	$-aW_5$	aP_5	$-aE_5$	0	0	0	b_5
3	0	0	0	0	$-aW_6$	aP_6	0	0	0	b_6
1	0	0	0	0	0	0	aP_7	$-aE_7$	0	b_7
2	0	0	0	0	0	0	$-aW_8$	aP_8	$-aE_8$	b_8
3	0	0	0	0	0	0	0	$-aW_9$	aP_9	b_9

1	$-R_1$	0	0	0	0	0	0	0	Q_1
0	1	$-R_2$	0	0	0	0	0	0	Q_2
0	0	1	0	0	0	0	0	0	Q_3
0	0	0	1	$-R_4$	0	0	0	0	Q_4
0	0	0	0	1	$-R_5$	0	0	0	Q_5
0	0	0	0	0	1	0	0	0	Q_6
0	0	0	0	0	0	1	$-R_7$	0	Q_7
0	0	0	0	0	0	0	1	$-R_8$	Q_8
0	0	0	0	0	0	0	0	1	Q_9

Figura 6. Método de direcciones alternantes LGS-ADI.

En la **tabla 2** se presentan las relaciones de recurrencia del método. Se observa que, en el último nodo de la última fila o columna, la relación Q es igual a la incógnita ϕ. La incógnita calculada al término del algoritmo LGS-x es empleada para actualizar su valor en el algoritmo LGS-y.

Tabla 2. Relaciones de recurrencia para el método LGS- ADI.

	LGS – x	**LGS – y**
Primera columna (LGS – x) / Primera fila (LGS – y)	$R_{1,j} = \frac{aE_{1,j}}{aP_{1,j}}$	$R_{i,1} = \frac{aN_{1,j}}{aP_{1,j}}$
	$Q_{1,j} = \frac{b_{1,j}}{aP_{1,j}}$	$Q_{i,1} = \frac{b_{i,1}}{aP_{i,1}}$
Columnas / filas intermedias	$R_{i,j} = \frac{aE_{i,j}}{aP_{i,j} - aW_{i,j} \cdot P_{i-1,j}}$	$R_{i,j} = \frac{aN_{i,j}}{aP_{i,j} - aS_{i,j} \cdot P_{i,j-1}}$
	$Q_{i,j} = \frac{b_{i,j} + aW_{i,j} \cdot Q_{i-1,j}}{aP_{i,j} - aW_{i,j} \cdot P_{i-1,j}}$	$Q_{i,j} = \frac{b_{i,j} + aS_{i,j} \cdot Q_{i,j-1}}{aP_{i,j} - aS_{i,j} \cdot P_{i,j-1}}$
Última columna (LGS – x) / última fila (LGS – y)	$R_{N,j} = 0$	$R_{i,N} = 0$
	$Q_{N,j} = \phi_{N,j}$	$Q_{i,N} = \phi_{i,N}$

Las ecuaciones de sustitución hacia atrás están dadas por las **ecuaciones** 32 y 33:

$$\phi_{i,j} = R_{i,j} \cdot \phi_{i+1,j} + Q_{i,j} \tag{32}$$

$$\phi_{i,j} = R_{i,j} \cdot \phi_{i,j+1} + Q_{i,j} \tag{33}$$

ISSN: 2594-1925

5. Resultados y discusiones

5.1. Problema 1

5.1.1. Tiempo de cómputo e iteraciones

En la **tabla 3** se observa que la disminución de tiempo de cómputo fue de 61.92% al emplear el solucionador LGS-ADI, esto es equivalente a 17.86 s. En contraste, la disminución de las iteraciones fue mayor, con un 74.91% y 1544 iteraciones menos. Esto se debe en parte a que con la subrutina LGS-ADI se realizan mayores operaciones, pero se aceleran los cálculos hacia el criterio de convergencia.

Tabla 3. Tiempo de cómputo e iteraciones del problema 1.

Tiempo de cómputo (s)			Iteraciones		
Gauss-Seidel	**LGS-ADI**	**Disminución**	**Gauss-Seidel**	**LGS-ADI**	**Disminución**
29.14	11.28	17.86 (61.92%)	2061	517	1544 (74.91%)

En la **figura 7** se observan los residuales del problema por cada solucionador empleado. Se aprecia la congruencia observada en la **tabla 3**, con Gauss-Seidel, la pendiente de la curva es más pronunciada, con un total de 2001 iteraciones. Los residuales no oscilan debido a la naturaleza lineal del problema.

