



**UNIVERSIDAD NACIONAL AUTÓNOMA  
DE MÉXICO**

---

---

**FACULTAD DE CIENCIAS**

**Resolución de Ecuaciones Diferenciales Parciales  
Mediante el Método de Diferencias Finitas y su  
Paralelización**

**T E S I S**

**QUE PARA OBTENER EL TÍTULO DE:**

**Matemático**

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

**Omar Jonathan Mendoza Bernal**



**DIRECTOR DE TESIS:  
Dr. Antonio Carrillo Ledesma  
Ciudad Universitaria, CD. MX. 2016**



Universidad Nacional  
Autónoma de México



**UNAM – Dirección General de Bibliotecas**  
**Tesis Digitales**  
**Restricciones de uso**

**DERECHOS RESERVADOS ©**  
**PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL**

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

1. Datos del alumno

Mendoza

Bernal

Omar Jonathan

5526 0701

Universidad: Universidad Nacional Autónoma de México

Facultad de Ciencias

Matemáticas

099196604

2. Datos del Tutor

Dr.

Antonio

Carrillo

Ledesma

3. Datos del Sinodal 1

Dr.

Ernesto

Rubio

Acosta

4. Datos del Sinodal 2

Dr.

Arturo

Olvera

Chávez

5. Datos del Sinodal 3

Dra.

Ursula Xiomara

Iturrarán

Viveros

6. Datos del Sinodal 4

Dr.

Guillermo

Hernández

García

7. Datos del trabajo escrito

Resolución de Ecuaciones Diferenciales Parciales mediante  
el Método de Diferencias Finitas y su Paralelización

101 p.

2016

Agradecimientos:

Quiero agradecer a todas las personas que hicieron posible que este trabajo llegara a su fin:

- Al director, Dr. Antonio Carrillo por su paciencia y apoyo.
- A los sinodales, por sus comentarios.
- A mis padres Raquel Bernal Salazar y Jorge Gregorio Mendoza Jasso, sin ellos nada hubiera sido posible.
- A mis hermanos Alberto y Toño.

# Índice general

<b>1. Introducción</b>	<b>6</b>
1.1. Ecuaciones Diferenciales . . . . .	6
1.2. Ecuaciones Diferenciales Parciales . . . . .	7
1.3. Problemas Bien Planteados . . . . .	8
1.4. Ecuación de Laplace, Calor y Onda . . . . .	10
1.5. Métodos de Discretización . . . . .	16
1.6. Métodos de Solución de Sistemas de Ecuaciones . . . . .	17
1.7. Necesidad del Cómputo Paralelo. . . . .	18
1.8. Objetivos . . . . .	18
<b>2. Método de Diferencias Finitas</b>	<b>20</b>
2.1. Diferencias Finitas . . . . .	20
2.2. Aproximación a la Primera Derivada . . . . .	21
2.3. Aproximación a las Derivadas en 1D, 2D y 3D . . . . .	22
2.4. Estencil para Problemas en 3D . . . . .	24
2.5. Condiciones de frontera . . . . .	26
2.6. Procedimiento General del Método de Diferencias Finitas . . . . .	27
2.6.1. Problema en una dimensión con Condiciones de Frontera Dirichlet . . . . .	28
2.6.2. Problema en una dimensión con Condiciones de Frontera Neumann . . . . .	29
2.6.3. Problema en una dimensión con Condiciones de Frontera Robin . . . . .	30
2.7. $a(x)$ no constante . . . . .	31
2.8. Discretización del tiempo . . . . .	32
2.8.1. Ecuaciones Con Primera Derivada Temporal (Esquema Theta) . . . . .	32
2.8.2. Ecuaciones Con Segunda Derivada Temporal . . . . .	33
2.8.3. Upwind . . . . .	35
2.9. Consistencia, estabilidad, convergencia y error del Método de Diferencias Finitas . . . . .	35
2.9.1. Error Global . . . . .	35
2.9.2. Error Local de Truncamiento . . . . .	36
2.10. Estabilidad de Métodos con Paso de Tiempo . . . . .	37
2.10.1. Análisis de von Neumann Simplificado para métodos con Paso de Tiempo . . . . .	37
<b>3. Resolución de Grandes Sistemas de Ecuaciones</b>	<b>39</b>
3.1. Estructura Óptima para el uso de Matrices . . . . .	39
3.2. Métodos de Resolución . . . . .	41
3.2.1. Descomposición LU . . . . .	41
3.2.2. Descomposición Cholesky . . . . .	42

3.2.3.	Factorización LU de Sistemas Tridiagonales . . . . .	42
3.2.4.	Método del Gradiente Conjugado . . . . .	43
3.2.5.	GMRES . . . . .	45
<b>4.</b>	<b>Implementación Computacional</b>	<b>46</b>
4.1.	Scilab . . . . .	46
4.2.	MatLab (Octave) . . . . .	47
4.3.	C++ . . . . .	48
4.4.	Biblioteca gmm++ . . . . .	49
4.5.	Método de Descomposición de Dominio de Subestructuración . . . . .	50
<b>5.</b>	<b>Aplicaciones y Pruebas</b>	<b>53</b>
5.1.	Ejemplos . . . . .	53
5.1.1.	Ecuación de Poisson . . . . .	53
5.1.2.	Ejemplos de Advección-Difusión . . . . .	55
5.1.3.	Ecuación de Calor o Difusión . . . . .	58
5.1.4.	Ecuación de Onda en 1D . . . . .	62
5.1.5.	Ecuación de Advección con Upwind . . . . .	64
5.1.6.	Ecuación de Helmholtz . . . . .	66
5.2.	Método de Descomposición de Dominio de Schur o Subestructuración . . . . .	67
<b>6.</b>	<b>Comentarios Finales y Conclusiones</b>	<b>69</b>
6.1.	Conclusiones . . . . .	69
6.2.	Trabajo Futuro . . . . .	70
<b>A.</b>	<b>Cómputo Paralelo</b>	<b>71</b>
<b>B.</b>	<b>Método de Descomposición de Dominio de Subestructuración</b>	<b>76</b>
<b>C.</b>	<b>Crank-Nicolson 1D Para la Ecuación de Calor Scilab, C++ y MatLab</b>	<b>83</b>
C.1.	Scilab . . . . .	83
C.2.	C++ . . . . .	85
C.3.	MatLab . . . . .	88
<b>D.</b>	<b>Ecuación de Advección con upwind en python</b>	<b>92</b>
D.1.	Python . . . . .	92
D.2.	Ejemplo . . . . .	92
<b>E.</b>	<b>Ecuación de Helmholtz</b>	<b>96</b>
E.1.	Ejemplo Numérico . . . . .	96
<b>F.</b>	<b>Gráficos en MatLab</b>	<b>100</b>

# Capítulo 1

## Introducción

### 1.1. Ecuaciones Diferenciales

La comprensión de la naturaleza y sus fenómenos, necesita del auxilio de las Ecuaciones Diferenciales tanto Ordinarias como Parciales, ya que estas constituyen una herramienta esencial para matemáticos, físicos e ingenieros, véase[13], pues, sucede que las leyes físicas que gobiernan los fenómenos de la naturaleza se expresan habitualmente en forma de ecuaciones diferenciales, por lo que estas, en sí, constituyen una expresión cuantitativa de dichas leyes. La enorme importancia de las ecuaciones diferenciales (ordinarias y parciales) para la ciencia, se debe principalmente al hecho de que la investigación de muchos problemas puede reducirse a la solución de tales ecuaciones. Los cálculos que requiere las trayectorias de proyectiles, la investigación de la estabilidad de aeronaves en vuelo o el curso de una reacción química, las leyes de conservación de la masa y de la energía, modelación de yacimientos petroleros, de acuíferos etc. se expresan en forma de ecuaciones diferenciales. Las ecuaciones del movimiento de los cuerpos (la segunda ley de Newton) es una ecuación diferencial, como lo es la propagación de las ondas, la transmisión del calor, el movimiento de partículas subatómicas, todo ello depende de la solución de ecuaciones diferenciales.

La teoría de Ecuaciones Diferenciales se ha convertido en uno de los campos de estudio más importantes en matemáticas, debido a su frecuente aplicación en diferentes campos de la física, ingeniería y otras ciencias, véase[11]. Muchas de la leyes de la naturaleza se describen en términos de ecuaciones diferenciales ordinarias (EDO, ecuaciones que involucran las derivadas de una función de una sola variable) o bien, de ecuaciones diferenciales parciales (EDP, donde las ecuaciones contienen ecuaciones parciales de funciones de varias variables). Históricamente, la teoría de EDO y EDP resultó del estudio de ciertas ecuaciones encontradas en la física.

Entre las EDP más representativas se encuentran:

- La ecuación de onda. Surge al describir fenómenos relativos a la propagación de ondas en un medio continuo. Los estudios de ondas acústicas, ondas de agua, ondas electromagnéticas y vibraciones mecánicas están basados en esta ecuación.
- La ecuación de calor. Constituye una herramienta de gran utilidad para dar solución a problemas de flujo de calor. Esta ecuación aparece también en una gran cantidad de problemas, por ejemplo: la concentración de material en difusión.
- La ecuación de Laplace. Esta ecuación puede modelar la distribución de temperatura en estado estacionario para una región, además, de que aparece en problemas de la física como: potenciales electrostáticos, potenciales de hidrodinámica, potenciales armónicos en teoría de elasticidad.

Las ecuaciones diferenciales parciales se suelen clasificar en grupos, los cuales engloban las propiedades matemáticas y físicas cualitativamente similares. La ecuación del calor  $\frac{\partial u}{\partial t} = k \frac{\partial^2 u}{\partial x^2}$  es un ejemplo de ecuación diferencial parcial del tipo **parabólico**. La ecuación de onda  $\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$  tipifica a las ecuaciones del tipo **hiperbólico**. La ecuación de Laplace  $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$  es un ejemplo de ecuación **Elíptica**.

La deducción física de estas tres ecuaciones puede ser consultada en cualquier libro del tema, por ejemplo, véase [6, 5]

Existen varios métodos o técnicas para obtener soluciones explícitas (analíticas o semi-analíticas) de algunas ecuaciones de interés físico, véase [11], sin embargo, muchas tienen soluciones con expresiones complicadas, por ejemplo, que involucran series o alguna representación integral. En muchas situaciones se necesitan cálculos numéricos de las soluciones de las ecuaciones. En un análisis de, por ejemplo, una solución de una ecuación en series de Fourier, podemos imaginar calcular en una computadora los primeros cien términos de la serie, sin embargo, existen métodos más eficientes para obtener resultados numéricos, especialmente cuando se utiliza una computadora. véase [12].

## 1.2. Ecuaciones Diferenciales Parciales

Una ecuación diferencial es una ecuación en donde aparecen derivadas de una o más funciones desconocidas. Dependiendo del número de variables independientes respecto de las que se derive, las ecuaciones diferenciales se clasifican en *ordinarias* o *parciales*, véase [9, 5, 6]

**Definición.** Una ecuación diferencial parcial (EDP) para la función  $u = (x_1, x_2, \dots, x_n)$  con  $u : \Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}$  es una relación de la forma

$$F(x_1, x_2, \dots, u_{x_1}, u_{x_2}, \dots, u_{x_1 x_1}, u_{x_1 x_2}, \dots) = 0 \tag{1.2.1}$$

donde  $u_{x_i}$  representa la derivada parcial de  $u$  con respecto a la variable independiente  $x_i$ .

Entonces se dice que  $F$  es una función que depende de las variables  $(x_1, x_2, \dots, x_n)$  y de una función (o variable dependiente)  $u$  y de las derivadas de  $u$ .

**Definición.** Una función es solución de una EDP, si en alguna región  $\Omega \in \mathbb{R}^n$  del espacio de sus variables independientes, la función y sus derivadas satisfacen a  $F$ , es decir, al sustituir la función solución y sus derivadas se obtiene una identidad.

El orden de una EDP es  $n$  si las derivadas de mayor orden que ocurren en  $F$  son de orden  $n$ . Una EDP es lineal si  $F$  es lineal en la función incógnita y en sus derivadas, y la EDP es cuasi-lineal, si  $F$  es lineal en al menos una de las derivadas de más alto orden; de otra manera la EDP es no-lineal.

**Definición.** Sea  $\Omega \subset \mathbb{R}^n$  el dominio de una función  $u$ , además  $u \in C^2(\mathbb{R}^n)$ , entonces se define el *laplaciano*  $\Delta u$  como

$$\nabla \cdot \nabla u = \nabla^2 u = \Delta u = \sum_{i=1}^n u_{x_i x_i} = \sum_{i=1}^n \frac{\partial^2 u}{\partial x_i^2}$$

Algunos ejemplos de ecuaciones diferenciales parciales son:

Ecuación		Linealidad
Ecuación de Laplace	$\Delta u = 0$	Lineal
Ecuación de Poisson	$\Delta u = f$	Lineal
Ecuación de onda	$u_{tt} = c^2 \Delta u$	Lineal
Ecuación de calor	$u_t = c \Delta u$	Lineal
Ecuación de Korteweg-de Vries	$u_t + uu_x + cu_{xxx} = 0$	Cuasi-Lineal
Ecuación eikonal	$(u_x)^2 + (u_y)^2 + (u_z)^2 = c^2 F$	No-lineal

## EDP Lineales de Segundo Orden

Consideremos la EDP general de segundo orden para  $n$  variables definida en  $\Omega \in \mathbb{R}^n$  en forma de un operador diferencial

$$\mathcal{L}u = \sum_{i,j=1}^n a_{ij} u_{x_i x_j} + \sum_{i=1}^n b_i u_{x_i} + cu \tag{1.2.2}$$

Si las funciones  $a_{ij}, b_i$  y  $c$  no dependen de  $\underline{x} \in \mathbb{R}^n$ , se dice que la ecuación diferencial parcial es de coeficientes constantes. Como  $u_{x_i x_j} = u_{x_j x_i}$  para cualquier función con segunda derivada continua, sin pérdida de generalidad podemos asumir la simetría de  $a_{ij} = a_{ji}$ . A la suma  $\sum_{i,j=1}^n a_{ij} u_{x_i x_j}$  se le conoce como la parte principal de la EDP, la cual se puede escribir como

$$\left( \frac{\partial}{\partial x_1} \quad \cdots \quad \frac{\partial}{\partial x_n} \right) \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} \frac{\partial u}{\partial x_1} \\ \vdots \\ \frac{\partial u}{\partial x_n} \end{pmatrix}$$

por la igualdad de las derivadas cruzadas podemos decir que la matriz de la parte principal de la EDP será real y simétrica.

