UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

IUMPA
Institut Universitari de Matemàtica Pura i Aplicada

Universidad Politécnica de Valencia

Escuela Técnica Superior de Ingeniería del Diseño

Túnel de viento virtual: Multigrid y Diferencias Finitas Compactas

Trabajo Fin de Máster

Máster en Ingeniería Aeronáutica

Valencia Septiembre 2020

Autor:
Ricardo Arnaiz García-Morato

Tutor:
Sergio Hoyas Calvo

Dedicado a mi compañera
de vida, por hacerme ser quien soy
y acompañarme en este
largo viaje

Especial agradecimiento a mis
padres, por creer siempre en mi
y a mis hermanas, por estar siempre
conmigo.

Agradecimientos también a mi
tutor, el Dr. Sergio Hoyas Calvo,
por la supervisión, ayuda
y orientación durante el transcurso
del proyecto.

Resumen

La turbulencia es probablemente el problema abierto de la física con más aplicaciones en el día a día. Para poder estudiar un flujo turbulento se deben, o bien resolver las ecuaciones de Navier-Stokes, o bien estudiarlo mediante experimentos en túnel de viento. Los estudios en túnel de viento tienen diferentes inconvenientes, como puede ser el gasto energético o la dificultad de probar varias configuraciones, mientras que la resolución analítica de las ecuaciones de Navier-Stokes es inabordable. La solución es el estudio numérico de las ecuaciones.

Existen numerosos métodos para la solución de las ecuaciones de Navier-Stokes, siendo el más preciso la simulación numérica directa, donde no se hace ninguna hipótesis sobre la física del problema, más allá de las que nos llevan a las ecuaciones de Navier-Stokes. El objetivo del trabajo es realizar un método de base para la realización de un túnel de viento virtual. Se tiene, por tanto, que resolver las ecuaciones de Navier-Stokes de la forma más eficiente posible. Para el proyecto del túnel de viento virtual, se ha optado por un método mixto de diferencias finitas compactas – transformada discreta de Fourier para la discretización espacial del dominio. Esto nos lleva a que para resolver el problema, no va a ser necesario resolver una ecuación sino miles de ecuaciones 2D.

Al aplicar este método obtenemos un sistema en el que hay que resolver sistemas de ecuaciones del orden de las decenas de millones de puntos. Para resolver este tipo de problemas se ha optado por un sistema multigrid permite resolver grandes sistemas de ecuaciones de manera rápida y eficiente. En este TFM, se ha realizado un estudio de la aplicabilidad del método multigrid y su uso en problemas 2D discretizados mediante el método de las diferencias finitas compactas.

Palabras clave: CFD,CTFD, Diferencias finitas compactas, Multigrid, DNS

Abstract

Turbulence is probably the open problem in physics with the most applications in daily life. To be able to study turbulent flow you must either solve the Navier-Stokes equations or study it by conducting2 wind tunnel experiments. Wind tunnel studies have different disadvantages, such as the energy expenditure or the difficulty of testing various configurations, while the analytical resolution of the Navier-Stokes equations is unapproachable. The solution is the numerical study of the equations.

There are numerous methods for solving the Navier-Stokes equations, the most accurate being direct numerical simulation, where no hypotheses are made about the physics of the problem beyond those that lead to the Navier-Stokes equations. The aim of the work is to make a basic method for the implementation of a virtual wind tunnel. Therefore, the Navier-Stokes equations have to be solved in the most efficient way possible. For the virtual wind tunnel project, a mixed method of compact finite difference - discrete Fourier transform has been chosen for the spatial discretization of the domain. This leads us to the fact that to solve the problem, it will not be necessary to solve one equation but thousands of 2D equations.

By applying this method we obtain a system in which we have to solve systems of equations of the order of tens of millions of points. To solve this type of problems we have chosen a multigrid system that allows us to solve large systems of equations quickly and efficiently. In this TFM, a study has been made of the applicability of the multigrid method and its use in discrete 2D problems using the compact finite difference method.

Keywords: CFD, CTFD, Compact finite difference, Multigrid, DNS

Índice general

Agradecimientos **I**

Resumen **III**

Abstract **V**

Índice de figuras **X**

Índice de tablas **XII**

Lista de abreviaturas **XIII**

1. Introducción **1**
1.1. Mecánica de fluidos computacional 2
1.2. CFD en Ingeniería Aeronáutica 3
1.3. Simulación numérica en CFD 4
1.3.1. RANS 5
1.3.2. LES 7
1.3.3. DNS 7
1.4. Estado del arte DNS 9
1.5. Objetivo del trabajo 10

2. Motivación del trabajo y herramientas **11**
2.1. Métodos numéricos 13
2.1.1. Discretización espacial 15
2.1.2. Discretización temporal 16
2.1.3. Resolución ecuaciones lineales 17
2.2. Diferencias finitas compactas 18
2.2.1. Expresión genérica del método 19
2.2.2. Obtención de las matrices CTFD 22
2.2.3. Planteamiento ecuación diferencial 24
2.2.4. CTFD vs. FD 25
2.3. Runge-Kutta tres subpasos 28
2.3.1. Condición de estabilidad de Courant-Friedrichs-Lewy 30
2.4. Multigrid 30
2.4.1. Métodos iterativos 31
2.4.2. Ideas multigrid 34

2.4.3. Trasferencia entre mallas . . . 36
2.4.4. Ciclos multigrid . . . 37
2.5. Resumen herramientas numéricas . . . 40

3. Multigrid-CTFD 1d **43**
3.1. Discretización del dominio 1D . . . 43
3.2. Diferencias finitas compactas 1D . . . 44
3.3. Proceso iterativo 1D . . . 46

4. Multigrid-CTFD 2d **51**
4.1. Discretización del dominio 2D . . . 51
4.2. Diferencias finitas compactas 2D . . . 52
4.3. Proceso de iterativo 2D . . . 57

5. Ecuación del calor **61**
5.1. Discretización espacial del dominio Ec. calor . . . 61
5.2. Discretización temporal Ec. calor . . . 62
5.3. Diferencias finitas compactas Ec. calor . . . 62
5.4. Proceso iterativo Ec. calor . . . 63

6. Resultados **65**
6.1. Multigrid-CTFD 1D . . . 65
6.2. Multigrid-CTFD 2D . . . 70
6.3. Ecuación del calor 2D . . . 75

7. Conclusiones y trabajos futuros **79**
7.1. Conclusiones . . . 79
7.2. Trabajos futuros . . . 80

8. Pliego de condiciones y presupuesto **81**
8.1. Pliego de condiciones . . . 81
8.2. Presupuesto . . . 83
8.2.1. Relación horas de trabajo empleadas . . . 83
8.2.2. Recursos materiales y licencias Software . . . 84
8.2.3. Coste total del proyecto . . . 85

Bibliografía **87**

Índice de figuras

1.1. Cascada de energía de Kolmogorov. Eje horizontal: logaritmo número de onda. Eje vertical: logaritmo energía cinética. 5

2.1. Dominio arbitrario en 3D con dirección Z periódica 12
2.2. Eliminación de la geometría en el dominio 3D con *Immerse Boundary Method* 12
2.3. Dominio discretizado en el intervalo $0 \leq x \leq 1$. Espaciado $h = \frac{1}{n}$ y el punto $x_j = jh$ para $0 \leq j \leq n$ 14
2.4. Error relativo en el cálculo de la primera derivada de $f(x) = 3 \cdot \sin(1+2x)$ mediante CTFD y FD 26
2.5. Error relativo en el cálculo de la segunda derivada de $f(x) = 3 \cdot \sin(1+2x)$ mediante CTFD y FD 27
2.6. Solución de la ecuación diferencial $-\mathbf{u}'' + \mathbf{u} = \mathbf{f}$ analítica y mediante CTFD y FD 28
2.7. Error relativo en el cálculo de la solución de la ecuación diferencial $-\mathbf{u}'' + \mathbf{u} = \mathbf{f}$ mediante CTFD y FD 28
2.8. Reducción del error de aproximación tras 10 iteraciones sobre aproximación inicial (a) baja frecuencia, (b) alta frecuencia, (c) suma de ambas frecuencias. En negro continuo error inicial, rojo Jacobi ponderado, azul Jacobi original y negro discontinuo Gauss-Seidel. 32
2.9. Error de aproximación en norma infinita en función del número de iteraciones con el método de Gauss-Seidel 33
2.10. Onda con número de onda $k = 6$ en malla Ω^h, $N = 24$, proyectada en malla Ω^{2h}, $N = 12$. 35
2.11. Reducción del error de aproximación en una malla de 64 celdas tras 12 iteraciones con Gauss-Seidel y tras un ciclo *two-grid correction* con tres iteraciones en el nivel fino y 6 en el grueso. 36
2.12. Estructura de mallas ciclo en V con 5 niveles de malla 39
2.13. Estructura de mallas ciclo μ con $\mu = 2$ y 4 niveles de malla 40
2.14. Estructura de mallas fmg con ciclo V y 4 niveles de malla 40

3.1. Dominio discretizado en el intervalo $0 \leq x \leq 1$ mediante una ley tipo tangente hiperbólica con $N = 16$ celdas, $N + 1 = 17$ nodos y $\gamma = 1{,}8$. 43

4.1. Dominio discretizado en el intervalo $0 \leq x \leq 2$, $0 \leq y \leq 2$ mediante una ley tipo tangente hiperbólica no equiespaciada con $N = 16$ celdas en eje X y $M = 8$ celdas en eje Y. 52

4.2. Distribución de valores no nulos a lo largo de la matriz G del sistema en 2D, con $N = 16$, $M = 8$, $J_1 = 3$ y $J_2 = 5$. 56
4.3. Ponderación de los valores al restringir desde una malla Ω^h hasta Ω^{2h}. 57
4.4. Ponderación de los valores al interpolar desde una malla Ω^{2h} (rojo) hasta Ω^h (negro). 58

6.1. Error en norma infinita obtenido mediante multigrid 1D con Fortran y Matlab y con diferentes nodos de cálculo 67
6.2. Valor de la función $e^x\cos(2x-1)$ (eje Y) en función de los nodos del dominio x_i (eje X) 69
6.3. Error absoluto y relativo de la solución v_i tras resolver el sistema 6.1a con función solución 6.3 mediante multigrid y CTFD con $N = 128$. . 70
6.4. Error en norma infinita obtenido mediante multigrid 2D con Fortran y con diferentes nodos de cálculo 72
6.5. Tiempo por ciclo obtenido mediante multigrid 2D con Fortran y con diferentes nodos de cálculo 73
6.6. Valor de la función $e^x\cos(2x-1)$ (eje Y) en función de los nodos del dominio x_i (eje X) 74
6.7. Error absoluto v_{ij} tras resolver el sistema ec. (6.1a) con función solución ec. (6.3) mediante multigrid y CTFD con $N = 128$, $M = 128$ y tolerancia entre iteraciones de 10^{-12}. 74
6.8. Error en norma infinita obtenido en la Ecuación del calor en función del número de nodos y del tamaño del paso temporal 77
6.9. Error en norma infinita obtenido en la Ecuación del calor en función del número de pasos temporales para $N = 32$. 77

Índice de tablas

3.1. Porcentaje de llenado de las matrices CTFD en función del número de celdas y del número de nodos utilizados a cada lado del nodo de cálculo 45

6.1. Resultados de la ecuación $-\Delta u(x)+\sigma u(x) = f(x)$ resuelta mediante multigrid-CTFD 1d en Fortran. J_1=1 y J_2=1 66
6.2. Resultados de la ecuación $-\Delta u(x)+\sigma u(x) = f(x)$ resuelta mediante multigrid-CTFD 1d en Matlab. J_1=1 y J_2=1 66
6.3. Resultados de la ecuación $-\Delta u(x)+\sigma u(x) = f(x)$ resuelta mediante multigrid-CTFD 1d en Fortran. J_1=2 y J_2=2 68
6.4. Resultados de la ecuación $-\Delta u(x)+\sigma u(x) = f(x)$ resuelta mediante multigrid-CTFD 1d en Matlab. J_1=2 y J_2=2 68
6.5. Resultados de la ecuación $-\Delta u(x,y) + \sigma u(x,y) = f(x,y)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=1 71
6.6. Resultados de la ecuación $-\Delta u(x,y) + \sigma u(x,y) = f(x,y)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=2 71
6.7. Resultados de la ecuación $-\Delta u(x,y) + \sigma u(x,y) = f(x,y)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=2 y J_2=1 72
6.8. Tiempo de resolución de la ec. (6.8a) en función de ν con $N = 2048$ y $M = 2048$ y $J_1 = 1$ y $J_2 = 2$. 73
6.9. Resultados de la ecuación $\frac{\partial u(x,y,t)}{\partial t} - \sigma\Delta u(x,y,t) = f(x,y,t)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=2. $N = 64$ y $M = 64$. $t_0 = 0$, $t_1 = 10^{-3}$. 76
6.10. Resultados de la ecuación $\frac{\partial u(x,y,t)}{\partial t} - \sigma\Delta u(x,y,t) = f(x,y,t)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=2. $N = 32$ y $M = 32$. $t_0 = 0$, $t_1 = 10^{-3}$. 76
6.11. Resultados de la ecuación $\frac{\partial u(x,y,t)}{\partial t} - \sigma\Delta u(x,y,t) = f(x,y,t)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=2. $N = 16$ y $M = 16$. $t_0 = 0$, $t_1 = 10^{-3}$. 76
6.12. Resultados de la ecuación $\frac{\partial u(x,y,t)}{\partial t} - \sigma\Delta u(x,y,t) = f(x,y,t)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=2. $N = 32$ y $M = 32$. $t_0 = 0$, $dt = 10^{-6}$. 76

8.1. Desglose de costes de las horas empleadas en la realización del trabajo por el personal humano 84
8.2. Desglose costes asociados al material utilizado para la realización del trabajo 84

8.3. Costes totales del trabajo incluyendo costes indirectos, beneficio industrial e IVA. 85

Lista de abreviaturas

CFD	*Computational Fluid Dynamics*
CLF	*Courant-Friedrichs-Lewy*
CTFD	*Compact Finite Difference*
DNS	*Direct Numerical Simulation*
EVM	*Eddy Viscosity Models*
FD	*Finite Difference*
FEM	*Finite Element Method*
FVM	*Finite Volume Method*
FMG	*Full-Multigrid*
LES	*Large Eddy Simulation*
RANS	*Reynolds Averaged Navier-Stokes*
RSM	*Reynolds Stress Equation Model*
SGS	*Sub-Grid Scales*

Capítulo 1

Introducción

La utilización de la dinámica de fluidos ha sido muy importante para el desarrollo de la humanidad a lo largo de toda la historia, sin embargo, no fue hasta el siglo XV con las primeras ideas desarrolladas por Leonardo Da Vinci cuando se comenzó con el estudio de la mecánica de fluidos.

A pesar de ello, 17 siglos antes, entre IV y III A.C. en la Antigua Grecia, cuna de la cultura occidental, se establecieron los primeros conceptos que más adelante serían de vital importancia dentro de la mecánica de fluidos. Uno de los principales contribuidores fue Aristóteles, mediante la idea de que un cuerpo en movimiento a través de un fluido encuentra algún tipo de resistencia. Escribió: "*Es imposible decir por qué un cuerpo que se ha puesto en movimiento en el vacío debe llegar a detenerse. ¿Por qué, en efecto, debería detenerse en un punto y no en otro? Como consecuencia, o bien permanecerá necesariamente en reposo, o, si está en movimiento, se moverá indefinidamente a menos que choque con algún obstáculo*"[2]. Este concepto, con el paso de los siglos, sería lo que hoy se conoce como resistencia aerodinámica o *drag*.

Desde las ideas desarrolladas por Leonardo Da Vinci en el siglo XV hasta mediados del siglo XVII, el estudio de la mecánica de fluidos quedó algo estancado, apareciendo muy pocos estudios sobre la materia. Sin embargo, fue a finales de este siglo cuando se realizó uno de los principales avances en la historia de la mecánica de fluidos. Gracias a los estudios realizados por Edme Mariotte y Christian Huygens entre los años 1673 y 1690, y a los nuevos fundamentos teóricos expuestos por Isaac Newton entre esos años, se llegó a la conclusión de que la fuerza ejercida por un fluido sobre un cuerpo no dependía de manera lineal de la velocidad como se pensaba hasta entonces, sino que variaba según el cuadrado de la misma.

A partir de este momento, los estudios sobre la mecánica de fluidos no paran de surgir, y se realizan grandes avances en la materia hasta la actualidad, sobre todo en el siglo XVIII. La relación entre la presión y la velocidad de Daniel Bernoulli, la invención del tubo de Pitot por Henry Pitot, las ecuaciones que gobiernan los fluidos no viscosos por Leonhard Euler o la generalización de dichas ecuaciones para flujos no viscosos, las ecuaciones de Navier-Stokes, realizadas por Claude-Louis Marie Henri Navier y George Gabriel Stokes años más tarde, son solo unos ejemplos de los grandes avances que ocurrieron durante estos años.

Entre mitad de siglo XIX y mitad de siglo XX, siguieron surgiendo importantes avances en la materia como el estudio sobre el flujo turbulento de Osborne Reynolds o la teoría de la capa límite de Ludwig Prandtl. Fue a partir de mitad de siglo XX cuando surgió la necesidad de calcular problemas muy complejos de imposible resolución analítica. Hasta este momento, se estudiaban problemas sencillos con condiciones triviales, que permitían resolver las ecuaciones de la mecánica de fluidos de forma analítica. Sin embargo, problemas de emergente importancia, como el cálculo de flujos hipersónicos alrededor de vehículos de reentrada, hacen imposible la resolución analítica de las ecuaciones. Es en este momento cuando nace una nueva rama de la mecánica de fluidos, denominada mecánica de fluidos computacional (CFD por sus siglas en ingles, *Computational Fluid Dynamics*).

1.1. Mecánica de fluidos computacional

La mecánica de fluidos computacional nace por necesidad a mediados del siglo pasado, y se convierte por importancia con el paso de los años en la tercera rama de la mecánica de fluidos, junto con la experimental y la teórica, para complementarse unas con otras a la hora de solventar problemas y realizar avances.

Debido a las gran cantidad de potencia y de memoria necesaria para la resolución de los diferentes problemas de mecánica de fluidos, el avance del CFD va muy ligado a los avances en la tecnología de los ordenadores. Uno de los primeros estudios relacionados con CFD fue el realizado por Kopal (1947), en el que resolvió numéricamente las ecuaciones que rigen el flujo supersónico alrededor de conos afilados y compiló las grandes tablas de datos resultantes.

Entre los años 50 y 60 se tratan de calcular flujos hipersónicos alrededor de vehículos de reentrada, que combinan fenómenos físicos muy complejos con las ecuaciones de la mecánica de fluidos, resultando imposible desde el punto de vista analítico. Sin embargo, los primeros problemas altamente complejos desde el punto de vista únicamente de la mecánica de fluidos, sin añadir ecuaciones adicionales, surgen en los años 70, como flujos con zonas subsónicas y supersónicas o flujos viscosos que requieren resolver las ecuaciones de Navier-Stokes. Esto, unido a los grandes avances realizados en estos años en el campo de los ordenadores, permitió que se realizaran grandes avances en la materia y mitad de los años 80 ya era viable resolver las ecuaciones de Euler en 3D[3].

En la actualidad, los estudios mediante CFD cubren un amplio espectro de problemas y están presentes en numerosas industrias, como en el diseño de aeronaves, barcos, coches y turbomaquinaría. Pero no sólo en estos campos, sino que también se utiliza en meteorología, oceanografía, astrofísica, biología y un largo etcétera. Gracias a los avances en la tecnología de los ordenadores, es posible calcular a día de hoy, por ejemplo, el diseño de tuberías en apenas segundos en ordenadores personales. Sin embargo, siguen existiendo problemas que requerirían de miles de años para poder ser resuelto con la tecnología actual, como resolver el flujo completo alrededor de un avión.

Sin embargo, el mayor beneficio aportado a la industria por la aplicación de CFD

ha sido su impacto sobre los ensayos el túnel de viento y en banco. Actualmente, resulta más beneficioso desde el punto de vista económico calcular numéricamente las propiedades de un aeronave que medirlas en un túnel de viento.

Los ventajas que ofrece la aplicación CFD respecto los ensayos experimentales son varias. Por una parte, resulta muy complicado verificar el principio de semejanza en ensayos a escala, sobre todo cuando se realizan a altos números de Reynolds. Además, la obtención de medidas experimentales es compleja, ya que muchas veces los instrumentos de medida perturban el campo fluido o simplemente no disponen de la precisión necesaria. Por otra parte, resulta impensable replicar ciertas condiciones en un laboratorio, como por ejemplo una condición de contorno adiabática, o conocer la influencia de cada término de las ecuaciones en el resultado final.

No obstante, sería incorrecto pensar que la mecánica de fluidos computacional representa una tecnología madura. Quedan muchos problemas abiertos que no han sido todavía resueltos con precisión, como la turbulencia, o el acoplamiento de CFD con otras disciplinas como la mecánica de los sólidos.

1.2. CFD en Ingeniería Aeronáutica

Tal y como se ha expuesto en la sección anterior, la mecánica de fluidos computacional está presente en numerosos campos de la ingeniería y de la ciencia en general. Sin embargo, uno de los campos donde mayor impacto ha tenido el uso de CFD ha sido en la Ingeniería Aeronáutica.

La mecánica de fluidos computacional ha tenido un gran impacto, aunque gradual, en el desarrollo de aviones tanto comerciales como militares, obteniendo especialmente buenos resultados en el diseño de alas en configuración de alta velocidad y su integración con los motores. Sin embargo, no sólo ha tenido gran impacto en el diseño de aviones, sino que también ha representado una herramienta imprescindible para el desarrollo de los helicópteros.

En la actualidad, en las dos principales empresas de fabricación de aeronaves, Airbus y Boeing, el uso de CFD para el desarrollo de los aviones está prácticamente presente en todas las partes los mismos. El diseño de las alas, toberas, fuselaje, cabina, estabilizadores, etc, está altamente influido por el desarrollo mediante CFD. Otros elementos como la reversa, el sistema antihielo o las góndolas también se ven influenciados por el uso de CFD, aunque en menor medida y siempre junto con otras técnicas de desarrollo. Sin embargo, en el diseño de ciertos elementos como las superficies de control, el uso del CFD todavía está muy limitado con la tecnología actual.

Actualmente, el uso del CFD en aeronáutica está presente principalmente en dos fases. Por una parte, en la fase de análisis, obteniendo solución al campo fluido para estudiar ciertas variables de interés como velocidad, perfil de temperaturas o presiones. Además, si se automatiza el proceso y se obtienen suficientes datos, se puede llegar a crear una base de datos aerodinámicos para certificación o simuladores de vuelo. Por otra parte, en la fase de diseño. Antiguamente en esta fase se utilizaba

el túnel de viento, sin embargo, una ligera variación en el perfil alar requiere la construcción de un modelo nuevamente. El enfoque actual consiste en optimizar un parámetro mediante la variación de las variables de diseño con ciertas restricciones.

De cara al futuro, un potencial uso del CFD en aeronáutica sería durante las fases de certificación donde, por ejemplo, cambios mínimos en la configuración no requieran pasar de nuevo por todos los procesos.

1.3. Simulación numérica en CFD

Uno de los principales campos de la mecánica de fluidos que representa uno de los retos más difíciles para la disciplina es el estudio de los flujos turbulentos. Estos flujos representan la mayor parte de los que se encuentran tanto en ingeniería como en todos los ámbitos de la vida cotidiana. Es por ello que muchos de los esfuerzos del CFD se centran en el estudio de dichos flujos. Para poder estudiar el comportamiento de los flujos, tanto turbulentos como laminares, es necesario resolver las ecuaciones de Navier-Stokes, que expresadas en notación de Einstein:

$$\frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial P}{\partial x_i} + \nu \frac{\partial^2 U_i}{\partial x_j^2} \tag{1.1}$$

donde U_i son las componentes de la velocidad, P es la presión, rho la densidad y los subíndices iyj toman valores del 1 al 3.

Estos flujos se caracterizan por ser altamente irregulares, donde las fuerzas inerciales predominan por encima de las fuerzas viscosas. Esta relación viene determinada por el número de Reynolds:

$$Re = \frac{UL}{\nu} \tag{1.2}$$

donde U es la velocidad relativa entre el fluido y el cuerpo, L es la longitud característica y ν es la viscosidad cinemática del fluido.

El flujo turbulento, además de presentarse como altamente irregular, tiene un claro carácter no lineal debido a las inestabilidades que presenta, de forma que la resolución analítica de las ecuaciones de Navier-Stokes se vuelve imposible. A pesar de que la solución analítica no es posible, sí existen ciertos desarrollos matemáticos que permiten describir con bastante precisión la turbulencia, destacando entre ellos la cascada de energía de Kolmogorov.

La teoría, desarrollada por Andréi Kolmogórov, afirma que la energía entra al fluido en las escalas más grandes, donde la viscosidad no juega un papel importante y no puede haber disipación de energía. Esta energía, en forma de vórtices, se va descomponiendo en vórtices más pequeños, trasfiriendo la energía a estos. Este proceso continúa hasta llegar finalmente a las escalas más pequeñas, donde las fuerzas viscosas son muy altas y la energía termina por disiparse. Atendiendo a lo anterior, se podría decir que existen tres escalas de la turbulencia, como se muestra de forma esquemática en la Figura 1.1.

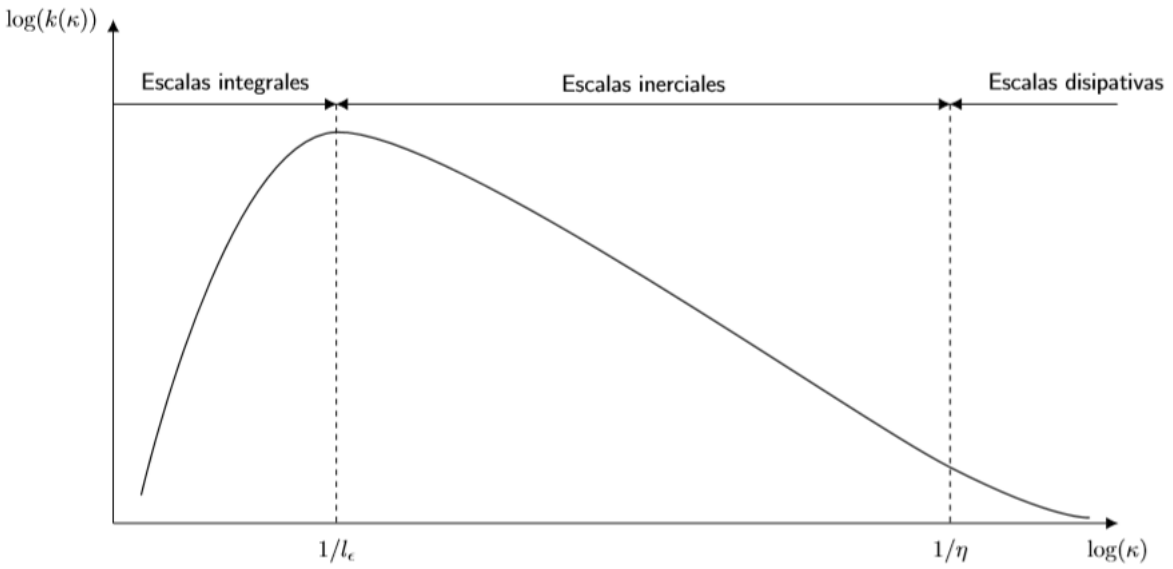

Figura 1.1: Cascada de energía de Kolmogorov. Eje horizontal: logaritmo número de onda. Eje vertical: logaritmo energía cinética.

De esta forma, en función de la escala de longitud se distinguen tres regiones, cada una con diferentes características:

- Escalas integrales: aparecen grandes torbellinos, del tamaño de la longitud característica del problema, y es donde ocurre la mayor producción de energía.
- Escalas inerciales: torbellinos desde grandes a pequeños, no se ven influenciadas por al producción ni disipación de energía, sino que la transfieren desde los torbellinos grandes a los pequeños. Son escalas isótropas y universales.
- Escalas disipativas: torbellinos muy pequeños donde dominan efectos viscosos y se produce la disipación de energía

Por lo tanto, las ecuaciones de Navier-Stokes deben ser resueltas numéricamente y, de forma analítica se conoce la existencia de tres escalas de turbulencia. A día de hoy, los tres principales métodos numéricos para la solución de los flujos turbulentos son RANS (*Reynolds Averaged Navier-Stokes*), LES (*Large Eddy Simulation*) y DNS (*Direct Numerical Simulation*).

1.3.1. RANS

Cada uno de los métodos ofrece un nivel de detalle diferente, y su elección depende del objetivo de la simulación y de la aplicación. El que menor precisión genera es el método RANS. Como su propio nombre indica, consiste en promediar las ecuaciones de Navier-Stokes tras expresar las variables de la forma:

$$u(x, y, z) = \bar{u}(x, y, z) + u'(x, y, z) \tag{1.3}$$

donde $\bar{u}$ es la media temporal y u' es un termino de fluctuación de promedio 0. Aplicando esta descomposición a las ecuaciones de Navier-Stokes, ec. (1.1), y desarrollando un poco los diferentes términos, se obtienen unas ecuaciones análogas a las originales a excepción del término no lineal,

$$\frac{\partial \overline{u_i' u_j'}}{\partial x_j}$$

que aparece debido a las derivadas cruzadas. Este término es conocido como *tensor de esfuerzos de Reynolds* e incluye nuevas incógnitas al problema sin introducir ninguna ecuación. Esto es conocido como el *problema de cierre*. Por lo tanto, se deben incorporar nuevas ecuaciones al problema para poder resolverlo. Actualmente los métodos para incorporar nuevas ecuaciones se podrían dividir en dos grandes grupos, los que se basan en la hipótesis de viscosidad turbulenta y los que utilizan el modelo de ecuación de esfuerzos de Reynolds.

El primero de los métodos se basa en la hipótesis de Boussinesq, que dice que introduce el término ficticio de la viscosidad turbulenta, y dice que los términos del tensor de esfuerzos de Reynolds son proporcionales a los esfuerzos cortantes en cada punto del fluido, siendo la razón de proporcionalidad la viscosidad turbulenta. Esto se ve representado en la siguiente ecuación:

$$-\overline{u_i' u_j'} = \nu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij} \tag{1.4}$$

donde ν_t es la viscosidad turbulenta y k es la energía cinética turbulenta. Los modelos que utilizan está aproximación se conocen como EVM (*Eddy Viscosity Models*) y utilizan una o dos ecuaciones adicionales para obtener los valores de k y ν_t. Para hallar estos valores, se utilizan tablas calibradas mediante ensayos LES o DNS. Existen varios modelos EVM, siendo los más utilizados Spallart-Allmaras, k-omega o k-epsilon y las diferentes variantes de los mismos. Cada uno de ellos funciona mejor en aplicaciones diferentes y, aunque la aproximación de Boussinesq tiene baja precisión en algunos flujos, en ciertas aplicaciones se acerca mucho a la realidad.

El segundo de los métodos para resolver el tensor de esfuerzos de Reynolds es el usado en los modelos RSM (*Reynolds Stress Equation Model*), en el que se añaden ecuaciones al problema para resolver los valores del tensor de esfuerzos de Reynolds directamente, lo que conlleva un aumento de precisión, pero un aumento de costo computacional respecto de los métodos EVM.