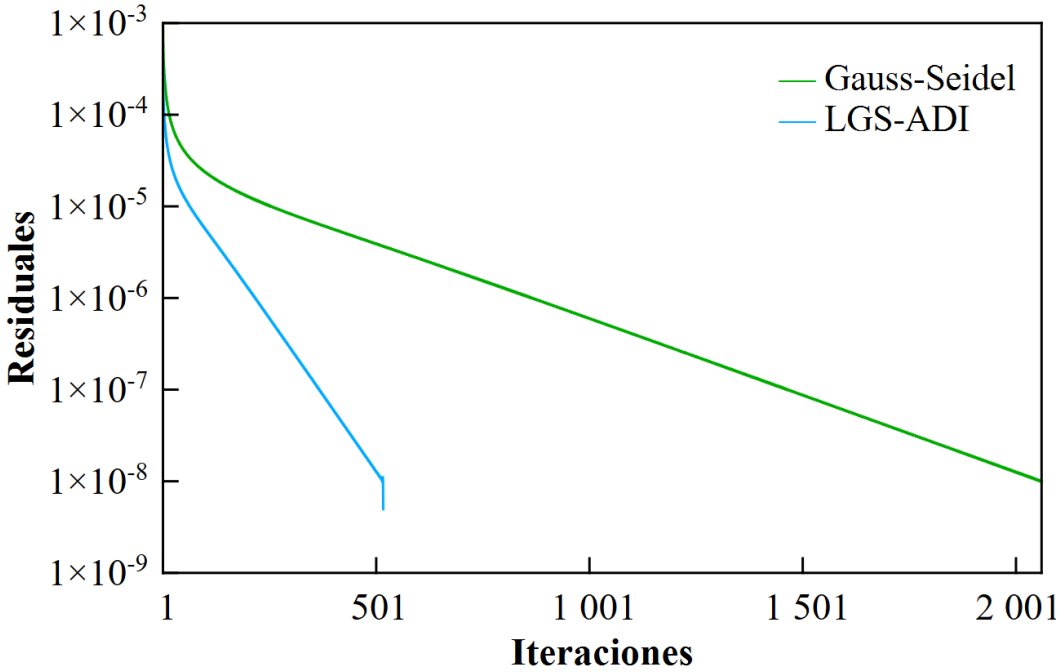


Figura 7. Residuales del problema de conducción de calor.

ISSN: 2594-1925

Aunque las diferencias puedan resultar despreciables en términos del tiempo, la cierto es que, el solucionador LGS-ADI es eficiente en problemas de conducción aplicados, sobre todo en problemas lineales. Esta subrutina fue empleada para resolver la conducción de calor en ventanas de vidrio bajo condiciones climáticas de la ciudad de San Francisco de Campeche [32].

5.1.2. Verificación

La **figura 8** compara los resultados obtenidos del código Python con el de la solución analítica reportada por Incropera et al. [25]. Los resultados se obtuvieron sobre línea vertical en x= 0.5 m. Se aprecia buena concordancia, con un error relativo porcentual máximo de 0.16% en y= 0.1 m.

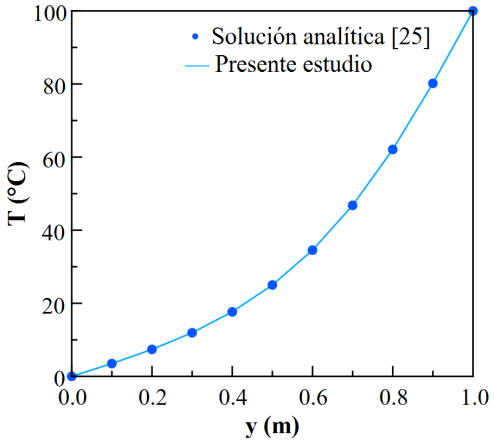


Figura 8. Verificación del problema de conducción de calor.

En la **figura 9** se ilustran las isotermas del bloque sólido. Puesto que la temperatura es mucho mayor en la pared horizontal superior, las isotermas se concentran en las esquinas superiores y disminuyen conforme se avanza a la pared horizontal inferior donde la temperatura es menor e igual a 0 °C.

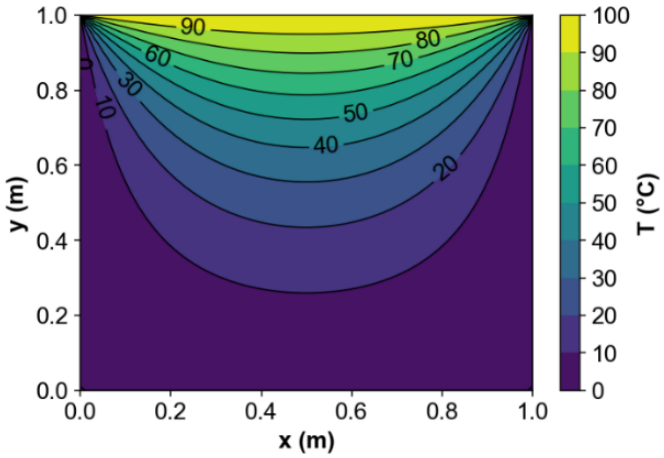


Figura 9. Isotermas del bloque sólido del problema 1 en Python.

