UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Mecânica

CLAUDIO ALFREDO TORRES RODRIGUEZ

Desarrollo de código bidimensional transitorio para el estudio de la transferencia de calor en flujo laminar e incompresible aplicado a convección natural con alta variación de relaciones de aspecto

Desenvolvimento de código bidimensional transiente para estudo da transferência de calor em escoamento laminar e incompressível aplicado a convecção natural com alta variação das razões de aspecto

CAMPINAS
2022

CLAUDIO ALFREDO TORRES RODRIGUEZ

Desarrollo de código bidimensional transitorio para el estudio de la transferencia de calor en flujo laminar e incompresible aplicado a convección natural con alta variación de relaciones de aspecto

Desenvolvimento de código bidimensional transiente para estudo da transferência de calor em escoamento laminar e incompressível aplicado a convecção natural com alta variação das razões de aspecto

Disertación de Maestría presentada en la Facultad de Ingeniería Mecánica de la Universidad Estadual de Campinas como parte de los requisitos para la obtención del título de Maestría en Ingeniería Mecánica, en el área de Térmica y Fluidos.

Dissertação de Mestrado apresentada à Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Mecânica, na Área de Térmica e Fluidos.

Orientador: Prof. Dr. Rogério Gonçalves dos Santos.

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO CLAUDIO ALFREDO TORRES RODRIGUEZ
E ORIENTADA PELO PROF. DR ROGÉRIO GONÇALVES DOS SANTOS.

.....
PROF. DR ROGÉRIO GONÇALVES DOS SANTOS

CAMPINAS
2022

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

Rodríguez, Claudio Alfredo Torres, 1986-
R618d Desarrollo de código bidimensional transitorio para estudio de transferencia de calor en flujo laminar e incompresible aplicado a convección natural con alta variación de relaciones de aspecto / Claudio Alfredo Torres Rodríguez. – Campinas, SP : [s.n.], 2022.

Orientador: Rogério Gonçalves dos Santos.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Fluidodinâmica computacional. 2. C++ (Linguagem de programação de computador). 3. Programação linear. I. Santos, Rogério Gonçalves dos, 1978-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Desenvolvimento de código bidimensional transiente para estudo da transferência de calor em escoamento laminar e incompressível aplicado a convecção natural com alta variação de razões de aspeto
Palavras-chave em inglês:
Computational fluid dynamics
C++ (Computer program language)
Linear Programming
Área de concentração: Térmica e Fluídos
Titulação: Mestre em Engenharia Mecânica
Banca examinadora:
Rogério Gonçalves dos Santos [Orientador]
Kamal Abdel Radi Ismail
Carlos Teofilo Salinas Sedano
Data de defesa: 15-03-2022
Programa de Pós-Graduação: Engenharia Mecânica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: 0000-0002-9850-1377.
- Currículo Lattes do autor: http://lattes.cnpq.br/5954442422808173

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

DISSERTAÇÃO DE MESTRADO ACADÊMICO

Desarrollo de código bidimensional transitorio para el estudio de la transferencia de calor en flujo laminar e incompresible aplicado a convección natural con alta variación de relaciones de aspecto

Desenvolvimento de código bidimensional transiente para estudo da transferência de calor em escoamento laminar e incompressível aplicado a convecção natural com alta variação das razões de aspecto

Autor: **Claudio Alfredo Torres Rodriguez**

Orientador: Prof. Dr. Rogério Gonçalves dos Santos
Coorientador:

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

Prof. Dr. Rogerio Gonçalves dos Santos
FEM/ UNICAMP

Prof. Dr. Kamal Abdel Radi Ismail
FEM/ UNICAMP

Prof. Dr. Carlos Teofilo Salinas Sedano
Universidade Santa Cecilia Santos

A Ata de Defesa com as respectivas assinaturas dos membros encontra-se no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 15 de março de 2022.

Resumen

En el presente trabajo se desarrolla un código numérico en lenguaje C++, que consigue resolver las ecuaciones de Navier-Stoke, balance de masa y balance de energía de forma bidimensional en estado transitorio y estacionario, para fluidos incompresibles en régimen laminar. Implementando el algoritmo SIMPLE para acoplar satisfactoriamente los campos de velocidad y de presión. Adoptando para esto la hipótesis de Boussinesq de manera a modelar los efectos de la variación de la densidad debido a la temperatura para los problemas de convección natural. Esté código fue utilizado para analizar numéricamente la relación de aspecto de la convección natural en una cavidad rectangular calentada diferencialmente en dos paredes verticales. Este método de transferencia de calor tiene un interés especial en las ciencias térmicas, mecánica de fluidos y en una amplia gama de aplicaciones de ingeniería. Para estas simulaciones numéricas, las temperaturas de las paredes fueron asumidas como constantes. El objeto fue el de determinar, los efectos de variar la relación de aspecto de la cavidad H/L (Altura/Longitud) y los efectos de los distintos esquemas numéricos, sobre el comportamiento del flujo de fluido en la transferencia de calor dentro de la cavidad. Las simulaciones fueron realizadas para relación de especto 0,25; 0,5; 1,0; 2,0 e 4,0 y números de Rayleigh 10^3, 10^4, 10^5 y 10^6 usando los esquemas numéricos UpWind, Ley de potencia y Esquema Central. Los resultados muestran que la relación de aspecto tiene menor influencia en los campos de velocidad y temperatura que el número de Rayleigh. Finalmente, para analizar la escalabilidad del código, se estudia la implementación de la herramienta PETSc (Portable, Extensible Toolkit for Scientific Computation) y se realiza una discusión respecto a su utilidad.

Palabras Clave: CFD, CODIGO C++, SIMPLE, PETSc, RELACIÓN DE ASPECTO

Resumo

No presente trabalho, desenvolveu-se um código numérico em linguagem C ++, que resolve a equação na forma bi-dimensional de Navier-Stoke, de balanço de massa e energia no estado estacionário e transitório, para fluidos incompressíveis em regime laminar. Implementando um algoritmo SIMPLE para acoplar velocidade e pressão. A hipótese de Boussinesq foi adotada para modelar os efeitos de variação de densidade devido a temperatura em problemas de convecção natural. Usando este novo código, a relação de aspecto na convecção natural foi analisada numericamente em uma cavidade retangular aquecida em um dos lados e resfriada no lado oposto. Este método de transferência de calor tem um interesse especial em ciências térmicas, mecânica de fluidos e uma ampla gama de aplicações em engenharia. Nas simulações numéricas, as temperaturas das diferentes paredes foram assumidas como constantes. O objetivo foi determinar os efeitos da relação de aspecto H / L (Altura / Comprimento) e do esquema numérico sobre o comportamento do fluxo e a transferência de calor na cavidade. As simulações foram realizadas para razões de aspecto 0,25; 0,5; 1,0; 2,0 e 4,0 e números de Rayleigh 10^3, 10^4, 10^5 e 10^6 usando os esquemas UpWind, Lei de Potência e Central. Os resultados mostram que a relação de aspecto tem menor influência na temperatura e no campo de velocidade do que o número de Rayleigh. Finalmente, para analisar a escalabilidade, um novo código paralelo foi implementado usando a ferramenta PETSc (Portable, Extensible Toolkit for Scientific Computation) e uma avaliação sobre o uso desta foi realizada.

Palavras Chave: CFD, CODIGO C++, SIMPLE, PETSc, RELAÇÃO DE ASPECTO

Abstract

In the present work, was developed a numerical code homemade on C++ language, which solve in the bi-dimensional form the Navier-Stoke equation, the mass and energy balance on steady state or transient, for non-compressible fluid in laminar regime. Implementing a SIMPLE algorithm to couple the differential equations of velocity and pressure. The Boussinesq hypothesis was adopted to model the effects of density variation due to temperature in natural convection problems. Using this new code the aspect ratio in natural convection was numerically analyzed in a rectangular cavity heated on one of the sides and cooled on the opposite side. This spontaneous method of heat transfer has an especial interest in thermal sciences, fluid mechanics and a wide range of applications in engineering. In the numerical simulations temperatures of the different walls were assumed to be constant. The objective was to determine the effects of the aspect ratio H/L (Height/Length) and the numerical scheme on flow behavior and heat transfer in the cavity. Simulations were performed for aspect ratios 0.25, 0.5, 1.0, 2.0 and 4.0 and Rayleigh number 10^3, 10^4, 10^5 and 10^6 using the UpWind, Power Law and Central schemes. Results show that the aspect ratio has lower influence in the temperature and streamlines fields than Rayleigh number. Finally to analyze the scalability a new parallel code was implemented using PETSc (Portable, Extensible Toolkit for Scientific Computation) and an evaluation about this tool was done.

Key Word: CFD, CODE C++, SIMPLE, PETSc, ASPECT RATIO

Lista de Ilustraciones

Lista de Tablas

Lista de Abreviaturas y Siglas

C_P	calor especifico a presión constante
v	componente vertical de la velocidad
u	componente horizontal de la velocidad
λ	conductividad térmica
x	coordenada cartesiana en la dirección horizontal
y	coordenada cartesiana en la dirección vertical
ρ	densidad
δ	distancia entre nodos de una malla
$\dot{F}$	flujo convectivo
$\dot{D}$	flujo difusivo
Pe	número de Peclet
Ra	Número de Rayleigh
P	presión
ϕ	propriedad intensiva de la ecuación general de transporte
T	temperatura
Γ	termino difusivo de la ecuación general de transporte
S	termino fuente de la ecuación general de transporte
t	tiempo
μ	viscosidad dinámica

Índice

1 INTRODUCIÓN

La convección natural es un fenómeno físico donde un fluido se mantiene en movimiento debido a la diferencia de densidad causada por un gradiente de temperatura y bajo acción de la graverdad. Es además un mecanismo de transferencia de calor.

Para hacer el análisis numérico del fenómeno físico de la convección natural frecuentemente se utilizan técnicas de Dinámica de fluidos computacional (CFD). Es un área de estudio que involucra a la mecánica de fluidos, a la termodinámica, y que utiliza métodos numéricos para lograr simulaciones computacionales de fenómenos físicos. El objeto de estas simulaciones es el poder predecir el comportamiento de los casos de estudio sin necesariamente ser llevados aún a la práctica experimental, logrando la obtención de datos a bajos recursos, sea por un deseo de economía o por una inviabilidad física de realizar el experimento práctico. Algunas de las disciplinas donde suelen implementarse más comúnmente es en la Aerodinámica, la Combustión, Aeronáutica y la Mecánica de Fluidos en general.

Los métodos aplicados para la simulación mediante la fluidodinámica computacional son en líneas generales la resolución numérica de las ecuaciones diferenciales parciales de la conservación de la cantidad de movimiento y conservación de la masa para los casos más simples. Hoy día el mundo de la ingeniería ya no busca los casos más sencillos (o más simplificado) necesariamente, sino los más realistas por lo que generalmente en los casos más complejos se resuelven también otras ecuaciones diferenciales como, por ejemplo, la ecuación del balance de energía y de especies químicas.

La convección natural es un fenómeno físico frecuente, producido por la diferencia de temperatura en el cuerpo del fluido, encontrándose una parte a mayor temperatura que otra, resultando así que, si tomamos como cuerpo de control una pequeña porción de fluido que sea parte del universo de fluido total, siendo que ese cuerpo de control se encuentra a mayor temperatura, tiende a dilatarse por el aumento de temperatura, eso quiere decir que ocupa un mayor volumen manteniendo siempre la misma masa, esto solo se traduce en una disminución de la densidad, por consiguiente como es conocido en un volumen donde existen dos fluidos

de distintas propiedades, el de menor densidad tenderá a subir y el de mayor densidad tenderá a bajar, por acción del campo gravitatorio.

Así es como la porción de fluido a mayor temperatura tiende a subir y la porción a menor temperatura tiende a bajar. Generando así un movimiento del fluido de manera natural y sin necesidad de ser impulsado por algún otro elemento físico móvil como las aspas de un ventilador o las turbinas de un motor, sino simplemente por acción de la diferencia de temperatura.

También durante este movimiento de fluido existe una transferencia de calor en todo el fluido ya que, el fluido siempre tiende a llegar al equilibrio térmico, así pues, si existe una fuente de calor, este fluido siempre está en movimiento.

1.1 Objetivo General

El objetivo general de trabajo es el lograr el desarrollo de un código computacional numérico que sea bidimensional y capaz de resolver las ecuaciones de Navier-Stokes, conservación de la masa y conservación de la energía en régimen laminar, estacionario y transitorio, para flujos incompresibles en malla estructurada. Y que este código sea abierto para alumnos de graduación y de posgraduación, y lo suficientemente ligero computacionalmente para ser ejecutado en cualquier dispositivo portátil y económicamente accesible.

1.2 Objetivos Específicos

El objetivo específico inicial es el desarrollo de un código en el lenguaje C++ de manera que pueda ejecutarse correctamente en computadores personales sin excesivos recursos y que de igual manera logre resolver problemas de la mecánica de fluidos y transferencia de calor, con el objeto de que cualquier alumno pueda utilizarlo sin restricción y adaptarlo a su utilidad especifica y caso de estudio.

El segundo objetivo específico es lograr darle una utilidad al código desarrollado mediante la producción bibliográfica, a través de un artículo de revista o de congreso. Demostrar así su utilidad y el impacto del trabajo.

El tercer objetivo específico es analizar la factibilidad de escalar el código con un procesamiento de datos en paralelo utilizando una herramienta en base a MPI conocida como PETSc.

1.3 Presentación del trabajo

En este trabajo se presenta la metodología del desarrollo de un código o software de fluidodinámica computacional CFD escrito en el lenguaje de programación C++ capaz de funcionar en computares sencillos sin muchos requerimientos computacionales. El cual fue utilizado para el estudio del fenómeno conocido como convección natural en cavidades cerradas y cómo afecta la variación del aspecto de la cavidad al comportamiento del fluido en su interior para el caso de fluido incompresible y paredes verticales calentadas diferencialmente.

También se expondrá en este trabajo el método o algoritmo utilizado para el desarrollo del código, las ecuaciones que gobiernan los fenómenos de estudio y se detallarán las aproximaciones tomadas.

Este código, además, puede resolver otros problemas fisícos para geometrías relativamente sencillas utilizando una malla estructurada, fluidos incompresibles y régimen laminar, sea transitorio o permanente.

1.4 Un poco de historia

El físico y matemático inglés Lewis Fry Richardson en 1922 publicó un trabajo en la prensa de la Universidad de Cambridge, conocido en español como “Predicción del clima por procesos numéricos”, convirtiéndose así en el pionero del análisis numérico del clima y cuyo trabajo es señalado históricamente como el primer intento de predicción de flujo mediante métodos numéricos utilizando un método de diferencias finitas. Aunque sus predicciones fueron erradas, pocos años más tarde diferentes autores comenzaron a utilizar sus métodos numéricos como base para perfeccionar la técnica.

Cerca de una década más tarde, comienzan los primeros intentos de resolver ecuaciones linealizadas en dos dimensiones utilizando como método simplificaciones y transformaciones para el caso del flujo alrededor de un cilindro, según Milne-Thomson en 1973.

El desarrollo de las primeras técnicas de CFD llegan a finales de la década del 60 con Francis H. Harlow en 1955, quien presentó un método conocido como PARTICLE IN CELL, este método consiste en seguir a una partícula del fluido en un campo Lagrangiano, mientras los valores físicos como velocidad o densidad se calculan desde puntos fijos de la malla en simultáneo.

El siguiente método es el Fromm y Harlow. En 1963 el VORTICITY STREAM FUNCTION METHOD, conocido como vorticidad-corriente, fue uno los primeros métodos en tratar de resolver sistema de ecuaciones en derivadas parciales, en el cual a cada una de las ecuaciones de momento se aplican derivadas cruzadas, logrando así las ecuaciones para dos variables, la corriente y la vorticidad.

Poco tiempo después Hess & Smith en 1967 publican un trabajo con el que proponen un método para discretizar la superficie de una geometría arbitraria en paneles, creando así lo que posteriormente se conoció como método de paneles. La evolución de este método se aplicó principalmente a la simulación de las superficies de las geometrías de navíos y aeronaves. Se presume que este fue el primer modelo tridimensional publicado.

Los investigadores de Boeing Earll Murman y Julian Cole en 1970 fueron los primeros en publicar un método para resolver ecuaciones de flujo potencial de manera no lineal ya que el método de paneles no les permitía calcular la velocidad transitoria y otras alinealidades.

En la década de 1970 dos personajes importantemente históricos para este trabajo, el profesor Dudley Brian Spalding (9 January 1923 – 27 November 2016) de la universidad imperial de Londres y el entonces alumno suyo, hoy reconocido profesor Suhas Patankar, desarrollaron un algoritmo llamado SIMPLE (Semi Implicit Method for Pressure Linked Equations) que por sus siglas en inglés significa método semi implícito para ecuaciones ligadas a la presión. Este método es utilizado hasta el día de hoy para resolver problemas de fluidos y transferencia de calor, por su método iterativo para el acople de la presión. De hecho, este algoritmo será utilizado para este trabajo. Este método lo describe el profesor Patankar en 1980.

Posteriormente Jameson, Schmidt y Turkel 1981, con el objeto de mejorar las soluciones numéricas para flujos transitorios resolvieron las ecuaciones de Euler, utilizando el método de volúmenes finitos en tres dimensiones aplicando Runge-Kutta como esquema numérico y pasos de tiempo. Jameson creó un código tridimensional transitorio llamado FLO57.

En la actualidad existen varios softwares comerciales para la simulación de CFD. Algunos de ellos son citados con los trabajos de revisión bibliográfica. Destacándose por la cantidad de investigadores e ingenieros que lo utilizan se encuentra el ANSYS FLUENT. ANSYS es una compañía fundada en 1970 para desarrollar comercialmente softwares direccionados al mundo de la ingeniería.

2 REVISIÓN DE LA LITERATURA Y ESTADO DEL ARTE

Para iniciar la investigación de la literatura fue necesario dividir los fenómenos de estudio y generar una forma de clasificación, de manera a poder situar cada trabajo o artículo en grupos comparables entre sí. Así son separados algunos de los trabajos estudiados en ocho grupos, lo cual no quiere decir que puedan existir más grupos u otras maneras de clasificarlos.

Esto también tiene que ver a la manera en la que se describe el presente trabajo, cuando se trata de un código que resuelve problemas de flujo laminar en 2D, por ejemplo, o cuando, cuando se define la condición del fluido como incompresible o se define si un caso es estacionario o no.

2.1 Clasificación por tipo de cavidad

Una de manera de discriminar es la de clasificar en tipos de cavidades, en la cual se puede evidenciar dos tipos que me resultaron importantes, las cavidades cerradas y las cavidades abiertas.

Por un lado, las cavidades abiertas se describen como aquellas que permiten al flujo atravesar las fronteras del dominio o cavidad que decidamos tomar. En cambio, las cavidades cerradas son aquellas donde el fluido no abandona la cavidad, no existe flujo de fluido para un medio externo, sino que es contenida por las paredes de la cavidad que tomamos como volumen de control.

2.1.1 Cavidades cerradas

El trabajo de Y. Ji en 2014, aunque trata de ambos tipos de cavidades en este apartado le daré un enfoque a su trabajo con cavidades, pues Ji nos presenta dos cavidades cerradas en donde simula numéricamente a través de la fluidodinámica computacional el comportamiento el aire confinado y la convección natural existente debido al calentamiento diferencial de dos paredes opuestas y dos paredes adiabáticas, variando la razón de aspecto entre una y otra. La simulación se llevó a cabo utilizando un código comercial el Ansys CFX.

Entre sus conclusiones Ji, consigue demostrar la efectividad de cuatro modelos de turbulencia capaces de simular el comportamiento de la convección natural del aire.

2.1.2 Cavidades Abiertas

El estudio de las cavidades abiertas mediante CFD ha tomado bastante protagonismo en las últimas décadas para el mundo de la ingeniería, principalmente para la ingeniería civil y la arquitectura. Más aun con la moda de las fachadas doble para protegerse del calor, como nos muestran Pérez, Patiño, Amparo, & Jiménez en 2013 quienes lograron realizar un modelado matemático simplificado de una fachada doble para su simulación en CFD, comparando el flujo por convección natural y por convección forzada, donde se puede apreciar como el flujo de aire entre las fachadas favorece notoriamente a la refrigeración del sitio y además concluyeron que la ventilación forzada claramente favorece el intercambio de energía al aumentar la velocidad del fluido en cuestión (aire) y que para llevar el estudio de una manera más exhaustiva es necesario mejorar el modelo matemático para un modelo más complejo.

Otro ejemplo de estudio de cavidad abierta es el estudio del viento en terrenos abiertos o con obstáculos, un trabajo que llamo mi atención fue el de García, Boulanger, Duque, & Giraldo en 2008, científicos de la Universidad de Medellín en Colombia, quienes estudiaron vientos convectivos naturales debido a la temperatura del terreno mediante simulaciones en CFD, buscando determinar el efecto de las estructuras de concreto en la temperatura atmosférica. Pudiendo concluir que pese a los años de estudio dedicado a estos fenómenos aún existen problemas para abordar y puntos para mejorar en las simulaciones, sobre todo con la generación de malla y con la resolución numérica. Pese a lograr una simulación para comportamiento del viento generado por convección natural del aire, no fue incluido en el estudio el viento con flujo transversal. Por lo cual no concluyen el efecto de este, pero los autores resaltan la manera en la que la incluirán para trabajos futuros.

2.2 Clasificación por tipo de régimen de flujo

Hablar del régimen del flujo, es describir la manera en la que el fluido se mueve, esta puede ser ordenada o caótica. En general a la forma ordenada se la define como flujo laminar

donde el fluido sigue cierto orden distribuyéndose en capaz que no se mezclan y cada partícula de fluido sigue una trayectoria suave, a la cual se la conoce como línea de corriente. Las condiciones para la existencia de este tipo de flujo, se basa en la geometría, velocidad y viscosidad y puede ser estimada su existencia según un numero adimensional. Reynolds 1883.

Por el contrario, el flujo turbulento es un flujo caótico y desordenado, cuya condición de existencia también depende de la velocidad, la viscosidad y la geometría. Por ejemplo, para tuberías de sección circular, Reynolds estableció un numero adimensional, que actualmente llamamos número de Reynolds que permite determinar si el flujo será laminar o turbulento, el numero depende de la velocidad, siendo directamente proporcional, del diámetro del tubo, al cual también es proporcional y a la viscosidad a la cual es inversamente proporcional.

2.2.1 Régimen Laminar

Uno de los usos del flujo laminar en la ciencia, es el de la velocidad laminar de combustión, utilizado para caracterizar el comportamiento de los combustibles. Así como lo presentan Henriksen y otros 2019 quienes buscan alternativas al combustible de jets, tratando generar un combustible de jets sintético cuya producción pueda ser viable y sustentable para remplazar los actuales combustibles fósiles. Y para ellos analizan y describen el comportamiento de distintos sustitutos, a través de la simulación en CFD.

2.2.2 Régimen turbulento

Una aplicación de fluidodinámica computacional de vanguardia para la industria es el trabajo de Guevara & Belalcázar en 2018, quienes decidieron tomar un innovador dispositivo conocido como válvulas de micro-desviasión, utilizadas para generar microburbujas con el objeto de mejorar significativamente ciertos procesos industriales. Los autores refieren que, si bien este instrumento industrial ya cuenta con numerosos estudios numéricos realizados en CFD, ninguno de los anteriores utilizo un modelo de turbulencia detallado para estado transitorio y permanente. Para este caso los autores realizaron una simulación en CFD tridimensional y evaluaron el rendimiento de una válvula micro-desviada biestable, para la

generación de microburbujas, utilizando un modelo con aproximaciones para estado estable y transitorio, con los modelos k - ε estándar y de turbulencia k – ε RNG. Concluyendo así satisfactoriamente que para bajos números de Reynolds y altas frecuencias de operación, la válvula puede servir a una amplia gama de aplicaciones industriales.

2.3 Clasificación por compresibilidad

La compresibilidad por definición física sencilla, para un volumen de fluido, es la capacidad de variar la densidad del fluido. Para la mecánica de fluidos es necesario abordar una óptica más profunda, donde no se trata de que el fluido pueda variar su densidad o no, sino se trata de, si efectivamente lo hace durante el flujo.

En mecánica de los fluidos generalmente se asume que los flujos de gases son compresibles y los flujos de fluidos no, aún que para la mayoría de los aspectos prácticos esto puede aplicarse, no es la verdad absoluta. La verdad es que la compresibilidad del flujo depende de la velocidad del flujo, pues los campos de presión varían con la velocidad por ende las densidades. Y sin excepción todos los fluidos varían su densidad, por lo que asumir el flujo incompresible es apenas una aproximación, ya que en ciertos flujos la variación de la densidad es menor al 2% o aún mucho menor.

En un resumen práctico, los gases también pueden tener un flujo incompresible, toda vez que su velocidad de flujo sea suficientemente baja. También así los fluidos, existen fluidos líquidos que pueden presentar un comportamiento de flujo compresible siempre que sus velocidades sean suficientemente altas. Más aspectos prácticos relacionados con la velocidad y la velocidad del sonido lo encontramos en el libro publicado por Gerhart y otros en 1992.

2.3.1 Flujos incompresibles

En las últimas dos décadas ha crecido el interés por el entendimiento del flujo a través de membranas porosas, Montoya, Orozco, & Londoño en 2012 nos presentan un estudio realizado a la presión y al perfil de velocidades relacionados al flujo de un fluido incompresible a través de membranas porosas con diferentes condiciones de

empaquetamiento. Los autores representan a la membrana porosa a través de un empaquetamiento de esferas en arreglo triangular desfasado. Los resultados se obtienen mediante el software comercial FLUENT y muestran que la distribución espacial de las esferas dentro del canal tiene una gran influencia en la caída de presión. Entre las conclusiones del trabajo se afirma que el diámetro de las esferas influye de gran manera en la caída de presión, generándose mayor caída de presión a menor diámetro de esferas.

2.3.2 Flujos compresibles

Un perfil aerodinámico es la sección transversal de un ala de avión o de una aspa o hélice. En este caso Prasad y otros en 2013 realizaron una fusión de perfiles aerodinámicos modelados en CFD, tanto para flujos compresibles como incompresibles, buscando un diseño optimizado de perfil aerodinámico. En sus conclusiones Pasad declara haber conseguido mejorar las ventajas del nuevo perfil por sobre los perfiles anteriores que fueron fusionados, utilizándolos a velocidad supersónicas como subsónicas.

2.4 Clasificación por cantidad de dimensiones

Aun cuando todos los casos reales de flujo de fluidos son en tres dimensiones, en ocasiones cuando las líneas de flujo son coplanares y en la dimensión perpendicular a este plano la geometría es regular y proyectada como prisma, es posible simplificarlo en un estudio numérico 2D, analizando el plano de las líneas de flujo, asumiendo la profundidad en un valor predeterminado igual al de una unidad de medida ejemplo 1 metro, teniendo en cuenta que en el caso de necesitarse las unidades reales basta con multiplicar adimensionalmente la relación entre la profundidad real en la misma unidad de medida y la unidad.

En otras ocasiones, cuando las líneas de flujo son alabeadas, no resulta tan conveniente una simplificación en dos dimensiones, porque se utiliza una simulación 3D.

2.4.1 Simulaciones en 2D

Como ya había dicho anteriormente, los modelos 3D pueden reducidos a un orden menor de 2D, como los muestra García en 2016, quien plantea un modelo de compresor que axial que si bien puede ser simulado en 3D mediante la fluidodinámica computacional (CFD), debido a los altos costos computacionales requeridos para la simulación 3D, García decide tomar la alternativa práctica de reducir el orden para evaluar el comportamiento del compresor axial de manera rápida, utilizando códigos de 2D y 1D, con ayuda de métodos semi-empiricos y correlaciones experimentales. Los resultados obtenidos muestran la capacidad del modelo de representar la física del proceso, a un costo computacional no alto. Ello según el autor confirma el potencial uso efectivo de esta herramienta para los ensayos de banco o de aplicaciones industriales, donde el análisis rápido y fiable toma importancia.

2.4.2 Simulaciones 3D

Un ejemplo de trabajo en 3D es el de Mateus & Gualdrón en 2011, donde utilizan el software comercial AnsysFluent junto con la herramienta EcoFursim©, para simular hornos industriales de destilerías. Buscando determinar las temperaturas picos en las paredes de los tubos de los hornos, ya que estas temperaturas son parámetros importantes para determinar la vida útil del tubo junto con su extensión de una corrida operación y controlar la operación en sí de los hornos industriales.

Según los autores estas temperaturas son muy difíciles de calcular con precisión y los simuladores comerciales de hornos comúnmente fallan en el cálculo. La fluidodinámica computacional (CFD) se torna así en la única técnica que calcula temperaturas pico de paredes de tubo con la precisión y exactitud necesarias, esto se debe a que los flujos de calor tanto convectivos como por radiación pueden ser calculados, aun pese a la complejidad de la geometría de los hornos y quemadores. Los autores compararon los resultados con datos de hornos de la refinería de Barrancabermeja (Barrancabermeja, Colombia) y validaron así las simulaciones. Aclarando que el caso de estudio es para hornos de refinería calentados por fuego directo y para el calentamiento no reactivo de hidrocarburos o petróleo.

Entre las conclusiones del citado trabajo, los autores destacan que los puntos calientes o de temperatura pico, son los puntos ideales para la instalación de termopares para la medición

y el control de los hornos. Dando así una aplicación práctica directa a los resultados de la simulación.

2.5 Clasificación por régimen de tiempo

Cuando se describe a régimen con relación al tiempo o temporal, se refiere a dos estados el transitorio y el permanente o estable. Cuál es la diferencia entre ellos, el régimen transitorio se caracteriza por tener distintos valores en sus campos de datos con el paso del tiempo, es decir sus parámetros varían con el tiempo, por ejemplo, si tomamos el campo de velocidad y analizamos los puntos en los que medimos la velocidad del fluido, el vector velocidad se encuentra cambiante, y con él, la presión.

Sin embargo, para el régimen permanente o estable o estacionario (se puede encontrar de las 3 maneras en la literatura en inglés steady state), los campos de velocidad y presión permanecen invariantes y en caso de que exista transferencia de temperatura el sistema debe encontrase en equilibrio térmico, es decir, el campo de temperaturas también debe ser invariante. Esto no quiere decir que el fluido no se mueva, muy por el contrario, quiere decir que el movimiento del fluido tiene siempre la misma velocidad en cada punto.

2.5.1 Régimen transitorio

Gualdrón, Mora, & Mateus en 2013, diseñaron un modelo de tanque de mezclado con agitación forzada para crudos de petróleo con diferentes propiedades que inicialmente se encuentran separados debido a su diferencia de densidad, así es definido un estado inicial. El estado final se define como el estado de mezcla homogénea. El objeto del estudio es observar el proceso de mezclado y calcular las propiedades de la mezcla en función de las fracciones volumétricas. Estudiando también el tamaño de la malla y el diferencial de tiempo.

Este trabajó se realizó recurriendo a la simulación a través de la fluidodinámica computacional o CFD, comparando los resultados con datos experimentales y validando así la investigación. Los autores concluyen las simulaciones fueron satisfactorias y exitosas con un tiempo de computo de entre 10 y 12hs, con una CPU comercial de multinúcleos. Teniendo una rápida convergencia para los valores del Ansys determinados por defecto.

2.5.2 Régimen permanente o estacionario

Ordoñez-Viñán en 2018 buscaron caracterizar el perfil de velocidad de flujos incompresible en tuberías de PVC, diseñando para ello un modelo para la simulación en CFD, el cual es comparado y validado con datos experimentales de un banco de prueba de perdidas. Fundamentando su modelo en las ecuaciones de Navier-Stokesde flujo incompresible para la conservación de la cantidad de movimiento y conservando la cantidad de masa, pues en los ensayos experimentales, se encuentran tuberías llenas impulsadas por bombas en ciclo cerrado es decir el flujo es estacionario. Para las simulaciones se utilizó el software comercial Ansys CFX. Simulándose únicamente fluidos turbulentos entre sus conclusiones el trabajo asume que el error máximo encontrado de 8,6% es válido para simulaciones estacionarias.

2.6 Tipos de mallas

En general las mallas son clasificadas como estructuradas y no estructuradas, esto no quiere decir que sea la única forma de clasificación, también tienen subclasificaciones, pero desde el punto de vista de la programación es un parámetro importante, por la lógica y la filosofía de la programación. Las mallas estructuras son aquellas que presentan una estructura como las matrices, donde cada pequeño volumen de control seria interpretado como un elemento de una matriz, así en número de elementos en cada fila o columna es definido e igual para los demás. En mi experiencia de programación, esto simplifica las cosas a gran medida, ya que es posible utilizar únicamente dos ciclos anidados para recorrer todos los volúmenes de control y tener definido de una manera exacta quienes son los volúmenes vecinos que intervienen en las ecuaciones de dominio, por lo general en este tipo de mallas estructuradas los volúmenes son rectangulares o prismáticos.

Por el contrario, las mallas no estructuradas no siguen una estructura y desde el punto de vista de programación cada una tiene una identificación propia también carente de estructura sino más bien como un ciclo corrido, por lo general sus volúmenes de control son triángulos o exaedros, incluso pentágonos o tetraedros.

También existen las mallas hibridas y las multi-bloques. Las mallas hibridas son aquellas que en parte utilizan mallas estructuradas y parte no estructuradas. Mientras que las multi-bloques, divide la geometría en bloques donde los bloques son arbitrarios y cada bloque tiene una malla que puede ser estructurada o no.

2.6.1 Malla estructurada

De la mano Immonen en 2018 nos llega un manuscrito que introduce didácticamente Un método de transformación paramétrica para generar mallas estructuradas para aplicaciones de flujo de superficie libre marina con simetría plana. El artículo presenta un método para transformar una malla de plantilla paramétrica, rectangular y estructurada en malla 3D de manera que incorpora una superficie de casco dada correctamente alineada con una zona de interfaz aire-agua predefinida, un límite de alta calidad la región de la capa cerca del casco, y una transición suave de la malla a la estructura de malla de la plantilla alejada del casco. El autor afirma que el método propuesto se validó con éxito.

2.6.2 Mallas no estructuradas

Un trabajo presentado por Fuentes, Guerrero, & Sánchez en 2013 explica convenientemente de una manera didáctica la discretización para flujo difusivo en malla no estructurada, exponiendo los inconvenientes superados y los persistentes, para mallas generales, tomando como base el trabajo de Date en 2005. El autor refiere que el método puede ser aplicado a elementos de cualquier tipología siempre y cuando se conozca los detalles de la geometría. Además, demostró la eficiencia y eficacia del nuevo método sobre las limitaciones anteriores del método de Date.

2.7 Tipos de geometrías

No he conseguido en la literatura un autor que defina de manera clara y precisa hasta qué punto una geometría deja de ser simple y se torna compleja. Pero a la vez todos

convergen en que los cuadrados o rectángulos son geometrías sencillas, en ciertos calos las polinias de hasta seis líneas pueden ser considerados sencillas, esto para el caso 2D o el prisma de las mencionadas en el caso 3D. A la vez también se puede afirmar que las geometrías compuestas por varias curvas distintas, en 2D o 3D, son efectivamente geometrías completas.

Otro punto para analizar es que el facto que computacionalmente realza la importancia de definir las geometrías es poder definir las mallas y las condiciones de contorno. Entonces también es común que se le llame geometría simple o sencilla a toda aquella que permita un mallado estructurado.

2.7.1 Geometrías simples

Cortes y otros en 2014 presentan un artículo de revista que nos enseña didácticamente como modelar algunos casos básicos de convección natural en CFD. El objeto de este trabajo según los autores fue el de orientar a los nuevos usuarios de CFD en la manera de su aplicación con el interés de reducir los errores debido al factor humano, caso que es reiterativo en ciertas simulaciones de ambientes cerrados.

Si bien el enfoque es un poco más civil y con vistas a ambientes de edificios. Este trabajo presenta claramente como las geometrías sencillas tienen gran protagonismo aún en nuestros días en el mundo de las ingenierías y las simulaciones CFD. Y a veces, aunque la geometría real no sea tan sencilla, recurrir a las simplificaciones siempre es una opción válida, toda vez que la variación en los resultados debido a las simplificaciones realmente pueda ser despreciable.

2.7.2 Geometrías complejas

Una tesis de doctorado presentada por Aljure de la Universidad de Cataluña en 2007 nos ilustra el procedimiento de simulación en CFD de geometrías complejas, yendo gradualmente desde una esfera hasta un modelo realista de un automóvil en cámara de viento. El trabajo abarca la resolución numérica de las ecuaciones de Navier-Stokes en las distintas

geometrías, ordenadas por orden de complejidad. El autor afirma que para cada capítulo logro una publicación bibliográfica, por lo que yo hayo el trabajo bastante rico en contenido.

2.8 Cantidad de fases

Un flujo bifásico es aquel que contiene dos tipos de fluidos en su flujo, estos pueden ser de la misma materia en distintas fases como es el caso de las cañerías de vapor, que además de contener vapor, en menor o mayor grado contienen condensado de vapor (agua a altas temperaturas), teniendo así una fase gaseosa y otra líquida del mismo material.

El caso más común de flujo bifásico es de un fluido liquido cualquiera y el aire como fluido gaseoso. Pero esto no quiere decir que uno de los fluidos deba ser necesariamente gaseoso, también puede darse el caso de dos fluidos líquidos inmiscibles, el agua y el aceite siempre son los ejemplos más claros de ello.

Habiendo presentado el caso de los flujos bifásicos es más fácil presentar el del flujo monofásico, que consiste sencillamente de un flujo compuesto por un único fluido en un único estado, liquido o gaseoso.

2.8.1 Flujo monofásico

El trabajo de Jines en 2017 representa el comportamiento de un flujo monofásico o de una sola fase, incompresible. Obtenido mediante la simulación en CFD a través del software comercial FLUENT, con el objeto de caracterizar el factor de perdida conocido como KL para un accesorio de tubería industrial, codo de 90°. Utilizando el Método de Colebrook para determinar el coeficiente de ficción. Jines logro concluir satisfactoriamente la valides de su modelo con una diferencia porcentual de 3.14% respecto a los valores de la literatura.

2.8.2 Flujo bifásico

La cavitación es uno los principales problemas de la ingeniería en la actualidad, este fenómeno ocurre cuan la presión del fluido liquido baja hasta el punto de evaporación,

obteniéndose así una estructura bifásica en el fluido, obteniéndose una interfaz liquido-vapor. Moll y otros en 2011 logran una primera aproximación para caracterizar la cavitación mediante la fluido dinámica computacional. Sin embargo el autor presenta opciones para mejora la precisión de las simulaciones que puedan ser más semejantes a los resultados experimentales y en un futuro trabajo espera poder estudiar los daños que este fenómeno ocasiona.

3 METODOLOGÍA

Existen casos o fenómenos físicos que son considerados de interés para la ciencia actual que pueden ser formulados a través de las ecuaciones diferenciales, pero son difíciles, muy costosas o aún imposibles de resolver de manera analítica. Sin embargo, como están dadas las ecuaciones diferenciales es posible lograr una aproximación numérica bastante precisa, recurriendo claramente a los métodos numéricos, esto es hablando en líneas generales para toda la ciencia contemporánea y en particular eso ocurre frecuentemente en nuestra área de estudio la mecánica de fluidos.

Para estos casos de la mecánica de fluidos que son más factibles de resolverse por métodos numéricos es que recurrimos a la fluidodinámica computacional o CFD. En este capítulo se presenta la metodología utilizada para fundamentar matemáticamente los procedimientos o sentencias ejecutadas en los códigos escritos en este trabajo. En otra palabra se describe el algoritmo matemático utilizado de los códigos de programación. Posteriormente estos códigos tomarán valor real cuando cumplan su finalidad y sean utilizados efectivamente en el objeto de estudio.

3.1 Ecuaciones gobernantes

Como ya se ha mencionado en el capítulo 1 de este trabajo se trata del desarrollo de un código computacional o software que aplica el algoritmo SIMPLE para resolver casos de convección natural. Este algoritmo es descripto por Patankar en 1980 aplicando el método de volúmenes finitos donde las ecuaciones gobernantes son las de la mecánica de fluidos.

Para fluidos Newtonianos e incompresibles con transferencia de calor, las ecuaciones fundamentales que gobiernan los fenómenos de fluidos son las ecuaciones de Navier-Stokes, conservación de la masa y conservación de la energía. Para otros casos donde se incluyan más fenómenos como el caso de reacciones químicas, pueden incluirse otras ecuaciones como las ecuaciones de conservación de las especies químicas. En coordenadas cartesianas

bidimensionales, estas ecuaciones citadas anteriormente pueden escribirse según White en 2014 de la siguiente manera.

Ecuaciones de Navier-Stokes para ambos ejes cartesianos:

$$\frac{\partial(\rho u)}{\partial t}+\frac{\partial(\rho u.u)}{\partial x}+\frac{\partial(\rho v.u)}{\partial y}\ =-\frac{\partial P}{\partial x}+\frac{\partial}{\partial x}(\mu\frac{\partial u}{\partial x})+\frac{\partial}{\partial y}(\mu\frac{\partial u}{\partial y})+\rho g_x \qquad (3.1)$$

$$\frac{\partial(\rho v)}{\partial t}+\frac{\partial(\rho u.v)}{\partial x}+\frac{\partial(\rho v.v)}{\partial y}\ =-\frac{\partial P}{\partial y}+\frac{\partial}{\partial x}(\mu\frac{\partial v}{\partial x})+\frac{\partial}{\partial y}(\mu\frac{\partial v}{\partial y})+\rho g_y \qquad (3.2)$$

donde las variables “x” e “y” son las posiciones de cada partícula en sistema cartesiano, “u” y “v” son las componentes de la velocidad en los ejes “x” e “y” respectivamente, “t” representa el tiempo, “P” la presión, “μ” la viscosidad dinámica, “gx” y “gy” las componentes de la gravedad en los ejes “x” e “y” respectivamente y “ρ” la densidad.

Ecuación de conservación de la masa:

$$\frac{\partial\rho}{\partial t}+\frac{\partial(\rho u)}{\partial x}+\frac{\partial(\rho v)}{\partial y}\ =0 \qquad (3.3)$$

Ecuación de la conservación de la energía térmica:

$$\frac{\partial(C_P\rho T)}{\partial t}+\frac{\partial(C_P\rho uT)}{\partial x}+\frac{\partial(C_P\rho vT)}{\partial y}\ =\frac{\partial}{\partial x}(\lambda\frac{\partial T}{\partial x})+\frac{\partial}{\partial y}(\lambda\frac{\partial T}{\partial y}) \qquad (3.4)$$

donde la variable “T” es la temperatura, la variable “λ” es la conductividad térmica y la variable “C_P” es el calor específico a presión constante.

En este trabajo como se trata de un caso de convección natural de fluidos incompresibles, es posible adoptar la aproximación de Bousinesq. Esta aproximación es utilizada para problemas donde la temperatura varía de punto a punto produciendo una transferencia de calor y un flujo de fluido, donde el flujo necesariamente satisface la conservación de la masa, conservación de la energía y conservación de momento (Navier-

Stoke). En esta aproximación la hipótesis fundamental es que la variación de la densidad es tan pequeña que puede ser despreciada excepto cuando tiene injerencia la gravedad, es decir, la diferencia de densidad causa una fuerza de flotación positiva o negativa, dependiendo del valor de esta, aun cuando muy pequeña que sea la diferencia es suficiente para lograr estas fuerzas. Tritton en 1977

Aproximación de Boussinesq:

$$\rho g_y = \rho g \beta (T - T_0) \qquad (3.5)$$

3.2 Métodos numéricos

Cuando hablamos de métodos numéricos estamos hablando de la discretización de las ecuaciones diferenciales para una posterior resolución por métodos iterativos o analíticos. Los métodos más conocidos y aplicados a la ingeniería son, diferencias finitas, elementos finitos y volúmenes finitos. El objetivo de estos métodos numéricos es el de convertir las ecuaciones de derivadas parciales en sistema de ecuaciones lineales o algebraicos y así resolverlos por métodos iterativos, aunque a veces son resueltos por métodos directos para obtener una primera aproximación. Pero como los métodos directos no son los más exactos, esta primera solución por método directo es la solución inicial de partida para que a través de un método iterativo se logre mayor precisión en los resultados. En los siguientes párrafos se hablará más de los métodos de diferencias finitas y de elementos finitos dejando un apartado en particular para volúmenes finitos debido a que este último es el método utilizado en este trabajo.

El método de diferencias finitas es el más antiguo de los métodos citados para la solución de ecuaciones en derivadas parciales. Se cree que este método fue introducido a la literatura en el siglo XVII por Euler. El punto inicial de este método es una ecuación diferencial de una variable ϕ. Esta variable desconocida ϕ tiene un comportamiento de entre los puntos de una malla, el cual se describe mediante la ecuación diferencial, la cual en este método es sustituida por una aproximación utilizando la serie de Taylor. La serie de Taylor nos permite una aproximación tanto para la primera como para la segunda derivada,

utilizando los valores de ϕ para los nodos y los nodos como índice de coordenadas. El resultado de esta operación es que para cado nodo resulta así una ecuación algebraica en la cual las incógnitas son el valor de ϕ para ese nodo y el valor de ϕ para los nodos contiguos, quedando al final tantos números de ecuaciones como nodos, es decir, como incógnitas. Cada nodo tiene un subíndice, según su ubicación, normalmente este es utilizado como coordenada. La principal desventaja de utilizar este método es que no se garantiza la conservación de la masa, pues el método no es conservativo, por lo cual requiere de especial cuidado al momento de modelar el sistema y la precisión depende directamente de los errores de truncamiento de la serie de Taylor. Si bien el método funciona para geometrías simples, es muy dificultoso para aplicarlo a geometría complejas lo cual es una desventaja. Xamán 2015.

El método de elementos finitos, lo introdujo Tuner en 1956, inicialmente los planteó como un análisis de estructura, pero, diez años después, el método fue utilizado en la solución de ecuaciones de campos en medios continuos. Se puede decir que el método de elementos finitos es generalización de los métodos de principio variacional o método de Ritz y de residuos ponderados como por ejemplo el método de Galerkin o también el método de mínimos cuadrados. Está basado en la idea de que la solución de la variable incógnita ϕ de una ecuación diferencial puede ser representada como una combinación lineal de un parámetro desconocido Ci y de una función de ϕi para todo el dominio. Las funciones son aproximaciones seleccionadas de tal manera que satisfagan las condiciones de contorno del problema y los valores desconocidos Ci se determinan de forma que satisfagan la ecuación diferencial. Claramente para estos métodos variacionales y ponderados existe una desventaja en la construcción de las funciones de aproximación de ϕi que satisfaga las condiciones de contorno del problema, más aún teniendo en cuenta que en los problemas con geometría compleja se debe dividir la geometría general en geometrías más pequeñas o regiones y estas aproximaciones deben cumplir con todas ellas en diferentes tipos de condiciones de contorno en todo el dominio.

3.3 Método Aplicado (Volúmenes Finitos)

El método aplicado para este trabajo es el método de volúmenes finitos y para explicarlo supongamos que tenemos un volumen de control, una ecuación conservativa que rige al dominio y conocemos las condiciones de contorno. Entonces el método de volúmenes finitos nos permitiría calcular los valores de una variable ϕ para todo el dominio basándose en la ecuación general conservativa, como por ejemplo la conservación de la masa.

Esto se logra dividiendo el volumen de control en pequeños volúmenes de control infinitesimales que no deben superponerse unos a otros sino ser contiguos y dar continuidad a todo el dominio, estos volúmenes son puntos llamados nodos. Este conjunto de nodos y volúmenes de control infinitesimales son llamados malla computacional o grilla. En la práctica computacional los pequeños volúmenes de control no siempre son realmente infinitesimales, a veces pueden tener un orden superior sin necesariamente representar un error porcentual significativo en el resultado.

Este método, como los otros dos anteriores, se basa en la discretización de las ecuaciones que gobiernan los fenómenos de estudio en cada pequeño volumen de control. Normalmente para la mecánica de fluido se aplican las ya conocidas ecuaciones conservativas en su manera integral o también la ecuación general de transporte. Una vez discretizada la ecuación diferencial esta es aproximada a una ecuación algebraica para cada uno de los volúmenes de control, donde para cada nodo corresponde un valor de la variable ϕ y la ecuación algebraica la describe como una variable dependiente de los mismos valores de ϕ pero asociadas a los nodos vecinos, es decir, para cada nodo ni existe un valor de ϕi que se corresponden y aplicando la ecuación algebraica a cada nodo ni se logra una ϕi=f(ϕvecinos) que es función de los vecinos y por consiguiente obtenemos tantas incógnitas como números de nodos y tantas ecuaciones como incógnitas, armando un sistema de ecuaciones capaz de ser resuelta por métodos numéricos. Explicaciones detalladas de la discretización de este método pueden ser halladas en Patankar (1980) o Versteeg e Malalasekera (2007).

3.4 Ecuación General Conservativa de Convección-Difusión y su Discretización

Existe una ecuación que de forma general puede representar los fenómenos de transporte, en ella se pueden representar las ecuaciones conservativas gobernantes que se describen en este capítulo en el apartado 3.1, conocida como Ecuación general conservativa de convección-difusión o ecuación general de transporte, que puede ser representada de muchas maneras, pero la que utilizaré para este libro será la siguiente: Xamán en 2015

$$\frac{\partial(\rho\phi)}{\partial t} + \vec{\nabla}.\left(\rho\vec{V}\phi\right) = \vec{\nabla}.\left(\Gamma\vec{\nabla}\phi\right) + S \quad (3.6)$$

Esta es una ecuación diferencial para una variable genérica ϕ, que posee un coeficiente de difusión representado por Γ y también posee un término fuente o de generación representado por S. Anteriormente se había expresado que esta ecuación puede representar a las ecuaciones que definimos como gobernantes, esto es asociando valores a las variables ϕ, Γ y S para convertir la ecuación general en una ecuación específica. Esto se muestra en la Tabla 3.1 extraída de la Tabla 2.2 de Xamán en 2015.

Ecuación Conservativa	$\boldsymbol{\phi}$	$\boldsymbol{\Gamma}$	$\boldsymbol{S}$
Conservación de la masa	1	0	0
Conservación de cantidad de movimiento eje X	u	μ	$-\frac{\partial P}{\partial x} + F_x + S_x$
Conservación de cantidad de movimiento eje Y	v	μ	$-\frac{\partial P}{\partial y} + F_y + S_y$
Conservación de cantidad de movimiento eje Z	w	μ	$-\frac{\partial P}{\partial z} + F_z + S_z$
Conservación de la enérgia	e_{int}	λ	$-\frac{\partial u_j}{\partial z_j} + \Phi + S_E$

Tabla 3. 1: Equivalente de la formulación general de convección difusión

3.4.1 Dominio espacial de la ecuación general

Para poder integrar una ecuación diferencial cualquiera, es necesario definir el dominio de la ecuación y el espacio de integración, es decir, el área donde esta ecuación es válida y las fronteras entre las cuales será integrada. También para los métodos numéricos es así, bajo el concepto de que la integral por definición es la suma de los espacios infinitesimales, en métodos numéricos se aplica casi el mismo concepto, pero los espacios son finitos y aún lo suficientemente pequeños para lograr una muy buena aproximación.

Así pues, para el método aplicado de volúmenes finitos, dividimos todo el dominio de una manera ordenada y estructurada, en volúmenes más pequeños definidos, tal que no sean superpuestos, pero sean continuos en todo el domino sin dejar espacios vacíos. De esta manera se crean un número finitos de volúmenes de control pequeños y con cada uno de ellos un punto nodal o nodo, Patankar en 1980 describe dos prácticas para este método, una de ellas donde las fronteras de los volúmenes de control están ubicadas en el punto de medio de dos nodos y otra donde las fronteras están localizadas sobre los nodos.