**Definición.** Sea  $V$  un espacio vectorial sobre un campo  $K$  y sea

$$A : V \rightarrow V$$

un operador de  $V$ . Un elemento  $v \in V$  se llama **eigenvector** o **vector propio** de  $A$  si existe  $\lambda \in K$  tal que  $Av = \lambda v$ . Si  $v \neq 0$ , entonces  $\lambda$  está determinado de forma única. En este caso se dice que  $\lambda$  es un **eigenvalor** o **valor propio** de  $A$  correspondiente a  $v$ . También se dice que  $v$  es un **eigenvector** con el eigenvalor  $\lambda$ .

Si  $A$  es una matriz cuadrada de  $n \times n$  con coeficientes en  $K$ , entonces un **eigenvector** de  $A$  es, por definición, un **eigenvector** de la aplicación lineal de  $K^n$  en si mismo representado pi esta matriz relativa a la base. Así, un **eigenvector propio** de  $X$  de  $A$  es un vector columna de  $K^n$  para el cual existe  $\lambda \in K$  tal que  $AX = \lambda X$ , véase [3].

**Definición.** Sea  $A$  una matriz real simétrica de  $n \times n$  asociada a la parte principal de la EDP ( $\sum_{i,j=1}^n a_{ij} u_{x_i x_j}$ ), entonces

- Si todos los eigenvalores de la matriz  $A$  son distintos de cero y además del mismo signo, entonces se dice que la ecuación es del tipo **Elíptico**.
- Si uno y sólo uno de los eigenvalores de la matriz  $A$  es igual a cero, entonces se dice la ecuación es del tipo **Parabólico**.
- Si todos los eigenvalores de la matriz  $A$  son distintos de cero y además  $n - 1$  de ellos tienen el mismo signo, entonces se dice que la ecuación es del tipo **Hiperbólico**.

Algunos ejemplos de EDP según el tipo:

Ecuación		Tipo
Ecuación de Laplace	$\Delta u = 0$	Elíptica
Ecuación de Poisson	$\Delta u = f$	Elíptica
Ecuación de calor	$u_t + \Delta u = f$	Parabólica
Ecuación de onda	$u_{tt} + \Delta u = f$	Hiperbólica

### 1.3. Problemas Bien Planteados

En general, cada ecuación diferencial parcial tiene varias soluciones, entonces para plantear problemas que tengan solución única, es necesario complementar el planteamiento con condiciones de frontera adecuadas y con condiciones iniciales. Un problema se considera “bien planteado” cuando este posee una solución única, véase [11]

### Problema de valores en la frontera

Para el laplaciano consideremos el dominio  $\Omega \in \mathbb{R}^n$  con frontera  $\partial\Omega$ . Entonces el caso más general de un problema con valores en la frontera es: dadas las funciones  $f_\Omega$  y  $g_\partial$  ( $f_\Omega$  definida en  $\Omega$  y  $g_\partial$  en  $\partial\Omega$ )

$$\begin{cases} \Delta u & = f_\Omega \\ \alpha \frac{\partial u}{\partial n} + \beta u & = g_\partial \end{cases}$$

donde  $\alpha$  y  $\beta$  son funciones. Este tipo de fronteras son llamadas del tipo Robin. Los casos particulares para ésta condición ocurren y han sido estudiados con mucha atención. Si  $\alpha = 0$ , lo que se obtiene es el problema de Dirichlet

$$\begin{cases} \Delta u & = f_\Omega \\ \beta u & = g_\partial \end{cases}$$

y cuando  $\beta = 0$  se obtiene el problema de Neumann

$$\begin{cases} \Delta u & = f_\Omega \\ \alpha \frac{\partial u}{\partial n} & = g_\partial \end{cases}$$

### Problema de valores en la frontera con condición inicial

Para formular este tipo de problemas, se considera el dominio  $\Omega \times [0, T]$  con  $T > 0$ , donde  $[0, T]$  es un intervalo en la recta real y, generalmente es el tiempo.

### Ecuación de Calor

Para la ecuación de calor, además de las condiciones en la frontera, la solución debe de satisfacer las condiciones iniciales

$$u(\underline{x}, 0) = u_0(\underline{x}) \quad \forall \underline{x} \in \Omega$$

las condiciones en la frontera son esencialmente las mismas que para el problema de valores en la frontera que se consideraron en la ecuación de Laplace. De esta manera, el problema más general para el caso de un problema en la frontera con condición inicial para la ecuación del calor consiste en, dadas las funciones  $u_o \in \Omega$ ,  $f_\Omega \in [0, T] \times \Omega$  y  $g_\Omega \in [0, T] \times \partial\Omega$  es

$$\begin{cases} \frac{\partial u}{\partial t} - \Delta u & = f_\Omega & (0, T) \times \Omega \\ \alpha \frac{\partial u}{\partial n} + \beta u & = g_\Omega & (0, T) \times \partial\Omega \\ u(\underline{x}, 0) & = u_0(\underline{x}) & \forall \underline{x} \in \Omega \end{cases}$$

### Ecuación de Onda

Para la ecuación de onda además de las condiciones de frontera, la solución debe de satisfacer las condiciones iniciales

$$\begin{cases} u(\underline{x}, 0) & = w(\underline{x}) \\ \frac{\partial u}{\partial t}(\underline{x}, 0) & = z(\underline{x}) \end{cases}$$

para todo  $\underline{x} \in \Omega$ . Véase [5, 6, 9]

Las condiciones de frontera son de manera general idénticas a las de la ecuación del calor. Así, el problema más general del problema de valores en la frontera con condición inicial para la ecuación de onda consiste de, dadas las funciones  $u_0, u'_0, f_\Omega$  y  $g_\Omega$ , encontrar la función que cumpla

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} - \Delta u = f_\Omega & \Omega \times (0, T) \\ \alpha \frac{\partial u}{\partial n} + \beta u = g_\partial & \partial\Omega \times (0, T) \end{cases}$$

junto con sus condiciones iniciales.

### Condiciones de Frontera

Las condiciones de frontera o condiciones de contorno de una ecuación diferencial ordinaria o parcial se suelen clasificar en tres tipos:

- Dirichlet, se especifican los valores de la solución que necesita la frontera del dominio.
- Neumann, se le especifican los valores de la derivada de una solución tomada sobre la frontera.
- Robin, es una combinación de las anteriores, podemos tener una parte con condición Dirichlet y otra parte de la frontera con condición Neumann.

## 1.4. Ecuación de Laplace, Calor y Onda

Como se mencionó, las ecuaciones de Laplace, calor y onda son los prototipos de ecuaciones elípticas, parabólicas e hiperbólicas respectivamente. En esta sección se mostrará una manera de resolver estas ecuaciones de manera analítica o semi-analítica, en capítulo cinco se muestra la resolución de estas ecuaciones en forma numérica usando el método de diferencias finitas, el cual se describe a detalle y se muestra su uso en el presente trabajo.

### Ecuación de Laplace

La ecuación de Laplace es el prototipo para una ecuación diferencial parcial del tipo elíptico. Para la ecuación de Laplace existe una terminología especializada, pues las soluciones de  $\Delta u = 0$  se les llama *funciones armónicas*, véase [9, 5]. Las funciones armónicas aparecen en situaciones variadas, por ejemplo, en electrostática, en análisis complejo y en el movimiento browniano.

Las funciones armónicas son invariantes bajo ciertas transformaciones rígidas, como rotaciones y traslaciones. Si se definen las nuevas coordenadas  $X' = X + A$  entonces  $\Delta_x u = \Delta_{x'} u$ , sin importar la dimensión de  $X$ . Decimos que  $f(x)$  es radial si  $f(x) = f(|x|)$ , es decir, si es constante en esferas concéntricas. Para hallar las funciones armónicas que son invariantes bajo rotaciones, o lo que es igual radiales, consideraremos los casos para  $n = 2, 3$ .

### Coordenadas Polares

Si se hace el cambio de coordenadas

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sen\alpha \\ -\sen\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

se seguirá cumpliendo que  $u_{xx} + u_{yy} = u_{x'x'} + u_{y'y'}$ . Usando coordenadas polares

$$\begin{aligned} x &= r \cos\theta \\ y &= r \sen\theta \end{aligned}$$

entonces tenemos que

$$\begin{aligned} \frac{\partial^2}{\partial x^2} &= \cos^2\theta \frac{\partial^2}{\partial r^2} - 2 \left( \frac{\text{sen}\theta \cos\theta}{r} \right) \frac{\partial^2}{\partial r \partial \theta} \\ &+ \frac{\text{sen}^2\theta}{r^2} \frac{\partial^2}{\partial \theta^2} + 2 \left( \frac{\text{sen}\theta \cos\theta}{r} \right) \frac{\partial}{\partial \theta} + \frac{\text{sen}^2\theta}{r} \frac{\partial}{\partial r} \\ \frac{\partial^2}{\partial y^2} &= \text{sen}^2\theta \frac{\partial^2}{\partial r^2} - 2 \left( \frac{\text{sen}\theta \cos\theta}{r} \right) \frac{\partial^2}{\partial r \partial \theta} \\ &+ \frac{\cos^2\theta}{r^2} \frac{\partial^2}{\partial \theta^2} + 2 \left( \frac{\text{sen}\theta \cos\theta}{r} \right) \frac{\partial}{\partial \theta} + \frac{\cos^2\theta}{r} \frac{\partial}{\partial r} \end{aligned}$$

sumando obtenemos

$$\Delta = \frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} + \frac{1}{r} \frac{\partial^2}{\partial \theta^2}$$

el cual define el laplaciano en coordenadas polares.

Si  $u$  es una función armónica y radial, entonces se cumplirá que

$$u_{rr} + \frac{1}{r} u_r = 0$$

es decir  $(ru_r)_r = 0$ . De esto se tiene que una solución radial en  $\mathbb{R}^2$  tendrá la forma

$$u(r, \theta) = c_1 \log(r) + c_2$$

a la cual se le llama *solución radial fundamental*, véase [9, 5].

### Coordenadas Esféricas

Para las coordenadas esféricas consideramos

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2} \\ x &= r \cos\theta \sin\phi \\ y &= r \text{sen}\theta \text{sen}\phi \\ z &= r \cos\phi \end{aligned}$$

tendríamos que el operador laplaciano corresponde a

$$\begin{aligned} \Delta u &= \frac{1}{r} \frac{\partial}{\partial r} (r^2 u_r) + \frac{1}{r^2 \text{sen}\theta} \frac{\partial}{\partial \theta} (u_\theta \text{sen}\theta) \\ &+ \frac{1}{r^2 \text{sen}^2\theta} u_{\phi\phi} \end{aligned}$$

si buscáramos las funciones armónicas invariantes bajo rotaciones(radiales), tendríamos

$$u_{rr} + \frac{2}{r} u_r = 0$$

por lo que

$$(r^2 u_r)_r = 0$$

entonces  $u = -\frac{c_1}{r} + c_2$  es solución y se le llama *solución radial fundamental* en  $\mathbb{R}^3$ .

Nótese que en ambos casos la solución radial fundamental tiene una singularidad en el origen, véase [9, 8, 5, 6].

## Ecuación de Calor

La ecuación del calor es el prototipo de ecuación diferencial parcial del tipo parabólico. Consideremos la distribución de la temperatura  $u(x, t)$  en una barra homogénea de longitud  $L$ , con temperatura inicial  $f(x)$  y con temperatura igual a cero en los extremos. Entonces la ecuación que describe lo anterior es

$$\begin{cases} u_t = ku_{xx} \\ 0 < x < L \\ t > 0 \end{cases}$$

con condiciones

$$\begin{cases} u(0, t) = 0 \\ u(L, t) = 0 \\ u(x, 0) = f(x) \end{cases}$$

para  $t \geq 0$ .

$u(x, t)$  representa la temperatura del punto  $x$  en el instante  $t$  y  $k > 0$  es una constante proporcional a la conductividad e inversamente proporcional a la densidad y al calor específico.

Tomemos

$$u(x, t) = X(x)T(t) \tag{1.4.1}$$

entonces

$$\begin{aligned} u_t(x, t) &= X(x)T'(t) \\ u_{xx}(x, t) &= X''(x)T(t) \end{aligned}$$

sustituyendo en 1.4.1 tenemos que

$$X(x)T'(t) = kX''(x)T(t)$$

si dividimos por  $kX(x)T(t)$

$$\frac{T'(t)}{kT(t)} = \frac{X''(x)}{X(x)} \tag{1.4.2}$$

Observemos que si derivamos con respecto al tiempo tenemos que

$$\frac{X''(x)}{X(x)} = 0$$

lo que implica que 1.4.2 es constante, entonces

$$\frac{T'(t)}{kT(t)} = \frac{X''(x)}{X(x)} = -\lambda \tag{1.4.3}$$

a la cual se le conoce como *constante de separación*, véase [5, 15].

De 1.4.3 obtenemos un par de ecuaciones diferenciales ordinarias

$$\begin{aligned} X''(x) + \lambda X(x) &= 0 \\ T'(t) + \lambda kT(t) &= 0 \end{aligned}$$

además de tener las condiciones de frontera

$$\begin{aligned} u(0, t) &= X(0)T(t) = 0 \\ u(L, t) &= X(L)T(t) = 0 \end{aligned}$$

esto implica que  $X(0) = X(L) = 0$ .

Se resuelve

$$X''(x) + \lambda X(x) = 0 \quad (1.4.4)$$

con las condiciones de frontera  $X(0) = X(L) = 0$ .

- Si  $\lambda = 0$ , entonces  $X''(x) = 0$ , lo que implica que

$$X(x) = ax + b$$

aplicando las condiciones

$$\begin{aligned} X(0) &= b = 0 \\ X(L) &= aL = 0 \end{aligned}$$

entonces  $a = 0$  y  $X(x) = 0$ .