Debido a la descomposición que se realiza en el método RANS, tanto con modelos EVM como RSM, es el único método que permite calcular problemas estacionarios, ya que no calcula ninguna de las escalas de la turbulencia, por definición no estacionarias, sino que las modela. Es el método más barato computacionalmente, además del más rápido, pero el menos preciso. Por ello, es el método más extendido en las simulaciones CFD industriales, sobre todo la combinación RANS-EVM.

1.3.2. LES

El método LES, por su parte, es un paso más complejo, ya que se resuelven las escalas más grandes, mínimo las integrales, y el resto de escalas son filtradas mediante un filtro de convolución de tipo paso-bajo, aplicado mediante la discretización espacial. De esta forma, se calculan las escalas donde se genera la mayor parte de la energía, mientras que las más pequeñas, denominadas SGS (*Sub-Grid Scales)* son modeladas. La descomposición de variables es similar a la ec. (1.3), resultando:

$$u(x, y, z) = \bar{u}(x, y, z) + u'(x, y, z) \tag{1.5}$$

donde, en este caso, $\bar{u}$ es la componente filtrada en lugar de la media temporal, y u' es el término residual que representa el movimiento de las SGS. Al aplicar esta descomposición en ec. (1.1), las ecuaciones obtenidas difieren de las originales en que el término del producto de las variables filtradas es distinto del filtro del producto de las variables.

$$\overline{u_i u_j} \neq \bar{u}_i \bar{u}_j$$

La diferencia entre ambos términos es conocido como el término de esfuerzos residuales τ_{ij}^r, que debe tener en cuenta el movimientos de las escalas no resueltas, las SGS. Al igual que ocurre en RANS, la complejidad del método LES, así como el grado de precisión obtenido, radica en obtener buenos modelos que ofrezcan el valor del tensor de esfuerzos residuales. La mayor parte de los métodos que existen para obtener el valor de τ_{ij}^r se basan, al igual que en RANS, en la hipótesis de Boussinesq. Uno de los métodos más utilizados, además de ser el primero y de ser en el que se basan métodos más complejos, es el Smagorinksy-Lilly, donde la viscosidad turbulenta depende principalmente de dos variables, C_s o coeficiente de Smagorinsky, y Δ_x, el tamaño de malla. La precisión del modelo depende principalmente de la elección de la variable C_s, que es independiente del flujo, por lo que es una de las complicaciones del método.

En general, a pesar de obtener una precisión mucho mayor que el método RANS, el coste computacional del LES sigue estando órdenes de magnitud por encima, por lo que su uso en aplicaciones industriales es más limitado, aunque bastante generalizado en campos como la meteorología.

1.3.3. DNS

El último de los métodos es DNS. Es el método más complejo computacionalmente ya que resuelve directamente, sin modelos adicionales ni hipótesis sobre la física del problema, todas las escalas de la turbulencia, desde las escalas integrales hasta las disipativas.

Según la hipótesis de Kolmogorov, en alguna escala L se generan torbellinos de velocidad u_L que, tras un tiempo característico $T_L = L/u_L$, se vuelven inestables, generando tras la inestabilidad torbellinos más pequeños transfiriéndoles energía por unidad de masa y tiempo, ϵ.

$$\epsilon \sim \frac{u_L^2}{T_L} = \frac{u_L^3}{L} \tag{1.6}$$

Este proceso sigue sucediendo hasta que se alcanza el equilibrio, y la tasa de disipación de energía es igual en todas las escalas.

$$u_l \approx (\epsilon l)^{1/3} \tag{1.7}$$

Esta relación es utilizada para relacionar todas las escalas de la cascada de energía de Kolmogorov. Se acostumbra a caracterizar la velocidad de los torbellinos más grandes como la desviación típica de las fluctuaciones de una de las componentes de velocidad, u'. Esto queda justificado porque los torbellinos más grandes tienen velocidades más grandes.

Teniendo en cuenta que las escalas disipativas o de Kolmogorov son aquellas en las que las fuerzas viscosas son del orden de las inerciales, el tiempo en el que se produce la inestabilidad del torbellino de tamaño l, $T_l = l/u_l$, debe de ser igual que el tiempo que requiere la viscosidad para disipar el torbellino, $T_\nu = l^2/\nu$. De esta forma se puede obtener el tamaño de la escala de Komogorov, η.

$$\frac{\eta^2}{\nu} = \frac{\eta}{(\epsilon\eta)^{1/3}} \rightarrow \eta = \left(\frac{\nu^3}{\epsilon}\right)^{1/4} \tag{1.8}$$

Teniendo en cuenta que el tamaño de malla, h, debe recoger desde las escalas más grandes, L, hasta las más pequeñas, η, el número de puntos en 3D necesarios para una DNS sería:

$$N_{3D} = \frac{L}{\eta} = Re^{9/4} \tag{1.9}$$

donde Re es el Reynolds turbulento de los torbellinos de la escala integral:

$$Re = \frac{u'L}{\nu} \tag{1.10}$$

Teniendo en cuenta lo anterior, es posible hacerse una idea del alto coste computacional que requiere realizar una DNS, ya que el número de puntos necesarios crece exponencialmente con el número de Reynolds. Además, el número de operaciones a realizar escala con $Re_\tau^{15/4}$[7], donde Re_τ se define como el número de Reynolds asociado a la velocidad de fricción u_τ.

Además, no sólo existen restricciones para el tamaño de malla, sino que para obtener la precisión necesaria se requiere el uso de mallas estructuradas con un posicionamiento estricto, dejando su validez únicamente para geometrías muy sencillas, de escasa aplicación industrial pero muy importantes a nivel científico, como para obtener valores que luego pueden servir para tabular las constantes de los modelos tanto de RANS como LES.

Para tratar de disminuir el costo computacional al aplicar DNS, se suelen utilizar dominios periódicos en diferentes direcciones del dominio, pudiéndose utilizar por tanto las transformadas discretas de Fourier, que son muy eficientes computacionalmente. Si todo el problema es resuelto en el dominio de Fourier, entonces se denominan métodos espectrales, mientras que si algún termino es resuelto en el espacio físico, se denominan pseudo-espectrales.

1.4. Estado del arte DNS

Una de las primeras simulaciones DNS fue llevada a cabo por Orszag y Patterson[11] en el año 1972, donde simularon turbulencia isótropa y homogénea mediante métodos espectrales, en un dominio en 3D de 32 puntos en cada dimension, con un $Re_\lambda = 35$. Dicho estudio, que en su momento fue un auténtico logro, podría ser resuelto en ordenadores personales en cuestión de segundos.

Desde los primeros estudios DNS hasta la actualidad, la potencia computacional de los mayores superordenadores ha ido aumentando de forma exponencial. Aproximadamente, desde los años 60 hasta los años 90, la potencia de los superordenadores, medida en *flops*, operaciones de coma flotante por segundo, aumentó en 2 ordenes de magnitud cada 10 años[15], pasando de los 100 *kflops* de IBM 7090 a 1 *Gflop* de NEC SX-2. Sin embargo, Desde los años 90 hasta la actualidad, el aumento conseguido ha sido aproximadamente de 3 ordenes de magnitud tal y como puede observarse en [16].

Se podría llegar a pensar que, si desde el estudio de Orszag y Patterson hasta hoy, la potencia de los ordenadores ha aumentado en 12 órdenes de magnitud, las simulaciones DNS también podrían hacerlo. Sin embargo, esto es un error ya que como se ha explicado en párrafos anteriores, el costo computacional de una DNS está altamente ligado al número de Reynolds, y la relación no es lineal, sino que es muy exponencial.

Al tiempo de la redacción del presente trabajo, el estudio publicado con el mayor número de Reynolds de fricción alcanzado en una simulación DNS de flujos turbulentos en canales, es de $Re_\tau = 8000$. Este estudio fue publicado por Yamamoto y Tsuji en 2018 [18]. Sin embargo, también se ha llegado a realizar simulaciones a $Re_\tau = 10000$, generando una base de datos de 75TB con $8{\cdot}10^{10}$ puntos, también en el año 2018 [5], aunque de esta simulación todavía no hay artículo. Por lo tanto, en flujos de turbulentos en canales, las simulaciones DNS ya están alcanzando números de Reynolds del orden de los obtenidos en experimentos, donde por ejemplo, Ivan Marusic *et al.* alcanzaron números de Reynolds del orden $Re_\tau \sim 10^5$ en tuberías [9]. La ventaja que en este caso ofrecen las simulaciones DNS respecto de los estudios exprimentales es que, gracias a las mallas tan finas que se deben conseguir para realizar correctamente la simulación, se extrae una cantidad de datos mucho mayor y más completos, eso sí, a costa del gran coste computacional de simular dichas mallas.

Sin embargo, a pesar del gran avance que se está realizando en simulaciones DNS, todavía se está a gran distancia de alcanzar los valores de Re_τ que se dan en experimentos como la turbulencia atmosférica, donde se alcanzan valores del orden de $Re_\tau \sim 10^6$ [6]. Además, uno de los problemas de DNS no sólo reside en tener la potencia necesaria para calcular flujos a grandes números de Reynolds, sino que las bases de datos generadas a partir de dichos cálculos puede llegar a ser del orden de *Peta-bytes* [7], lo que puede generar problemas tanto de almacenamiento como de intercambio de datos entre grupos de investigación.

1.5. Objetivo del trabajo

El objetivo principal del presente trabajo es la realización de un método base en Fortran que permita la realización de un túnel de viento virtual, es decir, un programa que resuelva las ecuaciones de Navier-Stokes para un flujo turbulento mediante DNS en un dominio establecido. Debido al uso de métodos mixtos a la hora de discretizar el dominio 3D del problema, es posible desacoplar una de las dimensiones en la resolución, y tratarlo como miles de problemas en 2D. El objetivo del método base a realizar debe ser por tanto el de resolver de forma eficiente y precisa problemas en derivadas parciales en 2D de gran tamaño.

Para poder ayudar en la consecución del objetivo, se dispone ya de una serie de rutinas en Matlab que permiten calcular las matrices en 1D de las diferencias finitas compactas para la discretización espacial del dominio. Para poder llevar a cabo el objetivo final del trabajo, se deben alcanzar los siguientes objetivos:

- **Desarrollo de rutinas**, tanto en Matlab como en Fortran, que permitan el uso de las matrices de diferencias finitas compactas en los diferentes programas.
- **Desarrollo de algoritmos de métodos iterativos**, tanto en Matlab como en Fortran, que permitan resolver sistemas de ecuaciones lineales.
- **Desarrollo de algoritmos de trasferencia de malla**, tanto en Matlab como en Fortran, que permitan trasferir valores entre diferentes mallas.
- **Desarrollo de rutinas multigrid 1d**, tanto en Matlab como en Fortran, que permitan resolver grandes problemas de ecuaciones diferenciales en 1D con mallas no equiespaciadas.
- **Desarrollo de rutinas multigrid 2d** en Fortran que permitan resolver problemas de ecuaciones diferenciales de gran tamaño en 2D con mallas no equiespaciadas.
- **Desarrollo de rutinas en fortran** que permitan implementar todas las rutinas desarrolladas para resolver un sistema de ecuaciones en derivadas parciales en un dominio no equiespaciado en 2D de decenas de millones de puntos.
- **Desarrollo de rutinas de verificación** en Matlab que permitan comprobar la aplicabilidad del método y su funcionalidad.

Capítulo 2

Motivación del trabajo y herramientas

Tal y como se ha expuesto en el capítulo anterior, el estudio de la turbulencia, y más concretamente de los flujos turbulentos, es de vital importancia en el mundo de la ingeniería y de la investigación, ya que la gran parte de los flujos que se encuentran son turbulentos. Para ello, se deben estudiar de forma experimental, en túneles de viento, o mediante la resolución de las ecuaciones de Navier-Stokes.

El uso del túnel de viento ha sido tradicionalmente la herramienta utilizada para el cálculo de flujos turbulentos alrededor de vehículos u objetos. Sin embargo, tiene una serie de inconvenientes. En primer lugar, los estudios en túneles de viento conllevan un gran gasto energético, que se convierte en aumento de costes. Además, si se desean estudiar diferentes geometrías, se deben crear las maquetas necesarias para ello, con su consiguiente gasto tanto de tiempo como económico. Pero no solo tiene inconvenientes relacionados con la economía, sino que la obtención de medidas es muy complicada debido a la presencia de aparatos de medida que pueden llegar a perturbar el flujo, además que sólo es posible experimentar con ensayos estáticos.

La otra forma de estudiar los flujos turbulentos es mediante la resolución de las ecuaciones de Navier-Stokes. Analíticamente, a fecha actual, todavía no se ha encontrado una solución a las mismas, por lo que abordarlas desde el punto de vista numérico es la solución. Esto es posible gracias al gran avance que se está realizando en el mundo de los ordenadores. No sólo eso, sino que como se puede observar en [16], el ritmo de crecimiento computacional de los ordenadores sigue siendo elevado gracias a una mejora en las estructuras de los mismos. La tendencia actual es incluir más *cores* en los procesadores o nodos, y crear una gran red de comunicación entre diferentes procesadores o nodos. De esta forma, cada procesador tiene acceso rápido a su memoria, mientras que puede acceder mediante la red de comunicación de alta velocidad a la memoria de otros procesadores. Esto es lo que se conoce como cálculo paralelo y ha permitido avanzar mucho en los estudios de problemas que requieren gran capacidad computacional.

Ejemplo de uno de estos superordenadores es *Supermuc*, situado en el centro de Supercomputación de Leibniz, en la Academia de Ciencias y Humanidades de Baviera. Este ordenador dispone de 6336 nodos de cálculo o procesadores, con 48

cores por nodo, cada uno de ellos con 2GB de memoria RAM.

Gracias a los avances en las estructuras de los superordenadores, se pueden intentar resolver problemas más complejos, como las ecuaciones de Navier-Stokes mediante DNS en un entorno controlado y compatibilizar el uso de túneles de viento experimentales con túneles de viento virtuales. El objetivo del trabajo es crear por tanto un método base que permita la realización del túnel de viento virtual.

Idea de resolución

La idea para resolver las ecuaciones de Navier-Stokes es partir de un dominio en 3D como en la Figura 2.1. En este punto se supone que el dominio en z es lo suficientemente grande como para hacer la solución del flujo periódica en esta dirección.

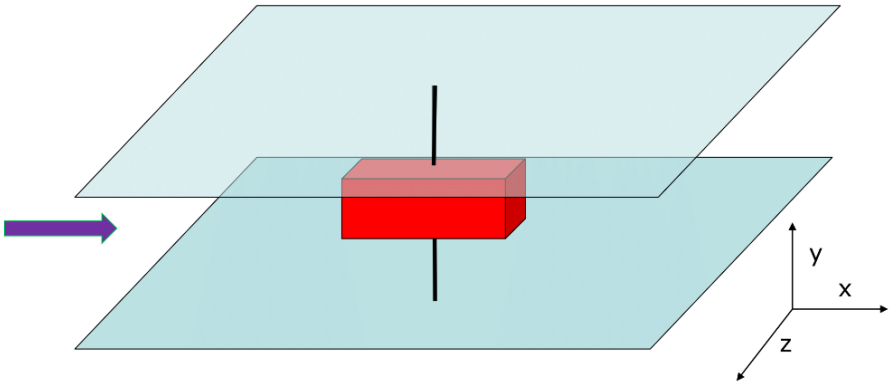

Figura 2.1: Dominio arbitrario en 3D con dirección Z periódica

Al disponer de una dimensión (z) tan grande donde se supone un flujo periódico, es posible emplear la técnica de la trasformada rápida de Fourier para discretizar el problema dicha dimensión z y desacoplar el problema respecto de las otras dos dimensiones. Aplicando esta técnica de métodos mixtos para la discretización espacial del dominio se consigue que, en lugar de tener que resolver las ecuaciones en el domino 3D, se tengan que resolver miles de ecuaciones en el dominio 2D, simplificando considerablemente el problema.

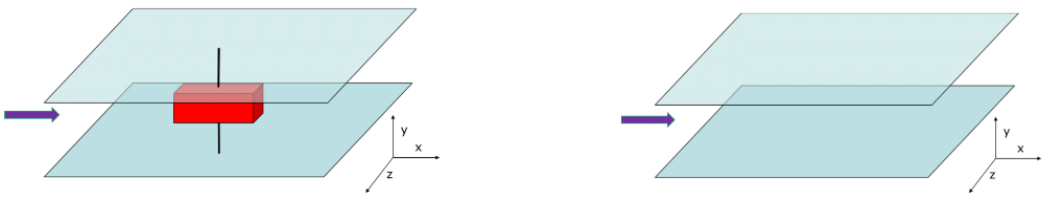

Figura 2.2: Eliminación de la geometría en el dominio 3D con *Immerse Boundary Method*

Por otra parte, para mantener la geometría del problema intacta durante la resolución de diferentes estudios, se utiliza el *Immerse Boundary Method*. Mediante la aplicación de este método es posible sustituir las fronteras del cuerpo de estudio presente en la geometría por fuerzas virtuales e incluir dichas fuerzas el término f de las ecuaciones de Navier-Stokes (Figura 2.2). Las ecuaciones de Navier-Stokes para un flujo incompresible:

$$D_t\vec{u} \;=\; -\nabla p + \Delta\vec{u} + f(\vec{x},t), \tag{2.1}$$

se pueden reescribir mediante la utilización del método *Immerse Boundary Method* en:

$$D_t\vec{u} \;=\; -\nabla p + \Delta\vec{u} + g_{INM}(\vec{x},t). \tag{2.2}$$

Por otra parte, desacoplando la dimensión z mediante la aplicación de las trasformadas rápidas de Fourier se llega a:

$$\partial_t\phi^k \;=\; \Delta\phi^k + g^k(\phi,x,y,t), \quad k=1,\ldots,m, \tag{2.3}$$

donde se dispone de m sistemas de ecuaciones en 2D, cada uno correspondiente con el modo de Fourier asociado de la discretización de la dimensión z. Debido a la cantidad de problemas en 2D que se deben resolver, así como al gran tamaño de los mismos, se deben de emplear una serie de técnicas numéricas que permitan resolverlos con gran precisión y de forma eficiente. En el siguiente apartado se presentan los diferentes métodos existentes para la resolución de problemas similares, así como las herramientas numéricas escogidas para la resolución del problema planteado y el por qué de está elección.

Adicionalmente, cabe destacar que la aplicación tanto de las trasformadas rápidas de Fourier como del método *Immerse Boundary Method* queda fuera del ámbito de este trabajo. El objetivo del trabajo es resolver ecuaciones del tipo:

$$\mathbf{u}_t - \Delta\mathbf{u} = \mathbf{f} \tag{2.4}$$

ya que estas son muy similares a las ecuaciones de Navier-Stokes que se obtienen tras aplicar *Immerse Boundary Method* y la trasformada rápida de Fourier a un fluido incompresible en un dominio periódico en z (ec. (2.3)). Es por esto que los esfuerzos del mismo se destinan a resolver ecuaciones como la mostrada anteriormente.

2.1. Métodos numéricos

Para presentar los diferentes métodos numéricos, es buena solución utilizar un problema ejemplo para desarrollar las diferentes ideas. Por ejemplo, supóngase el problema de la distribución de temperatura estacionaria en una barra en 1D, con los extremos de la barra actuando como condiciones de contorno. Este problema

viene dado por el siguiente problema de condiciones de contorno de segundo orden:

$$-u''(x) + \sigma u(x) = f(x) \quad 0 < x < 1, \quad \sigma \geq 0, \tag{2.5a}$$

$$u(0) = u(1) = 0. \tag{2.5b}$$

Mientras que el problema puede ser resuelto fácilmente mediante métodos analíticos, el objetivo es resolverlo mediante métodos numéricos. El primer paso para resolver el problema es dividir el dominio continuo $[0,\ 1] \in \mathbb{R}$ en n subintervalos, lo que devolvería $n+1$ puntos discretos, $\mathbf{x} = \{x_0,\ x_1, ...,\ x_{n-1},\ x_n\}$, comúnmente conocidos como nodos o puntos del mallado (Figura 2.3). Por ello, si se evalúa la función continua $f(x)$ en los puntos del dominio discreto $\mathbf{x}$, se obtiene el conjunto de imágenes de la función para cada nodo, $\mathbf{f} = \{f_0,\ x_f,\ ...,\ f_{n-1},\ f_n\}$. Así mismo, aplicando lo anterior a la función solución continua, $u(x)$, se pasaría a obtener el vector solución discreto, $\mathbf{u} = \{u_0,\ u_1, ...,\ u_{n-1},\ u_n\}$.

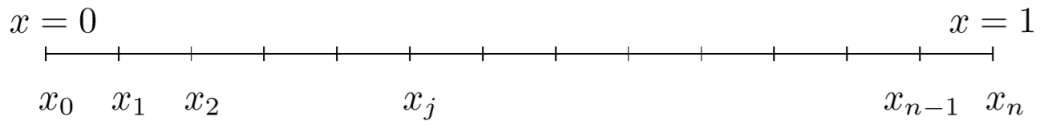

Figura 2.3: Dominio discretizado en el intervalo $0 \leq x \leq 1$. Espaciado $h = \frac{1}{n}$ y el punto $x_j = jh$ para $0 \leq j \leq n$

Una vez discretizado el dominio continuo, se debe resolver la ec. (2.5a) en cada uno de los nodos interiores del dominio discreto $x_j,\ 1 \leq j \leq n-1$, ya que los nodos extremos u_0 y u_n son condiciones de contorno y su valor es conocido. Esto devuelve $n-1$ ecuaciones así como $n-1$ incógnitas, por lo que se dispone de un sistema compatible determinado.

Sin embargo, para poder resolver el sistema, es necesario escribir el término diferencial de la función solución, $u''(x)$, como una combinación lineal de $\mathbf{u}$. Existen numerosos métodos que realizan esta aproximación, desde métodos basados en las diferencias finitas (FD, *Finite Difference*), volúmenes finitos (FVM, *Finite Volume Method*) o métodos espectrales como la transformada de Fourier. De la existencia de dichos métodos así como de la elección de uno de ellos para la realización del presente trabajo se hablará más adelante.

Supóngase ahora la utilización de diferencias finitas de segundo orden para la discretización espacial, y un espaciado entre nodos constante de valor h, las Ecuaciones 2.5a y 2.5b pasarían a ser:

$$\frac{-v_{j-1} + 2v_j - v_{j+1}}{h^2} + \sigma v_j = f_j \quad 1 \leq j \leq n-1, \tag{2.6a}$$

$$u_0 = u_n = 0 \tag{2.6b}$$

donde $\mathbf{v}$ es la aproximación a la solución $\mathbf{u}$ en cada nodo. Escribiendo el sistema de ecuaciones en forma matricial se obtiene:

$$\frac{1}{h^2}\begin{pmatrix} 2+\sigma h^2 & -1 & & & & & & \\ -1 & 2+\sigma h^2 & -1 & & & & & \\ & \cdot & \cdot & \cdot & & & & \\ & & \cdot & \cdot & \cdot & & & \\ & & & \cdot & \cdot & \cdot & & \\ & & & & -1 & 2+\sigma h^2 & -1 & \\ & & & & & -1 & 2+\sigma h^2 & \end{pmatrix}\begin{pmatrix} v_1 \\ \cdot \\ \cdot \\ v_j \\ \cdot \\ \cdot \\ v_{n-1} \end{pmatrix} = \begin{pmatrix} f_1 \\ \cdot \\ \cdot \\ f_j \\ \cdot \\ \cdot \\ f_{n-1} \end{pmatrix} \tag{2.7}$$

Por lo tanto, se ha pasado de tener un problema en el dominio continuo, a un sistema de ecuaciones lineales en forma matricial del tipo $A\mathbf{v} = \mathbf{f}$. Existen principalmente dos grandes grupos de métodos para resolver este tipo de problemas. Por una parte, los métodos directos como por ejemplo la eliminación gaussiana o la descomposición LU, y por otra parte los métodos iterativos, como el de Jacobi o el de Gauss-Seidel.

Por lo tanto, para la resolución numérica de ecuaciones diferenciales de una o más variables, es necesario escoger un discretizador espacial, así como un método para resolver el sistema de ecuaciones lineales que se termina planteando. Además, si una de las variables de las que depende la función es el tiempo, como en el caso de la ecuación del calor (ec. (2.8)), es necesario también escoger un integrador o discretizador temporal.

$$\frac{\partial u(x,t)}{\partial t} = \sigma \frac{\partial^2 u(x,t)}{\partial x^2} \tag{2.8}$$

2.1.1. Discretización espacial

Existen numerosos métodos para realizar la discretización espacial de los operadores diferenciales. Cada uno de los métodos ofrece una serie de ventajas y desventajas para su utilización respecto a otros y la elección de uno de los métodos dependerá de la aplicación y problema concreto que se quiera calcular. A continuación se listan algunos de los métodos más utilizados.

Diferencias finitas - FD

Este método se basa en obtener el valor de las derivadas espaciales en los diferentes nodos del dominio mediante una combinación lineal de los valores de la función en los diferentes nodos. Para ello se hace uso de las series truncadas de Taylor. El error que se comete en la discretización de los operadores diferenciales proviene del error en el truncamiento de las series de Taylor. Uno de los puntos positivos que tienen las diferencias finitas es la flexibilidad a la hora de colocar los nodos del dominio. Sin embargo, a igualdad de nodos, produce peores resultados que otros métodos.

Volúmenes finitos - FVM

En volúmenes finitos, en torno a cada nodo del dominio se construye un volumen de control, sin solaparse con el volumen de los nodos adyacentes, de modo que la

suma de los volúmenes discretizados es igual al volumen total del dominio. La ecuación diferencial se plantea de forma integral en cada volumen de control, obteniendo como resultado una versión discretizada de la misma, y se calcula el flujo a través de las fronteras de los volúmenes de control adyacentes, que actuarán como condiciones de contorno del volumen de cálculo. Uno de los puntos positivos de los volúmenes finitos en el uso CFD es que la conservación de las ecuación se produce de manera explícita por la propia definición del método.

Elementos finitos - FEM

El método de los elementos finitos (FEM, *Finite Element Method*) hace uso de la formulación débil de las ecuaciones diferenciales, expresando la solución como un vector sobre un espacio de funciones. En función del valor de las funciones es obtienen diferentes variaciones del método. Generalmente son funciones polinómicas de carácter nodal, es decir, son nulas en todo el dominio excepto cerca del nodo que representan. Como punto negativo, al igual que las diferencias finitas, a igual número de puntos que otros métodos se obtiene menor precisión.

Diferencias finitas compactas - CTFD

Es el método que se va a desarrollar a lo largo del trabajo. Es un método muy similar a las diferencias finitas, sin embargo, las derivadas espaciales en los diferentes nodos no se expresan como una combinación lineal del valor de la función en los nodos, sino que una combinación lineal de las derivadas espaciales en los nodos se expresa como una combinación lineal del valor de la función en los nodos. De esta forma, para calcular la derivada en un nodo es necesario resolver el sistema de ecuaciones que se forma de la aplicación en todos los nodos. Como punto positivo, ofrece la flexibilidad a la hora de ubicar los nodos al igual que las diferencias finitas o volúmenes finitos, pero ofreciendo una precisión similar a los métodos espectrales.

Métodos espectrales

Estos métodos se basan en descomponer la función solución como una suma de funciones base. Estas funciones deben presentar ciertas propiedades que faciliten el cálculo de las soluciones. Algunos de los métodos espectrales más conocidos son el método de Fourier o de Chebyshev. La ventaja de estos métodos es que una vez se aproxima la función, el cálculo de las derivadas es exacto, por lo que la precisión obtenida es muy alta. Sin embargo, la situación de los nodos en el dominio viene impuesta por directamente por el método, por lo que solo se pueden resolver geometrías muy simples.

2.1.2. Discretización temporal

A la hora de realizar la integración temporal de las ecuaciones del tipo 2.8, existen numerosos métodos numéricos que se distinguen según diferentes características. Por ejemplo, en función del número de pasos, pueden ser unipaso o multipaso, o en función del sistema de ecuaciones resultante, pueden ser implícitos o explícitos.

Método de Euler explícito

Es uno de los métodos más famosos, así como uno de los más sencillos para el cálculo de integración numérica. Es un método unipaso y, como su nombre indica, de tipo explícito. Para calcular el tiempo futuro, se hace uso del tiempo actual, de ahí que sea un método explícito. El error de truncamiento global es del orden de Δt, mientras que el local es del orden Δt^2. Una de sus mayores desventajas es que los pasos temporales tienen que ser muy pequeños para que el método sea estable.

Método de Euler implícito

Al igual que el Euler explícito, el implícito es uno de los más conocidos. Sin embargo, no es de tan sencilla aplicación. Es también un método unipaso, pero, para calcular el tiempo futuro, se hace uso del tiempo futuro, por lo que aparece una relación no lineal y se debe resolver un sistema de ecuaciones. Los órdenes de error son igual que en el método explícito, sin embargo, la ventaja del implícito es que permite pasos temporales mayores.

Método de Crank-Nicolson

Es un método implícito y, al contrario de los métodos de Euler, es de segundo orden en tiempo, además de ser numéricamente estable. Es una combinación entre los métodos de Euler implícitos y explícitos, por lo que también requiere de una solución de un sistema de ecuaciones. Es muy utilizado en problemas de difusión.

Método de Runge-Kutta de tres subpasos

Los métodos de Runge-Kutta, en sus diferentes versiones, son métodos multipaso que, en función de la versión, pueden ser tanto implícitos como explícitos. En concreto, este método de Runge-Kutta es de tres subpasos, ideal para funciones que no dependan explícitamente del tiempo, es de tipo implícito y de tercer orden para funciones lineales y de segundo orden para funciones no lineales. Es muy bueno desde el punto de vista computacional ya que es de bajo almacenamiento, tanto como un Euler explícito. Es el método que se utilizará en el presente trabajo, con el método de Euler implícito para la verificación de soluciones.

2.1.3. Resolución ecuaciones lineales

Para la solución de ecuaciones lineales de forma numérica, existen diferentes tipos de métodos. Como se ha comentado anteriormente, se dividen principalmente en dos grandes grupos, métodos directos e iterativos.

Métodos directos

Los métodos directos son aquellos que, en un número finito de pasos son capaces de calcular, teóricamente, la solución exacta del problema. Evidentemente se obtendrá un error en la resolución relacionado con la aritmética empleada y con la precisión del ordenador. Para ciertos problemas, existen métodos directos muy

optimizados, sin embargo, tienen un gran problema. A medida que el tamaño del problema se vuelve más y más grande, el tiempo de resolución aumenta considerablemente, hasta llegar a un punto que son irresolubles. Además, no solo el tiempo de cálculo es altísimo, sino que la memoria necesaria para almacenar todas las matrices es inviable desde el punto de vista computacional. Debido a esto, se descarta su uso en CFD.

Métodos iterativos

Los métodos iterativos son aquellos que, partiendo de una solución inicial, aproximada o no, e iterando sobre esa solución, son capaces de llegar a una aproximación al valor exacto de la solución con gran precisión. La ventaja de los métodos iterativos sobre los métodos directos es que no es necesario almacenar la matriz del sistema, por lo que es mucho más barato computacionalmente y permite resolver problemas de gran tamaño. Sin embargo, el gran problema que tienen los métodos iterativos es que reducen rápidamente los errores de alta frecuencia, pero una vez la solución aproximada es cercana a la solución exacta, la reducción del error es muy lenta.