También Xamán en 2015 describe dos maneras de tratas los volúmenes en las fronteras, suponiendo que necesariamente un nodo debe coincidir con la frontera del dominio para poder guardar las condiciones de contorno, el tratamiento puede ser de contacto con la frontera, donde la interfase del volumen de control coincide plenamente con la frontera, así existirá un nodo en contacto con la frontera del dominio, pero con un espesor de volumen nulo en la dirección normal a la frontera. En cambio, cuando se trata de sin contacto con la frontera, la interfase del volumen de control no coincide con la frontera, entonces es necesario que el volumen de control tenga un espesor en la frontera y que el nodo coincida con la frontera.

Para este trabajo se utilizará una malla de contacto con la frontera y de interfases centradas a los nodos como se muestra en la Figura 3.1.

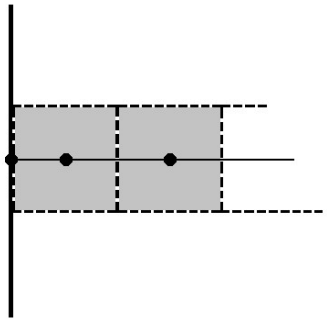

Figura 3. 1: Malla en contacto con la frontera y nodos centrados

El tamaño y la forma de los volúmenes de control puede ser uniforme en todo el dominio computacional o podría ser ajustado en tamaños variables para cubrir las distintas necesidades del problema, así podemos definir tres tipos de mallas.

– Malla estructurada: las mallas estructuradas son aquellas formadas por un conjunto de volúmenes de control y nodos que pueden ser identificados de una manera única mediante un grupo de índices ordenados *(i,j,k)* en el caso 3D o *(i,j)* en el caso 2D. Este tipo de mallas es una de las más simples, puede ser representada como una malla cartesiana a través de la conversión correcta. Tiene grandes ventajas a la hora de ser aplicada a geometrías simples, aunque muchas veces acumulan puntos en regiones que no necesariamente sean de interés. Es de uso muy común en el método de volúmenes finitos, de hecho, será utilizada en este trabajo.

– Malla no estructurada: para geometrías más completas, son recomendadas aquellas mallas que pueden adaptarse más fácilmente a la forma arbitraria de la geometría. Estas mallas no recorren ninguna estructura cartesiana, sino que se encuentran cubriendo el dominio concentrándose únicamente en los puntos de mayor interés o en los puntos con gradientes de datos muy grandes para facilitar la convergencia, formada generalmente por triángulos o tetraedros. Inicialmente este tipo de mallas pueden ser utilizadas por cualquier esquema de discretización, pero los que mejores se adaptan son los métodos de volúmenes finitos y elementos finitos. En la actualidad existen varios trabajos y softwares comerciales dedicados a la generación automatizada de este tipo de malla, pero gran ventaja en cuanto a las geometrías a veces puede ser contrarrestadas con la estructura irregular de datos que son producidos y la necesidad

de algoritmos bastante más complejos y costosos en términos computaciones para resolver las matrices.

– Malla híbrida: En las mallas híbridas se acoplan las diferentes mallas expuestas anteriormente, en este tipo de trabajo es necesario tener cuidado con el acoplamiento de las diferentes mallas. J. H. Ferziger y M Peric en 2002

3.4.2 Discretización de la ecuación general

Para la discretización es adoptada una mala estructurada uniforme, de contacto en la frontera con los nodos centrados en el volumen de control. Como se muestra en la **Figura 3.2**.

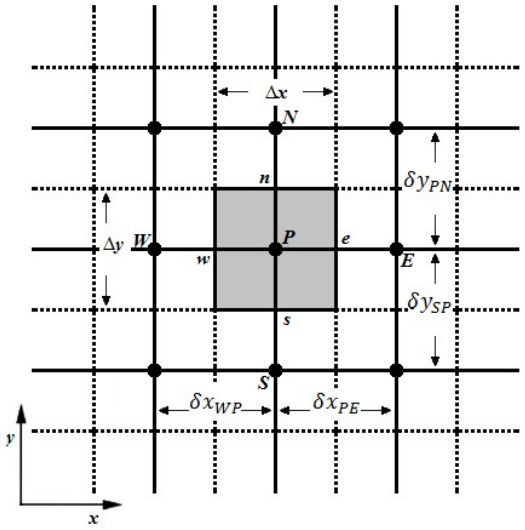

Figura 3. 2: Malla estructurada, volumen de control de referencia P, fronteras y volúmenes de control vecinos.

En esta imagen **Figura 3.2** se ve un volumen de control y un nodo P para una mala bidimensional. En la figura también están representados en mayúsculas los nodos vecinos al norte *(N)*, sur *(S)*, este *(E)* y oeste *(W)*, y las interfases de los volúmenes de control en

minúsculas *(n, s, e, w)*, así como el tamaño del volumen de control *(Δx e Δy)* y las distancias con los nodos vecinos (δx_{WP}, δx_{PE}, δy_{SP}, δy_{PN}). La notación utilizada en la figura es la empleada para la integración de la ecuación general.

La ecuación general **Eq. 3.6**, puede ser escrita en dos dimensiones en las coordenadas cartesianas de la siguiente manera.

$$\frac{\partial(\rho\phi)}{\partial t}+\frac{\partial(\rho u\phi)}{\partial x}+\frac{\partial(\rho v\phi)}{\partial y}=\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right)+S \tag{3.7}$$

Si tomamos esta **Eq. 3.7** para cada pequeño volumen de control alrededor de un nodo P genérico como es representado en la **Figura 3.2** y lo integramos por términos en ambos ejes cartesianos, tendríamos:

1º Término. $$\int_s^n\int_w^e\frac{\partial(\rho\phi)}{\partial t}dxdy=\frac{\partial(\rho\phi)}{\partial t}\Delta x\Delta y \tag{3.8a}$$

2º Término. $$\int_s^n\int_w^e\frac{\partial(\rho u\phi)}{\partial x}dxdy=[(\rho u\phi)_e-(\rho u\phi)_w]\Delta y \tag{3.8b}$$

3º Término. $$\int_s^n\int_w^e\frac{\partial(\rho v\phi)}{\partial y}dxdy=[(\rho v\phi)_n-(\rho v\phi)_s]\Delta x \tag{3.8c}$$

4º Término. $$\int_s^n\int_w^e\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)dxdy=\left[\left(\Gamma\frac{\partial\phi}{\partial x}\right)_e-\left(\Gamma\frac{\partial\phi}{\partial x}\right)_w\right]\Delta y \tag{3.8d}$$

5º Término. $$\int_s^n\int_w^e\frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right)dxdy=\left[\left(\Gamma\frac{\partial\phi}{\partial y}\right)_n-\left(\Gamma\frac{\partial\phi}{\partial y}\right)_s\right]\Delta x \tag{3.8e}$$

6º Término. $$\int_s^n\int_w^e S\,dxdy=\bar{S}\Delta x\Delta y \tag{3.8f}$$

Con lo que la ecuación integrada quedaría así:

$$\begin{aligned}&\frac{\partial(\rho\phi)}{\partial t}\Delta x\Delta y+[(\rho u\phi)_e-(\rho u\phi)_w]\Delta y+[(\rho v\phi)_n-(\rho v\phi)_s]\Delta x=\ldots\\&\ldots\left[\left(\Gamma\frac{\partial\phi}{\partial x}\right)_e-\left(\Gamma\frac{\partial\phi}{\partial x}\right)_w\right]\Delta y+\left[\left(\Gamma\frac{\partial\phi}{\partial y}\right)_n-\left(\Gamma\frac{\partial\phi}{\partial y}\right)_s\right]\Delta x+\bar{S}\Delta x\Delta y\end{aligned} \tag{3.9}$$

donde el $\bar{S}$ representa al valor medio del término fuente dentro del volumen de control, $(\rho u\phi)$ y $(\rho u\phi)$ representan a los términos convectivos en los distintos ejes y, $\left(\Gamma\frac{\partial\phi}{\partial x}\right)$ y $\left(\Gamma\frac{\partial\phi}{\partial y}\right)$ representan a los términos difusivos.

Para finalizar la discretización es necesario además integrar la **Eq. 3.9** en el tiempo, y para ello es necesario definir un perfil del comportamiento de la variable ϕ en el tiempo, para cada instante de (t) a $(t + \Delta t)$. Según Xamán en 2015. La integral de la variable ϕ puede representarse como combinación lineal de sus valores conocidos final e inicial para un intervalo de tiempo lo suficientemente pequeño, de la siguiente manera:

$$\int_t^{t+\Delta t} \phi \, dt = (f\phi + (1-f)\phi^0)\Delta t \qquad 0 \leq f \leq 1 \tag{3.10}$$

donde ϕ^0 representa al valor de la variable ϕ para el instante (t) y ϕ representa a la misma variable para el instante de tiempo $(t + \Delta t)$. Cada valor de f representa a un esquema de integración diferente.

- **Totalmente implícito:** para este trabajo el esquema de integración utilizado es el esquema totalmente implícito, donde el valor de f es igual a la unidad $(f = 1)$. Todo esquema donde f posea algún valor no nulo, es un esquema implícito porque depende del comportamiento de la variable ϕ en el tiempo, pero en caso del esquema totalmente implícito el resultado de la ecuación **Eq. 3.10** es independiente al valor de ϕ^0, por lo cual el esquema se torna incondicionalmente estable, como lo explica Patankar en 1980. Debido a esta estabilidad es que fue escogido este esquema para el presente trabajo.

- Crank Nicolson o semi implícito: este esquema se da cuando el valor f es igual 0.5 $(f = 0.5)$ y es bastante utilizado para intervalos de tiempos muy pequeños debido a que también es un esquema incondicionalmente estable, pero en contra partida al ser un esquema linear puede no dar una concordancia física en cuanto a resultados para diferenciales de tiempo muy grandes.

- Esquema explícito: en este esquema el valor de f es nulo $(f = 1)$. Por lo que la ecuación **Eq. 3.10** toma una dependencia exclusiva de ϕ^0. Aunque esto simplifique los cálculos este esquema no es recomendado para flujos transitorios de fluidos, transferencia de calor por convección o transporte de masa. Si puede arrojar resultados más realistas para casos sencillos de difusión.

Utilizando un esquema totalmente implícito la ecuación **Eq. 3.10** integrada en el tiempo, resulta de la siguiente manera:

$$\frac{(\rho\phi)_P-(\rho\phi)_P^0}{\Delta t}\Delta x\Delta y+[(\rho u\phi)_e-(\rho u\phi)_w]\Delta y+[(\rho v\phi)_n-(\rho v\phi)_s]\Delta x=\ldots$$
$$\ldots\left[\left(\Gamma\frac{\partial\phi}{\partial x}\right)_e-\left(\Gamma\frac{\partial\phi}{\partial x}\right)_w\right]\Delta y+\left[\left(\Gamma\frac{\partial\phi}{\partial y}\right)_n-\left(\Gamma\frac{\partial\phi}{\partial y}\right)_s\right]\Delta x+\bar{S}\Delta x\Delta y \qquad (3.11)$$

3.4.3 Interpolación numérica

Observando la ecuación **Eq. 3.11** se denota la necesidad conocer los valores de la variable ϕ en las interfaces del volumen de control y sus derivadas. Inicialmente conocemos los valores de ϕ em los nodos por lo que es conveniente escribir los valores de ϕ en las caras en función de sus valores en los nodos. Para esto es necesario hacer una interpolación numérica.

Usualmente el primer método de interpolación enseñado en los cursos de ingeniería es el de diferencias centradas de segundo orden, que puede ser más fácilmente comprensible imaginándola como una media aritmética entre los nodos o una regla de tres simple, aunque lo que realmente se piensa es en la definición de la derivada. Así por ejemplo el valor diferencial de ϕ, en el eje "X" en la posición de la interfase "e" sería el siguiente.

$$\left(\frac{\partial\phi}{\partial x}\right)_e=\left(\frac{\phi_E-\phi_P}{\partial x_{PE}}\right) \qquad (3.12)$$

Pero este no es el único método, para el cálculo de los valores de ϕ en las interfases de los volúmenes de control, existen varios tipos de interpolaciones que normalmente las clasificamos en esquemas de bajo y de alto orden, esto es según la cantidad de valores conocidos que utilice para resolver su aproximación, por ejemplo, un esquema que utiliza dos nodos para realizar la interpolación es un esquema de segundo orden. Para este trabajo son utilizados esquemas de bajo orden en beneficio del costo computacional. Los esquemas representados en este trabajo son:

-Esquema Central o de diferencias centradas: como ya fue explicado, este esquema es utilizado para calcular los valores de ϕ en las fronteras del volumen de control y se puede

interpretar como una media aritmética a los valores entre los dos nodos. Este esquema da buenos resultados para casos donde se analiza fenómenos con baja velocidad y no se aconseja para casos altamente convectivos, ya que en este esquema no se analiza la dirección del flujo.

-Esquema UpWind: dado que el esquema central no considera la dirección del flujo, para casos puramente convectivos es posible pensar en una manera de considerar la dirección del flujo utilizando el esquema UpWind. Este es un esquema de primer orden, que propone tomar el valor de la interfase igual al valor de aguas arriba, utilizando así solamente el valor del nodo siguiente, considerando la corriente principal del flujo.

-Esquema Exponencial: Este esquema representa una solución exacta de un problema unidimensional visto desde la base de una solución analítica.

-Esquema Híbrido: si el esquema Exponencial tiene soluciones exactas resultantes del análisis unidimensional, esto no se cumple en cuanto exactitud para los casos multidimensionales. Esto hace el esquema Exponencial no sea recomendado para problemas complejos, pues deja de brindad exactitud y conlleva un alto costo computacional. Por lo cual se plantea el esquema Híbrido, que es una combinación de los esquemas Centrado y UpWind, utilizando el esquema Centrado en regiones de baja velocidad y el Esquema UpWind en alta velocidad.

-Esquema de Ley de Potencia (Power Law): este esquema fue desarrollado por el Prof. Patankar en 1980, esta formulación tiene mayor precisión que los esquemas mencionados, Central y UpWind. Este esquema utiliza una aproximación lineal por tramos para representar datos exactos; se puede utilizar esta aproximación con resultados tan exactos como al esquema exacto o exponencial, pero con mucho menor costo computacional.

La revisión de los esquemas de baja orden mencionados anteriormente fue basada en los comentarios encontrados en Xamán en 2015, Patankar en 1980 y Versteeg y Malalasekera en 2007.

3.4.4 Simplificación de la ecuación discretizada

A modo de una escritura más simple y abreviada de la ecuación **Eq. 3.11**, se definen los siguientes términos:

-Flujos convectivos:

$$\dot{F}_e = (\rho u)_e \Delta y \quad \dot{F}_w = (\rho u)_w \Delta y \quad \dot{F}_n = (\rho v)_n \Delta x \quad \dot{F}_s = (\rho v)_s \Delta x \tag{3.13}$$

-Flujos difusivos:

$$\dot{D}_e = \left(\frac{\Gamma_e}{\delta x_{PE}}\right) \Delta y \quad \dot{D_w} = \left(\frac{\Gamma_w}{\delta x_{WP}}\right) \Delta y \quad \dot{D}_n = \left(\frac{\Gamma_n}{\delta x_{PN}}\right) \Delta x \quad \dot{D}_s = \left(\frac{\Gamma_s}{\delta x_{SP}}\right) \Delta x \tag{3.14}$$

-Número de Péclet:

$$Pe_e = \frac{\dot{F}_e}{\dot{D_e}} \quad Pe_w = \frac{\dot{F_w}}{\dot{D_w}} \quad Pe_n = \frac{\dot{F_n}}{\dot{D_n}} \quad Pe_s = \frac{\dot{F_s}}{\dot{D_s}} \tag{3.15}$$

El número de Péclet es un número adimensional, nombrado así en honor el físico francés Juan Claude Eugene Péclet, que como puede observarse en la ecuación **Eq. 3.15**, define la relación entre el flujo convectivo y el flujo difusivo.

Se descompone el término fuente en dos términos, uno dependiente de ϕ y otro independiente.

$$\bar{S} = S_C + S_P \phi_P \tag{3.16}$$

Podemos escribir la ecuación **Eq. 3.11** en una notación simplificada de coeficientes agrupados como sigue.

$$a_P \phi_P = a_E \phi_E + a_W \phi_W + a_N \phi_N + a_S \phi_S + b \tag{3.17}$$

donde:

$$a_E = D_e A(|Pe_e|) + max[-F_e, 0] \quad (3.18a)$$

$$a_W = D_w A(|Pe_w|) + max[F_w, 0] \quad (3.18b)$$

$$a_N = D_n A(|Pe_n|) + max[-F_n, 0] \quad (3.18c)$$

$$a_S = D_s A(|Pe_s|) + max[F_s, 0] \quad (3.18d)$$

$$a_P^0 = \rho^0 \frac{\Delta x \Delta y}{\Delta t} \quad (3.18e)$$

$$a_P = a_E + a_W + a_N + a_S + \rho \frac{\Delta x \Delta y}{\Delta t} - S_P \Delta x \Delta y \quad (3.18f)$$

$$b = a_P^0 \phi_P^0 + S_C \Delta x \Delta y \quad (3.18g)$$

La función A = (|Pe|) depende del esquema numérico utilizado. En la **Tabla 3.2** (Extraída de la Tabla 4.2 del Xamán 2015) son presentadas las equivalencias de la función A = (|Pe|) para los diferentes esquemas de interpolación.

Esquema numérico	A(|Pe|)
Centrado	1-0.5|Pe|
Upwind	1
Exponecial	$\frac{\|Pe\|}{e^{\|Pe\|} - 1}$
Híbrido	max [0,(1-0.5|Pe|)]
Ley de Potencia	max [0,(1-0.5|Pe|)]^5

Tabla 3. 2: Función A = (|Pe|)

3.4.5 Malla desplazada o dislocada

La malla desplazada es una técnica muy útil para las soluciones de ecuaciones que involucran la cantidad de movimiento. En esta técnica se generan tres mallas super puestas para almacenar valores de las variables en las fronteras, así como es representado en la **Figura 3.3**.

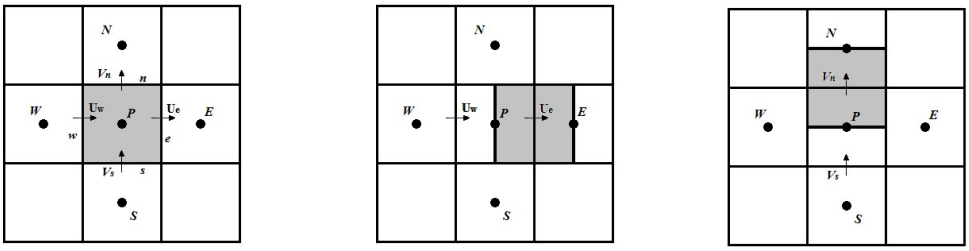

Figura 3. 3: Malla MAC o malla desplazada

-La Malla Principal: que es la que se vio hasta ahora, es una malla centrada en los volúmenes de control que se utiliza para almacenar en el punto P los valores de las variables escalares en forma genérica la variable ϕ, en la práctica los valores escalares como por ejemplo, presión, temperatura, concentración de especies, etc.

-La Malla u: es igual a la malla principal pero dislocada en el sentido positivo del eje x de forma que las caras este y oeste coincidan con los nodos de la malla principal, en ella son almacenados los valores vectoriales en el eje x, principalmente la velocidad horizontal.

-La Malla v: es análoga a la malla u, pero se desplaza de la malla principal en sentido positivo del eje y, de forma que las fronteras sur y norte coincidan con los nodos de la malla principal donde son almacenados los valores verticales de la velocidad y si existiera otro valor vectorial también se almacena su componente vertical.

Dos son las razones por la cual se opta por este tipo de malla. Primero porque para el cálculo de los flujos convectivos no es necesario interpolar las velocidades, dado que estas son calculadas en las caras de los volúmenes de control de la malla principal. Segundo, en el cálculo de los términos fuentes de las ecuaciones de cantidad de movimiento, el valor de la

presión no es interpolado, generando así estabilidad numérica y resultados físicamente realistas.

3.4.6 Acoplamiento de las ecuaciones gobernantes.

La resolución de la ecuación de cantidad de movimiento bidimensional envuelve un cálculo de las dos componentes de la velocidad (u e y) y de la presión (P), así inician las dificultades para el cálculo de tres variables, dado que el término convectivo de la ecuación de cantidad de movimiento es altamente no lineal y que no existe una ecuación de transporte para la presión, con lo que tendría dos ecuaciones apenas para tres variables.

La ecuación de continuidad de la masa y cantidad de movimiento están fuertemente acopladas por las velocidades que son variables de ambas ecuaciones y la presión es apenas término fuente en la ecuación de cantidad de movimiento.

En este caso de estudio en particular se analizan fluidos incompresibles, por lo que no existe una ecuación que acople la presión con la velocidad. Esto se daría en caso de fluidos compresibles por la ecuación de gases perfectos.

Para superar esta dificultad en 1980 el Prof. Patankar nos presentó el algoritmo SIMPLE (Semi-implicit Method for Pressure Linked Equations), en el cual primero se trata de resolver las ecuaciones de cantidad de movimiento, suponiendo un campo de presión estimado para luego tomar esta distribución de velocidad y de presión para corregirlas con la ecuación de continuidad de la masa de manera iterativa hasta que el campo de velocidades satisfaga a ambas ecuaciones conservativas.

La estructura del algoritmo SIMPLE puede ser presentada como sigue:

-Paso 1: Descomponer el término fuente, de manera que la presión aparezca explícitamente, así:

$$b^u = -A_e(P_E - P_P) + b_1^u \tag{3.19a}$$

$$b^v = -A_n(P_N - P_P) + b_1^v \tag{3.19b}$$

donde A_i representa a la superficie de la cara i del volumen de control A partir de ahora la ecuación de cantidad de movimiento puede ser escrita de la siguiente manera en coeficientes agrupados:

$$a_e^u u_e = \sum a_{vizinhos} u_{vizinhos} - A_e(P_E - P_P) + b_1^u \quad (3.20a)$$

$$a_n^v v_n = \sum a_{vizinhos} v_{vizinhos} - A_n(P_N - P_P) + b_1^v \quad (3.20b)$$

-Paso 2: Para resolver las ecuaciones de cantidad de movimiento se necesita estimar el campo de presión inicial **P*** (este * indica que la distribución de presión es una distribución supuesta, no calculada o medida necesariamente). Con él se calcula un campo de velocidad que puede no satisfacer la ecuación de continuidad de la masa, el campo de velocidad obtenido en base a **P*** es identificado como **u*** y **v***. La ecuación **Eq. 3.20** es reescrita de la forma:

$$a_e^u u*_e = \sum a_{vizinhos} u*_{vizinhos} - A_e(P*_E - P*_P) + b_1^u \quad (3.21a)$$

$$a_n^v v*_n = \sum a_{vizinhos} v*_{vizinhos} - A_n(P*_N - P*_P) + b_1^v \quad (3.21b)$$

-Paso 3: Si la presión **P*** no es la correcta para satisfacer la ecuación de conservación de la masa, debe existir una P que lo haga, entonces se propone que **P** sea obtenida a través de una presión de corrección **P'** de la forma:

$$P = P* + P' \quad (3.22)$$

Con esto la velocidad correcta también puede ser interpretada como una velocidad corregida, de forma que existe una velocidad de corrección tal que:

$$u = u* + u' \quad (3.23a)$$

$$v = v* + v' \quad (3.23b)$$

-Paso 4: Sustituyendo las ecuaciones en **Eq. 3.22** y **Eq. 3.23** en la ecuación de cantidad de movimiento **Eq. 3.19** y sustrayendo la ecuación **Eq. 3.21**, se puede llegar a la expresión:

$$a_e^u u'_e = \sum a_{vizinhos} u'_{vizinhos} - A_e(P'_E - P'_P) + b_1^u \quad (3.24a)$$

$$a_n^v v'_n = \sum a_{vizinhos} v'_{vizinhos} - A_n(P'_N - P'_P) + b_1^v \quad (3.24b)$$

-**Paso 5:** De las ecuaciones **Eq. 3.24** se pueden escribir las velocidades de la siguiente manera:

$$u'_e = d_e^u(P'_P - P'_E) \quad (3.25a)$$

$$v'_n = d_n^v(P'_P - P'_N) \quad (3.25b)$$

donde d_e^u y d_n^v son coeficientes conforme con los algoritmos que se utilizarán.

Para el desarrollo de este trabajo fue escogido el algoritmo SIMPLE. Para obtener las correcciones de velocidades se desprecia por completo el término de sumatoria de los elementos vecinos. Y así los coeficientes d_e^u y d_n^v pueden escribirse de la siguiente manera:

$$d_e^u = A_e / a_e^u \quad (3.26a)$$

$$d_n^v = A_n / a_n^v \quad (3.26b)$$

La aproximación realizada es este ponto es válida cuando la solución tiende a converger. Esta es la razón por la cual el algoritmo SIMPLE requiere una buena aproximación inicial y se recomienda desacelerar la corrección de presión ya que esta inicialmente por estas ecuaciones está sobreestimada.

-Paso 6: Al final es necesario calcular el término de corrección de presión P' y para eso es necesario volver a la ecuación de continuidad de la masa **Eq. 3.3** e integrarla en el volumen de control de la malla principal:

$$\frac{\rho_P - \rho_P^0}{\Delta t}\Delta x \Delta y + [(\rho u)_e - (\rho u)_w]\Delta y + [(\rho v)_n - (\rho v)_s]\Delta x = 0 \quad (3.27)$$

Tomando la ecuación **Eq. 3.27** para poder expresarla en términos de la presión corregida y de las ecuaciones **Eq. 3.23** y **Eq. 3.25** y utilizando la misma notación utilizada anteriormente de coeficientes agrupados, resulta:

$$a_P P'_P = a_E P'_E + a_W P'_W + a_N P'_N + a_S P'_S + b \tag{3.28}$$

donde:

$$a_E = \rho_e d_e^u \Delta y \tag{3.29a}$$
$$a_E = \rho_e d_e^u \Delta y \tag{3.29b}$$
$$a_N = \rho_n d_n^v \Delta x \tag{3.29c}$$
$$a_S = \rho_s d_s^v \Delta x \tag{3.29d}$$
$$a_P = a_E + a_W + a_N + a_S \tag{3.29e}$$
$$b = \frac{\rho_P^0 - \rho_P}{\Delta t}\Delta x \Delta y + [(\rho u *)_w - (\rho u *)_e]\Delta y + [(\rho v *)_s - (\rho v *)_n]\Delta x \tag{3.29f}$$

El término fuente en la **Eq. 3.29f** es igual a la ecuación de continuidad en términos de la velocidad estimada. Por lo que un parámetro del proceso es que este debe ir tendiendo a cero.

-Paso 7: Cuando la matriz de la presión de corrección fue calculada, es posible calcular las velocidades corregidas, tomar estas como aproximación inicial e iniciar de nuevo un siguiente proceso iterativo.

3.4.7 Algoritmos variantes al algoritmo SIMPLE

Para conseguir acoplar numéricamente de las ecuaciones de conservación de conservación de masa, cantidad de movimiento y presión, en ausencia de la ecuación de

estado de gases. Esto es para fluidos incompresibles. Existen varios algoritmos alternativos al algoritmo SIMPLE, en este trabajo hablaremos de tres de ellos.

-ALGORITMO SIMPLER: el profesor Patankar en el año 1980 propone este algoritmo, que fue el primer método propuesto como alternativa al SIMPLE. Este algoritmo alternativo presenta como principal característica su mejora en cálculo de la matriz del campo de presión. Sin embargo, según Anderson en 1984, este algoritmo alternativo representa una cantidad superior de cálculos envueltos, cerca de un 30% por sobre el SIMPLE. Con esto también se reduce la velocidad de convergencia.

-ALGORITMO SIMPLEC: como una modificación al algoritmo SIMPLE, este algoritmo fue propuesto por Van Doormaa y Raithby en 1984, Este algoritmo utiliza los mismos conceptos y pases que su predecesor el algoritmo SIMPLE, la diferencia radica únicamente en los términos de la **Eq. 3.26** variando los valores de d_e^u y d_n^v para la corrección de la velocidad y de la presión. En este algoritmo no es necesario una aproximación inicial muy buena para lograr la convergencia, basta con desacelerar la corrección del campo de presión.

-ALGORITMO PISO: este algoritmo fue desarrollado por Issa en 1986, como un procedimiento para acoplar la presión y velocidad, el cual al principio era visto como un algoritmo para una solución no iterativa de problemas de flujos compresibles transitorios. Pero posteriormente fue adoptado para problemas de régimen permanente con un proceso iterativo. Este algoritmo utiliza un paso para predecir los valores de las variables de interés y luego dos pasos para corregir. Según Xamán en 2015, este algoritmo puede ser visto con una modificación del algoritmo SIMPLE con la adición de un posterior paso correctivo.

3.5 Condiciones de contorno y su tratamiento numérico

Todo problema para su planteamiento requiere unas condiciones iniciales y una definición, a esta definición de las fronteras llamamos condiciones de contorno. Para correr una simulación numérica es necesario predefinir estos valores, tanto las condiciones iniciales

de los campos de presión, velocidad y temperatura como ejemplo, y las condiciones de contorno.

Las condiciones de contorno más conocidas en métodos numéricos son: La Condición de Dirichlet también conocida como condición de primer tipo o primera clase y La Condición de Neumann también conocida como segundo tipo o segunda clase.

3.5.1 Condición de Dirichlet.

La condición de contorno de Dirichlet es conocida así en honor a Johann Peter Gustav Lejeune Dirichlet (1805-1859), quien planteó la condición como un problema por primera vez, también conocida como condición de primer tipo o de primera clase. Cheng 2005

En este tipo condición de contorno, los nodos del mallado espacial que se encuentran esta condición toman un valor constante, independiente a los valores de los nodos vecinos. Por ejemplo, una pared de temperatura constante, la temperatura en ese nodo frontera permanece invariante e independiente a la temperatura del nodo vecino.

Para el tratamiento en el algoritmo SIMPLE es necesario introducir este hecho en el tratamiento de los coeficientes agrupados para lograr dos cosas, la primera que independiente a la iteración el valor de la frontera permanezca constante y segundo, cuando nodo vecino a la frontera entre a ser tratado por la ecuación de coeficientes agrupados, pueda tomar los valores del nodo frontera, porque aunque el nodo frontera sea independiente al nodo vecino, esto no ocurre en sentido contrario, el nodo vecino necesariamente es dependiente del valor en la frontera.

Quedan así los valores para la ecuación de coeficientes agrupados en los nodos fronteras:

$$a_P\phi_P = a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S + b \tag{3.30}$$

donde se deben imponer los valores:

$$a_E = a_W = a_N = a_S = 0 \tag{3.31a}$$

$$a_P = 1 \tag{3.31b}$$

$$b = \phi_{especificado} \tag{3.31c}$$

Sustituyendo los valores de la ecuación **Eq. 3.31** en la ecuación **Eq. 3.28** se obtiene:

$$a_P = \phi_{especificado} \tag{3.32}$$

donde $\phi_{especificado}$ es el valor que deseamos asignar a la frontera.

3.5.2 Condición de Neumann.

La condición de Neumann o condición de segundo tipo o de segunda clase, es llamada así en honor al matemático Carl Neumann quien planteó esta condición como solución a problemas de contorno para derivadas parciales y/o ecuaciones diferenciales ordinarias. Chang 2005.

En este tipo de condición no se mantiene constante un valor específico del nodo, sino se mantiene constante una taza da variación, es decir, la derivada de un elemento se mantiene constante. Por ejemplo, el calor entrante o saliente a una pared.

Para aplicar al algoritmo SIMPLE tomemos una constante *A* como la taza de variación impuesta de la variable ϕ y resulta así:

$$\frac{\partial \phi}{\partial n} = A \tag{3.33}$$

haciendo una aproximación lineal de la ecuación Eq. 3.33 y tomando como ejemplo la frontera sur del dominio podemos escribir la siguiente ecuación.

$$\frac{\phi_N - \phi_P}{\delta y} = A \tag{3.34}$$

Considerando la ecuación **Eq. 3.30**, el valor de los coeficientes debe ser:

$$a_E = a_W = a_S = 0 \tag{3.35a}$$
$$a_P = a_N = 1 \tag{3.35b}$$
$$b = A\delta y \tag{3.35c}$$

De forma que substituyendo los valores descriptos en **Eq. 3.35** na **Eq. 3.30** el valor de la variable en la frontera es:

$$\phi_N - \phi_P = A\delta y \tag{3.36}$$

O procedimiento es análogo para cualquier frontera del dominio.

3.6 Validación del código.

Para poder afirmar con propiedad que un cierto método funciona es necesario recurrir a la comparación científica. Ocurre lo mismo con un código numérico que se ha escrito. Es necesario recurrir a la comparación para poder determinar si este código es efectivo o no, si puede o no predecir con una presión aceptable los fenómenos físicos de estudio. Para el caso de este trabajo se ha tomado como fenómeno de estudio la convección natural, es el fenómeno que se buscas predecir mediante simulaciones numéricas. Por lo que es necesario validar la capacidad del código de solucionar problemas de conducción de calor pura y convección.

Para validar estos procedimientos se siguió como guía al profesor Xamán en 2015, quien sugiere realizar la validación con cuatro problemas y comparar resultados con problemas de la literatura:

-Problema 1: iniciar por resolver un problema de conducción pura de calor y al compararlo con algún resultado de la literatura sea experimental o analítico, no se recomienda

la comparación en este punto con resultados numéricos pues la precisión que puede conseguir con resultados experimentales y analíticos son bastante buenas. Al lograr validar este problema, estamos validando no solamente los casos de conducción de calor, sino todos los términos difusivos genéricos del Algoritmo SIMPLE, en otra manera de decirlo, también se encuentran validados los términos difusivos de todas las demás ecuaciones de transporte.

-Problema 2: habiendo sido validada la capacidad del código de resolver problemas difusivos, es necesario validar también los términos convectivos, esto se logra resolviendo un problema de convección-difusión, pero como aún no podemos acoplar la presión, es necesario que el problema sea netamente térmico y con los parámetros de fluidos conocidos, es decir, el campo de velocidades debe ser conocido. Con esto quedan validados la capacidad del código de calcular los valores de todos los términos tanto difusivos como convectivos de las distintas ecuaciones de transporte que se aplican a la forma general descripta en la **Eq. 3.6**. Y no únicamente al caso térmico.

-Problema 3: para este problema lo necesario es acoplar la presión a la velocidad, a través de la ecuación de coeficientes agrupados del algoritmo SIMPLE y la corrección de la presión, para validar este problema, Xamán 2015 y otros autores sugieren resolver el problema conocido como **Driven-Cavity Problem** o cavidad de pared deslizante. Y así validar el algoritmo SIMPLE por completo.

-Problema 4: por último, es necesario lograr el acople del campo de temperatura con el campo de velocidades y al tratarse de fluidos incompresibles es necesario recurrir a la aproximación de Boussinesq.

En el siguiente capítulo será presentado planteamiento de modelo físico, modelo matemático y de discretización para cada problema. Y la posterior comparación de los resultados del código con los resultados de la literatura para su validación.

4 VALIDACIÓN DE CÓDIGO NUMÉRICO

En este capítulo se desarrolla el proceso de validación de ambos códigos desarrollados, el código serie o monoprocesador escrito en C++ y el código multiprocesador o paralelo escrito mediante la herramienta PETSc. El proceso para ambos códigos es el mismo puesto que de hecho ambos códigos deberían resolver satisfactoriamente con una precisión aceptable problemas de mecánica de fluidos aplicando y mismo algoritmo matemático.

Para esto se respetan los pasos descriptos en el capítulo anterior de realizar la validación mediante la comparación con cuatro problemas de la literatura de manera ordenada. En el capítulo se desarrolla primero el código monoprocesador o serie, debido a que cronológicamente del desarrollo este fue el primero en escribirse.

4.1 Conducción de calor en régimen transitorio

El objetivo de desarrollar un ejercicio de conducción de calor, utilizando el código es el de dar validez al proceso de cálculo del algoritmo, por el cual términos difusivos de la ecuación general de transporte son estimados. Es decir, que, con una semejanza entre resultados de la literatura y resultado práctico de haber aplicado el código, podremos concluir parcialmente qué código puede ser utilizado con una precisión aceptable para predecir otros problemas semejantes que envuelvan a la difusión pura en la aplicación de la ecuación general de transporte.

Para esto se requiere un problema resuelto de conducción de calor bidimensional en estado transitorio de la literatura aplicada a un material uniforme con varias condiciones de contorno. En este caso en particular se escogió un ejercicio resuelto del libro de transferencia de calor de Holman en 1999.

4.1.1 Modelo Físico

El modelo físico presentado trata de una franja de cerámica rectangular de 1 x 2 cm embutida en un material de alta conductividad térmica con paredes isotérmicas de temperatura constante, la pared inferior de la cerámica es adiabática y la superior está expuesta a la convección de un líquido con un coeficiente de transferencia de calor constante y una temperatura del líquido constante.

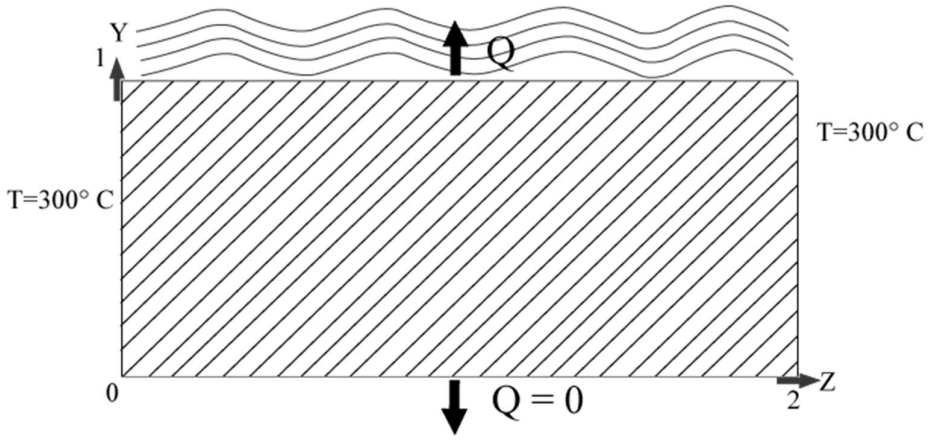

Figura 4. 1: Modelo físico de problema de transferencia de calor en régimen transitorio.

4.1.2 Modelo Matemático

Para el modelo matemático la ecuación que gobierna el fenómeno es la ecuación diferencial de conducción de calor.

$$\frac{\partial(\rho T)}{\partial t} = \frac{\partial}{\partial x}\left(\frac{\lambda}{C_P}\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\lambda}{C_P}\frac{\partial T}{\partial y}\right) \tag{4.1}$$

Con las siguientes condiciones de contorno para t > 0:

$$\left[\frac{\partial T}{\partial y}\right]_{y=0} = 0 \quad (4.2a)$$

$$\left[\frac{\partial T}{\partial y}\right]_{y=1} = h_{conv} * A_s(T(x,1) - T_\infty);\ para\ x\ \epsilon[0,2] \quad (4.2b)$$

$$T(0,y) = 300°C;\ para\ y\ \epsilon[0,1] \quad (4.2c)$$

$$T(2,y) = 300°\text{C};\ para\ y\ \epsilon[0,1] \quad (4.2d)$$

Las condiciones iniciales del problema son:

$$T(x,y) = 300°C;\ para\ t = 0 \quad (4.3a)$$

$$T_\infty = 50°c \quad (4.3b)$$

$$h_{conv} = 200\ \frac{W}{m^2\ °C} \quad (4.3c)$$

$$\rho = 1600\ \frac{Kg}{m^3} \quad (4.3d)$$

$$c = 0.8\ \frac{KJ}{Kg\ °C} \quad (4.3e)$$

$$\Delta t = 0.2\ s \quad (4.3f)$$

$$k = 3.0\ \frac{W}{m°C} \quad (4.3g)$$

4.1.3 Discretización

Aplicando la metodología descripta en el capítulo anterior para volúmenes finitos, la ecuación de conducción y la integral del volumen de control usando esquema implícito en el tiempo podemos reescribir la ecuación en forma de componentes agrupados de la siguiente manera:

$$a_P T_P = a_E T_E + a_W T_W + a_N T_N + a_S T_S + b \quad (4.4)$$

Describiendo los términos de la siguiente manera:

$$a_E = \frac{\Gamma_e}{\delta x_{PE}} \delta y \tag{4.5a}$$

$$a_W = \frac{\Gamma_e}{\delta x_{WP}} \delta y \tag{4.5b}$$

$$a_N = \frac{\Gamma_n}{\delta y_{PN}} \delta x \tag{4.5c}$$

$$a_S = \frac{\Gamma_s}{\delta y_{SP}} \delta x \tag{4.5d}$$

$$a_P^0 = \rho^0 \frac{\Delta x \Delta y}{\Delta t} \tag{4.5e}$$

$$a_P = a_E + a_W + a_N + a_S + \rho_P \frac{\Delta x \Delta y}{\Delta t} \tag{4.5f}$$

$$b = a_P^0 T_P^0 \tag{4.5g}$$

4.1.4 Resultados de la validación

Se realizaron las simulaciones con ambos códigos teniendo en cuenta los parámetros descriptos en la **Sección 4.1.2** y comparadas con los resultados de la literatura que fueron obtenidos mediante una analogía de circuitos. Teniendo la siguiente table de resultado.

Punto		t0=0s	t1=2s	t2=4s	t3=6s	t4=8s	t5=10s	t6=12s
1	Central superior							
	Lit. Holman	300,000	268,750	253,130	245,310	239,480	235,350	231,970
	CODE Mono	300,000	271,839	254,588	243,496	235,059	228,699	223,630
	CODE Petsc	300,000	277,835	262,792	252,309	244,816	239,330	235,223
2	Central central							
	Lit. Holman	300,000	300,000	294,140	287,550	282,380	277,790	273,950
	CODE Mono	300,000	296,298	290,900	284,947	279,055	273,528	268,498
	CODE Petsc	300,000	297,031	293,056	288,975	285,171	281,776	278,811
3	Central inferior							
	Lit. Holman	300,000	300,000	300,000	297,800	293,960	290,080	286,320
	CODE Mono	300,000	299,384	298,010	295,965	293,420	290,560	287,546
	CODE Petsc	300,000	299,547	298,607	297,287	295,721	294,030	292,305
Diferencia %								
1	Mono/Holman	0,00%	1,15%	0,58%	0,74%	1,85%	2,83%	3,60%
	Petsc/Holman	0,00%	3,38%	3,82%	2,85%	2,23%	1,69%	1,40%
2	CODE Mono	0,00%	1,23%	1,10%	0,91%	1,18%	1,53%	1,99%
	CODE Petsc	0,00%	0,99%	0,37%	0,50%	0,99%	1,44%	1,77%
3	CODE Mono	0,00%	0,21%	0,66%	0,62%	0,18%	0,17%	0,43%
	CODE Petsc	0,00%	0,15%	0,46%	0,17%	0,60%	1,36%	2,09%
Diferencia % máxima		**0,00%**	**3,38%**	**3,82%**	**2,85%**	**2,23%**	**2,83%**	**3,60%**

Tabla 4. 1: Comparación de resultados obtenidos.

Los puntos comparados son los puntos medios de la cerámica mostrado en la **Figura 4.2**, para las simulaciones realizadas se utilizaron mallas estructuradas. En el caso del mono procesador se utilizaron mallas de 7x7, 9x9, 11x11 y 13x13 nodos, únicamente con el objeto de verificar la congruencia de resultados, utilizando números impares para coincidir con la posición del nodo P en la misma posición que sugiere el ejercicio propuesto por la literatura.

En cuanto a las simulaciones en paralelos, los resultados no varían mucho en cuanto a los resultados de bajo número de nodos, pero decidí correr el programa con malla de 45x45, 55x55 y 101x101 únicamente con el objeto de verificar también la congruencia de resultados. Se verifica que para ambos casos a diferencia porcentual máxima es menos del 4% por lo que es posible considerar como una predicción aceptable y fiable.

4.2 Convección y difusión con velocidades conocidas

El objeto de este problema es el de validar los términos convectivos en la ecuación general de transporte. Representa el paso previo al acople de temperatura para el algoritmo SIMPLE. Para realizar esta validación se escogió el problema conocido como "Problema de Smith-Hutton" planteado por Smith y Hutton en 1982, este problema es planteado por algunas universidades como paso de validación para los esquemas de resolución del término convectivo de la ecuación general de transporte, un ejemplo de lo mencionado es el Centro Tecnológico de transferencia de calor de la Universidad Politécnica de Cataluña, que se agrega a referencia como CTTC. Donde el flujo interno se mueve de manera senoidal.

4.2.1 Modelo Físico

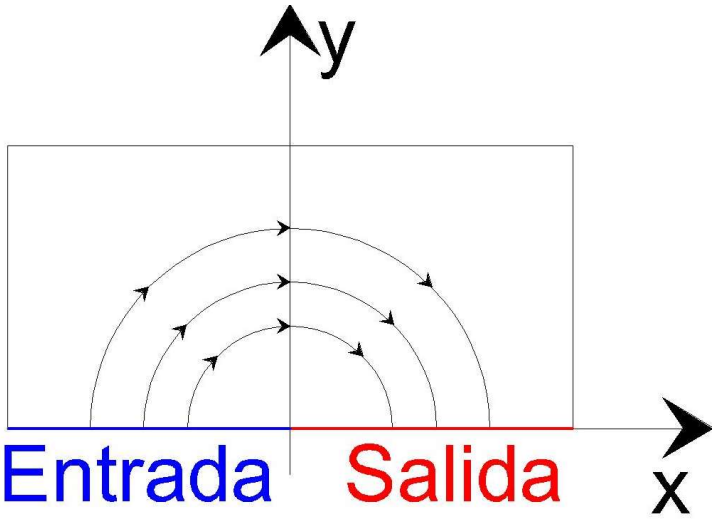

Figura 4. 2: Modelo físico del problema de Smith-Hutton

El modelo expresa una cavidad rectangular con velocidades senoidales conocidas.

$$u(x,y) = 2y(1-x^2) \tag{4.6a}$$

$$v(x,y) = -2x(1-y^2) \tag{4.6b}$$

4.2.2 Modelo Matemático

La ecuación que gobierna el fenómeno es la ecuación general de transporte, aplicada para convección conducción, Eq. 3.6.

$$\frac{\partial(\rho\phi)}{\partial t} + \vec{\nabla}.\left(\rho\vec{V}\phi\right) = \vec{\nabla}.\left(\Gamma\vec{\nabla}\phi\right) + S$$

Con las siguientes condiciones de contorno:

$$\left[\phi = 1 + \tanh\left((2x+1)\alpha\right)\right]_{y=0} \;;\; para\ toda\ "x"\ entre[-1{,}0] \tag{4.7a}$$

$$\left[\frac{\partial\phi}{\partial y}\right]_{y=0} = 0;\ para\ x\ \epsilon[0{,}1] \tag{4.7b}$$

$$[\phi = 1 - \tanh(\alpha)\,]_{x=-1} \;;\; para\ toda\ "y"\ entre[0{,}1] \tag{4.7c}$$

$$[\phi = 1 - \tanh(\alpha)\,]_{x=1} \;;\; para\ toda\ "y"\ entre[0{,}1] \tag{4.7d}$$

$$[\phi = 1 - \tanh(\alpha)\,]_{x=1} \;;\; para\ toda\ "x"\ entre[-1{,}1] \tag{4.7e}$$

4.2.3 Discretización

Escribimos la ecuación que gobierna el fenómeno de forma de coeficientes agrupados como ya se vio en la metodología del capítulo anterior, teniendo así:

$$a_P\phi_P = a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S \tag{4.8}$$

donde los términos internos son discretizados como

$$a_E = \left(\frac{\Gamma_e}{\delta x_{PE}}\right)\Delta y - (\rho u)_e \alpha_e \tag{4.9a}$$

$$a_W = \left(\frac{\Gamma_w}{\delta x_{WP}}\right)\Delta y + (\rho u)_w (1 - \alpha_w) \tag{4.9b}$$

$$a_N = \left(\frac{\Gamma_n}{\delta x_{PN}}\right)\Delta x - (\rho v)_n \alpha_n \tag{4.9c}$$

$$a_S = \left(\frac{\Gamma_s}{\delta x_{SP}}\right)\Delta x + (\rho v)_s (1 - \alpha_s) \tag{4.9d}$$

$$a_P = a_E + a_W + a_N + a_S + \rho(u_e - u_w + v_s - v_n) \tag{4.9e}$$

4.2.4 Resultados de la validación

Para la validación de ambos códigos con el resultado de la literatura se tomaron las siguientes hipótesis: $\alpha = 10$ y $\frac{\rho}{\Gamma} = (10; 10^3; 10^6)$. Los resultados de la literatura se expresan en la **Tabla 4.2**.

Posición en X	ρ/Γ=10	ρ/Γ=10exp3	ρ/Γ=10exp6
0,0	1,9890	2,0000	2,0000
0,1	1,4020	1,9990	2,0000
0,2	1,1460	1,9997	2,0000
0,3	0,9460	1,9850	1,9990
0,4	0,7750	1,8410	1,9640
0,5	0,6210	0,9510	1,0000
0,6	0,4800	0,1564	0,0360
0,7	0,3490	0,0010	0,0010
0,8	0,2270	0,0000	0,0000
0,9	0,1110	0,0000	0,0000
1,0	0,0000	0,0000	0,0000

Tabla 4. 2: Resultados de la literatura del problema Smith-Hutton

En las figuras siguientes se comparan gráficamente los resultados de la literatura con las simulaciones para ambos códigos.

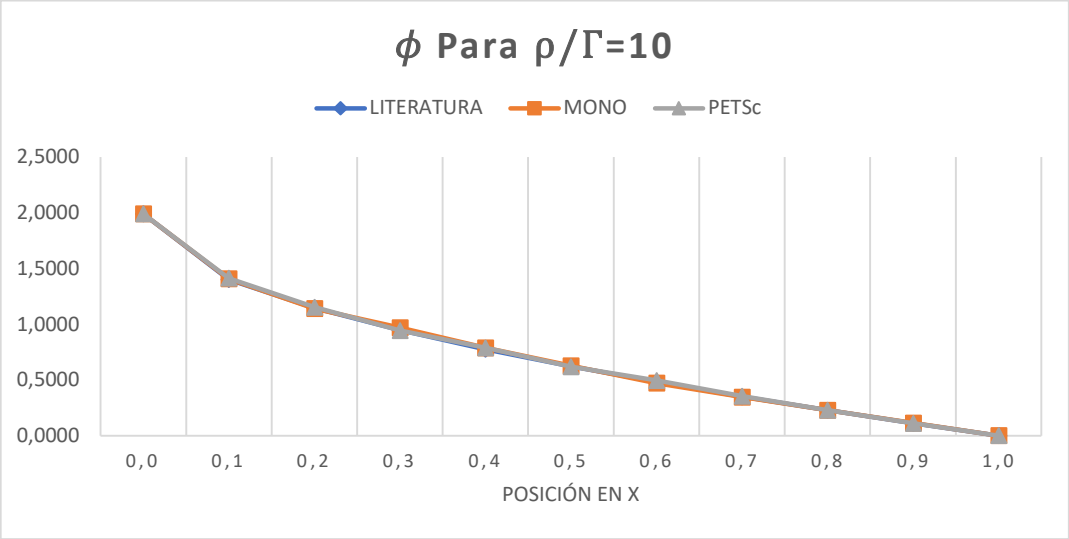

Figura 4. 3: Resultados para ρ/Γ=10.