ISSN: 2594-1925

5.2. Problema 2

5.2.1. Análisis del criterio de convergencia e independencia de malla

En la **figura 10** se muestra la independencia espacial con mallas de 150x150 y 200x200 elementos. El esquema *upwind* fue configurado. La malla de 150x150 y el criterio de convergencia para el residual de presión de $1x10^{-6}$ producen un error relativo porcentual máximo de 34.75% comparado con la solución numérica de Ghia et al. [24]. Al exigir un residual menor, de $1x10^{-8}$, el error se reduce 16.11%, pero las diferencias aún son significativas. El mismo análisis se efectuó con la malla de 200x200, el error relativo porcentual fue de 41.25% con el residual de $1x10^{-6}$ y de 15.21% con el residual de $1x10^{-8}$. El comportamiento evidencia que a este número de Reynolds se recomienda emplear mallas no uniformes con refinamiento en la cercanía de las paredes. La otra estrategia empleada consiste en estabilizar la simulación utilizando una malla uniforme con un esquema *upwind*, y después cambiar por un esquema de segundo orden o híbrido con el fin de reducir la difusión numérica inducida por el esquema de primer orden [30]. Esto a su vez, previene problemas de divergencia si se define un esquema de orden mayor.

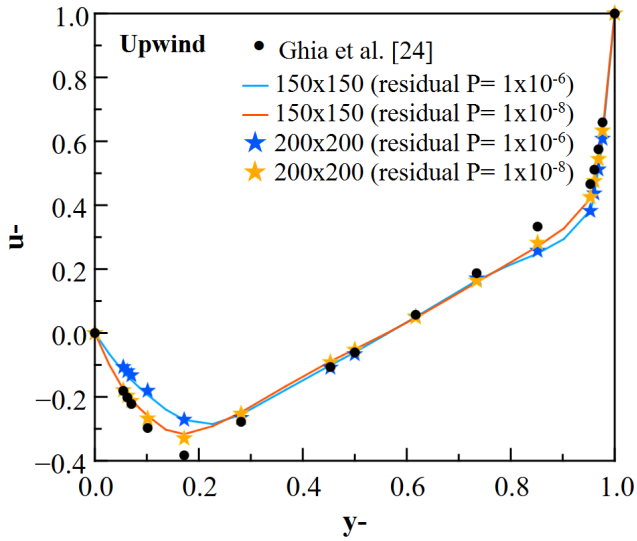


Figura 10. Proceso de independencia de malla espacial y análisis de residuales con el esquema *upwind*.

En la **figura 11** se ilustran los resultados al emplear ambas mallas con un esquema híbrido. Al realizar algunas pruebas se determinó que el residual de presión igual a $1x10^{-8}$ resulta ideal en este caso. En primera instancia, se observa una mejor aproximación, el error máximo se redujo a 4.67% al emplear la malla de 150x150, y a 5.10% con la malla de 200x200. Por lo tanto, los resultados con la malla de 150x150 son suficientes.

ISSN: 2594-1925

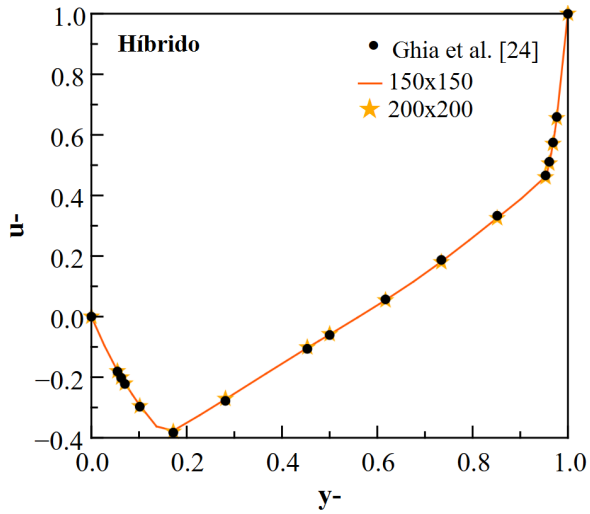


Figura 11. Proceso de independencia de malla espacial y análisis de residuales con el esquema híbrido.

5.2.2. Tiempo de cómputo e iteraciones

En la **tabla 4** se presentan los resultados de la simulación numérica empleando la malla uniforme de 150x150, el esquema *upwind* y un criterio de convergencia para la ecuación de presión de $1x10^{-6}$. Al comparar el tiempo de cómputo de los dos solucionadores, se encontró que el método LGS-ADI reduce el tiempo en 5.08 min, siendo equivalente al 11.00%. Como se suscitó en el problema difusivo, la disminución en el número de iteraciones fue significativamente mayor, ya que el uso de LGS-ADI permitió evitar 613 iteraciones, es decir, el 57.07%.

Tabla 4. Tiempo de cómputo e iteraciones del problema 2 con el esquema *upwind*.

Tiempo de cómputo (min)			Iteraciones		
Gauss-Seidel	**LGS-ADI**	**Disminución**	**Gauss-Seidel**	**LGS-ADI**	**Disminución**
46.15	41.07	5.08 (11.00%)	1074	461	613 (57.07%)

Por otra parte, una vez que se obtuvieron resultados con el esquema *upwind*, estos se configuraron como condiciones iniciales para la segunda parte de la simulación con el esquema híbrido y un criterio de convergencia de $1x10^{-8}$. La **tabla 5** resume los resultados y algunos comportamientos observados. La reducción del tiempo computacional alcanzó un máximo de 19.02 min, lo que representa el 15.04%. En cuanto a las iteraciones, la reducción fue de 47.15%, esto puede atribuirse al esquema numérico que tiende a producir oscilaciones. Estas simulaciones se ejecutaron con un factor de subrelajación de presión de 0.2, cuando este valor se aumentó a 0.4, se registró la divergencia con el solucionador Gauss-Seidel, por lo que únicamente los resultados del solucionador LGS-ADI se reportaron y evidentemente tanto el tiempo como las iteraciones disminuyeron aún más.