Multigrid

El método multigrid nace como solución al problema de reducción del error de los métodos iterativos y es el que se utilizará en el presente trabajo. Se basan principalmente en la utilización de mallas de diferente tamaño, debido a dos conceptos. Por una parte, como se ha comentado en los métodos iterativos, son muy buenos reduciendo el error de alta frecuencia. La clave de este método es que, cuando el error en un tamaño de malla se vuelve de baja frecuencia, es decir, se reduce lentamente, este se restringe a una malla con menos puntos, donde el error será de mayor frecuencia y se reducirá rápidamente. Por otra parte, los métodos iterativos son más rápidos cuanto mejor es la aproximación inicial. Además, cuantos menos puntos tiene la malla, más rápido se llega a la convergencia, por lo que se podría calcular una aproximación a la solución en una malla con pocos puntos, y extrapolarla luego a una malla con más puntos. Estos dos conceptos son en los que se basa el método multigrid y por lo que es tan potente.

2.2. Diferencias finitas compactas

El método de las diferencias finitas, ampliamente utilizado en la resolución de problemas de contorno, se basa en calcular valores diferenciales de la función en un nodo, a partir una combinación lineal de valores de la función en los nodos contiguos. Escrito de forma matricial:

$$\mathbf{u}^{(k)} = B\mathbf{u} \tag{2.9}$$

donde $\mathbf{u}^{(k)}$ es la derivada k-ésima de la función en los diferentes nodos, y B es una matriz que establece las relaciones lineales de la función. Sin embargo, debido a que este método tenía ciertos problemas de precisión en 1992, S.K. Lele, propuso una variación del método llamada diferencias finitas compactas [8] (CTFD - *Compact*

Finite Difference). Lele propuso que una combinación lineal de la derivada de la función en los nodos es otra combinación lineal del valor de la función en los nodos. De esta forma, la igualdad ya no es directa, sino que se debe resolver un sistema de la forma:

$$A\mathbf{u}^{(k)} = B\mathbf{u} \tag{2.10}$$

Este sistema es compatible indeterminado, es decir, el número de soluciones para obtener un mismo error son infinitas. En función del número de nodos involucrados a ambos lados de la ec. (2.10), la precisión del método será mayor o menor.

Las principales ventajas que ofrece el método CTFD es que mantiene la flexibilidad en la colocación de nodos de los métodos de diferencias finitas, pero con un orden de error mucho mayor, obteniendo una precisión cercana a los métodos espectrales. De esta forma, a igualdad de puntos con respecto a las diferencias finitas, se obtiene una precisión mucho mayor, mientras que para obtener la misma precisión, se necesitan muchos menos puntos. Esto es especialmente interesante para problemas de CFD de grandes dominios, no tanto por el tamaño del dominio, sino por el número de nodos involucrados. Como contraposición, se aumente el costo computacional al tener que resolver las incógnitas de la ec. (2.10) al inicio del programa.

2.2.1. Expresión genérica del método

El desarrollo matemático que se realiza a continuación, se realiza sobre un dominio unidimensional por facilidad a la hora de desarrollar las fórmulas. Debe destacarse que, a la hora de discretizar el dominio bidimensional, se realizará de forma desacoplada los ejes X e Y, de forma que será necesario de forma, las ecuaciones a resolver, no deben contener términos no lineales que dependan simultáneamente de derivadas de ambos ejes.

Para comenzar aplicando el método, lo primero que se debe hacer es definir el número de nodos utilizados para el cálculo de la derivada y de la función. En este caso se define J_1 como el número de nodos a utilizar en el cálculo de la derivada a cada lado de un nodo i, y J_2 el número de nodos a utilizar a cada lado de un nodo i. Por tanto, la ec. (2.10) aplicada en un sólo nodo i se puede reescribir de la forma:

$$\sum_{n=-J_1}^{J_1} \alpha_{i,i+n} u_{i+n}^{(k)} = \sum_{n=-J_2}^{J_2} \beta_{i,i+n} u_{i+n}, \quad 0 \leq i \leq N \tag{2.11}$$

donde $N+1$ es el número de nodos del dominio, y α y β son los coeficiente de las matrices A y B de la ec. (2.10) respectivamente.

$$A = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,N+1} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,N+1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{N+1,1} & \alpha_{N+1,2} & \cdots & \alpha_{N+1,N+1} \end{pmatrix} \tag{2.12a}$$

$$B = \begin{pmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,N+1} \\ \beta_{2,1} & \beta_{2,2} & \cdots & \beta_{2,N+1} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N+1,1} & \beta_{N+1,2} & \cdots & \beta_{N+1,N+1} \end{pmatrix} \tag{2.12b}$$

Las matrices A y B son matrices diagonales, con $2J_1 + 1$ y $2J_2 + 1$ diagonales respectivamente. Ahora, en la ec. (2.11) se va a descomponer la derivada $u^{(k)}$ en su desarrollo en serie de potencias de Taylor, siendo la expresión genérica del mismo:

$$\frac{d^k u_j}{dx^k} = \sum_{n=0}^{\infty} \frac{\Delta x_{j,i}^n}{n!} \frac{d^{k+n} u_i}{dx^{k+n}} \tag{2.13}$$

De esta forma, es posible sustituir la ec. (2.13) en la ec. (2.11). Teniendo en cuenta que la serie de Taylor no puede tener términos infinitos a la hora de trabajar numéricamente, se debe establecer un orden de truncamiento predefinido, siendo de valor $H - k$ en la derivada y de valor H en la función. Se obtiene por lo tanto la siguiente ecuación:

$$\sum_{n=-J_1}^{J_1} \left(\alpha_{i,i+n} \sum_{m=0}^{H-k} \frac{\Delta x_{i+n,i}^m}{m!} \frac{d^{k+m} u_i}{dx^{k+m}} + \mathcal{O}(H-k+1) \right) = \sum_{n=-J_2}^{J_2} \left(\beta_{i,i+n} \sum_{m=0}^{H} \frac{\Delta x_{i+n,i}^m}{m!} \frac{d^m u_i}{dx^m} + \mathcal{O}(H+1) \right) \tag{2.14}$$

Se asume a partir de ahora que la igualdad anterior es exacta tras eliminar los términos $\mathcal{O}(H-k+1)$ y $\mathcal{O}(H+1)$ en ambos lados de la ecuación. Se debe destacar que el orden de la ecuación seguirá siendo H. Sin embargo, los nodos característicos de la derivada, J_1, los nodos característicos de la función, J_2, y el orden de truncamiento, H, son linealmente dependientes con dos grados de libertad, relacionados tal que $H = 2(J_1 + J_2)$.

Recuperando la ec. (2.14), es posible agrupar los diferentes términos de la igualdad en función de la derivada $u_i^{(p)}$ en un punto de estudio dado, obteniendo la siguiente ecuación:

$$\sum_{n=0}^{H} \Psi_n u_i^{(p)} = 0 \tag{2.15}$$

donde los términos constantes Ψ_n son una combinación lineal de los términos α y β y de las características geométricas del mallado, por ende, totalmente independientes de los diferentes valor de las derivadas de la función. Por lo tanto, se deduce de la ec. (2.15) que, para obtener orden H en el método, se necesitan $H + 1$ condiciones. Agrupando los términos de la ec. (2.14) de la forma mostrada en la ec. (2.15), se

obtiene un conjunto de $H+1$ ecuaciones de la forma:

$$0 = \sum_{n=-J_2}^{J_2} \beta_{i,i+n} u_i \tag{2.16a}$$

$$0 = \sum_{n=-J_2}^{J_2} \beta_{i,i+n} \frac{\Delta x^1_{i+n,i}}{1!} u_i^{(1)} \tag{2.16b}$$

$$\vdots$$

$$0 = \sum_{n=-J_2}^{J_2} \beta_{i,i+n} \frac{\Delta x^{(k-1)}_{i+n,i}}{(k-1)!} u_i^{(k-1)} \tag{2.16c}$$

$$\sum_{n=-J_1}^{J_1} \alpha_{i,i+n} u_i^{(k)} = \sum_{n=-J_2}^{J_2} \beta_{i,i+n} \frac{\Delta x^k_{i+n,i}}{k!} u_i^{(k)} \tag{2.16d}$$

$$\sum_{n=-J_1}^{J_1} \alpha_{i,i+n} \frac{\Delta x^1_{i+n,i}}{1!} u_i^{(k+1)} = \sum_{n=-J_2}^{J_2} \beta_{i,i+n} \frac{\Delta x^{(k+1)}_{i+n,i}}{(k+1)!} u_i^{(k+1)} \tag{2.16e}$$

$$\vdots$$

$$\sum_{n=-J_1}^{J_1} \alpha_{i,i+n} \frac{\Delta x^{(H-k)}_{i+n,i}}{(H-k)!} u_i^{(H)} = \sum_{n=-J_2}^{J_2} \beta_{i,i+n} \frac{\Delta x^H_{i+n,i}}{H!} u_i^{(H)} \tag{2.16f}$$

donde el conjunto de ecuaciones, independientes todas de los valores de la derivada $u_i^{(p)}$, forman un sistema de ecuaciones homogéneo del tipo $A\mathbf{x} = 0$ en los coeficientes α y β. Los sistemas de ecuaciones homogéneos, son compatibles determinados si la única solución es la trivial, y compatibles indeterminados si tienen alguna solución distinta de la trivial. En este caso, se dispone de $H+1$ ecuaciones y de $H+2$ incógnitas, por lo que se tiene un sistema compatible indeterminado y se puede encontrar una solución diferente de la trivial.

Para resolver el sistema de ecuaciones compatible indeterminado se utiliza lo que se conoce como la técnica de pivotación, es decir, asumir un coeficiente de la solución como conocido. De esta forma se consigue trasformar el sistema compatible indeterminado en compatible determinado, eliminando una incógnita.

El coeficiente escogido para pivotar ha sido $\alpha_{i,i}$, es decir, el coeficiente que acompaña al valor de la derivada en el propio nodo. Este coeficiente sólo tiene influencia en la ec. (2.16d), ya que en resto de términos aparece multiplicado por el valor $\Delta x_{i,i}$, que es la distancia desde el nodo i hasta el propio nodo y por lo tanto, cero, por lo que su valor se anula.

Escogiendo el valor unitario para el término pivote y eliminando ya el valor de la derivada, la ec. (2.16d) puede reescribirse de la forma:

$$1 = \sum_{n=-J_2}^{J_2} \beta_{i,i+n} \frac{\Delta x^k_{i+n,i}}{k!} - \sum_{n=-J_1}^{J_1} \alpha_{i,i+n} \quad con\ n \neq 0 \tag{2.17}$$

Mediante la técnica mostrada en la ec. (2.17) y eliminando el término de las derivadas, el sistema de ecuaciones ec. (2.16a)-ec. (2.16f) puede reescribirse en forma matricial de la forma:

$$\begin{pmatrix} 1 & \cdots & 1 & \cdots & 1 & 0 & \cdots & 0 \\ \Delta x_{i,i-J_2} & \cdots & 0 & \cdots & \Delta x_{i,i+J_2} & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \vdots & \vdots & \vdots \\ \frac{\Delta x^{(k-1)}_{i,i-J_2}}{(k-1)!} & \cdots & 0 & \cdots & \frac{\Delta x^{(k-1)}_{i,i+J_2}}{(k-1)!} & 0 & \cdots & 0 \\ \frac{\Delta x^{k}_{i,i-J_2}}{k!} & \cdots & 0 & \cdots & \frac{\Delta x^{k}_{i,i+J_2}}{k!} & -1 & \cdots & -1 \\ \frac{\Delta x^{(k+1)}_{i,i-J_2}}{(k+1)!} & \cdots & 0 & \cdots & \frac{\Delta x^{(k+1)}_{i,i+J_2}}{(k+1)!} & -\Delta x_{i,i-J_1} & \cdots & -\Delta x_{i,i+J_1} \\ \vdots & & \vdots & & \vdots & \vdots & \vdots & \vdots \\ \frac{\Delta x^{H}_{i,i-J_2}}{H!} & \cdots & 0 & \cdots & \frac{\Delta x^{H}_{i,i+J_2}}{H!} & -\frac{\Delta x^{(H-k)}_{i,i-J_1}}{(H-k)!} & \cdots & -\frac{\Delta x^{(H-k)}_{i,i+J_1}}{(H-k)!} \end{pmatrix} \begin{pmatrix} \beta_{i,i-J_2} \\ \vdots \\ \beta_{i,i} \\ \vdots \\ \beta_{i,i+J_2} \\ \alpha_{i,i-J_1} \\ \vdots \\ \alpha_{i,i+J_1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2.18}$$

donde se resuelven $2J_2 + 1$ términos de β y $2J_1$ términos de α.

2.2.2. Obtención de las matrices CTFD

Mediante la aplicación de ec. (2.18) en un nodo i del dominio, es posible obtener los coeficientes α_i y β_i para dicho nodo i de las matrices ec. (2.12a) y ec. (2.12b) respectivamente. Es decir, aplicando la ec. (2.18) en un nodo, únicamente se completa una fila de las matrices ec. (2.12a) y ec. (2.12b). Por lo tanto, se debe aplicar dicho sistema matricial en todos los nodos del dominio.

Las matrices A y B que se obtienen de la anterior aplicación, son matrices cuadradas de tamaño $(N+1)\times(N+1)$ y, tal y como se ha comentado antes, diagonales con $2J_1+1$ y $2J_2+1$ diagonales respectivamente. Teniendo en cuenta que en el caso general $N > \{J_1, J_2\}$, y que en la mayor parte de casos prácticos $N >> \{J_1, J_2\}$, la mayor parte de las matrices A y B estarán llenas de ceros. Esta característica que presentan las matrices, similar a las matrices obtenidas por diferencias finitas, es de vital importancia a la hora de aplicar posteriormente un método iterativo, multigrid principalmente, para resolver el sistema de ecuaciones resultante.

Sin embargo, para poder aplicar el sistema matricial (2.18) en todos los nodos del domino, es necesario hacer una distinción entre los nodos que pertenecen al dominio interno y los nodos que pertenecen al domino fronterizo.

Nodos del dominio interno

Definiendo Ω como el dominio del problema y $\bar{\Omega}_I$ como el dominio interno, los nodos pertenecientes al dominio interno son aquellos que cumplen:

$$x_i \in \bar{\Omega}_I \subset \Omega \tag{2.19}$$

Sin embargo, la pertenencia al dominio interno $\bar{\Omega}_I$ se ve condicionada por la elección de las variables J_1 y J_2, que representan los nodos característicos de la

función derivada y de la función respectivamente. De está forma, teniendo en cuenta lo anterior, para que un nodo i pertenezca al dominio interno debe cumplir:

$$x_i \in \bar{\Omega}_I \subset \Omega,\ i \in \{0, N\} \Longleftrightarrow \min\{i - J_1, i - J_2\} \geq 0\ \&\ \max\{i + J_1, i + J_2\} \leq N \tag{2.20}$$

De esta forma, en todos los puntos del dominio que cumplan la ec. (2.20), la ec. (2.18) es de plena aplicación, pues se dispone de toda la información necesaria para resolver el sistema.

Nodos del dominio fronterizo

Definiendo ahora $\bar{\Omega}_F$ como el dominio fronterizo, los nodos pertenecientes al dominio fronterizo son aquellos que cumplen:

$$x_i \in \bar{\Omega}_F \subset \Omega \tag{2.21}$$

teniendo en cuenta que $\bar{\Omega}_I \cup \bar{\Omega}_F = \Omega$ y que $\bar{\Omega}_I \cap \bar{\Omega}_F = \emptyset$. De esta forma, una definición más formal, similar a la enunciada para el dominio interno en la ec. (2.20) sería:

$$x_i \in \bar{\Omega}_F \subset \Omega,\ i \in \{0, N\} \Longleftrightarrow \min\{i - J_1, i - J_2\} < 0\ \&\ \max\{i + J_1, i + J_2\} > N \tag{2.22}$$

En este caso, al contrario de lo que ocurre con los puntos del dominio interior, la ec. (2.18) no es de plena aplicación, pues no se dispone de suficiente información, ya que se haría uso de información de nodos que no pertenecen al dominio. En este punto se debe diferenciar los puntos del dominio fronterizo inferior y superior. Los puntos del dominio fronterizo inferior son aquello que, cumpliendo la ec. (2.22), cumplen además que $r = \max\{J_1, J_2\} \rightarrow i \in \{0, r-1\}$. Por otra parte, los puntos del dominio fronterizo superior son aquellos que, cumpliendo con ec. (2.22), cumplen también con $r = \max\{J_1, J_2\} \rightarrow i \in \{N - r + 1, N\}$.

Para poder resolver los puntos del dominio fronterizo, se deben modificar los sumatorios de la ec. (2.14) introduciendo el parámetro r, teniendo en cuenta que la modificación del sumatorio también depende de los parámetros J_1 y J_2. Por ejemplo, en el caso que $J_1 > J_2$, la ecuación del dominio fronterizo inferior quedaría:

$$\sum_{n=-i}^{r} \left(\alpha_{i,i+n} \sum_{m=0}^{H-k} \frac{\Delta x_{i+n,i}^m}{m!} \frac{d^{k+m} u_i}{dx^{k+m}} + \mathcal{O}(H - k + 1) \right) = \sum_{n=\max\{-i,-J_2\}}^{J_2} \left(\beta_{i,i+n} \sum_{m=0}^{H} \frac{\Delta x_{i+n,i}^m}{m!} \frac{d^m u_i}{dx^m} + \mathcal{O}(H + 1) \right) \tag{2.23}$$

En el caso de que $J_2 > J_1$ o $J_2 = J_1$, los sumatorios de la ec. (2.14) se modificarían de forma análoga a la ec. (2.23). En el caso del dominio fronterizo superior, la ecuación sería:

$$\sum_{n=-r}^{N-i}\left(\alpha_{i,i+n}\sum_{m=0}^{H-k}\frac{\Delta x_{i+n,i}^{m}}{m!}\frac{d^{k+m}u_i}{dx^{k+m}}+\mathcal{O}(H-k+1)\right)= \sum_{n=-J_2}^{\min\{J_2,N-i\}}\left(\beta_{i,i+n}\sum_{m=0}^{H}\frac{\Delta x_{i+n,i}^{m}}{m!}\frac{d^{m}u_i}{dx^{m}}+\mathcal{O}(H+1)\right) \quad (2.24)$$

2.2.3. Planteamiento ecuación diferencial

Hasta el momento, el desarrollo realizado para las diferencias finitas compactas trata de obtener unas matrices A y B tal que:

$$A\mathbf{u}^{(k)} = B\mathbf{u} \quad (2.25)$$

El potencial de las diferencias finitas compactas reside en su aplicación sobre ecuaciones diferenciales. Supóngase una ecuación diferencial de la forma

$$-\mathbf{u}'' + \sigma\mathbf{u} = \mathbf{f} \quad (2.26)$$

donde $\mathbf{u}$ y $\mathbf{f}$ son dos funciones que dependen de la misma variable. Es posible resolver esta ecuación mediante el uso de diferencias finitas compactas. Para ello, haciendo uso de la definición del método, se puede obtener dos matrices A y B tal que

$$A\mathbf{u}'' = B\mathbf{u} \quad (2.27)$$

mediante el proceso explicado anteriormente. De esta forma, premultiplicando la ec. (2.26) por la matriz A se obtiene:

$$-A\mathbf{u}'' + \sigma A\mathbf{u} = A\mathbf{f} \quad (2.28)$$

Haciendo uso ahora de ec. (2.27) y sustituyendo en ec. (2.28) se llega hasta la siguiente expresión:

$$-B\mathbf{u} + \sigma A\mathbf{u} = A\mathbf{f} \quad (2.29)$$

formándose un sistema de ecuaciones lineal en forma matricial. Este sistema puede ser resuelto directamente mediante las diferentes propiedades de las matrices, obteniendo la función solución $\mathbf{u}$ buscada:

$$\mathbf{u} = (-B + \sigma A)^{-1} A\mathbf{f} \quad (2.30)$$

Sin embargo, esta solución está incompleta, pues en ningún momento se han tenido en cuenta las condiciones de contorno. Por ejemplo, supóngase ahora el siguiente problema, con condiciones de contorno de tipo Dirichlet,

$$-u''(x) + \sigma u(x) = f(x) \quad 0 < x < 1, \quad \sigma \geq 0, \quad (2.31a)$$
$$u(0) = \alpha, \quad (2.31b)$$
$$u(1) = \beta, \quad (2.31c)$$

donde discretizando la ecuación anterior y escribiéndola en forma matricial se obtiene:

$$-\mathbf{u}'' + \sigma\mathbf{u} = \mathbf{f}, \tag{2.32a}$$
$$u_0 = \alpha, \tag{2.32b}$$
$$u_N = \beta. \tag{2.32c}$$

Se debe tener en cuenta ahora una descomposición del vector $\mathbf{u}$ en la ec. (2.32a) de forma que por un lado se encuentren las incógnitas del problema, y por otro las condiciones de contorno. Expresado en forma matricial:

$$\mathbf{u} = \mathbf{u}_x + \mathbf{u}_{cc} \rightarrow \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} 0 \\ u_1 \\ \vdots \\ u_{N-1} \\ 0 \end{pmatrix} + \begin{pmatrix} u_0 \\ 0 \\ \vdots \\ 0 \\ u_N \end{pmatrix} \tag{2.33}$$

Por lo tanto, premultiplicando la ec. (2.32a) por la matriz A, haciendo uso de la igualdad ec. (2.27) y de la descomposición de $\mathbf{u}$ se obtiene,

$$-B\left(\mathbf{u}_x + \mathbf{u}_{cc}\right) + \sigma A\left(\mathbf{u}_x + \mathbf{u}_{cc}\right) = A\mathbf{f}, \tag{2.34}$$

por lo que ya es posible obtener directamente el valor del vector solución $\mathbf{u}_x$:

$$\mathbf{u}_x = \left(-B + \sigma A\right)^{-1}\left[A\mathbf{f} + \left(B - \sigma A\right)\mathbf{u}_{cc}\right] \tag{2.35}$$

2.2.4. CTFD vs. FD

Para observar bien el potencial de las diferencias finitas compactas respecto de las tradicionales diferencias finitas, se estudiarán unos ejemplos. En primer lugar, se observará la precisión en el cálculo de derivadas para posteriormente hacer lo mismo en el cálculo de una ecuación diferencial.

Cálculo de la primera derivada

Para la evaluación de la precisión se utilizará un dominio en 1D con $N = 100$ celdas y $N + 1 = 101$ nodos. El mallado será equidistante con punto inicial $x_0 = -1$ y punto final $x_N = 1$, por lo que el tamaño de malla será $h = \frac{x_N - x_0}{N} = 0{,}02$. Como función de prueba se establece,

$$f(x) = 3 \cdot \sin(1 + 2x), \tag{2.36}$$

por lo que la primera derivada será,

$$f'(x) = 6 \cdot \cos(1 + 2x). \tag{2.37}$$

Se va a utilizar por una parte un esquema de diferencias finitas centradas de orden 2, y por otra parte un esquema de diferencias fintias compactas con $J_1 = 1$

y $J_2 = 1$. Es decir, tanto en FD como en CTFD se utiliza 1 nodo a cada lado del nodo de cálculo. En la Figura 2.4 se muestra el error relativo, expresado como,

$$\epsilon_R \;=\; \left|\frac{f_n - f_a}{f_a}\right|, \tag{2.38}$$

al calcular la primera derivada de $f(x)$ mediante FD y CTFD.

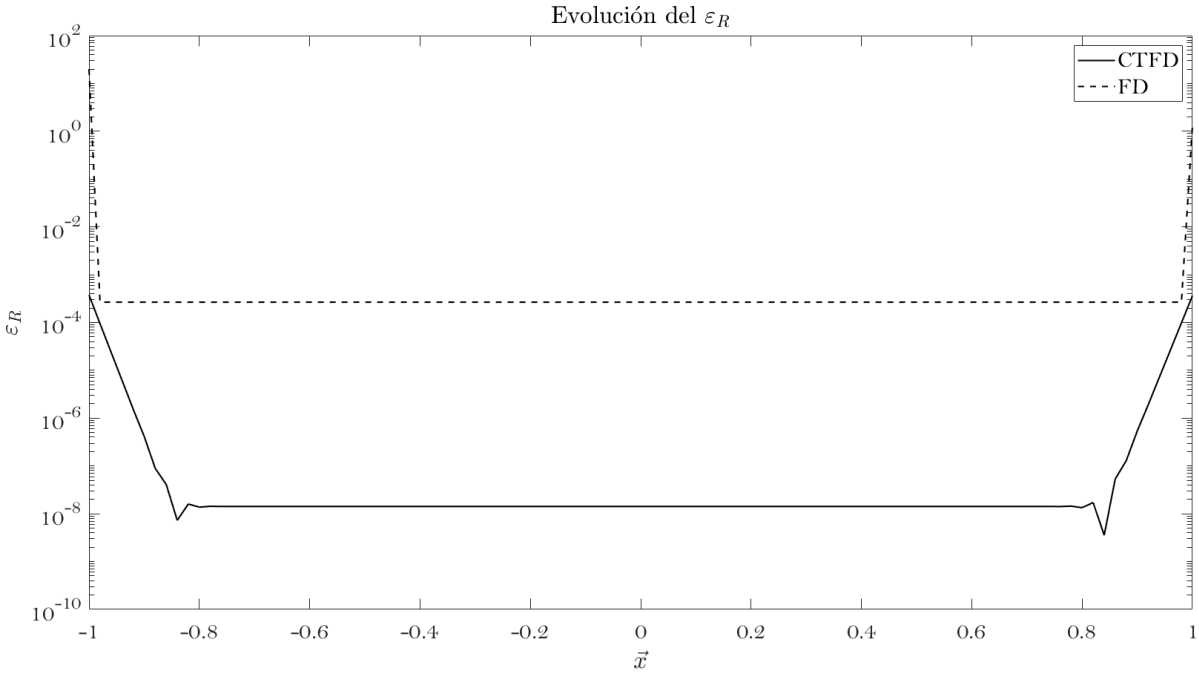

Figura 2.4: Error relativo en el cálculo de la primera derivada de $f(x) = 3{\cdot}\sin(1{+}2x)$ mediante CTFD y FD

Como se puede observar, la precisión obtenida mediante CTFD se sitúa varios ordenes por encima de la obtenida con FD a pesar de utilizar los mismos nodos de cálculo, y 1 nodo de apoyo a cada lado del nodo de cálculo. Además, el error obtenido en los nodos extremos mediante FD es excesivamente grande, debido a la falta de información en esos nodos de cálculo, sin embargo, gracias a las hipótesis aplicadas en los nodos fronterizos mediante CTFD, se consigue que, aunque la precisión sea menor en estos nodos, se mantenga dentro de un orden aceptable.

Cálculo de la segunda derivada

Para el cálculo de la segunda derivada, el proceso es análogo al seguido para la primera derivada. Los datos de cálculo son los mismos, mientras que la segunda derivada de $f(x)$ es:

$$f''(x) \;=\; -12 \cdot \sin(1 + 2x). \tag{2.39}$$

En este caso, los resultados del error relativo se pueden observar en la Figura 2.5.

De nuevo, los resultados extraídos son análogos a los obtenidos en la primera derivada.

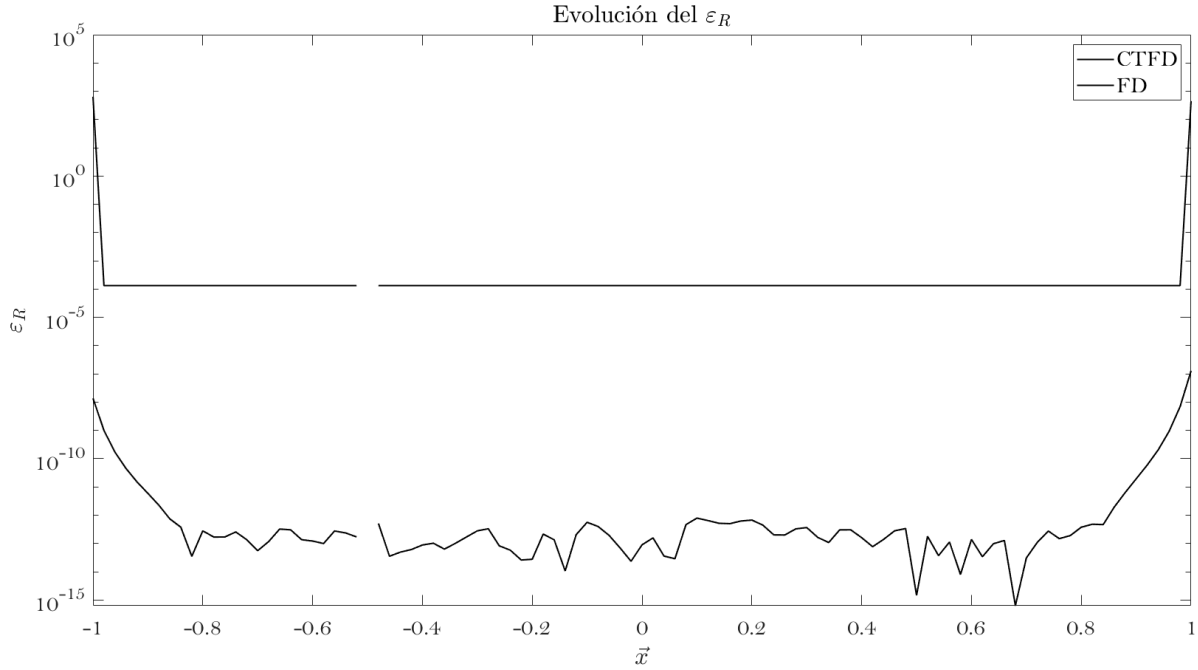

Figura 2.5: Error relativo en el cálculo de la segunda derivada de $f(x) = 3{\cdot}\sin(1{+}2x)$ mediante CTFD y FD

Cálculo de una ecuación diferencial

En este caso se va a resolver una ecuación diferencial para comparar la precisión de ambos métodos. La ecuación a resolver es la siguiente

$$\begin{aligned} -u''(x) + \sigma u(x) &= f(x) \quad -1 < x < 1, \quad \sigma = 1, && \text{(2.40a)} \\ u(-1) &= 0, && \text{(2.40b)} \\ u(1) &= 2{,}7279, && \text{(2.40c)} \end{aligned}$$

donde se ha utilizado una función test para poder comparar la precisión del método $u(x) = 3\sin(1 + x)$, y, por lo tanto, la función $f(x) = 6\sin(1 + x)$.

De esta forma, se va a calcular el error relativo obtenido mediante los métodos numéricos respecto de la solución analítica, es decir:

$$\epsilon_R = \left| \frac{u_n - u_a}{u_a} \right|. \tag{2.41}$$

En este caso se han utilizado únicamente 21 nodos de cálculo, es decir, $N = 20$ celdas. En la Figura 2.6 se puede observar la solución $u(x) = 3\sin(1 + x)$, así como las aproximaciones mediante diferencias finitas compactas y diferencias finitas centradas. Por ambos métodos, el nivel de aproximación es suficiente para, a simple vista, ver una buena aproximación. Para poder observar más detenidamente el nivel de precisión, se representa el error relativo en la Figura 2.7. Al igual que ocurría con el cálculo de las derivadas, el nivel de aproximación mediante CTFD es varios órdenes de magnitud superior al obtenido mediante FD. Cabe destacar que en los

puntos extremo, $x_0 = -1$ y $x_N = 1$, el error relativo es 0, ya que estos puntos son condiciones de contorno.