- Si  $\lambda < 0$ , se define una variable  $-\alpha^2 = \lambda$ , entonces se resuelve

$$X''(x) - \alpha^2 X(x) = 0$$

y la solución es

$$X(x) = ae^{\alpha x} + be^{-\alpha x}$$

de donde tenemos que

$$\begin{aligned} X(0) &= a + b \\ \Rightarrow a &= -b \end{aligned}$$

entonces

$$\begin{aligned} X(x) &= ae^{\alpha x} - ae^{-\alpha x} \\ &= 2a \left( \frac{e^{\alpha x} - e^{-\alpha x}}{2} \right) \\ &= 2a \operatorname{senh}(\alpha x) \end{aligned}$$

Ahora si  $x = L$ , para  $L \neq 0$

$$X(L) = 2a \operatorname{senh}(\alpha L) = 0$$

lo que implica que  $a = 0$  y

$$X(x) = 0$$

- Por último, si  $\lambda > 0$ , se define  $\alpha^2 = \lambda$  y entonces

$$X''(x) + \alpha X(x) = 0$$

donde la solución general es

$$X(x) = a \cos(\alpha x) + b \operatorname{sen}(\alpha x)$$

aplicando las condiciones de frontera

$$\begin{aligned} X(0) &= a = 0 \\ X(L) &= b \operatorname{sen}(\alpha L) = 0 \end{aligned}$$

de donde si  $b = 0$  entonces  $X(x) = 0$ , por tanto  $b \neq 0$  y

$$\text{sen}(\alpha L) = 0$$

y esto pasa cuando  $\alpha_n L = n\pi$  para  $n = 1, 2, \dots$ , por tanto

$$\begin{aligned}\lambda_n &= \alpha_n^2 \\ &= \frac{n^2 \pi^2}{L}\end{aligned}$$

son los eigenvalores asociados a la ecuación diferencial 1.4.4. Y sus eigenfunciones

$$X_n(x) = \text{sen}\left(\frac{n\pi}{L}x\right)$$

Entonces, sustituyendo  $\lambda_n$  en la otra ecuación diferencial que nos queda por resolver

$$T'(t) + \lambda_n k T(t) = 0$$

obtenemos como solución

$$T_n(t) = e^{-\frac{n^2 \pi^2}{L^2} kt}$$

para  $n = 1, 2, \dots$

Entonces sea

$$\begin{aligned}u_n(x, t) &= b_n X_n(x) T_n(t) \\ &= b_n \text{sen}\left(\frac{n\pi}{L}x\right) e^{-\frac{n^2 \pi^2}{L^2} kt}\end{aligned}$$

estas funciones satisfacen la ecuación de calor con las condiciones iniciales y de frontera que se especificaron. Por el principio de superposición se tiene que

$$u(x, t) = \sum_{n=1}^{\infty} b_n \text{sen}\left(\frac{n\pi}{L}x\right) e^{-\frac{n^2 \pi^2}{L^2} kt}$$

por último debemos encontrar las  $b_n$  que

$$f(x) = u(x, 0) = b_n \text{sen}\left(\frac{n\pi}{L}x\right)$$

esta expresión es una representación mediante series de Fourier de  $f(x)$  en términos de senos, por tanto la  $b_n$  que buscamos se define como

$$b_n = \frac{2}{L} \int_0^L f(x) \text{sen}\left(\frac{n\pi}{L}x\right) dx$$

para  $n = 1, 2, \dots$  por lo que la solución a la ecuación diferencial parcial es

$$u(x, t) = \frac{2}{L} \sum_{n=1}^{\infty} \left( \int_0^L f(x) \text{sen}\left(\frac{n\pi}{L}x\right) dx \right) e^{-\frac{n^2 \pi^2}{L^2} kt} \text{sen}\left(\frac{n\pi}{L}x\right)$$

Para más detalles se puede consultar [9, 8, 5, 6].

## Ecuación de Onda

La ecuación de onda es el prototipo de ecuación diferencial parcial del tipo hiperbólico. La ecuación en una dimensión está dada por

$$u_{tt} = c^2 u_{xx}$$

que puede pensarse como el modelo matemático de una cuerda vibrando, véase[8, 6]. Para resolver en este caso se puede factorizar el operador, obteniendo

$$u_{tt} - c^2 u_{xx} = (\partial_t - c\partial_x)(\partial_t + c\partial_x)u$$

Sea  $v = u_t + cu_x$  de modo que querríamos resolver  $v_t - cv_x = 0$ . La solución general de esta ecuación es  $v(x, t) = h(x + ct)$  y habría que resolver

$$u_t + cu_x = h(x + ct) \tag{1.4.5}$$

Sea  $w(x + t) = u(x + t) - f(x + ct)$  donde  $u$  es solución de 1.4.5. Así se tiene que  $w_t + cw_x = 0$ , por lo que  $w(x + t) = g(x - ct)$  es la solución general. Entonces la solución general de la ecuación de onda es

$$u(x, t) = f(x + ct) + g(x - ct) \tag{1.4.6}$$

Consideremos ahora el problema de valores iniciales

$$\begin{cases} u_{tt} &= c^2 u_{xx} & x \in \mathbb{R}, t \in \mathbb{R} \\ u(x, 0) &= \varphi(x) & x \in \mathbb{R} \\ u_t(x, 0) &= \phi(x) & x \in \mathbb{R} \end{cases}$$

con  $\varphi$  y  $\phi$  describiendo la posición y velocidad inicial del sistema. Se puede suponer cierta regularidad en estas funciones. Para hallar la solución volvemos a 1.4.6 y notemos que

$$\begin{aligned} \varphi(x) &= f(x) + g(x) \\ \phi'(x) &= f'(x) + g'(x) \end{aligned}$$

y derivando con respecto a  $t$  y evaluando luego en  $t = 0$  obtenemos

$$\phi(x) = cf'(x) - cg'(x)$$

A partir de este par de ecuaciones, obtenemos que

$$\begin{aligned} f'(x) &= \frac{1}{2} \left( \varphi'(x) + \frac{\phi(x)}{c} \right) \\ g'(x) &= \frac{1}{2} \left( \varphi'(x) - \frac{\phi(x)}{c} \right) \end{aligned}$$

integrando se obtiene que

$$\begin{aligned} f(x) &= \frac{1}{2}\varphi(x) + \frac{1}{2c} \int_0^x \phi(s) ds + A \\ g(x) &= \frac{1}{2}\varphi(x) - \frac{1}{2c} \int_0^x \phi(s) ds + B \end{aligned}$$

Como  $f + g = \varphi$  entonces  $A + B = 0$  y por lo tanto, al evaluar  $f(x + ct)$  y  $g(x - ct)$  obtenemos

$$\begin{aligned} u(x, t) &= \frac{1}{2}\varphi(x + ct) + \frac{1}{2c} \int_0^{x+ct} \phi(s) ds + \frac{1}{2}\varphi(x - ct) - \frac{1}{2c} \int_0^{x-ct} \phi(s) ds \\ &= \frac{1}{2}(\varphi(x + ct) + \varphi(x - ct)) + \frac{1}{2} \int_{x-ct}^{x+ct} \phi(s) ds \end{aligned} \tag{1.4.7}$$

a la expresión 1.4.7 se le conoce como *fórmula de D’Alambert*. Véase [9, 8, 5, 6].

Desafortunadamente, mientras las ecuaciones diferenciales pueden describir gran variedad de problemas, sólo una pequeña porción de ellas pueden ser resueltas de manera exacta por funciones elementales (polinomios,  $\sin(x)$ ,  $\cos(x)$ ,  $e^x$ ) y sus combinaciones. Si la solución analítica no la conocemos, lo que queremos entonces es encontrar una aproximación a su solución. Una de las técnicas para encontrar la aproximación de la solución son los métodos numéricos para resolución de ecuaciones diferenciales parciales, con los cuales se obtienen aproximaciones numéricas de la solución en ciertos puntos del dominio de la ecuación. Estos métodos se les llama *Métodos de Discretización*.

## 1.5. Métodos de Discretización

En la búsqueda de una descripción cualitativa de un determinado fenómeno, por lo general se plantea un sistema de ecuaciones diferenciales ordinarias o parciales, válidas para cierto dominio y donde se imponen sobre este, una serie de condiciones en la frontera y de condiciones iniciales. Después de esto, el modelo matemático se considera completo, y es aquí donde aparece la mayor dificultad, dado que solamente en la forma más simple de la ecuación, con fronteras geométricas triviales puede ser resuelta en forma exacta con los métodos matemáticos que disponemos.

La idea de estos métodos consiste en aproximar las derivadas que aparecen en los problemas de ecuaciones diferenciales de forma que se reduzca a resolver un sistema algebraico de ecuaciones que representa la solución de la ecuación diferencial en algunos puntos. Una vez teniendo el sistema lineal  $Ax = b$ , se procede a resolver dicho sistema, utilizando las técnicas de álgebra lineal que nos ofrezcan mejor desempeño computacional. Entre las diferentes formas de discretización posibles están los siguientes métodos, véase [8, 12, 4]

- Método de Diferencias Finitas
- Método de Volumen Finito
- Método de Elemento Finito

### Método de Volumen Finito

El método de Volumen Finito, véase[2], divide el dominio de estudio en un número de volúmenes de control que no se traslapan, de manera que hay un volumen rodeando a cada punto de la malla. Las integrales de volumen de la ecuación diferencial parcial que contienen la divergencia son transformadas a integrales de superficie, usando el teorema de la divergencia o de Gauss. Estos términos entonces son evaluados como flujos en la superficie en cada volumen finito. Que el flujo de entrada en el volumen sea igual al flujo de salida, hacen que este método sea conservativo. Una ventaja del método es que se puede formular fácilmente en mallas no estructuradas. El método de volumen finito es muy usado para problemas de dinámica de fluidos.

### Método de Elemento Finito

El método de elemento finito, véase[1], es otra técnica numérica para aproximar la solución de ecuaciones diferenciales parciales. El dominio se divide en subdominios que no se traslapan, a los cuales se les denomina “elementos finitos”. Se usan métodos variacionales (cálculo variacional) para minimizar una función de error y producir una solución estable. El método se puede aplicar a la mayoría de los problemas de ciencia e ingeniería.

### Método de Diferencias Finitas

La técnica fundamental para los cálculos numéricos en diferencias finitas se basa en aproximar  $f(x)$  mediante un polinomio cerca de  $x = x_0$ , véase[12]. Una de las aproximaciones clásicas es mediante la serie de Taylor, la cual también nos permite, aparte de la aproximación de las derivadas, el cálculo del error en la aproximación mediante su fórmula del residuo.

Al resolver ecuaciones parciales, se analizan funciones de dos o más variables, por ejemplo,  $u(x, y)$ ,  $u(x, t)$  y  $u(x, y, t)$ . Algunas derivadas parciales (no todas) se pueden obtener de la misma forma como si fueran funciones de una variable. Por ejemplo, para una función  $u(x, y)$ , la derivada parcial  $\frac{\partial u}{\partial x}$  es, en realidad, una derivada ordinaria  $\frac{du}{dx}$  si mantenemos a  $y$  fija.

Por ejemplo, en los problemas físicos a menudo se necesita calcular el laplaciano  $\Delta u = \nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ . Utilizando una fórmula de diferencias centradas para la segunda derivada, obtenemos la forma del laplaciano en diferencias finitas

$$\begin{aligned} \nabla^2 u(x_0, y_0) &\approx \frac{u(x_0 + \Delta x, y_0) - 2u(x_0, y_0) + u(x_0 - \Delta x, y_0)}{(\Delta x)^2} \\ &+ \frac{u(x_0, y_0 + \Delta y) - 2u(x_0, y_0) + u(x_0, y_0 - \Delta y)}{(\Delta y)^2} \end{aligned}$$

Aquí el error es el máximo  $O((\Delta x)^2)$  y  $O((\Delta y)^2)$ . A menudo suele tomarse  $\Delta x = \Delta y$  y así se obtiene la aproximación estándar de diferencias finitas de cinco puntos para el laplaciano.

En esta forma de discretización es conveniente utilizar la notación matricial para analizar la ecuación diferencial parcial. Una vez discretizada la ecuación, lo que obtenemos es un sistema de ecuaciones  $Ax = b$ , donde el tamaño de la matriz y de los vectores depende del tipo de condiciones de frontera o de contorno que se le hayan dado al problema y de la partición del dominio realizada.

El método que se desarrolla en este trabajo es el *Método de Diferencias Finitas*, el cual es uno de los métodos clásicos de solución de ecuaciones diferenciales. Este método se detalla en el capítulo dos de este trabajo.

## 1.6. Métodos de Solución de Sistemas de Ecuaciones

Una vez la ecuación discretizada, como resultado de aplicar el método de diferencias finitas (también los métodos de volumen finito y elemento finito producen resultados similares), el problema de resolver la ecuación diferencial parcial de manera numérica se reduce a la solución del sistema  $Ax = b$ . Donde la matriz  $A$  es una matriz bandada (contiene muchos ceros) y dependiendo de la dimensión del problema puede ser de tres, cinco o siete bandas.

Para resolver el sistema de ecuaciones  $Ax = b$  se cuenta con una gran variedad de métodos, los cuales se dividen en: *Métodos Directos* y *Métodos Iterativos*.

Los Métodos Directos (*Véase Capítulo 3*) proporcionan la solución en un número fijo de pasos y sólo están sujetos a los errores del redondeo, estos métodos de solución se aplican a matrices densas. Los Métodos Iterativos (*Véase Capítulo 3*) como su nombre lo destaca se realizan iteraciones para aproximar la solución, y son usadas para la solución del sistema cuando la matriz es dispersa (sparse matrix).

Directos:

- Descomposición LU, *véase* sección 3.2.1
- Descomposición de Cholesky, *véase* sección 3.2.2
- Descomposición LU para matrices tridiagonales, *véase* sección 3.2.3

Iterativos:

- Gradiente Conjugado (CGM, Conjugate Gradient Method), *véase* sección 3.2.4
- GMRES (Generalized Minimal Residual), *véase* sección 3.2.5

Dependiendo de las características de la matriz  $A$ , se usa un tipo de método, por ejemplo, si:

- Matriz es simétrica: Cholesky, CGM
- Matriz no simétrica: LU, GMRES

Otro factor para tomar en cuenta es que al ser la matriz bandeda, se debe de encontrar una forma de almacenar óptimamente en RAM a la matriz  $A$ , ya que podría agotar la memoria por su tamaño, además de facilitar el cálculo que involucran la solución del sistema (básicamente la multiplicación de matriz-vector). Estos esquemas de almacenamiento continuo en memoria para los elementos no cero de la matriz, requieren, por parte del esquema, del conocimiento de la ubicación de los valores dentro de la matriz original. Existen varios métodos para el almacenamiento de los datos, entre los cuales se encuentran, Compressed Row Storage(CRS) y Compressed Diagonal Storage(CDS) (*Véase la sección 3.1*) son los que nos interesan. Puede darse el caso en que la matriz sea simétrica, lo cual reduce aún más el número de elementos a almacenar en memoria, pues podemos almacenar solamente la mitad de los valores. En el caso de que los coeficientes de la ecuación diferencial parcial sean constantes, los valores dentro de la banda son iguales y no es necesario guardar todos los valores, solamente un representante de ellos.

## 1.7. Necesidad del Cómputo Paralelo.

El cómputo en paralelo surge de la necesidad de resolver problemas complejos en tiempos razonables, donde se hace uso de las ventajas de memoria y velocidad de los procesadores y de la interconexión de varias máquinas para la distribución del cálculo y del procesamiento de los datos. Para lograrlo existen principalmente dos bibliotecas openMP y MPI.

OpenMP ([www.openmp.org](http://www.openmp.org)) permite la programación multiproceso de memoria compartida. Se compone de un conjunto de directivas de compilador, funciones de la propia biblioteca y variables de entorno. Se basa en el modelo fork-join, donde una tarea muy pesada se divide en hilos de procesamiento para después “juntar” o “recolectar” los resultados y al final unirlos, *véase [23]*.