ISSN: 2594-1925

Tabla 5. Tiempo de cómputo e iteraciones del problema 2 con el esquema híbrido.

	Gauss-Seidel (α_P= 0.2)	**LGS-ADI (α_P= 0.2)**	**Disminución**	**LGS-ADI (α_P= 0.4)**	**Disminución**
Tiempo de cómputo (min)	126.33	107.32	19.02 (15.04%)	74.98	51.35 (59.35%)
Iteraciones	1777	939	838 (47.15%)	672	1105 (62.18%)

En la **figura 12** se muestran los residuales de la presión y de componentes de velocidades empleando el esquema híbrido y el factor de subrelajación de presión igual a 0.2. Se observa que los residuales poseen un comportamiento oscilante. Además, hasta en la iteración 280 los residuales de presión son aproximadamente iguales tanto con el solucionador Gauss-Seidel y LGS-ADI, después de esto, se aprecia que el LGS-ADI acelera la convergencia. Los resultados sugieren que el ahorro en las iteraciones y el tiempo de cómputo son proporcionales a un criterio de convergencia más exigente.

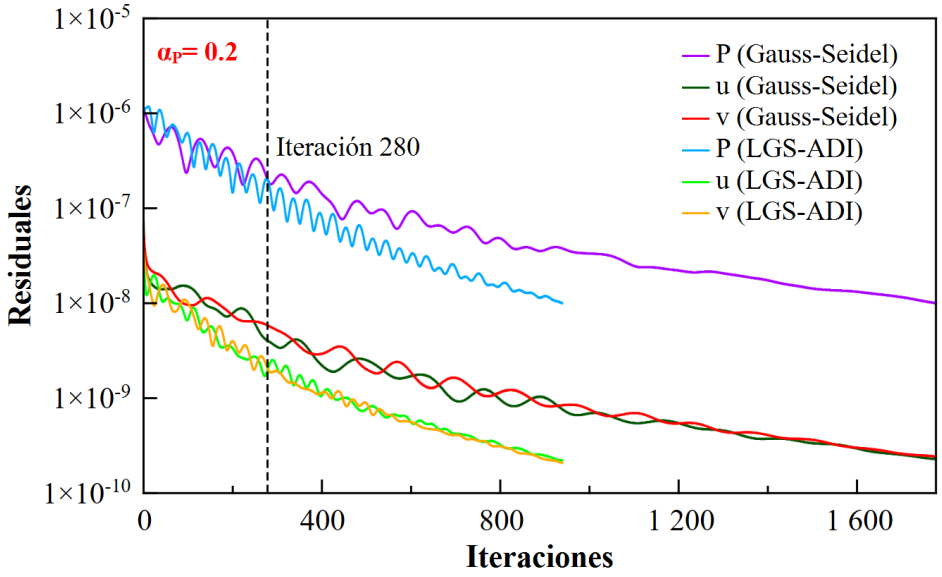


Figura 12. Residuales del problema de flujo laminar con α_P= 0.2.

En la **figura 13** se ilustran los residuales considerando el factor de subrelajación de presión de 0.4. Se demuestra que, con este valor, la simulación diverge al emplear el solucionador Gauss-Seidel. Inicialmente, la presión se desestabiliza y esto se propaga a través del campo de velocidades.

ISSN: 2594-1925

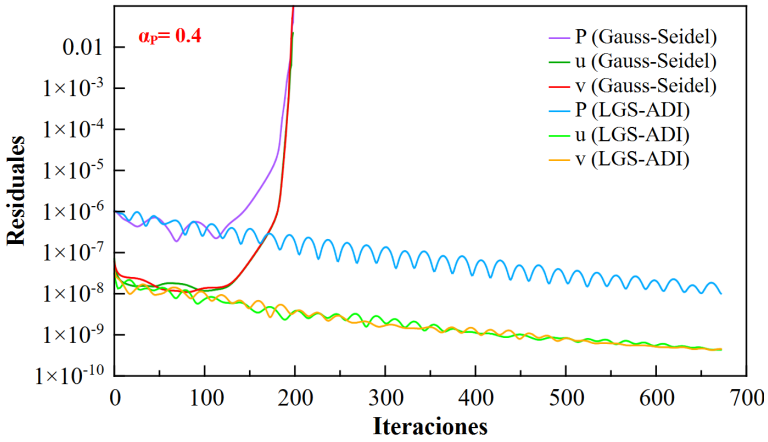


Figura 13. Residuales del problema de flujo laminar con α_P= 0.4.

5.2.3. Verificación

En la **tabla 6** se presenta la verificación del código Python al comparar con la solución de referencia de Ghia et al. [24] y con AbdelMigid et al. [33]. El error relativo porcentual máximo del presente estudio con respecto a la solución de referencia es de 4.667%, el cual se obtuvo en y-= 0.617. Del mismo modo, el error máximo comparado con la solución de AbdelMigid et al. [33] fue de 4.523%, valor obtenido en las cercanías de la tapa móvil, en y-= 0.969.