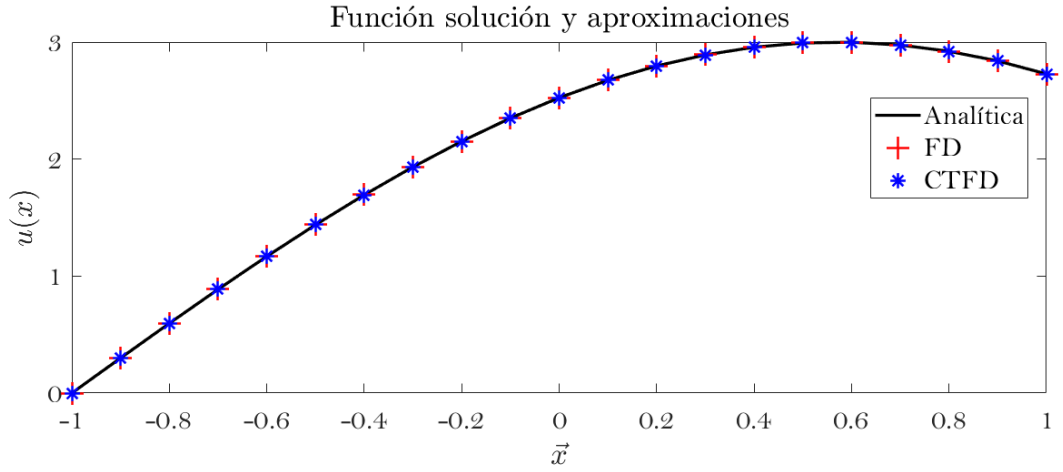

Figura 2.6: Solución de la ecuación diferencial $-\mathbf{u}'' + \mathbf{u} = \mathbf{f}$ analítica y mediante CTFD y FD

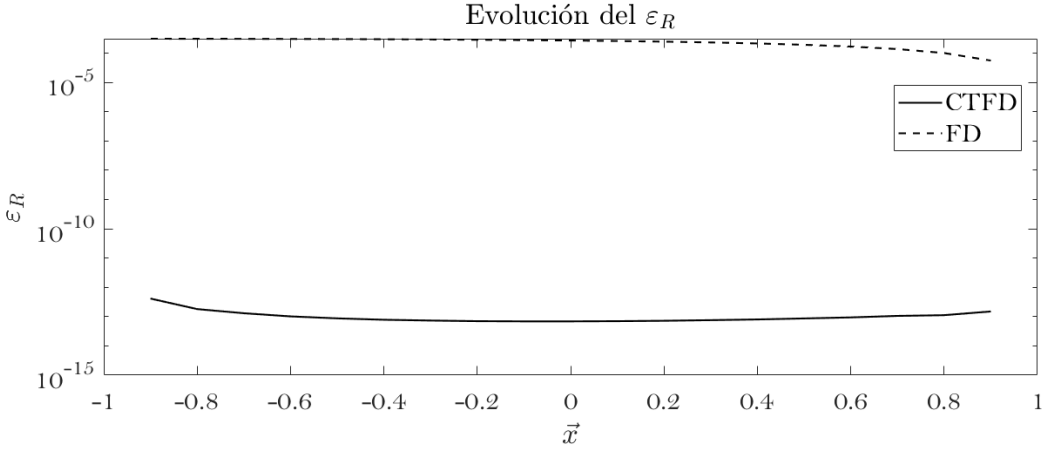

Figura 2.7: Error relativo en el cálculo de la solución de la ecuación diferencial $-\mathbf{u}'' + \mathbf{u} = \mathbf{f}$ mediante CTFD y FD

2.3. Runge-Kutta tres subpasos

Los métodos de Runge-Kutta son métodos para la integración temporal, tanto explícitos como implícitos. Llevan años siendo utilizados, y la expresión general del método es la siguiente,

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \sum_{i=1}^{s} b_i k_i, \tag{2.42}$$

donde s determina el número de pasos del método y k_i son una serie de coeficientes de aproximación intermedios, evaluados en la función de manera local.

$$k_i \;=\; f\left(u^n + \Delta t \sum_{i=1}^{s} a_{ij}k_i,\; t_n + c_i\Delta t\right), \quad i = 1, \ldots, s \tag{2.43}$$

Los coeficientes del esquema numérico a_{ij}, b_i y c_i dependen de la cuadratura utilizada. En función del valor de estos coeficientes el valor de estos coeficientes, se obtiene un tipo de método u otro. Por ejemplo, si el valor de los coeficientes a_{ij} es nulo y $j \geq i$, se obtiene un método explícito, ya que los valores de k_i no dependen de si mismos. El problema de estos métodos es la cantidad de recursos computacionales en forma de memoria que utilizan, por lo que en problemas con grandes dominios, el gasto de memoria es muy alto.

El esquema numérico para la integración temporal utilizado en el trabajo fue propuesto inicialmente por A. Wray [17]. Se trata de un Runge-Kutta de tres pasos de bajo gasto de memoria, pudiendo ser equiparable el gasto a un Euler explícito si se optimiza bien el programa. Sin embargo, el esquema exacto que se utilizará en el trabajo fue propuesto por P.R. Spalart *et al.* [14] para poder tratar los términos no viscosos de las ecuaciones de Navier-Stokes de forma implícita.

Es un esquema diseñado para funciones que no dependan explícitamente del tiempo y requiere dos niveles de almacenamiento. Mientras que para funciones no lineales es de orden 2, para funciones lineales es de orden 3. Supóngase que se dispone de una ecuación diferencial donde se dispone de términos lineales y términos no lineales. Ésta podría escribirse de la siguiente forma,

$$\frac{\partial \mathbf{u}}{\partial t} \;=\; L(\mathbf{u}) + N(\mathbf{u}), \tag{2.44}$$

donde el operador L reúne la parte lineal de la ecuación, y el operador N la parte no lineal. Típicamente, L reúne los términos de presión y los términos viscosos, mientras que N reúne el término convectivo. Sin embargo, cualquier término lineal puede ser incluido en el operador no lineal N. Mientras que el tratamiento de los términos en el operador L será implícito, el de los términos en N será explícito. Para avanzar desde el tiempo t, $\mathbf{u}^n$, hasta el tiempo $t + \Delta t$, $\mathbf{u}^{n+1}$, se deben realizar tres subpasos tal que:

$$\mathbf{u}' \;= \mathbf{u}^n + \Delta t \left[L\left(\alpha_1 \mathbf{u}^n + \beta_1 \mathbf{u}'\right) + \gamma_1 N^n\right], \tag{2.45a}$$

$$\mathbf{u}'' \;= \mathbf{u}' + \Delta t \left[L\left(\alpha_2 \mathbf{u}' + \beta_2 \mathbf{u}''\right) + \gamma_2 N' + \zeta_1 N^n\right], \tag{2.45b}$$

$$\mathbf{u}^{n+1} \;= \mathbf{u}'' + \Delta t \left[L\left(\alpha_3 \mathbf{u}'' + \beta_3 \mathbf{u}^{n+1}\right) + \gamma_3 N'' + \zeta_2 N'\right], \tag{2.45c}$$

donde $N^n \equiv N(\mathbf{u}^n)$, $N' \equiv N(\mathbf{u}')$ y $N'' \equiv N(\mathbf{u}'')$. Los valores para los coeficientes α, β, γ y ζ que mejor satisfacen las condiciones para obtener el orden buscado, fueron halladas en [14] y tienen el siguiente valor:

$$\begin{aligned}
\gamma_1 &= \frac{8}{15}, \quad \gamma_2 = \frac{5}{12}, \quad \gamma_3 = \frac{3}{4}, \quad \zeta_1 = -\frac{17}{60}, \quad \zeta_2 = -\frac{5}{12}, \\
\alpha_1 &= \frac{29}{96}, \quad \alpha_2 = -\frac{3}{40}, \quad \alpha_3 = \frac{1}{6}, \\
\beta_1 &= \frac{37}{160}, \quad \beta_2 = \frac{5}{24}, \quad \beta_3 = \frac{1}{6}.
\end{aligned} \tag{2.46}$$

2.3.1. Condición de estabilidad de Courant-Friedrichs-Lewy

A la hora de discretizar el dominio espacial del problema, se debe tener en cuenta que el dominio de dependencia numérico debe de contener al dominio de dependencia físico, de lo contrario, se pueden generar divergencias. Dicho de una forma más sencilla, se debe asegurar que si un punto de la física está afectado por una zona del dominio, es necesario que ese mismo punto numérico se estime a partir de la información de esa zona del dominio. De esta forma surge la *condición de estabilidad de Courant-Friedrichs-Lewy* o *CFL*:

$$\Delta t_{max} = C \left(\frac{\Delta x}{\lambda} \right)_{min} \tag{2.47}$$

por la cual se establece un valor máximo del paso temporal en función del mínimo tamaño de celda, del número de *Courant* y de los valores propios del sistema. En general, estos autovalores son la velocidad de propagación de la información (la velocidad del fluido en las ecuaciones de Euler por ejemplo). El *CFL* normalmente suele ser menor que 1 y, el cumplimiento de esta condición es necesario pero no suficiente para garantizar la estabilidad del sistema.

Debido a la condición *CFL*, el coste de integrar en el tiempo crece según el cuadrado de la inversa del espaciado entre nodos, debido al aumento de coste del cálculo de la solución y al aumento de pasos temporales necesarios. La ventaja del método de RK empleado en este trabajo respecto a un método de Euler es la posibilidad de utilizar número mucho mayores de *CFL* manteniendo el sistema estable. De este modo, se pueden emplear pasos temporales más grandes sin comprometer la convergencia del proceso, disminuyendo de forma muy grande el coste computacional.

2.4. Multigrid

Para poder entender el método multigrid y saber por qué nace, es necesario hablar primero de los métodos iterativos para la resolución de ecuaciones. Supóngase un sistema de ecuaciones escrito en forma matricial,

$$A\mathbf{u} = \mathbf{f}, \tag{2.48}$$

donde A es la matriz del sistema, $\mathbf{u}$ el vector incógnita y $\mathbf{f}$ el lado derecho del sistema. Este sistema matricial podría haber sido obtenido por ejemplo a partir de una ecuación diferencial aplicada en un dominio discreto, tal y como pasaría con las ecuaciones de Navier-Stokes, siendo más o menos complicado la obtención de A y $\mathbf{f}$.

Los métodos más precisos y más antiguos son los métodos de resolución directa, mediante los cuales, a través de operaciones sobre la matriz A, se puede obtener el valor exacto del vector solución $\mathbf{u}$, obteniendo un error respecto del valor real debido a la aritmética empleada. Estos métodos obtienen el valor de $\mathbf{u}$ de forma muy rápida, estando tremendamente optimizados para según qué aplicaciones. El problema de estos métodos es que el número de operaciones a realizar crece exponencialmente con el tamaño del sistema de ecuaciones. De esta forma, en grandes problemas de fluidos con muchos nodos en el dominio, el sistema de ecuaciones es muy grande y, por lo tanto, irresoluble mediante métodos directos.

2.4.1. Métodos iterativos

Debido a las dificultades para resolver grandes sistemas mediante métodos directos, es cuando aparecen los métodos iterativos. Estos métodos se basan en utilizar una aproximación inicial del vector $\mathbf{u}$, $\mathbf{v}$, e iterar sobre esta aproximación hasta obtener la precisión buscada respecto de la solución.

Supóngase el siguiente problema:

$$-\mathbf{u}'' = \mathbf{f}, \quad 0 < x < 1, \tag{2.49a}$$
$$u(0) = u(1) = 0, \tag{2.49b}$$

discretizado mediante diferencias finitas centradas de segundo orden en los $N+1$ nodos del dominio. El problema discreto se convertiría en:

$$-u_{j-1} + 2u_j - u_{j+1} = h^2 f_j, \quad 1 \leq j \leq N-1, \tag{2.50a}$$
$$u_0 = u_N = 0 \tag{2.50b}$$

Uno de los métodos iterativos más sencillos es el método de Jacobi. Este método utiliza la aproximación inicial en los nodos $j-1$ y $j+1$ para calcular la siguiente aproximación en el nodo j. La estructura sería la siguiente:

$$v_j^{(1)} = \frac{1}{2}\left(v_{j-1}^{(0)} + v_{j+1}^{(0)} + h^2 f_j\right), \quad 1 \leq j \leq N-1, \tag{2.51}$$

donde $\mathbf{v}^{(1)}$ representa la aproximación actual y $\mathbf{v}^{(0)}$ representa la aproximación pasada. Esto crea una estructura iterativa donde, una vez obtenido el valor de todos los nodos de $\mathbf{v}^{(1)}$, esta aproximación se convertiría en $\mathbf{v}^{(0)}$ y se repetiría el proceso hasta alcanzar la convergencia o el valor de precisión deseado. Otro de los métodos básico es el método de Jacobi ponderado,

$$v_j^{(*)} = \frac{1}{2}\left(v_{j-1}^{(0)} + v_{j+1}^{(0)} + h^2 f_j\right), \quad 1 \leq j \leq N-1, \tag{2.52a}$$
$$v_j^{(1)} = (1-\omega)v_j^{(0)} + \omega v_j^{(*)}, \quad 1 \leq j \leq N-1, \tag{2.52b}$$

donde la nueva iteración $v_j^{(1)}$ se obtiene mediante una ponderación ω de la iteración anterior $v_j^{(0)}$ y la iteración intermedia $v_j^{(*)}$. Esta modificación respecto del método

de Jacobi original, tiene una gran importancia, ya que la selección del parámetro ω aporta grandes beneficios a la hora de resolver ciertos problemas.

Ambos métodos de Jacobi no utilizan las nuevas aproximaciones de los nodos j hasta la siguiente iteración, lo cual puede llegar a disminuir la velocidad de convergencia. Debido a esto nace el método de Gauss-Seidel. La estructura de iteración es similar a la de Jacobi original, con la diferencia de que el valor de $v_j^{(1)}$ es actualizado automáticamente para el cálculo de los siguientes nodos:

$$v_j^{(1)} = \frac{1}{2}\left(v_{j-1}^{(1)} + v_{j+1}^{(0)} + h^2 f_j\right), \quad 1 \leq j \leq N-1. \tag{2.53}$$

La principal característica de la mayoría de métodos iterativos es conocida como *smoothing factor*, o factor de alisamiento, y es que se produce una fuerte amortiguación de los términos oscilatorios de alta frecuencia. Esta propiedad es independiente del tamaño de malla, lo cual es muy importante a la hora de desarrollar multigrid.

Supóngase el problema descrito en ec. (2.49a). Para simplificar los cálculo y las demostraciones, se considerará $\mathbf{f} = 0$, de esta forma, $\mathbf{u} = 0$ y el error de aproximación $\mathbf{e} = \mathbf{u} - \mathbf{v} = -\mathbf{v}$. En este caso, de nuevo para facilitar cálculos, se discretiza el dominio mediante diferencias finitas centradas de segundo orden con $N = 64$ celdas equiespaciadas, por lo que hay $N + 1 = 65$ nodos de cálculo. Se van a calcular tres aproximaciones iniciales para comprobar cómo afecta el factor de alisamiento a la hora de alcanzar la convergencia del método.

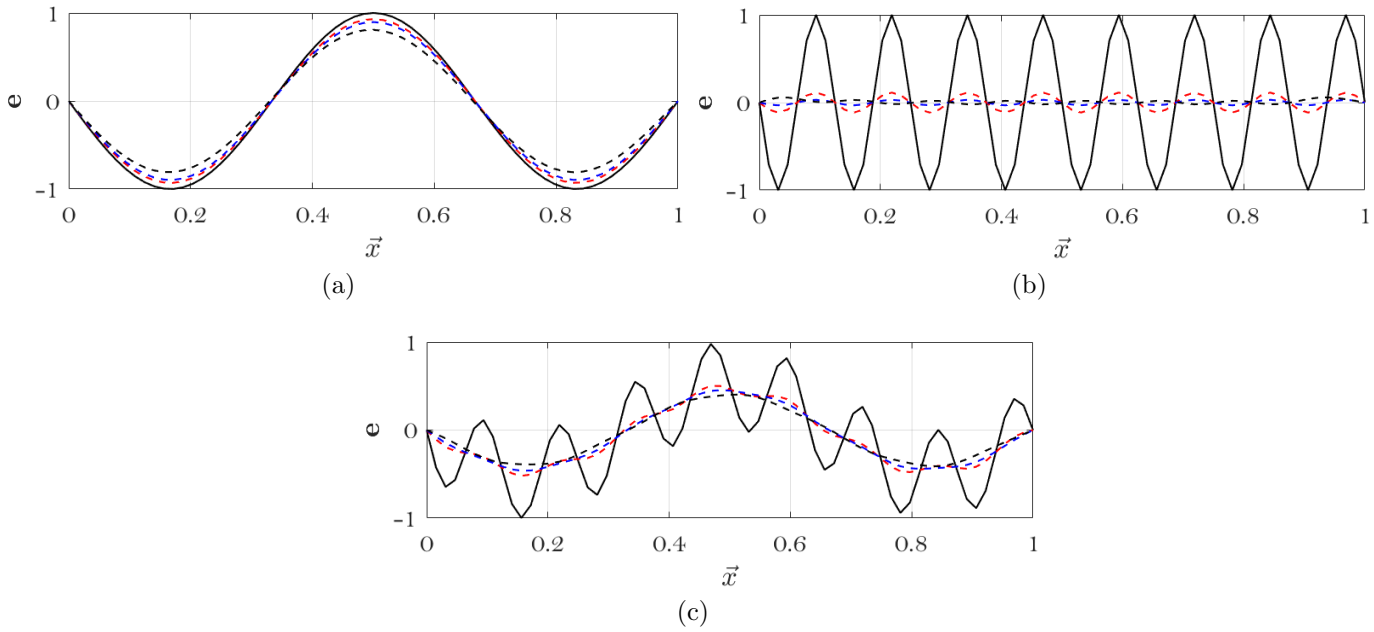

Figura 2.8: Reducción del error de aproximación tras 10 iteraciones sobre aproximación inicial (a) baja frecuencia, (b) alta frecuencia, (c) suma de ambas frecuencias. En negro continuo error inicial, rojo Jacobi ponderado, azul Jacobi original y negro discontinuo Gauss-Seidel.

En la fig. 2.8a se muestra el error inicial y el error en la aproximación tras 10 iteraciones con la aproximación inicial,

$$v_j \;=\; \sin\left(\frac{3j\pi}{N}\right), \quad 1 \leq j \leq N-1, \tag{2.54}$$

aplicando los métodos de Jacobi original, ponderado y Gauss-Seidel. Por otra parte, el la fig. 2.8b se muestra lo mismo, pero esta vez la aproximación inicial es,

$$v_j \;=\; \sin\left(\frac{16j\pi}{N}\right), \quad 1 \leq j \leq N-1, \tag{2.55}$$

y finalmente, en la fig. 2.8c la aproximación inicial es,

$$v_j \;=\; \frac{1}{2}\left(\sin\left(\frac{3j\pi}{N}\right) + \sin\left(\frac{16j\pi}{N}\right)\right), \quad 1 \leq j \leq N-1. \tag{2.56}$$

La conclusión que se puede obtener de la Figura 2.8 es que, efectivamente, tras apenas 10 iteraciones lo efectos oscilatorios de alta frecuencia son eliminados de manera efectiva, mientras que los de baja frecuencia persisten en el tiempo. Esto hace que la convergencia de los métodos iterativos sea muy rápida en las primeras iteraciones, mientras que conforme avanzan las iteraciones, más lenta es la convergencia. Esto puede observarse en la Figura 2.9, donde se ha utilizado como aproximación inicial,

$$v_j \;=\; \frac{1}{3}\left(\sin\left(\frac{3j\pi}{N}\right) + \sin\left(\frac{9j\pi}{N}\right) + \sin\left(\frac{32j\pi}{N}\right)\right), \quad 1 \leq j \leq N-1. \tag{2.57}$$

En las primeras 5/10 iteraciones, se reduce el error rápidamente, llegando el error global a reducirse más de la mitad, ya que se eliminan los modos del error de alta frecuencia, mientras que en las siguiente 90 iteraciones, donde los modos del error error ya son de baja frecuencia, la convergencia comienza a ser muy lenta. Esto pone de manifiesto la capacidad de los métodos iterativos estándares de eliminar rápidamente el error, siempre y cuando sea de alta frecuencia.

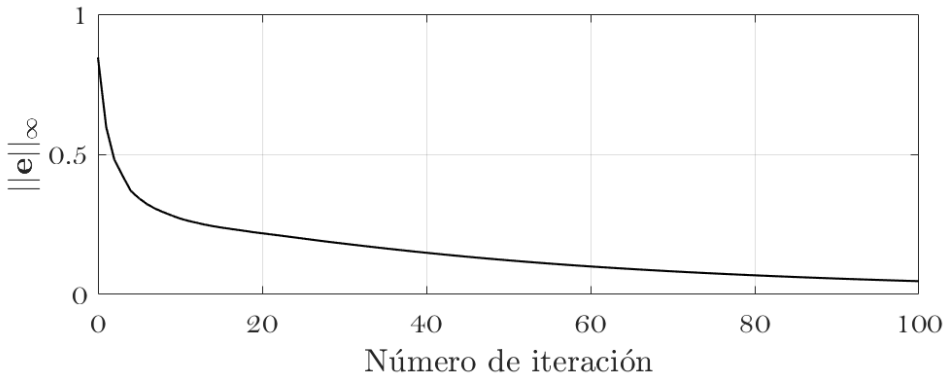

Figura 2.9: Error de aproximación en norma infinita en función del número de iteraciones con el método de Gauss-Seidel

Se ha mostrado los principales problemas de los métodos iterativos a la hora de converger la solución. Cabe destacar que el número de iteraciones es proporcional

a N^2, es decir, si se dobla el número de celdas, el número de iteraciones para converger con la misma tolerancia se multiplica por 4. Sin embargo, el tiempo hasta la convergencia sigue un ritmo exponencial, ya que las iteraciones con más número de incógnitas son más lentas.

2.4.2. Ideas multigrid

Por lo tanto, el multigrid nace con la idea de acelerar la convergencia de los métodos iterativos. Existen varias alternativas para conseguirlo. Por una parte, cuanto mejor sea la aproximación inicial, menos tardará el método en converger a la precisión deseada. Para conseguir una mejor aproximación inicial en la malla de estudio, es posible realizar primero unas iteraciones en una malla más gruesa. De esta forma se consigue una mejor aproximación inicial para comenzar a iterar en la malla de estudio. La base detrás de esto es que, iterar en mallas más gruesas es mucho más barato y rápido computacionalmente, y al obtener una mejor aproximación inicial en la malla de estudio, se necesitan menos iteraciones en dicha malla.

La idea es que, para obtener la aproximación en la malla gruesa, se puede iterar en una malla todavía más gruesa, y así repetidamente. De esta forma se formaría la estrategia conocida como *nested iteration*. El esquema sería el siguiente:

- Iterar $A\mathbf{v} = \mathbf{f}$ en una malla muy gruesa para obtener una aproximación inicial para una malla más fina
  $\vdots$
- Iterar $A\mathbf{v} = \mathbf{f}$ en Ω^{4h} para obtener una aproximación inicial en Ω^{2h}
- Iterar $A\mathbf{v} = \mathbf{f}$ en Ω^{2h} para obtener una aproximación inicial en Ω^{h}
- Iterar $A\mathbf{v} = \mathbf{f}$ en Ω^{h} hasta la convergencia

En este caso, Ω^h es el dominio con un tamaño de malla h, mientras que Ω^{ih} es el dominio con un tamaño de malla $ih,\ i > 1$.

Por otra parte, el principal problema de los métodos iterativos es que, una vez que los errores son de baja frecuencia, la velocidad de convergencia baja mucho. Supóngase una onda con número de onda k en una malla Ω^h con N celdas de la forma:

$$w^h_{k,j} = \sin\left(\frac{kj\pi}{N}\right), \quad 0 \leq j \leq N, \tag{2.58}$$

donde el superíndice en w indica la malla en la que se representa la onda, k representa el número de onda y j los nodos en los que se representa. Si ahora se coge una malla (Ω^{2h}) con la mitad de celdas, $N/2$, los puntos del mallado corresponderían con los puntos pares de la malla Ω^h, por lo que la onda puntos pares de la onda ec. (2.58) podría ser proyectada en la malla Ω^{2h} de la forma:

$$w^h_{k,2j} = \sin\left(\frac{k2j\pi}{N}\right) = \sin\left(\frac{kj\pi}{N/2}\right) = w^{2h}_{k,j}, \quad 0 \leq j \leq N/2. \tag{2.59}$$

Por lo tanto, atendiendo a la ec. (2.59), una onda con número de onda k en Ω^h, seguirá siendo una onda de número de onda k en Ω^{2h}. Esto puede observarse en la Figura 2.10, donde se ha representado una onda w_6 tanto en una malla con $N = 24$ celdas, como en una con $N/2 = 12$ celdas.

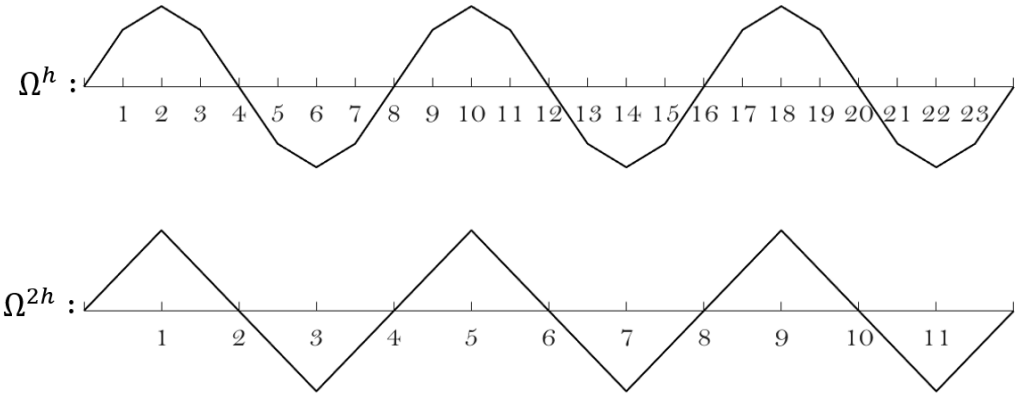

Figura 2.10: Onda con número de onda $k = 6$ en malla Ω^h, $N = 24$, proyectada en malla Ω^{2h}, $N = 12$.

Efectivamente, el número de onda se mantiene al proyectar una onda desde una malla Ω^h hasta otra malla Ω^{2h}. Lo que ocurre sin embargo es que, debido a que la malla Ω^{2h} poseen la mitad de nodos, esta onda se aprecia de mayor frecuencia que en la malla Ω^h, lo que resulta realmente interesante. Cabe destacar que esta característica ocurre para todas las ondas en Ω^h cuyo número de onda es $k \leq N/2$. Para valores $k > N/2$, las ondas se representan en Ω^{2h} como ondas con número de onda $k = (N - k)$ debido al efecto de alisamiento.

Lo anterior evidencia que, los errores de baja frecuencia en una malla fina Ω^h, se muestran como errores de alta frecuencia en una malla más gruesa Ω^{2h}. Esto es una de las principales propiedades para desarrollar el método multigrid ya que, cuando se itera en una malla Ω^h hasta tener errores de baja frecuencia que convergen lentamente, se puede restringir este error a una malla Ω^{2h}, donde los errores ahora serán de alta frecuencia, y se reducirán rápidamente. De nuevo, una vez que en la malla Ω^{2h} se alcancen errores de baja frecuencia, se puede pasar a una malla Ω^{4h} y seguir iterando en esa malla.

Sin embargo, el valor que se proyecta entre mallas no es la aproximación $\mathbf{v}$ a la solución o el error $\mathbf{e}$, sino que se restringe el residual $\mathbf{r}$ definido como:

$$\mathbf{r} \;=\; \mathbf{f} - A\mathbf{v} \;=\; A\mathbf{e}, \tag{2.60}$$

de modo que cuando $\mathbf{r} = 0$, $\mathbf{e} = 0$, y la aproximación $\mathbf{v} = \mathbf{u}$.

Este proceso es lo que se conoce como *correction scheme* o esquema de corrección. Su estructura es la siguiente:

- Iterar $A\mathbf{v} = \mathbf{f}$ en una malla Ω^h

- Obtener el residual $\mathbf{r} = A\mathbf{e}$
- Iterar el residual $A\mathbf{e} = \mathbf{r}$ en Ω^{2h} para obtener una aproximación del error $\mathbf{e}^{2h}$
- Corregir la aproximación obtenida en Ω^h con el error estimado en Ω^{2h}: $\mathbf{v}^h \leftarrow \mathbf{v}^h + \mathbf{e}^{2h}$

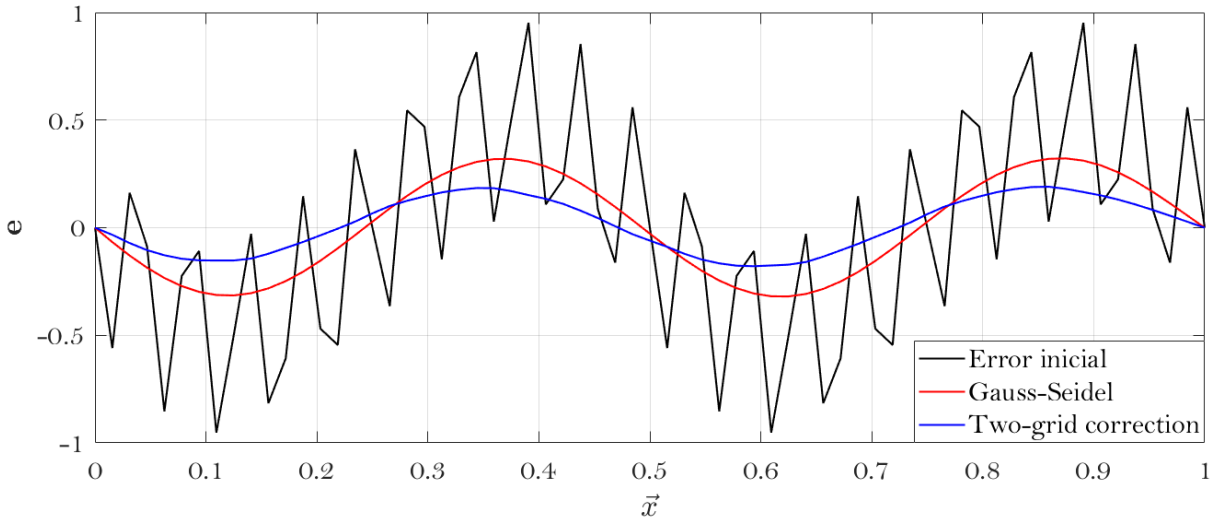

Figura 2.11: Reducción del error de aproximación en una malla de 64 celdas tras 12 iteraciones con Gauss-Seidel y tras un ciclo *two-grid correction* con tres iteraciones en el nivel fino y 6 en el grueso.

Si se utilizan dos mallas, como en el caso anterior, el esquema se conoce como *two-grid correction*. En la Figura 2.11 se puede observar la diferencia respecto del método de Gauss-Seidel. Se ha iterado el problema ec. (2.49a) con las mismas condiciones que con los métodos iterativos, es decir, $\mathbf{f} = 0$, y $N = 64$. La aproximación inicial se ha escogido $\mathbf{v} = (\mathbf{w}_4 + \mathbf{w}_{32})/2$. Se han realizado 12 iteraciones mediante Gauss-Seidel (rojo), mientras que se han hecho 6 iteraciones en Ω^h y 6 iteraciones en Ω^{2h} mediante *two-grid correction* (azul). Se puede observar cómo se ha reducido un 75 % la norma del error respecto del error inicial mediante *two-grid correction*, mientras que con Gauss-Seidel se ha reducido un 55 %.