MPI ([www.open-mpi.org](http://www.open-mpi.org)) es una librería que define una sintaxis y un grupo de funciones de paso de mensajes para explotar los sistemas con múltiples procesadores. Message Passing Interface(MPI) es una técnica de programación concurrente para la sincronización de procesos y permitir la exclusión mutua. Su principal característica es que no precisa de memoria compartida, por lo que es muy importante en programación de sistemas distribuidos. Los elementos principales que intervienen en el paso de mensajes son: el proceso que envía una tarea y el que la recibe, *véase [24]*.

Usando este tipo de cómputo se puede reducir el tiempo de ejecución de la resolución de la ecuación de forma considerable, ya que es posible mediante múltiples métodos numéricos que paralelizan la resolución del sistema algebraico asociado. *Véase apéndices A y B.*

Resumiendo, al tratar de resolver numéricamente una ecuación diferencial parcial por diferencias finitas, lo primero que hacemos es discretizar la ecuación, una vez hecho esto, se trata de resolver el sistema de ecuaciones  $Ax = b$  mediante el método(directo o iterativo) más conveniente utilizando técnicas de cómputo en paralelo y así, al final se obtendrá la solución del problema.

## 1.8. Objetivos

### Generales

Los objetivos generales de este trabajo son:

- Desarrollar la metodología para resolver ecuaciones diferenciales parciales mediante el método numérico de diferencias finitas
- Desarrollar los algoritmos necesarios para resolver las ecuaciones diferenciales parciales en una, dos y tres dimensiones
- Implementar en una aplicación computacional la solución numérica del método de diferencias finitas
- Mostrar que la implementación en paralelo de la metodología estudiada puede ser eficiente en equipos masivamente paralelos

## Particulares

Los objetivos particulares del trabajo son:

- Describir el método numérico de diferencias finitas
- Desarrollar la metodología de diferencias finitas para ecuaciones diferenciales parciales: Elípticas, Parabólicas e Hiperbólicas
- Describir el estencil general para ecuaciones diferenciales parciales en tres dimensiones
- Implementar la solución numérica del método de diferencias finitas en lenguajes como C++ y paquetes de uso generalizado como Scilab, Octave y MatLab
- Mostrar mediante ejemplos la solución numérica y su implementación computacional para resolver ecuaciones parciales en una, dos y tres dimensiones
- Mostrar mediante un ejemplo, que se puede obtener una paralelización de la metodología desarrollada, que tiene alta eficiencia en equipos paralelos

Para poder cumplir con los objetivos planteados en este trabajo en el capítulo dos y tres se desarrolla la metodología y los algoritmos para la solución de ecuaciones diferenciales parciales. En el capítulo cinco se muestran las aplicaciones de esta metodología, resolviendo problemas en espacio y tiempo, así como un ejemplo de paralelización mediante descomposición de dominio. Además, los códigos de los ejemplos se colocaron en la página web <http://mmc2.geofisica.unam.mx/omb>.

## Capítulo 2

# Método de Diferencias Finitas

En este capítulo se presenta una introducción general al Método de Diferencias Finitas (véase [12, 17]) para aproximar la solución de ecuaciones diferenciales. El marco teórico se plantea en forma general, aplicable a la solución de problemas específicos. Los tipos de problemas que se resolverán son los que generan ecuaciones diferenciales parciales (ya sean elípticas, parabólicas o hiperbólicas) para las distintas condiciones de frontera (Dirichlet, Neumann y Robin).

El método de diferencias finitas es una clásica aproximación para encontrar la solución numérica de las ecuaciones diferenciales parciales. El método consiste básicamente en que las derivadas son reemplazadas por aproximaciones en diferencias finitas, convirtiendo el problema de ecuaciones diferenciales en un problema algebraico  $Ax = b$  resoluble por métodos matriciales, véase [16].

Con la disponibilidad de poderosas computadoras, la aplicación de métodos numéricos para resolver problemas de ciencia e ingeniería se ha convertido, en la práctica, algo normal en la comunidad científica. Los métodos numéricos florecen donde los trabajos prácticos/experimentales son limitados. Cuando el trabajo experimental es complicado, la teoría de métodos numéricos auxilia en la solución de los problemas y así se puede obtener información valiosa. En ciencia, las soluciones numéricas y experimentales son vistas como complementarias una a otra. Es común que la experimentación valide o verifique el método numérico y así poder extender este último para resolver problemas más complejos. Es común encontrar que los métodos numéricos son la base en la solución de problemas de la ciencia, más aún, los avances en los métodos matemáticos y en la capacidad computacional han hecho posible el resolver problemas de las ciencias sociales, medicina, economía, etc.

### 2.1. Diferencias Finitas

El objetivo es aproximar la solución de ecuaciones diferenciales, es decir, encontrar una función (o alguna aproximación discreta a esta función) que satisfaga la relación de su(s) derivada(s) en alguna región del espacio y/o tiempo, con una serie de valores en su frontera. En general, resolver una EDP es complicado y raramente se puede encontrar una solución analítica que satisfaga el problema. El Método de Diferencias Finitas procede reemplazando las derivadas de la ecuación con una aproximación de una diferencia finita. Con esto obtenemos un sistema de ecuaciones algebraicas a resolver, algo que es eficiente resolverse utilizando una computadora.

#### Aproximaciones polinómicas.

La técnica para los cálculos de diferencias finitas se basan en las aproximaciones polinómicas a  $f(x)$  cerca de un punto  $x = x_0$ . Una aproximación a  $f(x)$  se puede obtener mediante su recta tangente en  $x = x_0$

$$f(x) \approx f(x_0) + (x - x_0) f'(x_0)$$

una aproximación lineal ( $h = x - x_0$ ). También podemos considerar una aproximación cuadrática a  $f(x)$

$$f(x) \approx f(x_0) + hf'(x_0) + h^2 \frac{f''(x_0)}{2!}$$

Cada una de estas aproximaciones polinómicas es más precisa a medida que aumenta el grado y si  $x$  está suficientemente cerca de  $x_0$ .

### Error de truncamiento

En las aproximaciones polinómicas se obtiene directamente una fórmula del error a partir de la expresión

$$f(x) = f(x_0) + hf'(x_0) + \dots + \frac{h^n}{n!} f^{(n)}(x_0) + R_n \quad (2.1.1)$$

conocida como *serie de Taylor* (véase[20]) con residuo. El error de truncamiento tiene una fórmula similar al término  $n + 1$  de la serie, pero se evalúa en un punto intermedio, usualmente desconocido

$$R_n = \frac{h^{n+1}}{(n+1)!} f^{(n+1)}(\xi_{n+1})$$

donde  $x_0 < \xi_{n+1} < x_0 + \Delta x$ . Para que el resultado sea válido,  $f(x)$  debe de tener  $n + 1$  derivadas continuas.

## 2.2. Aproximación a la Primera Derivada

Existen distintas maneras de generar la aproximación a la primera derivada, de las cuales nos interesa aquella que ofrezca la mejor aproximación posible con el menor esfuerzo computacional. Usando la serie de Taylor 2.1.1 podemos aproximar las derivadas de varias formas, a saber, por diferencias progresivas, diferencias regresivas y diferencias centradas.

### Diferencias Progresivas

Considerando la ecuación 2.1.1 y sea  $h > 0$

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!} f''(\epsilon)$$

de ésta ecuación se obtiene la siguiente expresión para la aproximación de la primera derivada

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2!} f''(\epsilon)$$

en este caso la aproximación a  $f'(x)$  mediante *diferencias progresivas* es de primer orden, es decir,  $O(\Delta x)$ . Es común escribir la ecuación anterior como

$$f'(x) = \frac{f(x+h) - f(x)}{h} - O_p(h)$$

siendo  $O_p(h)$  el error de truncamiento, definido como

$$O_p(h) = \frac{h^2}{2!} f''(\epsilon)$$

### Diferencias Regresivas

Considerando nuevamente la ecuación 2.1.1

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!} f''(\epsilon)$$

así la aproximación a la primera derivada es

$$f'(x) = \frac{f(x) - f(x-h)}{h} - \frac{h^2}{2!} f''(\epsilon)$$

en este caso la aproximación a  $f'(x)$  mediante *diferencias regresivas* también es de primer orden  $O(h)$ . Entonces se puede escribir la derivada como

$$f'(x) = \frac{f(x) - f(x-h)}{h} - O_r(h)$$

siendo una vez más  $O_r(h) = \frac{h^2}{2!} f''(\epsilon)$  el error de truncamiento.

## Diferencias Centradas

Comparando las diferencias progresivas y regresivas observamos que el error de truncamiento es  $O(h)$ , para obtener una mejor aproximación podemos calcular la media de esas aproximaciones.

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2!} f''(x) + \frac{h^3}{3!} f'''(\epsilon_p) \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2!} f''(x) - \frac{h^3}{3!} f'''(\epsilon_r) \end{aligned}$$

restando las ecuaciones anteriores

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{h^3}{3!} (f'''(\epsilon_p) + f'''(\epsilon_r))$$

esta última expresión da lugar a la aproximación de la primera derivada mediante *diferencias centradas*

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O_c(h^2)$$

donde  $O_c(h^2)$  es el error de truncamiento, el cual es de segundo orden. Haciendo una comparación de los errores de truncamiento de *diferencias progresivas*, *diferencias regresivas* y *diferencias centradas* tenemos que

$$\lim_{h \rightarrow 0} O_c(h^2) < \lim_{h \rightarrow 0} O_p(h) = \lim_{\Delta h \rightarrow 0} O_r(h)$$

por lo tanto, la mejor aproximación a la primera derivada se logra con *diferencias centradas*.

Estas tres aproximaciones son consistentes, lo que significa que el error de truncamiento se anula cuando  $h \rightarrow 0$ .

## 2.3. Aproximación a las Derivadas en 1D, 2D y 3D

De manera análoga, se pueden obtener aproximaciones en diferencias finitas de derivadas de orden mayor. En esta parte se desarrollará la manera de calcular la derivadas de orden uno y dos con diferencias centradas en una, dos y tres dimensiones. Para encontrar la aproximación en dos y tres dimensiones observemos que cada una de las derivadas parciales es en realidad una derivada ordinaria si mantenemos fijas las demás variables.

### Derivada de Orden Uno en Una Dimensión

Partiendo del desarrollo de la *serie de Taylor 2.1.1* se tiene que

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2!} f''(x) + \frac{h^3}{3!} f'''(\epsilon_p) \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2!} f''(x) - \frac{h^3}{3!} f'''(\epsilon_r) \end{aligned}$$

restando las ecuaciones anteriores obtenemos

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{h^3}{3!} (f'''(\epsilon_p) + f'''(\epsilon_r))$$

de donde esta última expresión da lugar a la aproximación de la primera derivada

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \quad (2.3.1)$$

## Derivada de Orden Dos en Una Dimensión

Partiendo del desarrollo de la *serie de Taylor 2.1.1* se tiene que

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2!} f''(x) + \frac{h^3}{3!} f'''(x) + \frac{h^4}{4!} f^{iv}(\epsilon_p) \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2!} f''(x) - \frac{h^3}{3!} f'''(x) + \frac{h^4}{4!} f^{iv}(\epsilon_p) \end{aligned}$$

sumando éste par de ecuaciones para obtener la aproximación a la segunda derivada

$$f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} - \frac{h^2}{12} f^{iv}(\epsilon_c)$$

así la aproximación a la segunda derivada usando *diferencias centradas*, con error de truncamiento  $O_c(h^2)$  es

$$f''(x) \approx \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} \quad (2.3.2)$$

## Derivada de Orden Uno en Dos Dimensiones

Utilizando 2.3.1 se obtendrá la aproximación a las derivadas parciales en dos variables  $x$  y  $y$ , lo cual se hace manteniendo una de las variables fija. Sea  $f(x, y) \in C^1$ .

Si se mantiene  $y$  fija se tiene que

$$\frac{\partial f}{\partial x} \approx \frac{f(x+h_1, y) - f(x-h_1, y)}{2h_1}$$

Si ahora se mantiene  $x$  fija se obtiene lo siguiente

$$\frac{\partial f}{\partial y} \approx \frac{f(x, y+h_2) - f(x, y-h_2)}{2h_2}$$

## Derivada de Orden Dos en Dos Dimensiones

Utilizando 2.3.2 se obtendrá la aproximación a las segundas derivadas parciales en dos variables  $x$  y  $y$ , lo cual se hace manteniendo una de las variables fija. Sea  $f(x, y) \in C^2$ .

Si se mantiene  $y$  fija se tiene que

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f(x-h_1, y) - 2f(x, y) + f(x+h_1, y)}{h_1^2}$$

Si ahora se mantiene  $x$  fija se obtiene lo siguiente

$$\frac{\partial^2 f}{\partial y^2} \approx \frac{f(x, y-h_2) - 2f(x, y) + f(x, y+h_2)}{h_2^2}$$

## Derivada de Orden Uno en Tres Dimensiones

Utilizando 2.3.1 se obtendrá la aproximación a las derivadas parciales en dos variables  $x, y, z$ , lo cual se hace manteniendo una de las variables fija. Sea  $f(x, y) \in C^1$ .

Si se mantienen  $y, z$  fijas se tiene que

$$\frac{\partial f}{\partial x} \approx \frac{f(x + h_1, y, z) - f(x - h_1, y, z)}{2h_1}$$

Si ahora se mantienen  $x, z$  fijas

$$\frac{\partial f}{\partial y} \approx \frac{f(x, y + h_2, z) - f(x, y - h_2, z)}{2h_2}$$

Si ahora se mantienen  $x, y$  fijas

$$\frac{\partial f}{\partial z} \approx \frac{f(x, y, z + h_3) - f(x, y, z - h_3)}{2h_3}$$

## Derivada de Orden Dos en Tres Dimensiones

Nuevamente usando 2.3.2 se obtendrá la aproximación a las segundas derivadas parciales en tres variables  $x, y, z$ . Lo que se hace es mantener dos de las tres variables fijas. Sea  $f(x, y, z) \in C^2$ .

Si mantenemos  $y, z$  fijas se tiene que

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f(x - h_1, y, z) - 2f(x, y, z) + f(x + h_1, y, z)}{h_1^2}$$

Ahora se mantienen fijas  $x, z$

$$\frac{\partial^2 f}{\partial y^2} \approx \frac{f(x, y - h_2, z) - 2f(x, y, z) + f(x, y + h_2, z)}{h_2^2}$$

y por último se mantiene fijas  $x, y$

$$\frac{\partial^2 f}{\partial z^2} \approx \frac{f(x, y, z - h_3) - 2f(x, y, z) + f(x, y, z + h_3)}{h_3^2}$$

### 2.4. Estencil para Problemas en 3D

El estencil para los problemas de una y dos dimensiones pueden obtenerse como caso particular del de tres dimensiones, haciendo ceros los terminos que no deben de aparecer, por lo cual, trataremos en esta sección unicamente el estencil para un problema tridimensional.