Tabla 6. Verificación del problema 2.

y-	u- (Ghia et al. [24])	u- (Presente estudio)	Error (%)	u- (AbdelMigid et al. [33])	Error (%)
0.000	0.000	0.000	0.000	0.000	0.000
0.055	-0.181	-0.179	1.131	-0.179	0.939
0.063	-0.202	-0.200	1.215	-0.202	0.050
0.070	-0.222	-0.219	1.275	-0.224	0.720
0.102	-0.297	-0.293	1.379	-0.302	1.514
0.172	-0.383	-0.377	1.639	-0.388	1.410
0.281	-0.278	-0.273	1.889	-0.279	0.467
0.453	-0.107	-0.103	2.991	-0.108	1.315
0.500	-0.061	-0.059	2.737	-0.062	1.974
0.617	0.057	0.054	4.667	0.058	1.754
0.734	0.187	0.181	3.202	0.191	1.923
0.852	0.333	0.326	2.258	0.338	1.592
0.953	0.466	0.462	0.844	0.478	2.489
0.961	0.511	0.508	0.709	0.528	3.365
0.969	0.575	0.572	0.532	0.601	4.523
0.977	0.659	0.656	0.545	0.676	2.472
1.000	1.000	1.000	0.000	1.000	0.000

ISSN: 2594-1925

En la **figura 14** se ilustran los resultados de las verificaciones para cada componente de velocidad. En cuanto a la componente vertical (v-), el error relativo porcentual máximo respecto a la solución de referencia es de 2.191% en x-= 0.8047. Respecto a la solución de AbdelMigid et al. [33], el error máximo es de 2.747% en x-= 0.9531.

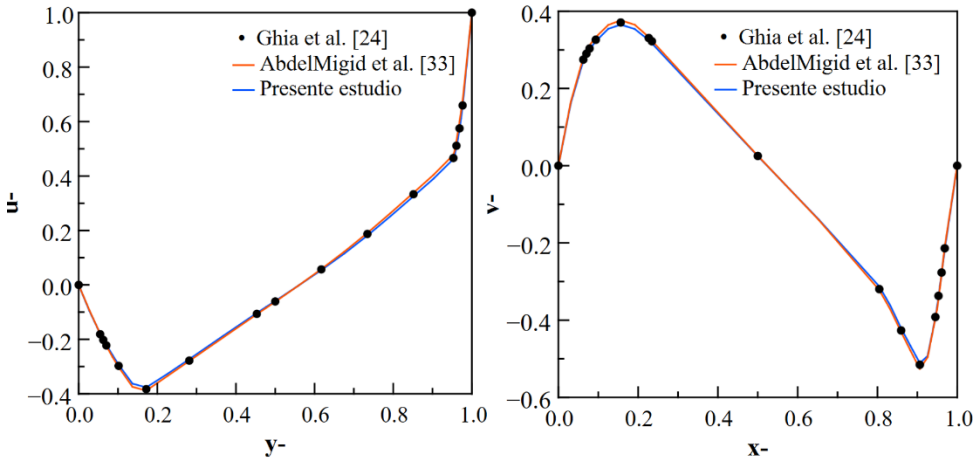


Figura 14. Verificación del problema de flujo laminar: a) componente u y b) componente v.

Finalmente, en la **figura 15** se ilustra el campo de la magnitud de velocidad U del presente código empleando el solucionador LGS-ADI. Los hallazgos se contrastan con los obtenidos por Patel et al. [34], quienes desarrollaron su código numérico en Matlab empleando mallas desplazadas y el algoritmo SIMPLE. Los resultados reproducen las estructuras características del flujo a Re= 1000, como la formación de dos vórtices en las esquinas inferiores de la cavidad.

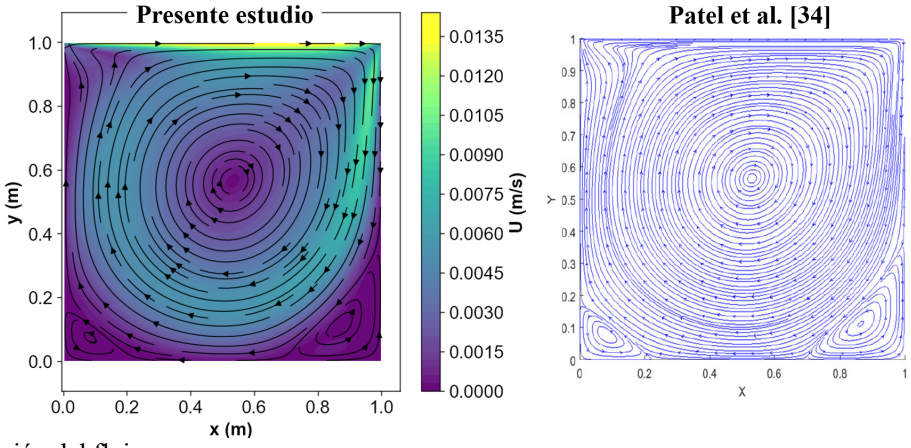


Figura 15. Visualización del flujo.