2.4.3. Trasferencia entre mallas

Las ideas anteriormente enunciadas son la base del multigrid, tanto la idea proporcionada por *nested iteration* como por *two-grid correction*. Sin embargo queda por definir un tema importante y es cómo se hace la trasferencia entre mallas del residual o de la aproximación inicial. Se debe destacar que está prácticamente generalizado en el uso de multigrid la relación 2 entre celdas de mallas contiguas. Es decir, las mallas un paso más finas tienen el doble de celdas que la malla anterior.

Dentro de la trasferencia entre mallas, se deben diferenciar dos procesos. Por una parte, cuando se pasa de malla fina a malla gruesa, y por otra cuando se pasa de malla gruesa a malla fina.

Restricción de malla

El primero de los procesos es conocido como restricción de malla, y ocurre cuando se proyectan ciertos valores de una malla con N celdas, a otra malla con la mitad de celdas, $N/2$. Existen diferentes métodos para llevar a cabo la restricción de malla, siendo de ellos el más obvio el conocido como *injection*, en el cual el valor en los nodos de la malla Ω^{2h} reciben el valor de los nodos pares en la malla Ω^h, es decir:

$$v_j^{2h} \;=\; v_{2j}^h, \quad 0 \leq j \leq N/2. \tag{2.61}$$

Esta restricción de malla, a pesar de ser muy simple, es muy efectiva y puede ser realmente interesante en ciertos casos. Sin embargo, tiene algunos problemas. Por ejemplo, si el valor en Ω^h es de muy alta frecuencia, en Ω^{2h} tras haber realizado la restricción *injection* se vuelve de muy baja frecuencia. A pesar de ello, la idea es realizar la restricción de malla cuando el error en la malla Ω^h es de baja frecuencia, por lo que funciona correctamente.

Por otra parte, otro método más potente es conocido como *full weighting*, y los valores en la malla Ω^{2h} se obtienen mediante una ponderación de los valores en Ω^h de la forma:

$$v_j^{2h} \;=\; \frac{1}{4}\left(v_{j-1}^h + 2v_j^h + v_{j+1}^h\right), \quad 1 \leq j \leq N/2 - 1, \tag{2.62}$$

Interpolación de malla

El segundo de los procesos es la interpolación de malla, y ocurre cuando se desea pasar valores de la malla en Ω^{2h} a la malla Ω^h. Existen numerosos métodos para llevar a cabo la interpolación, sin embargo, uno de los más utilizados gracias a los resultados que ofrece, es también uno de los más sencillo y es la interpolación lineal. De esta forma, el valor en los nodos de la malla Ω^h se obtendría de la siguiente forma:

$$v_{2j}^h \;=\; v_j^{2h}, \tag{2.63a}$$

$$v_{2j+1}^h \;=\; \frac{1}{2}\left(v_j^{2h} + v_{j+1}^{2h}\right), \quad 0 \leq j \leq N/2 - 1. \tag{2.63b}$$

Una de las ventajas que tiene utilizar la interpolación lineal y *full-weighting* para restringir es que ambas matrices son traspuestas entre ellas.

2.4.4. Ciclos multigrid

Una vez que se tienen en mente todos los conceptos explicados durante la sección, es posible pasar a explicar los ciclos multigrid. Antes de comenzar, se va a realizar un apunte en la notación utilizada para facilitar la explicación. Se va a denominar $\mathbf{f}$ al lado derecho de la ecuación, independientemente de si es el vector inicial, o el vector residual, ya que no deja de ser el vector del lado derecho, mientras que se va a denominar $\mathbf{v}$ al vector que se va a iterar.

Ciclo V

El primero de los ciclos es el más sencillo de todos, aunque muy potente. Supóngase el esquema *two-grid correction*. En lugar de iterar en la malla Ω^{2h} e interpolar a continuación a la malla Ω^h, se restringe el valor obtenido hasta la malla Ω^{4h}, y así sucesivamente. Supóngase la existencia de $l > 1$ mallas diferentes, con espaciado entre mallas $h,\ 2h,\ 4h,\ ...,\ Lh = 2^{l-1}h$. El ciclo tendría la siguiente estructura:

- Iterar ν_1 veces en $A^h\mathbf{v}^h = \mathbf{f}^h$ con la aproximación inicial $\mathbf{v}^h$
- Calcular el residual $\mathbf{r}^h$ y restringirlo a $\rightarrow \mathbf{f}^{2h}$
  - Iterar ν_1 veces en $A^{2h}\mathbf{v}^{2h} = \mathbf{f}^{2h}$ con la aproximación inicial $\mathbf{v}^{2h} = \mathbf{0}$
  - Calcular el residual $\mathbf{r}^{2h}$ y restringirlo a $\rightarrow \mathbf{f}^{4h}$
    - Iterar ν_1 veces en $A^{4h}\mathbf{v}^{4h} = \mathbf{f}^{4h}$ con la aproximación inicial $\mathbf{v}^{4h} = \mathbf{0}$
    - Calcular el residual $\mathbf{r}^{4h}$ y restringirlo a $\rightarrow \mathbf{f}^{8h}$

      $\vdots$

      $\diamond$ Resolver $A^{Lh}\mathbf{v}^{Lh} = \mathbf{f}^{Lh}$

      $\vdots$
    - Interpolar $\mathbf{v}^{8h}$ y corregir $\mathbf{v}^{4h} \leftarrow \mathbf{v}^{4h} + \mathbf{v}^{8h}$
    - Iterar ν_2 veces en $A^{4h}\mathbf{v}^{4h} = \mathbf{f}^{4h}$ con la aproximación inicial $\mathbf{v}^{4h}$
  - Interpolar $\mathbf{v}^{4h}$ y corregir $\mathbf{v}^{2h} \leftarrow \mathbf{v}^{2h} + \mathbf{v}^{4h}$
  - Iterar ν_2 veces en $A^{2h}\mathbf{v}^{2h} = \mathbf{f}^{2h}$ con la aproximación inicial $\mathbf{v}^{2h}$
- Interpolar $\mathbf{v}^{2h}$ y corregir $\mathbf{v}^h \leftarrow \mathbf{v}^h + \mathbf{v}^{2h}$
- Iterar ν_2 veces en $A^h\mathbf{v}^h = \mathbf{f}^h$ con la aproximación inicial $\mathbf{v}^h$

Esta estructura es lo que se conoce como ciclo en V, debido a al forma que adopta en forma de diagrama (fig. 2.12). Cabe destacar que, normalmente, la malla Ω^{Lh} tiene un tamaño lo suficientemente pequeño como para resolver la ecuación mediante métodos directos, lo que disminuirá la cantidad de ciclos en V necesarios para alcanzar la precisión deseada. Por otra parte, gracias a lo evidenciado en la Figura 2.9, el valor de ν suele ser muy bajo, entre 3 y 10, ya que por encima de ese valor se estaría iterando errores de baja frecuencia y disminuyendo la eficacia del método.

El ciclo en V se puede escribir de forma recursiva, lo que resulta más intuitivo y compacto tanto visualmente como a la hora de realizar la programación.

1. Iterar ν_1 veces en $A^h\mathbf{v}^h = \mathbf{f}^h$ con solución inicial $\mathbf{v}^h$
2. Si Ω^h es la malla más gruesa, ir al punto 4. Si no:
   2.1. Calcular el residual $\mathbf{r}^h$ y restringirlo a $\rightarrow \mathbf{f}^{2h}$
   2.2. Realizar ciclo en V recursivo en Ω^{2h} con aproximación inicial $\mathbf{v}^{2h} = \mathbf{0}$
3. Interpolar $\mathbf{v}^{2h}$ y corregir $\mathbf{v}^h \leftarrow \mathbf{v}^h + \mathbf{v}^{2h}$

4. Iterar ν_2 veces en $A^h\mathbf{v}^h = \mathbf{f}^h$ con solución inicial $\mathbf{v}^h$

Como se ha comentado anteriormente, si la malla gruesa es suficientemente pequeña (pocas celdas), en lugar de realizar iteraciones se puede resolver directamente mediante métodos directos para mejorar la velocidad de convergencia.

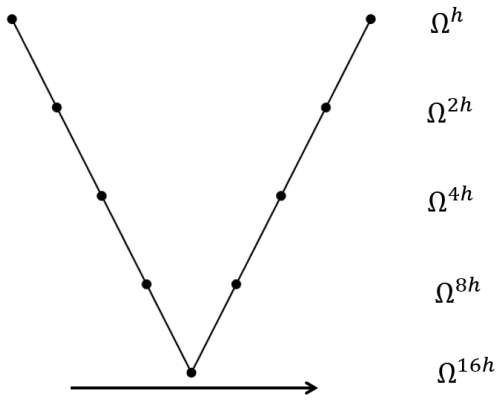

Figura 2.12: Estructura de mallas ciclo en V con 5 niveles de malla

Ciclo μ

La estructura recursiva de los ciclos μ tienen la siguiente estructura:

1. Iterar ν_1 veces en $A^h\mathbf{v}^h = \mathbf{f}^h$ con solución inicial $\mathbf{v}^h$
2. Si Ω^h es la malla más gruesa, ir al punto 4. Si no:
   2.1. Calcular el residual $\mathbf{r}^h$ y restringirlo a $\rightarrow \mathbf{f}^{2h}$
   2.2. Realizar ciclo μ recursivo en Ω^{2h} con aproximación inicial $\mathbf{v}^{2h} = \mathbf{0}$ μ veces
3. Interpolar $\mathbf{v}^{2h}$ y corregir $\mathbf{v}^h \leftarrow \mathbf{v}^h + \mathbf{v}^{2h}$
4. Iterar ν_2 veces en $A^h\mathbf{v}^h = \mathbf{f}^h$ con solución inicial $\mathbf{v}^h$

En la práctica, los ciclos μ sólo adoptan valores de $\mu = 1$ y $\mu = 2$. Realmente, el ciclo $\mu = 1$ corresponde con un ciclo en V, mientras que el ciclo $\mu = 2$ corresponde a un ciclo W, de nuevo por la forma que adopta en forma de diagrama (fig. 2.13).

FMG

Hasta ahora, tanto en el ciclo V como en los ciclos μ sólo se ha estado aplicando la idea de *correction scheme*, mientras que no se ha aplicado *nested iteration*. La integración de *nested iteration* con los ciclos, ya sean ciclos en V o ciclos μ, forman lo que se conoce como full-multigrid (FMG).

La idea del FMG es conseguir la mejor aproximación a la solución final posible antes de comenzar el ciclo en V en la malla Ω^h. Esto es lo que se puede observar en

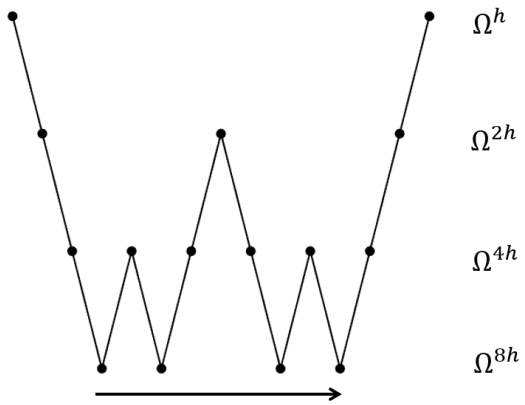

Figura 2.13: Estructura de mallas ciclo μ con $\mu = 2$ y 4 niveles de malla

la Figura 2.14. Para ello, se comienza en la malla más gruesa (Ω^{8h} en este caso), se realizan ν_0 iteraciones, normalmente 1 por experimentos previos [4], y se interpola esa solución a la siguiente malla (Ω^{4h}). En este momento se realiza un ciclo en V y, después, se interpola la solución a la siguiente malla, y así sucesivamente. Todo el trabajo previo a la realización del último ciclo en V no sólo es bastante barato computacionalmente, ya que son mallas muy pequeñas, sino que mejora mucho la convergencia en el ciclo V final.

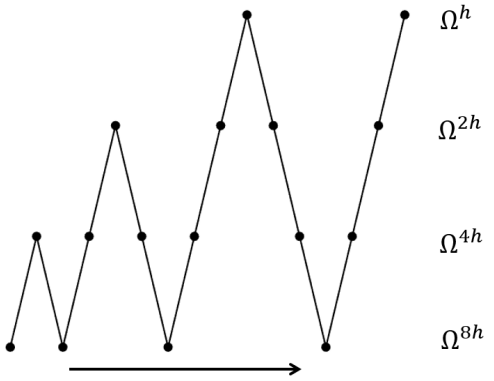

Figura 2.14: Estructura de mallas fmg con ciclo V y 4 niveles de malla

2.5. Resumen herramientas numéricas

En esta sección se va a resumir rápidamente cuáles van a ser las herramientas utilizadas para la resolución de las ecuaciones diferenciales.

- Discretización espacial: Diferencias finitas compactas
- Discretización temporal: Runge-Kutta tres subpasos
- Resolución de sistemas de ecuaciones: Multigrid con Gauss-Seidel para las iteraciones y descomposición LU para método directo

La elección de las diferencias finitas compactas reside en que se considera como un método semi-espectral, ya que ofrece una precisión similar a los métodos espectrales pero con la flexibilidad a la hora de colocar los nodos del resto de métodos. Respecto a los valores de J_1 y J_2 se han estudiará la mejor combinación entre precisión y tiempo de cálculo. Cabe destacar que los programas están preparados para trabajar con cualquier valor de J_1 o J_2.

Por otra parte, la elección de Runge-Kutta de tres subpasos reside en que ofrece un buen orden de error en la resolución de los pasos temporales, con un bajo coste de almacenamiento.

Finalmente, la elección del método multigrid se basa en que actualmente es el mejor método numérico para la resolución de grandes sistemas de ecuaciones. A su vez, el método de Gauss-Seidel ofrece mejores resultados que los métodos de Jacobi y, alguna de sus variaciones, como el Rojo-Negro, son más intuitivas a la hora de paralelizar el código.

Capítulo 3

Multigrid-CTFD 1d

En este capítulo se va a desarrollar el proceso seguido para la resolución de una ecuación diferencial en 1D, mediante multigrid y diferencias finitas compactas.

La ecuación diferencial a resolver es muy común en el campo de la mecánica de fluidos y es la siguiente:

$$-\Delta u(x) + \sigma u(x) = f(x), \quad 0 \leq x \leq 1, \ \sigma \geq 0, \tag{3.1a}$$

$$u(0) = \alpha \tag{3.1b}$$

$$u(1) = \beta \tag{3.1c}$$

donde Δ es el operador laplaciano, que en 1D $\Delta = \nabla^2 = \frac{\partial^2}{\partial x^2}$. Por otra parte, se debe discretizar el dominio de la ecuación para poder ser resuelta numéricamente.

3.1. Discretización del dominio 1D

En este caso se ha optado por una discretización del dominio no uniforme, teniendo más densidad de celdas en los extremos y menos en el centro, lo cuál es de vital importancia a la hora de resolver problemas de mecánica de fluidos. La ley de mallado utilizada es de tipo tangente hiperbólica, y sigue la siguiente fórmula:

$$x_i = \frac{x_N - x_0}{2} - \frac{L}{2}\frac{\tanh\left(\gamma\left(1 - \frac{2i}{N}\right)\right)}{\tanh(\gamma)}, \quad 0 \leq i \leq N, \tag{3.2}$$

donde L es la longitud del dominio, N el número de celdas del dominio y γ un parámetro de ajuste de densidad de nodos cerca de la frontera. Por ejemplo, con $N = 16$ celdas y $\gamma = 1{,}8$, el dominio discretizado mediante la ley ec. (3.2) se puede observar en la Figura 3.1.

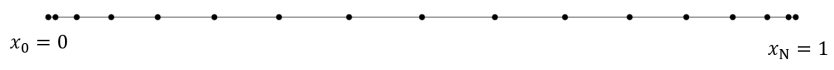

Figura 3.1: Dominio discretizado en el intervalo $0 \leq x \leq 1$ mediante una ley tipo tangente hiperbólica con $N = 16$ celdas, $N + 1 = 17$ nodos y $\gamma = 1{,}8$.

Una vez se dispone del dominio discretizado, se puede reescribir la ec. (3.1a) en forma discreta y compacta, como muestra la siguiente ecuación:

$$-\mathbf{u}_{xx} + \sigma\mathbf{u} = \mathbf{f}, \quad \sigma \geq 0, \tag{3.3a}$$

$$u_0 = \alpha \tag{3.3b}$$

$$u_1 = \beta \tag{3.3c}$$

donde $\mathbf{u}$ es el vector solución discreto, con $N+1$ valores, mientras que $\mathbf{f}$ es el vector del lado derecho discreto, con $N+1$ valores.

$$\mathbf{u} = \begin{pmatrix} u_0 \\ \vdots \\ u_i \\ \vdots \\ u_N \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} f_0 \\ \vdots \\ f_i \\ \vdots \\ f_N \end{pmatrix}$$

3.2. Diferencias finitas compactas 1D

Una vez se dispone de la ecuación discretizada, se debe aplicar el método CTFD para obtener el valor de la segunda derivada del vector solución en función del vector solución. Para ello, se obtienen las matrices A_{xx} y B_{xx} que permiten esta relación de la forma:

$$A_{xx}\mathbf{u}_{xx} = B_{xx}\mathbf{u} \tag{3.4}$$

donde las matrices A_{xx} y B_{xx} son matrices cuadradas de tamaño $(N+1)\times(N+1)$. Estas matrices son diagonales, con $2J_1+1$ y $2J_2+1$ diagonales respectivamente. Teniendo en cuenta esto, y que, de forma general, $N > \{J_1, J_2\}$, el porcentaje de celdas vacías (ceros) dentro de las matrices de CTFD es muy alto. En función del número de celdas y del valor de J, el porcentaje de llenado de las matrices se puede calcular a partir de:

$$\%\text{ llenado de la matriz} = \frac{(N+1) + \sum_{m=1}^{J} 2(N+1-m)}{(N+1)(N+1)} \times 100 \tag{3.5}$$

donde en la Tabla 3.1 se puede ver este valor con diferentes valores de N y J. Teniendo en cuenta que para la resolución mediante DNS se necesitan mallas extremadamente grandes, es decir, con un número de celdas muy grande, el porcentaje de llenado va a estar por debajo del 1 % en todo caso. Esto hace que trabajar con las matrices completas sea una pérdida de espacio en memoria.

Una de las ventajas que tiene trabajar con el método multigrid, y en general con métodos iterativos, es que permite no almacenar completamente las matrices del sistema, sino guardar en memoria únicamente los datos que sean necesarios. De esta forma, se tienen que crear algoritmos que permitan la manipulación de las matrices almacenadas únicamente sus diagonales.

Una vez obtenidas las matrices CTFD en forma diagonal, se premultiplica la ec. (3.3a) por la matriz A_{xx}, obteniéndose la siguiente ecuación:

$$-A_{xx}\mathbf{u}_{xx} + \sigma A_{xx}\mathbf{u} = A_{xx}\mathbf{f}, \quad \sigma \geq 0, \tag{3.6}$$

donde, aplicando la igualdad ec. (3.4) se llega a la siguiente expresión:

$$-B_{xx}\mathbf{u} + \sigma A_{xx}\mathbf{u} = (-B_{xx} + \sigma A_{xx})\mathbf{u} = A_{xx}\mathbf{f}, \quad \sigma \geq 0. \tag{3.7}$$

Si se descompone el vector solución $\mathbf{u}$ en condiciones de contorno e incógnitas de la forma:

$$\mathbf{u} \;=\; \mathbf{u}_x + \mathbf{u}_{cc} \rightarrow \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} 0 \\ u_1 \\ \vdots \\ u_{N-1} \\ 0 \end{pmatrix} + \begin{pmatrix} u_0 \\ 0 \\ \vdots \\ 0 \\ u_N \end{pmatrix}, \tag{3.8}$$

entonces la ec. (3.7) se puede reescribir de la forma:

$$-B_{xx}\left(\mathbf{u}_x + \mathbf{u}_{cc}\right) + \sigma A_{xx}\left(\mathbf{u}_x + \mathbf{u}_{cc}\right) \;=\; A_{xx}\mathbf{f}, \tag{3.9}$$

y, separando a un lado de la ecuación las incógnitas y a otro lado de la ecuación las condiciones de contorno, se obtiene:

$$\left(-B_{xx} + \sigma A_{xx}\right)\mathbf{u}_x \;=\; A_{xx}\mathbf{f} + \left(B_{xx} - \sigma A_{xx}\right)\mathbf{u}_{cc}. \tag{3.10}$$

N	**J=1**	**J=2**	**J=3**
4	52,00 %	76,00 %	92,00 %
8	30,86 %	48,15 %	62,96 %
16	16,96 %	27,34 %	37,02 %
32	8,91 %	14,60 %	20,11 %
64	4,57 %	7,55 %	10,49 %
128	2,31 %	3,84 %	5,35 %
256	1,16 %	1,94 %	2,71 %
512	0,58 %	0,97 %	1,36 %
1024	0,29 %	0,49 %	0,68 %
2048	0,15 %	0,24 %	0,34 %

Tabla 3.1: Porcentaje de llenado de las matrices CTFD en función del número de celdas y del número de nodos utilizados a cada lado del nodo de cálculo

Se ha obtenido por lo tanto un sistema de ecuaciones de la forma:

$$G\mathbf{u} = \mathbf{H}, \tag{3.11}$$

donde la matriz G y el vector $\mathbf{H}$ en este caso corresponden a:

$$G \;=\; \left(-B_{xx} + \sigma A_{xx}\right), \tag{3.12a}$$

$$\mathbf{H} \;=\; A_{xx}\mathbf{f} + (B_{xx} - \sigma A_{xx})\,\mathbf{u}_{cc}. \tag{3.12b}$$

Una vez eliminadas las ecuaciones correspondientes a las condiciones de contorno, la matriz G tiene un tamaño $(N-1)\times(N-1)$, mientras que los vectores incógnita $\mathbf{u}$ y el vector del lado derecho $\mathbf{H}$ tienen un tamaño $N-1$. Sin embargo, se debe recordar que la matriz G se almacena en forma diagonal, de tal forma, el tamaño de la matriz será en este caso $\max\{J_1, J_2\}\times(N-1)$.

3.3. Proceso iterativo 1D

Una vez se dispone de un sistema de ecuaciones en la forma ec. (3.11), se puede iniciar el proceso de solución del mismo, en este caso, mediante el método multigrid. Para ello, al ser un método iterativo, se debe establecer una tolerancia tal que, una vez superada, el método deje de computarse. Conceptualmente se seguiría el siguiente esquema:

```
do while ( error>tolerancia )
  vnew  = ciclomultigrid(vold)
  error = Error(vnew-vold)
  vold  = vnew
enddo
```

En este caso, *ciclomultigrid*(*vold*) corresponde a realizar un ciclo multigrid, ya sea ciclo en V, W o FMG, sobre la aproximación inicial v_{old}, mientras que Error() corresponde a una función que calcule la norma, cualquiera que se haya definido, del vector error.

Para poder comenzar con el proceso iterativo, se debe definir primero una aproximación inicial al vector $\mathbf{v}$ solución. Normalmente, en problemas de mecánica de fluidos, no se conoce la solución al problema, pero si se conocen aproximaciones a la misma, por lo que comenzar con una de estas aproximaciones es lo recomendable. En los casos donde no se conozca ninguna aproximación a la solución, se puede, o bien comenzar con aproximación inicial nula, es decir, cero, o bien una función lineal entre las condiciones de contorno.

El algoritmo recursivo (*ciclomultigrid*(*vold*) en el esquema superior) para la resolución del sistema sigue los siguientes pasos:

1. Iteración del sistema de ecuaciones
2. Restricción del residual
3. Comprobación de malla
4. Interpolación del error
5. Iteración del sistema de ecuaciones

1. Iteración del sistema de ecuaciones

En primer lugar se debe realizar la iteración del sistema de ecuaciones para la malla actual. Si es la malla más fina, Ω^h, la aproximación inicial $\mathbf{v}$ a utilizar será, si es el primer ciclo, la definida al comienzo del programa mientras que, si no, será la aproximación obtenida en el ciclo anterior. En caso de no ser la malla más fina, la aproximación inicial siempre será el vector $\mathbf{0}$. Tal y como se explicó en la sección Multigrid, el número de iteraciones a realizar, ν_1 es muy bajo, entre 3 a 10 iteraciones, ya que el objetivo principal es eliminar el error de alta frecuencia (fig. 2.9).

El algoritmo de resolución utiliza el método de Gauss-Seidel, sin embargo, requiere ciertas modificaciones para trabajar con las matrices del sistema en forma diagonal. El código en lenguaje fortran sería algo similar a:

```
do i=1,iteraciones
    do j=-D(1)+1,-D(1)+N-1
        temp = H(j-D(1))-G(1,j+D(1))*v(j+D(1))
        do k=2,nd
           if (D(k) /= 0) temp = temp-G(k,j+D(1))*v(j+D(k))
        enddo
        v(j) = temp/G(nd/2+1,j+D(1))
    enddo
enddo
```

donde la el vector $\mathbf{D}$ es un vector de nd posiciones, siendo nd el número de diagonales de la matriz G. El vector $\mathbf{D}$ contiene la posición relativa de las diagonales respecto de la diagonal principal. Por otra parte, los vectores $\mathbf{H}$ y $\mathbf{v}$, para poder generalizar el código para varios valores de J_1 y J_2, tienen una cantidad de ceros al comienzo y al final del vector es función de $\max\{J_1, J_2\}$.

2. Restricción del residual

Una vez se dispone de la aproximación en la malla Ω^h, se debe calcular el residual en dicha malla para poder interpolarlo a una malla el doble de gruesa Ω^{2h}. Recordando, el vector residual se calcula como:

$$\mathbf{r}^h = \mathbf{H}^h - G^h\mathbf{v}^h \tag{3.13}$$

por lo que el primer paso es calcular el producto $G\mathbf{v}$. El código que realiza la multiplicación de una matriz diagonal por un vector es el siguiente:

```
do i=-D(1)+1,-D(1)+N-1
    temp = G(1,i+D(1))*v(i+D(1))
    do j=2,nd
        temp = temp+G(j,i+D(1))*v(i+D(j))
    enddo
    v(i+D(1)) = temp
enddo
```

Una vez se ha calculado la multiplicación, la obtención del residual se consigue mediante una resta punto a punto con el vector $\mathbf{H}$. Una vez se dispone del residual,

se debe restringir a la malla Ω^{2h}. Tal y como se indicó en el anterior capítulo, existen varios métodos para llevar a cabo este paso. El escogido es el que se conoce como *full weighting*. El algoritmo para llevarlo a cabo es el siguiente:

```
do i = 1,N/2-1
        H2h(i) = 0.25d0*(rh(2*i-1)+rh(2*i+1))+0.5d0*rh(2*i)
enddo
```

donde el vector **H2h** corresponde al lado derecho del nuevo sistema de ecuaciones en la malla Ω^{2h} y tiene un tamaño $N/2-1$, mientras que **rh** es el residual en la malla Ω^h.

Por lo tanto, ahora en la nueva malla ya se dispone del sistema de ecuaciones:

$$G^{2h}\mathbf{v}^{2h} = \mathbf{H}^{2h} \tag{3.14}$$

donde $\mathbf{H}^{2h}$ corresponde al residual restringido de la malla Ω^h, $\mathbf{v}$ corresponde al nuevo vector incógnita, y G^{2h} contiene el valor de las diagonales de las matrices CTFD para la malla Ω^{2h}.

3. Comprobación de malla

Antes de seguir con el proceso iterativo, se debe comprobar si la malla a calcular Ω^{2h} es la más gruesa del proceso o no. En caso de ser la malla más gruesa se debe resolver $\mathbf{v}^{2h}$. Existen dos opciones, o bien se resuelve iterando, volviendo al punto 1. del proceso, o bien se resuelve por métodos directos. La norma general es que la malla más gruesa del proceso es lo suficientemente pequeña, celdas muy grandes, sistema de ecuaciones pequeño, como para poder ser resuelta por métodos directos.

En el caso que nos atañe, siempre se han calculado mallas suficientemente pequeñas como para ser resueltas mediante procesos directos, en concretos mediante la descomposición LU de matrices. En caso de no ser la malla más gruesa, se vuelve al punto 1. y se continua con el proceso.

4. Interpolación del error

Llegados a este punto ya se dispone de una aproximación de $\mathbf{v}^{2h}$ que, recordando, corresponde con el error de la aproximación de $\mathbf{v}^h$, aunque por facilidad de notación los vectores reciban la misma identificación. Por lo tanto, el objetivo es interpolar el valor de $\mathbf{v}^{2h}$ desde la malla Ω^{2h} hasta la malla $\Omega 2h$ para poder corregir el valor de $\mathbf{v}^h$.

El método de interpolación utilizado, y uno de los más sencillos, es la interpolación lineal, que es la operación traspuesta de la restricción *full weighting*. El código empleado para ello es el siguiente:

```
do i=2,N-1,2
    vvec(i  ) = v2h(i/2)
    vvec(i+1) = v2h(i/2)
enddo
```

```
do i=1,N-1
    eh(i) = 0.5d0*(vvec(i)+vvec(i+1))
enddo
```

donde **vvec** es un vector auxiliar de tamaño $N-1$, $\mathbf{v}^{2h}$ es el vector solución de la malla Ω^{2h} de tamaño $N/2-1$, y $\mathbf{e}^h$ es el error de aproximación en la malla Ω^h de tamaño $N-1$.

Por lo tanto, ahora se debe corregir la aproximación $\mathbf{v}^h$ de la forma:

$$\mathbf{v}^h = \mathbf{v}^h + \mathbf{e}^h \tag{3.15}$$

Una vez corregida la aproximación $\mathbf{v}^h$, se debe volver a iterar la el sistema.

5. Iteración del sistema de ecuaciones

Una vez se dispone de la aproximación corregida, se debe volver a iterar la solución mediante el algoritmo descrito en el punto 1. del proceso iterativo. En este caso, en lugar de realizar ν_1 iteraciones, se realizan ν_2 iteraciones. Este valor suele ser el mismo para ambas variables, sin embargo, en ciertos casos un mayor valor del término ν_2 puede derivar en una reducción del número de ciclos totales del proceso.

En función del nivel de malla que se esté calculando, existen dos opciones. Si el es la malla más fina, se sale del algoritmo, se calcula el error respecto el valor antiguo de **v** y se evalúa si es menor que la tolerancia. En caso afirmativo, el programa termina y se analizan los resultados. En caso negativo, se vuelve a iniciar el proceso en el punto 1. Por otra parte, si no es la malla más fina, entonces se vuelve al punto 4. para interpolar la solución a la malla un paso más fina.

En este momento ya se dispone de un programa funcional para la resolución de ecuaciones diferenciales ordinarias en 1D. En el Capítulo 6 se verificará la eficacia del programa desarrollado, mientras que en el siguiente capítulo se detallarán los pasos para evolucionar el código de 1D a 2D.