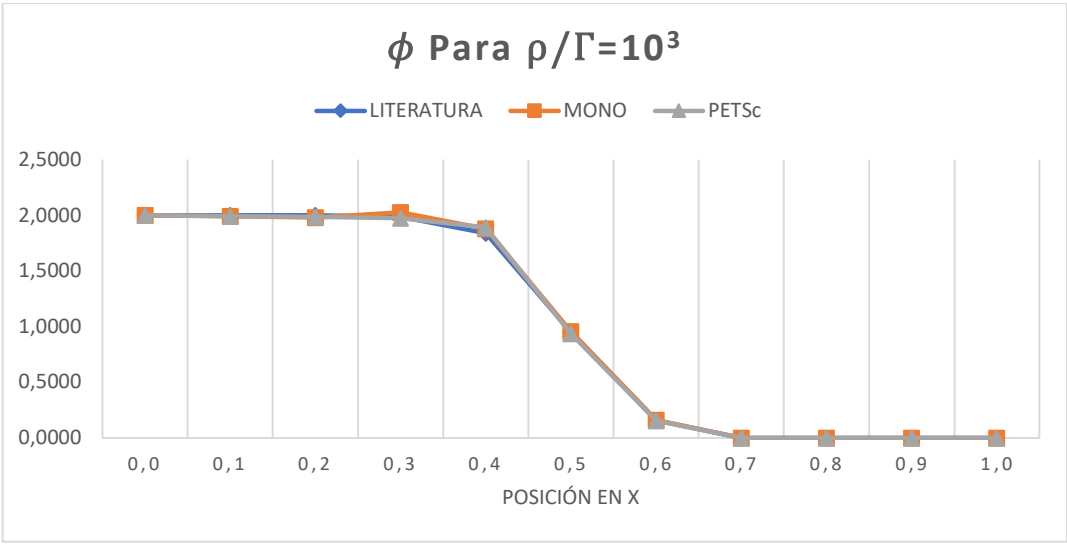

Figura 4. 4: Resultados para ρ/Γ=(10)³.

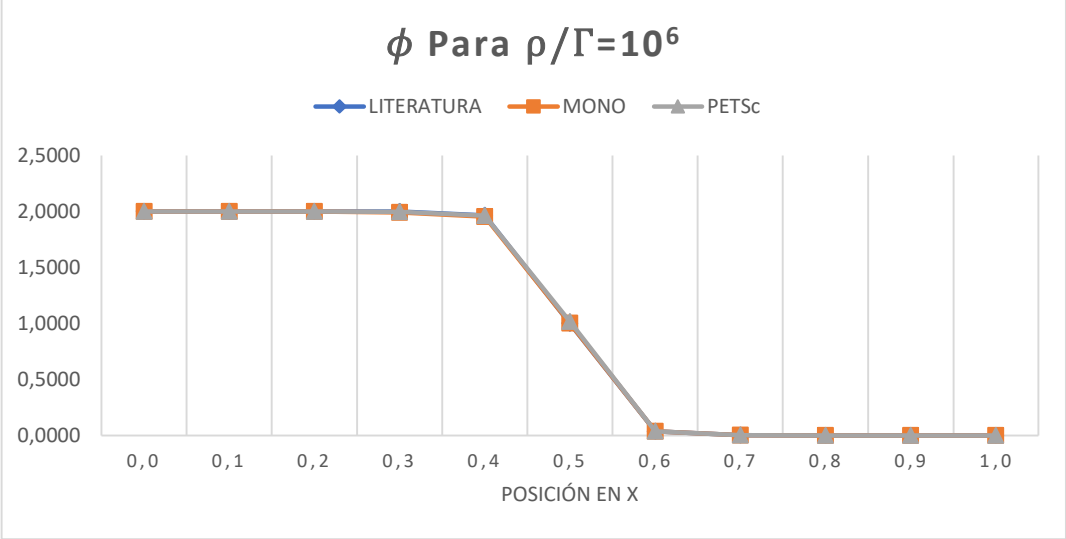

Figura 4. 5: Resultados para ρ/Γ=(10)⁶.

Las figuras muestran buenos resultados para la validación y las diferencias porcentuales para el código monoprocesador alcanza un 2.69% como máximo y para el PETSc 2.79%. Con esto podemos concluir que los términos convectivos de los códigos pueden predecir fenómenos físicos semejantes.

4.3 Acople de presión – Driven cavity

El objeto de este problema es el de validar el acople de la presión a ecuación de cantidad de movimiento en régimen permanente, estimando así los campos de velocidades y de presión de manera a llegar al equilibrio en la conservación de masa y cantidad de movimiento.

Teniendo así una relación entre la presión y la velocidad, conseguida mediante el algoritmo SIMPLE. Para este problema se considera un flujo laminar, permanente e incompresible.

4.3.1 Modelo Físico

El modelo físico consiste en una cavidad cuadrada cerrada, donde las paredes laterales e inferior se encuentra en una condición de no deslizamiento, es decir las velocidades son iguales a 0. Mientras que la pared superior es una placa infinita que se desliza con una velocidad constante U_o, como lo muestra la figura.

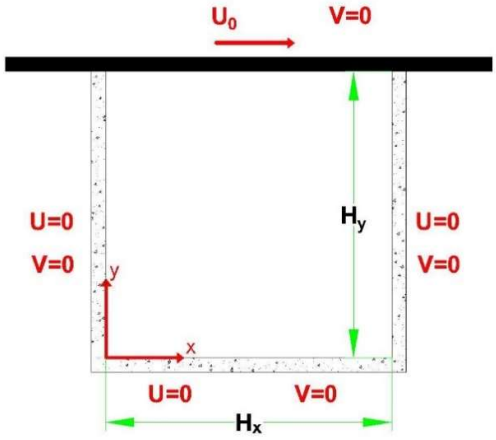

Figura 4. 6: Problema de Pared Deslizante.

4.3.2 Modelo Matemático

Las ecuaciones que gobiernan este problema de estudio son la ecuación de conservación de la masa y conservación de cantidad de movimiento tanto para los ejes X e Y. Las condiciones de contorno estas dadas por velocidades definidas.

$$u(x, H_y) = U_0 \quad para\ toda\ "x"\ entre [0, H_x] \tag{4.10a}$$

$$u(x, 0) = 0 \quad para\ toda\ "x"\ entre [0, H_x] \tag{4.10b}$$

$$u(0,y) = u(H_x,y) = 0 \quad para\ toda\ "y"\ entre\ [0,H_y] \tag{4.10c}$$

$$v(0,y) = v(H_x,y) = 0 \quad para\ toda\ "y"\ entre\ [0,H_y] \tag{4.10d}$$

$$v(x,0) = v(x,H_y) = 0 \ \ para\ toda\ "y"\ entre\ [0,H_x] \tag{4.10e}$$

4.3.3 Discretización

Las ecuaciones de cantidad de movimiento son integradas en un volumen de control usando un esquema implícito en el tiempo. El esquema numérico de interpolación utilizado es el UPWIND con la notación de conjuntos agrupados Y el algoritmo SIMPLE para el acople del campo de presión y velocidad. Como se explica en la metodología en el capítulo anterior.

4.3.4 Resultados

Para la comparación de los resultados son consultadas tres fuentes, Xáman en 2015, quien sugiere tomar una malla uniforme de nodo centrado en el volumen de controlo con nodo en contacto en la frontera. Fue utilizada una malla de 100x100 siguiendo estas sugerencias. Zienkiewicz en 2006 replica el problema en elementos finitos.

Existen varias soluciones numéricas para este problema, sin embargo, soluciones experimentales para este caso de estudio escasean. Por lo cual se selecciona el trabajo experimental del profesor Ghia en 1982 para comparar los resultados.

Para esto en el **Código Monoprocesador** fue calculada la velocidad vertical en el plano horizontal medio (V=? para $y=H_y/2$) y la velocidad horizontal en el plano vertical medio (U=? para $x=H_x/2$). Para números de Reynolds de 100 y 1000. En los siguiente 4 figuras podemos ver una conciencia aceptable entre predicción por simulación numérica y resultado experimental.

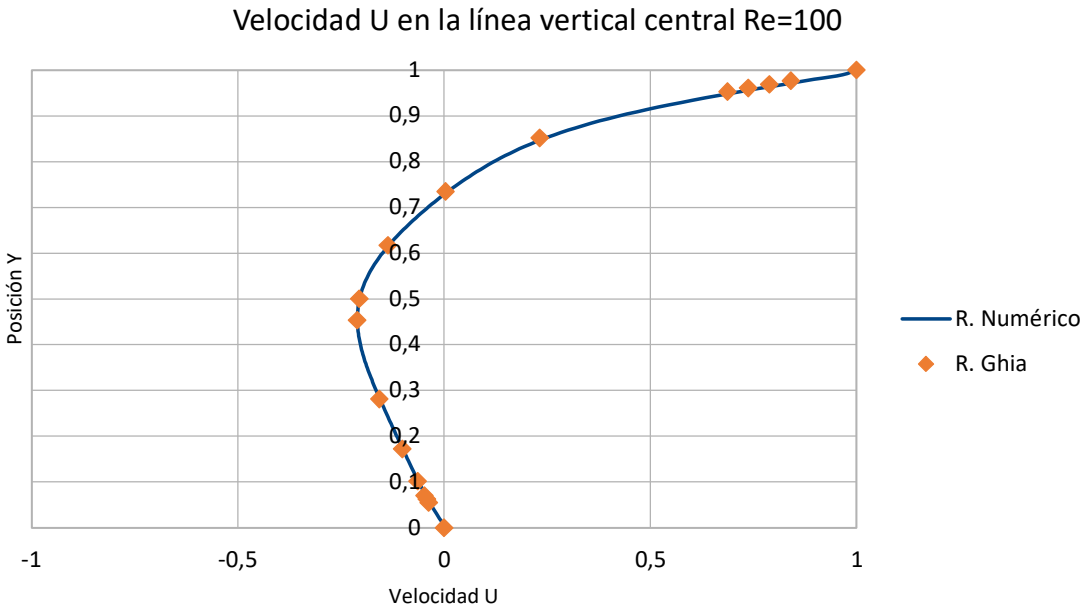

Figura 4. 7: Comparación de resultados Velocidad U en x=H_x⁄2 para Re=100

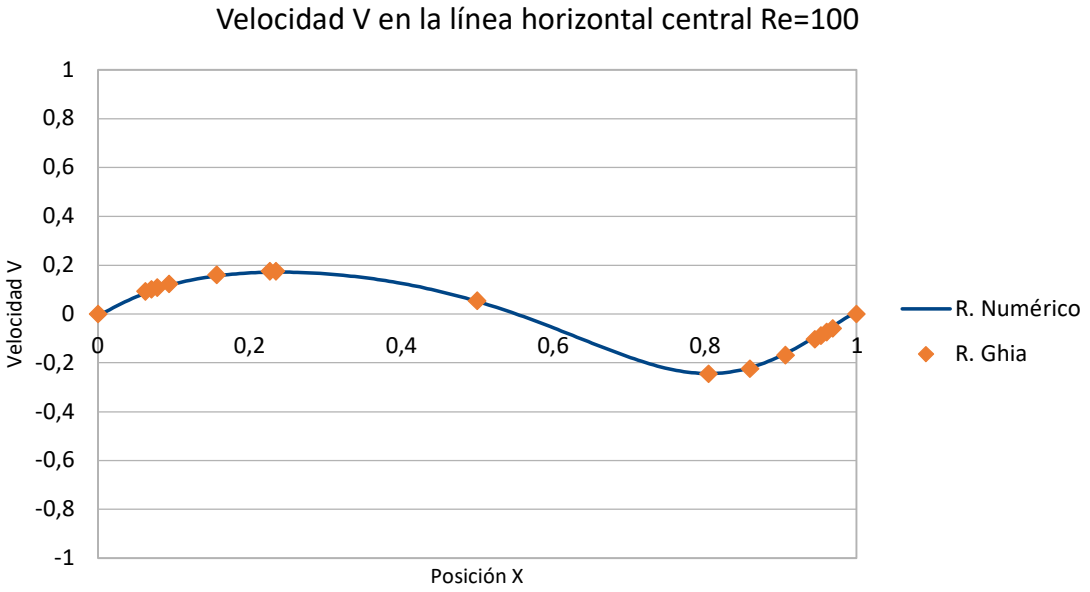

Figura 4. 8: Comparación de resultados Velocidad V en y=H_y⁄2 para Re=100

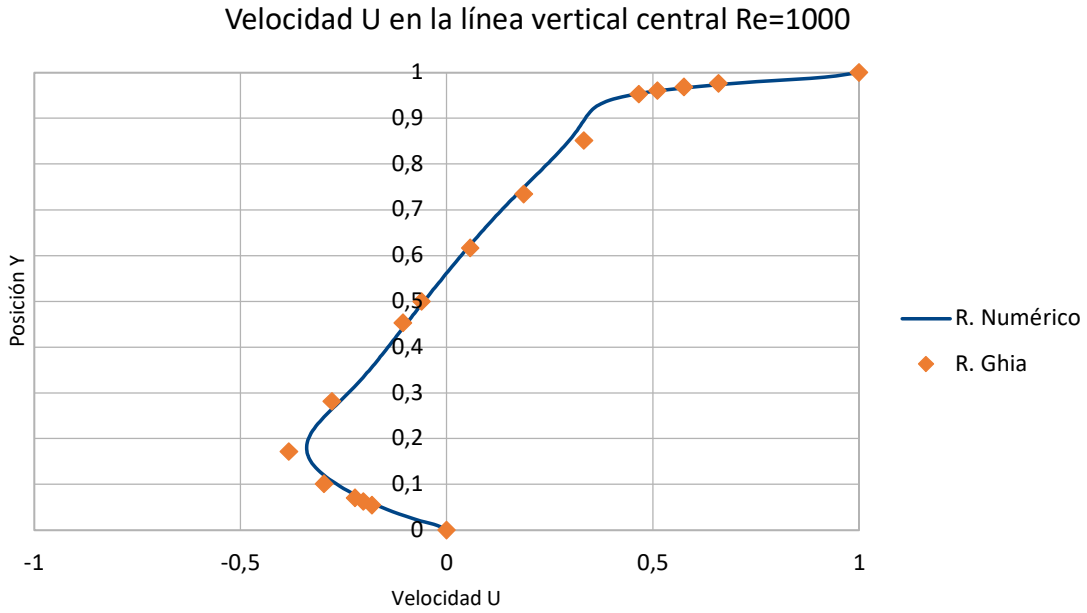

Figura 4. 9: Comparación de resultados Velocidad U en x=H_x⁄2 para Re=1000

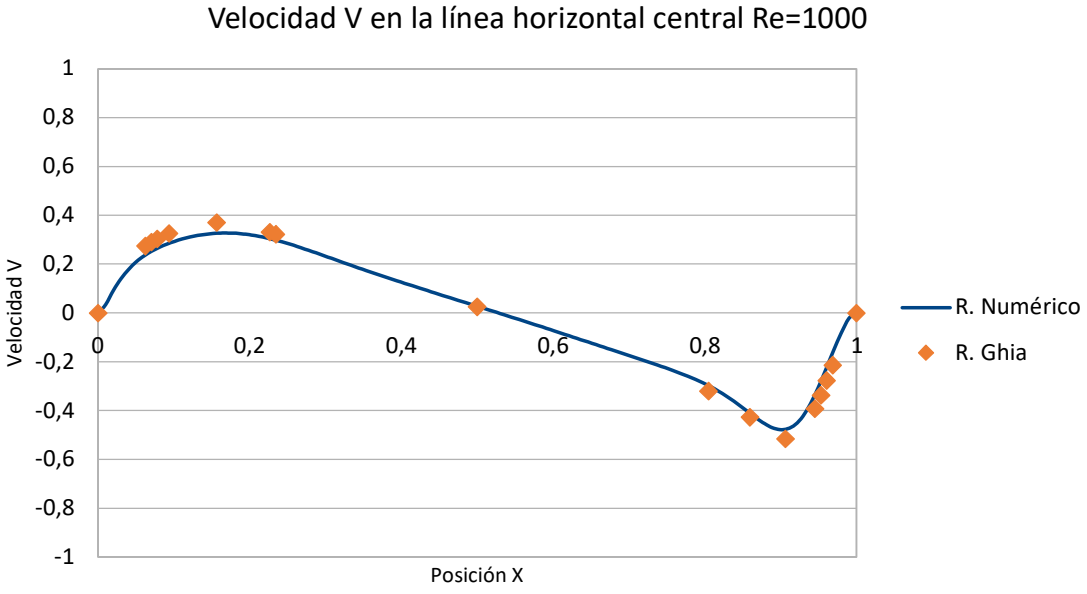

Figura 4. 10: Comparación de resultados Velocidad V en y=H_y⁄2 para Re=1000

Con esto podemos dar por validado el funcionamiento de nuestro código monoprocesador para el algoritmo SIMPLE.

En cuanto a la paralelización del código, no se ha logrado el acople de presión utilizando la herramienta PETSc. Esto responde a que la estructura de solución de las matrices de la herramienta PETSc, no admite que una única iteración para el campo de velocidad, mientras solicitamos cerca de doscientas iteraciones para el campo de presión. Pues no admite una FUNCIÓN SOLUCIÓN anidada en la estructura de otra FUNCIÓN SOLUCIÓN, ya que las funciones vienen predeterminadas.

4.4 Cavidad sometida a calor diferencial

El problema trata del acople de las ecuaciones de transporte de conservación de cantidad de movimiento, conservación de la masa y conservación de la energía. Donde el movimiento de la masa de aire se da por la diferencia de temperatura entre sus paredes verticales que calientan diferencialmente una cavidad cuadrada totalmente cerrada. Para ello se considera al aire un fluido incompresible de propiedades constantes para todo el dominio, a excepción de la densidad en campo de la temperatura, pues a través de la aproximación de Boussinesq, que suma un término de empuje en la ecuación de cantidad de movimiento. Se calculan los campos de velocidades, presión, temperatura y líneas de corriente. Para validar el acople de la ecuación de conservación de la energía al proceso iterativo en desarrollo.

4.4.1 Modelo Físico

El modelo Físico consiste en una masa de aire confinada en una cavidad cuadrada bidimensional de paredes impermeables. Donde la pared izquierda es mantenida de forma isotérmica a una temperatura contante (T_h), la pared derecha es mantenida de forma

isotérmica a una temperatura constante (T_c) y las paredes horizontales son adiabáticas, por lo que no permiten la fuga de calor. Como se muestra en la siguiente figura.

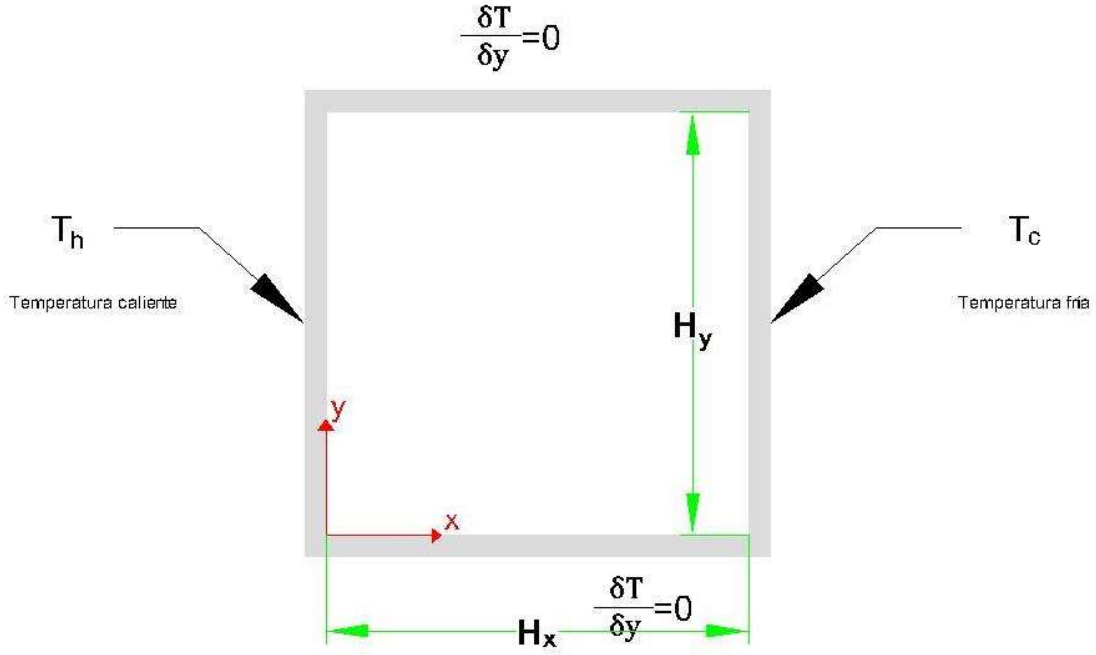

Figura 4. 11: Modelo Físico de cavidad cuadrada con calentamiento diferencial

4.4.2 Modelo Matemático

El fenómeno en estudio es gobernado con las ecuaciones de conservación de la masa, ecuación conservación de la cantidad de movimiento en los ejes X e Y y la ecuación de conservación de la energía.

Reescribiendo la ecuación de conservación de la cantidad de movimiento en el eje Y (Navier Stokes) y agregando la aproximación de Boussinesq tenemos.

$$\frac{\partial(\rho v)}{\partial t}+\frac{\partial(\rho u.v)}{\partial x}+\frac{\partial(\rho v.v)}{\partial y}=\frac{-\partial P}{\partial y}+\frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial y}\right)+\rho g\beta(T-T_0) \qquad (4.11)$$

donde β es el coeficiente térmico de expansión volumétrica.

Para las velocidades se define la condición de no deslizamiento en todas las fronteras del dominio:

$$u(x,0) = u(x,H_y) = 0 \quad para\ toda\ "x"\ entre\ [0,H_x] \tag{4.12a}$$

$$u(0,y) = u(H_x,y) = 0 \quad para\ toda\ "y"\ entre\ [0,H_y] \tag{4.12b}$$

$$v(0,y) = v(H_x,y) = 0 \quad para\ toda\ "y"\ entre\ [0,H_y] \tag{4.12c}$$

$$v(x,0) = v(x,H_y) = 0 \quad para\ toda\ "x"\ entre\ [0,H_x] \tag{4.12d}$$

Para a temperatura, tenemos las siguientes condiciones de contorno:

$$\left[\frac{\partial T}{\partial y}\right]_{y=0} = \left[\frac{\partial T}{\partial y}\right]_{y=H_y} = 0 \tag{4.13a}$$

$$T(0,y) = T_H \quad para\ toda\ "y"\ entre\ [0,H_y] \tag{4.13b}$$

$$T(H_x,y) = T_C \quad para\ toda\ "y"\ entre\ [0,H_y] \tag{4.13c}$$

Para caracterizar a transferencia de calor del sistema se calcula el número de Nusselt (Nu) a lo largo de la pared vertical izquierda, siendo este una taza entre la transferencia de calor por convección y la transferencia de calor por conducción. Se calculan así los números de Nusselt local para la posición (Nu(y)), como el número de Nusselt medio para la posición ($\bar{Nu}(y)$). De la siguiente manera:

$$Nu(y) = \frac{q_{conv}}{q_{cond}} = \frac{\left[\frac{\partial T}{\partial x}\right]_{x=0}}{\frac{T_h - T_C}{H_x}} \tag{4.14a}$$

$$\bar{Nu}(y) = \frac{1}{H_y}\int_{y=0}^{y=H_y} Nu\,(y).dy \tag{4.14b}$$

Para poder comparar los campos de temperatura y velocidad resultados de la simulación con los resultados de la literatura, son definidos los siguientes valores adimensionales:

$$u* = \frac{uH_x\rho C_P}{\lambda} \quad v* = \frac{vH_y\rho C_P}{\lambda} \tag{4.15a}$$

$$x* = \frac{x}{H_x} \quad y* = \frac{y}{H_y} \tag{4.15b}$$

$$T* = \frac{T-T_c}{T_f-T_c} \quad Ra = \frac{gH^3{}_x\beta(T_h-T_c)}{\lambda\nu} \tag{4.15c}$$

4.4.3 Discretización

Las ecuaciones que gobiernan el fenómeno son integradas en un volumen de control utilizando un esquema implícito en el tiempo. La solución es obtenida utilizando una técnica de falso transitorio. Se opta por la utilización del esquema de interpolación UPWIND. Para el acoplamiento de los campos de presión y velocidad es utilizado el algoritmo SIMPLE y para el acople de la ecuación de la energía se utiliza la hipótesis de Boussineq.

4.4.4 Resultados

Este problema fue simulado utilizando una malla estructurada uniforme de 180x180, que fue escogida luego de realizar la simulación con varios números de mallas. Teniendo el menor tiempo de cómputo a buena fiabilidad. Se utilizó un número de Prandtl constante igual a 0.71 y los resultados son comparados mediante las siguientes tablas con los resultados de Xáman en 2015. Se comparan Velocidad horizontal máxima (U_{max}) en el plano medio vertical, Velocidad vertical máxima (V_{max}) en el plano medio horizontal, el número de Nusselt medio ($\bar{Nu}$), máximo (Nu_{max}) y mínimo (Nu_{min}).

Ra= 1E3			
Velocidad		Valores de literatura	diferencia %
Umax	1,37E-01	0,137	0,34%
Vmax	1,38E-01	0,139	0,52%
Nusselt	Simulado		
Medio=	1,12	1,118	0,18%
Maximo=	1,5075	1,508	0,03%
Minimo=	0,69559	0,691	0,66%

Tabla 4. 3: Tabla comparativa de diferencia relativa con la literatura para Ra=10^3

Ra=1E4			
Velocidad		Valores de literatura	diferencia %
Umax	1,92E-01	0,191	0,55%
Vmax	2,32E-01	0,232	0,04%
Nusselt	Simulado		
Medio=	2,248	2,243	0,22%
Maximo=	3,5422	3,533	0,26%
Minimo=	0,5945	0,588	1,11%

Tabla 4. 4: Tabla comparativa de diferencia relativa con la literatura para Ra=10^4

Ra=1E5			
Velocidad		Valores de literatura	diferencia %
Umax	1,32E-01	0,131	0,68%
Vmax	2,57E-01	0,257	0,18%
Nusselt	Simulado		
Medio=	4,5384	4,514	0,54%
Maximo=	7,7628	7,714	0,63%
Minimo=	0,7593	0,747	1,65%

Tabla 4. 5: Tabla comparativa de diferencia relativa con la literatura para Ra=10^5

Ra=1E6			
Velocidad		Valores de literatura	diferencia %
Umax	7,83E-02	0,078	0,35%
Vmax	2,62E-01	0,262	0,06%
Nusselt	Simulado		
Medio=	8,9068	8,783	1,41%
Maximo=	17,915	17,511	2,31%
Minimo=	1,081	1,051	2,85%

Tabla 4. 6: Tabla comparativa de diferencia relativa con la literatura para Ra=10^6

Las diferencias relativas con respecto a los valores de referencia de la literatura fueron indicadas de manera porcentual, la mayor diferencia relativa obtenida fue para Ra= 10^6 y corresponden a un 2,85%, con esto podemos dar por validado el proceso de acople de la ecuación de conservación de la energía mediante la hipótesis de Boussinesq.

4.5 Conclusiones parciales sobre los códigos

Podemos concluir que el código monoprocesador se desempeña correctamente conforme su objetivo inicial, ahora debería bien pues ser puesto a prueba realizando algún estudio científico deseado con cierto grado de fiabilidad.

En el caso del código escrito para el caso PETSc, no es posible hacer estas simulaciones con el algoritmo SIMPLE, si bien puede lograr resolver el problema de Driven Cavity, sería sustituyendo el algoritmo SIMPLE por método de resolución por vorticidad. En general PETSc, demostró ser una excelente herramienta para casos más prácticos y con una estructura de programación bajo la misma filosofía de funcionamiento de esta herramienta. Pero su filosofía de programación sencillamente no permite la implementación del Algoritmo SIMPLE, en esta etapa de su Desarrollo, es probable que en versiones siguientes pueda ser compatible.

5 RESUTADOS DE SIMULACIONES DE CONVECCION NATURAL

Una vez validado el código podemos utilizarlo para estudiar un fenómeno físico y predecir su comportamiento en el caso de este trabajo se presenta el fenómeno de convección natural. En este capítulo se presenta a dicho fenómeno en simulaciones numéricas y cómo interviene en el comportamiento del fluido la variación de la relación de aspecto.

La convección natural es un fenómeno físico muy común causado por una diferencia en el campo de la temperatura, donde por contracción y dilatación térmica, la temperatura incide en el comportamiento de la densidad del fluido. A su vez la diferencia de densidad entre partículas o volúmenes de control en un fluido tiende a generar un desplazamiento vertical, desplazando para arriba a la partícula más ligera (volumen menos denso) y para bajo a la partícula más pesada (volumen más denso), generando así un movimiento donde la velocidad de este aumenta, conforme aumente la diferencia de temperatura.

Este fenómeno es gobernado por las ecuaciones de Navier-Stokes, la conservación de la masa y conservación de la energía, además se asume la hipótesis de Boussinesq, asumiendo un fluido incompresible y analizando el comportamiento bidimensional en régimen laminar permanente.

Se realiza un análisis de malla para tener la mejor relación entre precisión y tiempo de cómputo y también se implementan tres esquemas de interpolación numérica para los volúmenes de control, CENTRAL, UPWIND y LEY DE POTENCIA. Como la diferencia gráfica entre los tres esquemas es poco apreciable para el ojo humano en la cantidad de líneas de corriente graficadas, se presentan solamente los gráficos del esquema central y una comparación numérica con los otros esquemas de interpolación.

5.1 Modelo Físico

El modelo físico es una cavidad rectangular cerrada que contiene al fluido en una sola fase, donde su aspecto es definido por la relación entre su altura H y su ancho L, conocida como relación de aspecto (AR=H/L), como se indica en la **Figura 5.1**.

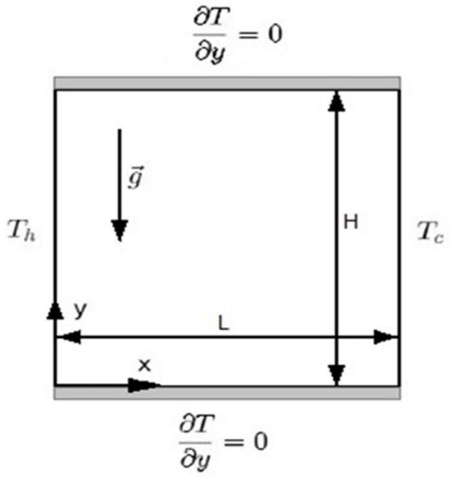

Figura 5. 1 Modelo Físico de cavidad cerrada y sus condiciones de contorno.

En esta cavidad se presentan paredes verticales isotérmicas de temperaturas diferentes que son las que causan el calentamiento diferencial, a la izquierda una pared más caliente con una tempera fija T_h y a la derecha una pared más fría con una temperatura T_c, y dos paredes horizontales adiabáticas que impiden la fuga de la energía térmica.

El modelo matemático idéntico al modelo presentado en el Capítulo anterior en el ejercicio de validación de acople de la ecuación de la energía, por esa razón no se vuelve a presentar.

5.2 Condiciones de contorno

Las condiciones de contorno en las paredes verticales se puede observar una condición de primer tipo también conocida como condición de Dirichlet en la cual se estable como condición a una temperatura constante, para la pared izquierda T= T_h y para la pared derecha T= T_c, y una condición de no deslizamiento, es decir ambas velocidades tanto vertical como horizontal son nulas u=v=0. Esto es para todo volumen en contacto con las paredes.

Las paredes horizontales, en cambio, presentan una condición de contorno mixta, esto es una condición de contorno de segundo tipo o también llamada condición de Von Neumann para el campo de la temperatura, donde el gradiente de temperatura es 0 ($\partial T/\partial y=0$), y una condición de primer tipo para el campo de temperaturas semejante al caso anterior, es decir, las velocidades en contacto con las paredes son nulas en ambas direcciones u=v=0.

5.3 Análisis de Malla

Para garantizar que los resultados tengan un cierto grado de fiabilidad es necesario hacer un análisis de malla. Para ellos tomamos el último caso de validación de la literatura y volvemos a acudir a los resultados de Xamán en 2015, comparando con el caso de estudio de relación de aspecto igual a uno AR=1. Y de esta manera también buscar utilizar la menor cantidad posible de recursos computacionales sin sacrificar resultados.

Nusselt	Medio=	Dif %Lit	Dif %200	Dif %100	Dif %Sgte
Literatura	8,7830				
200 nodos	8,8960	1,29%		1,39%	
180 nodos	8,9062	1,40%	0,11%	1,28%	0,11%
160 nodos	8,9214	1,58%	0,29%	1,11%	0,17%
140 nodos	8,9429	1,82%	0,53%	0,87%	0,24%
120 nodos	8,9730	2,16%	0,87%	0,54%	0,34%
100 nodos	9,0214	2,71%	1,41%		0,54%

Tabla 5. 1 Comparación de número de Nusselt Medio para distintos tamaños de malla

Nusselt	Maximo=	Dif %Lit	Dif %200	Dif %100	Dif %Sgte
Literatura	17,5110				
200 nodos	17,8630	2,01%		3,85%	
180 nodos	17,9100	2,28%	0,26%	3,60%	0,26%
160 nodos	18,0120	2,86%	0,83%	3,05%	0,57%
140 nodos	18,1200	3,48%	1,44%	2,47%	0,60%
120 nodos	18,2880	4,44%	2,38%	1,57%	0,93%
100 nodos	18,5790	6,10%	4,01%		1,59%

Tabla 5. 2 Comparación de número de Nusselt Máximo para distintos tamaños de malla

Nusselt	Minimo=	Dif %Lit	Dif %200	Dif %100	Dif %Sgte
Literatura	1,0510				
200 nós	1,0714	1,94%		7,47%	
180 nós	1,0811	2,86%	0,91%	6,63%	0,91%
160 nós	1,0937	4,06%	2,08%	5,54%	1,17%
140 nós	1,1101	5,62%	3,61%	4,13%	1,50%
120 nós	1,1296	7,48%	5,43%	2,44%	1,76%
100 nós	1,1579	10,17%	8,07%		2,51%

Tabla 5. 3 Comparación de número de Nusselt Mínimo para distintos tamaños de malla

Nusselt	Tiempo aprox.
Literatura	
200nos	24hs
180nos	6-8hs
160nos	5-8hs
140nos	4hs
120nos	2hs
100nos	0.6-1hs

Tabla 5. 4 Comparación de tiempo de cómputo para distintos tamaños de malla

Mediante las tablas anteriores se ha seleccionado a la malla de 180x180 nodos, puesto que ofrece una buena fiabilidad con los resultados de la literatura; basados en que su diferencia con la literatura no excede al 3%, la diferencia con una mañana mayor es menor al 1%, es decir aumentando el tamaño de malla, no se mejoró sustancialmente la precisión, pero

en caso contrario el tiempo de cómputo aumenta más de 300%, por lo cual podemos concluir que 180 es el tamaño que deseamos.

5.4 Gráficas de los resultados

Se presentan los gráficos de línea de corriente (campo de la velocidad) y líneas isotérmicas (campo de temperatura) para distintas cinco relaciones de aspecto (0,25; 0,5; 1; 2; 4) diferentes con número de Rayleigh de 10^3 hasta 10^6.

Para las siguientes cuatro gráficas desde **Figura 5.2** hasta **Figura 5.5**, se presentan los resultados para la relación de aspecto AR=0,25 en un mallado estructurado uniforme de 45x180 de manera que cada volumen de control sea cuadrado de acuerdo con los cálculos de la validación de malla que se estudió en el apartado anterior.

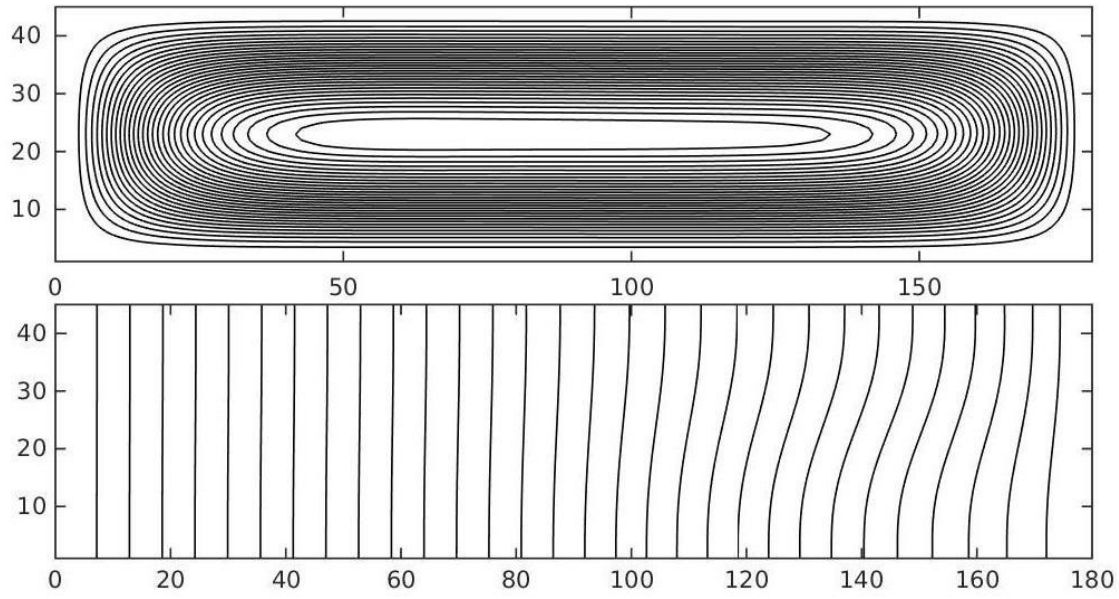

Figura 5. 2 Línea de corriente y líneas isotérmicas para relación de aspecto AR=0.25 y Rayleigh RA=10^3

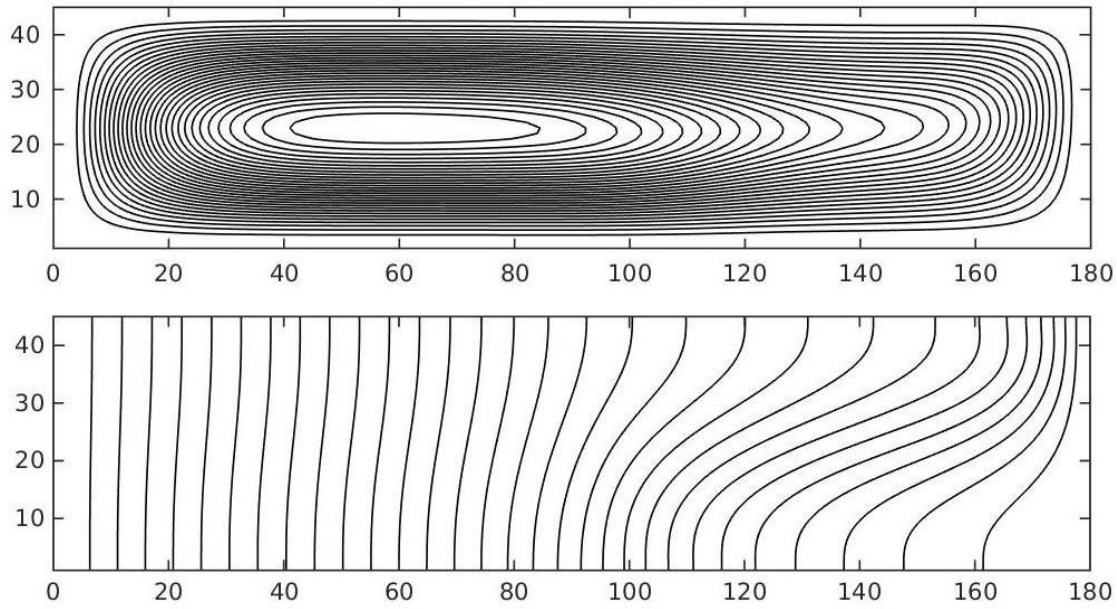

Figura 5. 3 Línea de corriente y líneas isotérmicas para relación de aspecto AR=0.25 y Rayleigh RA=10^4

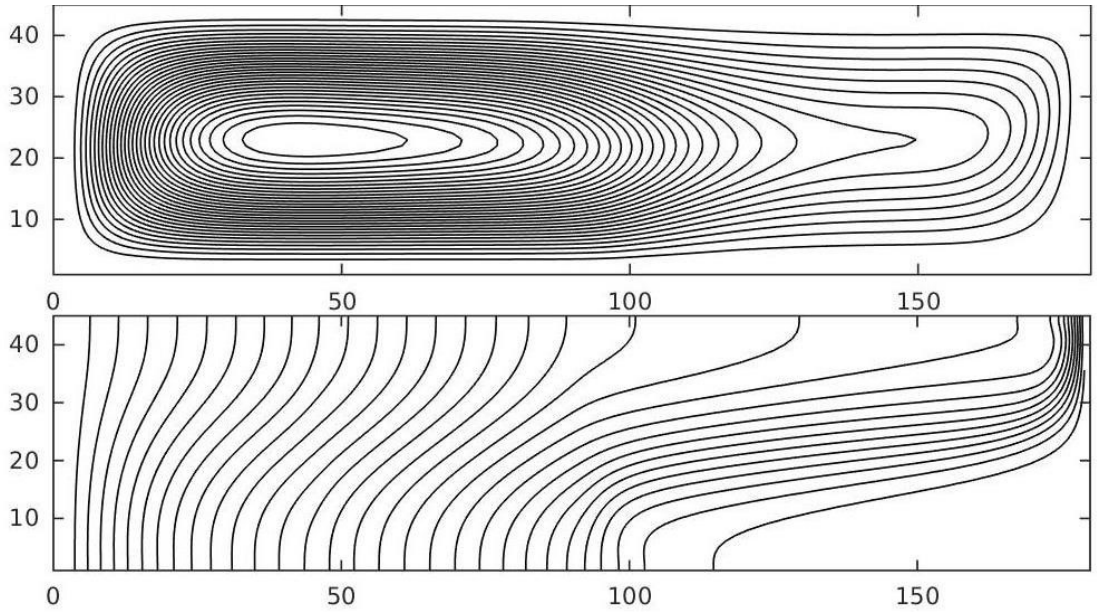

Figura 5. 4 Línea de corriente y líneas isotérmicas para relación de aspecto AR=0.25 y Rayleigh RA=10^5

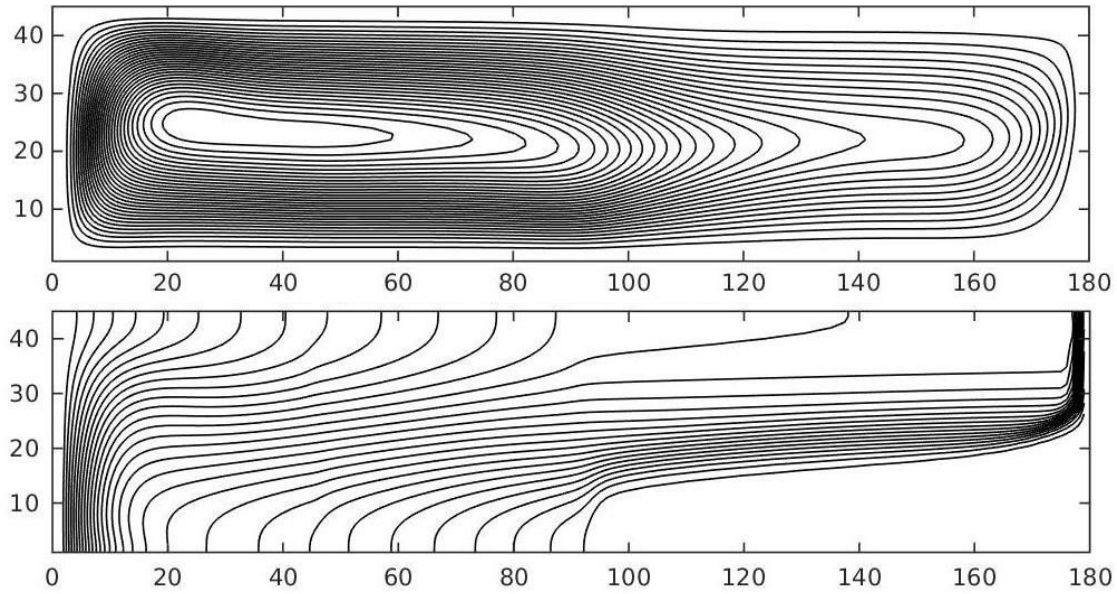

Figura 5. 5 Línea de corriente y líneas isotérmicas para relación de aspecto AR=0.25 y Rayleigh RA=10^6

Al observar la primera gráfica **Figura 5.2** para RA=10^3, se puede evidenciar un comportamiento bastante esperado a simple imaginación y comprensión del fluido, hay un flujo del fluido de manera casi circular con el vórtice centrado prácticamente en el centro de la cavidad y las líneas isotérmicas son casi verticales lo cual indica que el gradiente de temperatura es horizontal, es decir, la energía térmica se transfiere casi directamente de una pared a la otra a través del fluido, asemejándose mucho al comportamiento netamente difusivo.

En las gráficas **Figura 5.3** y **Figura 5.4**, se evidencia el principio convectivo cuando las líneas de temperatura comienzan a alabearse, a converger en ciertos puntos concentrados como en la esquina superior derecha, donde podemos interpretar como un punto de brusca variación de temperatura, mientras que el vórtice del flujo comienza a desplazarse levemente a la izquierda del lado de la pared más cálida.

En la **Figura 5.5**, lo llamativo es que ya logramos ver algunas líneas isotérmicas bastante horizontales, esto indica que el gradiente de temperatura dependiendo de la posición puede ser vertical u horizontal, es decir, la energía térmica se transmite en ambos sentidos, en el centro de la cavidad de manera vertical, mientras que cerca de las paredes de manera horizontal.

En las siguientes cuatro gráficas desde **Figura 5.6** hasta **Figura 5.9**, veremos los resultados de las simulaciones para la relación de aspecto AR=0,5 para los números de Rayleigh de 10^3 a 10^6. Se utiliza un mallado estructurado uniforme de 90x180 para que los volúmenes de control sean cuadrados, mismo caso del anterior.