Para crear la malla se divide el dominio de la ecuación diferencial  $\Omega \in \mathbb{R}^3$  en  $N_x, N_y$  y  $N_z$  nodos homogéneamente espaciados por cada dimensión. Usando lo desarrollado en diferencias finitas para las parciales en tres dimensiones, tenemos que para el  $n$ -ésimo nodo  $\underline{x}_i = (x_i, y_j, z_k)$  (fig 2.4.1) se tiene que

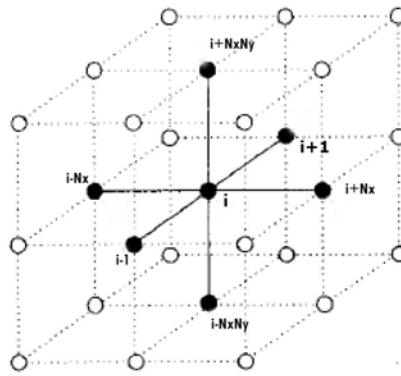


Figura 2.4.1: Nodos

$$\begin{aligned}
 \frac{\partial f}{\partial x}(x_i, y_j, z_k) &\approx \frac{f(x_i + h_1, y_j, z_k) - f(x_i - h_1, y_j, z_k)}{2h_1} \\
 &\approx \frac{f_{i+1, j, k} - f_{i-1, j, k}}{2h_1} \\
 &\approx \frac{f_{n+1} - f_{n-1}}{2h_1} \\
 \frac{\partial f}{\partial y}(x_i, y_j, z_k) &\approx \frac{f(x_i, y_j + h_2, z_k) - f(x_i, y_j - h_2, z_k)}{2h_2} \\
 &\approx \frac{f_{i, j+1, k} - f_{i, j-1, k}}{2h_2} \\
 &\approx \frac{f_{n+N_x} - f_{n-N_x}}{2h_2} \\
 \frac{\partial f}{\partial z}(x_i, y_j, z_k) &\approx \frac{f(x_i, y_j, z_k + h_3) - f(x_i, y_j, z_k - h_3)}{2h_3} \\
 &\approx \frac{f_{i, j, k+1} - f_{i, j, k-1}}{2h_3} \\
 &\approx \frac{f_{n+N_x N_y} - f_{n-N_x N_y}}{2h_3}
 \end{aligned}$$

y para las segundas parciales

$$\begin{aligned}
 \frac{\partial^2 f}{\partial x^2}(x_i, y_j, z_k) &\approx \frac{f(x_i - h_1, y_j, z_k) - 2f(x_i, y_j, z_k) + f(x_i + h_1, y_j, z_k)}{h_1^2} \\
 &\approx \frac{f_{i-1,j,k} - 2f_{i,j,k} + f_{i+1,j,k}}{h_1^2} \\
 &\approx \frac{f_{n-1} - 2f_n + f_{n+1}}{h_1^2} \\
 \frac{\partial^2 f}{\partial y^2}(x_i, y_j, z_k) &\approx \frac{f(x_i, y_j - h_2, z_k) - 2f(x_i, y_j, z_k) + f(x_i, y_j + h_2, z_k)}{h_2^2} \\
 &\approx \frac{f_{i,j-1,k} - 2f_{i,j,k} + f_{i,j+1,k}}{h_2^2} \\
 &\approx \frac{f_{n-N_x} - 2f_n + f_{n+N_x}}{h_2^2} \\
 \frac{\partial^2 f}{\partial z^2}(x_i, y_j, z_k) &\approx \frac{f(x_i, y_j, z_k - h_3) - 2f(x_i, y_j, z_k) + f(x_i, y_j, z_k + h_3)}{h_3^2} \\
 &\approx \frac{f_{i,j,k-1} - 2f_{i,j,k} + f_{i,j,k+1}}{h_3^2} \\
 &\approx \frac{f_{n-N_x N_y} - 2f_n + f_{n+N_x N_y}}{h_3^2}
 \end{aligned}$$

donde  $h_1, h_2, h_3$  son las distancias entre nodos en los ejes  $x, y, z$  respectivamente.

## 2.5. Condiciones de frontera

**Dirichlet.** Se especifica el valor que la función  $u(\underline{x}, t)$  toma en la frontera, es decir, es un valor conocido en el nodo o nodos de la frontera, este valor pasa al lado derecho del sistema  $Ax = b$ . No se necesita hacer más manipulación algebraica. Véase sección 2.6.1.

**Neumann.** En la condición de frontera tipo Neumann, lo que se conoce es la derivada de  $u(\underline{x}, t)$  a lo largo de la frontera  $\partial\Omega$ , es decir,  $\frac{\partial u}{\partial n}$ . La condición debe representarse como una diferencia finita añadiendo puntos ficticios a la malla en la frontera. Estos valores añadidos, no tienen ningún significado físico pues se encuentran por fuera del dominio del problema.

Por ejemplo, para un problema unidimensional en un dominio  $\Omega = (a, b)$  y condición de frontera Neumann en los puntos  $x = a$  y  $x = b$  igual a  $\frac{du_1}{dn} = g_1(x)$  y  $\frac{du_2}{dn} = g_2(x)$ , al usar el método del punto fantasma o ficticio, se añaden los puntos  $x_{-1} = x_0 - h = a - h$  y  $x_{n+1} = x_n + h = b + h$  a la malla. Entonces ahora se puede usar el método de diferencias finitas centradas para resolver la ecuación diferencial en todos los puntos de la malla donde la solución es desconocida. Se tiene entonces un sistema de  $n$  ecuaciones y  $n + 2$  incógnitas. Las ecuaciones adicionales para completar el sistema son las que corresponden a las condiciones de frontera Neumann.

Una primera aproximación es representar la condición como una diferencia progresiva

$$\begin{aligned}
 \frac{u_1 - u_0}{h} &= \frac{du_1}{dn} \\
 \frac{u_n - u_{n-1}}{h} &= \frac{du_2}{dn}
 \end{aligned}$$

para las condiciones en  $a$  y  $b$  respectivamente. Sin embargo, la aproximación a una derivada por diferencias progresivas es de primer orden. Teniendo en cuenta que la aproximación del método de diferencias finitas que se ha desarrollado es de segundo orden, necesitamos expresar tal condición de frontera como una aproximación

del mismo orden. Entonces, representando las derivadas como una diferencia finita centrada

$$\begin{aligned}\frac{u_1 - u_{-1}}{2h} &= \frac{du_1}{dn} = g_1(x) \\ \frac{u_{n+1} - u_{n-1}}{2h} &= \frac{du_2}{dn} = g_2(x)\end{aligned}$$

De las ecuaciones anteriores se tiene que  $u_{-1} = u_1 - 2hg_1$  y  $u_{n+1} = u_{n-1} + 2hg_2$ . Colocando esto en la discretización de diferencias finitas para los puntos  $x = a$  y  $x = b$  se tiene que

$$\begin{aligned}\frac{u_{-1} - 2u_0 + u_1}{h^2} &= f_0 \\ \frac{u_1 - 2hg_1 - 2u_0 + u_1}{h^2} &= f_0 \\ \frac{-u_0 + u_1}{h^2} &= \frac{f_0}{2} + \frac{g_1}{h}\end{aligned}$$

para el punto  $x = a$  y

$$\begin{aligned}\frac{u_{n-1} - 2u_n + u_{n+1}}{h^2} &= f_n \\ \frac{u_{n-1} - 2u_n + 2hg_2 + u_{n-1}}{h^2} &= f_n \\ \frac{u_{n-1} - u_n}{h^2} &= \frac{f_n}{2} + \frac{g_2}{h}\end{aligned}$$

para el punto  $x = b$ .

Con lo cual se tiene una aproximación de segundo orden para las fronteras tipo Neumann. Véase sección 2.6.2.

**Robin.** Al ser una combinación de las anteriores, se aplica la misma regla dependiendo de la condición de frontera en la que se encuentre el nodo. Véase sección 2.6.3.

Como referencia véase [25] capítulo 4.

## 2.6. Procedimiento General del Método de Diferencias Finitas

Dada la ecuación diferencial parcial, para poder utilizar el *Método de Diferencias Finitas* se debe de:

- Generar una malla del dominio, es decir, un conjunto de puntos en los cuales se buscará la solución aproximada a la ecuación diferencial parcial. Ver figura 2.6.1.
- Sustituir las derivadas correspondientes con alguna de las fórmulas de diferencias finitas centradas, para obtener un sistema algebraico de ecuaciones  $Ax = b$ .
- Resolver el sistema de ecuaciones para obtener la solución aproximada en cada punto de la malla.

A continuación se darán ejemplos de como proceder para resolver una ecuación diferencial mediante el método de diferencias finitas en una dimensión, tomando en cuenta las distintas fronteras que puede tener un problema, a saber, Dirichlet, Neumann y Robin.









sustituyendo 2.7.1 y 2.7.2 en 2.7.3

$$\frac{\partial}{\partial x} \left( a \frac{\partial u}{\partial x} \right) \approx \frac{a \left( x + \frac{h_1''}{2} \right) \frac{u(x + h_1'') - u(x)}{h_1''} - a \left( x - \frac{h_1'}{2} \right) \frac{u(x) - u(x - h_1')}{h_1'}}{\left( \frac{h_1' + h_1''}{2} \right)}$$

ésta aproximación es de segundo orden a  $\frac{\partial}{\partial x} \left( a \frac{\partial u}{\partial x} \right)$  en  $h_1$ , donde  $h_1 = \max\{h_1', h_1''\}$ .

## 2.8. Discretización del tiempo

Hasta ahora se ha visto como discretizar la parte espacial de las ecuaciones diferenciales parciales, lo cual nos permite encontrar la solución estática de los problemas. Sin embargo para ecuaciones del tipo parabólico e hiperbólico, las cuales dependen del tiempo, se necesita introducir una discretización en las derivadas con respecto al tiempo. Al igual que con las discretizaciones espaciales, podemos utilizar algún esquema de diferencias finitas en la discretización del tiempo.

### 2.8.1. Ecuaciones Con Primera Derivada Temporal (Esquema Theta)

Para la solución de ecuaciones diferenciales con derivada temporal ( $u_t$ ), se emplean diferentes esquemas en diferencias finitas para la discretización del tiempo. Estos esquemas se conocen de manera general como esquemas *theta* ( $\theta$ ), véase [19]

Si se define la ecuación diferencial parcial general de segundo orden como en 1.2.2  $\mathcal{L}u = \sum_{i,j=1}^n a_{ij}u_{x_i x_j} + \sum_{i=1}^n b_i u_{x_i} + cu$ , entonces la ecuación parabólica tiene la forma

$$\begin{aligned} u_t &= \mathcal{L}u \\ &= au_{xx} + bu_{yy} + cu_{zz} + du_x + eu_y + fu_z + gu + h \end{aligned}$$

entonces, el esquema  $\theta$  está dado por

$$u_t = (1 - \theta) (\mathcal{L}u)^j + \theta (\mathcal{L}u)^{j+1}$$

Diferentes casos del esquema *theta*:

- Para  $\theta = 0$  se obtiene un esquema de diferencias finitas hacia adelante en el tiempo, conocido como esquema *completamente explícito*, ya que el paso  $n + 1$  se obtiene de los términos del paso anterior  $n$ . Es un esquema sencillo, el cual es condicionalmente estable cuando  $k \leq \frac{h^2}{2}$  donde  $h = \Delta x$  y  $k = \Delta t$ .
- Para  $\theta = 1$  se obtiene el esquema de diferencias finitas hacia atrás en el tiempo, conocido como esquema *completamente implícito*, el cual es incondicionalmente estable.
- Para  $\theta = \frac{1}{2}$  se obtiene un esquema de diferencias finitas centradas en el tiempo, conocido como esquema *Crank-Nicolson*, este esquema también es incondicionalmente estable y es el más usado por tal razón.

Entonces en el esquema de *Crank-Nicolson* se toman una diferencia progresiva para el tiempo y se promedian las diferencias progresivas y regresivas en el tiempo para las derivadas espaciales.

Entonces, si tenemos la ecuación

$$\begin{aligned} u_t &= \mathcal{L}u \\ &= au_{xx} + bu_{yy} + cu_{zz} + du_x + eu_y + fu_z + gu + h \end{aligned}$$

donde los coeficientes pueden depender de  $(x, y, z, t)$ . Las discretizaciones correspondientes son

$$u_t \approx \frac{u_i^{j+1} - u_i^j}{k}$$

para el tiempo, donde  $k = \Delta t$  y

$$\begin{aligned} au_{xx} &\approx \frac{a}{2} \left[ \frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h_1^2} + \frac{u_{i-1}^{j+1} - 2u_i^{j+1} + u_{i+1}^{j+1}}{h_1^2} \right] \\ bu_{yy} &\approx \frac{b}{2} \left[ \frac{u_{i-n1}^j - 2u_i^j + u_{i+n1}^j}{h_2^2} + \frac{u_{i-n1}^{j+1} - 2u_i^{j+1} + u_{i+n1}^{j+1}}{h_2^2} \right] \\ cu_{zz} &\approx \frac{c}{2} \left[ \frac{u_{i-n1n2}^j - 2u_i^j + u_{i+n1n2}^j}{h_3^2} + \frac{u_{i-n1n2}^{j+1} - 2u_i^{j+1} + u_{i+n1n2}^{j+1}}{h_3^2} \right] \\ du_x &\approx \frac{d}{2} \left[ \frac{u_{i+1}^j - u_{i-1}^j}{2h_1} + \frac{u_{i+1}^{j+1} - u_{i-1}^{j+1}}{2h_1} \right] \\ eu_y &\approx \frac{e}{2} \left[ \frac{u_{i+n1}^j - u_{i-n1}^j}{2h_2} + \frac{u_{i+n1}^{j+1} - u_{i-n1}^{j+1}}{2h_2} \right] \\ fu_z &\approx \frac{f}{2} \left[ \frac{u_{i+n1n2}^j - u_{i-n1n2}^j}{2h_3} + \frac{u_{i+n1n2}^{j+1} - u_{i-n1n2}^{j+1}}{2h_3} \right] \end{aligned}$$

además de  $gu_i^j$  y  $h_i^j$  para el espacio.

Entonces una vez que se sustituyen las derivadas por su forma en diferencias finitas lo que sigue es formar el sistema  $Au^{j+1} = Bu^j + h^j$ , esto se logra, colocando del lado izquierdo de la igualdad los términos que contengan el paso del tiempo correspondiente a  $j + 1$  y del lado derecho a los correspondientes términos de  $j$ .