Capítulo 4

Multigrid-CTFD 2d

En este capítulo se va a desarrollar el proceso seguido para la resolución de la misma ecuación diferencial (ec. (3.1a)) que en 1D, pero en 2D (ec. (4.1a)), igualmente mediante diferencias finitas compactas y multigrid. Teniendo en cuenta que la mayor parte de código es la misma entre ambos problemas, la mayor complicación reside en adaptar dicho código a 2D.

$$-\Delta u(x,y) + \sigma u(x,y) = f(x,y), \quad 0 \leq x \leq 1,\ 0 \leq y \leq 1,\ \sigma \geq 0, \tag{4.1a}$$

$$u(0,y) = f_1(y), \quad u(1,y) = f_2(y) \tag{4.1b}$$

$$u(x,0) = f_3(x), \quad u(x,1) = f_4(x) \tag{4.1c}$$

En este caso, el operador Δ en 2D es $\Delta = \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$.

4.1. Discretización del dominio 2D

Al igual que en el caso 1D, se opta por una discretización del dominio no uniforme, con mayor densidad de nodos en zonas cercanas a las fronteras, donde una mayor densidad de nodos permite captar mejor los efectos cercanos a las paredes. En este caso, la ley de mallado es de tipo tangente hiperbólica y viene dada por la siguiente fórmula:

$$x_i = \frac{x_N - x_0}{2} - \frac{L_x}{2}\frac{\tanh\left(\gamma\left(1 - \frac{2i}{N}\right)\right)}{\tanh(\gamma)}, \quad 0 \leq i \leq N, \tag{4.2a}$$

$$y_j = \frac{y_N - y_0}{2} - \frac{L_y}{2}\frac{\tanh\left(\gamma\left(1 - \frac{2j}{M}\right)\right)}{\tanh(\gamma)}, \quad 0 \leq j \leq M, \tag{4.2b}$$

donde L_k es la longitud del dominio en la dirección k, N el número de celdas en el eje X, M el número de celdas en el eje Y, γ un parámetro de ajuste denominado parámetro de estiramiento. Cuánto mayor es el valor de gamma, más se estiran las celdas del centro del dominio, acumulando mayor densidad de nodos por tanto cerca de los extremos del mismo. Aplicando la leyes descritas en ambas direcciones del dominio, con un valor de $\gamma = 1{,}1$, $N = 16$ y $M = 8$, el dominio discretizado se puede observar en la Figura 4.1.

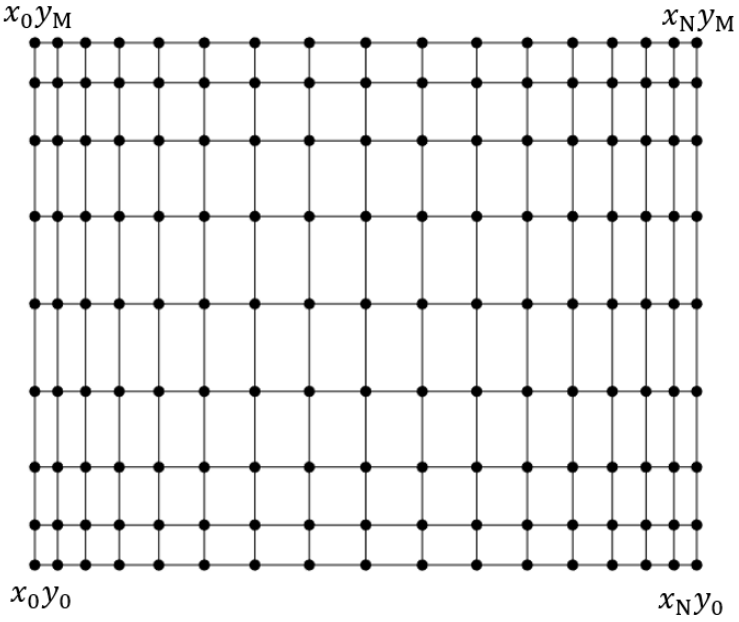

Figura 4.1: Dominio discretizado en el intervalo $0 \leq x \leq 2$, $0 \leq y \leq 2$ mediante una ley tipo tangente hiperbólica no equiespaciada con $N = 16$ celdas en eje X y $M = 8$ celdas en eje Y.

Con el dominio disretizado, la ec. (4.1a) se puede reescribir de forma discreta y compacta como muestra la siguiente ecuación:

$$-(\mathbf{u}_{xx} + \mathbf{u}_{yy}) + \sigma\mathbf{u} \;=\; \mathbf{f}, \quad \sigma \geq 0, \tag{4.3}$$

donde $\mathbf{u}$ es la matriz solución discreta, de tamaño $(N+1) \times (M+1)$, mientras que $\mathbf{f}$ es la matriz del lado derecho discreta de tamaño $(N+1) \times (M+1)$.

$$\mathbf{u} \;=\; \begin{pmatrix} u_{0,0} & u_{0,1} & \dots & u_{0,M} \\ u_{1,0} & u_{1,1} & \dots & u_{1,M} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N,0} & u_{N,1} & \dots & u_{N,M} \end{pmatrix}, \; \mathbf{f} \;=\; \begin{pmatrix} f_{0,0} & f_{0,1} & \dots & f_{0,M} \\ f_{1,0} & f_{1,1} & \dots & f_{1,M} \\ \vdots & \vdots & \ddots & \vdots \\ f_{N,0} & f_{N,1} & \dots & f_{N,M} \end{pmatrix}. \tag{4.4}$$

4.2. Diferencias finitas compactas 2D

Una vez ya se dispone de las ecuaciones discretizadas (ec. (4.3)) se puede aplicar CTFD para linealizar el sistema de ecuaciones. Debido a las propiedades del operador laplaciano, el problema queda desacoplado de las dos dimensiones del campo inicial, pudiendo trasformar el problema en dos problemas cuasi-independientes. De esta forma, el método CTFD se aplica en 1D en ambas dimensiones de la forma:

$$A_{xx}\mathbf{u}_{xx} \;=\; B_{xx}\mathbf{u} \tag{4.5a}$$

$$A_{yy}\mathbf{u}_{yy} \;=\; B_{yy}\mathbf{u} \tag{4.5b}$$

donde las matrices A_{xx} y B_{xx} tiene un tamaño $(N+1)\times(N+1)$, mientras que las matrices A_{yy} y B_{yy} tiene un tamaño $(M+1)\times(M+1)$. De nuevo, estas matrices, al igual que en el caso 1D, son matrices diagonales, A_{xx} y A_{yy} con $(2J_1+1)$ y B_{xx} y B_{yy} con $(2J_2+1)$ diagonales respectivamente. Normalmente, para ambas dimensiones se utilizan los mismos valores de J_1 y J_2, ya que ofrece mejores resultados.

Si se premultiplica la ec. (4.3) por la matriz A_{xx} se llega a:

$$-A_{xx}\mathbf{u}_{xx}-A_{xx}\mathbf{u}_{yy}+\sigma A_{xx}\mathbf{u} \;=\; A_{xx}\mathbf{f},\quad \sigma\geq 0, \tag{4.6}$$

donde, aplicando la igualdad ec. (4.5a) se llega a la siguiente expresión:

$$-B_{xx}\mathbf{u}-A_{xx}\mathbf{u}_{yy}+\sigma A_{xx}\mathbf{u} \;=\; A_{xx}\mathbf{f},\quad \sigma\geq 0, \tag{4.7}$$

Ahora el objetivo sería deshacerse del término $\mathbf{u}_{yy}$ mediante la aplicación de la igualdad ec. (4.5b). Sin embargo, a pesar de que su aplicación no es directa, es posible aplicando las propiedades de multiplicación de matrices y teniendo en cuenta la definición de las matrices de diferencias finitas compactas. Sabiendo que:

$$\mathbf{u}_{yy}A'_{yy}=\mathbf{u}B'_{yy}, \tag{4.8}$$

entonces es posible postmultiplicar la ec. (4.7) por la matriz A_{yy} traspuesta, obteniendo:

$$-B_{xx}\mathbf{u}A'_{yy}-A_{xx}\mathbf{u}_{yy}A'_{yy}+\sigma A_{xx}\mathbf{u}A'_{yy} \;=\; A_{xx}\mathbf{f}A'_{yy},\quad \sigma\geq 0. \tag{4.9}$$

Ahora sí, aplicando la igualdad ec. (4.8) se llega a:

$$-B_{xx}\mathbf{u}A'_{yy}-A_{xx}\mathbf{u}B'_{yy}+\sigma A_{xx}\mathbf{u}A'_{yy} \;=\; A_{xx}\mathbf{f}A'_{yy},\quad \sigma\geq 0. \tag{4.10}$$

La ec. (4.10) constituye el sistema de ecuaciones a resolver una vez aplicada la discretización mediante diferencias finitas compactas. Sin embargo, este sistema, tal y como está escrito, es inoperable desde el punto de vista numérico y debe ser transformado en un sistema del tipo:

$$G\mathbf{u} \;=\; \mathbf{H}, \tag{4.11}$$

donde $\mathbf{u}$ y $\mathbf{H}$ son vectores columna, mientras que G es una matriz cuadrada. La obtención del vector columna $\mathbf{H}$ a partir de la ec. (4.10) es directa. En primer lugar se deben realizar las multiplicaciones del lado derecho de la ecuación, obteniendo la matriz $\mathbf{h}$:

$$\mathbf{h} \;=\; A_{xx}\mathbf{f}A'_{yy} \tag{4.12}$$

Una vez obtenida la matriz $\mathbf{h}$, de tamaño $(N+1)\times(M+1)$, ésta se puede reordenar en forma de vector columna para obtener el vector $\mathbf{H}$. Esta reordenación se realiza de columna en columna, como se muestra en la ec. (4.13). Realizar dicha ordenación por columnas en lugar de por filas no es por casualidad, sino que obtiene ciertas ventajas a la hora de almacenar la matriz memoria en Fortran.

$$\mathbf{H} = \begin{pmatrix} h_{0,0} \\ h_{1,0} \\ \vdots \\ h_{N,0} \\ h_{0,1} \\ h_{1,1} \\ \vdots \\ h_{N,1} \\ h_{0,2} \\ \vdots \\ h_{N,M} \end{pmatrix} \tag{4.13}$$

La obtención de la matriz G, sin embargo, no es ni trivial ni directa, sino que requiere realizar un cierto análisis. Supónganse tres matrices aleatorias, A, B, C, de tamaño $N \times N$, $N \times M$ y $M \times M$ respectivamente. La multiplicación $A \times B$ viene dada por la siguiente fórmula:

$$(AB)_{i,j} = \sum_{k=1}^{N} A_{i,k} B_{k,j}, \quad 1 \leq i \leq N,\ 1 \leq j \leq M. \tag{4.14}$$

Si ahora este resultado se multiplica por la tercera matriz C, la matriz resultante viene dada por:

$$(ABC)_{i,j} = \sum_{h=1}^{M} AB_{i,h} C_{h,j}, \quad 1 \leq i \leq N,\ 1 \leq j \leq N. \tag{4.15}$$

Si se desarrolla la anterior función por ejemplo para el valor $i = 1$ y $j = 1$ se obtendría un resultado del tipo:

$$(ABC)_{1,1} = A_{1,1} C_{1,1} B_{1,1} + A_{1,1} C_{2,1} B_{1,2} + \ldots \tag{4.16}$$

donde, si se reescribe teniendo en cuenta que B es la matriz incógnita, se llega al siguiente resultado:

$$(ABC)_{1,1} = \alpha_1 B_{1,1} + \alpha_2 B_{1,2} \ldots \tag{4.17}$$

Si se sigue el mismo procedimiento para todos los valores de i y j se llega a una matriz de tamaño $N \times M$ donde, en cada posición de la matriz, se tienen expresiones como ec. (4.17). Esta matriz ABC se puede reordenar de igual forma que la matriz $\mathbf{h}$ en ec. (4.13) y separar los valores incógnita en un vector separado. Si ahora se denomina a la matriz ABC como $A_{xx}\mathbf{u}B'_{yy}$, el objetivo es crear un algoritmo que nos permita calcular los valores de α en ec. (4.17) y poder reescribir dicha multiplicación como $G_1\mathbf{u}$.

Dentro del algoritmo se debe tener en cuenta que las matrices de CTFD son diagonales, por lo que para ahorrar tiempo de cálculo sólo se deben calcular la multiplicación de los valores no nulos. El algoritmo debe calcular la siguiente fórmula:

$$G(k,h) \;=\; \sum_{n=\xi_h^1}^{\xi_h^2} \left(\sum_{m=\xi_k^1}^{\xi_k^2} A_{xx}(h,m)u(m,n)B'_{yy}(n,h) \right), \quad 1 \le k \le N,\ 1 \le h \le M, \tag{4.18}$$

donde los valores superiores e inferiores de los sumatorios son función del número de celdas de la malla, del número de diagonales de la matriz y del nodo de cálculo en cuestión. Se pueden obtener a partir del siguiente código:

```
if ( i < nd+1 ) then
     limit(1) = na-i-nd+1
     limit(2) = na
elseif ( i >= nd+1 .and. i+nd+1 <= N ) then
     limit(1) = 1
     limit(2) = na
elseif ( i+nd+1 > N ) then
     limit(1) = 1
     limit(2) = na+N-nd-i
endif
```

donde nd corresponde con el número de nodos utilizados a cada lado del nodo de cálculo, na es el número de diagonales de la matriz, y N el número de celdas en la dirección de la matriz en cuestión. Por otra parte, el código para la obtención de la matriz G_k es el siguiente:

```
do j = 0,M
  do i = 0,M
    call sumlimits(limit1,j,nb,M)
    do h = limit1(1),limit1(2)
      call sumlimits(limit2,i,na,N)
      do k = limit2(1),limit2(2)
        G(h*na-na+k,(j-1)*N+i)=A(k,i)*B(nb-h+1,j-nb+h+nb/2)
      enddo
    enddo
  enddo
enddo
```

donde nb y na son los nodos utilizados para el cálculo de A y B, N y M son las celdas de los ejes X e Y respectivamente y sumlimits es la función que calcula los límites del sumatorio.

A partir del desarrollo para la obtención de la matriz G, la ec. (4.10) se puede transformar en:

$$G_1\mathbf{u} - G_2\mathbf{u} - G_3\mathbf{u} \;=\; G\mathbf{u} \;=\; \mathbf{H} \tag{4.19}$$

donde las matrices G_1, G_2 y G_3 se obtienen de la aplicación del algoritmo a las matrices $A_{xx}A'_{yy}$, $B_{xx}A'_{yy}$ y $A_{xx}B'_{yy}$ respectivamente. El tamaño de la matriz G del problema es de $(N+1)(M+1)\times(N+1)(M+1)$, mientras que los vectores $\mathbf{u}$ y $\mathbf{H}$ son

de tamaño $(N+1)(M+1)$. La matriz G del sistema es una matriz dispersa, ordenada en bloques de diagonales. Al igual que ocurría en 1D, el porcentaje de llenado es muy bajo. Para este problema con $N = 16$, $M = 8$, $J_1 = 3$ y $J_2 = 5$, la matriz G antes de aplicar las condiciones de contorno tiene 23409 posiciones, mientras que el número de valores no nulos es de 2661, lo que deja un porcentaje de llenado de 11.36 %. Su distribución puede verse en la Figura 4.2.

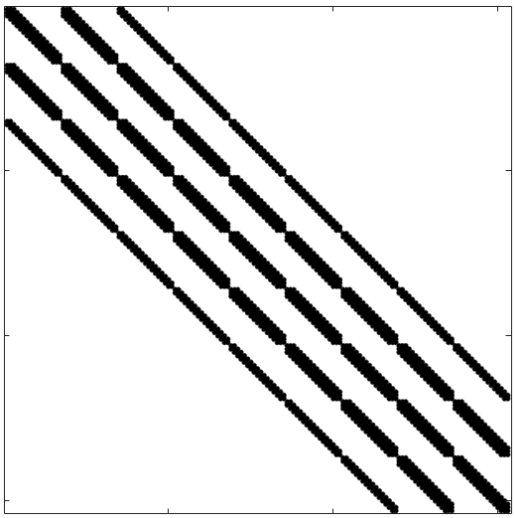

Figura 4.2: Distribución de valores no nulos a lo largo de la matriz G del sistema en 2D, con $N = 16$, $M = 8$, $J_1 = 3$ y $J_2 = 5$.

Ahora por lo tanto queda aplicar las condiciones de contorno antes de poder comenzar con el proceso iterativo para la resolución del problema. La descomposición de la matriz solución $\mathbf{u}$ en este caso viene dada por:

$$\mathbf{u} \;=\; \mathbf{u}_{inc} + \mathbf{u}_{cc} \tag{4.20a}$$

$$\mathbf{u}_{inc} \;=\; \begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ 0 & u_{1,1} & \dots & u_{1,M-1} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & u_{N-1,1} & \dots & u_{N-1,M-1} & 0 \\ 0 & 0 & \dots & 0 & 0 \end{pmatrix} \tag{4.20b}$$

$$\mathbf{u}_{cc} \;=\; \begin{pmatrix} u_{0,0} & u_{0,1} & \dots & u_{0,M-1} & u_{0,M} \\ u_{1,0} & 0 & \dots & 0 & u_{1,M} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ u_{N-1,0} & 0 & \dots & 0 & u_{0,M} \\ u_{N,0} & u_{N,1} & \dots & u_{N,M-1} & u_{N,M} \end{pmatrix} \tag{4.20c}$$

Estas matrices deben transformarse a vector columna, de la misma forma que $\mathbf{h}$ en la ec. (4.13). Aplicadas a la ec. (4.19) se obtiene:

$$G(\mathbf{u}_{inc} + \mathbf{u}_{cc}) \;=\; \mathbf{H} \rightarrow G\mathbf{u}_{inc} \;=\; \mathbf{H} - G\mathbf{u}_{cc}. \tag{4.21}$$

En 1D, a partir de obtener la ecuación anterior, el proceder es muy fácil, pues únicamente se eliminan la primera y última ecuación del sistema de ecuaciones correspondientes a las condiciones de contorno, y se puede comenzar con el proceso iterativo. Sin embargo, el problema en 2D se complica, pues las condiciones de contorno están mezcladas en mitad del vector. Por lo tanto, eliminar las ecuaciones correspondientes a las condiciones de contorno en 2D no es un proceso trivial, especialmente eliminar las filas y columnas correspondientes en la matriz G, pues está almacenada en forma diagonal. La obtención del algoritmo que permita eliminar estos valores de la matriz G es una de las principales dificultades encontradas a la hora de adaptar el código desde 1D a 2D.

En este momento, ya se dispone de todas las herramientas necesarias para comenzar con el algoritmo de iteración.

4.3. Proceso de iterativo 2D

El proceso para la realización del algoritmo de iteración es exactamente el mismo que para 1D. Tan solo existen dos diferencias de aplicación y estas residen en la restricción e interpolación de malla.

Restricción del residual

Para el caso 2D, al igual que ocurre en 1D, existen diferentes métodos para la restricción de valores desde una malla Ω^h hasta una malla con la mitad de celdas Ω^{2h}. La adaptación del método *full weighting* de 1D a 2D es trivial, simplemente se deben añadir más términos. En este caso, cada nodo de cálculo en Ω^{2h} está rodeado por 8 nodos en Ω^h más el propio nodo en dicha malla, en total 9 nodos. Los valores de ponderación de cada nodo en Ω^h sobre el nodo de cálculo en Ω^{2h} pueden verse en la Figura 4.3.

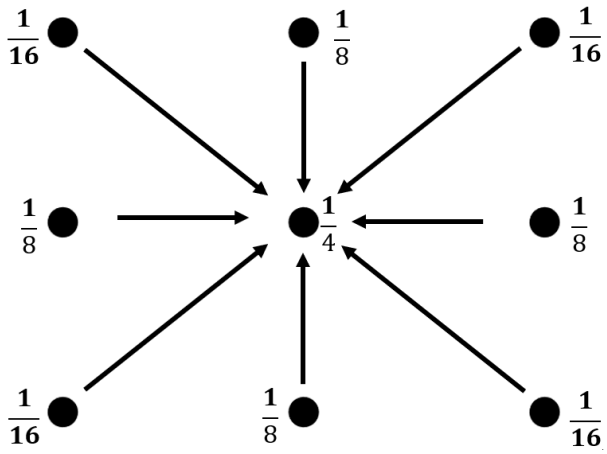

Figura 4.3: Ponderación de los valores al restringir desde una malla Ω^h hasta Ω^{2h}.

El código en Fortran empleado para la restricción de valores desde Ω^h hasta Ω^{2h} es el siguiente:

```
do j = 1,M/2-1
  do i = 1,N/2-1
    H2h(i+(j-1)*(N/2-1)) =
    1/16*(rh(2*(j-1)*(N-1)+2*i-1)+rh(2*(j-1)*(N-1)+2*i+1)
          +rh(2*(j-1)*(N-1)+2*i-1+2*(N-1))
          +rh(2*(j-1)*(N-1)+2*i+1+2*(N-1)))+
    1/8 *(rh(2*(j-1)*(N-1)+2*i  )+rh(2*(j-1)*(N-1)+2*i-1+N-1)
                +rh(2*(j-1)*(N-1)+2*i+1+N-1)
                +rh(2*(j-1)*(N-1)+2*i+2*(N-1)))+
    1/4 *(res(2*(j-1)*(N-1)+2*i+N-1))
  enddo
enddo
```

Interpolación del error

De nuevo, al igual que en la restricción de malla, la adaptación del código desde 1D a 2D para la interpolación lineal es trivial, añadiendo más términos. El la ponderación del valor el error en los nodos de la malla Ω^{2h} sobre el valor en nodos de la malla Ω^h se puede observar en la Figura 4.4. En este caso los nodos de la malla Ω^{2h} aparecen representados en color rojo, mientras que los nodos de la malla Ω^h están representados en color negro.

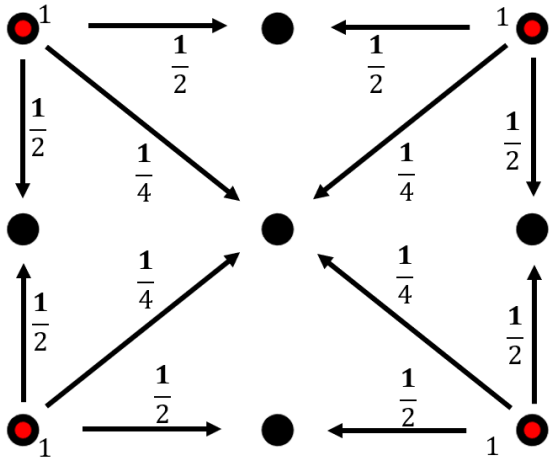

Figura 4.4: Ponderación de los valores al interpolar desde una malla Ω^{2h} (rojo) hasta Ω^h (negro).

Para poder realizar la interpolación, primero se debe calcular un vector de apoyo, *vvec*, mediante el siguiente bucle:

```
do j=2,M/2
  do i=2,N/2
     vvec((j-1)*(N/2+1)+i) = v2h((j-2)*(N/2-1)+i-1)
  enddo
enddo
```

Por otra parte, a raíz de la Figura 4.4 se pueden distinguir 4 tipos de relación entre nodos de diferentes mallas diferentes. En primer lugar, los nodos coincidentes entre mallas:

```
do jj=2,M-1,2
   do ii=1,N/2-1
      eh(2*i+(j-1)*(N-1)) = vvec(i+1+j/2*(N/2+1))
   enddo
enddo
```

Por otra parte, los nodos de Ω^h que se encuentran rodeados entre cuatro nodos de Ω^{2h}:

```
do jj=1,M,2
   do ii=1,N/2-1
      eh(2*i-1+(j-1)*(N-1)) = 0.25*(vvec(i+  (j-1)/2*(N/2+1))
      +vvec(i+1+(j-1)/2*(N/2+1))+vvec(i+  (j+1)/2*(N/2+1))
      +vvec(i+1+(j+1)/2*(N/2+1)))
   enddo
enddo
```

Los nodos de Ω^h que se encuentran horizontalmente entre dos nodos de Ω^{2h}:

```
do jj=1,M,2
   do ii=1,N/2-1
      eh(2*i-1+(j-1)*(N-1)) = 0.5*(vvec(i+1+(j-1)/2*(N/2+1))
      +vvec(i+1+(j+1)/2*(N/2+1)))
   enddo
enddo
```

Y, finalmente, los nodos de Ω^h que se encuentran verticalmente entre dos nodos de Ω^{2h}:

```
do jj=2,M-1,2
   do ii=1,N/2-1
      eh(2*i-1+(j-1)*(N-1)) = 0.5*(vvec(i+j/2*(N+1))
      +vvec(i+1+j/2*(N+1)))
   enddo
enddo
```

Con todo lo anterior, ya se dispone de un programa funcional para calcular ecuaciones diferenciales en 2D sin dependencia del tiempo. En el Capítulo 6 se verificará tanto la precisión como la eficacia del programa desarrollado, mientras que en el siguiente capítulo se introducirá la dependencia del tiempo en la derivada y se aplicará el método de Runge-Kutta de tres subpasos.

Capítulo 5

Ecuación del calor

En este capítulo se va a explicar la metodología seguida para la resolución de la ecuación del calor (ec. (5.1)) en un dominio bidimensional. El motivo de la elección de esta ecuación es que la mayoría de los problemas de la mecánica de fluidos se basan en resolver un problema similar a la ecuación del calor, tal y como se ha visto en ec. (2.3). Desarrollada la base para la resolución de esta ecuación, las modificaciones del programa para adaptarlo a otro tipo de problemas son mínimas.

$$\frac{\partial u(x,y,t)}{\partial t} - \sigma \Delta u(x,y,t) \;=\; f(x,y,t), \quad \sigma \geq 0, \tag{5.1}$$

La diferencia de resolución entre la ecuación del calor y la ecuación ec. (4.1a) es la inclusión del término temporal. En el caso de ec. (4.1a) se debe resolver un único sistema de ecuaciones una sola vez, sin embargo, en este caso, se debe resolver el sistema de ecuaciones tantas veces como pasos temporales se disponga. Por lo tanto, además de la discretización espacial, también se debe realizar una discretización temporal del dominio.

5.1. Discretización espacial del dominio Ec. calor

La discretización espacial del problema es idéntica a la utilizada en Sección 4.1, donde se disponen los nodos de forma que la densidad cerca de las fronteras sea mayor que en el centro del dominio. La ley utilizada en este caso es la misma que la mostrada en ec. (4.2a) y ec. (4.2b) para discretizar ambas direcciones, ya que gracias al parámetro de ajsute γ permite mucha libertad a la hora de agrupar más o menos nodos en las fronteras.

La ecuación del calor discretizada espacialmente se puede reescribir de la forma:

$$\mathbf{u}_t \;=\; \mathrm{L}\left(\mathbf{u}\right) + \mathrm{N}\left(\mathbf{u}\right), \tag{5.2}$$

donde los operadores L y N reúnen las partes lineales y no lineales de la ecuación de forma:

$$\mathrm{L}\left(\mathbf{u}\right) \;=\; \sigma\left(\mathbf{u}_{xx} + \mathbf{u}_{yy}\right) \tag{5.3a}$$

$$\mathrm{N}\left(\mathbf{u}\right) \;=\; \mathbf{f} \tag{5.3b}$$

y las matrices se agrupan de igual forma que en ec. (4.4).

5.2. Discretización temporal Ec. calor

Para la discretización temporal en este caso se divide el dominio temporal continuo en K intervalos de tiempo equidistantes:

$$t_i \;=\; t_0 + i\Delta t, \quad 0 \leq i \leq K. \tag{5.4}$$

De esta forma se consigue un dominio tridimensional, donde dos de las dimensiones son espaciales y una es temporal. Para avanzar desde el tiempo t hasta el tiempo $t+\Delta t$ se utiliza el método de Runge-Kutta de tres subspasos y bajo almacenamiento explicado en Sección 2.3. Aplicado a la ecuación presentada anteriormente, el esquema quedaría de la siguiente forma:

$$\mathbf{u}' \;=\; \mathbf{u}^n + \Delta t\left(\alpha_1\sigma\left(\mathbf{u}^n_{xx}+\mathbf{u}^n_{yy}\right)+\beta_1\sigma\left(\mathbf{u}'_{xx}+\mathbf{u}'_{yy}\right)+\gamma_1\mathbf{f}^n\right) \tag{5.5a}$$

$$\mathbf{u}'' \;=\; \mathbf{u}' + \Delta t\left(\alpha_1\sigma\left(\mathbf{u}'_{xx}+\mathbf{u}'_{yy}\right)+\beta_1\sigma\left(\mathbf{u}''_{xx}+\mathbf{u}''_{yy}\right)+\gamma_1\mathbf{f}'+\zeta_1\mathbf{f}^n\right) \tag{5.5b}$$

$$\mathbf{u}^{n+1} \;=\; \mathbf{u}'' + \Delta t\left(\alpha_1\sigma\left(\mathbf{u}''_{xx}+\mathbf{u}''_{yy}\right)+\beta_1\sigma\left(\mathbf{u}^{n+1}_{xx}+\mathbf{u}^{n+1}_{yy}\right)+\gamma_1\mathbf{f}''+\zeta_1\mathbf{f}'\right) \tag{5.5c}$$

donde el valor de las variables α, β, γ y ζ viene dado por ec. (2.46). El objetivo por lo tanto es resolver muchas veces un sistema ecuaciones del tipo:

$$\mathbf{u}^{n+1} \;=\; \mathbf{u}^n + \Delta t\left(\alpha_i\sigma\left(\mathbf{u}^n_{xx}+\mathbf{u}^n_{yy}\right)+\beta_i\sigma\left(\mathbf{u}^{n+1}_{xx}+\mathbf{u}^{n+1}_{yy}\right)+\gamma_1\mathbf{f}^n+\zeta_1\mathbf{f}^{n-1}\right) \tag{5.6}$$

5.3. Diferencias finitas compactas Ec. calor

Una vez se dispone de la ecuación discretizada (ec. (5.6)), la aplicación de CTFD para la obtención de un sistema de ecuaciones que se pueda resolver numéricamente es análoga a la seguida en la Sección 4.2. Premultiplicando la ec. (5.6) por la matriz A_{xx} se obtiene:

$$\begin{aligned} A_{xx}\mathbf{u}^{n+1} \;=\; & A_{xx}\mathbf{u}^n + \Delta t\alpha_i\sigma\left(A_{xx}\mathbf{u}^n_{xx}+A_{xx}\mathbf{u}^n_{yy}\right)+ \\ & \Delta t\beta_i\sigma\left(A_{xx}\mathbf{u}^{n+1}_{xx}+A_{xx}\mathbf{u}^{n+1}_{yy}\right)+\Delta t\gamma_i A_{xx}\mathbf{f}^n+\Delta t\zeta_i A_{xx}\mathbf{f}^{n-1} \end{aligned} \tag{5.7}$$

Aplicando ahora ec. (4.5a) y postmultiplicando por A'_{yy} se llega a :

$$\begin{aligned} A_{xx}\mathbf{u}^{n+1}A'_{yy} \;=\; & A_{xx}\mathbf{u}^nA'_{yy}+\Delta t\alpha_i\sigma\left(B_{xx}\mathbf{u}^nA'_{yy}+A_{xx}\mathbf{u}^n_{yy}A'_{yy}\right)+ \\ & \Delta t\beta_i\sigma\left(B_{xx}\mathbf{u}^{n+1}A'_{yy}+A_{xx}\mathbf{u}^{n+1}_{yy}A'_{yy}\right)+ \\ & \Delta t\gamma_i A_{xx}\mathbf{f}^nA'_{yy}+\Delta t\zeta_i A_{xx}\mathbf{f}^{n-1}A'_{yy} \end{aligned} \tag{5.8}$$

Si ahora se aplica ec. (4.8) y se agrupa en el lado izquierdo de la ecuación los términos en tiempo t^{n+1}, mientras que en el lado derecho se agrupan el resto de términos, la ecuación queda:

$$A_{xx}\mathbf{u}^{n+1}A'_{yy} - \Delta t\beta_i\sigma\left(B_{xx}\mathbf{u}^{n+1}A'_{yy} + A_{xx}\mathbf{u}^{n+1}B'_{yy}\right) = A_{xx}\mathbf{u}^{n}A'_{yy} + \Delta t\alpha_i\sigma\left(B_{xx}\mathbf{u}^{n}A'_{yy} + A_{xx}\mathbf{u}^{n}A'_{yy}\right) + \Delta t\gamma_i A_{xx}\mathbf{f}^{n}A'_{yy} + \Delta t\zeta_i A_{xx}\mathbf{f}^{n-1}A'_{yy} \quad (5.9)$$

donde la única incógnita es el la matriz $\mathbf{u}^{n+1}$. Por lo tanto, el lado derecho de la ecuación se puede calcular directamente, obteniendo como resultado una matriz $\mathbf{h}$ de tamaño $(N+1)\times(M+1)$. De nuevo, igual que en la sección anterior, esta matriz se debe reordenar en vector columna según ec. (4.13) para obtener el vector del lado derecho H. Por otra parte, si se utilizan las técnicas de obtención de las matrices G explicadas en la Sección 2.3, se puede reescribir el sistema de la siguiente manera:

$$\left(G_1 - \Delta t\beta_i\sigma\left(G_2 + G_3\right)\right)\mathbf{u}^{n+1} = H \quad (5.10)$$

donde las matrices G_1, G_2 y G_3 se obtienen de la aplicación del algoritmo a las matrices $A_{xx}A'_{yy}$, $B_{xx}A'_{yy}$ y $A_{xx}B'_{yy}$ respectivamente. Ahora ya se dispone de un sistema de ecuaciones donde se pueden aplicar métodos numéricos de resolución para obtener la solución del mismo.