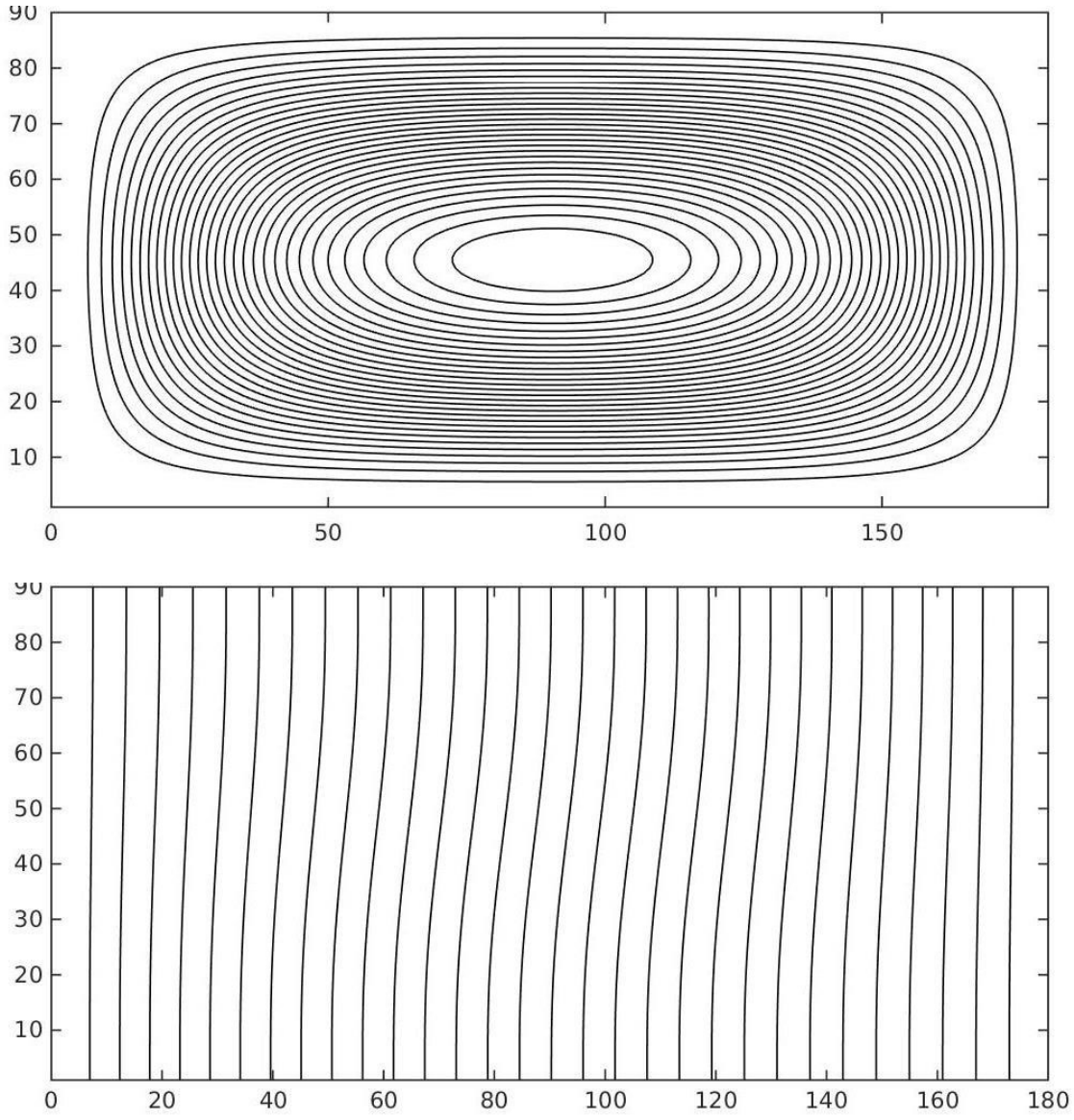

Figura 5. 6 Línea de corriente y líneas isotérmicas para relación de aspecto AR=0.5 y Rayleigh RA=10^3

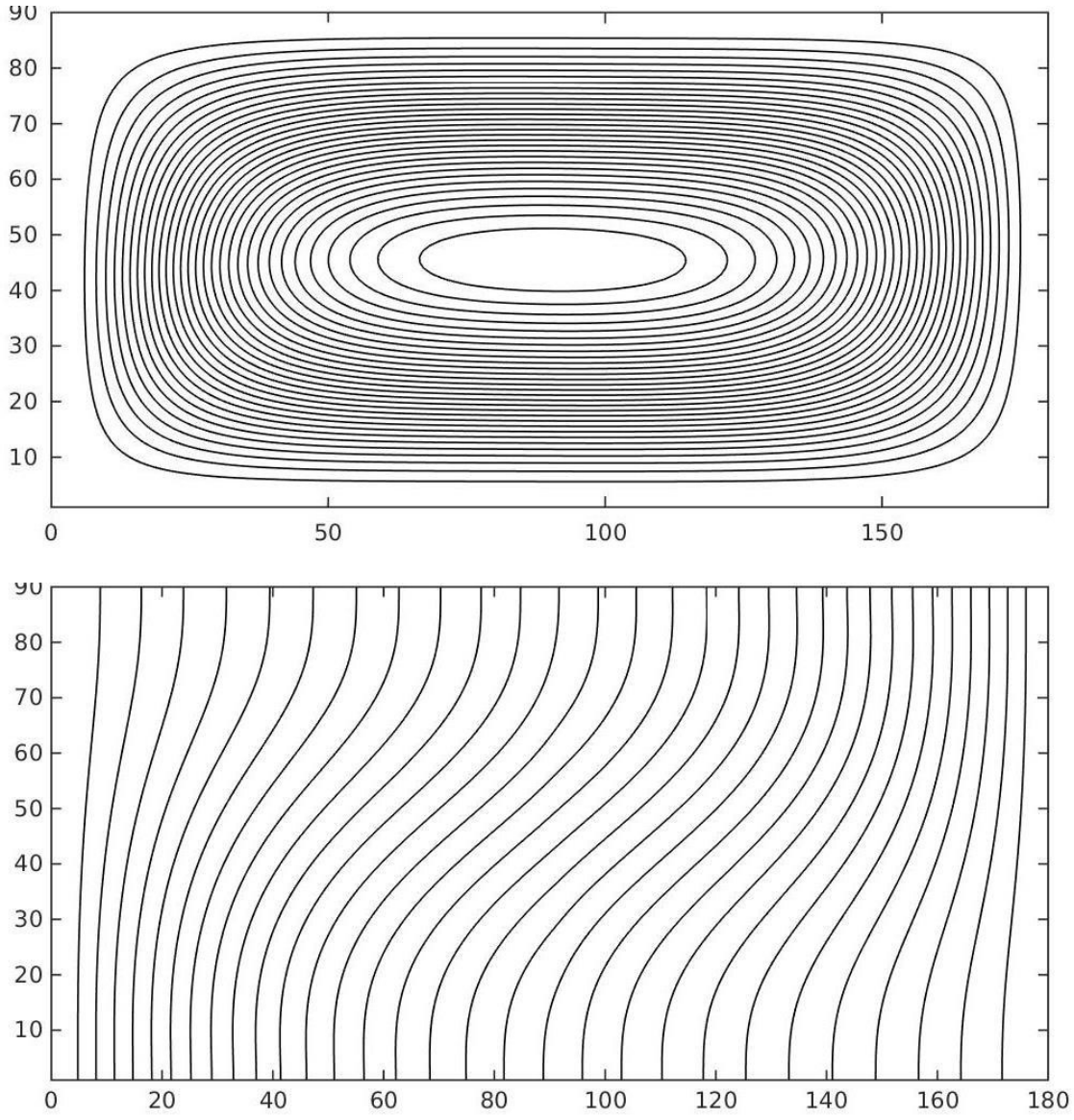

Figura 5. 7 Línea de corriente y líneas isotérmicas para relación de aspecto AR=0.5 y Rayleigh RA=10^4

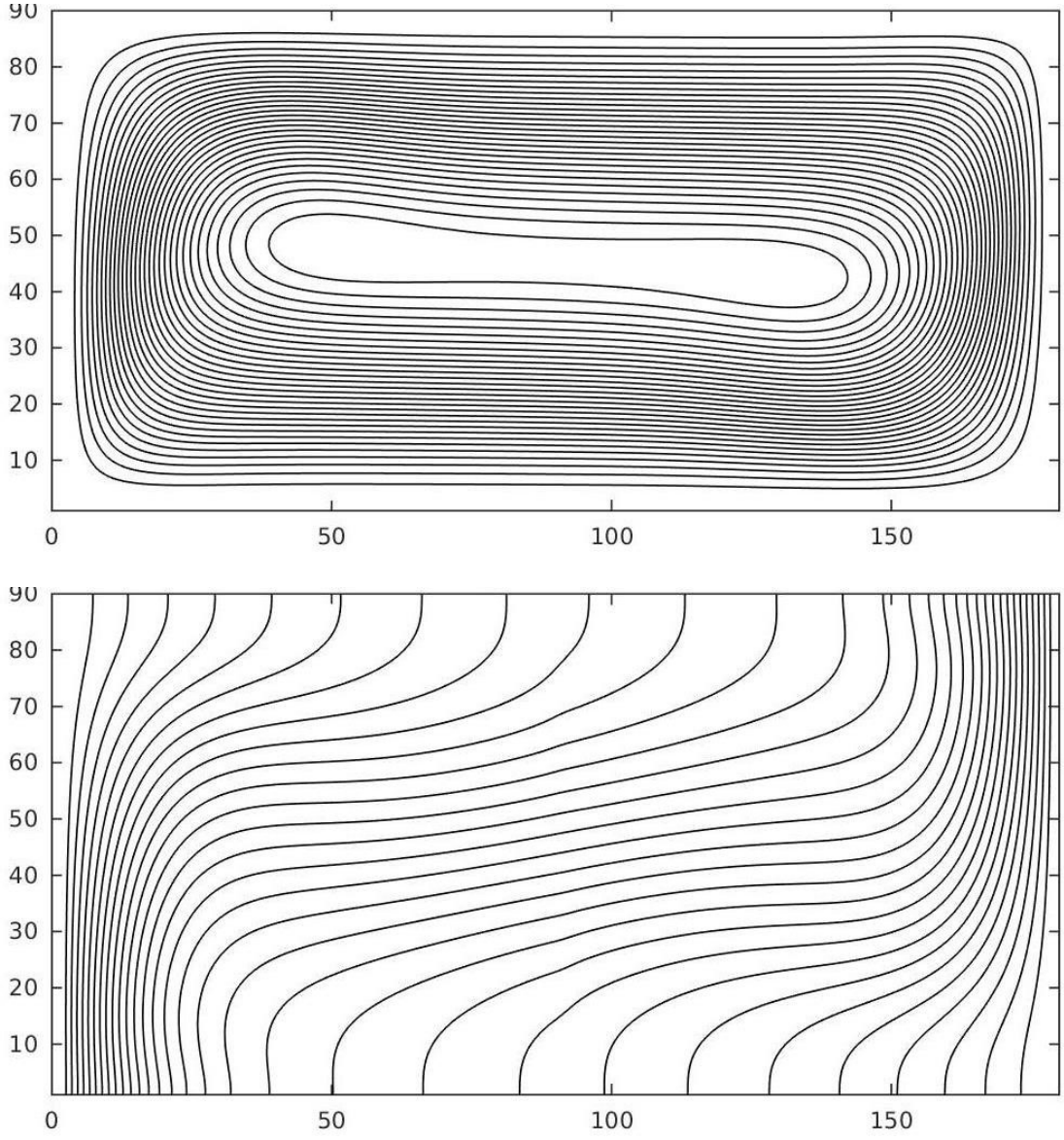

Figura 5. 8 Línea de corriente y líneas isotérmicas para relación de aspecto AR=0.5 y Rayleigh RA=10^5

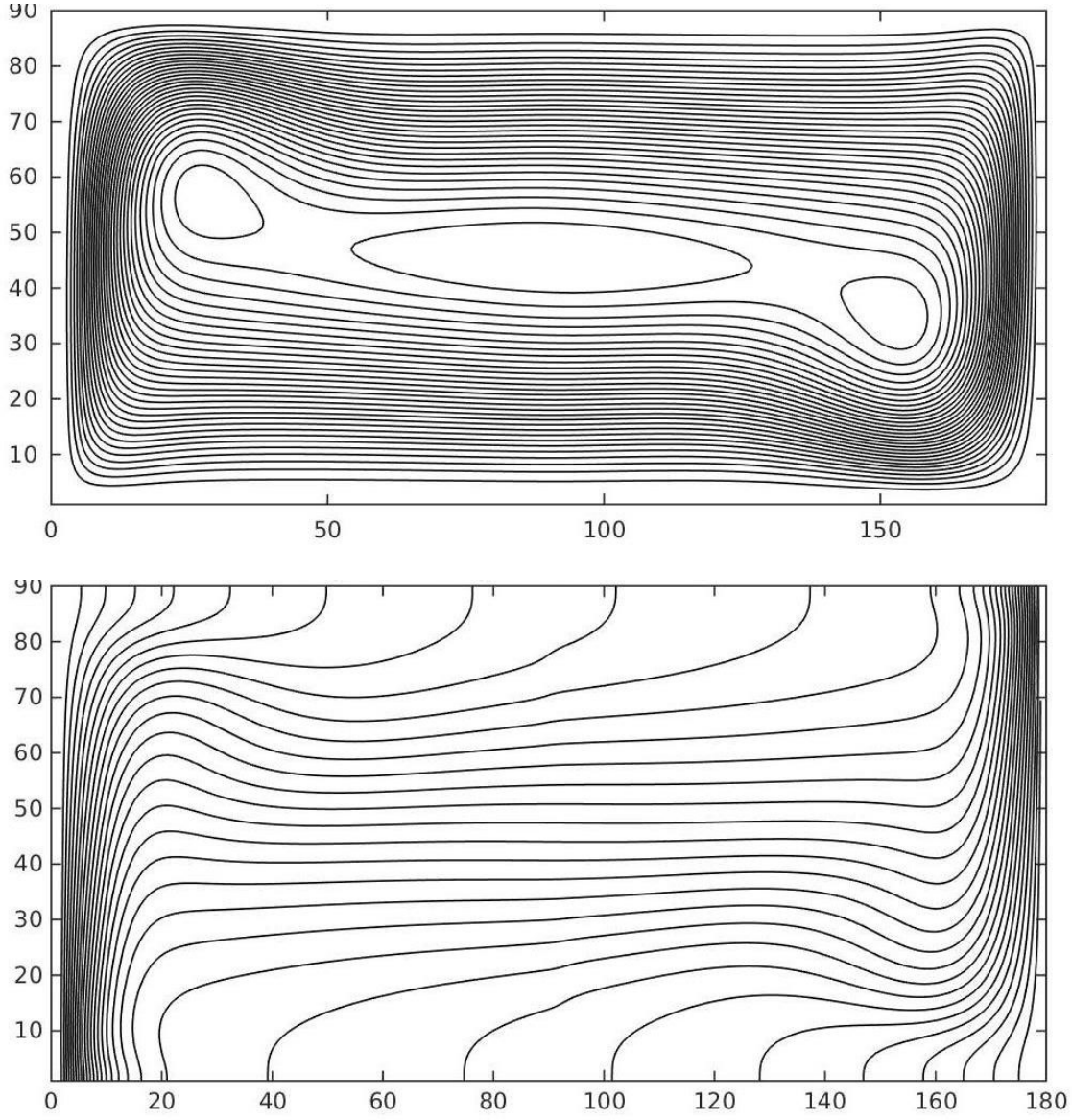

Figura 5. 9 Línea de corriente y líneas isotérmicas para relación de aspecto AR=0.5 y Rayleigh RA=10^6

En la **Figura 5.6**, se puede notar de nuevo el comportamiento mencionado para la **Figura 5.2**. Las líneas de corriente indican un flujo tendiendo a la forma circular con un vórtice centrado en la cavidad. Mientras que las líneas isotérmicas indican un gradiente en dirección horizontal casi como un sólido en difusión pura.

En la **Figura 5.7**, las líneas isotérmicas comienzan a mostrar una inclinación aproximándose a los 45°, esto indica que ahora la transferencia energía termina tiene una componente en su vertical. Y las líneas de corriente, tienden a formar geometrías más ovaladas acercándose más a la geometría de la cavidad.

Mientras que al subir el número de Rayleigh en la **Figura 5.8** y **Figura 5.9**, se denota como las líneas isotérmicas van tomando progresivamente una posición horizontal en el centro de la cavidad indicando un gradiente de temperatura vertical y unas líneas de corriente que tienden a adoptar la forma de la cavidad.

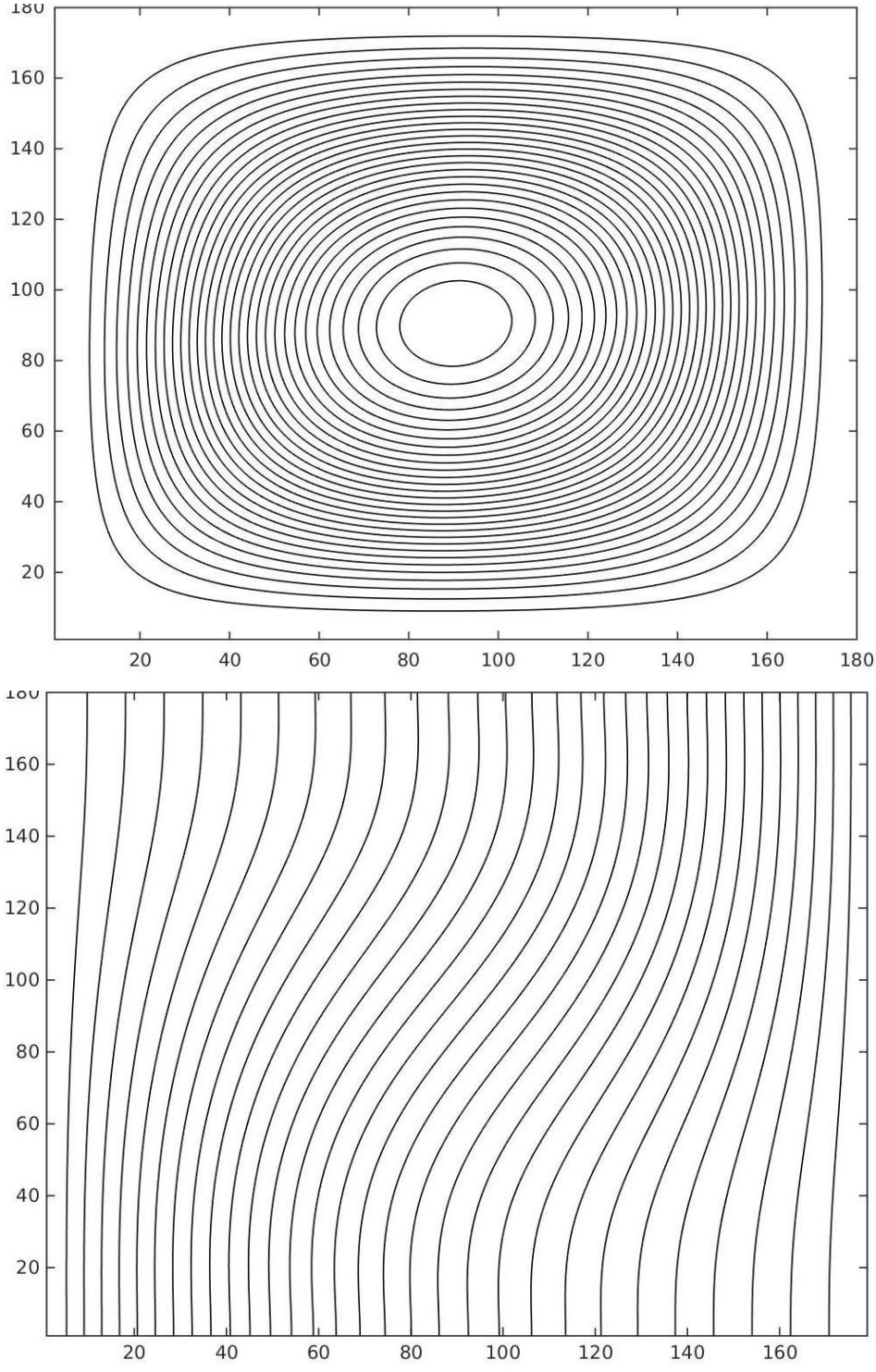

Figura 5. 10 Línea de corriente y líneas isotérmicas para relación de aspecto AR=1 y Rayleigh RA=10^3

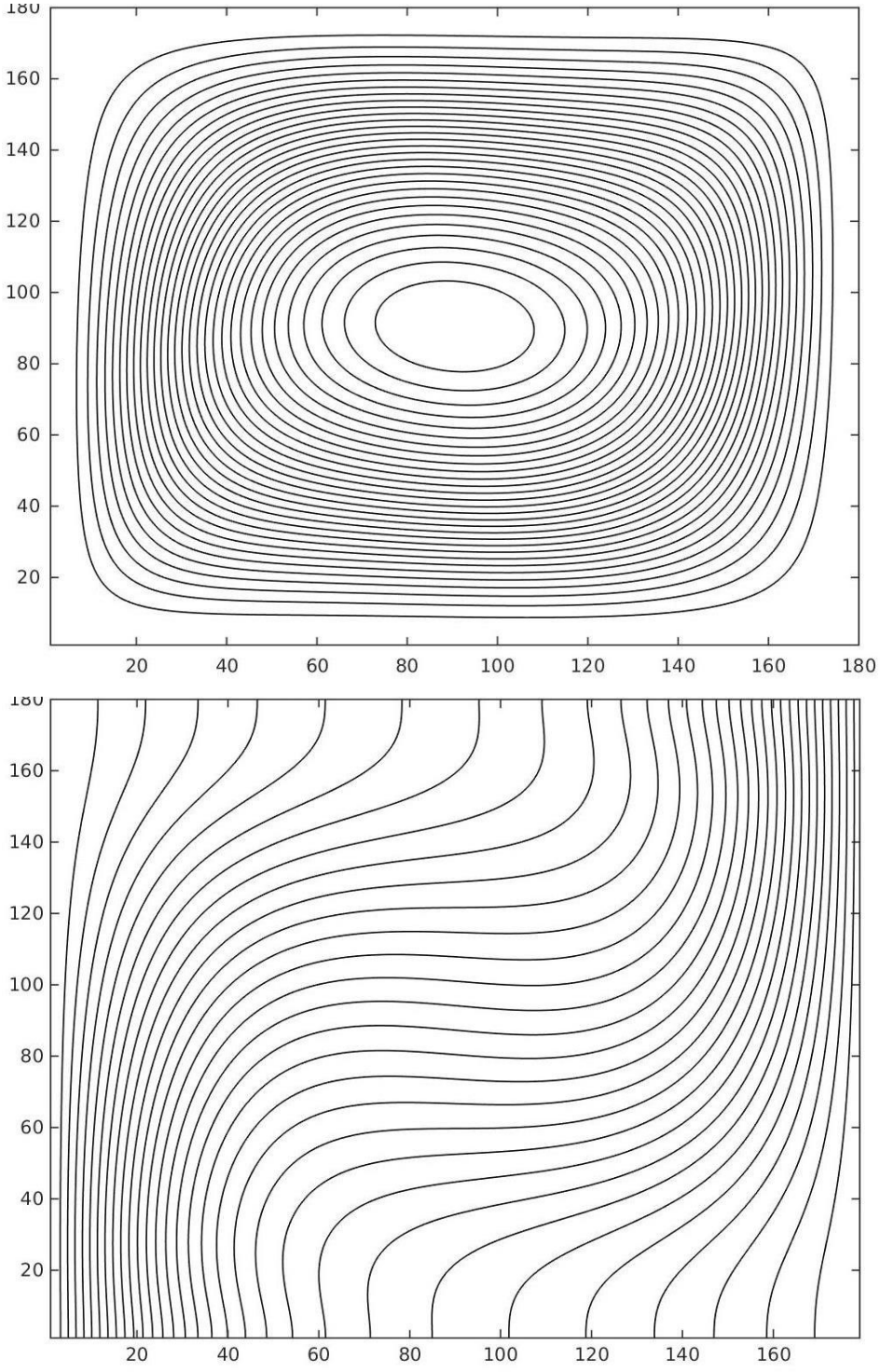

Figura 5. 11 Línea de corriente y líneas isotérmicas para relación de aspecto AR=1 y Rayleigh RA=10^4

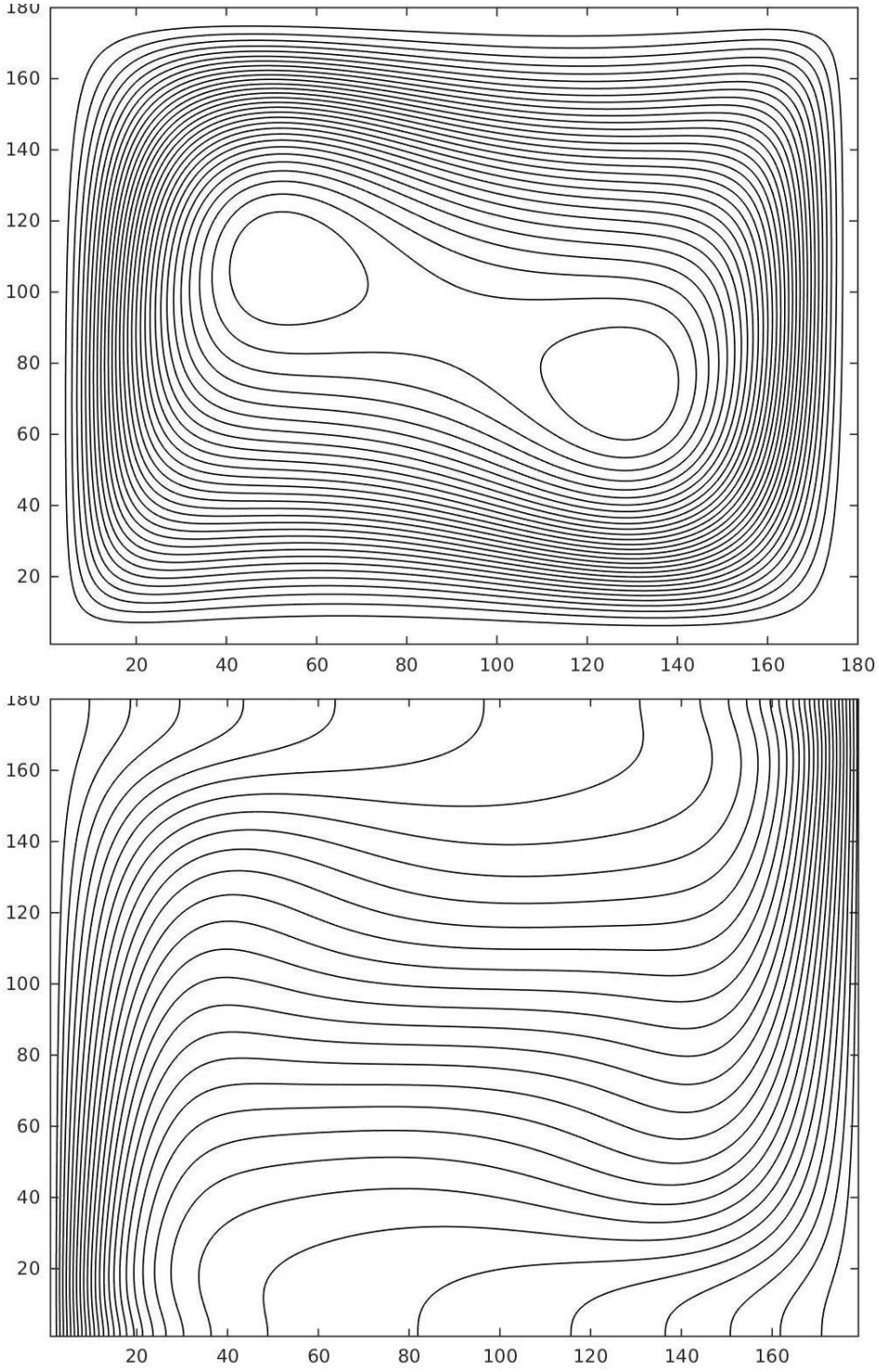

Figura 5. 12 Línea de corriente y líneas isotérmicas para relación de aspecto AR=1 y Rayleigh RA=10^5

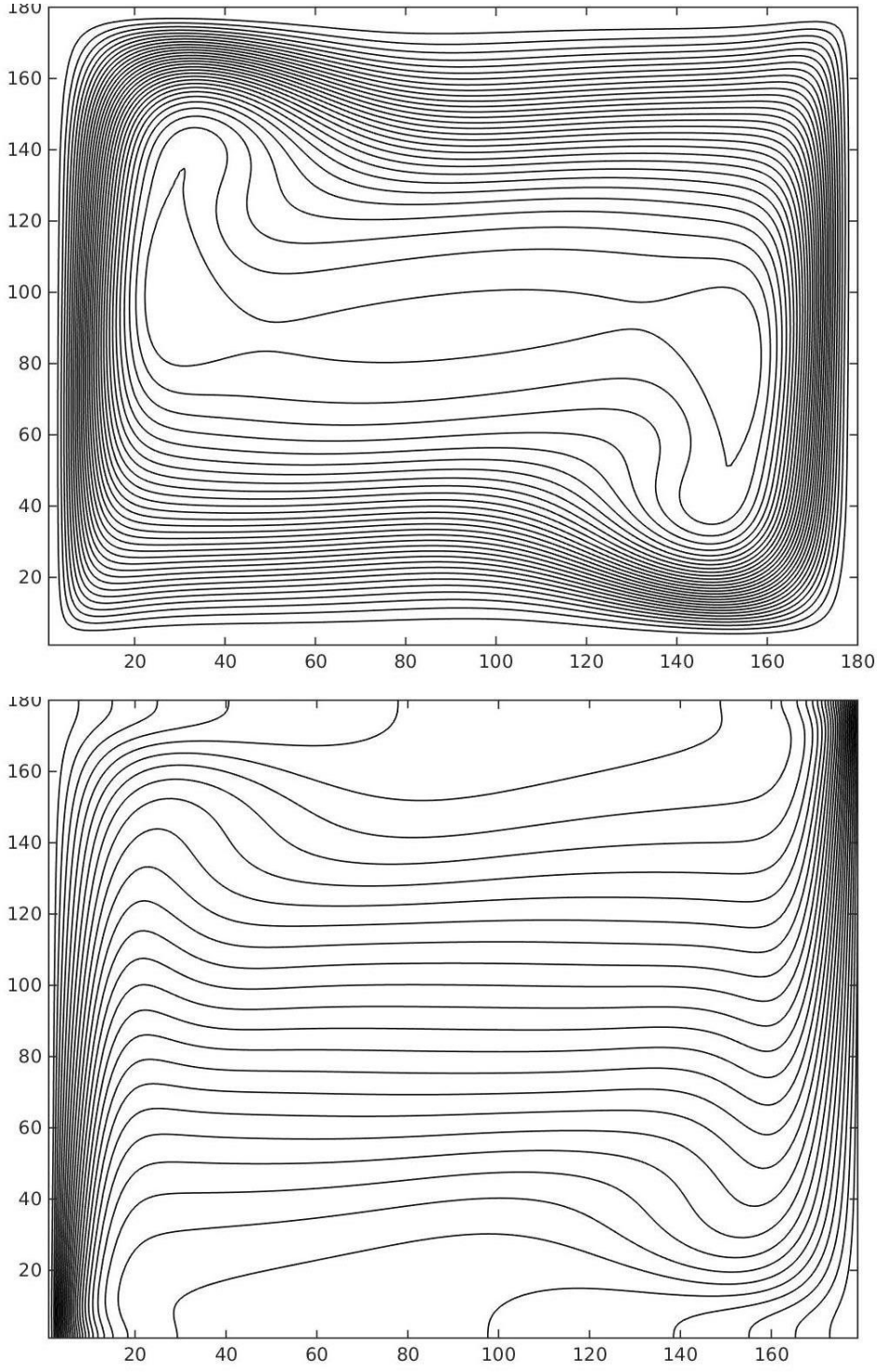

Figura 5. 13 Línea de corriente y líneas isotérmicas para relación de aspecto AR=1 y Rayleigh RA=10^6

En las figuras de relación de aspecto AR=1 podemos tener la referencia del comportamiento estándar de la convección natural, para entender en qué afecta la variación de esta razón al comportamiento del fluido; como ya hemos visto las gráficas anteriores, podemos denotar algunas pre-conclusiones para el comportamiento de la convección natural para la variación de la relación de aspecto para números inferiores a uno.

De la siguiente manera, **PARA LA ENERGÍA,** a medida que la relación de aspecto va disminuyendo se observa que en las líneas isotérmicas para bajos números de Rayleigh ($RA=10^3$ y $RA=10^4$) tienen a un comportamiento tendiendo a ser puramente difusivo, mientras que a medida que aumenta el número de Rayleigh, el efecto de la relación de aspecto va perdiendo relevancia.

En cambio, **PARA EL FLUJO**, a medida que la relación de aspecto va disminuyendo, el efecto de esta crece conforme crece el número de Rayleigh, desplazando el vórtice del flujo en dirección a la pared más caliente, perdiendo relevancia este efecto con la diminución del número de Rayleigh.

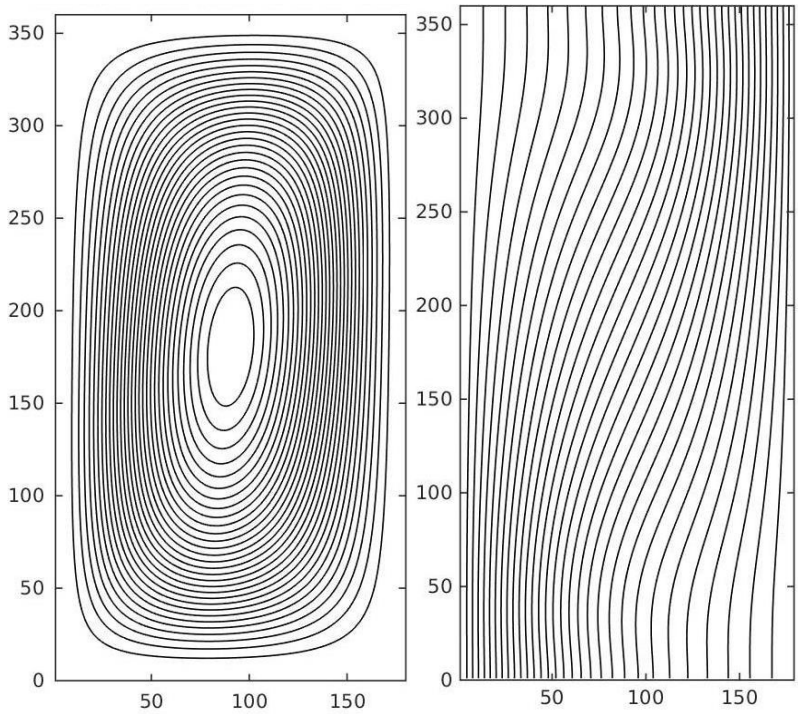

Figura 5. 14 Línea de corriente y líneas isotérmicas para relación de aspecto AR=2 y Rayleigh $RA=10^3$

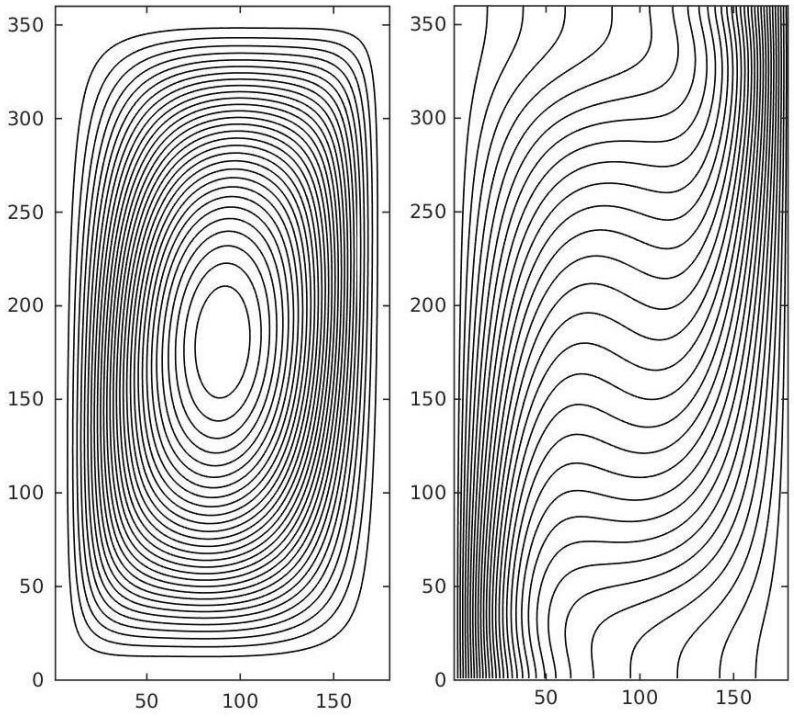

Figura 5. 15 Línea de corriente y líneas isotérmicas para relación de aspecto AR=2 y Rayleigh RA=10^4

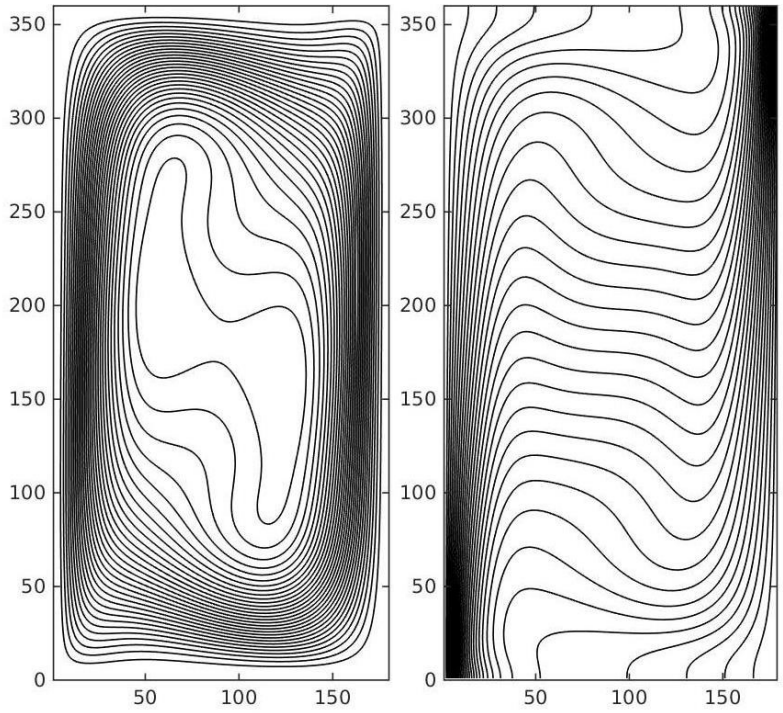

Figura 5. 16 Línea de corriente y líneas isotérmicas para relación de aspecto AR=2 y Rayleigh RA=10^5

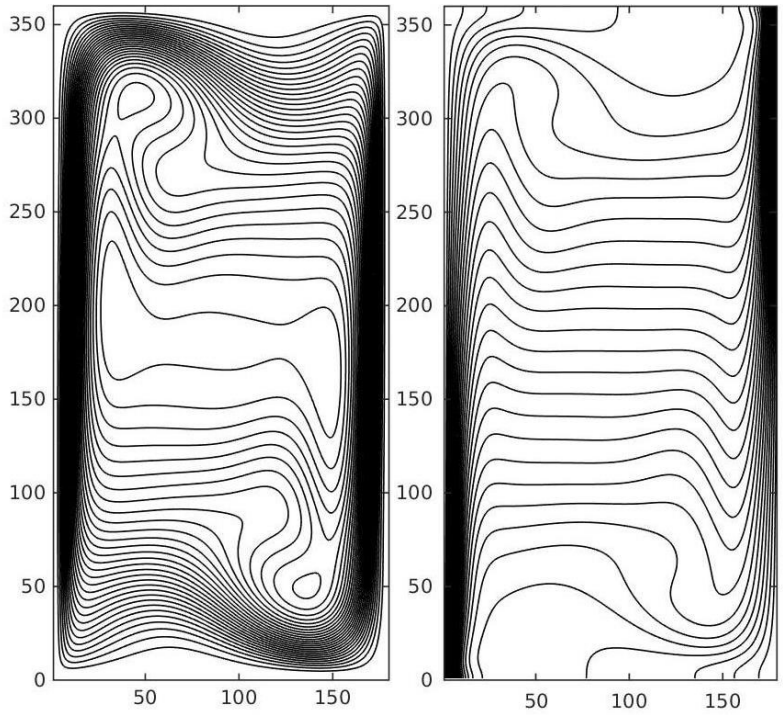

Figura 5. 17 Línea de corriente y líneas isotérmicas para relación de aspecto AR=2 y Rayleigh RA=10^6

En estas cuatro gráficas podemos ver que a diferencia de las gráficas con relación de aspecto menor a uno, aquí las líneas isotérmicas van tomando un carácter más horizontal desde los bajos números de Rayleigh, lo cual indica que hay una componente vertical del gradiente de temperatura en el centro de la cavidad. Teniendo las zonas de gradientes máximos en las esquinas, inferior izquierda y superior derecha.

Mientras que, para líneas de corriente, no se observa un comportamiento diferente a los mencionados anteriormente para las otras gráficas. Los vórtices son centrados y las líneas de flujo tienden a tomar la forma de la cavidad.

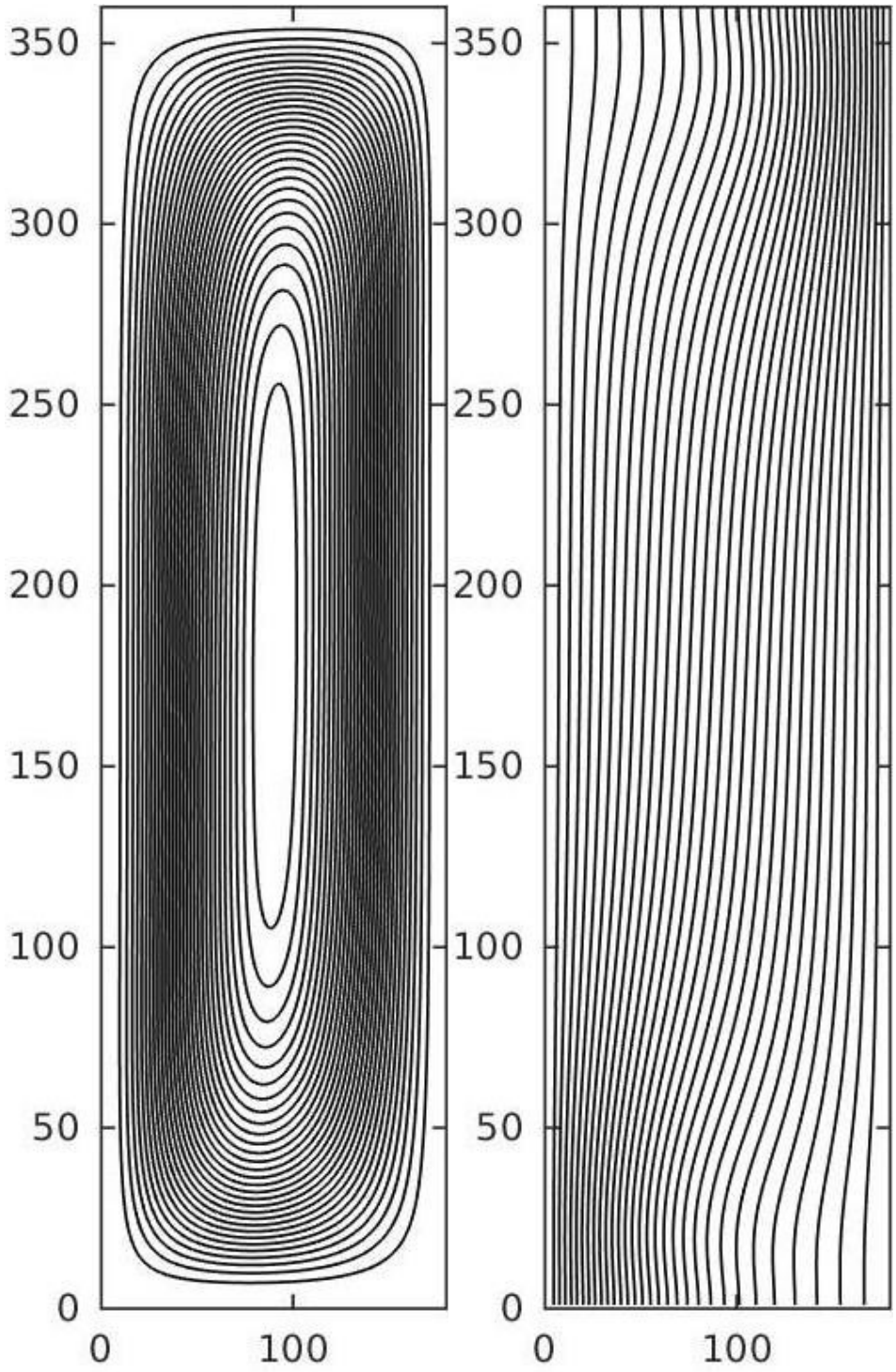

Figura 5. 18 Línea de corriente y líneas isotérmicas para relación de aspecto AR=4 y Rayleigh RA=10^3

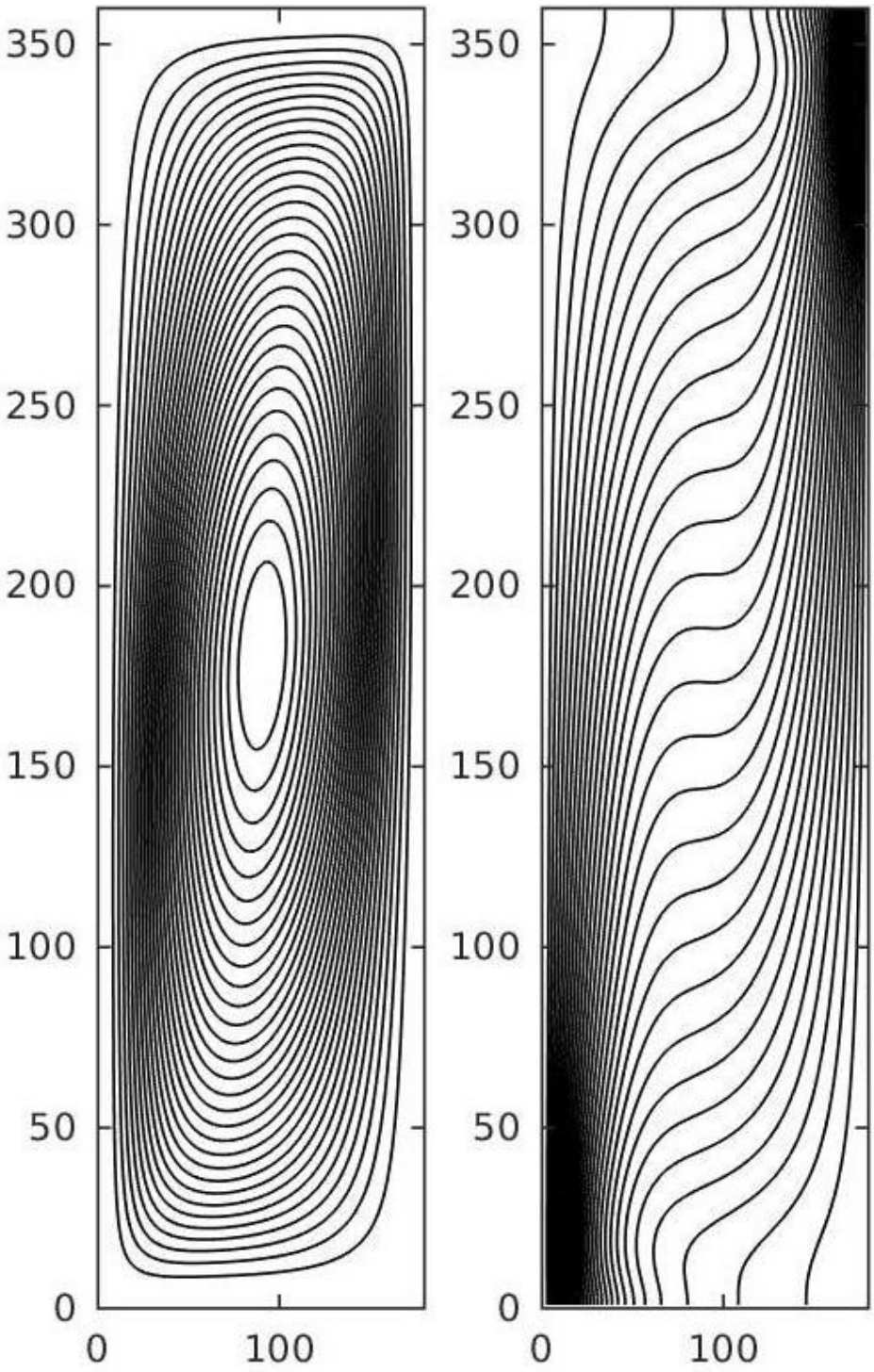

Figura 5. 19 Línea de corriente y líneas isotérmicas para relación de aspecto AR=4 y Rayleigh RA=10^4

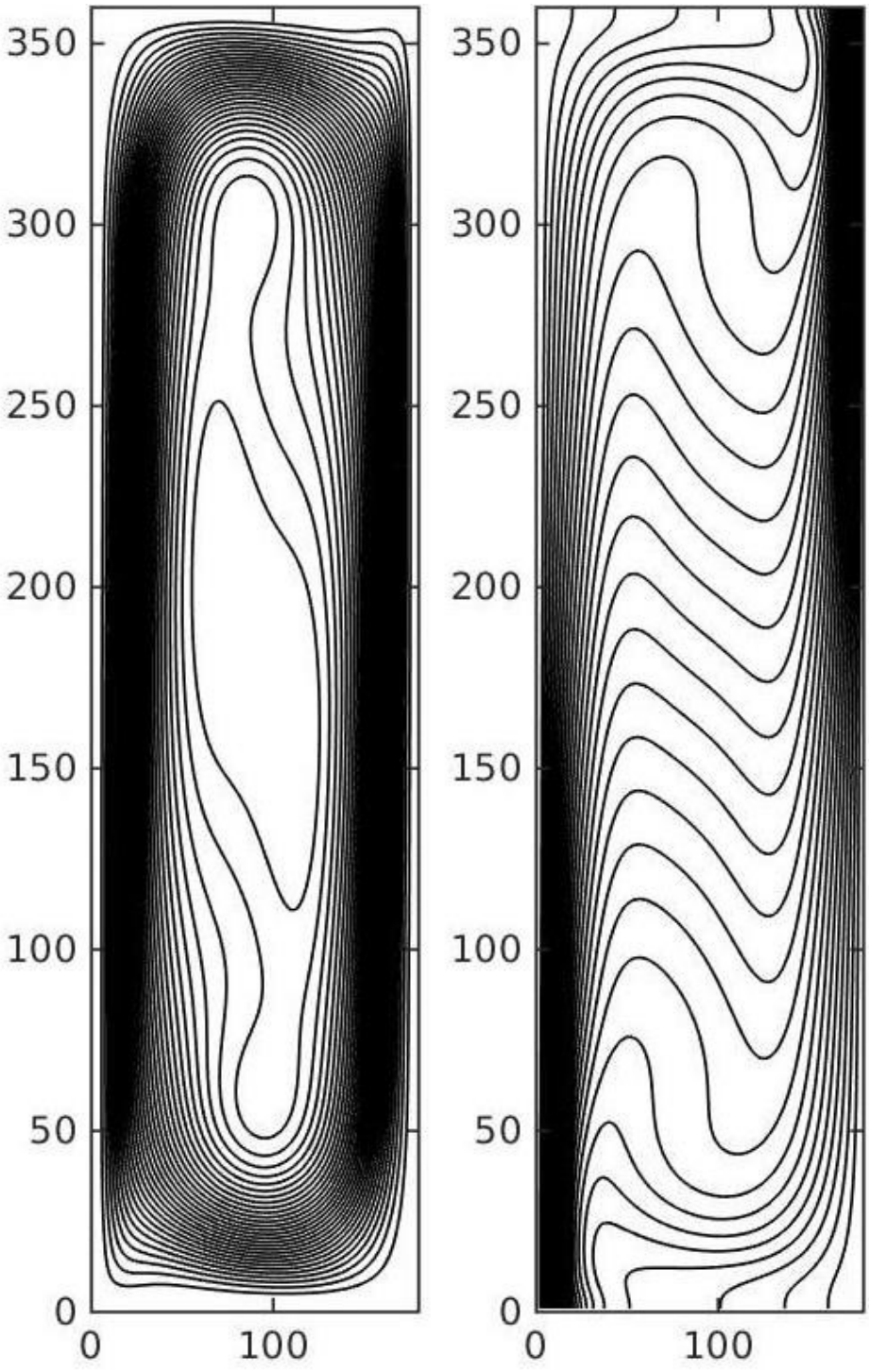

Figura 5. 20 Línea de corriente y líneas isotérmicas para relación de aspecto AR=4 y Rayleigh RA=10^5

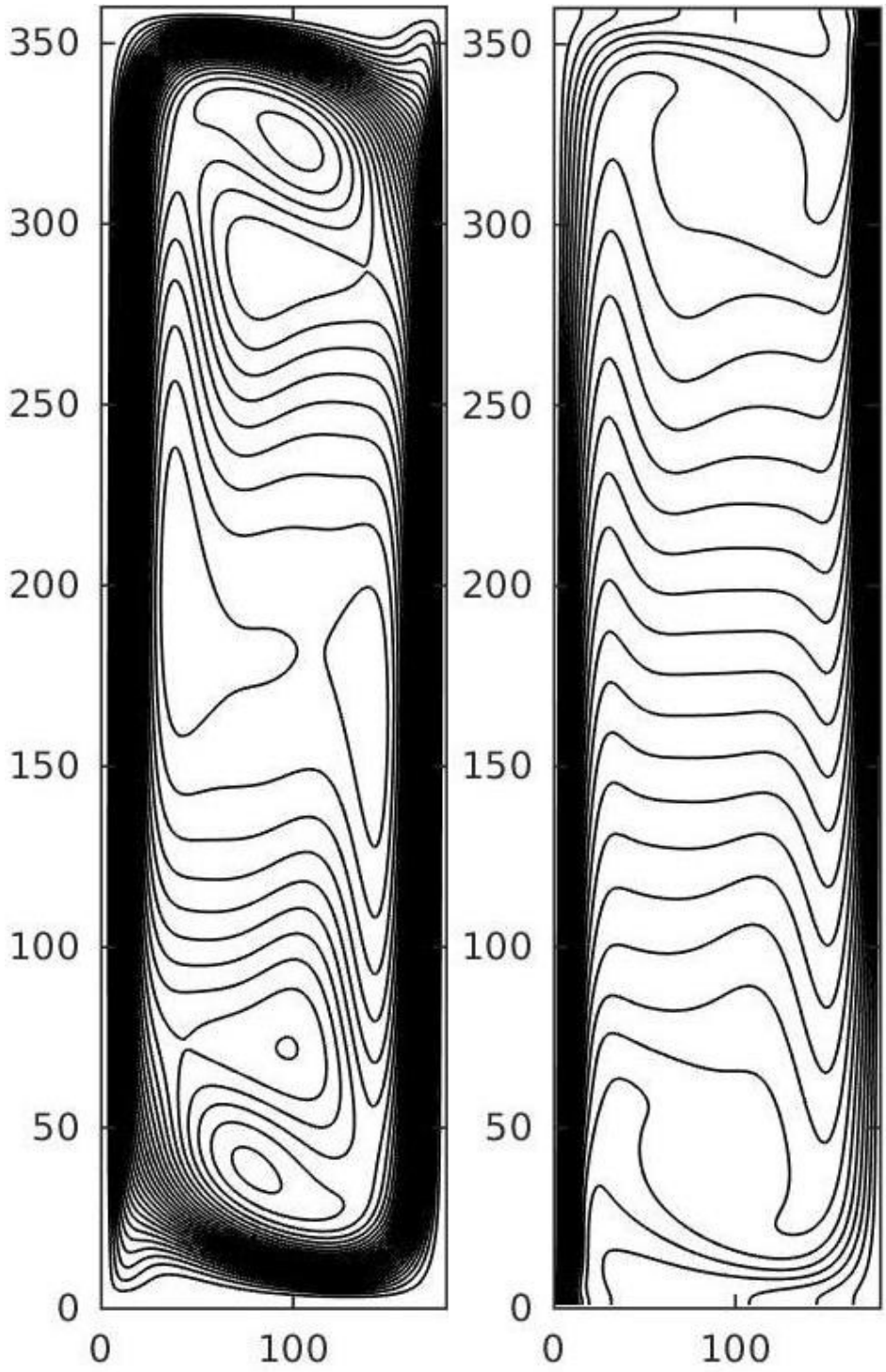

Figura 5. 21 Línea de corriente y líneas isotérmicas para relación de aspecto AR=4 y Rayleigh RA=10^6

En este punto para las gráficas de relación de aspecto AR=4, podemos denotar que las líneas isotérmicas aumentan el gradiente de temperatura cerca de las paredes y logran una zona que casi cubre lo largo de la pared, mientras que en el centro de la cavidad el gradiente es casi vertical.

En cuanto al flujo, algo llamativo es como el vórtice, va tomando mayor volumen conforme aumenta el número de Rayleigh, la interpretación de esto es que la principal variación de velocidad conforme posición se da en una línea perpendicular a las paredes en las proximidades de estas.