En este trabajo se demostró que el algoritmo LGS-ADI de direcciones alternantes es superior al solucionador clásico de Gauss-Seidel. Como futuros trabajos se recomienda la exploración de otras librerías disponibles en Python con el fin de optimizar el desempeño computacional de los códigos desarrollados, como numba [35]. El código también permite la incorporación de bibliotecas de álgebra lineal dispersa, como SciPy [36] y PyAMG [37], que posibilitan el uso de solucionadores iterativos avanzados y técnicas multimalla para la ecuación de presión.

ISSN: 2594-1925

6. Conclusiones

En este trabajo se implementó y se verificó un código CFD bidimensional en el lenguaje de programación Python para la solución numérica de problemas difusivos y flujos forzados laminares. Las ecuaciones se discretizaron mediante la metodología de volúmenes finitos, se emplearon mallas desplazadas para las componentes de velocidades y el acople con la presión se efectuó mediante el algoritmo SIMPLE. Los términos difusivos se discretizaron a través de un esquema centrado y para los convectivos la estrategia de solución consistió en estabilizar la simulación con un esquema de primer orden y posteriormente configurar un esquema híbrido con el fin de reducir la difusión numérica. Se implementaron dos solucionadores, Gauss-Seidel y LGS-ADI, evaluando el desempeño computacional de cada uno. Se extraen las siguientes conclusiones:

- El código CFD se programó adecuadamente debido a que al verificar el problema difusivo con la solución analítica el error relativo porcentual máximo fue de 0.16%. De igual manera, al verificar el problema de flujo laminar con la solución de referencia, el error relativo porcentual máximo fue de 4.667% en el componente horizontal y de 2.191% en el componente vertical.

- El algoritmo LGS-ADI se comporta adecuadamente en problemas difusivos lineales, logrando una reducción en el tiempo computacional equivalente a 61.92%, pero en problemas de flujo laminar el tiempo disminuye hasta un mínimo de 11% debido a las no linealidades involucradas. Aun así, en ambos casos, la reducción es significativa.

- El solucionador LGS-ADI permitió resolver el problema de flujo laminar a un factor de subrelajación mayor, mientras que con el solucionador Gauss-Seidel se registró la divergencia de la solución numérica.

- En el problema de flujo laminar, la reducción de tiempo máxima fue de 19 min, y se obtuvo con el solucionador LGS-ADI, el esquema híbrido y residuales del orden de $1x10^{-8}$.

- Se recomienda el algoritmo LGS-ADI para problemas difusivos y de flujos forzados laminares en geometrías cuadradas puesto que presentó un desempeño superior al solucionador clásico Gauss-Seidel.

Este trabajo contribuye al desarrollo de código propio para el modelado numérico mediante CFD. Como trabajos futuros, se plantea la programación de mallas no uniformes y la extensión del código a problemas de convección natural y acoplamientos sólido-fluido. Este código CFD en Python se empleará para el diseño y evaluación de estrategias pasivas que permitan reducir las ganancias de calor en edificaciones mexicanas, tales como ventanas de vidrios múltiples y sistemas de ventilación natural. Finalmente, se recomienda de igual manera, explorar otras librerías disponibles en Python como Numba, SciPy y PyAMG.

7. Agradecimientos

Edgard Elohim Canche Cauich (CVU: 2043558) agradece a la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) por la beca otorgada para la realización de los estudios de maestría.

ISSN: 2594-1925

8. Reconocimiento de autoría

Edgar Elohim Canche-Cauich: Conceptualización; Metodología; Redacción borrador original. *Felipe Noh-Pat:* Metodología; Validación; Redacción revisión y edición. *Miguel Ángel Gijón-Rivera*: Metodología; Redacción y revisión. *Manuel Jesús Rodríguez-Pérez*: Redacción; Revisión y edición. *Mauricio Iván Huchin-Miss:* Redacción; Revisión y edición.

Nomenclatura

Abreviaciones

2D	Bidimensional
CFD	Dinámica de Fluidos Computacional
LGS-ADI	Método de línea de Gauss-Seidel de direcciones alternantes
SIMPLE	Método semi-implícito para el acople de presión

Letras latinas

b	Término fuente
D	Conductancia de la difusión
F	Flujo másico convectivo
k	Conductividad térmica (W/mK)
P	Presión (Pa)
Q	Relación de recurrencia
R	Relación de recurrencia
T	Temperatura (K)
U	Magnitud de velocidad (m/s)
u	Componente horizontal de velocidad (m/s)
v	Componente vertical de velocidad (m/s)
u-	Componente horizontal adimensional de velocidad
v-	Componente vertical adimensional de velocidad
x	Coordenada horizontal
y	Coordenada vertical

Letras griegas

α	Factor de subrelajación
ϕ	Incógnita (T, u, v)
μ	Viscosidad dinámica (kg/ms)
ρ	Densidad (kg/m^3)