5.4. Proceso iterativo Ec. calor

El algoritmo de resolución del sistema de ecuaciones es idéntico al utilizado para la resolución de ecuaciones en derivadas parciales sin inclusión del término temporal. Sin embargo, la única diferencia reside en que este algoritmo de resolución se debe aplicar tantas veces como pasos temporales se disponga. Además, en cada paso temporal, se deben recalcular los valores del vector de lado derecho, lo que aumenta el tiempo de resolución del problema.

La estructura de resolución es la siguiente:

```
while t < tend
  do rkst = 1,3
    t = t + dt/3
    H = updateRHS(rkst, t)
    while error > tolerancia
      vnew  = ciclomultigrid(vold)
      error = Error(vnew-vold)
      vold  = vnew
    end
  end
end
```

donde updateRHS es una función que recalcula el valor del vector del lado derecho **H**, ciclomultigrid(vold) corresponde a la aplicación de un ciclo multigrid, ya sea ciclo V, W o FMG, sobre el sistema ec. (5.10) y Error es una función que calcula la norma vectorial, cualquiera que se haya definido, del vector error $vnew - vold$.

En este punto ya se dispone de un programa funcional para la resolución de ecuaciones diferenciales en derivadas parciales con términos temporales en 2D. En el siguiente capítulo se pondrá a prueba tanto la eficacia como la precisión del método desarrollado mediante la resolución de diferentes ejemplos.

Cabe destacar que el programa desarrollado es totalmente flexible a la hora de modificar los diferentes parámetros. Sin necesidad de alterar los algoritmos de cálculo, se puede modificar el inicio y fin del mallado, el número de celdas en ambas dimensiones, el inicio y final del tiempo de cálculo, el paso del tiempo de cálculo, los nodos de cálculo de las matrices de diferencias finitas compactas, tanto de la dimensión x como de la dimensión y, y el problema a resolver. Esto resulta muy útil y beneficioso tanto a la hora de estudiar diferentes problemas como para que en un futuro se adapte el código a otros propósitos.

Capítulo 6

Resultados

Una vez explicados los pasos seguidos hasta alcanzar el algoritmo final, es momento de valorar la precisión del método efectuado. Para ello, se evaluarán diferentes funciones en 1D, en 2D y en la ecuación del calor.

6.1. Multigrid-CTFD 1D

La ecuación a resolver es la expuesta en el Capítulo 3 con condiciones de contorno tipo Dirichlet:

$$-\Delta u(x) + \sigma u(x) = f(x), \sigma \geq 0, \tag{6.1a}$$

$$u(x_0) = \alpha \tag{6.1b}$$

$$u(x_N) = \beta \tag{6.1c}$$

donde el operador Δ representa el laplaciano de la función $\frac{\partial^2}{\partial x^2}$. Las condiciones de resolución de la ecuación son:

- Constante del problema $\sigma = 1$
- Dominio de resolución: $x \in \{-1, 1\}$
- Mallado del dominio tipo tangente hiperbólica (ec. (3.2))
- Multigrid: Iteraciones con Gauss-Seidel $\rightarrow \ \nu_1 = 3, \ \nu_2 = 3$
- Multigrid: Ciclo V

Por otra parte, se realizará la resolución del sistema tanto con diferente tamaños de malla como diferentes valores de J_1 y J_2, tanto en Matlab como en Fortran, para poder comparar ambos compiladores. Además, la malla más gruesa del método múltigrid siempre será la misma, siendo en este caso $N_{min} = 8$.

En este caso se va a resolver la ecuación ec. (6.1a) aplicando el siguiente valor a la función $f(x)$:

$$f(x) = 4\mathrm{e}^x \left(\cos(2x-1) - \sin(2x-1)\right) \tag{6.2}$$

Para poder evaluar la precisión del método una vez obtenida la solución numérica, se debe comparar con la solución analítica de ec. (6.1a). El valor de esta función solución es:

$$u(x) \;=\; \mathrm{e}^{x}\cos(2x-1) \tag{6.3}$$

por lo que las condiciones de contorno son:

$$u_0 \;=\; -0{,}36419, \quad u_N \;=\; 1{,}46869 \tag{6.4}$$

A continuación se muestran los diferentes resultados obtenidos:

$\mathbf{N_{max}}$	$\|\|e\|\|_\infty$	**Tiempo (s)**	**Ciclos V**	**t/c (s)**	**Δt/c**
16	8,93E-05	4,40E-05	7	6,29E-06	-
32	5,72E-06	9,30E-05	8	1,16E-05	1,85
64	3,59E-07	1,41E-04	8	1,76E-05	1,52
128	2,24E-08	2,67E-04	8	3,33E-05	1,89
256	1,40E-09	4,35E-04	8	5,44E-05	1,63
512	8,77E-11	7,85E-04	8	9,81E-05	1,80
1024	5,25E-12	1,65E-03	8	2,06E-04	2,10
2048	9,49E-13	2,87E-03	8	3,59E-04	1,74
4096	6,87E-12	5,58E-03	8	6,98E-04	1,94
8192	1,55E-11	1,36E-02	10	1,36E-03	1,95
16384	2,86E-11	5,02E-02	19	2,64E-03	1,95
32768	8,78E-11	4,37E-02	8	5,47E-03	2,07
65536	4,11E-10	1,07E-01	10	1,07E-02	1,95

Tabla 6.1: Resultados de la ecuación $-\Delta u(x) + \sigma u(x) \;=\; f(x)$ resuelta mediante multigrid-CTFD 1d en Fortran. J_1=1 y J_2=1

$\mathbf{N_{max}}$	$\|\|e\|\|_\infty$	**Tiempo (s)**	**Ciclos V**	**t/c (s)**	**Δt/c**
16	8,92E-05	3,93E-03	6	6,55E-04	-
32	5,72E-06	8,63E-03	7	1,23E-03	1,88
64	3,59E-07	1,18E-02	7	1,69E-03	1,37
128	2,24E-08	2,41E-02	7	3,44E-03	2,03
256	1,40E-09	5,20E-02	7	7,42E-03	2,16
512	8,70E-11	1,45E-01	7	2,07E-02	2,78
1024	5,05E-12	5,29E-01	7	7,56E-02	3,66
2048	1,89E-12	2,53	7	3,61E-01	4,78
4096	1,79E-12	10,66	8	1,33E+00	3,69
8192	2,17E-11	43,00	7	6,14E+00	4,61
16384	3,88E-11	202,00	9	2,24E+01	3,65

Tabla 6.2: Resultados de la ecuación $-\Delta u(x) + \sigma u(x) \;=\; f(x)$ resuelta mediante multigrid-CTFD 1d en Matlab. J_1=1 y J_2=1

En las tablas 6.1 y 6.2 se pueden observar los resultados obtenidos de calcular ec. (6.1a) tanto en fortran como en Matlab con valores de $J_1 = 1$ y $J_2 = 1$, es decir,

3 nodos de cálculo tanto para la derivada como para la función en las matrices de diferencias finitas compactas. El error en norma infinita se define como:

$$||e||_\infty \;=\; \max(|v_i - u_i|), \quad 0 \leq i \leq N \tag{6.5}$$

donde u es la solución analítica del problema y v la solución numérica. Por otra parte, la quinta columna representa el tiempo de cálculo por cada ciclo en V, mientras que la sexta columna representa el incremento de tiempo de cálculo por ciclo al aumentar el número de celdas.

Observando ambas tablas, rápidamente se pueden sacar varias conclusiones. Por una parte, el cálculo en Fortran resulta mucho más rápido que en Matlab. Para pocas celdas, hasta $N = 256$, Fortran es 2 órdenes de magnitud más rápido que Matlab para calcular ciclos en V, mientras que a medida que aumenta el número de celdas, esta diferencia también crece. Esto desemboca en que el cálculo en Matlab a partir de cierto número de celdas se vuelve excesivamente lento. Esto ocurre ya que el compilador de Matlab está optimizado para trabajar con operaciones vectoriales o matriciales, mientras que Fortran funciona mucho mejor en operaciones punto a punto, como las que se realizan en este tipo de cálculos.

Debido a lo anterior, el incremento de tiempo en Matlab al doblar el número de celdas no es consistente, mientras que en Fortran al doblar el número de celdas, también se dobla el tiempo de cálculo por ciclo, lo cual permite estimar tiempos de cálculo para mallas más grandes.

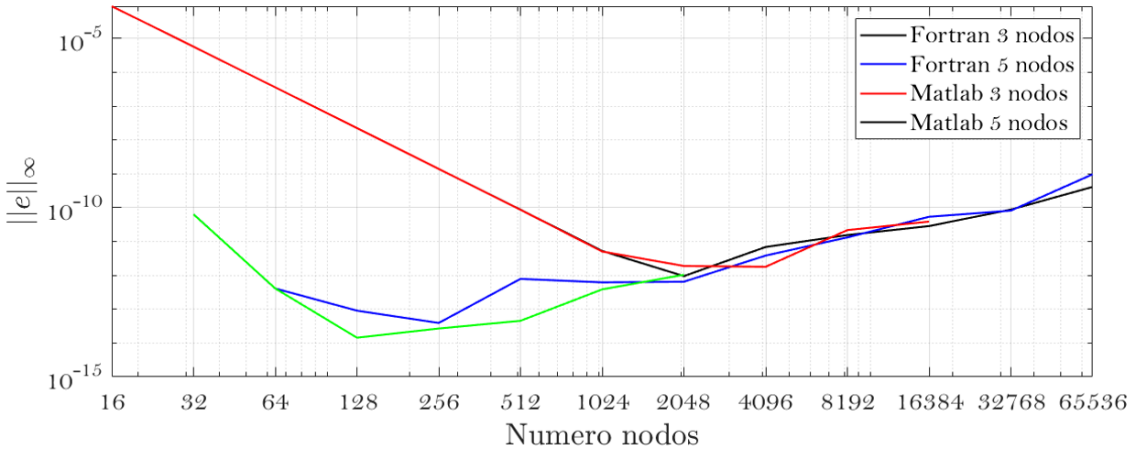

Figura 6.1: Error en norma infinita obtenido mediante multigrid 1D con Fortran y Matlab y con diferentes nodos de cálculo

En lo que respecta al uso de diferencias finitas compactas, se puede observar muy bien sus beneficios. En la Sección 2.2 se comentó que el orden de error era en torno a $\mathcal{O} = 2(J_1 + J_2)$, y observando los resultados, es posible observar cómo esto se cumple. Con apenas 256 celdas ya se está consiguiendo una precisión del orden de 10^{-9}, mientras que con diferencias finitas centradas de segundo orden se necesitarían del orden de 10^4 celdas para el mismo nivel de precisión.

Por otra parte, se observa que al aumentar el número de celdas, disminuye el error de truncamiento de las diferencias finitas compactas hasta un cierto punto, donde aumentar el número de celdas es contraproducente debido a un aumento del error de redondeo provocado por un mal condicionamiento de las matrices. De la solución a este problema se hablará en el capítulo siguiente.

Finalmente, cabe destacar la velocidad con la que el programa es capaz de resolver en Fortran sistemas de grandes dimensiones, resolviendo en pocos segundos con gran precisión.

En las tablas 6.3 y 6.4 se tienen los mismos resultados que el caso anterior, pero con valores de $J_1 = 2$ y $J_2 = 2$, es decir, con 5 nodos de cálculo de la derivada y de la función en las matrices de diferencias finitas compactas.

$\mathbf{N_{max}}$	$\|\|\mathbf{e}\|\|_\infty$	**Tiempo (s)**	**Ciclos**	**t/c (s)**	**Δt/c**
32	6,39E-11	8,54E-03	1111	7,69E-06	-
64	4,10E-13	3,40E-03	245	1,39E-05	1,81
128	9,08E-14	5,13E-03	201	2,55E-05	1,84
256	3,93E-14	9,90E-03	210	4,71E-05	1,85
512	7,86E-13	1,61E-02	177	9,07E-05	1,92
1024	6,22E-13	3,64E-02	202	1,80E-04	1,99
2048	6,49E-13	1,11E+00	3230	3,44E-04	1,91
4096	3,83E-12	1,11E-01	163	6,80E-04	1,98
8192	1,32E-11	4,19E-01	311	1,35E-03	1,98
16384	5,39E-11	4,20E-01	155	2,71E-03	2,01
32768	8,19E-11	9,17E-01	168	5,46E-03	2,01
65536	9,67E-10	1,42E+00	133	1,07E-02	1,96

Tabla 6.3: Resultados de la ecuación $-\Delta u(x) + \sigma u(x) = f(x)$ resuelta mediante multigrid-CTFD 1d en Fortran. J_1=2 y J_2=2

$\mathbf{N_{max}}$	$\|\|\mathbf{e}\|\|_\infty$	**Tiempo (s)**	**Ciclos**	**t/c (s)**	**Δt/c**
32	6,36E-11	6,16E-01	1783	3,46E-04	-
64	4,11E-13	2,03E-01	292	6,94E-04	2,01
128	1,44E-14	4,63E-01	253	1,83E-03	2,64
256	2,69E-14	1,06E+00	206	5,13E-03	2,81
512	4,55E-14	3,75E+00	195	1,92E-02	3,74
1024	3,86E-13	1,77E+01	227	7,78E-02	4,05
2048	1,04E-12	7,64E+01	192	3,98E-01	5,11

Tabla 6.4: Resultados de la ecuación $-\Delta u(x) + \sigma u(x) = f(x)$ resuelta mediante multigrid-CTFD 1d en Matlab. J_1=2 y J_2=2

La primera conclusión que se puede obtener es la gran cantidad de ciclos necesarios para alcanzar la convergencia del proceso en comparación al caso calculado con J_1=1 y J_2=1. Esto se debe a que la matriz del sistema obtenida, para uno o varios de los tamaños de malla, tiene valores propios cercanos a la unidad, lo que

ralentiza la convergencia del proceso. Para poder mejorar este comportamiento, se pueden incluir métodos de sobre-relajación para acelerar la convergencia.

En lo que respecta a los tiempos por ciclo, éstos son idénticos o muy parecidos a los obtenidos con J_1=1 y J_2=1, ya que apenas suman operaciones al proceso. Además, al igual que ocurre en el caso anterior, al doblar el número de celdas, también se dobla el tiempo de cálculo por ciclo, lo que es coherente, pues se realizan el doble de operaciones. En Matlab, donde los tiempos por ciclos son superiores, ya no es posible calcular grandes sistemas de ecuaciones, pues el aumento en el número de ciclos necesarios resulta en un tiempo total de cálculo muy elevado.

Finalmente, en lo que respecta a la precisión, se puede observar que ésta es mayor que la obtenida en el caso anterior, lo que concuerda con la teoría. Al conseguir un método con un orden de error muy bajo con pocos nodos, los errores por redondeo del ordenador entran en juego de forma más temprana y ya en 256/512 celdas se puede observar su influencia. De nuevo, ésto sólo es evitable mediante una modificación de las matrices de cálculo a través de un precondicionador.

Por otra parte y para ver resultados gráficos, el valor de la función analítica ec. (6.3) se puede observar en la Figura 6.2. Además, en la Figura 6.3 se puede observar el error obtenido en la solución del problema planteado con 128 celdas, es decir, 129 nodos de cálculo, con $J_1 = 1$ y $J_2 = 1$, y resuelto con Fortran. El error relativo se define como:

$$\varepsilon_R \;=\; \left|\frac{u_i - v_i}{u_i}\right|, \tag{6.6}$$

mientras que el error absoluto se define como:

$$\varepsilon_A \;=\; |u_i - v_i|\,. \tag{6.7}$$

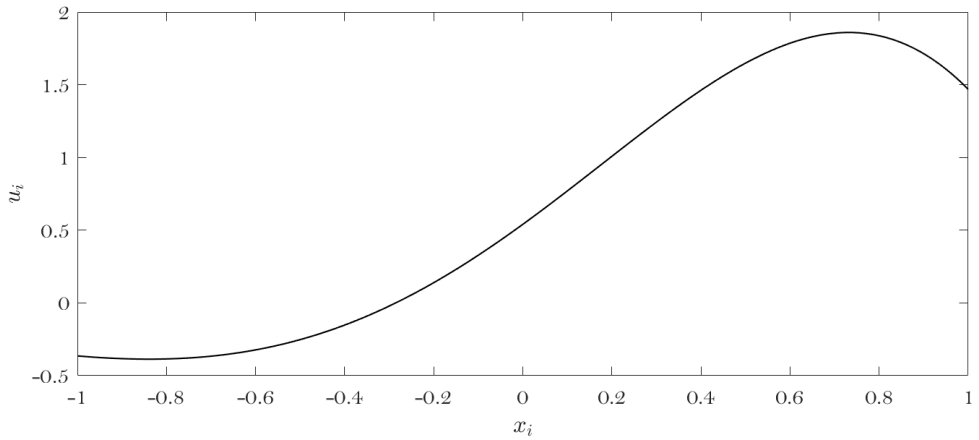

Figura 6.2: Valor de la función $e^x\cos(2x-1)$ (eje Y) en función de los nodos del dominio x_i (eje X)

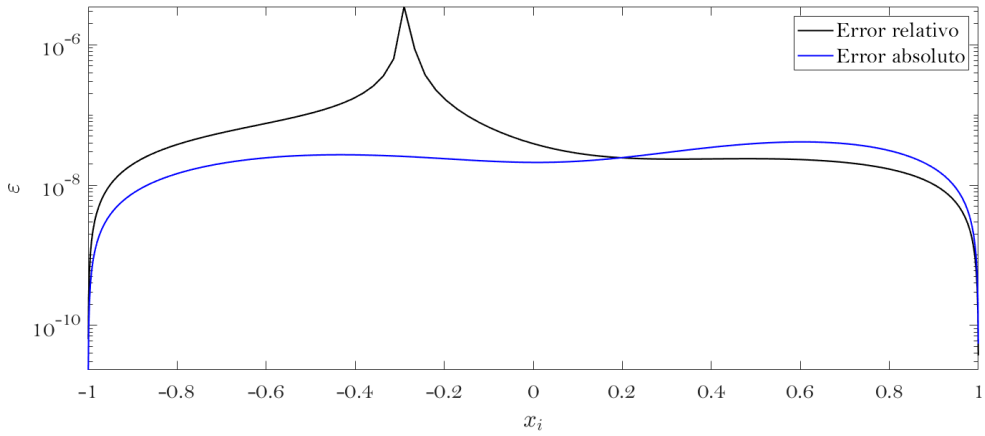

Figura 6.3: Error absoluto y relativo de la solución v_i tras resolver el sistema 6.1a con función solución 6.3 mediante multigrid y CTFD con $N = 128$.

6.2. Multigrid-CTFD 2D

En el caso de 2D, se va a resolver la ecuación propuesta en el Capítulo 4 con condiciones de contorno tipo Dirichlet:

$$-\Delta u(x,y) + \sigma u(x,y) = f(x,y), \quad \sigma \geq 0, \tag{6.8a}$$

$$u(x_0,y) = u_1(y), \quad u(x_N,y) = u_2(y) \tag{6.8b}$$

$$u(x,y_0) = u_3(x), \quad u(x,y_M) = u_4(x) \tag{6.8c}$$

donde en este caso, el operador Δ en 2D es $\Delta = \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. Las condiciones de resolución de la ecuación son:

- Constante del problema $\sigma = 1$
- Dominio de resolución: $x \in \{-1, 1\}$, $y \in \{-1, 1\}$
- Mallado del dominio tipo tangente hiperbólica (ec. (4.2a) y ec. (4.2b))
- Multigrid: Iteraciones con Gauss-Seidel $\rightarrow$ $\nu_1 = 3$, $\nu_2 = 3$
- Multigrid: Ciclo V

De nuevo, se utilizarán diferentes tamaños de malla así como valores de J_1 y J_2. Se debe destacar que las mallas utilizadas son cuadradas, sin embargo, el programa admite la introducción de diferentes valores de nodos para ambas direcciones, además, los resultados son extrapolables a otro tipo de mallados.

El valor de la función $f(x,y)$ en la ecuación ec. (6.8a) es el siguiente:

$$f(x,y) = 21\sin(2x)\cos(4y) \tag{6.9}$$

La solución analítica a la ec. (6.8a) es necesaria para poder evaluar la precisión del método. El valor de la misma es el siguiente:

$$u(x,y) \;=\; \sin(2x)\cos(4y) \tag{6.10}$$

por lo que las condiciones de contorno son:

$$\begin{aligned} u_{-1,y} \;&=\; -0{,}902\cos(4y),\; u_{1,y} \;=\; 0{,}902\cos(4y) \\ u_{x,-1} \;&=\; -0{,}653\sin(2x),\; u_{x,1} \;=\; 0{,}653\sin(2x) \end{aligned} \tag{6.11}$$

En las tablas 6.5, 6.6 y 6.7 se muestran los resultados obtenidos en Fortran con valores de J_1 y J_2 de (1,1), (1,2) y (2,2) respectivamente. En este caso no se muestran resultados obtenidos con Matlab debido a que las conclusiones que se podían obtener son análogas al caso 1D, donde Matlab es más lento realizando las iteraciones y para un cierto número de punto el tiempo de cálculo era excesivo. En la primera columna de las tablas abajo mostradas aparece el número de nodos totales del dominio teniendo en cuenta que las mallas son cuadradas.

Nodos totales	**N y M**	$\|\|e\|\|_\infty$	**Tiempo (s)**	**Ciclos**	**t/c (s)**	**Δt/c**
256	16	6,21E-04	9,46E-04	9	1,05E-04	-
1024	32	3,89E-05	2,01E-03	10	2,01E-04	1,91
4096	64	2,44E-06	6,13E-03	10	6,13E-04	3,05
16384	128	1,53E-07	2,38E-02	10	2,38E-03	3,88
65536	256	9,57E-09	1,09E-01	11	9,86E-03	4,15
262144	512	5,98E-10	4,68E-01	11	4,25E-02	4,31
1048576	1024	3,73E-11	2,04	11	1,85E-01	4,36
4194304	2048	2,71E-12	8,56	12	7,13E-01	3,85
16777216	4096	2,77E-12	36,04	12	3,00E+00	4,21

Tabla 6.5: Resultados de la ecuación $-\Delta u(x,y) + \sigma u(x,y) \;=\; f(x,y)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=1

Nodos totales	**N y M**	$\|\|e\|\|_\infty$	**Tiempo (s)**	**Ciclos**	**t/c (s)**	**Δt/c**
256	16	3,28E-05	1,10E-03	10	1,10E-04	
1024	32	5,39E-07	3,50E-03	11	3,18E-04	2,89
4096	64	8,47E-09	1,24E-02	12	1,03E-03	3,25
16384	128	1,33E-10	4,82E-02	12	4,02E-03	3,89
65536	256	2,07E-12	2,08E-01	12	1,73E-02	4,31
262144	512	1,35E-13	9,44E-01	13	7,26E-02	4,20
1048576	1024	2,71E-13	3,82E+00	13	2,94E-01	4,04
4194304	2048	3,81E-13	14,79	12	1,23E+00	4,20
16777216	4096	5,788E-13	61,57	12	5,13E+00	4,16

Tabla 6.6: Resultados de la ecuación $-\Delta u(x,y) + \sigma u(x,y) \;=\; f(x,y)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=2

Lo primero que se puede destacar es que, para mismo número de nodos totales, el tiempo por ciclo tanto en 1D como en 2D es idéntico (Tabla 6.1 y Tabla 6.5),

Nodos totales	**N y M**	$\|\|e\|\|_\infty$	**Tiempo (s)**	**Ciclos**	**t/c (s)**	**Δt/c**
256	16	5,24E-05	1,17E-03	9	1,30E-04	
1024	32	8,94E-07	3,17E-03	10	3,17E-04	2,44
4096	64	1,42E-08	1,05E-02	10	1,05E-03	3,31
16384	128	2,23E-10	4,03E-02	10	4,03E-03	3,84
65536	256	3,49E-12	1,95E-01	11	1,77E-02	4,39
262144	512	1,23E-13	7,90E-01	11	7,18E-02	4,06
1048576	1024	4,16E-13	3,23E+00	11	2,94E-01	4,09
4194304	2048	6,26E-13	13,38	11	1,22E+00	4,14
16777216	4096	1,69E-12	55,94	11	5,09E+00	4,18

Tabla 6.7: Resultados de la ecuación $-\Delta u(x,y) + \sigma u(x,y) = f(x,y)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=2 y J_2=1

lo cual tiene sentido desde el punto de vista numérico. Además, en este caso, al doblar el número de nodos en ambas direcciones, el tiempo de cálculo por ciclo debería aumentar por 4, lo cual ocurre tal y como se observa en las tablas. En el caso de la Tabla 6.5, debido a que se trabaja con menor orden de error en el método y no se disponen de excesivas celdas por dimensión, no entra en juego apenas el error de redondeo, a pesar de que con 4096 celdas por dirección parece que ya comienza a afectar. Sin embargo, en las tablas 6.6 y 6.7, donde el error del método es menor, a partir de 1024 por dimensión celdas ya comienza a el error de redondeo del ordenador. De nuevo, tal y como se ha comentado, esto es evitable aplicando un precondicionamiento a las matrices.

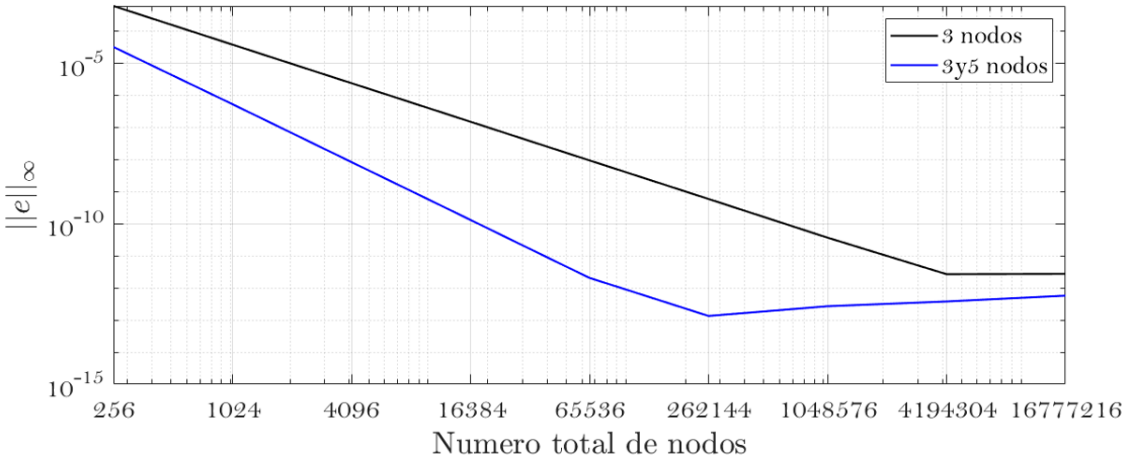

Figura 6.4: Error en norma infinita obtenido mediante multigrid 2D con Fortran y con diferentes nodos de cálculo

Por otra parte remarcar que con unos valores de J_1=1 y J_2=2 se están calculando 16 millones de céldas con un error de $5{,}78E-13$ en tan solo 60 segundos. Teniendo en cuenta que la velocidad del código, tal y como se explicará en el capítulo siguiente, se puede aumentar considerablemente, es una velocidad de cálculo muy buena para un sistema de ecuaciones tan grande. En el caso 1D se ha comentado que la diferencia de tiempo de cálculo por ciclo entre J_1=1 J_2=1 y J_1=2 J_2=2 era prácticamente

imperceptible, ya que apenas se añadían operaciones al cálculo. Sin embargo, en el caso 2D donde se disponen de muchos más nodos de cálculo, esta diferencia sí es apreciable ya que por ejemplo, en el caso de 16 millones de nodos, el tiempo por ciclo aumenta hasta 2 segundos cuando se pasa de J_1=1 J_2=1 a J_1=1 J_2=2.

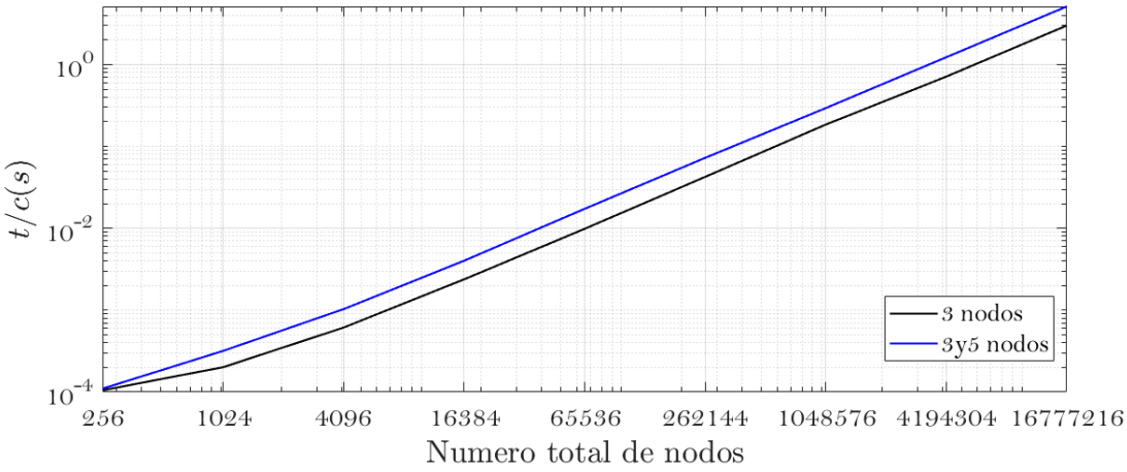

Figura 6.5: Tiempo por ciclo obtenido mediante multigrid 2D con Fortran y con diferentes nodos de cálculo

Señalar que en el caso 2D cuando se utiliza J_1=2 y J_2=2, la matriz resultante también tiene muchos valores propios cercanos a la unidad, lo que ralentiza mucho la convergencia. Por ejemplo, para calcular 16 millones de nodos, con 4096 celdas en cada dimensión, se tardan 393 segundos con 69 ciclos. De esta forma, el tiempo por ciclo son 5,8 segundos, un valor similar a los obtenidos con J_1=1 J_2=2 o J_1=2 J_2=1.