5.5 Comparación numérica entra distintos esquemas.

Para realizar un análisis de los resultados obtenidos utilizando diferentes esquemas numéricos. Se utiliza una comparación local, para ellos se realizan las mismas simulaciones en semejanza geométrica y de mallas, y se compara punto a punto la diferencia numérica entre ellas. Calculando la diferencia entre cada punto correspondido de las matrices de velocidad horizontal y vertical, la matriz de presión y la matriz de temperatura, para los esquemas UPWIND y LEY DE POTENCIA respecto al esquema CENTRAL

En las siguientes tablas se presentan las diferencias máximas calculada para cada matriz, las diferencias no son porcentuales ya que en algunos puntos donde el valor tiende a cero, puede presentar una indefinición de la relación.

Ra	Esquema	Matriz U	Matriz V	Matriz P	Matriz T
10^3	UPWIND	0.0017	0.0175	5.9090e-07	2.9774e-04
	LEY DE POTENCIA	1.4235e-05	0.0035	2.1834e-09	1.0999e-06
10^4	UPWIND	0.0094	0.0884	6.8476e-06	0.0016
	LEY DE POTENCIA	3.5484e-04	0.0021	0.3417	6.1307e-05
10^5	UPWIND	0.0172	0.1038	1.5776e-05	0.0017
	LEY DE POTENCIA	0.0017	0.0044	2.1980e-06	2.3343e-04
10^6	UPWIND	0.0295	0.5322	2.2019e-05	0.0013
	LEY DE POTENCIA	0.0036	0.0536	8.6288e-06	3.9486e-04

Tabla 5. 5 Diferencias máximas locales para los esquemas UPWIND y LEY DE POTENCIA, respecto al esquema CENTRAL para relación de aspecto AR=0.25

Ra	Esquema	Matriz U	Matriz V	Matriz P	Matriz T
10^3	UPWIND	7.0397e-04	0.0011	2.2498e-07	4.2401e-05
	LEY DE POTENCIA	3.5087e-05	1.1053e-04	2.8354e-09	1.3065e-09
10^4	UPWIND	0.0054	0.0225	2.2325e-06	2.9005e-04
	LEY DE POTENCIA	8.4581e-06	1.2174e-04	4.6072e-09	2.7089e-07
10^5	UPWIND	0.0218	0.1387	1.3491e-05	7.3408e-04
	LEY DE POTENCIA	7.7187e-04	0.0036	3.1276e-07	1.8930e-05
10^6	UPWIND	0.0548	0.5324	3.7815e-05	0.0013
	LEY DE POTENCIA	0.0038	0.0381	2.5058e-06	8.9918e-05

Tabla 5. 6 Diferencias máximas locales para los esquemas UPWIND y LEY DE POTENCIA, respecto al esquema CENTRAL para relación de aspecto AR=0.5

Ra	Esquema	Matriz U	Matriz V	Matriz P	Matriz T
10^3	UPWIND	0.0116	0.0074	8.5818e-07	2.3417e-04
	LEY DE POTENCIA	0.0702	0.0443	5.6818e-06	4.2709e-08
10^4	UPWIND	0.0119	0.0378	3.4329e-06	6.1877e-04
	LEY DE POTENCIA	0.0231	0.0489	6.7180e-06	8.1240e-06
10^5	UPWIND	0.0342	0.2286	9.9243e-06	9.8501e-04
	LEY DE POTENCIA	0.0354	0.1025	7.6284e-06	3.0025e-05
10^6	UPWIND	0.2067	0.4773	2.6009e-05	0.0013
	LEY DE POTENCIA	0.0471	0.1171	4.2449e-06	9.7211e-05

Tabla 5. 7 Diferencias máximas locales para los esquemas UPWIND y LEY DE POTENCIA, respecto al esquema CENTRAL para relación de aspecto AR=1

Ra	Esquema	Matriz U	Matriz V	Matriz P	Matriz T
10^3	UPWIND	0.0148	0.0106	1.7552e-04	2.9172e-04
	LEY DE POTENCIA	0.0193	0.0079	8.7825e-05	7.5381e-06
10^4	UPWIND	0.0145	0.0447	5.1399e-04	5.1501e-04
	LEY DE POTENCIA	0.0088	0.0148	5.0531e-04	3.6302e-05
10^5	UPWIND	0.1133	0.1250	6.3065e-04	7.8755e-04
	LEY DE POTENCIA	0.0115	0.0091	4.4773e-04	1.0701e-04
10^6	UPWIND	0.2384	1.1997	8.7489e-04	0.0017
	LEY DE POTENCIA	0.0281	0.0747	4.8616e-04	1.6282e-04

Tabla 5. 8 Diferencias máximas locales para los esquemas UPWIND y LEY DE POTENCIA, respecto al esquema CENTRAL para relación de aspecto AR=2

Ra	Esquema	Matriz U	Matriz V	Matriz P	Matriz T
10^3	UPWIND	0.0763	0.0051	5.6285e-05	2.2702e-04
	LEY DE POTENCIA	0.0439	6.8954e-04	2.5213e-05	4.5409e-06
10^4	UPWIND	0.0355	0.0269	0.0018	5.9973e-04
	LEY DE POTENCIA	0.0103	0.0016	0.0012	5.5316e-05
10^5	UPWIND	0.4164	0.1113	0.0013	6.9027e-04
	LEY DE POTENCIA	0.0573	0.0105	6.5654e-04	1.2926e-04
10^6	UPWIND	0.1982	0.6391	7.1956e-05	0.0011
	LEY DE POTENCIA	0.0290	0.0654	5.8324e-04	1.9456e-04

Tabla 5. 9 Diferencias máximas locales para los esquemas UPWIND y LEY DE POTENCIA, respecto al esquema CENTRAL para relación de aspecto AR=4

La primera observación para comparar los tres esquemas es que la diferencia entre las matrices de presión y temperatura es ínfima y que la mayor diferencia es apenas en el orden de los 0.2% para altos números de Rayleigh. En las matrices de velocidad se puede evidenciar que estas diferencias locales van en aumento conforme aumenta el número de Rayleigh para todas las relaciones de aspecto, siendo el caso más notorio para mayores relaciones de aspecto y mayores números de Rayleigh en la velocidad vertical V.

Los resultados del esquema de LEY DE POTENCIA son más semejantes al esquema CENTRAL, que los resultados del esquema UPWIND. Esto también se podrá observar en las gráficas para estos esquemas agregadas al anexo.

6 CONCLUSIÓN Y TRABAJOS FUTUROS

6.1 Conclusión

Fue escrito un código computacional bidimensional que permite resolver las ecuaciones de Navier-Stokes, conservación de la masa y energía, en régimen laminar y transitorio, para fluidos incompresibles. O código fue debidamente validado con cuatro soluciones de referencia.

El código está escrito en lenguaje C++ y puede ser utilizado abiertamente por cualquier aluno de la universidad.

Utilizando el código mono procesador fue realizado el estudio numérico del efecto de la variación de la relación de aspecto en la convección natural, produciendo un artículo científico publicado y presentado en congreso con los resultados de este presentados en el Capítulo 5.

Fue estudiada la posibilidad de paralelización del código utilizando un paquete de herramientas PETSc, o que dio un resultado negativo para la implementación con el algoritmo SIMPLE con una malla desplazada. El mismo demostró ser una buena herramienta para convección difusión como velocidades conocidas, pudiendo ser fácilmente escalable y de rápida implementación una vez conocido el lenguaje y los comandos.

Respecto al efecto de la variación aspecto en las cavidades, se concluye que el efecto de la variación de afecto es mucho menor que el efecto del número de Rayleigh, sin embargo, puede notarse que, para menor relación de aspecto, el vórtice del flujo se aproxima a la pared más fría, mientras que el mayor diferencial de temperatura tiene a estar situado en las proximidades de las paredes más calientes. Lo que se interpreta como una zona de cambio de presión y aumento de la velocidad.

En cuanto a los distintos esquemas numéricos analizados se puede concluir que no existe diferencias muy significativas entre ellos para bajas diferencias de aspecto o bajos números de Rayleigh, sin embargo las diferencias van aumentando conforme, aumentan o el número de Rayleigh o la relación de aspecto, pero aún así, los resultados son similares.

6.2 Trabajos futuros

El trabajo inmediato para el futuro será el de estudiar otras herramientas de escalabilidad para él código. Y comparar los resultados del nuevo código paralelizado con los resultados actuales e incluir resultados en régimen transitorio.

Además de la paralización del código para uso en computadores personales, otro trabajo a futuro sería la implementación de un código utilizando el paquete de herramientas Open MP y realizar la paralelización usando la herramienta MPI para lograr un código óptimo para lanzar en supercomputadores.

Un objetivo a futuro para la continuidad del trabajo es de incluir otros modelos físicos al código, como forzadores de flujo, modelos de radiación térmica y turbulencia. De manera a lograr su aplicación para dos casos de estudio, la convección forzada y el estudio de los fluidos compresibles.

Referencias bibliográficas

A. W. Date. Introduction to Computational Fluid Dynamics. Cambridge University Press 2005. ISBN 0521 853265.

Aljure, David. (2017). Aerodynamic Analysis of complex geometries using CFD. Centro Tecnológico de transferencia de calor. Departamento de Maquinas y Motores Térmicos. Universidad Politécnica de Cataluña. Recuperado de https://www.researchgate.net/publication/317565202_Aerodynamic_Analysis_of_complex_geometries_using_CFD

Anderson D., Tannehill J., Pletcher R. (1984). Computational fluid mechanics and heat transfer. Hemisphere, New York

Cheng, A. and D. T. Cheng (2005). Heritage and early history of the boundary element method, Engineering Analysis with Boundary Elements, 29, 268–302.

Cortés, Magdalena, Fazio, Paul, Rao, Jiwu, Bustamante, Waldo, & Vera, Sergio. (2014). Modelación CFD de casos básicos de convección en ambientes cerrados: Necesidades de principiantes en CFD para adquirir habilidades y confianza en la modelación CFD. Revista ingeniería de construcción, 29(1), 22-45. https://dx.doi.org/10.4067/S0718-50732014000100002

CTTC. A3-A Two-dimensional Steady Convection-Diffusion Equation: the Smith-Hutton problema. Universidad Politécnica de Cataluña.

E. Immonen, A parametric morphing method for generating structured meshes for marinefree surface flow applications with plane symmetry, Journal of Computational Design and Engineering (2018), doi:https://doi.org/10.1016/j.jcde.2018.11.002

F.H. Harlow (1955). "A Machine Calculation Method for Hydrodynamic Problems". Los Alamos Scientific Laboratory report LAMS-1956.

Frank M. White. Dinámica de fluidos. Mc Graw Hill 2014. I SBN 9788448166038

Fromm, J. E.; F. H. Harlow (1963). "Numerical solution of the problem of vortex street development". Physics of Fluids. 6 (7): 975.

Fuentes, D. A., Guerrero, P. J., & Sánchez, R. (2013). Cálculo del Flujo Difusivo en Dominios Complejos Mediante el Método de Volúmenes Finitos. Revista UIS Ingenierías, ISSN-e 2145-8456, ISSN 1657-4583, Vol. 11, Nº. 1, 2012, págs. 45-54. Recuperado de https://dialnet.unirioja.es/descarga/articulo/6299780.pdf

Garcia, J. D. (2016). NUEVO MODELO NUMÉRICO DE UN COMPRESOR AXIAL POR SIMULACIÓN EN 2D. Dyna, 91(2), 180-187. Recuperado de https://dialnet.unirioja.es/servlet/articulo?codigo=5391100

García, M. J., Boulanger, P., Duque, J. C., & Giraldo, S. (2008). Análisis CFD de vientos convectivos naturales debidos a la temperatura de un terreno basado en un modelo DEM integrado con imágenes infrarrojas Landsat. Ingeniería y Ciencia, ISSN 1794–9165 Volumen 4, número 8, diciembre de 2008, páginas 65–84. Recuperado de http://publicaciones.eafit.edu.co/index.php/ingciencia/article/view/214

Ghia, U.; Ghia, K.N. e Shin, C. High-re solutions for incompressible flow using the navierstokes equations and a multigrid method. Journal of computational physics, v. 48, n. 3, 387–411,1982.

Gualdrón, J. A., Mora, L. A., & Mateus, F. A. (2013). CFD simulation of crude oil homogenization in pilot plant scale. Ciencia Tecnologia y Futuro, 5(2), 19-30. Recuperado de https://dialnet.unirioja.es/descarga/articulo/4697665.pdf

Henriksen, Mathias & Vaagsaether, K & Gaathaug, André & Lundberg, Joachim & Forseth, Sissel & Bjerketvedt, Dag. (2019). Laminar burning velocity measurements for an outwardly propagating flame of dimethyl carbonate and air mixtures Conference: 9th International Seminar on Fire and Explosion Hazards, At St. Petersburg.

Hess, J.L.; A.M.O. Smith (1967). "Calculation of Potential Flow About Arbitrary Bodies". Progress in Aerospace Sciences. 8: 1–138

Issa R. I., Gosman A. D., Watkins A. P. (1986) The computation of compressible and incompressible recirculating flows by a non-iterative implicit scheme. J. Computational Physics, Vol. 62 pages 66-82.

Jameson, A., Schmidt, W. and Turkel, E., "Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA paper 81-1259, presented at the AIAA 14th Fluid and Plasma Dynamics Conference, Palo Alto California, 1981.

Ji, Yingchun. "CFD modelling of natural convection in air cavities." CFD Letters 6, no. 1 (2014): 15-31.

Jines, J. L. (2017). Simulación de Flujo Incompresible y de una Fase en Accesorio de Tuberías (Codo 90° de alto radio) mediante Dinámica de Fluidos Computacional (CFD), para Cálculo de Factor de Pérdida KL. Revista Tecnológica ESPOL – RTE, Vol. 30, N. 2, 56-74 (Agosto 2017) Recuperado de http://rte.espol.edu.ec/index.php/tecnologica/article/download/588/366

J. H. Ferziger y M. Peric Computational methods for fluid dynamics. 3ra Edición. Springer. 2002

J. P. Holman. Transferencia de calor. 10ma Edición. McGraw Hill 1999. ISBN 0-07-0229618-9

J. Xamán, M. Gijón-Rivera. Dinámica de fluidos computacional para ingenieros. Palibrio 2015.

Marco A Guevara L & Luis C Belalcazar C. (2018). CFD modeling and evaluation of a bi-stable micro-diverter valve. Ciencia Tecnologia y Futuro, 8(1), 77-84. Recuperado de https://dialnet.unirioja.es/servlet/articulo?codigo=6553078

Mateus, F. A., & Gualdrón, J. A. (2011). CFD Technique to calculate tube skin peak temperatures in refinery furnaces. Ciencia Tecnologia y Futuro, 4(4), 73-88. Recuperado de https://dialnet.unirioja.es/servlet/articulo?codigo=3844738

Milne-Thomson, L.M. (1973). Theoretical Aerodynamics. Physics of Fluids. Dover Publications. ISBN 978-0-486-61980-4.

Moll, F., Manuele, D., Núñez, M. G., Zabaleta, A. d., & García, A. F. (2011). Caracterización del tipo de cavitación mediante dinámica computacional de fluidos para posteriores

aplicaciones al estudio experimental del daño por cavitación. Mecánica Computacional Vol XXX, págs. 435-450. Recuperado de https://upcommons.upc.edu/handle/2117/15553

Montoya, T. L., Orozco, M. G., & Londoño, C. N. (2012). Evaluación computacional del flujo a través de membranas porosas. ITECKNE: Innovación e Investigación en Ingeniería, ISSN-e 2339-3483, ISSN 1692-1798, Vol. 9, Nº. 2, 2012, págs. 85-94. Recuperado de https://dialnet.unirioja.es/servlet/articulo?codigo=4991600

Murman, Earll and Cole, Julian, "Calculation of Plane Steady Transonic Flow," AIAA paper 70-188, presented at the AIAA 8th Aerospace Sciences Meeting, New York New York, January 1970.

Ordoñez-Viñán, M. A., Aquino-Arroba, S. M., Orozco-Cantos, L. S., Pozo-Safla, E. R., & Jácome-Domínguez, E. A. (2018). Modelización CFD para determinar el comportamiento del fluido en tuberías de PVC. Dominio de las Ciencias, 4(1), 434-446. Recuperado de https://dialnet.unirioja.es/descarga/articulo/6313253.pdf

Patankar, S. V. (1980). Numerical Heat Transfer and Fluid Flow. Taylor & Francis. ISBN 978-0-89116-522-4.

Pérez, M. M., Patiño, G. L., Amparo, P., & Jiménez, L. (2013). CFD model of air movement in ventilated façade: comparison between natural and forced air flow. International Journal of Energy and Environment, 4(3), 357-368. Recuperado de https://riunet.upv.es/handle/10251/45770.

Prasad, Bibin & Varghese, A & Jacob, J & Jayaram, M & Sathyajith, Susan & Krishna, S. (2013). CFD analysis of merged airfoil for incompressible and compressible flows. International Journal of Applied Engineering Research. 8. 259-271.

P. Gerhart, R. Gross, J. Hochstein, "Fundamentos de Mecanica de Fluidos", 2da.edicion, ed. Addison-Wesley iberoamericana. Eua.1992

Reynolds, O. (1883). An Experimental Investigation of the Circumstances Which Determine Whether the Motion of Water Shall Be Direct or Sinuous, and of the Law of Resistance in Parallel Channels. Philosophical Transactions of the Royal Society of London, 174, 935-982. Retrieved from http://www.jstor.org/stable/109431

Richardson, L. F. (1922). Weather prediction by numerical process. Publicado por la prensa de la universidad de Cambridge.

R. M. Smith and A. G. Hutton. The numerical treatment of advection: a performance comparison of current methods. Numerical Heat Transfer, 5:439–461, 1982.

Tritton, D. J Physical Fluid Dynamics. New York Van Nostrand Reinhold, 1977.

Turner, M., R. W. Clough, H. C. Martin y L. J. Topp, "Stiffness and Deflection Analysis of Complex Structures", J. Aeronautical Science 23 (9), pp. 805-823, Septiembre de 1956

Versteeg HK, Malalasekera W. An introduction to computational fluid dynamics: the finite volume method. Pearson Education; 2007

Van Doormaa J., Raithby G. (1984) Enhancements of the SIMPLE method for predicting incompressible fluid flow. Numerical Heat Transferer, Vol. 7, pages 147-163.

Zienkiewicz, O.C., Taylor, R.L. and Nithiarasu, P., *The finite element method for fluid dynamics*. 6th edition, Elsevier (2006). eBook ISBN: 9780080455594

ANEXO A – Resultados del esquema UPWIND

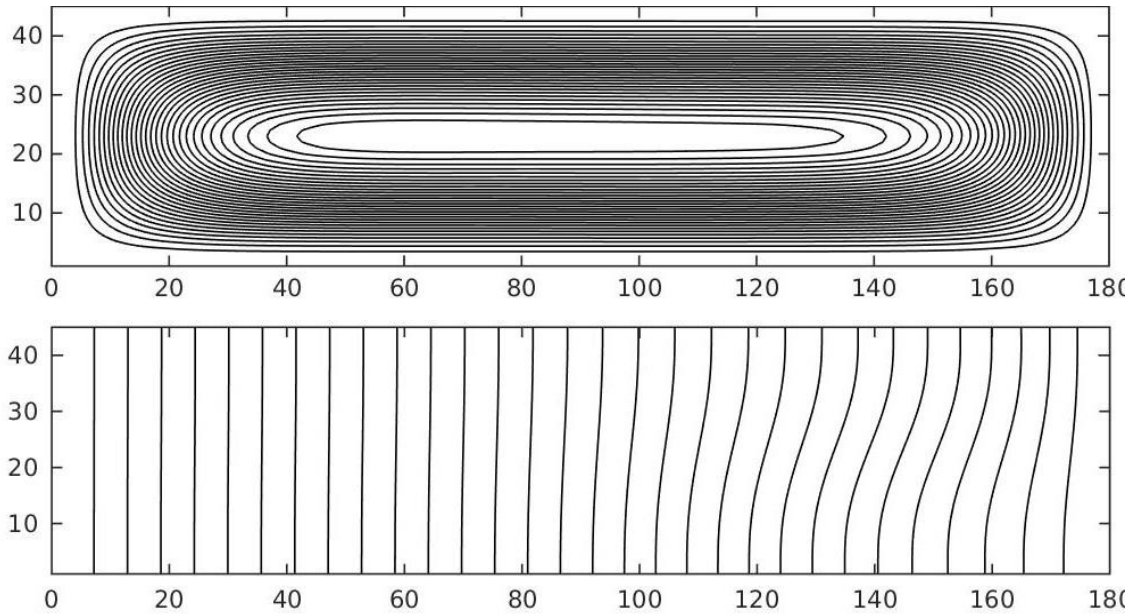

Figura A. 1 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 0,25; Ra=103 y esquema numérico UPWIND.

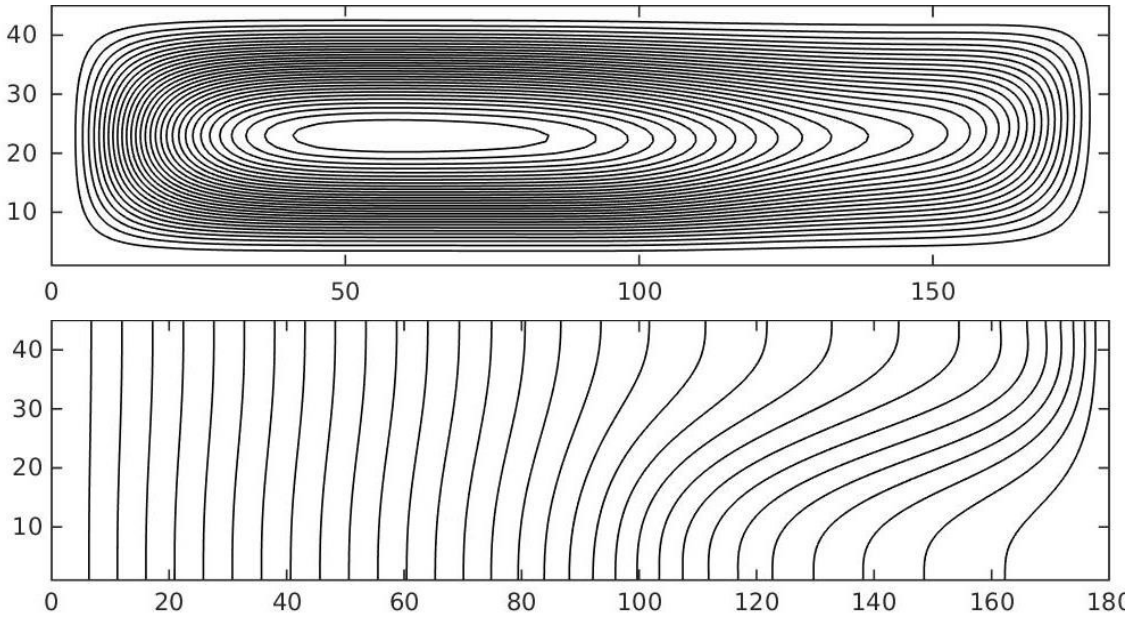

Figura A. 2 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 0,25; Ra=10^4 y esquema numérico UPWIND.

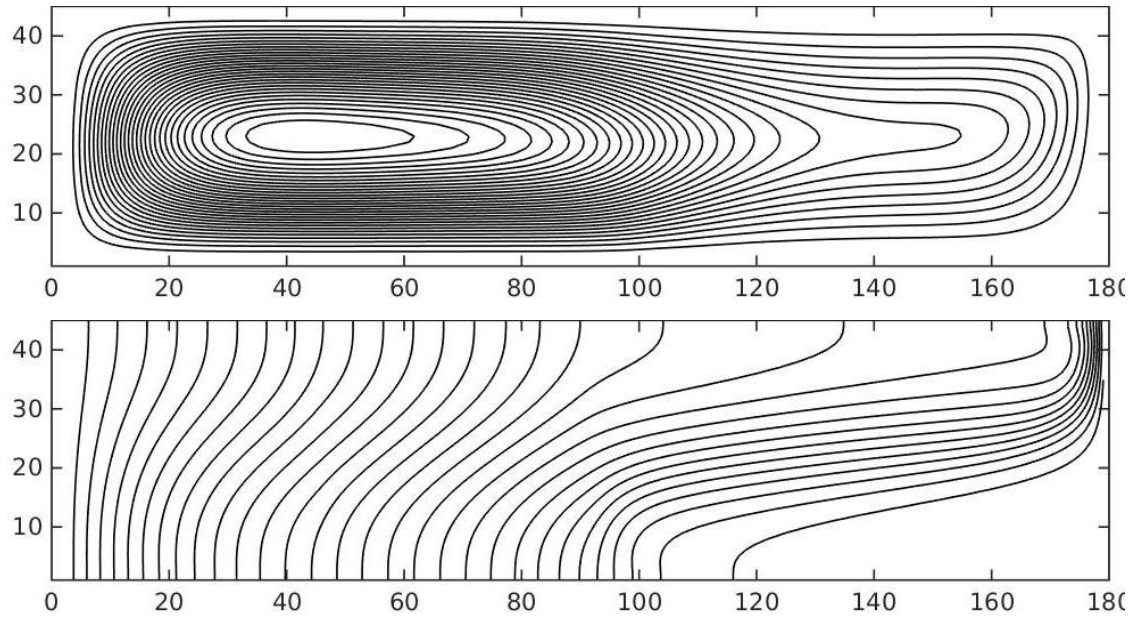

Figura A. 3 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 0,25; Ra=10^5 y esquema numérico UPWIND.

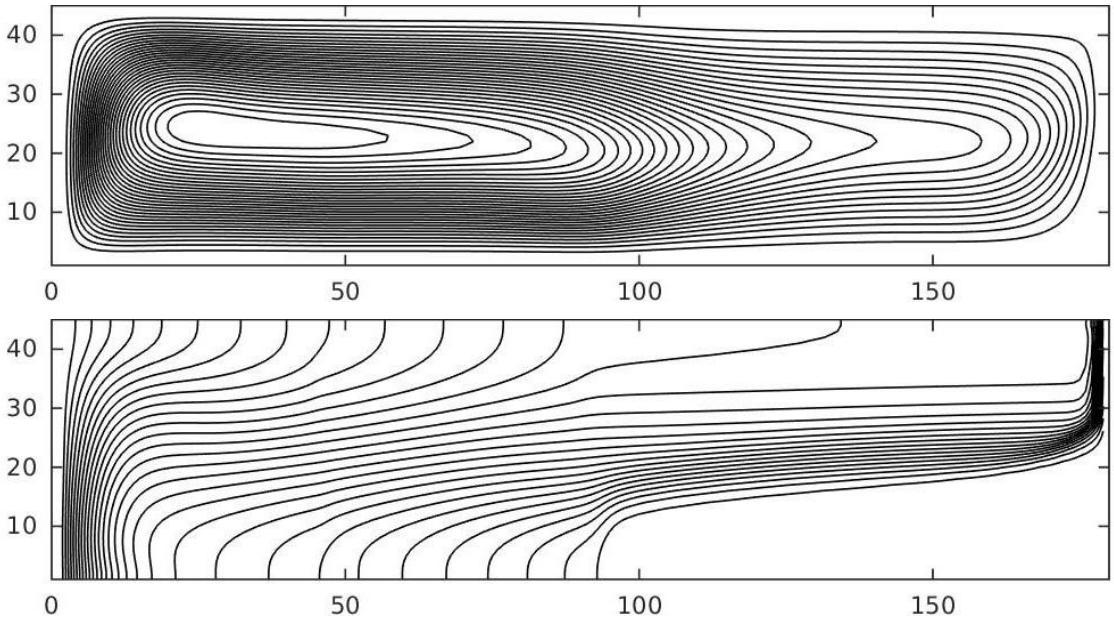

Figura A. 4 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 0,25; Ra=10^6 y esquema numérico UPWIND.

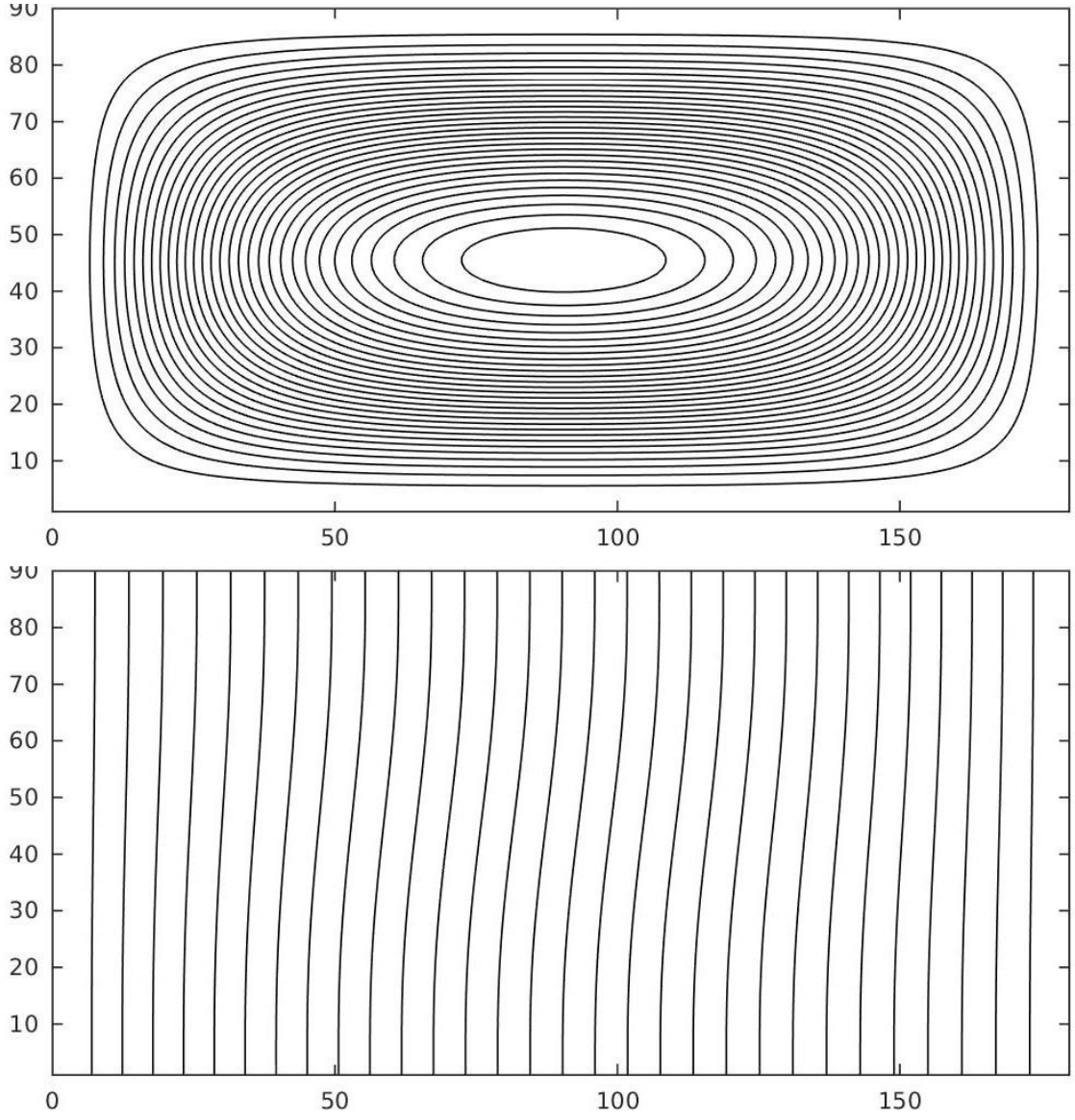

Figura A. 5 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 0,5; Ra=10^3 y esquema numérico UPWIND.

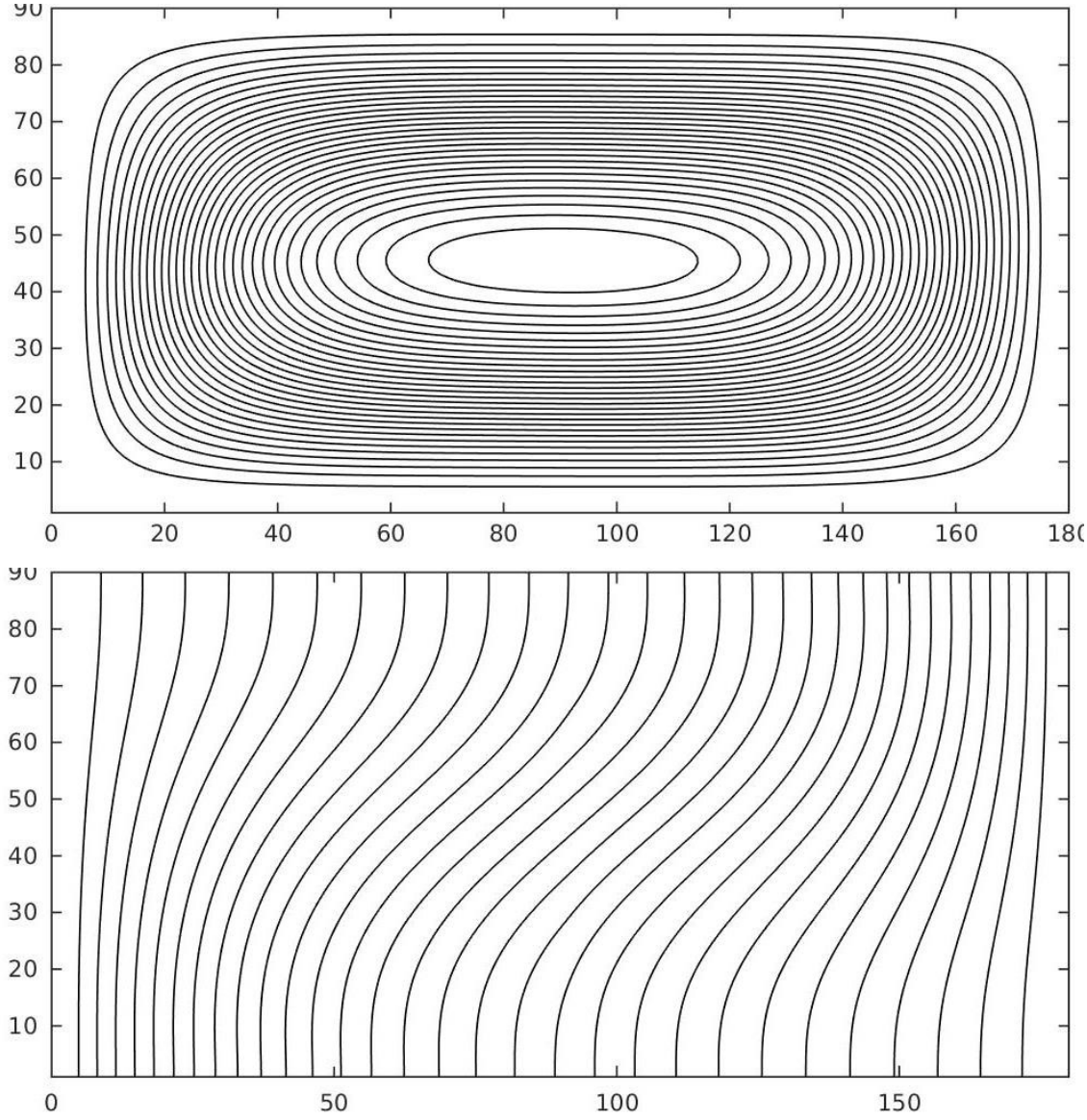

Figura A. 6 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 0,5; Ra=10^4 y esquema numérico UPWIND.

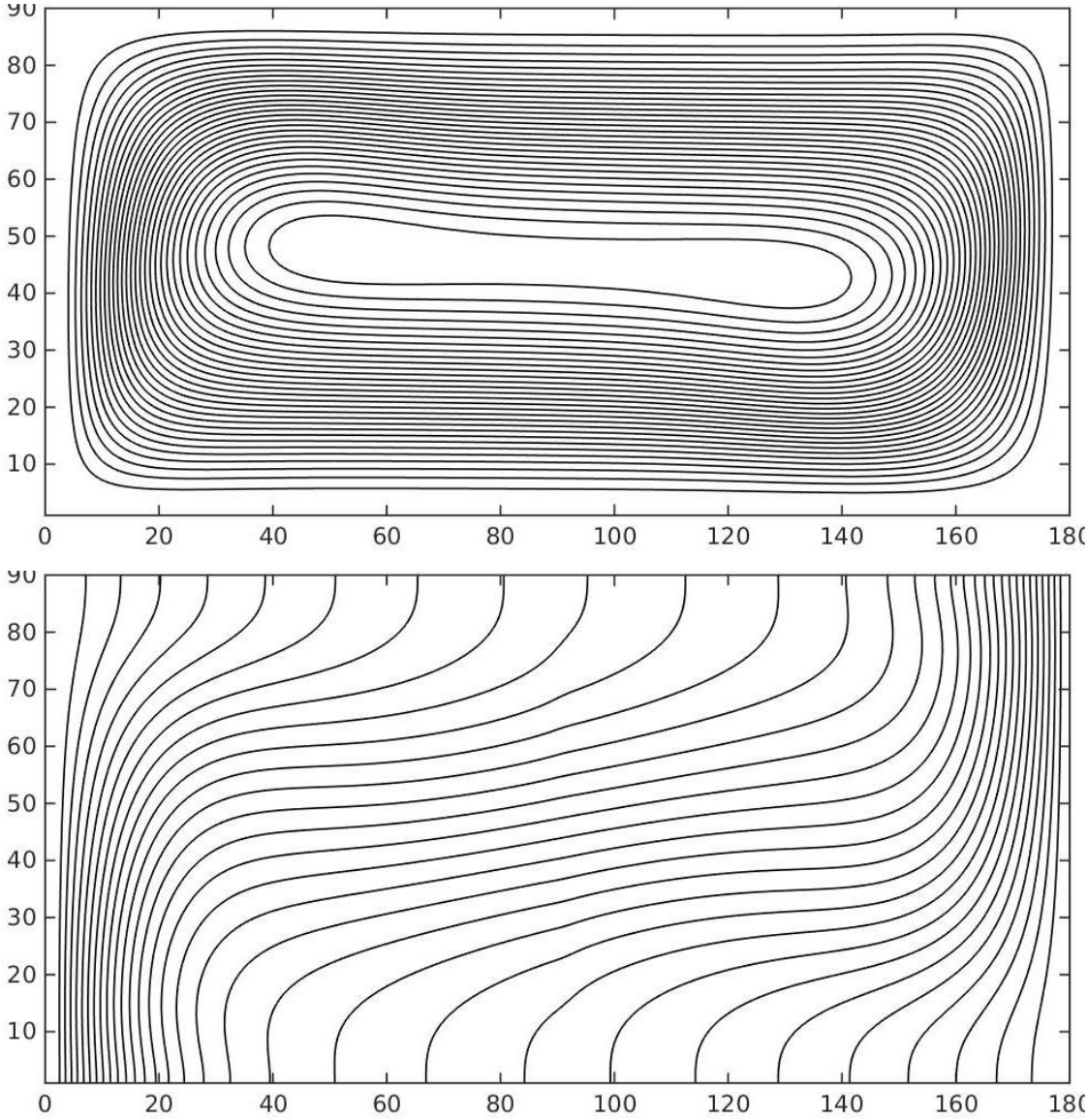

Figura A. 7 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 0,5; Ra=10^5 y esquema numérico UPWIND.

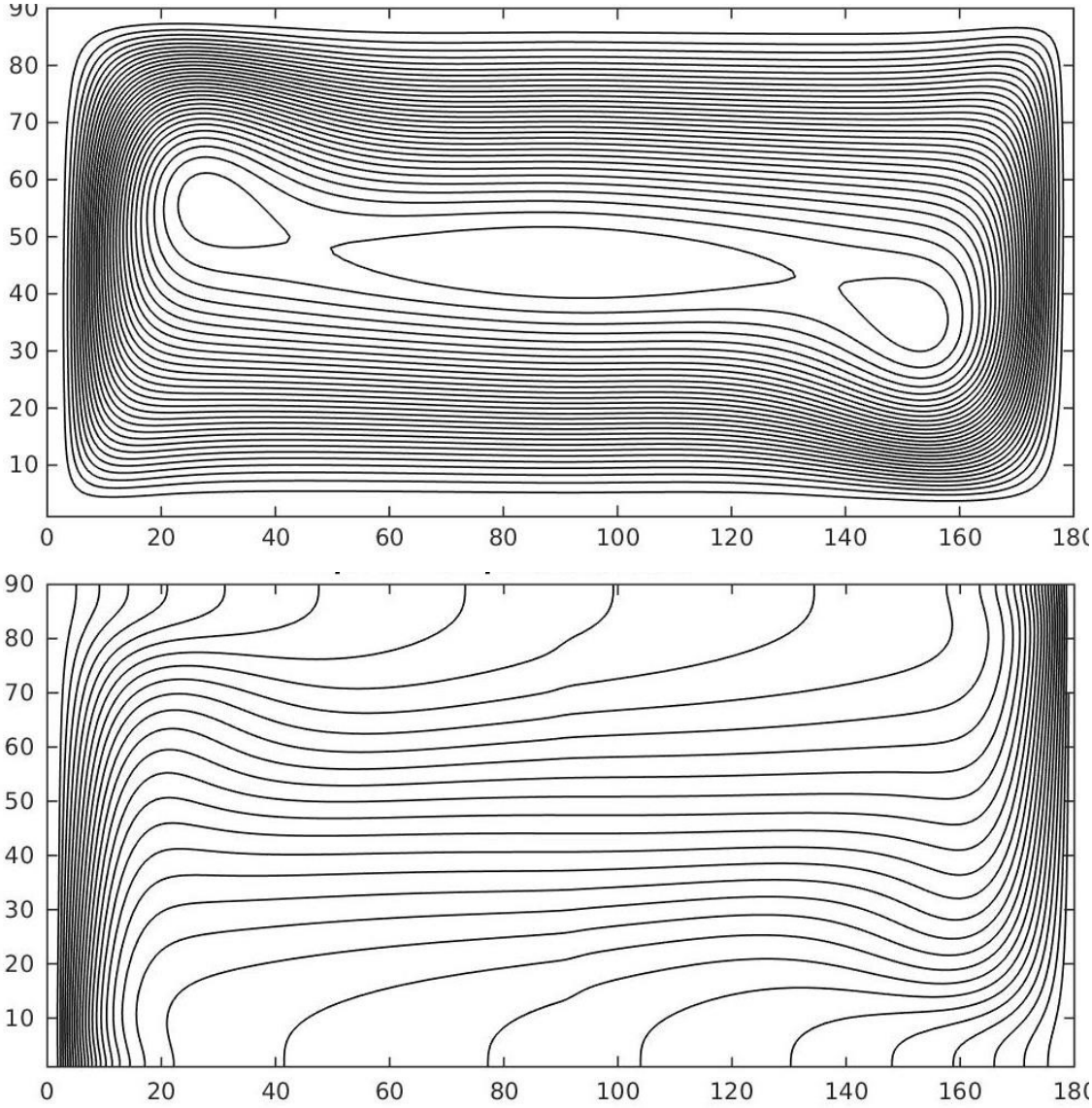

Figura A. 8 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 0,5; Ra=106 y esquema numérico UPWIND.

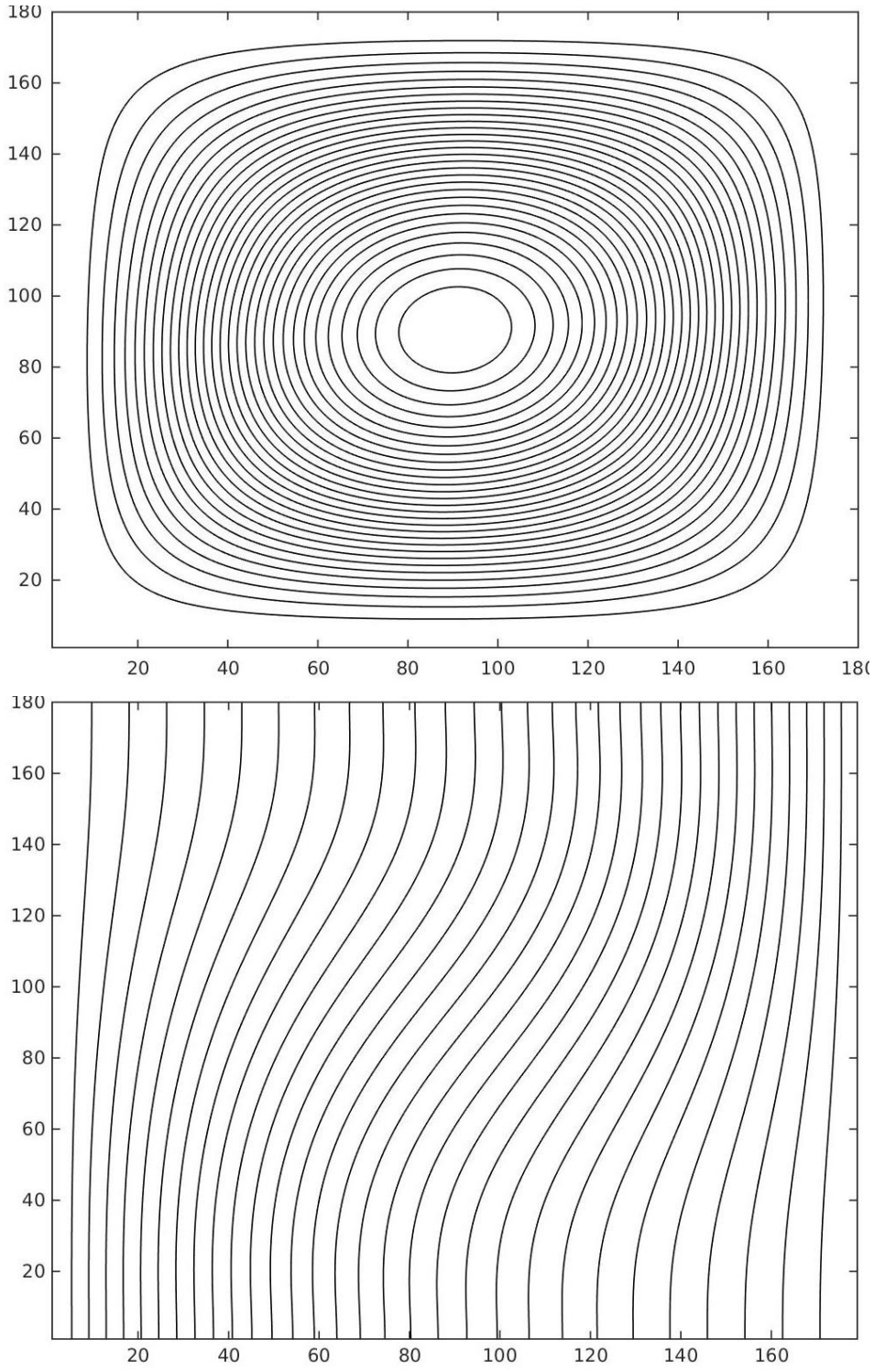

Figura A. 9 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 1; Ra=10^3 y esquema numérico UPWIND.

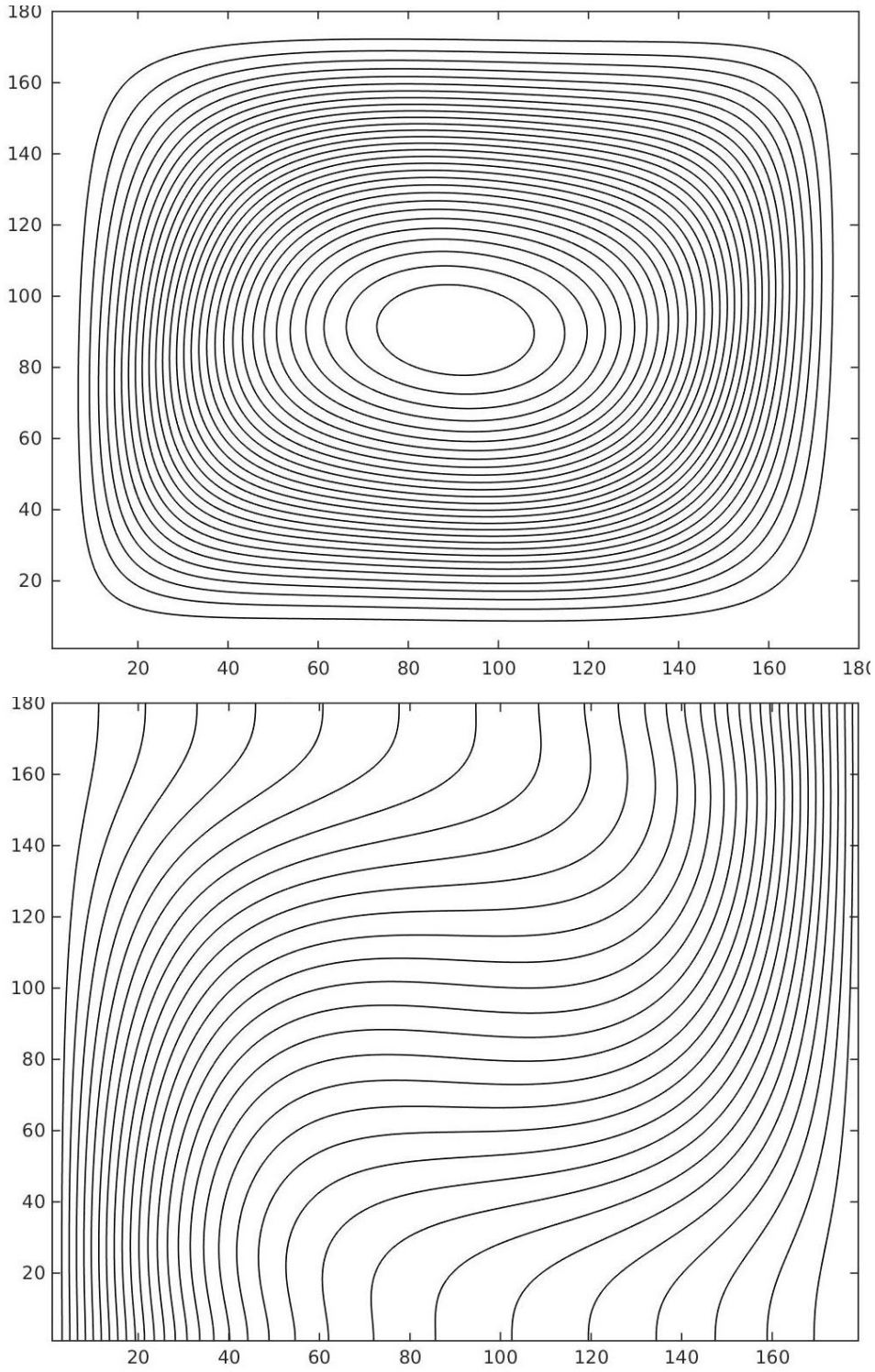

Figura A. 10 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 1; Ra=10^4 y esquema numérico UPWIND.