Subíndices

P	Principal
E, e	Este
W, w	Oeste
N, n	Norte
S, s	Sur

ISSN: 2594-1925

Referencias

[1] J. Xamán and M. Gijón-Rivera, *Dinámica de Fluidos Computacional para Ingenieros*. Ciudad de México, 2016.
[2] The OpenFOAM Foundation Ltd., "OpenFOAM." Accessed: Jun. 19, 2025. [Online]. Available: https://openfoam.org/
[3] G. Materano, C. Araujo, and A. A. V. Ochoa, "A new OpenFOAM proposal for the solution of diffusion problems," *Thermal Science and Engineering Progress*, vol. 25, p. 100982, Oct. 2021, doi: 10.1016/j.tsep.2021.100982
[4] T. Välikangas, *Conjugate heat transfer in OpenFOAM. In Proceedings of CFD with OpenSource software, 2016, edited by Nilsson. H*. 2016. Accessed: May 31, 2025. [Online]. Available: https://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2016/TuroValikangas/Report_Turo.pdf
[5] G. Vijaya Kumar, M. Kampili, S. Kelm, K. Arul Prakash, and H.-J. Allelein, "CFD modelling of buoyancy driven flows in enclosures with relevance to nuclear reactor safety," *Nuclear Engineering and Design*, vol. 365, p. 110682, Aug. 2020, doi: 10.1016/j.nucengdes.2020.110682
[6] J. S. Piña *et al.*, "Development of a boundary-coupled CFD model for collimated-diffusive radiation," *Int. J. Heat Mass Transf.*, vol. 248, p. 127170, Sep. 2025, doi: 10.1016/j.ijheatmasstransfer.2025.127170
[7] R. Manceau, "Industrial codes for CFD. Master," Poitiers, France, 2026.
[8] J. Carlier and M. V. Papalexandris, "An efficient tracking method of evaporative and flat free surfaces for turbulent convection," *Comput. Fluids*, vol. 257, p. 105882, May 2023, doi: 10.1016/j.compfluid.2023.105882
[9] A. Labihi *et al.*, "Effect of phase change material wall on natural convection heat transfer inside an air filled enclosure," *Appl. Therm. Eng.*, vol. 126, 2017, doi: 10.1016/j.applthermaleng.2017.07.112
[10] J. Ríos-Arriola, E. Gómez-Arias, I. Zavala-Guillén, N. Velázquez-Limón, G. Bojórquez-Morales, and J. E. López-Velázquez, "Numerical modeling of soil temperature variation under an extreme desert climate," *Geothermics*, vol. 112, 2023, doi: 10.1016/j.geothermics.2023.102731
[11] S. Sena, R. Goyal, and S. K. Tyagi, "Numerical modelling for improved prediction of ground temperature in seasonal snow-cover regions," *Cold Reg. Sci. Technol.*, vol. 239, p. 104578, Nov. 2025, doi: 10.1016/j.coldregions.2025.104578
[12] Secretaría de Medio Ambiente y Recursos Naturales, *Plan de Acción en Enfriamiento, México. Refrigerantes con Bajo Potencial de Calentamiento Global y Eficiencia Energética en Equipos de Refrigeración y Aire Acondicionado*. México. Semarnat, 2022.
[13] Y. Olazo-Gómez, I. Hernández-López, I. Zavala-Guillén, I. Hernández-Pérez, and D. García-Pérez, "Numerical study of a cool roof and double-glazing window coupled to an air-cavity under a tropical Mexican climate", *Case Studies in Thermal Engineering*, vol. 73, p. 106372, Sep. 2025, doi: 10.1016/j.csite.2025.106372
[14] J. Uriarte, *Análisis Térmico de una Habitación con Techo de Geometría Irregular y Cubiertas Reflectivas*. Cuernavaca, México, 2022. Accessed: Mar. 29, 2026. [Online]. Available: https://rinacional.tecnm.mx/bitstream/TecNM/4141/4/DM_Javier_Uriarte_Flores_2022.pdf