### 2.8.2. Ecuaciones Con Segunda Derivada Temporal

En las ecuaciones diferenciales parciales donde se requiera segunda derivada temporal (por ejemplo, la ecuación de onda), esta se aproxima por diferencias centradas en el tiempo

$$u_{tt} \approx \frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{k^2}$$

donde  $k = \Delta t$  es la partición del intervalo del tiempo. Mientras las demás derivadas parciales se sustituyen por su aproximación de diferencias finitas. Si

$$\begin{aligned} u_{tt} &= \mathcal{L}u \\ &= au_{xx} + bu_{yy} + cu_{zz} + du_x + eu_y + fu_z + hu + g \end{aligned}$$

donde los coeficientes pueden depender de  $(x, y, z, t)$ , las discretizaciones correspondientes son

$$u_{tt} \approx \frac{u_i^{j-1} - 2u_i^j + u_i^{j+1}}{k^2}$$

para el tiempo y

$$\begin{aligned}
 au_{xx} &\approx a \left[ \frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h_1^2} \right] \\
 bu_{yy} &\approx b \left[ \frac{u_{i-n_1}^j - 2u_i^j + u_{i+n_1}^j}{h_2^2} \right] \\
 cu_{zz} &\approx c \left[ \frac{u_{i-n_1n_2}^j - 2u_i^j + u_{i+n_1n_2}^j}{h_3^2} \right] \\
 du_x &\approx d \left[ \frac{u_{i+1}^j - u_{i-1}^j}{2h_1} \right] \\
 eu_y &\approx e \left[ \frac{u_{i+n_1}^j - u_{i-n_1}^j}{2h_2} \right] \\
 fu_z &\approx f \left[ \frac{u_{i+n_1n_2}^j - u_{i-n_1n_2}^j}{2h_3} \right]
 \end{aligned}$$

además de  $hu_i$  y  $g_i$  para el espacio. Al ser los errores de aproximación de segundo orden en espacio  $O(h_i^2)$  y en tiempo  $O(k^2)$ , se espera que las discretizaciones sean  $k \approx h_i$ .

Entonces una vez que se sustituyen las derivadas por su forma en diferencias finitas lo que sigue es formar el sistema  $u^{j+1} = 2u_i^j - u_i^{j-1} + k^2 B u^j$ , esto se logra, colocando del lado izquierdo de la igualdad los términos que contengan el paso del tiempo correspondiente a  $j+1$  y del lado derecho a los correspondientes términos de  $j$  y  $j-1$ .

$$\begin{aligned}
 u_{tt} &= \mathcal{L}u^j \\
 \frac{u_i^{j-1} - 2u_i^j + u_i^{j+1}}{k^2} &= \mathcal{L}u^j \\
 u_i^{j+1} &= 2u_i^j - u_i^{j-1} + k^2 \mathcal{L}u^j
 \end{aligned} \tag{2.8.1}$$

Los errores cometidos en la discretización son de orden cuadrático, pues se ha utilizado la diferencia central en tres puntos, tanto para el espacio como para el tiempo.

En la ecuación 2.8.1 para calcular  $u_i^{j+1}$  es necesario conocer  $u_{i-1}, u_i, u_{i+1}$  en los dos instantes inmediatos anteriores  $t_j, t_{j-1}$ . En particular para calcular  $u_i^1$  es necesario conocer  $u_i^0$  y  $u_i^{-1}$ . Entonces si  $u(\underline{x}, 0) = u_0$  es la condición inicial y  $\frac{du}{dt} = u_1(\underline{x}, 0)$  es la condición inicial de la primer derivada temporal, se tiene que

$$u_i^1 = 2u_i^0 - u_i^{-1} + k^2 \mathcal{L}u^0$$

donde tomamos

$$\begin{aligned}
 u_i^0 &= u_0(x_i) \\
 \frac{u_i^1 - u_i^{-1}}{2k} &= u_1(x_i)
 \end{aligned}$$

sustituyendo

$$u_i^1 = u_0(x_i) + k u_1(x_i) + k^2 \mathcal{L}u^0$$

lo cual permite calcular  $u_i^1$  a partir de las condiciones iniciales. Para el cálculo de  $u_i^{j+1}$  con  $j > 0$  se utiliza la fórmula 2.8.1 directamente.

### 2.8.3. Upwind

El esquema upwind (véase [8]), es un método de resolución especial para las ecuaciones hiperbólicas, en especial la ecuación de *advección*, la cual está dada por

$$u_t + au_x = 0$$

El esquema upwind más sencillo posible está dado por

$$\frac{u_i^{j+1} - u_i^j}{k} = \begin{cases} -a \left( \frac{u_i^j - u_{i-1}^j}{h} \right) & a \geq 0 \\ -a \left( \frac{u_{i+1}^j - u_i^j}{h} \right) & a < 0 \end{cases}$$

Entonces, si  $a \geq 0$  se tiene que

$$u_i^{j+1} = \frac{ak}{h} u_{i-1}^j + \left( 1 - \frac{ak}{h} \right) u_i^j$$

en caso de que  $a < 0$

$$u_i^{j+1} = \left( 1 + \frac{ak}{h} \right) u_i^j - \frac{ak}{h} u_{i+1}^j$$

En forma matricial se escribe como

$$u^{j+1} = Au^j$$

por lo que para conocer el tiempo en  $j + 1$  hay que conocer el tiempo  $j$  y, si  $j = 0$  entonces es  $u^0$  es la condición inicial del problema.

## 2.9. Consistencia, estabilidad, convergencia y error del Método de Diferencias Finitas

Cuando se usa algún método de resolución de ecuaciones diferenciales, se necesita conocer cuan exacta es la aproximación en comparación con la solución analítica (en caso de existir), véase [12, 17]

### 2.9.1. Error Global

Sea  $U = [U_1, U_2, \dots, U_n]^T$  el vector solución obtenido al utilizar el método de diferencias finitas, y  $u = [u(x_1), u(x_2), \dots, u(x_n)]$  las solución exacta en los puntos de la malla. El vector de error global se define como  $E = U - u$ . Lo que se desea es que el valor máximo sea pequeño. Usualmente se utilizan distintas normas para encontrar el error:

- La norma infinito  $\|E\|_\infty = \max_i |e_i|$ .
- La norma-1 la que se define como  $\|E\|_1 = \sum_i h_i |e_i|$
- La norma-2 la cual se define como  $\|E\|_2 = \left( \sum_i h_i |e_i|^2 \right)^{\frac{1}{2}}$

La norma infinito es en general la más apropiada para calcular los errores relativos a la discretización.

**Definición.** Un método de diferencias finitas se dice que es *convergente* si  $\lim_{h \rightarrow 0} \|E\| = 0$

### 2.9.2. Error Local de Truncamiento

Sea  $P\left(\frac{d}{dx}\right)$  un operador diferencial, por ejemplo

- Dado  $u'' = f(x)$ , entonces  $P\left(\frac{d}{dx}\right) = \frac{d^2}{dx^2}$ , y  $Pu = f$
- Si  $P\left(\frac{d}{dx}\right) = \frac{d^3}{dx^3} + a(x)\frac{d^2}{dx^2} + b(x)\frac{d}{dx} + c(x)$ , entonces  $Pu = f$  está dada por  $u''' + a(x)u'' + b(x)u' + c(x)u = f$

Sea  $P_h$  el operador de diferencias finitas. Por ejemplo, para la ecuación de segundo orden  $u''(x) = f(x)$ , el operador diferencial será

$$P_h u(x) = \frac{u(x-h) - 2u(x) + u(x+h)}{h^2}$$

Entonces, el *error local de truncamiento* se define como

$$T(x) = Pu - P_h u$$

Entonces, por ejemplo, para la ecuación diferencial  $u''(x) = f(x)$  tenemos que

$$\begin{aligned} T(x) &= Pu - P_h u \\ &= u''(x) - \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} \\ &= f(x) - \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} \end{aligned}$$

Nótese que el error local de truncamiento solo depende de la solución del stencil de diferencias finitas y no de la solución global. Es una de estas razones por la cual se le llama *error local*. El error local de truncamiento mide cuan bien la discretización de diferencias finitas aproxima a la ecuación diferencial.

**Definición.** El esquema de diferencias finitas es *consistente* si

$$\lim_{h \rightarrow 0} T(x) = \lim_{h \rightarrow 0} (Pu - P_h u) = 0$$

La consistencia no garantiza que el método de diferencias finitas trabaje. Se necesita otra condición para determinar cuando el método converge o no converge. Tal condición es la *estabilidad* del método de diferencias finitas.

**Lema.** *Lema de Hadamard*

*Si  $A$  es una matriz de diagonal estrictamente dominante, entonces  $A$  es invertible.*

**Definición.** Un método de diferencias finitas para la ecuación elíptica es estable si  $A$  es invertible y

$$\|A^{-1}\| \leq C$$

para toda  $h < h_0$ , donde  $C$  y  $h_0$  son constantes.

El siguiente teorema es importante para demostrar la convergencia de los métodos de diferencias finitas.

**Teorema.** (Teorema de Equivalencia de Lax-Richtmyer) *Un método de diferencias finitas consistente y estable es convergente.*

Usualmente es relativamente sencillo probar la consistencia, pero más complicado o hasta a veces casi imposible probar la estabilidad.

## 2.10. Estabilidad de Métodos con Paso de Tiempo

Para determinar la condición de estabilidad de los métodos con paso de tiempo, se utiliza el análisis de estabilidad de von Neumann, que puede describirse en los siguientes pasos:

Esquema discreto  $\implies$  transformada discreta de Fourier  $\implies$  crecimiento del factor  $g(\xi) \implies$  estabilidad si  $|g(\xi)| \leq 1$

### 2.10.1. Análisis de von Neumann Simplificado para métodos con Paso de Tiempo

Asúmase que se tiene un método de paso temporal  $U^{k+1} = f(U^k, U^{k+1})$ . Se tiene el siguiente teorema con el cual se puede conocer la estabilidad del esquema de diferencias finitas.

**Teorema.** Sea  $\theta = h\xi$ . Un esquema de diferencias finitas con paso de tiempo (con coeficientes constantes) es estable  $\Leftrightarrow$  existe una constante  $K$  (independiente de  $\theta, \Delta t$ , y  $h$ ) y una malla con espacios  $h_0$  y  $\Delta t_0$  tal que

$$|g(\theta, \Delta t, h)| \leq 1 + K\Delta t$$

para todo  $\theta$ ,  $0 < h \leq h_0$  y  $0 < \Delta t \leq \Delta t_0$ . Si  $g(\theta, \Delta t, h)$  es independiente de  $h$  y  $\Delta t$ , la condición para la estabilidad es

$$|g(\theta)| \leq 1$$

Este teorema muestra que para determinar la estabilidad de un esquema de diferencias finitas, se necesita considerar únicamente la amplificación del factor  $g(\xi\theta) = g(\theta)$ . Esta observación es debida a von Neumann, y es por esto, que este tipo de análisis lleva su nombre.

Para establecer la estabilidad mediante el análisis de von Neumann se siguen los siguientes pasos:

- Se sustituye  $u_j^k = e^{ijh\xi}$
- Se expresa  $u_j^{k+1} = g(\xi)e^{ijh\xi}$
- Se resuelve  $g(\xi)$  y se determina si  $|g(\xi)| \leq 1$

### Ejemplo de Estabilidad del Esquema Crank-Nicolson

Tomemos el ejemplo de la ecuación de Calor en una dimensión

$$u_t = \alpha^2 u_{xx}$$

la ecuación discretizada quedaría como

$$\frac{u_j^{k+1} - u_j^k}{\Delta t} = \mu \left[ u_{j-1}^k - 2u_j^k + u_j^k + u_{j-1}^{k+1} - 2u_j^{k+1} + u_j^{k+1} \right]$$

donde  $\mu = \frac{\alpha^2 \Delta t}{2h^2}$ .

Por lo visto en la sección anterior, para conocer la estabilidad hay que seguir una serie de pasos, entonces:

Sustituimos  $u_j^k = e^{ijh\xi}$  y  $u_j^{k+1} = g(\xi)e^{ijh\xi}$

$$g(\xi)e^{ijh\xi} - e^{ijh\xi} = \mu \left[ e^{i(j-1)h\xi} - 2e^{ijh\xi} + e^{i(j+1)h\xi} \right] + \mu \left[ g(\xi)e^{i(j-1)h\xi} - 2g(\xi)e^{ijh\xi} + g(\xi)e^{i(j+1)h\xi} \right]$$

dividiendo entre  $e^{ijh\xi}$

$$g(\xi) - 1 = \mu \left[ e^{-ih\xi} - 2 + e^{ih\xi} \right] + \mu g(\xi) \left[ e^{-ih\xi} - 2 + e^{ih\xi} \right]$$

entonces

$$g(\xi) \left( 1 - \left[ e^{-ih\xi} - 2 + e^{ih\xi} \right] \right) = 1 + \mu \left[ e^{-ih\xi} - 2 + e^{ih\xi} \right]$$

y finalmente

$$g(\xi) = \frac{1 + \mu [e^{-ih\xi} - 2 + e^{ih\xi}]}{1 - [e^{-ih\xi} - 2 + e^{ih\xi}]}$$

dado que  $e^{i\theta}$  es la exponencial compleja, tenemos

$$\begin{aligned} e^{i\theta} &= \cos\theta + i\sin\theta \\ e^{-i\theta} &= \cos\theta - i\sin\theta \end{aligned}$$

por lo que

$$g(\xi) = \frac{1 + \mu [2\cos\theta - 2]}{1 - \mu [2\cos\theta - 2]}$$

y así

$$g(\xi) = \frac{1 + 2\mu [\cos\theta - 1]}{1 - 2\mu [\cos\theta - 1]}$$

dado que  $-1 \leq \cos\theta \leq 1$ .

Entonces si  $\cos\theta = 1$

$$|g(\xi)| = 1$$

y si  $\cos\theta = -1$

$$|g(\xi)| = \left| \frac{1 - 4\mu}{1 + 4\mu} \right|$$

$\mu$  siempre es positivo, por tanto

$$|g(\xi)| = \left| \frac{1 - 4\mu}{1 + 4\mu} \right| < 1$$

entonces  $|g(\xi)| \leq 1$  para cualquier valor  $\theta, t, h$ . Por lo que Crank-Nicolson es incondicionalmente estable.

Para la ecuación de onda

$$u_{tt} = c^2 u_{xx}$$

siguiendo el mismo procedimiento se tiene que el método es estable si

$$c \leq \frac{h}{\Delta t}$$

## Capítulo 3

# Resolución de Grandes Sistemas de Ecuaciones

En el capítulo anterior se desarrolló un método para transformar un problema de ecuaciones diferenciales parciales en un sistema algebraico de ecuaciones

$$\underline{\underline{A}}x = \underline{b}$$

y así poder encontrar la solución al problema planteado. Donde  $\underline{\underline{A}}$  es una matriz bandada, es decir, contiene muchos elementos iguales a cero. Dependiendo de la dimensión del problema, ésta matriz puede ser tridiagonal, pentadiagonal o heptadiagonal, para una, dos y tres dimensiones respectivamente.