ν	**Tiempo resolución (s)**	**Iteraciones ciclo V**
1	17,32	26
2	13,69	14
3	15,46	12
4	19,22	12
5	22,61	11
6	24,9	10
10	33,3	7
100	178,5	6

Tabla 6.8: Tiempo de resolución de la ec. (6.8a) en función de ν con $N = 2048$ y $M = 2048$ y $J_1 = 1$ y $J_2 = 2$.

En el caso de la ecuación a resolver y con la ley de mallado escogida, cualquier combinación de $J_{1,2} \geq 3$ con $J_{1,2} \geq 2$ resulta en sistemas no convergentes mediante el método empleado, ya que el radio espectral de las matrices de iteración resulta ser mayor a la unidad. Para solucionar este problema se podría, o bien probar con diferentes leyes de mallado o bien aplicar un proceso de subrelajación o amortiguamiento de la solución en cada iteración mediante Gauss-Seidel.

En otro orden de cosas, en la Tabla 6.8 se puede observar el tiempo de resolución varía en función del valor de ν. Tal y como se comentó en la Sección 2.4, un alto valor de ν no mejora la convergencia pues tras un par de ciclos ya se elimina el error de alta frecuencia. En el caso mostrado, el valor óptimo está en $\nu = 2$, aunque este valor puede variar en función del problema o de las matrices de iteración. Finalmente, en la Figura 6.6 se puede observar la solución ec. (6.10), mientras que en la Figura 6.7 se puede observar el error absoluto de la solución con $N = 128$, $M = 128$, $J_1 = 1$ y $J_2 = 2$.

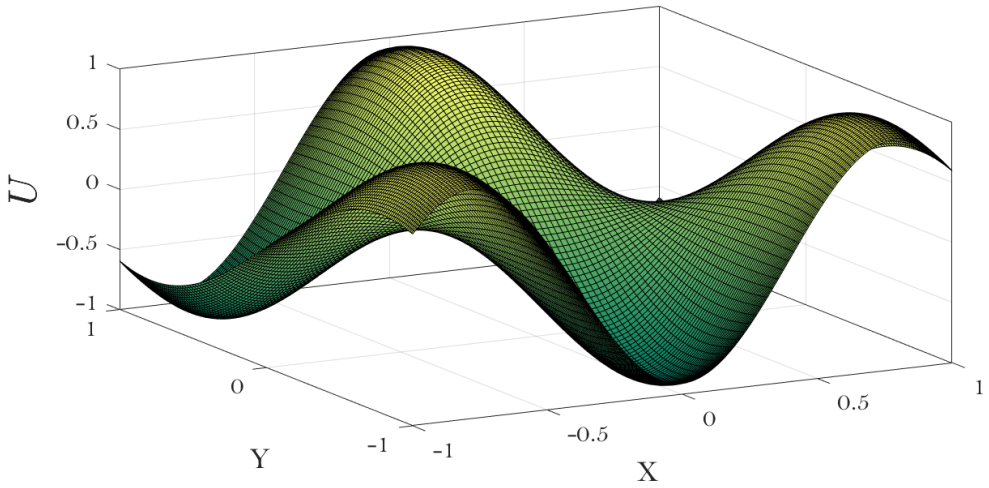

Figura 6.6: Valor de la función $e^x \cos(2x - 1)$ (eje Y) en función de los nodos del dominio x_i (eje X)

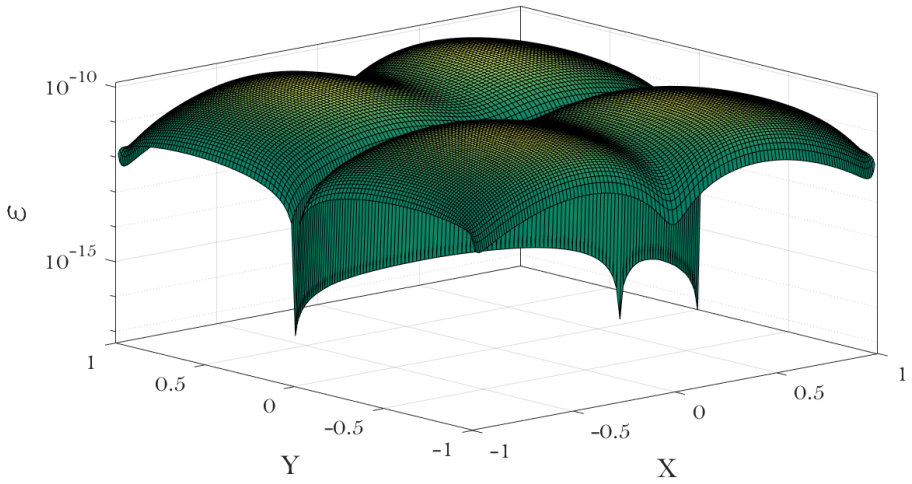

Figura 6.7: Error absoluto v_{ij} tras resolver el sistema ec. (6.1a) con función solución ec. (6.3) mediante multigrid y CTFD con $N = 128$, $M = 128$ y tolerancia entre iteraciones de 10^{-12}.

6.3. Ecuación del calor 2D

En el caso de la ecuación del calor, la ecuación a resolver es la expuesta en el Capítulo 5, de nuevo con condiciones de contorno tipo Dirichlet:

$$\frac{\partial u(x,y,t)}{\partial t} - \sigma \Delta u(x,y,t) = f(x,y,t), \quad \sigma \geq 0, \tag{6.12a}$$

$$u(x_0,y,t) = u_1(y,t), \quad u(x_N,y,t) = u_2(y,t) \tag{6.12b}$$

$$u(x,y_0,t) = u_3(x,t), \quad u(x,y_M,t) = u_4(x,t) \tag{6.12c}$$

donde en este caso, el operador Δ en 2D es $\Delta = \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. Las condiciones de resolución del caso son las mismas que en el caso 2D:

- Constante del problema $\sigma = 1$
- Dominio de resolución: $x \in \{-1, 1\}$, $y \in \{-1, 1\}$
- Mallado del dominio tipo tangente hiperbólica (ec. (4.2a) y ec. (4.2b))
- Multigrid: Iteraciones con Gauss-Seidel $\rightarrow$ $\nu_1 = 3$, $\nu_2 = 3$
- Multigrid: Ciclo V

Se utilizarán diferentes tamaños de malla así como diferentes pasos temporales y número de pasos temporales. Del mismo modos que en el caso anterior, las mallas son cuadradas.

La ecuación $f(x,y,t)$ tiene el siguiente valor:

$$f(x,y,t) = \sin(x)\cos(y) + 2t \cdot \sin(x)\cos(y) \tag{6.13}$$

Para poder comparar resultados se debe resolver la ec. (6.12a) de forma analítica, obteniendo el siguiente valor para $u(x,y,t)$:

$$u(x,y,t) = t \cdot \sin(x)\cos(y) \tag{6.14}$$

por lo que las condiciones de contorno son:

$$\begin{aligned} u_{-1,y,t} &= -0{,}844t \cdot \cos(y), \; u_{1,y,t} = 0{,}844t \cdot \cos(y) \\ u_{x,-1,t} &= 0{,}540t \cdot \sin(x), \; u_{x,1,t} = -0{,}540t \cdot \sin(x) \end{aligned} \tag{6.15}$$

En las tablas 6.9, 6.10 y 6.11 se muestran los resultados obtenidos de la resolución de la ec. (6.12a) con diferentes valores de N y M y de pasos temporales. Se ha integrado desde un tiempo $t_0 = 0$ hasta $t_1 = 10^{-3}$ variando el valor de Δt. La primera conclusión que se puede obtener es que el error del método viene determinado principalmente por el paso temporal y es prácticamente independiente del mallado. A medida que el paso temporal decrece, es decir, se tienen más pasos temporales para el mismo dominio temporal, el error de resolución decrece. Por otra parte, como se ha comentado, la influencia del número de celdas apenas influye en el resultado.

Nodos	**N y M**	$\|\|e\|\|_\infty$	**Tiempo (s)**	**Pasos**	**t/p (s)**
4096	64	2,14E-10	1421,85	100000	0,0142185
4096	64	2,13E-09	134,21	10000	0,013421
4096	64	2,13E-08	13,52	1000	0,01352
4096	64	2,09E-07	1,56	100	0,0156
4096	64	1,76E-06	0,14	10	0,014

Tabla 6.9: Resultados de la ecuación $\frac{\partial u(x,y,t)}{\partial t} - \sigma\Delta u(x,y,t) = f(x,y,t)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=2. $N = 64$ y $M = 64$. $t_0 = 0$, $t_1 = 10^{-3}$.

Nodos	**N y M**	$\|\|e\|\|_\infty$	**Tiempo (s)**	**Pasos**	**t/p (s)**
1024	32	1,43E-10	392,58	100000	0,0039258
1024	32	1,43E-09	37,44	10000	0,003744
1024	32	1,43E-08	3,73	1000	0,00373
1024	32	1,42E-07	0,37	100	0,0037
1024	32	1,35E-06	0,045	10	0,0045

Tabla 6.10: Resultados de la ecuación $\frac{\partial u(x,y,t)}{\partial t} - \sigma\Delta u(x,y,t) = f(x,y,t)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=2. $N = 32$ y $M = 32$. $t_0 = 0$, $t_1 = 10^{-3}$.

Nodos	**N y M**	$\|\|e\|\|_\infty$	**Tiempo (s)**	**Pasos**	**t/p (s)**
256	16	5,93E-11	126,135	100000	0,00126135
256	16	5,93E-10	12,59	10000	0,001259
256	16	5,93E-09	1,37	1000	0,00137
256	16	5,92E-08	0,16	100	0,0016
256	16	5,81E-07	0,0185	10	0,00185

Tabla 6.11: Resultados de la ecuación $\frac{\partial u(x,y,t)}{\partial t} - \sigma\Delta u(x,y,t) = f(x,y,t)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=2. $N = 16$ y $M = 16$. $t_0 = 0$, $t_1 = 10^{-3}$.

Nodos	**N y M**	$\|\|e\|\|_\infty$	**Tiempo (s)**	**Pasos**	**t/p (s)**
1024	32	2,25E-08	369,18	100000	0,0036918
1024	32	2,23E-08	36,38	10000	0,003638
1024	32	1,43E-08	3,66	1000	0,00366
1024	32	3,51E-09	0,3634	100	0,003634
1024	32	4,06E-10	0,03956	10	0,003956
1014	32	4,13E-11	0.0059	1	0.0059

Tabla 6.12: Resultados de la ecuación $\frac{\partial u(x,y,t)}{\partial t} - \sigma\Delta u(x,y,t) = f(x,y,t)$ resuelta mediante multigrid-CTFD 2d en Fortran. J_1=1 y J_2=2. $N = 32$ y $M = 32$. $t_0 = 0$, $dt = 10^{-6}$.

Por otra parte, se puede observar que el tiempo de cálculo por paso temporal

no varía si no se modifica el número de celdas, lo cual tiene sentido ya que se están resolviendo diferentes sistemas, pero con el mismo número de operaciones. Por otra parte, se puede observar que al doblar el número de nodos, se dobla el tiempo de cálculo ya que al pasar de 1024 nodos a 4096 (4 veces más), el tiempo por paso temporal pasa a ser aproximadamente 4 veces más.

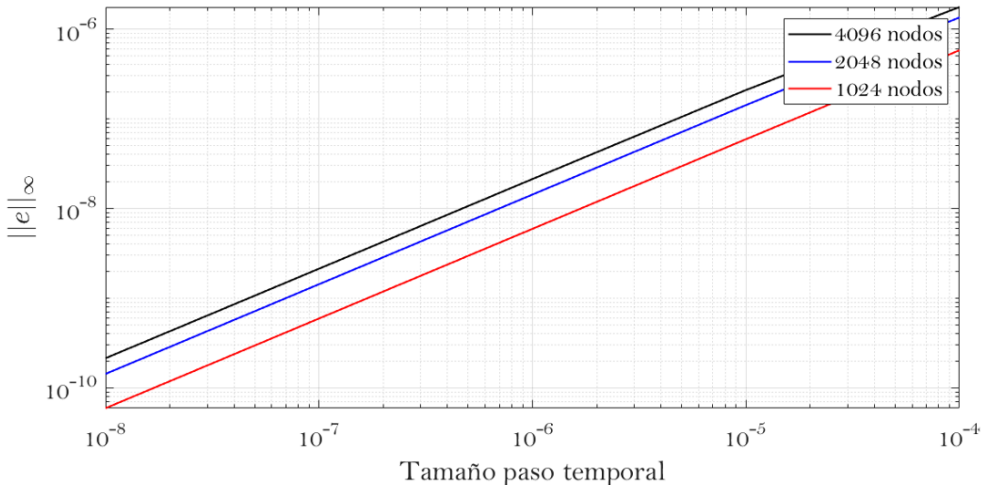

Figura 6.8: Error en norma infinita obtenido en la Ecuación del calor en función del número de nodos y del tamaño del paso temporal

Por otra parte, en la Tabla 6.12 se puede observar el resultado de resolver la ecuación con 1024 nodos (32 en cada dimensión) pero, esta vez, manteniendo el paso temporal constante ($\Delta t = 10^{-6}$) y modificando el número de pasos temporales. Lo que se desprende de los resultados es que el error local, es decir, el error obtenido en caso de partir de una solución sin errores (caso de 1 iteración) es muy bajo. Este error se va acumulando a medida que se realizan más pasos temporales pero tiene un límite, el denominado error global, siendo en este caso $2,25E-8$, lo cual es un valor más que correcto.

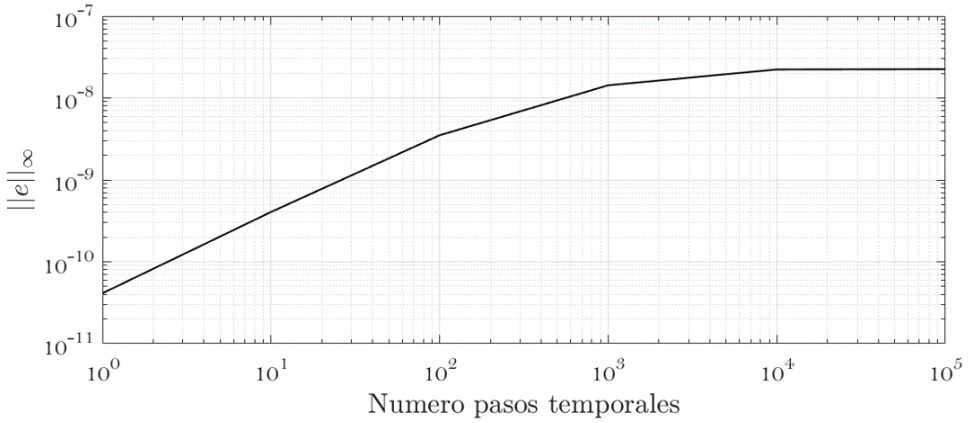

Figura 6.9: Error en norma infinita obtenido en la Ecuación del calor en función del número de pasos temporales para $N = 32$.

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

En base a los objetivos desarrollados en la Sección 1.5, se pueden extraer una serie de conclusiones relativas a la realización del presente trabajo. Desde el punto de vista de la investigación, se ha corroborado que el método de las diferencias finitas compactas, no muy extendido en la actualidad, es muy potente a la hora de calcular las derivadas de una función gracias principalmente a ser un método semi-espectral, que ofrece una precisión similar a métodos espectrales con una flexibilidad a la hora de situar los nodos típica de las diferencias finitas.

Por otra parte, se ha probado que el método multigrid es extremadamente útil a la hora de resolver sistemas de muy grandes dimensiones. En conjunto, se han conseguido rutinas capaces de resolver problemas en 1 y 2 dimensiones de forma precisa en escasos segundos aún con sistemas muy grandes.

Destacar en este punto que el programa realizado trabaja en doble precisión, lo que significa que se disponen de hasta 16 dígitos de precisión. La mayor parte del código DNS en la actualidad está desarrollado en precisión simple, 8 dígitos de error, por lo que a pesar de tener mucho error de redondeo en algunos casos, se han resuelto sistemas en 2D de 16 millones de nodos en menos de un minuto con un orden de precisión 5 órdenes de magnitud mejor de lo que se podría obtener con la mayoría de códigos DNS. Además, recalcar también que el programa realizado es totalmente flexible, es decir, permite modificar fácilmente diferentes parámetros del cálculo como la función a resolver, los tamaños de malla, el tipo de mallado, los nodos de cálculo de las diferencias finitas compactas, etc. Esto ofrece muchas posibilidades a la hora de seguir desarrollando el código en el futuro.

Finalmente, atendiendo a los resultados obtenidos en la Sección 6.3, se puede afirmar que se ha conseguido el objetivo principal del trabajo, que consiste en obtener un método base para poder realizar un túnel de viento virtual. Se ha demostrado la capacidad del código de resolver problemas similares a los obtenidos de las ecuaciones de Navier-Stokes de forma precisa y eficaz. No obstante, destacar que, tal y como se mencionará a continuación, todavía es posible mejorar sustancialmente el comportamiento del código.

7.2. Trabajos futuros

El objetivo principal era desarrollar un método base para la simulación de un túnel de viento virtual. Por lo tanto, al pensar en un posible trabajo futuro, éste surge de forma natural, y es simplemente adaptar el código hasta obtener un túnel de viento virtual. En la etapa actual de desarrollo, el código, tal y como se ha visto, es capaz de resolver ecuaciones en derivadas parciales con influencia temporal en 2D de forma eficiente, por lo que los pasos a seguir para conseguir un túnel de viento virtual serían los siguientes:

- **Precondicionamiento de las matrices:** En el punto actual de desarrollo, a medida que se aumenta el número de nodos de cálculo, comienzan a aparecer términos muy pequeños en las matrices debido a los minúsculos tamaños de malla que generan. Esto provoca que el error de redondeo provocado por la precisión limitada de los ordenadores disminuya la precisión del cálculo obtenido. Mediante el precondicionamiento de las matrices, estos términos tan pequeños se eliminan, permitiendo por lo tanto una mejora en la precisión del cálculo.

- **Optimización y paralelización del código:** A pesar de los buenos resultados obtenidos en cuanto a tiempo de cálculo, estos pueden ser mejorados sustancialmente mediante una optimización del código, disminuyendo el número de operaciones a realizar. Además, para mejorar aún más la velocidad, se debe realizar una paralelización del mismo código mediante OpenMP.

- **Transformada de Fourier:** Actualmente se están resolviendo ecuaciones en 2D pero, para obtener un túnel de viento virtual, es necesario operar en las tres dimensiones. Siguiendo la explicación realizada en el Capítulo 2, el siguiente paso sería añadir esta tercera dimensión mediante la aplicación de las trasformadas rápidas de Fourier en la dimensión z.

- ***Immerse Boundary Method*:** Una vez se puedan resolver flujos en 3D, el siguiente paso sería añadir geometrías de cálculo mediante el método *Immerse Boundary Method*.

Una vez llevados a cabo los avances anteriores, se dispondrá de un túnel de viento virtual capaz de simular por ejemplo casos de canales o capas límite. En caso de que alguna de las ampliaciones anteriores se lleven a cabo, este trabajo puede servir como punto de partida.

Capítulo 8

Pliego de condiciones y presupuesto

8.1. Pliego de condiciones

La normativa que regula el tipo de actividad realizada para minimizar los riesgos para el trabajador en este caso viene dada por el Real Decreto 488/1997 del 14 de abril, sobre disposiciones mínimas de seguridad y salud relativas al trabajo con equipos que incluyen pantallas de visualización (el constituido por un equipo con pantalla de visualización provisto, en su caso, de un teclado o dispositivo de adquisición de datos, de un programa para la interconexión persona-máquina, de accesorios ofimáticos y de un asiento y mesa o superficie de trabajo). Las cuatro variables atendidas para prever el tipo de riesgos a los que se enfrenta el trabajador, según esta normativa:

- Tiempo de trabajo con la pantalla de visualización.
- Tiempo de atención requerida ante la pantalla, que a su vez puede ser continua o discontinua.
- Exigencia y grado de complejidad de la tarea realizada ante la pantalla
- Necesidad de obtener una información de manera muy rápida.

Los riesgos presentes, según esta normativa, son:

- Seguridad (por contactos eléctricos)
- Higiene industrial (iluminación, ruido y condiciones termohigrométricas)
- Ergonomía (fatiga visual, física y mental)

Seguridad

En medidas de seguridad, la empresa debe haber adoptado medidas de emergencia en las que se incluyan las vías y salidas de evacuación en caso de que se declare

una emergencia. Además, todas las instalaciones contra incendios deben estar proyectadas, implantadas y mantenidas por empresas debidamente autorizadas por el organismo competente.

Finalmente, la instalación eléctrica debe estar proyectada, puesta en funcionamiento y mantenida por una empresa debidamente autorizada. Esta instalación debe evitar originar contactos con las personas, incendios y explosiones, ateniéndose para ello a lo establecido sobre tensiones y seguridad en los Reglamentos de Baja y Alta Tensión en vigor.

Higiene industrial

En lo referente a la iluminación del lugar de trabajo, ésta debe adaptarse a las características de la actividad que se realiza en él, según lo dispuesto en el anexo IV del Real Decreto 486/1997, de 14 de abril, por el que se establecen las disposiciones mínimas de seguridad y salud en los lugares de trabajo.

Por otra parte, los niveles de ruido deben ser evaluados por el empresario según el Real Decreto 1316/989 del 27 de octubre, sobre la protección de la salud y la seguridad de los trabajadores contra los riesgos relacionados con la exposición al ruido. En el marco de la Ley 31/1995, de Prevención de Riesgos Laborales el citado Real Decreto establece que los riesgos derivados de la exposición al ruido deben eliminarse en su origen o reducirse al nivel más bajo posible, teniendo en cuenta los avances técnicos. Para evitar exceder el $L_{A_{eq}}$ máximo establecido de 55 dB(A), deben utilizarse equipos con una emisión sonora mínima, además de optimizar la acústica del lugar del trabajo.

Finalmente, respecto a las condiciones termohigrométricas, la exposición a las condiciones ambientales no debe suponer un riesgo para la seguridad y la salud, ni debe ser una fuente de incomodidad o molestia, evitando:

- Humedad y temperaturas extremas: la temperatura operativa de confort debe mantenerse en el rango de 23 a 26°C en verano y de 20 a 24°C en invierno, mientras que la humedad relativa del aire debe mantenerse siempre entre el 45 % y el 65 %.
- Corrientes de aire molestas.

De este modo, el aislamiento térmico de los locales cerrados debe adecuarse a las condiciones climáticas propias del lugar.

Ergonomía

El diseño del puesto de trabajo está directamente relacionado con los problemas de postura, por lo que debe adecuarse correctamente para reducir los posibles problemas derivados de una larga estancia en posiciones estáticas. Se debe disponer de sillas regulables en altura y profundidad, así como en inclinación, con una suave prominencia en el respaldo para proporcionar apoyo lumbar. Además, la mesa de trabajo debe tener una superficie lo suficientemente grande como para permitir movimientos de trabajo y permitir cambios de postura.

Por otro lado, para reducir la fatiga visual, se debe situar la pantalla entre 45 y 75 cm del usuario. La pantalla debe colocarse de manera que su área útil pueda ser vista bajo ángulos comprendidos entra la línea de visión horizontal y la trazada a 60° bajo la horizontal. En el plano horizontal, la pantalla debe estar colocada dentro de un ángulo de 70°, dentro del campo de visión del usuario.

Los entornos en los que se ha llevado a cabo la actividad desarrollada en este trabajo cumplen lo dispuesto en el Real Decreto 488/1997 del 14 de abril, específico para las condiciones de trabajo dadas, y en el Real Decreto 486/1997 del 14 de abril, sobre condiciones mínimas de seguridad y salud aplicables a los lugares de trabajo.

8.2. Presupuesto

En esta sección se procederá a valorar económicamente el trabajo invertido en el desarrollo del proyecto, teniendo en cuenta tanto el número de horas dedicadas al trabajo como los recursos materiales y licencias de software necesarias para su realización. Como unidad para valorar económicamente el trabajo humano se utilizará la hora de trabajo, mientras que el coste unitario se medirá en euros/hora. Por otra parte, para valorar el uso de las máquinas se utilizarán los meses empleados, mientras que la unidad será euros/mes, teniendo en cuenta la vida útil de la máquina. Para software y material, se emplearán los euros.

8.2.1. Relación horas de trabajo empleadas

Para la realización del trabajo se han realizado una serie de tareas, las cuales han implicado un uso de horas de trabajo por parte tanto del ingeniero como del tutor. Las tareas realizadas de forma resumida son las siguientes:

- **Tarea 1:** Familiarización con el entorno de trabajo, las herramientas a utilizar y la documentación previa al inicio del mismo
- **Tarea 2:** Desarrollo conceptual de los algoritmos a implementar tanto en Matlab como en Fortran
- **Tarea 3:** Desarrollo e implementación de los algoritmos en Matlab
- **Tarea 4:** Desarrollo e implementación de los algoritmos en Fortran
- **Tarea 5:** Extracción y validación de resultados
- **Tarea 6:** Redacción de memoria y presentación

En la Tabla 8.1 se muestran las horas dedicadas a cada tarea, así como el coste humano asociado a cada una de las mismas tanto por parte del ingeniero como por parte del tutor/doctor.

	Concepto	Horas	Coste unitario	Subtotal
Tarea 1	Horas Ingeniero	30	15 €/h	450 €
	Horas Doctor	5	30 €/h	150€
Tarea 2	Horas Ingeniero	80	15 €/h	1200€
	Horas Doctor	15	30 €/h	450€
Tarea 3	Horas Ingeniero	100	15 €/h	1500€
	Horas Doctor	10	30 €/h	300€
Tarea 4	Horas Ingeniero	280	15 €/h	4200€
	Horas Doctor	20	30 €/h	600€
Tarea 5	Horas Ingeniero	30	15 €/h	450€
	Horas Doctor	5	30 €/h	150€
Tarea 6	Horas Ingeniero	120	15 €/h	1800€
	Horas Doctor	10	30 €/h	300€
TOTAL	Horas Ingeniero	640	15 €/h	9600€
	Horas Doctor	65	30 €/h	1950€
			Total	**11.550,00 €**

Tabla 8.1: Desglose de costes de las horas empleadas en la realización del trabajo por el personal humano

8.2.2. Recursos materiales y licencias Software

En este apartado se detalla el coste asociado a los recursos materiales (máquinas y material ofimática en este caso) utilizados en el trabajo, así como las licencias de Software necesarias para la implementación del mismo.

En la Tabla 8.2 se puede observar el desglose de costes para esta parte. Se debe tener en cuenta que la estación de trabajo utilizada para la implementación de los algoritmos ya se encuentra amortizada. Además, todo el software utilizado es libre, a excepción de Matlab, del que se ha utilizado la licencia gratuita asociada a la universidad. Los costes asociados al material adicional corresponden a material de oficina.

Concepto	Cantidad	Coste unitario	Subtotal
Ordenador portátil	1	1500 €	1500 €
Estación de trabajo	1	-€	-€
Software	1	-€	-€
Material adicional	1	40 €	40€
		Total	**1540,00€**

Tabla 8.2: Desglose costes asociados al material utilizado para la realización del trabajo

8.2.3. Coste total del proyecto

Sumando los costes de personal humano, los costes de materiales y añadiendo un 25 % de costes indirectos, un 6 % de beneficio industrial y un 21 % de IVA:

Categoría	Subtotal
Horas trabajo	11.550,00 €
Material y Software	1.540,00 €
Costes indirectos	3.272,50 €
Beneficio industial	785,40 €
IVA	2.748,90 €
Total	**19.896,80 €**

Tabla 8.3: Costes totales del trabajo incluyendo costes indirectos, beneficio industrial e IVA.

El coste neto estimado del trabajo asciende a **19.896,80€**.

Bibliografía

[1] Abbas-Bayoumi, A., and Becker, K. An industrial view on numerical simulation for aircraft aerodynamic design. *Journal of Mathematics in Industry 1*, 1 (2011), 10.

[2] Anderson Jr, J. D. Brief history of the early development of theoretical and experimental fluid dynamics^{0}. *Encyclopedia of Aerospace Engineering* (2010).

[3] Blazek, J. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[4] Briggs, W. L., Henson, V. E., and McCormick, S. F. *A multigrid tutorial*. SIAM, 2000.

[5] Hoyas, S., Oberlack, M., Kraheberger, S., and Alcantara-Avila, F. Turbulent channel flow at Re_τ=10000. `http://meetings.aps.org/link/BAPS.2018.DFD.G38.8/`, 2018.

[6] Hutchins, N., Chauhan, K., Marusic, I., Monty, J., and Klewicki, J. Towards reconciling the large-scale structure of turbulent boundary layers in the atmosphere and laboratory. *Boundary-layer meteorology 145*, 2 (2012), 273–306.

[7] Jiménez, J. Computers and turbulence. *European Journal of Mechanics-B/Fluids 79* (2020), 1–11.

[8] Lele, S. K. Compact finite difference schemes with spectral-like resolution. *Journal of computational physics 103*, 1 (1992), 16–42.

[9] Marusic, I., Monty, J. P., Hultmark, M., and Smits, A. J. On the logarithmic region in wall turbulence. *Journal of Fluid Mechanics 716* (2013).

[10] Oliveira, F., Pinto, M., Marchi, C., and Araki, L. Optimized partial semicoarsening multigrid algorithm for heat diffusion problems and anisotropic grids. *Applied Mathematical Modelling 36*, 10 (2012), 4665–4676.

[11] Orszag, S. A., and Patterson Jr, G. Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Physical Review Letters 28*, 2 (1972), 76.

[12] Peinado Asensi, I. Resolución de ecuaciones de convección-difusión en 2d usando el método de las diferencias finitas compactas. Master's thesis, Universitat Politècnica de València, 2019.

[13] Spalart, P., and Venkatakrishnan, V. On the role and challenges of cfd in the aerospace industry. *The Aeronautical Journal 120*, 1223 (2016), 209.

[14] Spalart, P. R., Moser, R. D., and Rogers, M. M. Spectral methods for the navier-stokes equations with one infinite and two periodic directions. *Journal of Computational Physics 96*, 2 (1991), 297–324.

[15] Van Gunsteren, W. F., and Berendsen, H. J. Computer simulation of molecular dynamics: methodology, applications, and perspectives in chemistry. *Angewandte Chemie International Edition in English 29*, 9 (1990), 992–1023.

[16] VVAA. Recurso web: Top 500. the list. `https://top500.org/statistics/perfdevel/`. Accedido: Agosto 2020.

[17] Wray, A. A. Minimal storage time advancement schemes for spectral methods. *NASA Ames Research Center, California, Report No. MS 202* (1990).

[18] Yamamoto, Y., and Tsuji, Y. Numerical evidence of logarithmic regions in channel flow at $\mathrm{Re}_\tau = 8000$. *Physical Review Fluids 3*, 1 (2018), 012602.

[19] Zhang, J. Multigrid method and fourth-order compact scheme for 2d poisson equation with unequal mesh-size discretization. *Journal of Computational Physics 179*, 1 (2002), 170–179.