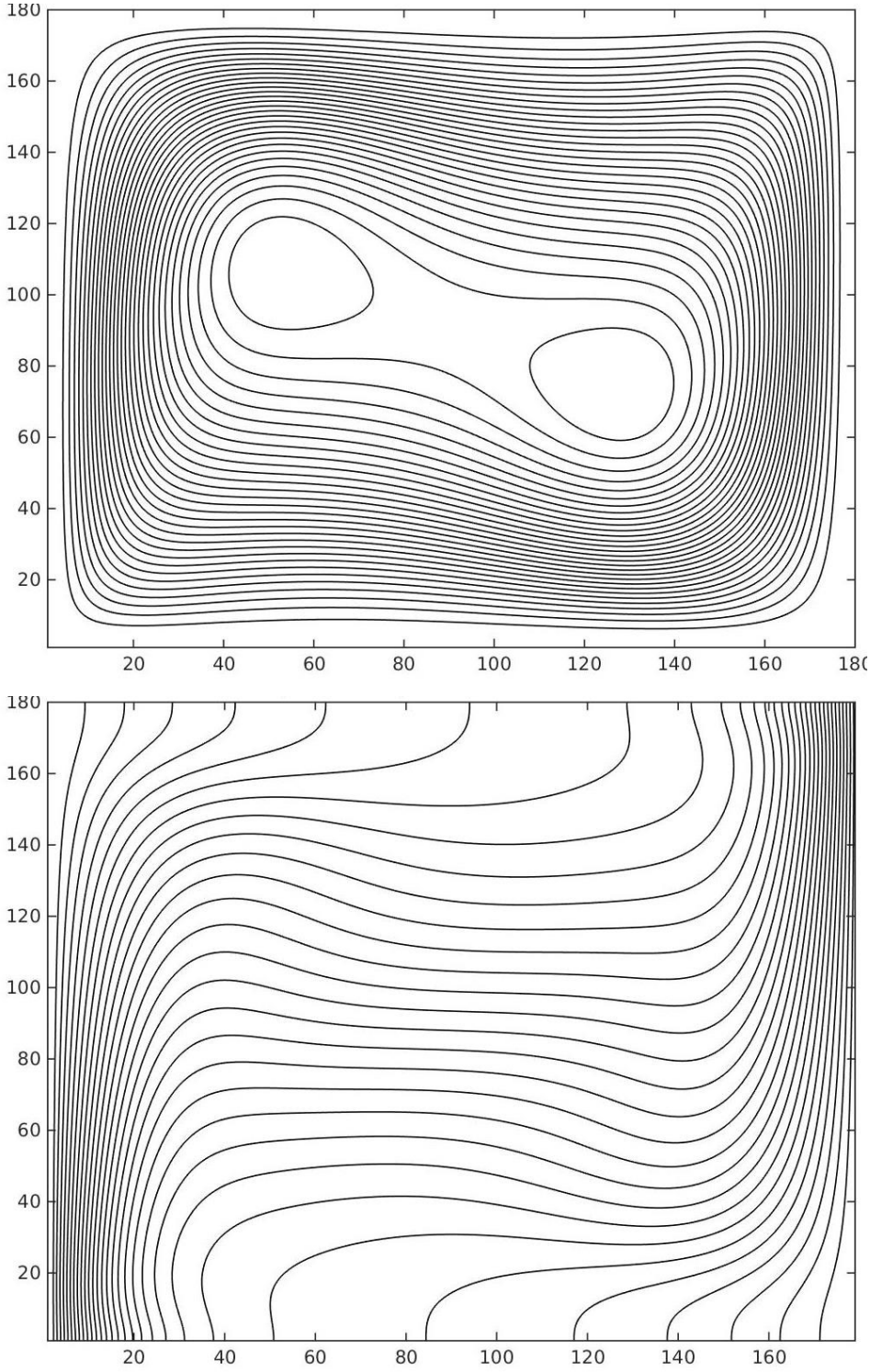

Figura A. 11 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 1; Ra=10^5 y esquema numérico UPWIND.

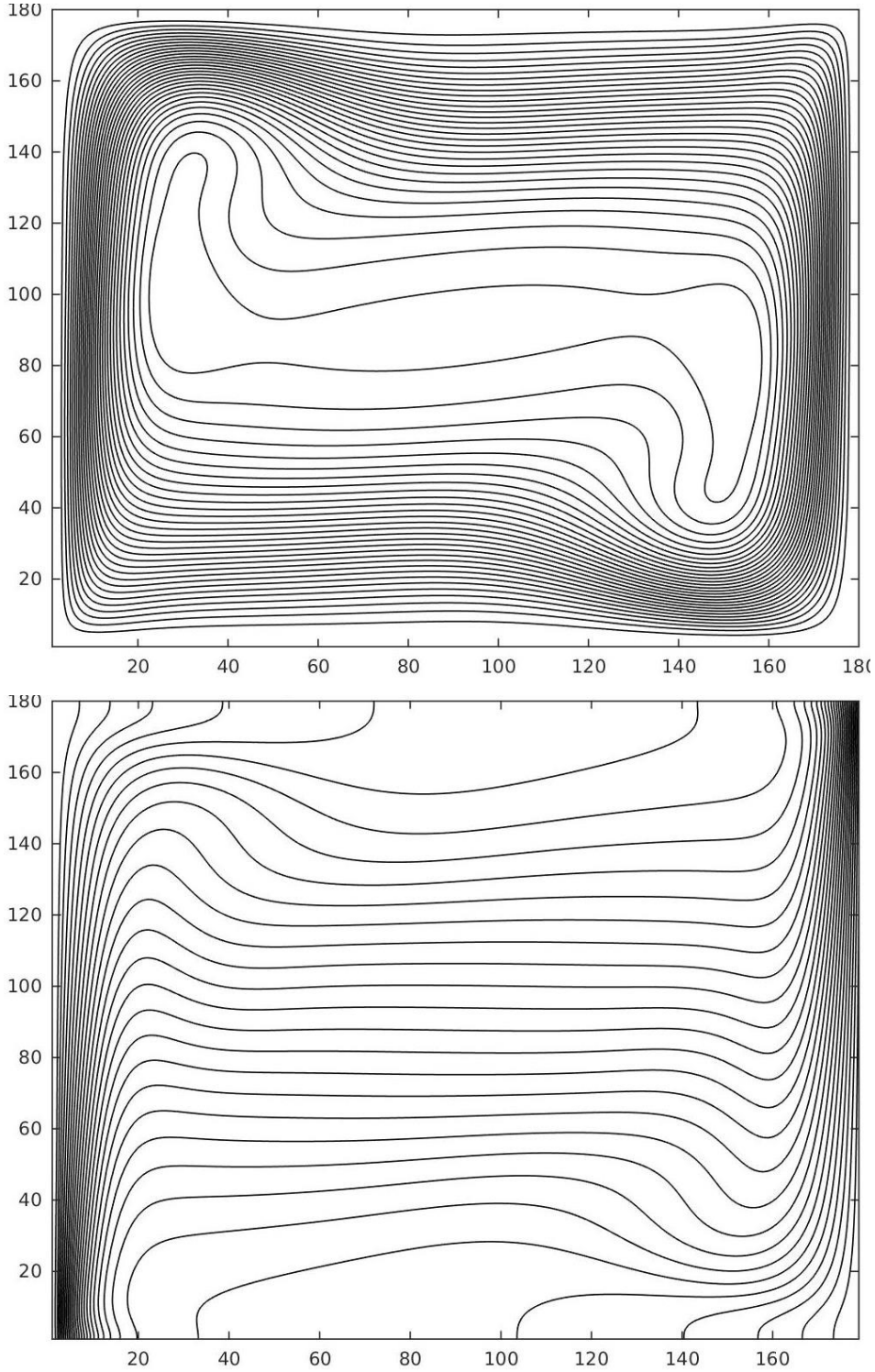

Figura A. 12 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 1; Ra=10^6 y esquema numérico UPWIND.

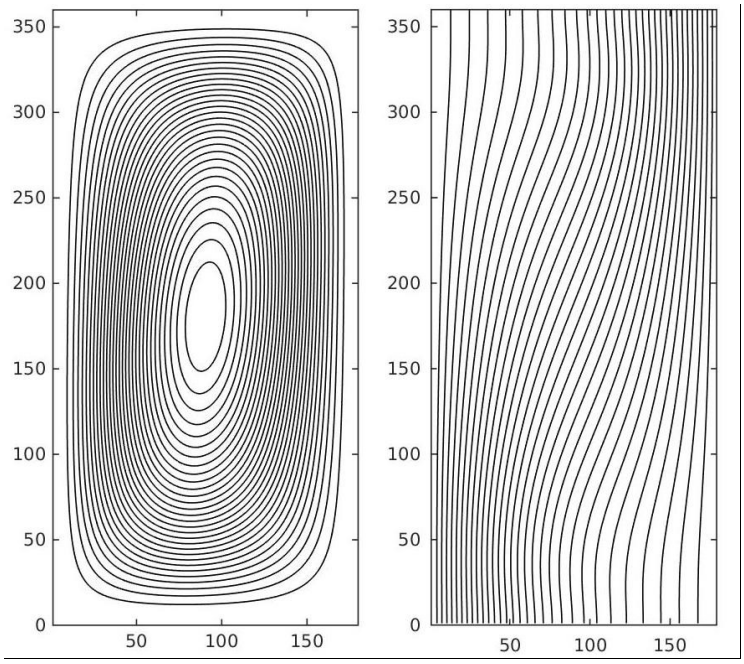

Figura A. 13 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 2; Ra=10^3 y esquema numérico UPWIND.

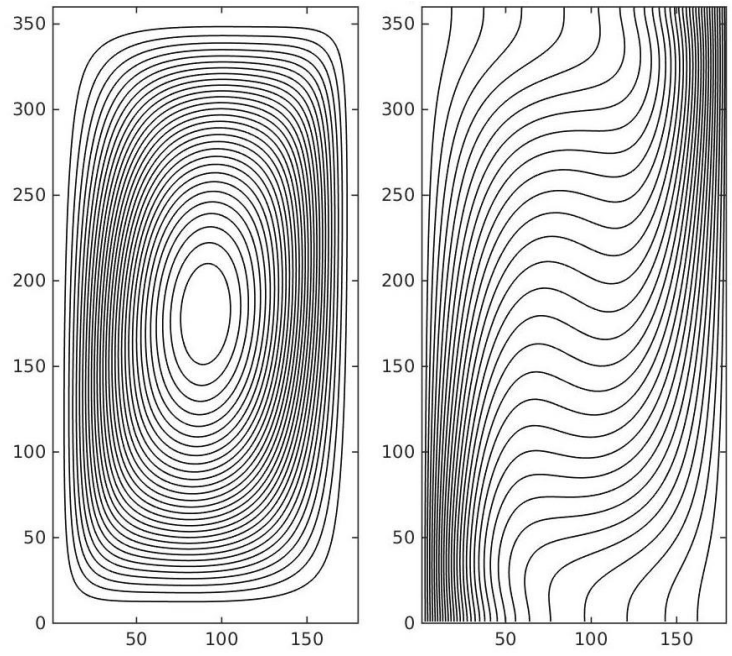

Figura A. 14 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 2; Ra=10^4 y esquema numérico UPWIND.

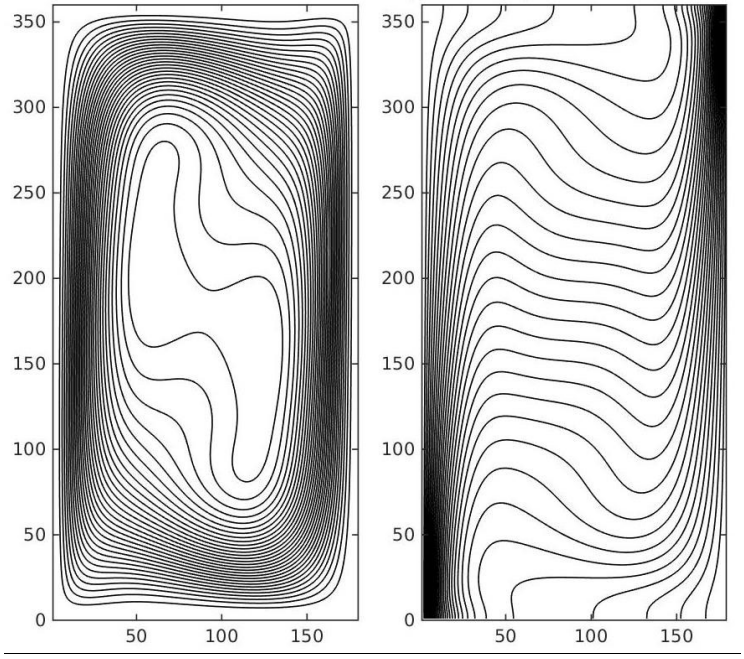

Figura A. 15 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 2; Ra=10^5 y esquema numérico UPWIND.

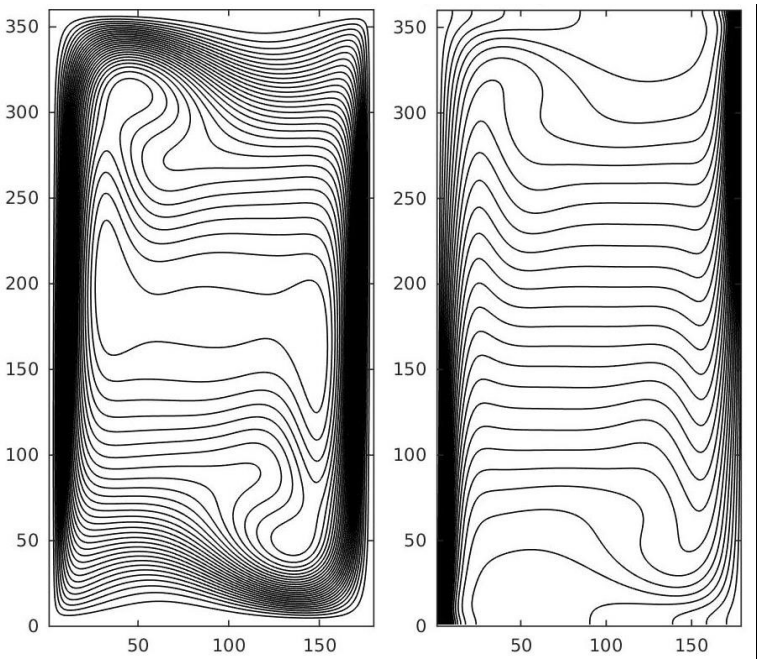

Figura A. 16 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 2; Ra=10^6 y esquema numérico UPWIND.

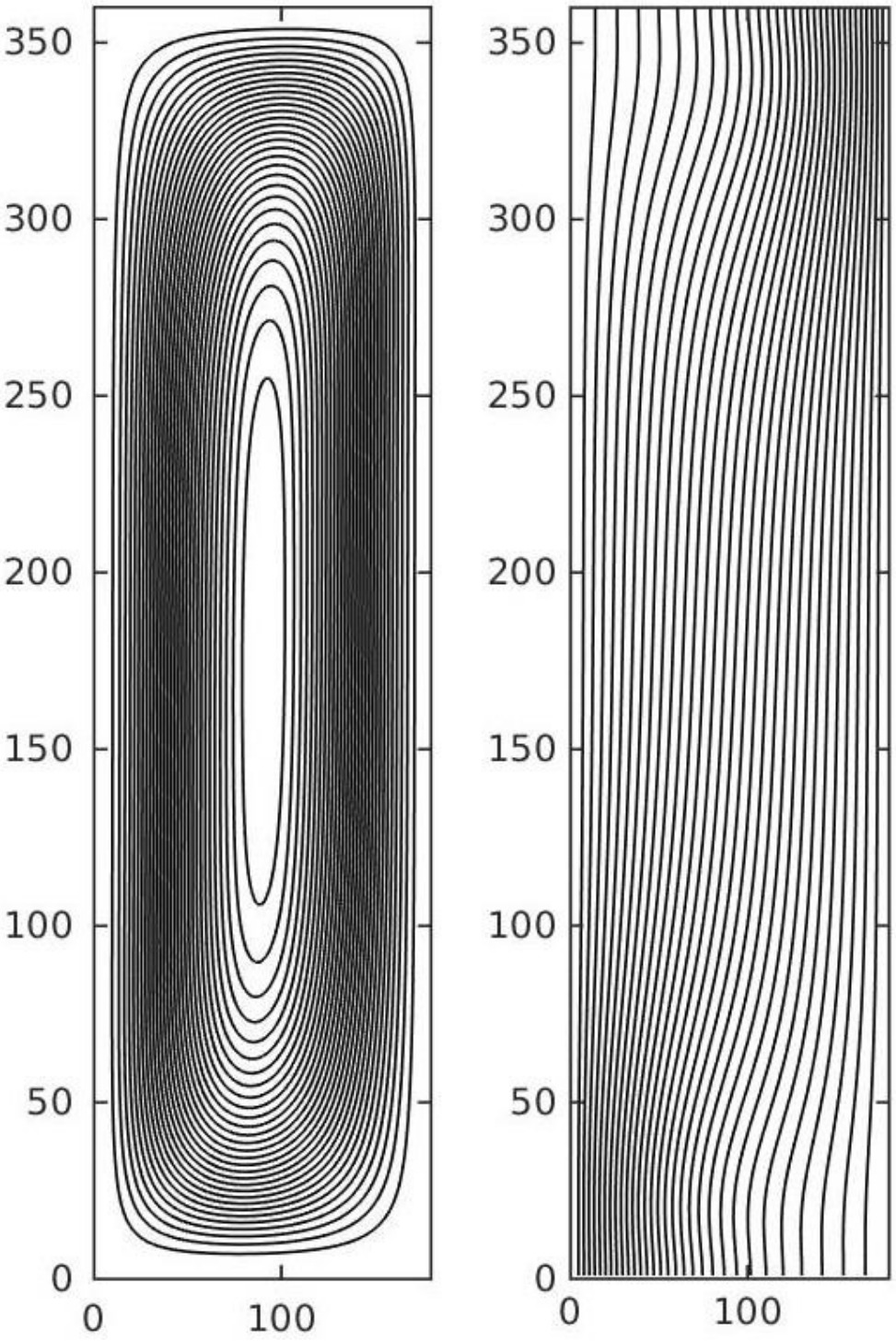

Figura A. 17 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 4; Ra=10^3 y esquema numérico UPWIND.

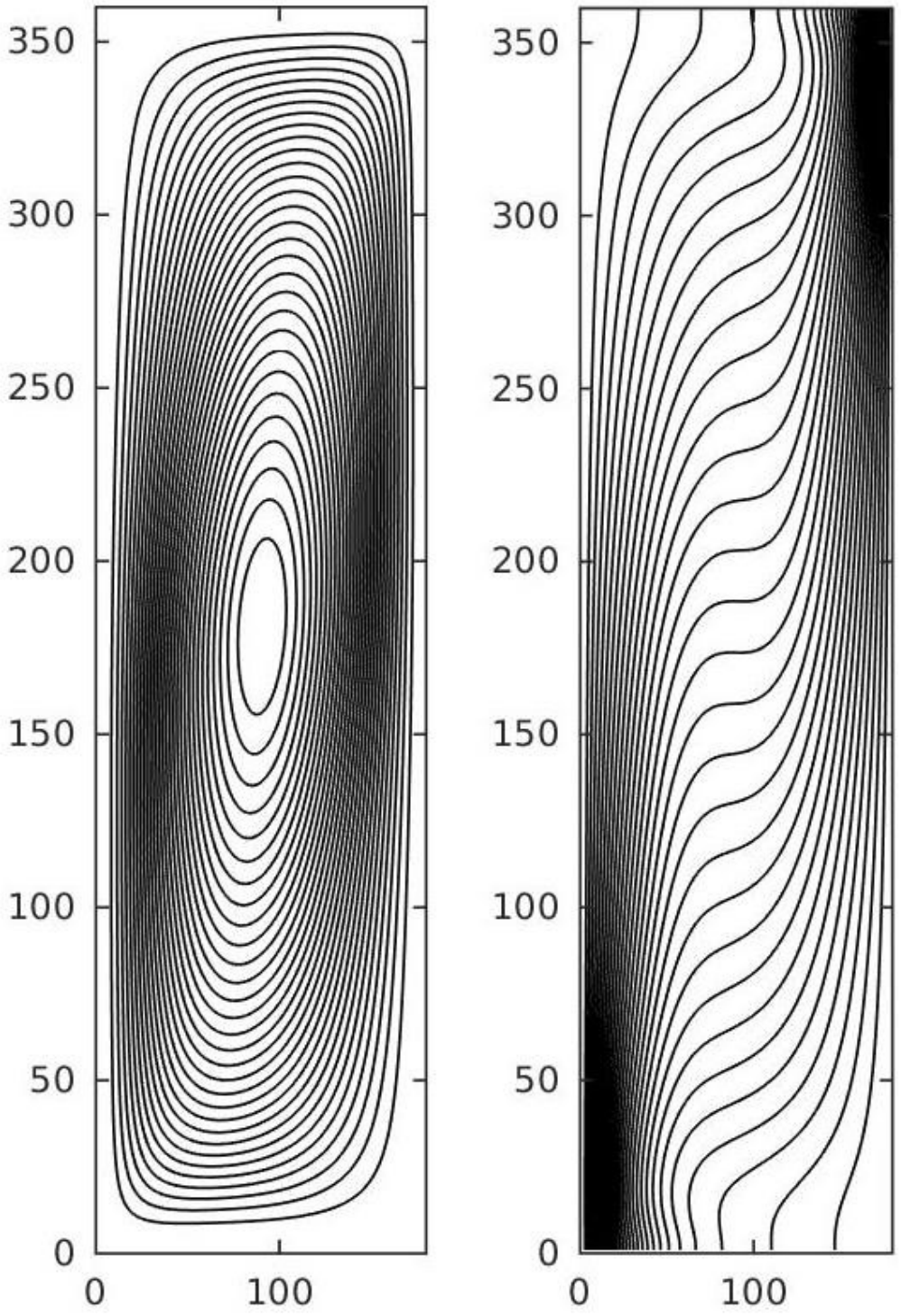

Figura A. 18 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 4; Ra=10^4 y esquema numérico UPWIND.

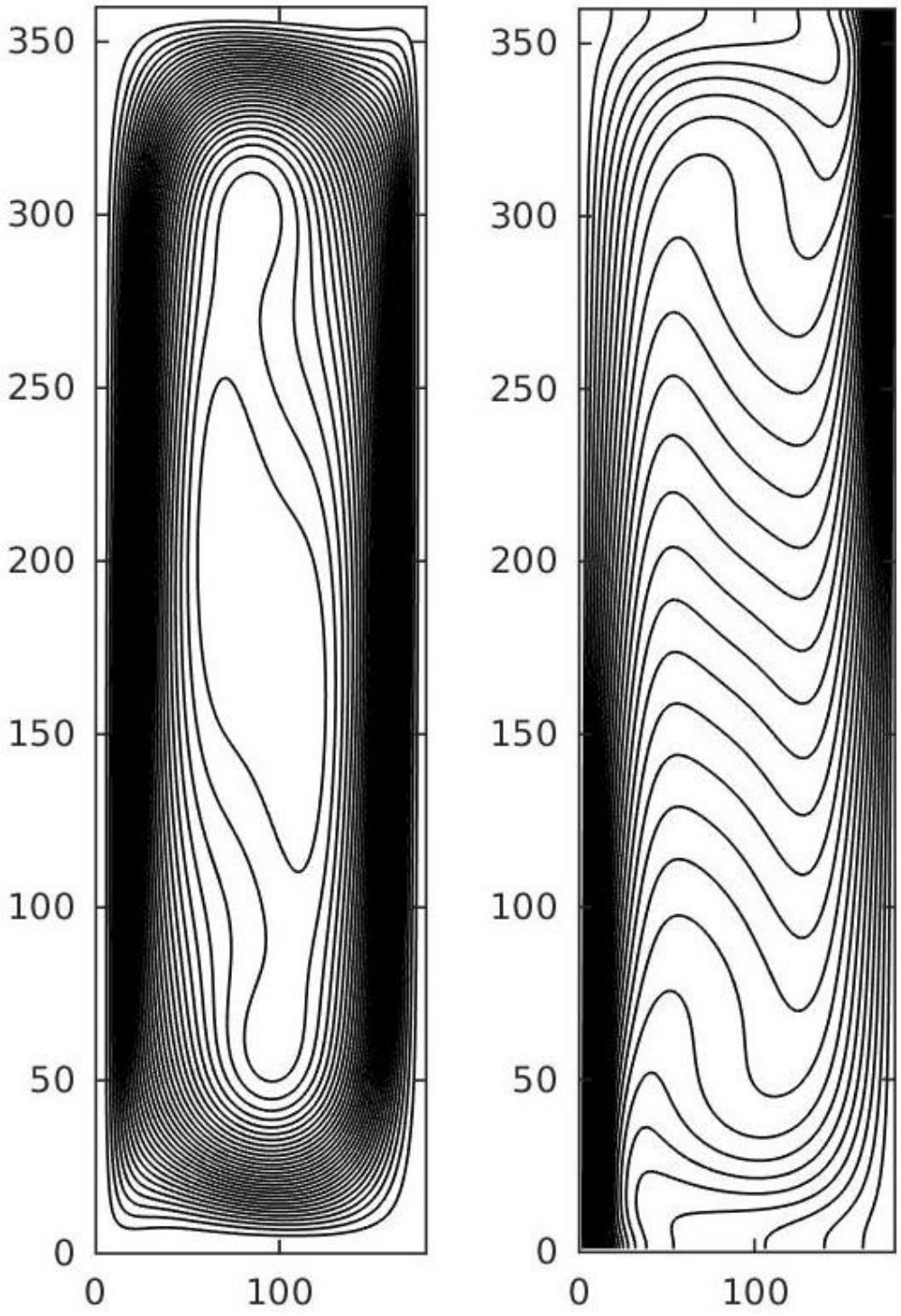

Figura A. 19 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 4; Ra=10^5 y esquema numérico UPWIND.

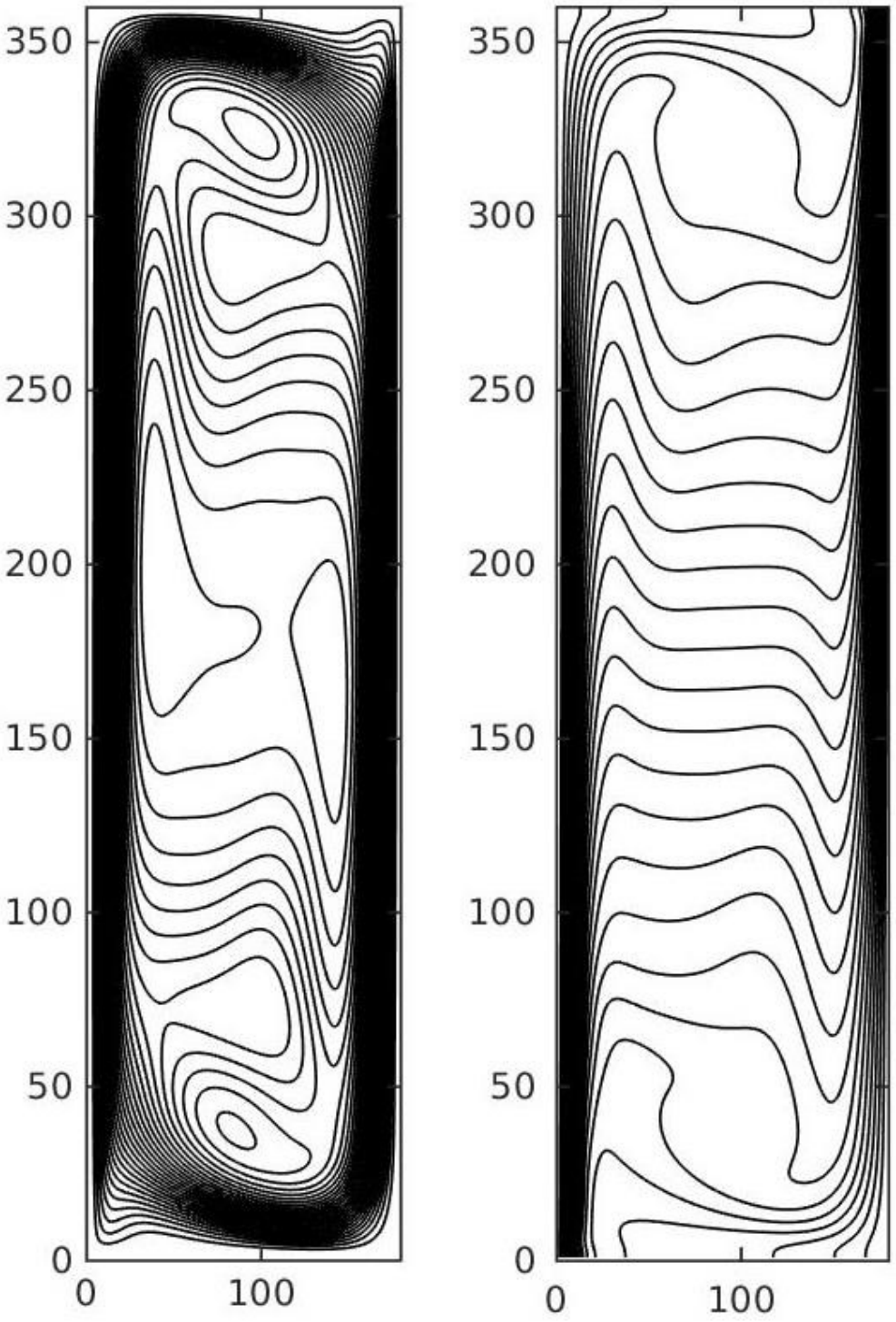

Figura A. 20 Figuras de líneas de corriente y campo de temperatura para la relación de aspecto 4; Ra=10^6 y esquema numérico UPWIND.

ANEXO B – Resultados del esquema LEY DE POTENCIA

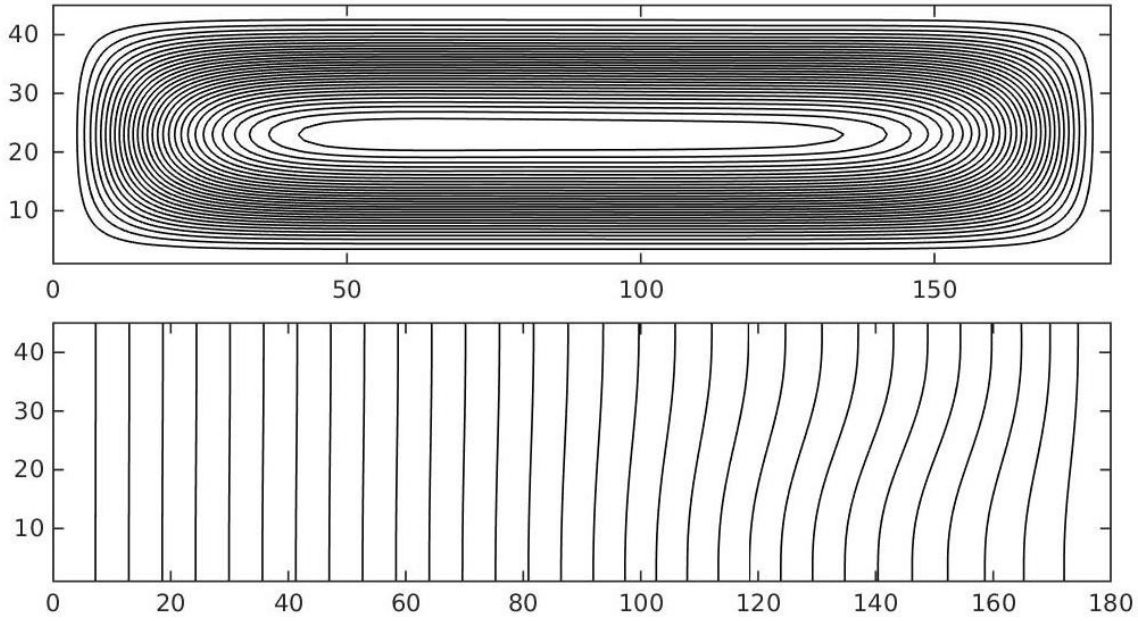

Figura B. 1 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 0,25; Ra=10^3 y esquema numérico LEY DE POTENCIA.

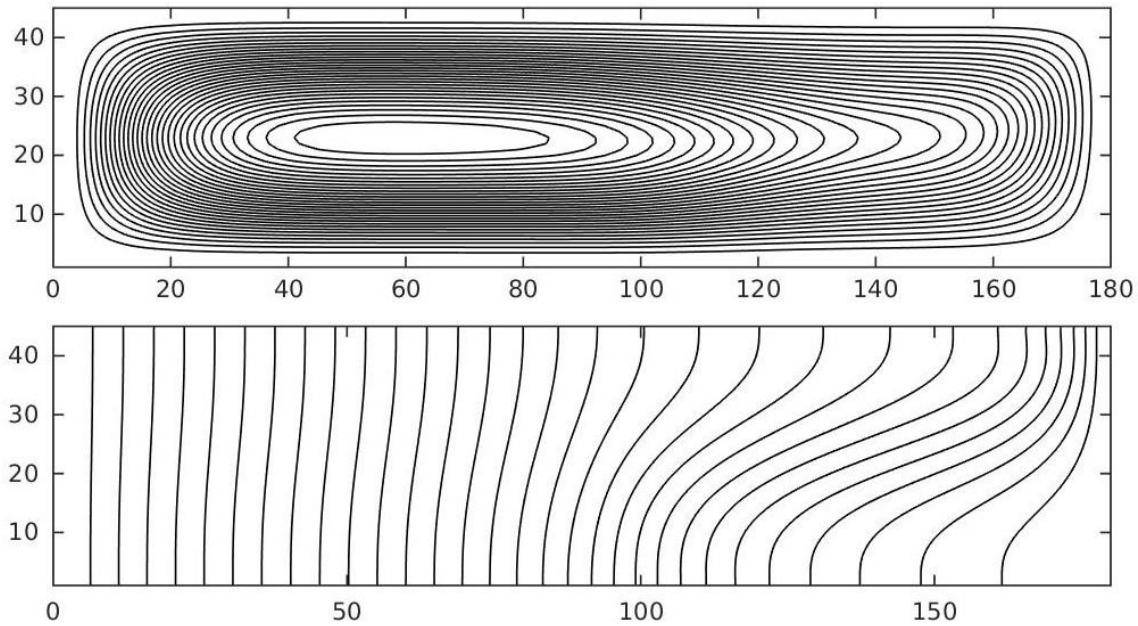

Figura B. 2 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 0,25; Ra=10^4 y esquema numérico LEY DE POTENCIA.

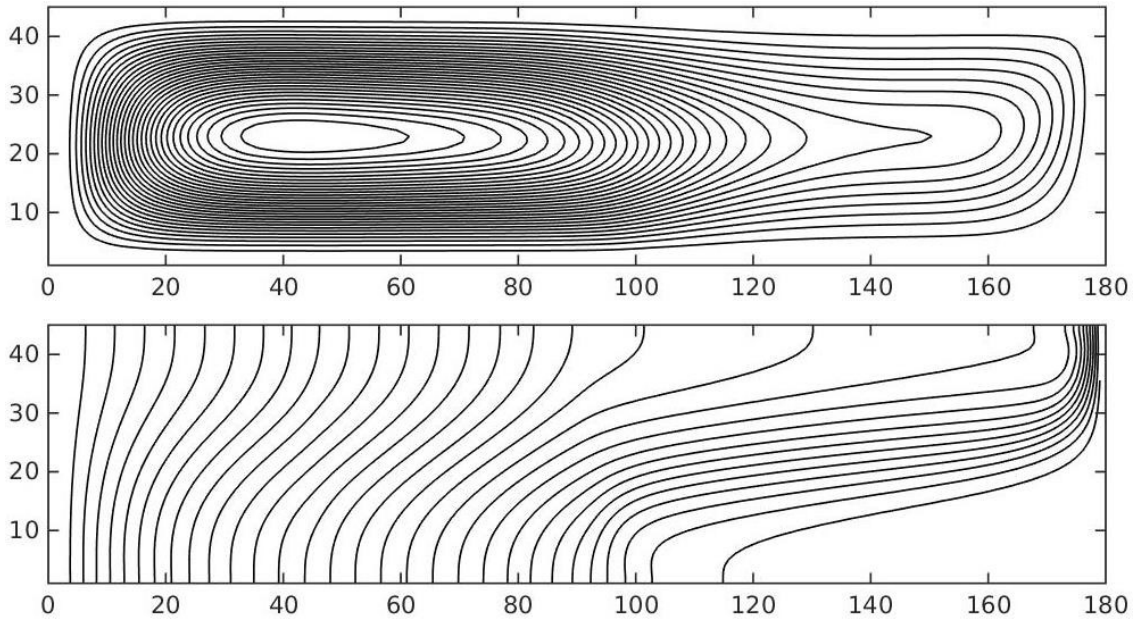

Figura B. 3 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 0,25; Ra=10^5 y esquema numérico LEY DE POTENCIA.

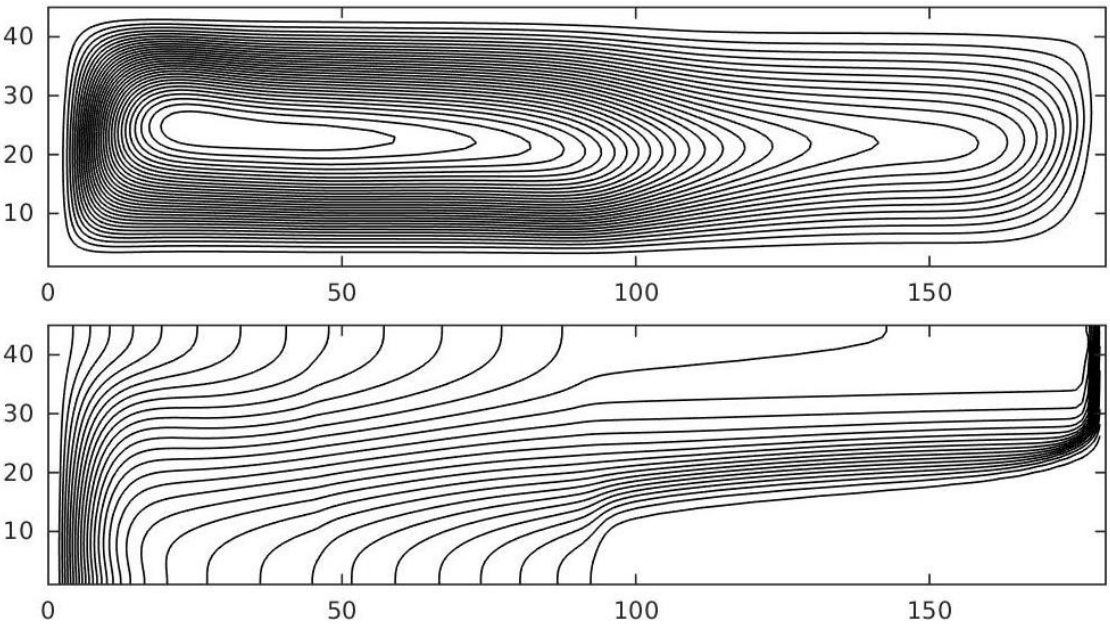

Figura B. 4 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 0,25; Ra=10^6 y esquema numérico LEY DE POTENCIA.

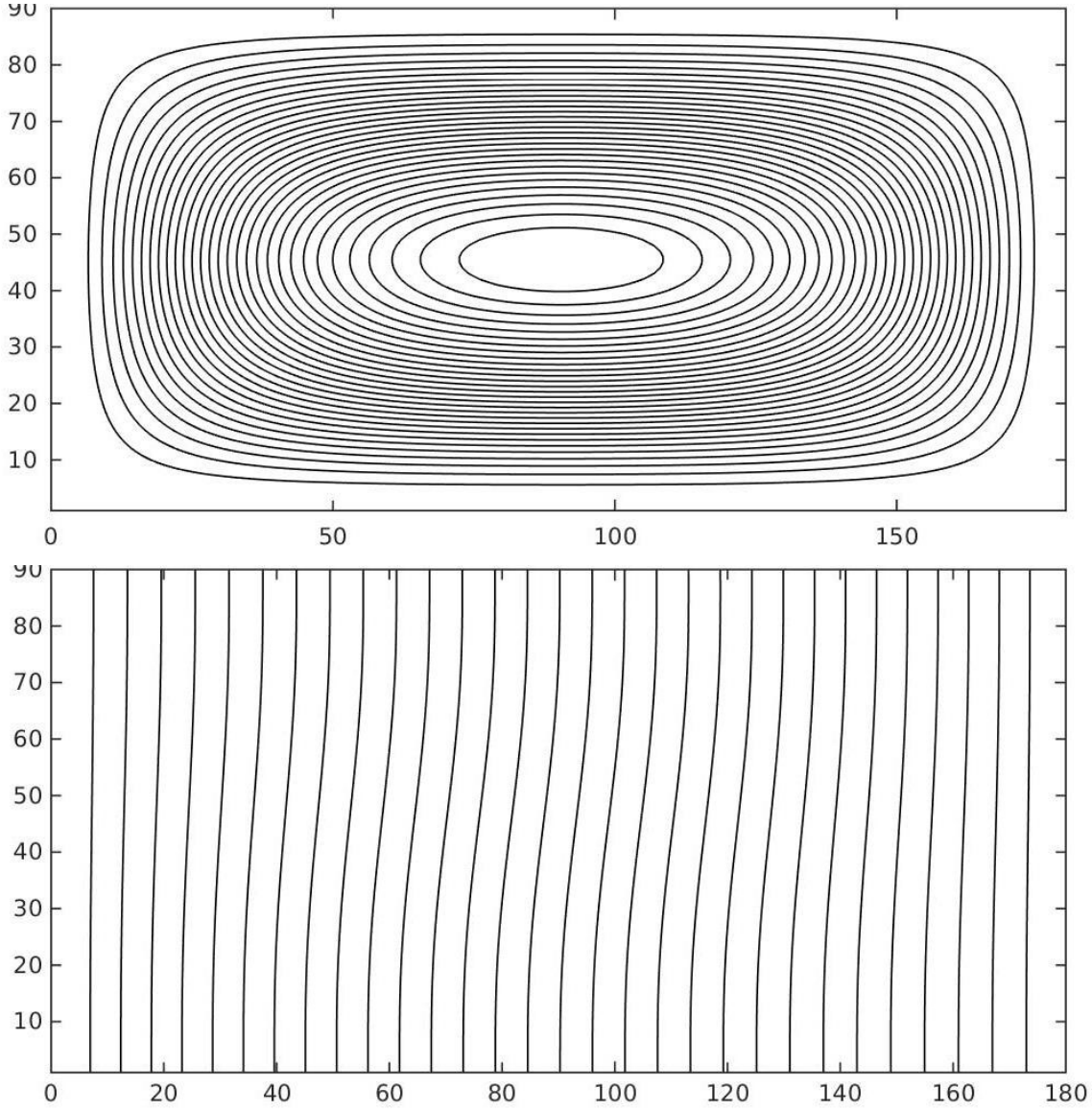

Figura B. 5 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 0,5; Ra=10^3 y esquema numérico LEY DE POTENCIA.

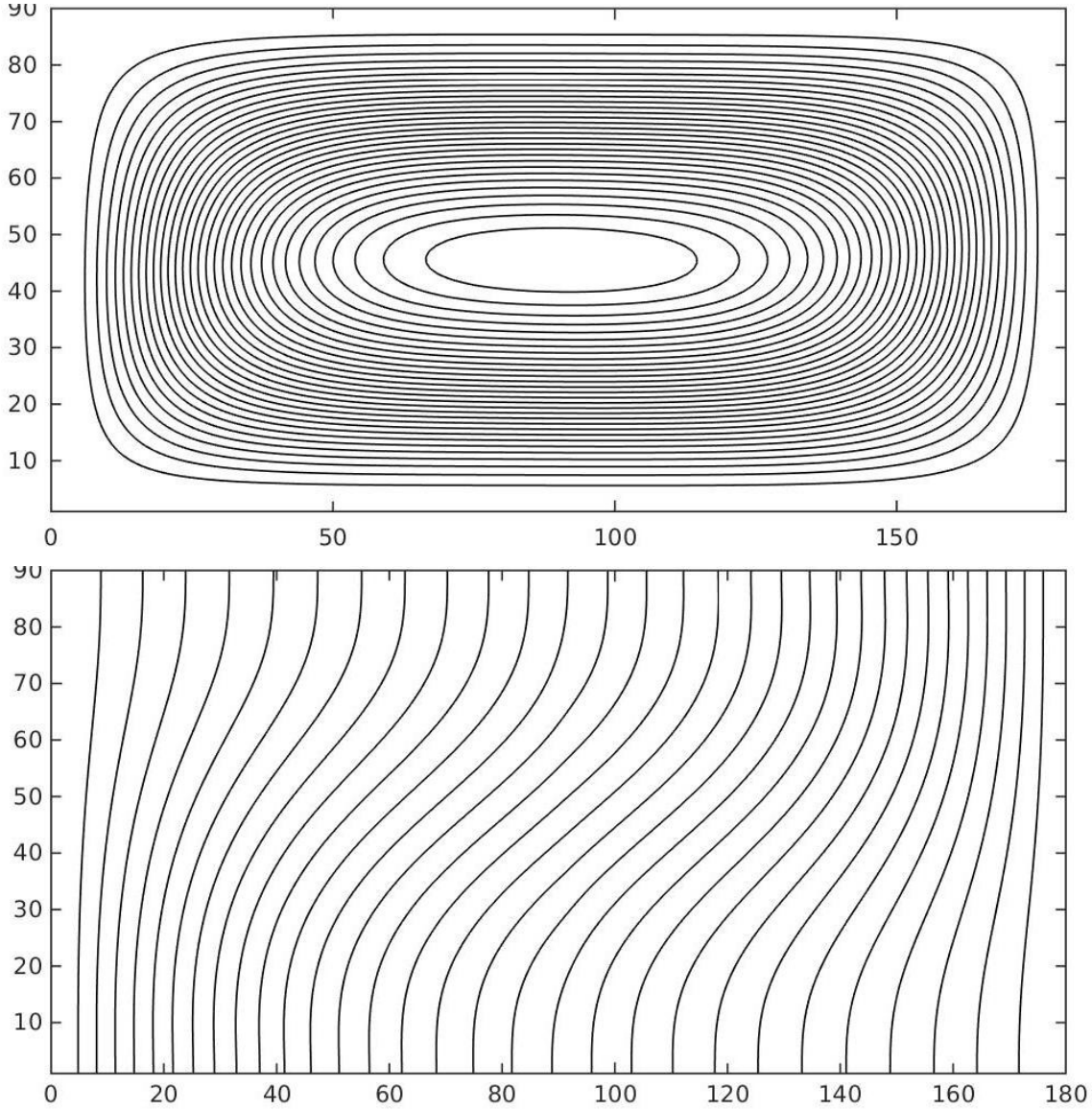

Figura B. 6 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 0,5; Ra=10^4 y esquema numérico LEY DE POTENCIA.

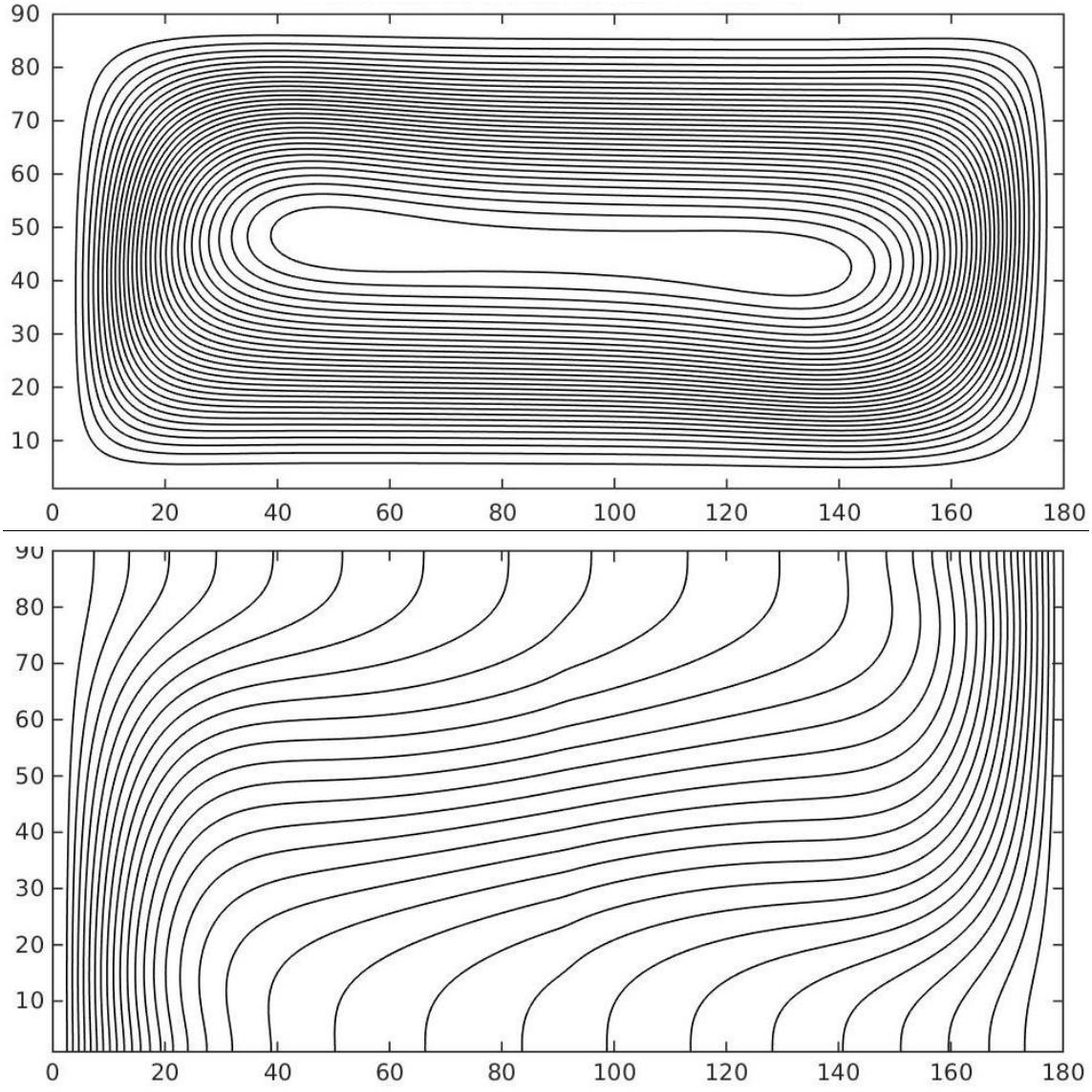

Figura B. 7 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 0,5; Ra=10^5 y esquema numérico LEY DE POTENCIA.

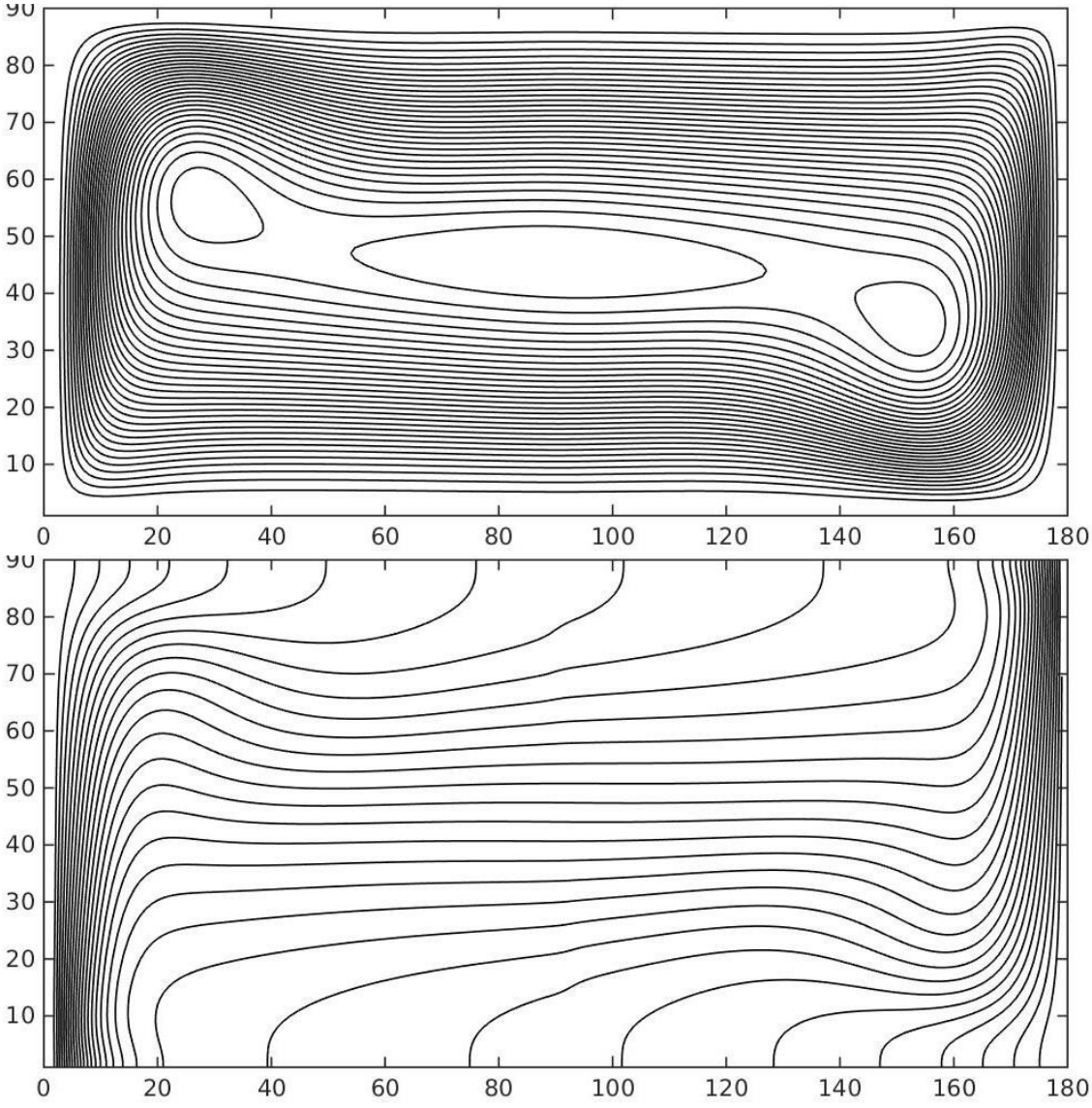

Figura B. 8 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 0,5; Ra=10^6 y esquema numérico LEY DE POTENCIA.