ISSN: 2594-1925

[15] J. Xamán, A. Rodriguez-Ake, I. Zavala-Guillén, I. Hernández-Pérez, J. Arce, and D. Sauceda, "Thermal performance analysis of a roof with a PCM-layer under Mexican weather conditions," *Renew. Energy*, vol. 149, pp. 773–785, Apr. 2020, doi: 10.1016/j.renene.2019.12.084
[16] J. Xamán, C. Jiménez-Xamán, G. Álvarez, I. Zavala-Guillén, I. Hernández-Pérez, and J. O. Aguilar, "Thermal performance of a double pane window with a solar control coating for warm climate of Mexico," *Appl. Therm. Eng.*, vol. 106, pp. 257–265, Aug. 2016, doi: 10.1016/j.applthermaleng.2016.06.011
[17] I. Zavala-Guillén, D. Barrera-Román, F. Noh-Pat, M. Sidón, D. García-Pérez, and A. Rodriguez-Ake, "Thermal analysis of multi-layered glazed window under Mexican climate," *Energy Build.*, vol. 329, p. 115259, Feb. 2025, doi: 10.1016/j.enbuild.2024.115259
[18] F. Noh-Pat, M. Gijón-Rivera, C. I. Rivera-Solorio, and M. Jiménez-Xamán, "Numerical analysis of the thermal performance of a lightweight insulating roof integrated with a phase change material," *Case Studies in Thermal Engineering*, vol. 77, p. 107634, Jan. 2026, doi: 10.1016/j.csite.2025.107634
[19] N. Rodrigues Marques Sakiyama, J. Frick, T. Bejat, and H. Garrecht, "Using CFD to Evaluate Natural Ventilation through a 3D Parametric Modeling Approach," *Energies (Basel).*, vol. 14, no. 8, p. 2197, Apr. 2021, doi: 10.3390/en14082197
[20] B. LeMesurier, *Introduction to Numerical Methods and Analysis with Python*. 2024. https://lemesurierb.people.charleston.edu/introduction-to-numerical-methods-and-analysis-python.pdf. Consultado abril de 2026.
[21] L. Barba and G. Forsyth, "CFD Python: the 12 steps to Navier-Stokes equations," *Journal of Open-Source Education*, vol. 1, no. 9, p. 21, Nov. 2018, doi: 10.21105/jose.00021
[22] S. Pawar and O. San, "CFD Julia: A Learning Module Structuring an Introductory Course on Computational Fluid Dynamics," *Fluids*, vol. 4, no. 3, p. 159, Aug. 2019, doi: 10.3390/fluids4030159
[23] B. Kumar, W. Liu, and W. Hao, "GPU Accelerated Computational Methods using Python and CUDA," Sweden, 2025.
[24] U. Ghia, K. N. Ghia, and C. T. Shin, "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method," *J. Comput. Phys.*, vol. 48, no. 3, pp. 387–411, Dec. 1982, doi: 10.1016/0021-9991(82)90058-4
[25] T. Bergman, A. Lavine, F. Incropera, and D. DeWitt, *Introduction to Heat Transfer*, 6th ed. New York, United States of America, 2011.
[26] C. J. Greenshields and H. G. Weller, *Notes on Computational Fluid Dynamics: General Principles*. Reading, UK: CFD Direct Ltd, 2022.
[27] J. Anderson, *Computational Fluid Dynamics*. New York: McGraw-Hill, 1995.
[28] M. Sahin and R. G. Owens, "A novel fully implicit finite volume method applied to the lid-driven cavity problem—Part I: High Reynolds number flow calculations," *Int. J. Numer. Methods Fluids*, vol. 42, no. 1, pp. 57–77, May 2003, doi: 10.1002/fld.442
[29] L. Davidson, "Lecture Notes on Computational Fluid Dynamics of Turbulent Flow, chapter 6." Accessed: Mar. 26, 2026. [Online]. Available: https://www.cfd-sweden.se/lada/comp_fluid_dynamics/postscript_files/chapter_6.pdf
[30] WolfDynamics, "Tips and tricks in OpenFOAM." Accessed: Mar. 29, 2026. [Online]. Available: https://www.wolfdynamics.com/wiki/tipsandtricks.pdf
[31] OpenCFD Ltd, "Solution and algorithm control," OpenFOAM User Guide, sec. 6.3. Accessed: Mar. 29, 2026. [Online]. Available: https://www.openfoam.com/documentation/user-guide/6-solving/6.3-solution-and-algorithm-control

ISSN: 2594-1925

[32] E. E. Canche-Cauich, F. Noh-Pat, M. A. Jiménez-Torres, F. R. Lezama-Zárraga, and B. Cortazar-Miranda, "Efecto de la conducción de calor en una ventana de vidrio doble. Memorias XXI Congreso Internacional, XXVII Congreso Nacional de Ciencias Ambientales y VIII Iberoamericano de Física y Química Ambiental," *Revista Internacional de Contaminación Ambiental*, vol. 40, Oct. 2024, doi: 10.20937/RICA.2024.40.ANCA

[33] T. A. AbdelMigid, K. M. Saqr, M. A. Kotb, and A. A. Aboelfarag, "Revisiting the lid-driven cavity flow problem: Review and new steady state benchmarking results using GPU accelerated code," *Alexandria Engineering Journal*, vol. 56, no. 1, pp. 123–135, Mar. 2017, doi: 10.1016/j.aej.2016.09.013

[34] M. R. Patel, J. U. Pandya, and V. K. Patel, "Numerical Analysis of Fluid Flow Behaviour in Four-Sided Square Lid-Driven Cavity Using the Finite Volume Technique," *Int. J. Appl. Comput. Math.*, vol. 8, no. 4, p. 153, Aug. 2022, doi: 10.1007/s40819-022-01353-x

[35] Numba, "Numba: A High-Performance Python Compiler." Accessed: Mar. 29, 2026. [Online]. Available: https://numba.pydata.org/

[36] P. Virtanen *et al.*, "Author Correction: SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nat. Methods*, vol. 17, no. 3, pp. 352–352, Mar. 2020, doi: 10.1038/s41592-020-0772-5

[37] N. Bell, L. N. Olson, and J. Schroder, "PyAMG: Algebraic Multigrid Solvers in Python," *J. Open Source Softw.*, vol. 7, no. 72, p. 4142, Apr. 2022, doi: 10.21105/joss.04142

ISSN: 2594-1925