En este capítulo se estudian los diferentes métodos para resolver el sistema de ecuaciones obtenido. Los métodos para resolver el sistema de ecuaciones  $\underline{\underline{A}}x = \underline{b}$  se agrupan en *Métodos Directos* y *Métodos Iterativos*.

La elección del método específico para resolver el sistema depende de las propiedades particulares de la matriz  $\underline{\underline{A}}$ . Los *Métodos Directos* proporcionan la solución en un número fijo de pasos y sólo están sujetos a los errores del redondeo, este tipo de métodos son utilizados principalmente cuando la matriz es densa. En los *Métodos Iterativos* como su nombre lo destaca, se realizan iteraciones para aproximar la solución y son utilizados principalmente cuando la matriz es dispersa.

Cabe mencionar que la mayoría del tiempo de cómputo necesario para resolver el problema de ecuaciones diferenciales parciales se consume en la solución del sistema algebraico de ecuaciones asociado a la discretización, por ello es importante elegir aquel método que minimice el tiempo invertido en este proceso.

### 3.1. Estructura Óptima para el uso de Matrices

Parte importante en la resolución de problemas con métodos numéricos es el resolver el sistema algebraico  $\underline{\underline{A}}x = \underline{b}$  pero, también la forma de almacenar de manera óptima la matriz  $\underline{\underline{A}}$  en memoria (para problemas “reales” éstas matrices pueden llenar la memoria RAM de nuestro sistema) además de facilitar los cálculos que involucra la resolución del sistema. El sistema  $\underline{\underline{A}}x = \underline{b}$  puede ser resuelto más eficientemente si los elementos de  $\underline{\underline{A}}$  que son iguales a cero no son almacenados. Estos esquemas de almacenamiento continuo en memoria para los elementos no cero de la matriz, requieren, por parte del esquema, del conocimiento de la ubicación de los valores dentro de la matriz original. Existen varios métodos para el almacenamiento de los datos, en ésta parte se describirán **Compressed Row Storage** y **Compressed Diagonal Storage**.

#### Compressed Row Storage(CSR)

El esquema **Compressed Row Storage** es un formato de almacenamiento de datos, el cual no hace ninguna suposición sobre la dispersión de los elementos de la matriz, y no almacena ningún elemento innecesario. Sin embargo no es el más eficiente, y se necesita una dirección indirecta para cada operación en un producto matriz-vector.

En el formato **CSR** se alojan los elementos no ceros del renglon en forma subsecuente. Si se supone que tenemos una matriz dispersa no simétrica, se crean tres vectores: uno para los elementos  $a_{ij}$  de la matriz (**val**) y un par de vectores de enteros (**col\_ind**, **row\_ptr**). El vector **val** guarda los valores no ceros de la matriz. El vector **col\_ind** guarda los índices de las columnas de los elementos de **val**. Esto es, si  $val(k) = a_{ij}$  entonces  $col\_ind(k) = j$ . El vector **row\_ptr** almacena las locaciones del vector **val** donde empieza un renglon, es decir, si  $val(k) = a_{ij}$  entonces  $row\_ptr(i) \leq k < row\_ptr(i + 1)$ . Por convención, se define  $row\_ptr(n + 1) = nnz + 1$ , donde  $nnz$  es el número de valores distintos de cero en la matriz  $\underline{\underline{A}}$ . El ahorro en el almacenamiento con ésta aproximación es considerable, ya que, en lugar de almacenar  $n^2$  valores, se almacenan  $2nnz + n + 1$  valores.

Por ejemplo, consideremos la matriz no simétrica  $\underline{\underline{A}}$  definida como

$$A = \begin{pmatrix} 10 & 0 & 0 & 0 & -2 & 0 \\ 3 & 9 & 0 & 0 & 0 & 3 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 3 & 0 & 8 & 7 & 5 & 0 \\ 0 & 8 & 0 & 9 & 9 & 13 \\ 0 & 4 & 0 & 0 & 2 & -1 \end{pmatrix}$$

Entonces el formato **CSR** para la matriz es

$$val = (10 \quad -2 \quad 3 \quad 9 \quad 3 \quad \dots \quad 4 \quad 2 \quad -1)$$

$$col\_ind = (1 \quad 5 \quad 1 \quad 2 \quad 6 \quad \dots \quad 2 \quad 5 \quad 6)$$

$$row\_ptr = (1 \quad 3 \quad 6 \quad 9 \quad 13 \quad 17 \quad 20)$$

Si la matriz  $\underline{\underline{A}}$  es simétrica, solo se almacena la porción triangular superior(o inferior) de la matriz.

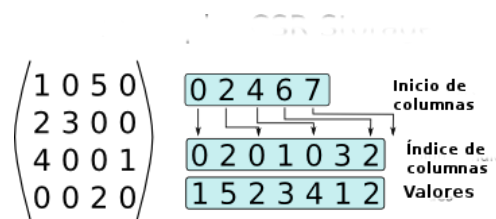


Figura 3.1.1: Ejemplo del formato CSR

### Compressed Diagonal Storage (CDS)

Si la matriz  $\underline{\underline{A}}$  es bandada con ancho de banda constante de renglón a renglón, se puede tomar ventaja de ésta estructura mediante un esquema de almacenamiento que guarda las subdiagonales de la matriz en locaciones consecutivas. Éste esquema es útil para discretizaciones que surgen de elemento finito o diferencias finitas, donde se usa una numeración estandar.

Entonces, decimos que la matriz  $\underline{\underline{A}} = (a_{ij})$  es bandada si existen dos números constantes no negativos  $p, q$  tal que  $a_{ij} \neq 0$  sólo si  $i - p \leq j \leq i + q$ . En éste caso podemos almacenar la matriz  $\underline{\underline{A}}$  en un matriz  $val = (1 : n, -p : q)$ .

Usualmente los esquemas de bandas almacenan algunos ceros, además de que el formato *CDS* puede también contener ceros que no corresponden del todo a la matriz  $\underline{\underline{A}}$ . Por ejemplo, si se considera una matriz  $\underline{\underline{A}}$  definida como

$$A = \begin{pmatrix} 10 & -3 & 0 & 0 & 0 & 0 \\ 3 & 9 & 6 & 0 & 0 & 0 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 0 & 0 & 8 & 7 & 5 & 0 \\ 0 & 0 & 0 & 9 & 9 & 13 \\ 0 & 0 & 0 & 0 & 2 & -1 \end{pmatrix}$$

Usando el formato *CDS*, se puede almacenar la matriz  $\underline{A}$  en otra matriz de  $(-p : q) \times n = 3 \times 6$  usando el mapeo  $val(i, j) = a_{i,i+j}$

Así, los renglones de  $val(:, :)$  de la matriz son:

$$\begin{aligned} val(:, -1) &= (0 \ 3 \ 7 \ 8 \ 9 \ 2) \\ val(:, 0) &= (10 \ 9 \ 8 \ 7 \ 9 \ -1) \\ val(:, +1) &= (-3 \ 6 \ 7 \ 5 \ 13 \ 0) \end{aligned}$$

Nótese que los dos ceros no existen en la matriz original.

### 3.2. Métodos de Resolución

#### Métodos Directos

En esta sección se mostrarán los algoritmos de resolución directa de sistemas lineales más comunes. El problema a resolver es

$$Ax = b$$

donde  $A \in M_n(\mathbb{R})$  y  $x, b \in \mathbb{R}^n$ . Por los resultados de álgebra lineal (véase [3]) este sistema de ecuaciones tiene una única solución si  $\det A \neq 0$ .

Como se mencionó, en estos métodos la solución  $\underline{x}$  se obtiene en un número fijo de pasos y que sólo están sujetos a los errores de redondeo. Entre los métodos de éste tipo se encuentran:

- **Descomposición LU** para matrices tanto simétricas como no-simétricas.
- **Descomposición de Cholesky** para matrices simétricas.
- **Descomposición LU** para matrices tridiagonales.

En ambos casos la matriz  $A$  es modificada y en la factorización  $LU$  el tamaño de la banda crece a  $2q + 1$  si  $q$  es el tamaño de la banda de  $A$ .

#### 3.2.1. Descomposición LU

Si se puede efectuar la eliminación gaussiana en el sistema  $\underline{A}\underline{x} = \underline{b}$  sin intercambios de renglones, entonces se puede factorizar la matriz  $\underline{A}$  en el producto de una matriz triangular inferior  $\underline{L}$  y una matriz superior  $\underline{U}$

$$\underline{A} = \underline{L}\underline{U}$$

donde

$$\underline{U} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & a_{n-1,n}^{(n-1)} \\ 0 & \cdots & \cdots & 0 & a_{nn}^{(n)} \end{pmatrix}$$

$$\underline{L} = \begin{pmatrix} 1 & \cdots & \cdots & \cdots & 0 \\ m_{21} & 1 & & & \vdots \\ \vdots & & 1 & & \vdots \\ \vdots & & & 1 & 0 \\ m_{m1} & \cdots & \cdots & m_{n,n-1} & 1 \end{pmatrix}$$

$$m_{ji} = \frac{a_{ji}^{(i)}}{a_{ii}^{(i)}}$$

Entonces el problema  $\underline{A}\underline{x} = \underline{b}$  lo podemos escribir como  $\underline{LU}\underline{x} = \underline{b}$  y se reduce a la solución sucesiva de los sistemas triangulares

$$\begin{aligned}\underline{L}\underline{y} &= \underline{b} \\ \underline{U}\underline{x} &= \underline{y}\end{aligned}$$

Una vez obtenida la factorización, obtenemos la solución haciendo

$$y_1 = \frac{b_1}{l_{11}}$$

y para toda  $i = 2, 3, \dots, n$

$$y_i = \frac{1}{l_{ii}} \left[ b_i - \sum_{j=1}^{i-1} l_{ij} y_j \right]$$

una vez que obtenemos  $\underline{y}$  se resuelve el sistema  $\underline{U}\underline{x} = \underline{y}$  por medio de la sustitución hacia atrás usando

$$\begin{aligned}x_n &= \frac{y_n}{u_{nn}} \\ x_i &= \frac{1}{u_{ii}} \left[ y_i - \sum_{j=i+1}^n u_{ij} x_j \right]\end{aligned}$$

La descomposición  $LU$  requiere  $\frac{n^3}{3}$  operaciones para una matriz densa, pero solo  $nq^2$  de operaciones para una matriz con ancho de banda  $q$ , lo cual lo hace muy económico computacionalmente.

### 3.2.2. Descomposición Cholesky

Cuando la matriz  $\underline{A}$  es simétrica y definida positiva ( $x^t A x > 0$ ), podemos obtener la descomposición  $\underline{LU} = \underline{A}$ . Así  $\underline{A} = \underline{LDU}$  y  $\underline{A}^t = \underline{LDL}^T$  y como  $\underline{A} = \underline{A}^t$  entonces  $\underline{U} = \underline{L}^t$  y donde  $\underline{D} = \text{diag}(\underline{U})$  es la diagonal con entradas positivas. Entonces el problema original se escribe como  $\underline{LL}^t x = b$ , y la solución se reduce a resolver los sistemas:

$$\begin{aligned}Ly &= b \\ L^t x &= y\end{aligned}$$

Cuando se puede aplicar ésta descomposición, el costo de cómputo es mas reducido que en las anteriores, ya que requiere  $\frac{n^3}{6}$  operaciones.

### 3.2.3. Factorización LU de Sistemas Tridiagonales

El siguiente algoritmo (factorización de Crout) factoriza la matriz  $A$  de  $n \times n$  cuya matriz es tridiagonal. Consecuencia de ello, ofrece una importante ventaja computacional a la hora de resolver el sistema pues sólo requiere  $(5n - 4)$  multiplicaciones/divisiones y  $(3n - 3)$  sumas/restas, ya que toma ventaja la tridiagonalidad de la matriz.

Se toma

$$\begin{aligned} l_{11} &= A_{11} \\ u_{12} &= A_{12}/l_{11} \end{aligned}$$

para  $i = 2, \dots, n-1$

$$\begin{aligned} l_{i,i-1} &= A_{i,i-1} \\ l_{ii} &= A_{ii} - l_{i,i-1}u_{i-1,i} \\ u_{i,i+1} &= A_{i,i+1}/l_{ii} \end{aligned}$$

por último

$$\begin{aligned} l_{n,n-1} &= A_{n,n-1} \\ l_{nn} &= A_{nn} - l_{n,n-1}u_{n-1,n} \end{aligned}$$

## Métodos Iterativos

En estos métodos se realizan iteraciones para aproximarse a la solución  $\underline{x}$  aprovechando las características propias de la matriz  $\underline{A}$  tratando de usar el menor número de pasos que en un método directo.

Un método iterativo para resolver el sistema algebraico

$$\underline{Ax} = \underline{b}$$

comienza con una aproximación inicial  $\underline{x}^0$  a la solución  $\underline{x}$  y genera una sucesión de vectores  $\{\underline{x}^n\}_{n=1}^{\infty}$  que converge a  $\underline{x}$ . Los métodos iterativos convierten al sistema  $\underline{Ax} = \underline{b}$  en otro equivalente  $\underline{u} = \underline{T}\underline{u} + \underline{c}$  para alguna matriz fija  $\underline{T}$  y un vector  $\underline{c}$ . Después de seleccionar el vector  $\underline{x}^0$  la sucesión de los vectores de la solución aproximada se genera calculando

$$\underline{x}^n = \underline{T}\underline{x}^{n-1} + \underline{c} \quad \forall n = 1, 2, 3, \dots$$

La convergencia a la solución la garantiza el siguiente teorema

**Teorema 1.** Si  $\|\underline{T}\| < 1$ , entonces el sistema lineal  $\underline{x} = \underline{T}\underline{x} + \underline{c}$  tiene una solución única  $\underline{x}^\alpha$  y las iteraciones  $\underline{x}^n$  definidas para  $\underline{x}^n = \underline{T}\underline{x}^{n-1} + \underline{c}$  convergen hacia la solución exacta  $\underline{x}^\alpha$  para cualquier aproximación lineal  $\underline{x}^0$ .