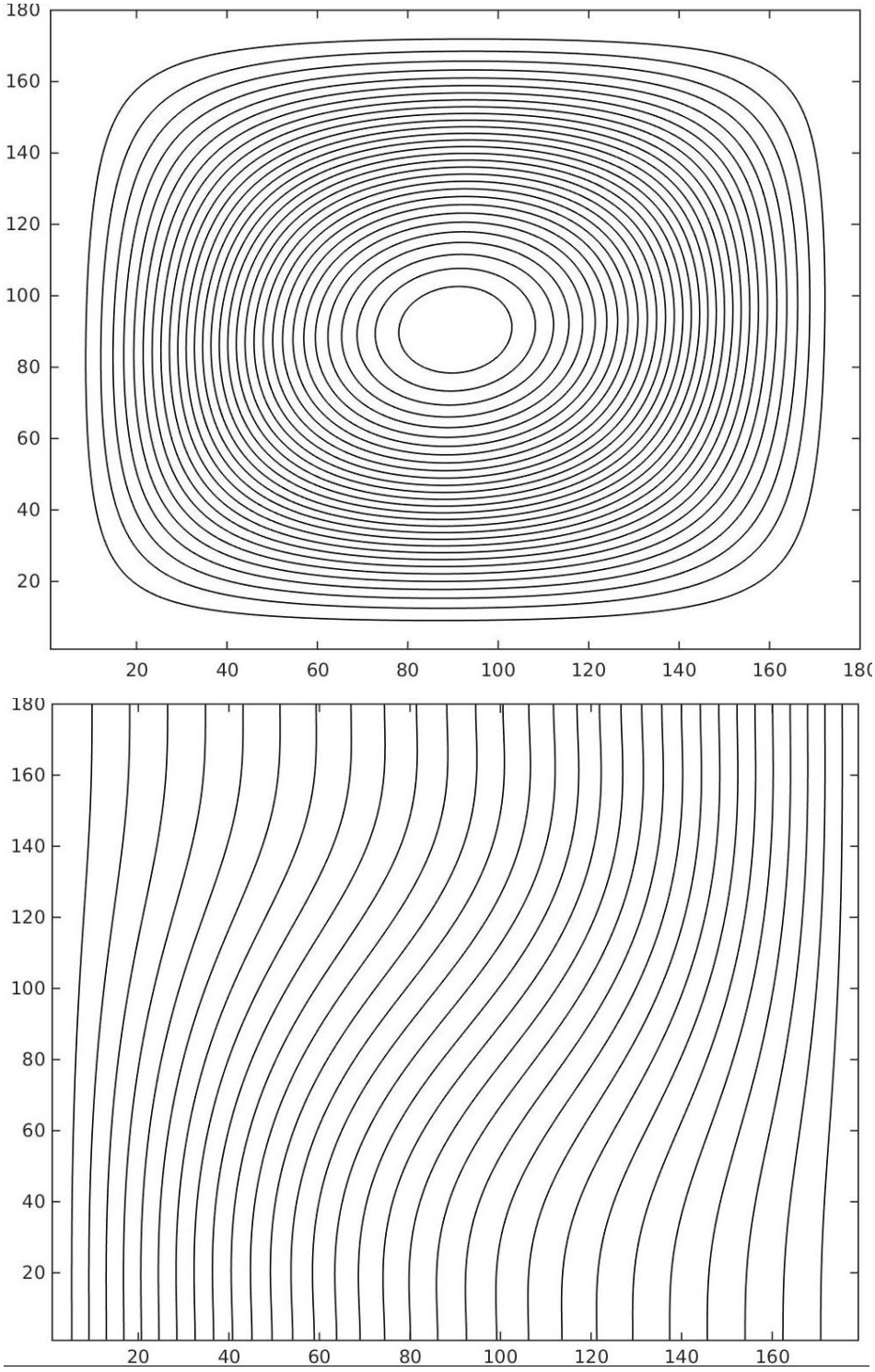

Figura B. 9 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 1; Ra=10^3 y esquema numérico LEY DE POTENCIA.

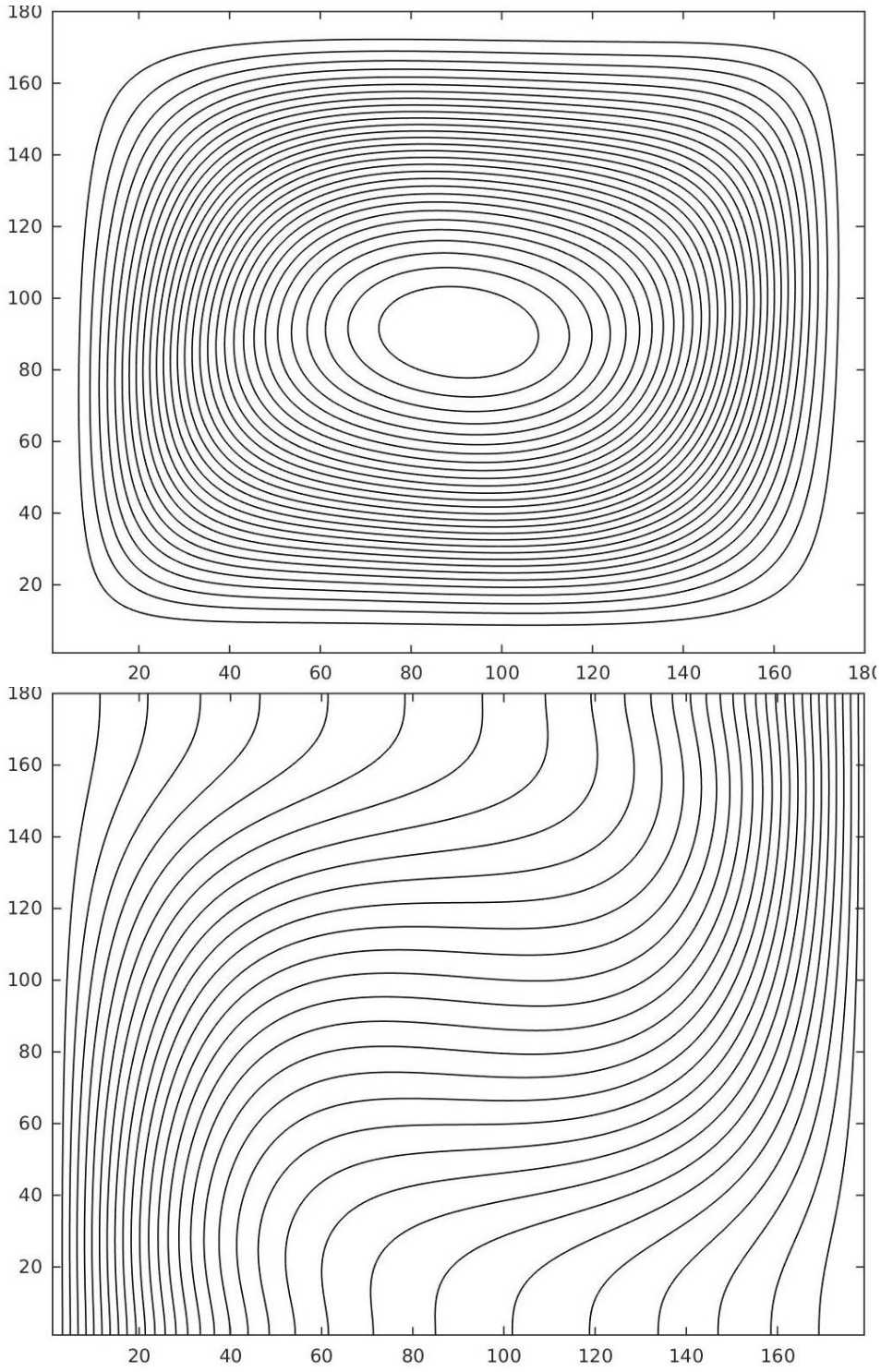

Figura B. 10 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 1; Ra=10^4 y esquema numérico LEY DE POTENCIA.

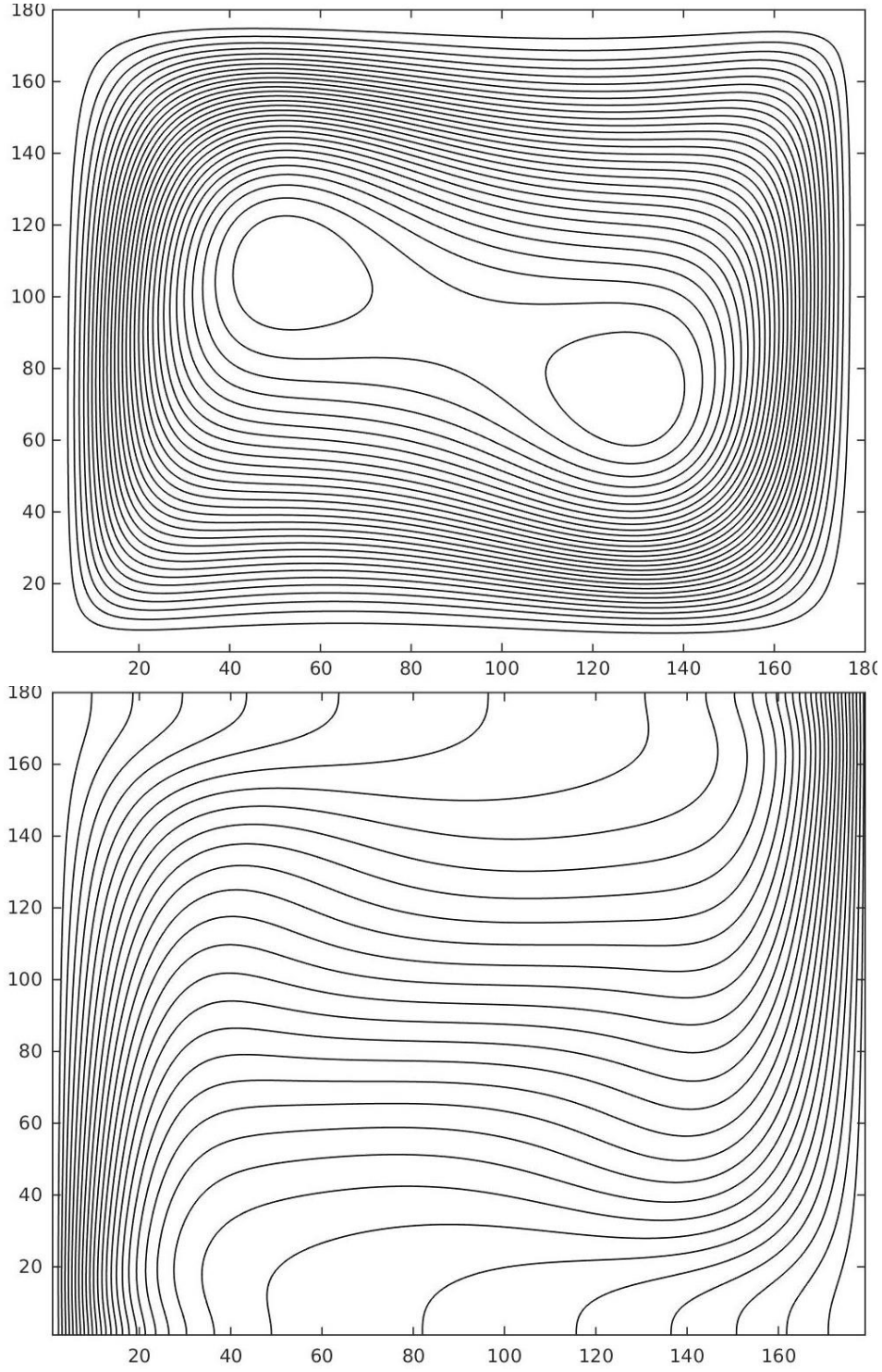

Figura B. 11 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 1; Ra=10^5 y esquema numérico LEY DE POTENCIA.

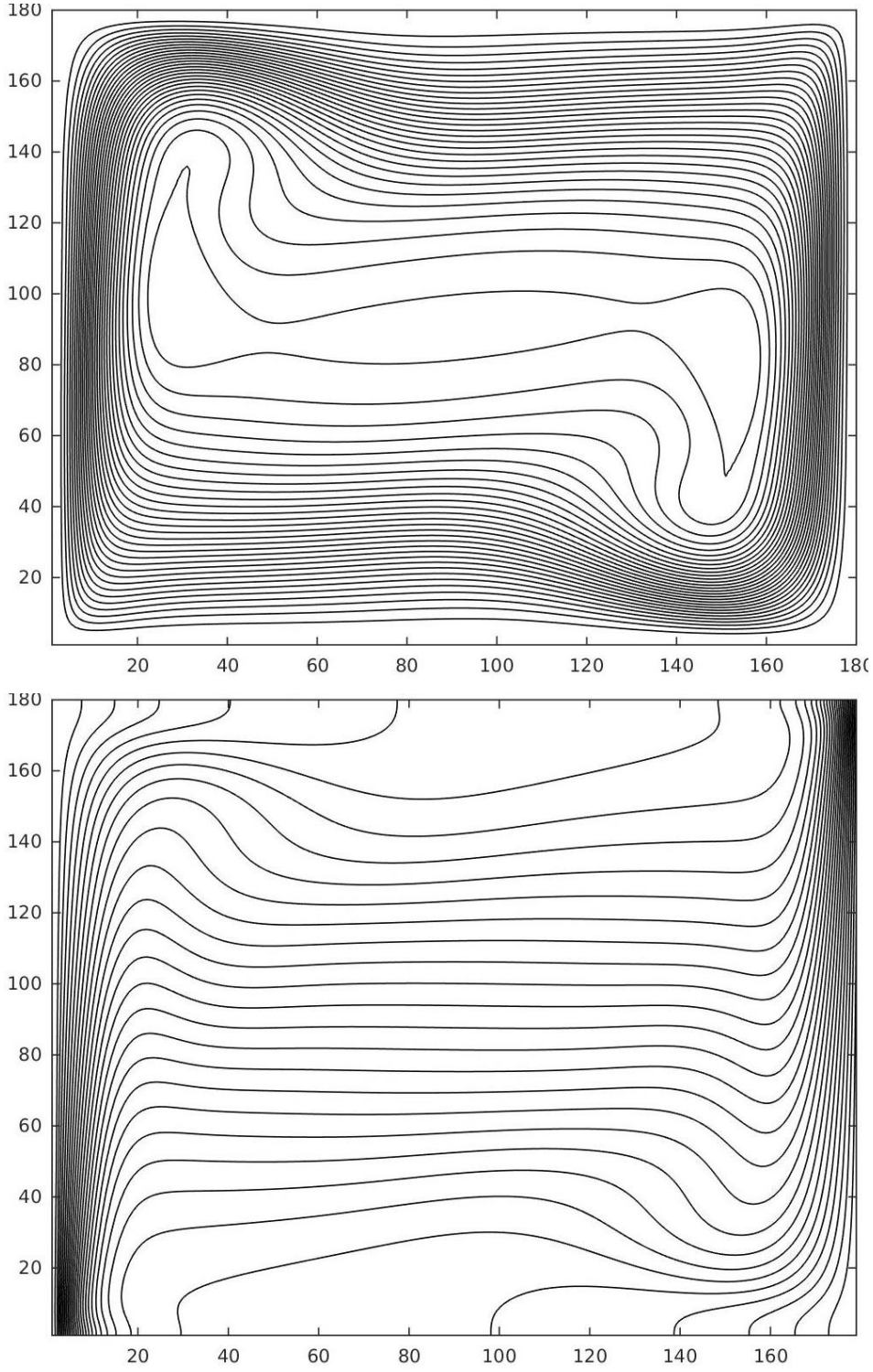

Figura B. 12 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 1; Ra=10^6 y esquema numérico LEY DE POTENCIA.

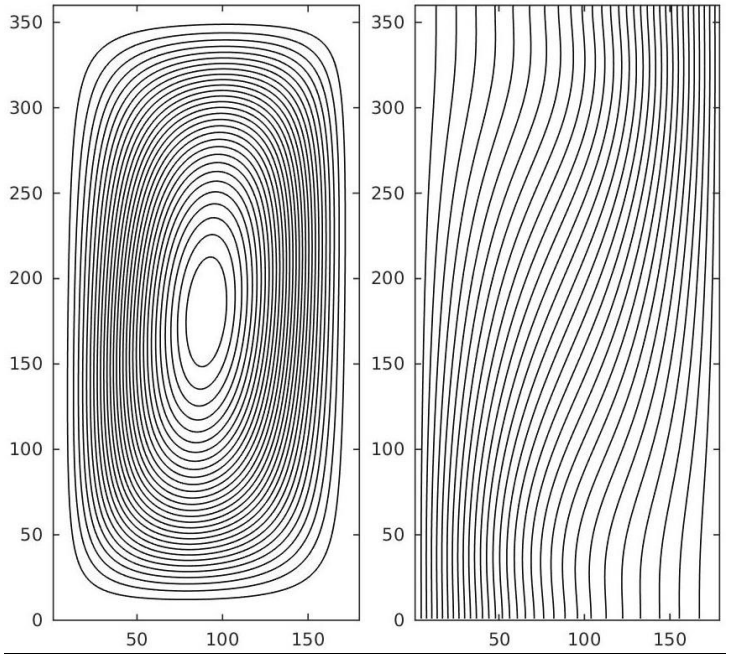

Figura B. 13 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 2; Ra=10^3 y esquema numérico LEY DE POTENCIA.

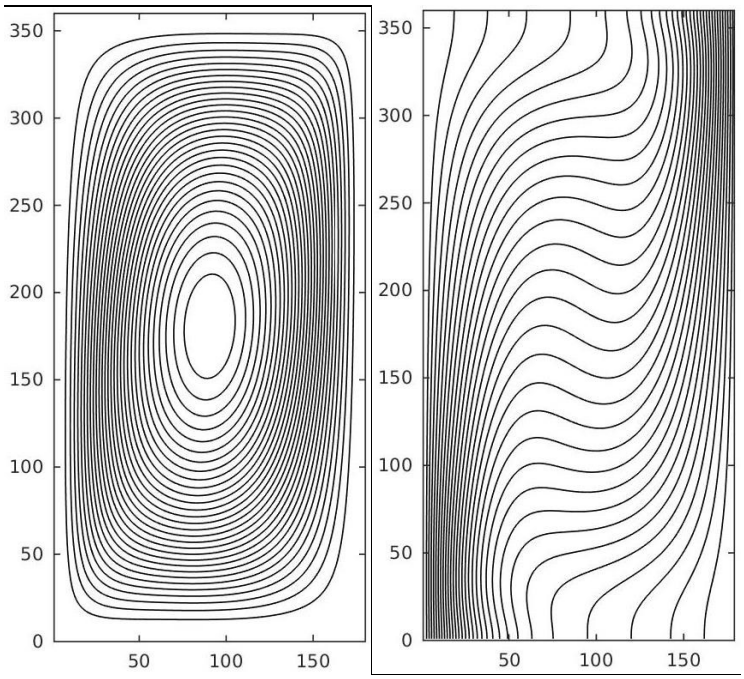

Figura B. 14 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 2; Ra=10^4 y esquema numérico LEY DE POTENCIA.

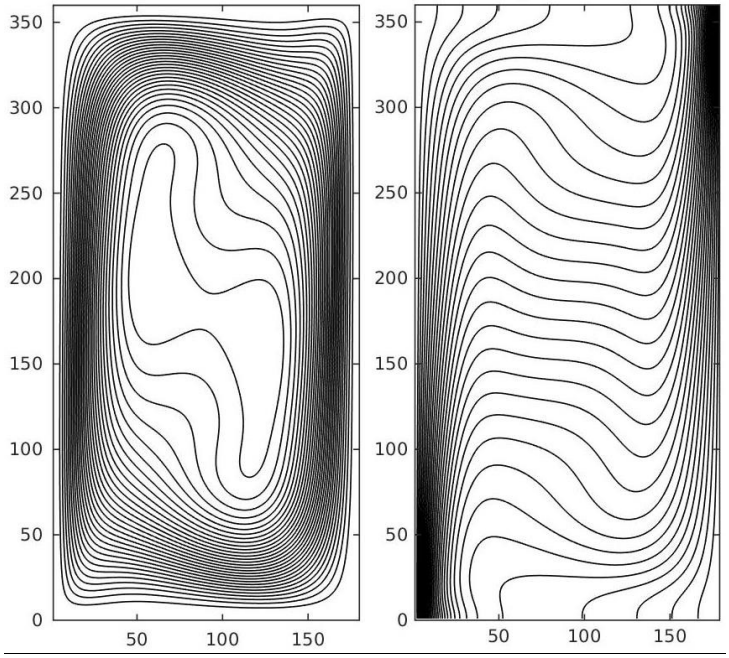

Figura B. 15 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 2; Ra=10^5 y esquema numérico LEY DE POTENCIA.

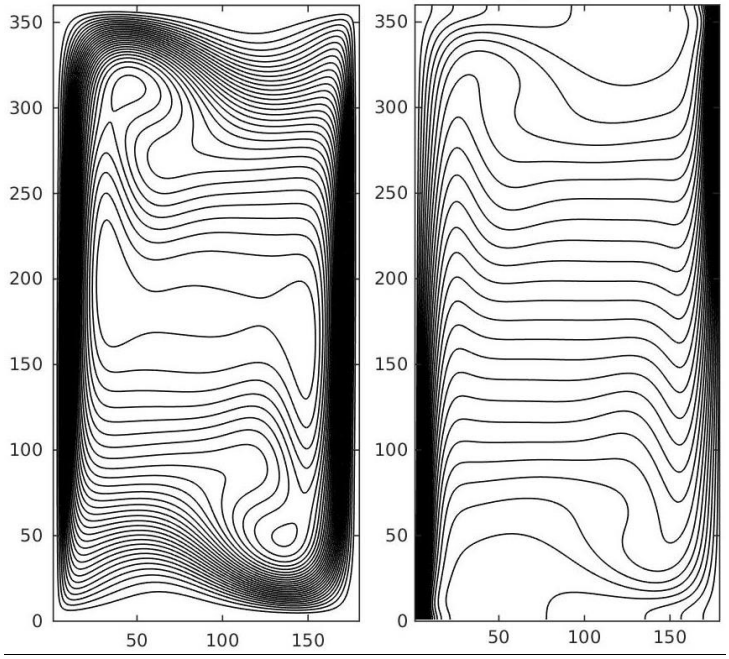

Figura B. 16 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 2; Ra=10^6 y esquema numérico LEY DE POTENCIA.

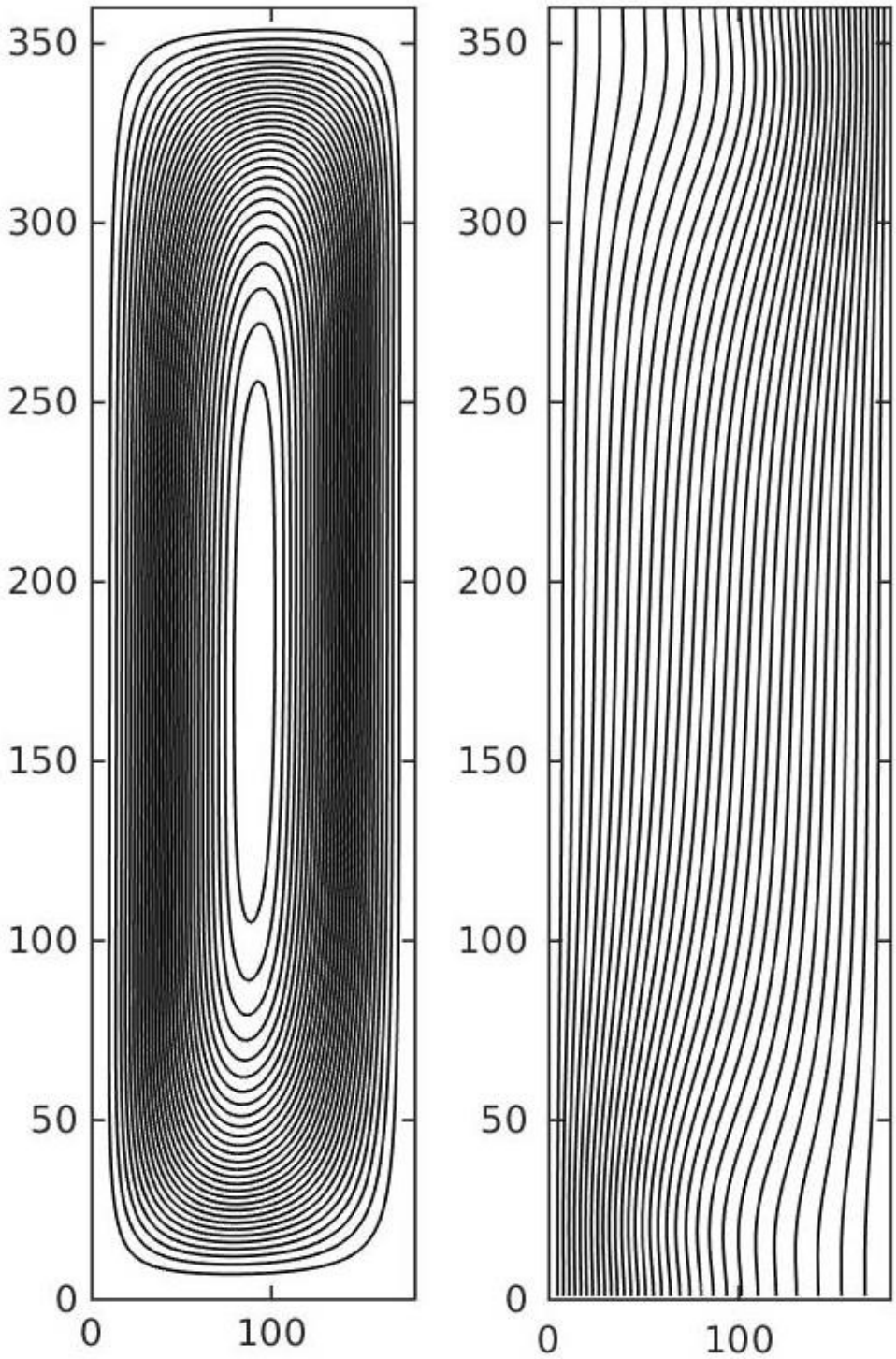

Figura B. 17 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 4; Ra=10^3 y esquema numérico LEY DE POTENCIA

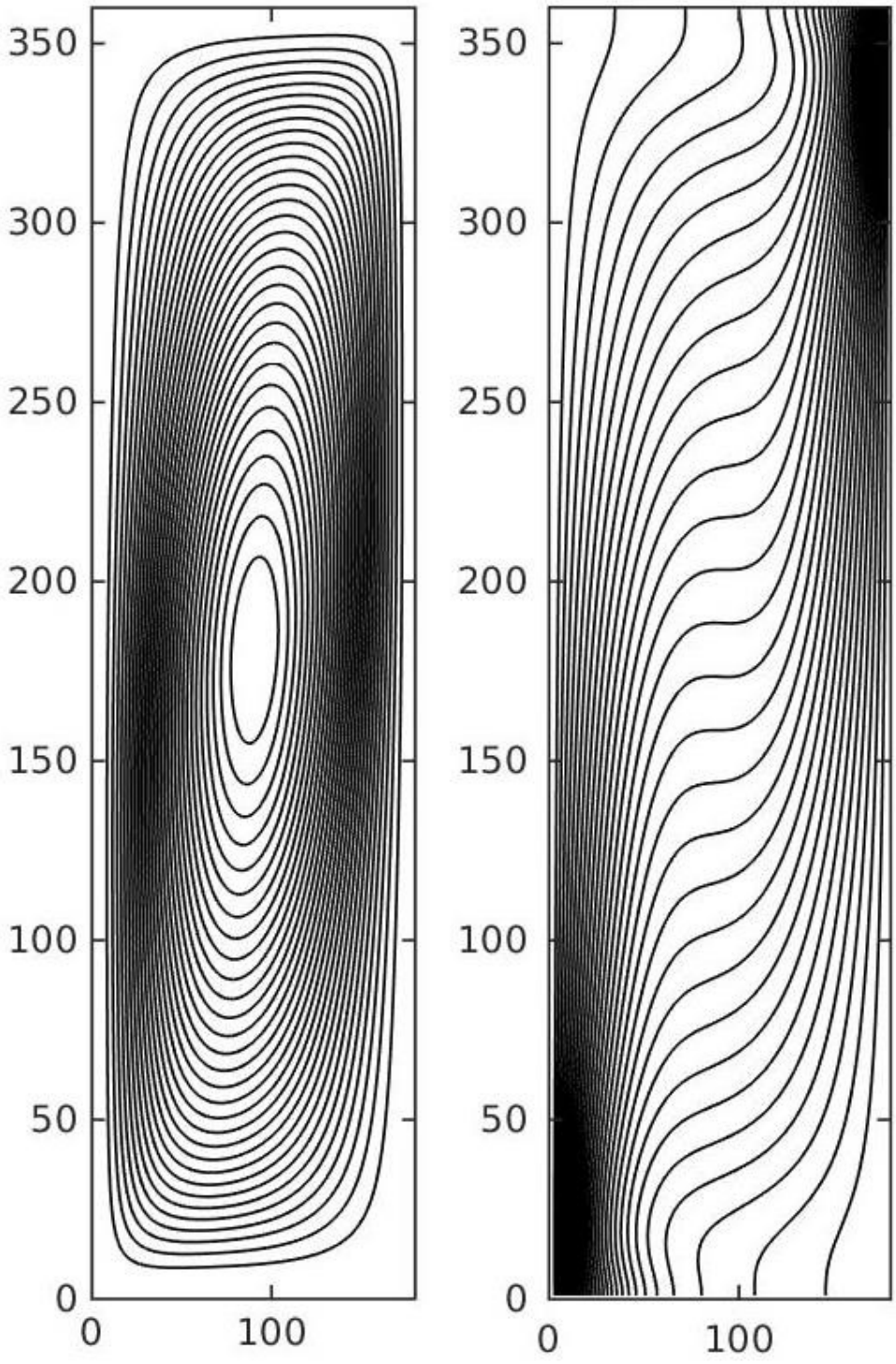

Figura B. 18 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 4; Ra=10^4 y esquema numérico LEY DE POTENCIA.

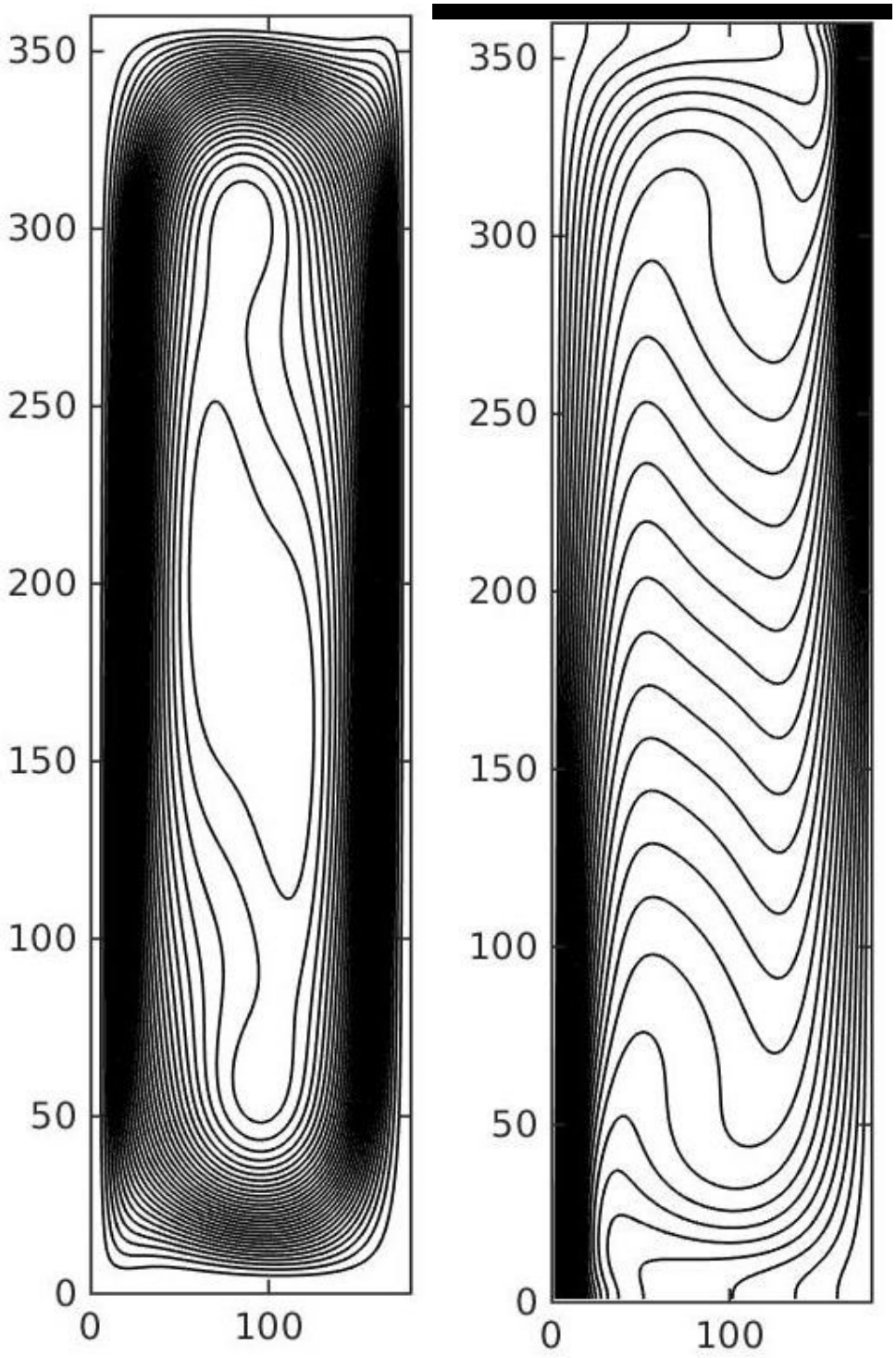

Figura B. 19 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 4; Ra=10^5 y esquema numérico LEY DE POTENCIA.

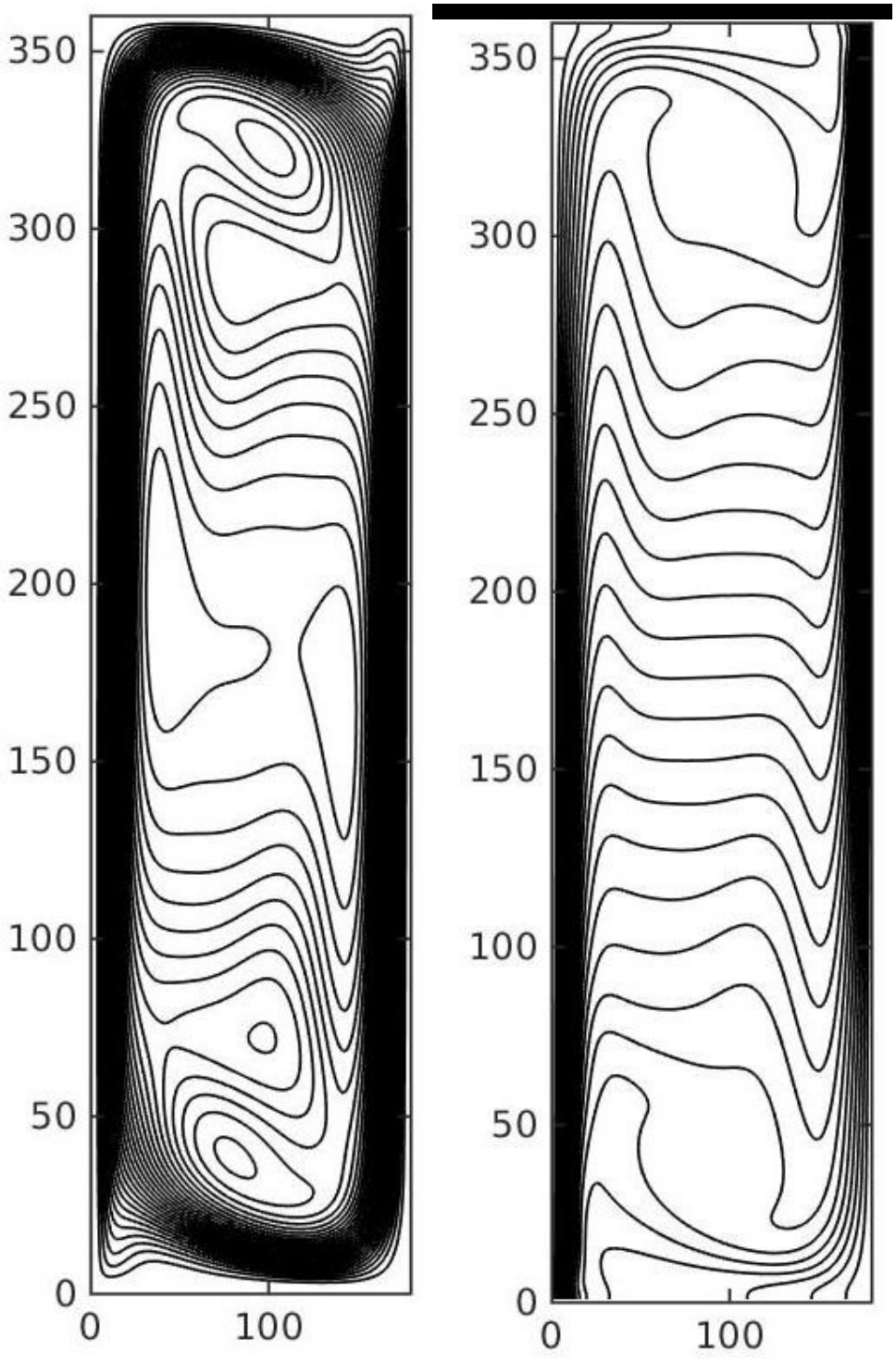

Figura B. 20 Figuras de líneas de corriente y campo de temperatura para a relación de aspecto 4; Ra=106 y esquema numérico LEY DE POTENCIA.

ANEXO C – CODIGO ABIERTO

El código consta de nueve librerías encargadas de almacenar las distintas funciones a utilizarse en la función principal.

C.1 Biblioteca “base.h”

En esta biblioteca se almacenan las variables globales de las matrices, los coeficientes de cada problema y se fine la función de la malla estructurada.

base.h

```
#include<iostream>
#include<math.h>
#include<stdlib.h>
#include<fstream>
using namespace std;
void Grid (void);


//Declaraciones globales
#define Nx 180
#define Ny 180
#define H 0.0040806985 //0.0408069855 //0.0189409248 //0.0087915985 //0.0040806985
//malla
long double X[Nx+1], DXP[Nx+1], DXU[Nx], Hx=H;
long double Y[Ny+1], DYP[Ny+1], DYU[Ny], Hy=H;
//mallas desplazadas
int Nxd=Nx-1, Nyd=Ny-1;
long double Xd[Nx], DXPd[Nx], DXUd[Nx-1];
long double Yd[Ny], DYPd[Ny], DYUd[Ny-1];

//moentum
long double U[Nx+1][Ny+1], V[Nx+1][Ny+1], Uold[Nx+1][Ny+1], Vold[Nx+1][Ny+1];
long double Ug[Nx+1][Ny+1], Vg[Nx+1][Ny+1], Uc[Nx+1][Ny+1], Vc[Nx+1][Ny+1]; //Uc y Vc son velocidades corregidas
long double P[Nx+1][Ny+1], Pc[Nx+1][Ny+1], Pcold[Nx+1][Ny+1], Pg[Nx+1][Ny+1]; //Pc es velovidad de correccion y Pg es presion supuesta
long double u=0, v=0, Rdifu=1.846E-5;

//transitorio
long double t=0, tmax=0, dt=0.0001;
```

```
//temperatura
long double Tnew[Nx+1][Ny+1], Told[Nx+1][Ny+1];
long double Tw=25, Te=15, Tn=15, Ts=15, Tinfinito=(Te+Tw)/2;
long double lambda=0.026, Cp=1000, Beta=2/(Te+Tw); // valores de conduccion

long double RdifT, ro=1.200, Ra=(9.81*Beta*(Tw-Te)*pow(H,3))/(pow(lambda,2)*0.71); //ro de difucion
vendria a ser miu de viscosidad


//Pr=0.71 y Ra=1000



void Grid (void)
{
int i, j;
double a;
DXP[1]=0;
DXPd[1]=0;
DXP[Nx]=0;
DXPd[Nxd]=0;
X[1]=0;
Xd[1]=0;
X[Nx]=Hx;
Xd[Nxd]=Hx;

DYP[1]=0;
DYPd[1]=0;
DYP[Ny]=0;
DYPd[Nyd]=0;
Y[1]=0;
Yd[1]=0;
Y[Ny]=Hy;
Yd[Nyd]=Hy;

//malla normal
for (i=2; i<Nx; i++)
        {

        //para mmallados regulares
        DXP[i]=Hx/(Nx-2);// aca nada mas tengo que variar para hacer mallado irregular

        //CALCULAR LA POSICION DE LOS NODOS
        X[i]=X[i-1]+DXP[i]/2+DXP[i-1]/2;
        }
for (i=1; i<Nx; i++)
        {
        DXU[i]=X[i+1]-X[i];
        }

for (i=2; i<Ny; i++)
        {
        DYP[i]=Hy/(Ny-2);;
        Y[i]=Y[i-1]+DYP[i]/2+DYP[i-1]/2;
        }
for (i=1; i<Ny; i++)
```

```
        {
        DYU[i]=Y[i+1]-Y[i];
        }

//malla desplazada
for (i=2; i<Nxd; i++)
        {
        Xd[i]=X[i]+DXP[i]/2;
        }
for (i=2; i<Nxd; i++)
        {
        DXPd[i]=(Xd[i+1]-Xd[i-1])/2;
        }
for (i=1; i<Nxd; i++)
        {
        DXUd[i]=Xd[i+1]-Xd[i];
        }

for (i=2; i<Nyd; i++)
        {
        Yd[i]=Y[i]+DYP[i]/2;
        }
for (i=2; i<Nyd; i++)
        {
        DYPd[i]=(Yd[i+1]-Yd[i-1])/2;
        }
for (i=1; i<Nyd; i++)
        {
        DYUd[i]=Yd[i+1]-Yd[i];
        }

}
```

C.2 Bibliotecas “file.h” y “print”

La biblioteca “file.h” guarda las funciones que gestionan los datos de entrada y los resultados de salido interactuando con archivos de extensión .DAT que contienen los datos tabulados, también gestiona el cálculo de los números de Nusselt máximo y mínimo. Mientras la biblioteca “print.h” permite gestión de los resultados en pantalla en tiempo real.

file.h

```
void saveresults (void)
{
        ofstream results ("solution.dat");
        results<<scientific<<showpos;
        results.precision(64);
```

```
	int i, j;
	for (j=Ny;j>=1;j--)
	{
		for (i=1; i<=Nxd; i++)
		{
			results<<Uc[i][j]<<"\t";
		}
		results<<"\n";
	}
	results<<"\n\n\n";

	for (j=Nyd;j>=1;j--)
	{
		for (i=1; i<=Nx; i++)
		{
			results<<Vc[i][j]<<"\t";
		}
		results<<"\n";
	}

	results<<"\n\n\n";

	for (j=Ny;j>=1;j--)
	{
		for (i=1; i<=Nx; i++)
		{
			results<<P[i][j]<<"\t";
		}
		results<<"\n";
	}

	results<<"\n\n\n";

	for (j=Ny;j>=1;j--)
	{
		for (i=1; i<=Nx; i++)
		{
			results<<Tnew[i][j]<<"\t";
		}
		results<<"\n";
	}

	results.close();
}

bool readguess (void)
{
	ifstream guess ("initial.dat");

	int i, j;
	bool e;
	for (j=Ny;j>=1;j--)
	{
		for (i=1; i<=Nxd; i++)
		{
			guess>>Ug[i][j];
```

```
                U[i][j]=Ug[i][j];
                Uold[i][j]=Ug[i][j];
                Uc[i][j]=Ug[i][j];
        }
    }

    for (j=Nyd;j>=1;j--)
    {
        for (i=1; i<=Nx; i++)
        {
            guess>>Vg[i][j];
            V[i][j]=Vg[i][j];
            Vold[i][j]=Vg[i][j];
            Vc[i][j]=Vg[i][j];
        }
    }

    for (j=Ny;j>=1;j--)
    {
        for (i=1; i<=Nx; i++)
        {
            guess>>P[i][j];
            Pg[i][j]=P[i][j];
        }
    }

    for (j=Ny;j>=1;j--)
    {
        for (i=1; i<=Nx; i++)
        {
            guess>>Told[i][j];
            //Told[i][j]=15;
            Tnew[i][j]=Told[i][j];
        }
    }

    e=guess.fail();

    guess.close();

    return e;
}

long double readvar (void)
{
    long double a;
    ifstream variable ("var.dat");

    variable>>a;

    variable.close();

    return a;

}
```

```
long double readpaso (void)
{
	long double a;
	ifstream variable ("paso.dat");

	variable>>a;

	variable.close();

	return a;

}

void nusselt (void)
{
	ofstream resultsp ("nusselt.dat");
	resultsp<<scientific<<showpos;
	resultsp.precision(4);
	int i, j;
	long double Nu[Ny], Num=0, Numax, Numin;

	for (i=1; i<Ny-1; i++)
	{
		Nu[i]=((Tw-Tnew[2][i+1])*Hy)/((Tw-Te)*(X[2]-X[1]));
		Num+=Nu[i]*DYP[i+1];

		if (i==1 || Nu[i]>Numax)
		{
			Numax=Nu[i];
		}
		if (i==1 || Nu[i]<Numin)
		{
			Numin=Nu[i];
		}
	}

	Num=Num/Hy;

	resultsp<<"Nusselt Medio= "<<Num<<"\nNusselt Maximo= "<<Numax<<"\nNusselt Minimo=
"<<Numin<<endl;

	resultsp<<"\n\n\n";

	resultsp<<"Xvector\t\tYvector\t\tNusseltvector";
	resultsp<<"\n";

	for (j=Ny;j>=1;j--)
	{
		if (j==1 || j==Ny)
		{
			resultsp<<X[j]<<"\t"<<Y[j];
		}
		else
		{
			resultsp<<X[j]<<"\t"<<Y[j]<<"\t"<<Nu[j-1];
		}
```

```
            resultsp<<"\n";
        }
        resultsp<<"\n\n\n";

        resultsp.close();
}
```

print.h

```
void print (char vec, char eje)
{
        int i, j;
        cout.precision(4);
        switch (vec)
        {
                case 'X':
                        cout<<"X=\t|"<<X[1];
                        for (i=2; i<=Nx; i++)
                                {
                                cout<<",\t"<<X[i];
                                }
                                cout<<"\t | \n";

                        cout<<"Xd=\t|"<<Xd[1];
                        for (i=2; i<=Nx; i++)
                                {
                                cout<<",\t"<<Xd[i];
                                }
                                cout<<"\t | \n";

                        cout<<"DXP=\t|"<<DXP[1];
                        for (i=2; i<=Nx; i++)
                                {
                                cout<<",\t"<<DXP[i];
                                }
                                cout<<"\t | \n";

                        cout<<"DXPd=\t|"<<DXPd[1];
                        for (i=2; i<=Nx; i++)
                                {
```

```
            cout<<",\t"<<DXPd[i];
            }
            cout<<"\t | \n";

        cout<<"DXU=\t|"<<DXU[1];
        for (i=2; i<Nx; i++)
            {
            cout<<",\t"<<DXU[i];
            }
            cout<<"\t | \n";

        cout<<"DXUd=\t|"<<DXUd[1];
        for (i=2; i<Nxd; i++)
            {
            cout<<",\t"<<DXUd[i];
            }
            cout<<"\t | \n";
        break;

case 'Y':
        cout<<"Y=\t|"<<Y[1];
        for (i=2; i<=Ny; i++)
            {
            cout<<",\t"<<Y[i];
            }
            cout<<"\t | \n";

        cout<<"Yd=\t|"<<Yd[1];
        for (i=2; i<=Ny; i++)
            {
            cout<<",\t"<<Yd[i];
            }
            cout<<"\t | \n";

        cout<<"DYP=\t|"<<DYP[1];
        for (i=2; i<=Ny; i++)
            {
            cout<<",\t"<<DYP[i];
```

```
        }
        cout<<"\t | \n";

    cout<<"DYU=\t|"<<DYU[1];
    for (i=2; i<Ny; i++)
        {
        cout<<",\t"<<DYU[i];
        }
        cout<<"\t | \n";

    cout<<"DYUd=\t|"<<DYUd[1];
    for (i=2; i<Nyd; i++)
        {
        cout<<",\t"<<DYUd[i];
        }
        cout<<"\t | \n";
    break;

case 'C':

    switch (eje)
    {
        /*case '0'
            for (j=Ny; j>=1; j--)
                {
                if(j==int(Ny/2))
                    {
                    cout<<"aW=\t|"<<aW[1][j];
                    }
                else
                    {
                    cout<<"\t|"<<aW[1][j];
                    }
                for (i=2; i<=Nx; i++)
                    {
                    cout<<",\t"<<aW[i][j];
                    }
                cout<<"\t | \n";
```

```
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"aE=\t|"<<aE[1][j];
                }
        else
                {
                cout<<"\t|"<<aE[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<aE[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"aS=\t|"<<aS[1][j];
                }
        else
                {
                cout<<"\t|"<<aS[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<aS[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";
```

```
for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"aN=\t|"<<aN[1][j];
                }
        else
                {
                cout<<"\t|"<<aN[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<aN[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";


for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"ap=\t|"<<ap[1][j];
                }
        else
                {
                cout<<"\t|"<<ap[1][j];
                }

        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<ap[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";
```

```
        for (j=Ny; j>=1; j--)
                {
                if(j==int(Ny/2))
                        {
                        cout<<"b=\t|"<<b[1][j];
                        }
                else
                        {
                        cout<<"\t|"<<b[1][j];
                        }

                for (i=2; i<=Nx; i++)
                        {
                        cout<<",\t"<<b[i][j];
                        }
                cout<<"\t | \n";
                }
        cout<<"\n\n";
        break;*/

case 'X':

        for (j=Ny; j>=1; j--)
                {
                if(j==int(Ny/2))
                        {
                        cout<<"aWx=\t|"<<aWx[1][j];
                        }
                else
                        {
                        cout<<"\t|"<<aWx[1][j];
                        }
                for (i=2; i<=Nxd; i++)
                        {
                        cout<<",\t"<<aWx[i][j];
                        }
                cout<<"\t | \n";
```

```
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"aEx=\t|"<<aEx[1][j];
                }
        else
                {
                cout<<"\t|"<<aEx[1][j];
                }
        for (i=2; i<=Nxd; i++)
                {
                cout<<",\t"<<aEx[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"aSx=\t|"<<aSx[1][j];
                }
        else
                {
                cout<<"\t|"<<aSx[1][j];
                }
        for (i=2; i<=Nxd; i++)
                {
                cout<<",\t"<<aSx[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";
```

```
for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"aNx=\t|"<<aNx[1][j];
                }
        else
                {
                cout<<"\t|"<<aNx[1][j];
                }
        for (i=2; i<=Nxd; i++)
                {
                cout<<",\t"<<aNx[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";


for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"apx=\t|"<<apx[1][j];
                }
        else
                {
                cout<<"\t|"<<apx[1][j];
                }

        for (i=2; i<=Nxd; i++)
                {
                cout<<",\t"<<apx[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";
```

```
        for (j=Ny; j>=1; j--)
                {
                        if(j==int(Ny/2))
                        {
                        cout<<"bx=\t|"<<bx[1][j];
                        }
                else
                        {
                        cout<<"\t|"<<bx[1][j];
                        }

                for (i=2; i<=Nxd; i++)
                        {
                        cout<<",\t"<<bx[i][j];
                        }
                cout<<"\t | \n";
                }
        cout<<"\n\n";
        break;

case 'Y':

        for (j=Nyd; j>=1; j--)
                {
                if(j==int(Nyd/2))
                        {
                        cout<<"aWy=\t|"<<aWy[1][j];
                        }
                else
                        {
                        cout<<"\t|"<<aWy[1][j];
                        }
                for (i=2; i<=Nx; i++)
                        {
                        cout<<",\t"<<aWy[i][j];
                        }
                cout<<"\t | \n";
```

```
        }
cout<<"\n\n";

for (j=Nyd; j>=1; j--)
        {
        if(j==int(Nyd/2))
                {
                cout<<"aEy=\t|"<<aEy[1][j];
                }
        else
                {
                cout<<"\t|"<<aEy[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<aEy[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";

for (j=Nyd; j>=1; j--)
        {
        if(j==int(Nyd/2))
                {
                cout<<"aSy=\t|"<<aSy[1][j];
                }
        else
                {
                cout<<"\t|"<<aSy[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<aSy[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";
```

```
for (j=Nyd; j>=1; j--)
        {
        if(j==int(Nyd/2))
                {
                cout<<"aNy=\t|"<<aNy[1][j];
                }
        else
                {
                cout<<"\t|"<<aNy[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<aNy[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";


for (j=Nyd; j>=1; j--)
        {
        if(j==int(Nyd/2))
                {
                cout<<"apy=\t|"<<apy[1][j];
                }
        else
                {
                cout<<"\t|"<<apy[1][j];
                }

        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<apy[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";
```

```
        for (j=Nyd; j>=1; j--)
                {
                        if(j==int(Nyd/2))
                        {
                        cout<<"by=\t|"<<by[1][j];
                        }
                else
                        {
                        cout<<"\t|"<<by[1][j];
                        }

                for (i=2; i<=Nx; i++)
                        {
                        cout<<",\t"<<by[i][j];
                        }
                cout<<"\t | \n";
                }
        cout<<"\n\n";
        break;

case 'P':

        for (j=Ny; j>=1; j--)
                {
                if(j==int(Ny/2))
                        {
                        cout<<"aWP=\t|"<<aWP[1][j];
                        }
                else
                        {
                        cout<<"\t|"<<aWP[1][j];
                        }
                for (i=2; i<=Nx; i++)
                        {
                        cout<<",\t"<<aWP[i][j];
                        }
                cout<<"\t | \n";
```

```
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"aEP=\t|"<<aEP[1][j];
                }
        else
                {
                cout<<"\t|"<<aEP[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<aEP[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"aSP=\t|"<<aSP[1][j];
                }
        else
                {
                cout<<"\t|"<<aSP[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<aSP[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";
```

```
for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"aNP=\t|"<<aNP[1][j];
                }
        else
                {
                cout<<"\t|"<<aNP[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<aNP[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";


for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"apP=\t|"<<apP[1][j];
                }
        else
                {
                cout<<"\t|"<<apP[1][j];
                }

        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<apP[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";
```

```
                for (j=Ny; j>=1; j--)
                        {
                                if(j==int(Ny/2))
                                {
                                cout<<"bP=\t|"<<bP[1][j];
                                }
                        else
                                {
                                cout<<"\t|"<<bP[1][j];
                                }

                        for (i=2; i<=Nx; i++)
                                {
                                cout<<",\t"<<bP[i][j];
                                }
                        cout<<"\t | \n";
                        }
                cout<<"\n\n";
                break;

        }
        break;

case 'T':
        for (j=Ny;j>=1;j--)
        {
        if (j==int(Ny/2))
                {
                cout<<"T=\t|"<<Tnew[1][j];
                }
        else
                {
                cout<<"\t|"<<Tnew[1][j];
                }

        for (i=2; i<=Nx; i++)
                {
```

```
                cout<<",\t"<<Tnew[i][j];
                }
        cout<<"\t | \n";
        }
        cout<<"\n";
        break;

case 'P':
switch (eje)
{
        case '0':
                for (j=Ny;j>=1;j--)
                {
                if (j==int(Ny/2))
                        {
                        cout<<"P=\t|"<<P[1][j];
                        }
                else
                        {
                        cout<<"\t|"<<P[1][j];
                        }

                for (i=2; i<=Nx; i++)
                        {
                        cout<<",\t"<<P[i][j];
                        }
                cout<<"\t | \n";
                }
                cout<<"\n";
                break;

        case 'C':
                for (j=Ny;j>=1;j--)
                {
                if (j==int(Ny/2))
                        {
                        cout<<"Pc=\t|"<<Pc[1][j];
                        }
```

```
                else
                        {
                        cout<<"\t|"<<Pc[1][j];
                        }

                for (i=2; i<=Nx; i++)
                        {
                        cout<<",\t"<<Pc[i][j];
                        }
                cout<<"\t | \n";
                }
                cout<<"\n";
                break;
}
break;

case 'V':
switch (eje)
{
        case '0':
                for (j=Nyd;j>=1;j--)
                {
                if (j==int(Ny/2))
                        {
                        cout<<"V=\t|"<<V[1][j];
                        }
                else
                        {
                        cout<<"\t|"<<V[1][j];
                        }

                for (i=2; i<=Nx; i++)
                        {
                        cout<<",\t"<<V[i][j];
                        }
                cout<<"\t | \n";
                }
                cout<<"\n";
```

```
                break;

        case 'C':
                for (j=Nyd;j>=1;j--)
                {
                if (j==int(Ny/2))
                        {
                        cout<<"Vc=\t|"<<Vc[1][j];
                        }
                else
                        {
                        cout<<"\t|"<<Vc[1][j];
                        }

                for (i=2; i<=Nx; i++)
                        {
                        cout<<",\t"<<Vc[i][j];
                        }
                cout<<"\t | \n";
                }
                cout<<"\n";
                break;
}
        break;

case 'U':
switch (eje)
{
        case '0':
                for (j=Ny;j>=1;j--)
                {
                if (j==int(Ny/2))
                        {
                        cout<<"U=\t|"<<U[1][j];
                        }
                else
                        {
                        cout<<"\t|"<<U[1][j];
```

```
                }

        for (i=2; i<=Nxd; i++)
                {
                cout<<",\t"<<U[i][j];
                }
        cout<<"\t | \n";
        }
        cout<<"\n";
        break;

    case 'C':
        for (j=Ny;j>=1;j--)
        {
        if (j==int(Ny/2))
                {
                cout<<"Uc=\t|"<<Uc[1][j];
                }
        else
                {
                cout<<"\t|"<<Uc[1][j];
                }

        for (i=2; i<=Nxd; i++)
                {
                cout<<",\t"<<Uc[i][j];
                }
        cout<<"\t | \n";
        }
        cout<<"\n";
        break;
}
    break;

/*case 'D':

    for (j=Ny; j>=1; j--)
        {
```

```
        if(j==int(Ny/2))
                {
                cout<<"Dw=\t|"<<Dw[1][j];
                }
        else
                {
                cout<<"\t|"<<Dw[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<Dw[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"De=\t|"<<De[1][j];
                }
        else
                {
                cout<<"\t|"<<De[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<De[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
```

```
                cout<<"Ds=\t|"<<Ds[1][j];
                }
            else
                {
                cout<<"\t|"<<Ds[1][j];
                }
            for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<Ds[i][j];
                }
            cout<<"\t | \n";
            }
        cout<<"\n\n";

        for (j=Ny; j>=1; j--)
            {
            if(j==int(Ny/2))
                {
                cout<<"Dn=\t|"<<Dn[1][j];
                }
            else
                {
                cout<<"\t|"<<Dn[1][j];
                }
            for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<Dn[i][j];
                }
            cout<<"\t | \n";
            }
        cout<<"\n\n";
        break;

case 'F':

        for (j=Ny; j>=1; j--)
            {
            if(j==int(Ny/2))
```

```
                {
                cout<<"Fw=\t|"<<Fw[1][j];
                }
        else
                {
                cout<<"\t|"<<Fw[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<Fw[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"Fe=\t|"<<Fe[1][j];
                }
        else
                {
                cout<<"\t|"<<Fe[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<Fe[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"Fs=\t|"<<Fs[1][j];
```

```
                }
        else
                {
                cout<<"\t|"<<Fs[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<Fs[i][j];
                }
        cout<<"\t | \n";
        }
    cout<<"\n\n";

    for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"Fn=\t|"<<Fn[1][j];
                }
        else
                {
                cout<<"\t|"<<Fn[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<Fn[i][j];
                }
        cout<<"\t | \n";
        }
    cout<<"\n\n";
    break;

case 'P':

    for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
```

```
                cout<<"Pew=\t|"<<Pew[1][j];
                }
        else
                {
                cout<<"\t|"<<Pew[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<Pew[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"Pee=\t|"<<Pee[1][j];
                }
        else
                {
                cout<<"\t|"<<Pee[1][j];
                }
        for (i=2; i<=Nx; i++)
                {
                cout<<",\t"<<Pee[i][j];
                }
        cout<<"\t | \n";
        }
cout<<"\n\n";

for (j=Ny; j>=1; j--)
        {
        if(j==int(Ny/2))
                {
                cout<<"Pes=\t|"<<Pes[1][j];
                }
```

```
                else
                        {
                        cout<<"\t|"<<Pes[1][j];
                        }
                for (i=2; i<=Nx; i++)
                        {
                        cout<<",\t"<<Pes[i][j];
                        }
                cout<<"\t | \n";
                }
        cout<<"\n\n";

        for (j=Ny; j>=1; j--)
                {
                if(j==int(Ny/2))
                        {
                        cout<<"Pen=\t|"<<Pen[1][j];
                        }
                else
                        {
                        cout<<"\t|"<<Pen[1][j];
                        }
                for (i=2; i<=Nx; i++)
                        {
                        cout<<",\t"<<Pen[i][j];
                        }
                cout<<"\t | \n";
                }
        cout<<"\n\n";
        break;*/
    }
}
```

C.3 Bibliotecas “funcionesMX.h”, “funcionesMY.h”, “funcionesP.h” y “funcionesT.h”

Estas bibliotecas definen las funciones que realizan la solución iterativa para sus respectivos campos, velocidad en X y en Y, presión y temperatura.

funcionesMX.h

```
//#include "base.h"

void coef_MX (void);
void solve_MX (void);
bool paradaX (void);
void remplaceU (void);
double A (double Pe);
double max (double a, double b);
void updateU (void);


long double aEx[Nx+1][Ny+1], aWx[Nx+1][Ny+1], aNx[Nx+1][Ny+1], aSx[Nx+1][Ny+1], apx[Nx+1][Ny+1], bx[Nx+1][Ny+1];

long double Fex[Nx+1][Ny+1], Fwx[Nx+1][Ny+1], Fnx[Nx+1][Ny+1], Fsx[Nx+1][Ny+1]; //Fuerza convectiva

long double Dex[Nx+1][Ny+1], Dwx[Nx+1][Ny+1], Dnx[Nx+1][Ny+1], Dsx[Nx+1][Ny+1]; //Fuerza difusiva (en temperatura conduccion)

long double Peex[Nx+1][Ny+1], Pewx[Nx+1][Ny+1], Penx[Nx+1][Ny+1], Pesx[Nx+1][Ny+1], Spx=0, Scx=0; // vectores de conveccion

void coef_MX (void)
{
int i,j;

//para frontera oeste
#pragma omp parallel for
for (j=1; j<=Ny; j++)
        {
        aEx[1][j]=0;
```

```
        aWx[1][j]=0;
        aSx[1][j]=0;
        aNx[1][j]=0;
        apx[1][j]=1;
        bx[1][j]=0;
        }

//para frontera este
#pragma omp parallel for
for (j=1; j<=Ny; j++)
        {
        aEx[Nxd][j]=0;
        aWx[Nxd][j]=0;
        aSx[Nxd][j]=0;
        aNx[Nxd][j]=0;
        apx[Nxd][j]=1;
        bx[Nxd][j]=0;
        }

//para frontera sur
#pragma omp parallel for
for (i=1; i<=Nxd; i++)
        {
        aEx[i][1]=0;
        aWx[i][1]=0;
        aSx[i][1]=0;
        aNx[i][1]=0;
        apx[i][1]=1;
        bx[i][1]=0;
        }

//para frontera norte
#pragma omp parallel for
for (i=2; i<Nxd; i++)
        {
        aEx[i][Ny]=0;
        aWx[i][Ny]=0;
        aSx[i][Ny]=0;
```

```
        aNx[i][Ny]=0;
        apx[i][Ny]=1;
        bx[i][Ny]=u;
        }
#pragma omp parallel for
for (j=2; j<Ny; j++)
        {
        for (i=2; i<Nxd; i++)
                {
                Dex[i][j]=Rdifu*DYP[j]/DXUd[i]; //Rdif debe llevar el mismo coef
                Dwx[i][j]=Rdifu*DYP[j]/DXUd[i-1];
                Dnx[i][j]=Rdifu*DXPd[i]/DYU[j];
                Dsx[i][j]=Rdifu*DXPd[i]/DYU[j-1]; //la formula es mas compleja formula (5.12) pero para
Rdif constante queda asi
                }
        }


//#pragma omp parallel for
for (j=2; j<Ny; j++)
        {
        for (i=2; i<Nxd; i++)
                {
                Fex[i][j]=((ro*U[i][j]+ro*U[i+1][j])/2)*DYP[j]; //pedir revisar
                Fwx[i][j]=((ro*U[i-1][j]+ro*U[i][j])/2)*DYP[j];
                Fnx[i][j]=((ro*V[i][j]+ro*V[i+1][j])/2)*DXPd[i];
                Fsx[i][j]=((ro*V[i][j-1]+ro*V[i+1][j-1])/2)*DXPd[i];

                Peex[i][j]=Fex[i][j]/Dex[i][j];
                Pewx[i][j]=Fwx[i][j]/Dwx[i][j];
                Penx[i][j]=Fnx[i][j]/Dnx[i][j];
                Pesx[i][j]=Fsx[i][j]/Dsx[i][j];
                }
        }


//#pragma omp parallel for
for (j=2; j<Ny; j++)
```

```
        {
        for (i=2; i<Nxd; i++)
                {
                aEx[i][j]=Dex[i][j]*A(Peex[i][j])+max(-Fex[i][j],0); //Rdif debe llevar el mismo coef
                aWx[i][j]=Dwx[i][j]*A(Pewx[i][j])+max(Fwx[i][j],0);
                aNx[i][j]=Dnx[i][j]*A(Penx[i][j])+max(-Fnx[i][j],0);
                aSx[i][j]=Dsx[i][j]*A(Pesx[i][j])+max(Fsx[i][j],0);
                apx[i][j]=/*(ro*DXP[i]*DYP[i]/dt)+*/aWx[i][j]+aEx[i][j]+aSx[i][j]+aNx[i][j]-
Spx*DXP[i]*DYP[j]/*+(Fex[i][j]-Fwx[i][j])+(Fnx[i][j]-Fsx[i][j])*/;
                bx[i][j]=/*(ro*DXP[i]*DYP[i]/dt)*Uold[i][j]+*/Scx*DXPd[i]*DYP[j]-DYP[j]*(P[i+1][j]-P[i][j]);
                //cout<<apx[i][j]<<"\n";
                }
        }

}


void solve_MX (void)
{
int i,j;
double aux;
#pragma omp parallel for
for (j=1; j<=Ny; j++)
        {
        for (i=1; i<=Nxd; i++)
                {
                U[i][j]=(aEx[i][j]*Uold[i+1][j]+aWx[i][j]*Uold[i-1][j]+aNx[i][j]*Uold[i][j+1]+aSx[i][j]*Uold[i][j-
1]+bx[i][j])/apx[i][j];
                //aux=(aEx[i][j]*Uold[i+1][j]+aWx[i][j]*Uold[i-1][j]+aNx[i][j]*Uold[i][j+1]+aSx[i][j]*Uold[i][j-
1]+bx[i][j])/apx[i][j];
                //U[i][j]=Uold[i][j]+0.89*(aux-Uold[i][j]);
                //cout<<U[i][j]<<"\n";
                }
        }
}
```

```
bool paradaX (double e)
{
	bool l;
	int i, j;
	double res=0;
	#pragma omp parallel for reduction(+:res)
	for (i=1; i<=Nxd; i++)
	{
		for (j=1; j<=Ny; j++)
		{
			res+=pow((U[i][j]-Uold[i][j]),2);
			//res+=pow((aEx[i][j]*U[i+1][j]+aWx[i][j]*U[i-1][j]+aNx[i][j]*U[i][j+1]+aSx[i][j]*U[i][j-
1]+bx[i][j]-apx[i][j]*U[i][j]),2);
		}
	}
	res=sqrt(res);
	//cout<<res;
	l=(res>=e);
	return l;
}

double A (double Pe)
{
	double x;
	/*centrado x=1-0.5*fabs(Pe);*/
	/*UPWIND, A=1, entonces*/ x=1;
	/*ley de potencia
	x=pow(max(0, (1-(0.1*(fabs(Pe))))),5);*/
	return x;
}

double max (double a, double b)
{
	if (a>=b) {return a;}
	else {return b;}
}
```

```
void remplaceU (void)
{
	int i,j;
	#pragma omp parallel for
	for (j=1; j<=Ny; j++)
		{
		for (i=1; i<=Nxd; i++)
			{
			Uc[i][j]=U[i][j]+(DYP[j]/apx[i][j])*(Pc[i][j]-Pc[i+1][j]);
			}
		}

}

void updateU (void)
{
	int i,j;
	#pragma omp parallel for
	for (j=1; j<=Ny; j++)
		{
		for (i=1; i<=Nxd; i++)
			{
			Uold[i][j]=U[i][j];
			}
		}

}
```

funcionesMY.h

```
v
}. //#include "base.h"

void coef_MY (void);
void solve_MY (void);
```

```
bool paradaY (void);
void remplaceV (void);
void updateV (void);

//double A (double Pe);
//double max (double a, double b);

long double aEy[Nx+1][Ny+1], aWy[Nx+1][Ny+1], aNy[Nx+1][Ny+1], aSy[Nx+1][Ny+1], apy[Nx+1][Ny+1], by[Nx+1][Ny+1];

long double Fey[Nx+1][Ny+1], Fwy[Nx+1][Ny+1], Fny[Nx+1][Ny+1], Fsy[Nx+1][Ny+1]; //Fuerza convectiva

long double Dey[Nx+1][Ny+1], Dwy[Nx+1][Ny+1], Dny[Nx+1][Ny+1], Dsy[Nx+1][Ny+1]; //Fuerza difusiva (en temperatura conduccion)

long double Peey[Nx+1][Ny+1], Pewy[Nx+1][Ny+1], Peny[Nx+1][Ny+1], Pesy[Nx+1][Ny+1], Spy, Scy; // vectores de conveccion

void coef_MY (void)
{
int i,j;

//para Ta frontera oeste
#pragma omp parallel for
for (j=1; j<=Nyd; j++)
        {
        aEy[1][j]=0;
        aWy[1][j]=0;
        aSy[1][j]=0;
        aNy[1][j]=0;
        apy[1][j]=1;
        by[1][j]=0;
        }

//para Tb frontera este
#pragma omp parallel for
for (j=1; j<=Nyd; j++)
```

```
        {
        aEy[Nx][j]=0;
        aWy[Nx][j]=0;
        aSy[Nx][j]=0;
        aNy[Nx][j]=0;
        apy[Nx][j]=1;
        by[Nx][j]=0;
        }

//para j=1 frontera sur
#pragma omp parallel for
for (i=1; i<=Nx; i++)
        {
        aEy[i][1]=0;
        aWy[i][1]=0;
        aSy[i][1]=0;
        aNy[i][1]=0;
        apy[i][1]=1;
        by[i][1]=0;
        }

//para j=Ny frontera norte
#pragma omp parallel for
for (i=1; i<=Nx; i++)
        {
        aEy[i][Nyd]=0;
        aWy[i][Nyd]=0;
        aSy[i][Nyd]=0;
        aNy[i][Nyd]=0;
        apy[i][Nyd]=1;
        by[i][Ny]=0;
        }
#pragma omp parallel for
for (j=2; j<Nyd; j++)
        {
        for (i=2; i<Nx; i++)
                {
                Dey[i][j]=Rdifu*DYPd[j]/DXU[i]; //Rdif debe llevar el mismo coef
```

```
            Dwy[i][j]=Rdifu*DYPd[j]/DXU[i-1]; //la formula es mas compleja formula (5.13) pero para
Rdif constante queda asi
            Dny[i][j]=Rdifu*DXP[i]/DYUd[j];
            Dsy[i][j]=Rdifu*DXP[i]/DYUd[j-1];
            }
      }


//#pragma omp parallel for
for (j=2; j<Nyd; j++)
      {
      for (i=2; i<Nx; i++)
            {
            Fey[i][j]=((ro*U[i][j]+ro*U[i][j+1])/2)*DYPd[j]; //pedir revisar
            Fwy[i][j]=((ro*U[i-1][j]+ro*U[i-1][j+1])/2)*DYPd[j];
            Fny[i][j]=((ro*V[i][j]+ro*V[i][j+1])/2)*DXP[i];
            Fsy[i][j]=((ro*V[i][j-1]+ro*V[i][j])/2)*DXP[i];

            Peey[i][j]=Fey[i][j]/Dey[i][j];
            Pewy[i][j]=Fwy[i][j]/Dwy[i][j];
            Peny[i][j]=Fny[i][j]/Dny[i][j];
            Pesy[i][j]=Fsy[i][j]/Dsy[i][j];
            }
      }


//#pragma omp parallel for
for (j=2; j<Nyd; j++)
      {
      for (i=2; i<Nx; i++)
            {
                  //Agregamos hipotesis de Boussinesq
                        Scy=(    (    (    ((Told[i][j+1]-Told[i][j])*DYP[j])/(2*DYU[j])    )+Told[i][j]    )-
Tinfinito)*ro*Beta*9.81;
                        //cout<<Told[i][j]<<"\n";
                  //
            aEy[i][j]=Dey[i][j]*A(Peey[i][j])+max(-Fey[i][j],0); //Rdif debe llevar el mismo coef
            aWy[i][j]=Dwy[i][j]*A(Pewy[i][j])+max(Fwy[i][j],0);
```

```
            aNy[i][j]=Dny[i][j]*A(Peny[i][j])+max(-Fny[i][j],0);
            aSy[i][j]=Dsy[i][j]*A(Pesy[i][j])+max(Fsy[i][j],0);
            apy[i][j]=/*(ro*DXP[i]*DYP[i]/dt)+*/aWy[i][j]+aEy[i][j]+aSy[i][j]+aNy[i][j]-
Spy*DXP[i]*DYP[j]/*+(Fey[i][j]-Fwy[i][j])+(Fny[i][j]-Fsy[i][j])*/;
            by[i][j]=/*(ro*DXP[i]*DYP[i]/dt)*Uold[i][j]+*/Scy*DXP[i]*DYPd[j]-DXP[j]*(P[i][j+1]-P[i][j]);

            }
      }

}


void solve_MY (void)
{
int i,j;
double aux;
#pragma omp parallel for
for (j=1; j<=Nyd; j++)
      {
      for (i=1; i<=Nx; i++)
            {
            V[i][j]=(aEy[i][j]*Vold[i+1][j]+aWy[i][j]*Vold[i-1][j]+aNy[i][j]*Vold[i][j+1]+aSy[i][j]*Vold[i][j-
1]+by[i][j])/apy[i][j];
            //aux=(aEy[i][j]*Vold[i+1][j]+aWy[i][j]*Vold[i-1][j]+aNy[i][j]*Vold[i][j+1]+aSy[i][j]*Vold[i][j-
1]+by[i][j])/apy[i][j];
            //V[i][j]=Vold[i][j]+0.97*(aux-Vold[i][j]);
            }
      }
}




bool paradaY (double e)
{
      bool l;
      int i, j;
```

```
	double res=0;
	#pragma omp parallel for reduction(+:res)
	for (i=1; i<=Nx; i++)
	{
		for (j=1; j<=Nyd; j++)
		{
			res+=pow((V[i][j]-Vold[i][j]),2);
			//res+=pow((aEy[i][j]*V[i+1][j]+aWy[i][j]*V[i-1][j]+aNy[i][j]*V[i][j+1]+aSy[i][j]*V[i][j-
1]+by[i][j]-apy[i][j]*V[i][j]),2);
		}
	}
	res=sqrt(res);
	//cout<<res;
	l=(res>=e);
	return l;
}
/*
double A (double Pe)
{
	double x;
	//centrado x=1-0.5*fabs(Pe);
	//UPWIND, A=1, entonces x=1;
	//ley de potencia
	x=pow(max(0, 1-0.1*fabs(Pe)),5);
	return x;
}

double max (double a, double b)
{
	if (a>=b) {return a;}
	else {return b;}
}
*/

void remplaceV (void)
{
	int i,j;
	#pragma omp parallel for
```

```
        for (j=1; j<=Nyd; j++)
                {
                for (i=1; i<=Nx; i++)
                        {
                        Vc[i][j]=V[i][j]+(DXP[i]/apy[i][j])*(Pc[i][j]-Pc[i][j+1]);
                        }
                }

}

void updateV (void)
{
        int i,j;
        #pragma omp parallel for
        for (j=1; j<=Nyd; j++)
                {
                for (i=1; i<=Nx; i++)
                        {
                        Vold[i][j]=V[i][j];
                        }
                }

}
```

funcionesP.h

```
//#include "gauss.h"

void coef_P (void);
void solve_P (void);
bool paradaP (void);
void remplaceP (void);
void resetPc (void);
void updatePc (void);


long double aEP[Nx+1][Ny+1]={0}, aWP[Nx+1][Ny+1]={0}, aNP[Nx+1][Ny+1]={0},
aSP[Nx+1][Ny+1]={0}, apP[Nx+1][Ny+1]={0}, bP[Nx+1][Ny+1]={0};
```

```
long double DeP[Nx+1][Ny+1], DwP[Nx+1][Ny+1], DnP[Nx+1][Ny+1], DsP[Nx+1][Ny+1]; //Fuerza difusiva (en temperatura conduccion)

long double PeeP[Nx+1][Ny+1], PewP[Nx+1][Ny+1], PenP[Nx+1][Ny+1], PesP[Nx+1][Ny+1], SpP, ScP;

void coef_P (void)
{
int i,j;
#pragma omp parallel for
for (j=1; j<=Ny; j++)
	{
	for (i=1; i<=Nx; i++)
		{
		apP[i][j]=1;
		}
	}


//#pragma omp parallel for
for (j=2; j<Ny; j++)
	{
	for (i=2; i<Nx; i++)
		{
		aEP[i][j]=ro*DYP[j]*(DYP[j]/apx[i][j]);
		aWP[i][j]=ro*DYP[j]*(DYP[j]/apx[i-1][j]);
		aNP[i][j]=ro*DXP[i]*(DXP[i]/apy[i][j]);
		aSP[i][j]=ro*DXP[i]*(DXP[i]/apy[i][j-1]);
		apP[i][j]=aWP[i][j]+aEP[i][j]+aSP[i][j]+aNP[i][j];
		bP[i][j]=/*(ro0-ro)*deltaVolumen/deltatiempo+*/((ro*U[i-1][j]*DYP[j])-
(ro*U[i][j]*DYP[j]))+((ro*V[i][j-1]*DXP[i])-(ro*V[i][j]*DXP[i]));
		//cout<<apP[i][j]<<"\n\n\nap\n\n\n";
		}
	}

}
```

```
void solve_P (void)
{
int i,j;
double aux;
#pragma omp parallel for
for (j=1; j<=Ny; j++)
        {
        for (i=1; i<=Nx; i++)
                {
                Pc[i][j]=(aEP[i][j]*Pcold[i+1][j]+aWP[i][j]*Pcold[i-
1][j]+aNP[i][j]*Pcold[i][j+1]+aSP[i][j]*Pcold[i][j-1]+bP[i][j])/apP[i][j];
                //cout<<Pc[i][j]<<"\n\n\nPc\n\n\n";
                //aux=(aEP[i][j]*Pcold[i+1][j]+aWP[i][j]*Pcold[i-
1][j]+aNP[i][j]*Pcold[i][j+1]+aSP[i][j]*Pcold[i][j-1]+bP[i][j])/apP[i][j];
                //Pc[i][j]=Pcold[i][j]+0.8*(aux-Pcold[i][j]);
                }
        }
}


bool paradaP (double e)
{
        bool l;
        int i, j;
        double res=0;
        #pragma omp parallel for reduction(+:res)
        for (i=1; i<=Nx; i++)
        {
                for (j=1; j<=Ny; j++)
                {
                        res+=pow((Pc[i][j]-Pcold[i][j]),2);
                        //res+=pow((aEP[i][j]*Pc[i+1][j]+aWP[i][j]*Pc[i-
1][j]+aNP[i][j]*Pc[i][j+1]+aSP[i][j]*Pc[i][j-1]+bP[i][j]-apP[i][j]*Pc[i][j]),2);
                }
        }
        res=sqrt(res);
```

```
        //cout<<res;
        l=(res>=e);
        return l;
}

void remplaceP (void)
{
        int i,j;
        #pragma omp parallel for
        for (j=1; j<=Ny; j++)
                {
                for (i=1; i<=Nx; i++)
                        {
                        P[i][j]=Pg[i][j]+Pc[i][j];
                        }
                }
}

void resetPc (void)
{
        int i,j;
        #pragma omp parallel for
        for (j=1; j<=Ny; j++)
                {
                for (i=1; i<=Nx; i++)
                        {
                        Pc[i][j]=0;
                        Pcold[i][j]=0;
                        }
                }
}

void updatePc (void)
{
        int i,j;
        for (j=1; j<=Ny; j++)
                {
                for (i=1; i<=Nx; i++)
```

```
                    {
                    Pcold[i][j]=Pc[i][j];
                    }
                }
}
```

funcionesT.h

```
//#include "base.h"

void coef_T (void);
void solve_T (void);
bool paradaT (void);
double A (double Pe);
double max (double a, double b);

long double aE[Nx+1][Ny+1], aW[Nx+1][Ny+1], aN[Nx+1][Ny+1], aS[Nx+1][Ny+1], ap[Nx+1][Ny+1], b[Nx+1][Ny+1];


long double Fe[Nx+1][Ny+1], Fw[Nx+1][Ny+1], Fn[Nx+1][Ny+1], Fs[Nx+1][Ny+1]; //Fuerza convectiva

long double De[Nx+1][Ny+1], Dw[Nx+1][Ny+1], Dn[Nx+1][Ny+1], Ds[Nx+1][Ny+1]; //Fuerza difusiva (en temperatura conduccion)

long double Pee[Nx+1][Ny+1], Pew[Nx+1][Ny+1], Pen[Nx+1][Ny+1], Pes[Nx+1][Ny+1], Sp, Sc; // vectores de conveccion

void coef_T (void)
{
int i,j;

RdifT=lambda/(Cp); //ro de difucion
Tinfinito=(Te+Tw)/2;
Beta=2/(Te+Tw);
Ra=(9.81*Beta*(Tw-Te)*pow(H,3))/((RdifT/ro)*(Rdifu/ro));
Sp=0, Sc=0;
```

```
//para Tw frontera oeste
#pragma omp parallel for
for (j=1; j<=Ny; j++)
        {
        aE[1][j]=0;
        aW[1][j]=0;
        aS[1][j]=0;
        aN[1][j]=0;
        ap[1][j]=1;
        b[1][j]=Tw;
        }

//para Te frontera este
#pragma omp parallel for
for (j=1; j<=Ny; j++)
        {
        aE[Nx][j]=0;
        aW[Nx][j]=0;
        aS[Nx][j]=0;
        aN[Nx][j]=0;
        ap[Nx][j]=1;
        b[Nx][j]=Te;
        }

//para j=1 Ts frontera sur
#pragma omp parallel for
for (i=2; i<Nx; i++)
        {
        aE[i][1]=0;
        aW[i][1]=0;
        aS[i][1]=0;
        aN[i][1]=1;
        ap[i][1]=1;
        b[i][Ny]=0; //von neuman
        /*if (i>1) b[i][1]=0;
        else b[i][1]=Tw; //von neuman*/
        }
```

```
//para j=Ny  Tn frontera norte
#pragma omp parallel for
for (i=2; i<Nx; i++)
        {
        aE[i][Ny]=0;
        aW[i][Ny]=0;
        aS[i][Ny]=1;
        aN[i][Ny]=0;
        ap[i][Ny]=1;
        b[i][Ny]=0; //von neuman
        /*if (i<Nx) b[i][Ny]=0; //von neuman
        else b[i][Ny]=Te;*/
        }

#pragma omp parallel for
for (j=2; j<Ny; j++)
        {
        for (i=2; i<Nx; i++)
                {
                De[i][j]=RdifT*DYP[i]/DXU[i]; //Rdif debe llevar el mismo coef
                Dw[i][j]=RdifT*DYP[i]/DXU[i-1];
                Dn[i][j]=RdifT*DXP[i]/DYU[j];
                Ds[i][j]=RdifT*DXP[i]/DYU[j-1];
                }
        }



for (j=2; j<Ny; j++)
        {
        for (i=2; i<Nx; i++)
                {
                //U[i][j]=u;
                //V[i][j]=v;
                Fe[i][j]=ro*Uc[i][j]*DYP[i]; //pedir revisar
                Fw[i][j]=ro*Uc[i-1][j]*DYP[i];
                Fn[i][j]=ro*Vc[i][j]*DXP[i];
```

```
            Fs[i][j]=ro*Vc[i][j-1]*DXP[i];

            Pee[i][j]=Fe[i][j]/De[i][j];
            Pew[i][j]=Fw[i][j]/Dw[i][j];
            Pen[i][j]=Fn[i][j]/Dn[i][j];
            Pes[i][j]=Fs[i][j]/Ds[i][j];
            }
        }



for (j=2; j<Ny; j++)
        {
        for (i=2; i<Nx; i++)
            {
            aE[i][j]=De[i][j]*A(Pee[i][j])+max(-Fe[i][j],0); //Rdif debe llevar el mismo coef
            aW[i][j]=Dw[i][j]*A(Pew[i][j])+max(Fw[i][j],0);
            aN[i][j]=Dn[i][j]*A(Pen[i][j])+max(-Fn[i][j],0);
            aS[i][j]=Ds[i][j]*A(Pes[i][j])+max(Fs[i][j],0);
            ap[i][j]=/*(ro*DXP[i]*DYP[i]/dt)+*/aW[i][j]+aE[i][j]+aS[i][j]+aN[i][j]-
Sp*DXP[i]*DYP[j]/*+(Fe[i][j]-Fw[i][j])+(Fn[i][j]-Fs[i][j])*/;
            b[i][j]=/*(ro*DXP[i]*DYP[i]/dt)*Told[i][j]+*/Sc*DXP[i]*DYP[j];
            //cout<<b[i][j]<<"\n\n\n\n\n\n";
            }
        }

}


void solve_T (void)
{
int i,j;
for (j=1; j<=Ny; j++)
        {
        for (i=1; i<=Nx; i++)
            {
            Tnew[i][j]=(aE[i][j]*Told[i+1][j]+aW[i][j]*Told[i-1][j]+aN[i][j]*Told[i][j+1]+aS[i][j]*Told[i][j-
1]+b[i][j])/ap[i][j];
```

```
                //cout<<Tnew[i][j]<<"\n\n\n\n\n\n";
                }
        }
}




bool paradaT (double e)
{
        bool l;
        int i, j;
        double res=0;
        for (i=1; i<=Nx; i++)
        {
                for (j=1; j<=Ny; j++)
                {
                        res+=pow((Tnew[i][j]-Told[i][j]),2);
                }
        }
        res=sqrt(res);
        //cout<<res;
        l=(res>=e);
        return l;
}

void updateT (void)
{
        int i,j;
        for (j=1; j<=Ny; j++)
                {
                for (i=1; i<=Nx; i++)
                        {
                        Told[i][j]=Tnew[i][j];
                        }
                }
}
```

C.4 Biblioteca “guess.h”

Es una biblioteca donde la función donde los valores iniciales y supuestos son cargados a las matrices.

guess.h

```
#include "files.h"
void takeg (void)// lee los valores iniciales o solucion inicial
{

    if (readguess ())
    {

        cout<<"\n\nFallo la lectura del archivo\n\nComenzamos a suponer\n\n";
        int i, j;

        for (i=1; i<=Nxd; i++)
        {
            for (j=1; j<=Ny; j++)
            {
                if (i==1 or j==1 or i==Nxd)
                {
                    Ug[i][j]=0;
                }
                else
                {
                    if (j==Ny)
                    {
                        Ug[i][j]=2;
                    }
                    else
                    {
                        Ug[i][j]=2;
```

```
                }
            }


            Uold[i][j]=Ug[i][j];
            U[i][j]=Ug[i][j];
        }
}


for (i=1; i<=Nx; i++)
{
        for (j=1; j<=Nyd; j++)
        {
            Vg[i][j]=2;

            Vold[i][j]=Vg[i][j];
            V[i][j]=Vg[i][j];
        }
}

for (i=1; i<=Nx; i++)
{
        for (j=1; j<=Ny; j++)
        {
            Pg[i][j]=0;
            P[i][j]=Pg[i][j];
        }
}

for (i=1; i<=Nx; i++)
{
        for (j=1; j<=Ny; j++)
        {
            Tnew[i][j]=Tinfinito;
            Told[i][j]=Tinfinito;
        }
}
```

```
        }
}

void takesubs (void) // reemplaza los valores supuestos entre iteraciones
{
        int i, j;
        long double q=1;
        //q=readpaso();
        for (i=1; i<=Nxd; i++)
        {
                for (j=1; j<=Ny; j++)
                {
                        Ug[i][j]=Ug[i][j]+((q*0.7)*(Uc[i][j]-Ug[i][j]));
                        U[i][j]=Ug[i][j];
                }
        }

        for (i=1; i<=Nx; i++)
        {
                for (j=1; j<=Nyd; j++)
                {
                        Vg[i][j]=Vg[i][j]+((q*0.7)*(Vc[i][j]-Vg[i][j]));
                        V[i][j]=Vg[i][j];
                }
        }

        for (i=1; i<=Nx; i++)
        {
                for (j=1; j<=Ny; j++)
                {
                        Pg[i][j]=Pg[i][j]+((q*0.8)*Pc[i][j]);
                        P[i][j]=Pg[i][j];
                }
        }

        for (i=1; i<=Nx; i++)
        {
                for (j=1; j<=Ny; j++)
```

```
            {
                Told[i][j]=Told[i][j]+((q*0.8)*(Tnew[i][j]-Told[i][j]));
                Tnew[i][j]=Told[i][j];
            }
        }
}
```

C.5 Biblioteca "gauss.h"

Es una biblioteca Auxiliar, que contiene un método directo de Gauss que podría ser aplicado para calcular las aproximaciones iniciales de la presión, antes de utilizar el método iterativo, no fue utilizado en este trabajo, pero es una herramienta desarrollada con el código. Pero carece de validación.

gauss.h

```
double a[((Nx*Ny)+2)][((Nx*Ny)+1)], x[((Nx*Ny)+1)];
double m[((Nx*Ny)+1)][((Nx*Ny)+1)], aux, s[((Nx*Ny)+1)];
double maux[((Nx*Ny)+2)][((Nx*Ny)+1)];

int n=(Nx*Ny);

void read_phi (char phi);
void return_phi (char phi);

void gaussdirect (char phi)
{
        read_phi (phi);
//declaro variables iniciales
int  i, j, p, k, nrow[((((Nx)*(Ny))+1)];
//float m[(((Nx-1)*(Ny-1))+1)][(((Nx-1)*(Ny-1))+1)], aux, s[(((Nx-1)*(Ny-1))+1)];
```

```
//Paso 1
for (i=1; i<=n; i++)
	{
	s[i]=0;
	for (j=1; j<=n; j++)
		{
		if (fabs(a[i][j])>s[i]) {s[i]=fabs(a[i][j]);}
		}
	if (s[i]==0)
		{
		cout<<"\nEl sistema no tiene solucion unica"<<i<<"..";
		goto stop;
		}
	nrow[i]=i;
	}
//Paso 2
for (i=1; i<=n; i++)
	{
	//cout<<"\n\t\t\t\ti="<<i;
	//cout<<"\n\t\t\t\tsi="<<s[i];
	//Paso 3 det maximo
	p=i;
	for (j=i+1; j<=n; j++)
		{
		if ((fabs(a[j][i])/s[j])>(fabs(a[p][i])/s[p]))
			{
			p=j;
			//cout<<"\n\t\t\t\tp="<<p;
			}
		}

	//Paso 4 verficar solucion unica
	if (a[p][i]==0)
		{
		cout<<"\n\nEl sistema no tiene solucion unica.\n\n";
		goto stop;
		}
	//Paso 5 intercambiar filas
```

```
        if (i!=p)
                {
                //#pragma omp parallel for
                for (k=1; k<=(n+1); k++)
                        {
                        aux=a[i][k];
                        a[i][k]=a[p][k];
                        a[p][k]=aux;
                        }
                }
        //Paso 6
        //#pragma omp parallel for
        for (j=i+1; j<=n; j++)
                {
                //cout<<"\n\t\t\t\tjbajo="<<j;
                                //Paso 7 hallar factor de multiplicacion
                m[j][i]=a[j][i]/a[i][i];

                //Paso 8 efectuar las operaciones entre lines
                for (k=i; k<=(n+1); k++)
                        {
                        a[j][k]=a[j][k]-m[j][i]*a[i][k];
                        }
                }
//cout<<"op";
        }

//Paso 9
if(a[n][n]==0)
        {
        cout<<"\n\nEl sistema no tiene solucion unica\n\n";
        }

//Paso 10
x[n]=a[n][n+1]/a[n][n];

//Paso 11
for (i=n-1; i>=1; i--)
```

```
        {
        aux=0;
        for (j=i+1; j<=n; j++)
                {
                aux=aux+(a[i][j]*x[j]);
                }
        x[i]=(a[i][n+1]-aux)/a[i][i];
//      cout<<"s";
        }

//Paso 12 devolver resultado
        return_phi (phi);
        cout<<"devolvio algo";

        if (0)
        {
        stop:
        cout<<"dio stop";
        }
        //cout<<"no se que pasa";
}

void read_phi (char phi)
{
        int i,j, contd=0;
        //double mtraux[((Nx*Ny)+1)][((Nx*Ny)+2)];
        if (phi=='P')
        {
                //#pragma omp parallel for
                for (j=1; j<=Ny; j++)
                        {

                        for (i=1; i<=Nx; i++)
                                {
                                        contd++;
                                        a[contd][contd]=-apP[i][j]; //punto central

                                        if (i!=1) {a[contd][(contd-1)]=aWP[i][j];} //punto a la ezquierda
```

```
                        if (i!=(Nx)) {a[contd][(contd+1)]=aEP[i][j];} //punto a la derecha

                        if (j!=Ny) {a[contd][(contd+Nx)]=aNP[i][j];} //punto de arriba que aca se
vuelve abajo

                        if (j!=1) {a[contd][(contd-Nx)]=aSP[i][j];} //punto de abajo que aca se
cuelce arriba

                        a[contd][(Nx*Ny)+1]=-bP[i][j];
                    }
                }
        }

        if (phi=='X')
        {
            //#pragma omp parallel for
            for (j=1; j<=Ny; j++)
                {

                for (i=1; i<=Nxd; i++)
                    {
                        contd++;
                        a[contd][contd]=-apx[i][j]; //punto central

                        if (i!=1) {a[contd][(contd-1)]=aWx[i][j];} //punto a la ezquierda

                        if (i!=Nxd) {a[contd][(contd+1)]=aEx[i][j];}

                        if (j!=Ny) {a[contd][(contd+Nxd)]=aNx[i][j];}

                        if (j!=1) {a[contd][(contd-Nxd)]=aSx[i][j];}

                        a[contd][(Nxd*Ny)+1]=-bx[i][j];
                    }
                }
            n=((Nxd*Ny));
        }
```

```
if (phi=='Y')
{
	//#pragma omp parallel for
	for (j=1; j<=Nyd; j++)
		{

		for (i=1; i<=Nx; i++)
			{
				contd++;
				a[contd][contd]=-apy[i][j]; //punto central

				if (i!=1) {a[contd][(contd-1)]=aWy[i][j];} //punto a la ezquierda

				if (i!=Nx) {a[contd][(contd+1)]=aEy[i][j];}

				if (j!=Nyd) {a[contd][(contd+Nx)]=aNy[i][j];}

				if (j!=1) {a[contd][(contd-Nx)]=aSy[i][j];}

				a[contd][(Nx*Nyd)+1]=-by[i][j];
			}
		}
	n=((Nx*Nyd));
}

if (phi=='T')
{
	//#pragma omp parallel for
	for (j=1; j<=Ny; j++)
		{

		for (i=1; i<=Nx; i++)
			{
				contd++;
				a[contd][contd]=-ap[i][j]; //punto central

				if (i!=1) {a[contd][(contd-1)]=aW[i][j];} //punto a la ezquierda
```

```
                                if (i!=Nx) {a[contd][(contd+1)]=aE[i][j];}

                                if (j!=Ny) {a[contd][(contd+Nx)]=aN[i][j];}

                                if (j!=1) {a[contd][(contd-Nx)]=aS[i][j];}

                                a[contd][(Nx*Ny)+1]=-b[i][j];
                        }
                }
        //n=((Nx*Ny));
        }

}

void return_phi (char phi)
{
        int i,j, contd=0;
        if (phi=='P')
        {
                //#pragma omp parallel for
                for (j=1; j<=Ny; j++)
                        {

                        for (i=1; i<=Nx; i++)
                                {
                                        contd++;
                                        Pc[i][j]=x[contd];
                                }
                        }

        }
        if (phi=='X')
        {
                //#pragma omp parallel for
                for (j=1; j<=Ny; j++)
                        {
                        for (i=1; i<=Nxd; i++)
```

```
                        {
                                contd++;
                                U[i][j]=x[contd];
                        }
                }


        }

        if (phi=='Y')
        {
                //#pragma omp parallel for
                for (j=1; j<=Nyd; j++)
                        {

                        for (i=1; i<=Nx; i++)
                                {
                                        contd++;
                                        V[i][j]=x[contd];
                                }
                        }
        }

        if (phi=='T')
        {
                //#pragma omp parallel for
                for (j=1; j<=Ny; j++)
                        {

                        for (i=1; i<=Nx; i++)
                                {
                                        contd++;
                                        Tnew[i][j]=x[contd];
                                }
                        }
        }
}
```

C.5 Función Principal

Aquí es donde el algoritmo SIMPLE efectivamente ocurre en sus siete pasos descriptos, utilizando las funciones de las bibliotecas para una escritura y notación simplificada.

simple.cxx

```
#include "base.h"
#include "funcionesMX.h"
#include "funcionesMY.h"
#include "funcionesP.h"
#include "funcionesT.h"
#include "print.h"
#include "guess.h"
//#include "gauss.h"
//#include "files.h"
bool parada (double e);
void paradaout (void);
main()
{
        int i, j, cont=0, cont2=0;
        long double q=0;


        Grid ();

        takeg ();

        coef_T();

        updateT ();

        first:
        cont++;
```

```
        //q=readvar();
        resetPc();

        //velocidades
        coef_MX();
        coef_MY();
        updateU ();
        updateV ();
        solve_MX();
        //if (q==2 || q==11){gaussdirect('X');}
        solve_MY();
        //if (q==2 || q==12){gaussdirect('Y');}
        //presion
        coef_P();
        for (i=1; i<=20; i++)
        {
        updatePc ();
        solve_P();
        }
        //if (q==3  || q==13){gaussdirect('P');}
        //correcciones
        remplaceP ();
        remplaceU ();
        remplaceV ();
        //temperatura
        coef_T();
        updateT ();
        solve_T();
        //if (q==3  || q==14){gaussdirect('T');}

if (parada(1E-7) /*|| q==1*/ )
        {
                saveresults ();
                nusselt();
                cout<<"\n\n"<<Ra<<"\n\n";
                return 1;
        }
        else
```

```
        {
                cout<<"\nContador= "<<cont<<"\n";
                paradaout();
                takesubs();
                goto first;
        }
}

bool parada (double e)
{
        bool l;
        int i, j;
        double res=0, resx=0, resy=0, resP=0, resT=0;
        #pragma omp parallel for reduction(+:resx)
        for (i=1; i<=Nxd; i++)
        {
                for (j=1; j<=Ny; j++)
                {
                        resx+=pow((Ug[i][j]-Uc[i][j]),2);
                }
        }
        resx=sqrt(resx);

        #pragma omp parallel for reduction(+:resy)
        for (i=1; i<=Nx; i++)
        {
                for (j=1; j<=Nyd; j++)
                {
                        resy+=pow((Vg[i][j]-Vc[i][j]),2);
                }
        }
        resy=sqrt(resy);


        res=resx+resy;
        l=(res<=e);
        return l;
}
```

```
void paradaout (void)
{
	int i, j;
	double res=0, resx=0, resy=0, resP=0, resT=0;
	for (i=1; i<=Nxd; i++)
	{
		for (j=1; j<=Ny; j++)
		{
			resx+=pow((Ug[i][j]-Uc[i][j]),2);
		}
	}
	resx=sqrt(resx);

	cout<<"\n Norma parada velocidad U=\t\t"<<resx;

	for (i=1; i<=Nx; i++)
	{
		for (j=1; j<=Nyd; j++)
		{
			resy+=pow((Vg[i][j]-Vc[i][j]),2);
		}
	}
	resy=sqrt(resy);

	cout<<"\n Norma parada velocidad V=\t\t"<<resy;

	for (i=1; i<=Nx; i++)
	{
		for (j=1; j<=Ny; j++)
		{
			resP+=pow((Pc[i][j]),2);
		}
	}
	resP=sqrt(resP);

	cout<<"\n Norma parada Presion P=\t\t"<<resP;
```

```
        for (i=1; i<=Nx; i++)
        {
                for (j=1; j<=Ny; j++)
                {
                        resT+=pow((Tnew[i][j]-Told[i][j]),2);
                }
        }
        resT=sqrt(resT);

        cout<<"\n Norma parada Temperatura T=\t\t"<<resT;

                cout<<"\n**************************************************\n"<<endl;
}
```