Entre los métodos más usados para el tipo de problemas tratados en éste trabajo están: *Conjugate Gradient Method (CGM)* para las matrices simétricas y *Generalized Minimal Residual (GMRES)* para matrices no simétricas. Estos dos últimos métodos están en el espacio de *Krylov*. Los métodos minimizan en la  $k$ -ésima iteración alguna medida de error sobre el espacio afín  $x_0 + K_k$  donde  $x_0$  es la iteración inicial y  $K_k$  es el  $k$ -ésimo espacio de Krylov

$$K_k = \left\langle \left\langle r_0, Ar_0, \dots, A^{k-1}r_0 \right\rangle \right\rangle$$

### 3.2.4. Método del Gradiente Conjugado

Existen un conjunto de métodos comúnmente utilizados en la resolución de grandes sistemas lineales, entre los cuales el método del gradiente conjugado es de los más populares para resolver el sistema de ecuaciones  $Ax = b$ , donde  $x$  es el vector incógnita,  $b$  es un vector conocido y  $A$  es una matriz cuadrada, simétrica ( $A = A^t$ ) y positiva-definida ( $x^t Ax > 0$ ).

**Proposición.** Si  $A$  es simétrica y positiva definida, los dos problemas siguientes son equivalentes

$$f(x) = \frac{1}{2}x^t Ax - x^t b + c \rightarrow \text{mín} \quad (3.2.1)$$

$$Ax = b \quad (3.2.2)$$

*Demostración.* Dado  $f(x) = \frac{1}{2}x^tAx - x^tb + c$  tenemos que el gradiente de  $f(x)$  es

$$\nabla f(x) = \frac{1}{2}x^tA + \frac{1}{2}Ax - b$$

al ser  $A$  simétrica

$$\nabla f(x) = Ax - b$$

□

El gradiente de  $f$ , utilizando la demostración de la proposición anterior está dado por

$$\nabla f(x) = Ax - b$$

ahora si  $x$  es solución, entonces es un punto crítico de  $f$  (mínimo o máximo) y al ser  $A$  positiva-definida, se garantiza que al hacer  $\nabla f(x) = 0$  el vector  $x$  sea un mínimo. Por tanto

$$0 = Ax - b \implies Ax = b$$

Sea  $x_0$  un punto de partida, se tiene

$$\begin{aligned} f(x) &= f(x_0) + \langle \nabla f(x_0), x - x_0 \rangle + \frac{1}{2}(x - x_0)^tA(x - x_0) \\ &= f(x_0) + \|x - x_0\| \|\nabla f(x_0)\| \cos\theta + \frac{1}{2}(x - x_0)^tA(x - x_0) \end{aligned}$$

donde  $\theta$  es el ángulo entre  $\nabla f(x_0)$  y  $(x - x_0)$ . Como se busca  $f(x)$  sea mínimo,  $\nabla f(x_0)$  y  $(x - x_0)$  deben de tener la misma dirección y el mismo sentido.

Supongamos que se ha encontrado  $x$ , y lo denotamos como  $x_1$ , se puede plantear para  $k \geq 0$ , obteniendo así el método del gradiente

$$x_{k+1} = x_k - t_k v_k$$

es decir, se busca una sucesión de vectores  $x_k$  tal que  $f(x_{k+1}) < f(x_k)$ .

En el algoritmo del gradiente conjugado se toma a la matriz  $A$  como positiva definida, y  $Ax = b$ . También como dato de entrada, el vector de búsqueda inicial  $x^0$  y se calcula  $r^0 = b - Ax^0$ ,  $p^0 = r^0$ , quedando el método numérico esquemáticamente como

$$\begin{aligned} \alpha^n &= \frac{\langle p^n, p^n \rangle}{\langle p^n, Ap^n \rangle} \\ x^{n+1} &= x^n + \alpha^n p^n \\ r^{n+1} &= r^n - \alpha^n Ap^n \\ \text{prueba de convergencia} \\ \beta^n &= \frac{\langle r^{n+1}, r^{n+1} \rangle}{\langle r^n, r^n \rangle} \\ p^{n+1} &= r^{n+1} + \beta^n p^n \\ n &\leftarrow n + 1 \end{aligned}$$

la solución aproximada será  $x^{n+1}$ .

### 3.2.5. GMRES

Si la matriz generada por la discretización del método de diferencias finitas no es simétrica, entonces una opción para resolver el sistema  $Ax = b$  es el método *GMRES* (Generalized Minimal Residual).

La idea básica del método consiste en construir una base ortonormal  $\{v^1, v^2, \dots, v^n\}$  para el espacio de *Krylov*  $K_n(A, v^n)$ . Para hacer  $v^{n+1}$  ortogonal a  $K_n(A, v^n)$  se usa un algoritmo modificado del método de Gram-Schmidt para la generalización de la base ortonormal. Sea  $V$  la matriz donde la  $j$ -ésima columna corresponde al vector  $v^j$  para  $j = 1, 2, \dots, n$  y sea  $H = (h_{ij})$  con  $1 \leq i, j \leq n$  donde las entradas no especificadas en el algoritmo son cero. Entonces la matriz  $H$  es una matriz superior de *Hessenberg*, es decir,  $h_{ij} = 0$  para  $j < i - 1$ .

El algoritmo *GMRES* recibe un vector inicial  $x^0$  y calcula  $r^0 = b - Ax^0$ , después se toma  $\beta^0 = \|r^0\|$ ,  $v^1 = \frac{r^0}{\beta^0}$ , quedando el método esquemáticamente

$$\begin{aligned} & \text{para } n = 1, 2, \dots \\ & \text{mientras } \beta^n < \tau\beta^0 \{ \\ & \quad w_0^{n+1} = Av^n \\ & \quad \text{para } l = 1, \dots, n \{ \\ & \quad \quad h_{l,n} = \langle w_l^{n+1}, v^l \rangle \\ & \quad \quad w_{l+1}^{n+1} = w_l^{n+1} - h_{l,n}v^l \\ & \quad \quad \} \\ & \quad h_{n+1,n} = \|w_{n+1}^{n+1}\| \\ & \quad v^{n+1} = \frac{w_{n+1}^{n+1}}{h_{n+1,n}} \\ & \text{Calcular } y^n \text{ tal que } \beta^n = \|\beta^0 e_1 - \hat{H}_n y^n\| \\ & \quad \} \end{aligned}$$

donde  $\hat{H}_n = (h_{ij})_{1 \leq i \leq n+1, 1 \leq j \leq n}$ , la solución aproximada será  $x^n = x^0 + V_n y^n$ .

Existen otros métodos iterativos para la solución del sistema  $Ax = b$ , como: Gauss-Seidel y Jacobi, Véase[7]. Sin embargo no alcanzan la eficiencia del Gradiente Conjugado o de GMRES. Por ejemplo, para el sistema

$$\begin{pmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 24 \\ 30 \\ -24 \end{pmatrix}$$

cuya solución es el vector  $x = \begin{pmatrix} 3 \\ 4 \\ -5 \end{pmatrix}$ ; el método de Gauss-Seidel requirió de 50 iteraciones, Jacobi de

108 iteraciones, mientras que, el método del Gradiente Conjugado sólo requirió 3 iteraciones. Dadas estas métricas, se optó por incluir en este texto los métodos iterativos más eficientes para matrices simétricas y no-simétricas, que son, Gradiente Conjugado y GMRES respectivamente.

## Capítulo 4

# Implementación Computacional

Para la implementación computacional del método de Diferencias Finitas se parte de la discretización desarrollada en el capítulo 2 la cual genera al menos un sistema algebraico, el cual puede ser resuelto por distintos métodos numéricos como los vistos en el capítulo 3, tomando en cuenta el almacenamiento de las matrices en la RAM y la minimización de las operaciones involucradas en la resolución del sistema algebraico.

Así, en este capítulo, se ven distintas formas de implementación computacional en paquetes interpretados como Scilab y MatLab(Octave), además del lenguaje de programación compilable C++ en conjunción de la librería GMM++; para que el usuario interesado en resolver problemas de ecuaciones diferenciales parciales pueda optar por aquel que resuelva sus necesidades pero siempre tomando en cuenta la facilidad de implementación versus el rendimiento computacional.

El software presenta características particulares que hacen que su complejidad alcance niveles importantes. A través de los años, los investigadores, ingenieros y desarrolladoras han intentado minimizar esa dificultad incorporando distintas ideas de otros ámbitos al software. Entre los paradigmas de desarrollo actuales más destacados se encuentra el **orientado a objetos**.

Detrás de este paradigma se encuentran visiones de cómo se puede disminuir la complejidad y obtener software de calidad. Incorpora una serie de técnicas particulares y se tiene una visión de como se puede enfocar el desarrollo. El paradigma orientado a objetos plantea que un sistema puede ser visto como objetos que tienen ciertas características y que colaboran entre ellos para poder realizar una tarea.

### 4.1. Scilab

**Scilab** (<http://www.scilab.org/>) es un paquete de software libre para cómputo científico, orientado al cálculo numérico, a las operaciones matriciales y especialmente a las aplicaciones científicas y técnicas.

Puede ser utilizado como una simple calculadora matricial, pero su interés principal radica en las cientos de funciones de propósito general como especializadas que posee así como en sus posibilidades para la visualización.

Scilab posee además un lenguaje de programación propio, muy parecido al lenguaje utilizado por Matlab, el cual permite escribir programas para resolver problemas en concreto o nuevas funciones.

Inicialmente desarrollado por INRIA (Institut National de Recherche en Informatique et Automatique), actualmente está a cargo de un consorcio de universidades, empresas y centro de investigación.

Declarar vectores y matrices es tan sencillo como escribir

```
b = zeros(5,1); // vector de tamaño 5
A = zeros(5,5) // matriz de tamaño 5x5
b(1) = 2; // asignación en la posición 1
b(3) = 41;
A(1,1) = 32; // asignación en la posición (1,1)
A(3,4) = 5;
SP = sparse(A); // SP es la matriz donde se guardan los valores no cero de A
```

Multiplicar matrices por vectores

```
A_ = A*b; // multiplicación matriz-vector
```

Solución de sistemas de ecuaciones  $Ax = b$

```
// Ax = b
x = inv(A)*b; // usando directamente la inversa
[L,U] = lu(A) // factorización LU
C = chol(A) // factorización cholesky
x = pcg(A,b) // gradiente conjugado
```

graficar

```
plot2d(x,y) // gráfica 2D
plot3d(x,y,z) // superficie
```

como se puede notar, Scilab ofrece ventajas en el tiempo de desarrollo de programas o pruebas numéricas, sin embargo, una desventaja es que es “más lento” en comparación con lenguajes como c++. Hay que medir qué nos importa más, las horas trabajando en un lenguaje de programación más eficiente o el rápido desarrollo de la implementación pero más tardado en la ejecución del programa.

## 4.2. MatLab (Octave)

MatLab (<http://www.mathworks.com/products/matlab/>) es un entorno para cómputo científico no libre, sumamente usado en el mundo. Ofrece un lenguaje de programación de alto nivel. Contiene enorme cantidad de funciones ya predefinidas que el desarrollador puede utilizar al instante. Una de las ventajas de MatLab es su enorme capacidad de ser ampliado y adecuarlo a las necesidades del usuario mediante los llamados *toolboxes*, que entre otras cosas ofrecen la capacidad de realizar: procesamiento de imágenes, análisis de datos, optimización, etc.

```
b = zeros(5,1); // vector de tamaño 5
A = zeros(5,5) // matriz de tamaño 5x5
b(1) = 2; // asignación en la posición 1
b(3) = 41;
A(1,1) = 32; // asignación en la posición (1,1)
A(3,4) = 5;
SP = sparse(A); // SP es la matriz donde se guardan los valores no cero de A
```

Multiplicar matrices por vectores

```
A_ = A*b; // multiplicación matriz-vector
```

Solución de sistemas de ecuaciones  $Ax = b$

```
// Ax = b
x = inv(A)*b; // usando directamente la inversa
[L,U] = lu(A) // factorización LU
C = chol(A) // factorización cholesky
x = pcg(A,b) // gradiente conjugado
```

graficar

```
plot(x,y) // gráfica 2D
mesh(x,y,z) // malla
surface(x,y,z) // superficie
```

Como se puede ver, al igual que Scilab, MatLab ofrece un desarrollo rápido y sin complicaciones.

Por otra parte, Octave(<http://www.gnu.org/software/octave/>) se considera el clon libre de MatLab y se desarrolla para ser una opción viable para sustituir MatLab en el ambiente científico. Octave ofrece la misma sintaxis de MatLab y la compatibilidad de los programas desarrollados en el.

### 4.3. C++

C++ es uno de los lenguajes más utilizados cuando se necesita eficiencia en el uso de los recursos de la(s) computadora(s). C++ se puede ver como la extensión del lenguaje C. C++ añade mecanismos modernos de la programación orientada a objetos, aunque, sin perder la forma estructural, por lo que se le considera a C++ un lenguaje híbrido.

#### Programación Orientada a Objetos

El modelo de objetos no es sólo una forma de programar sino que brinda los cimientos para poder desarrollar todo tipo de soluciones bajo sus principios. Muchas de las premisas de este paradigma han sido obtenidas de las ingenierías. La principal razón que hace atractivo a este modelo es la capacidad que tiene para combatir la complejidad inherente y disminuir los riesgos en el desarrollo. Otros paradigmas también tienen sus éxitos en estos puntos pero el modelo de objetos no se resiente tanto en proyectos de gran alcance. La escalabilidad no lo afecta de forma profunda como a otros paradigmas.

El modelo orientado a objetos presenta algunos elementos que son denominados **fundamentales**, véase [22]

- Jerarquía
- Abstracción
- Modularidad
- Encapsulamiento

Si cualquiera de estos elementos no está presente, no podemos considerar al paradigma como modelo de objetos.

#### Jerarquía

La jerarquía no es más que la posibilidad de realizar un ordenamiento en niveles de lo que se desea representar. De manera práctica, esa jerarquía se ve representada por la **herencia**, que puede ser de distintos tipos. Entre las más frecuentes se encuentran: la simple, la múltiple y la restrictiva. Todos los lenguajes orientados a objetos brindan la posibilidad de heredar el comportamiento.

#### Abstracción

Este término es quizá el más simple de entender, pero el que más problemas presenta a la hora de ser puesto en práctica. La abstracción es un proceso intelectual humano por el cual se es capaz de concentrarse particularmente en las características que interesan para la solución de una situación. Al atacar un problema, se intenta resolver mediante la aplicación de pasos que se han utilizado anteriormente y se sabe que funcionan. Al presentarse un caso nuevo, o que aparenta serlo, lo que se hace es obtener de él todas las similitudes con casos anteriores y a su vez tratar de ver cuáles son las particularidades que pueden ser de interés. Este proceso es específico de cada problema y de cada diseño, y en algunas ocasiones las características menos pensadas son las relevantes para la solución de la dificultad.

#### Modularidad

Este concepto no es propio de la orientación a objetos, sino que es uno de los que más se ha desarrollado en la ingeniería de software. El objetivo final es la división de un problema más complejo en unidades más pequeñas casi siempre sencillas. A su vez, la separación en módulos crea fronteras artificiales que permiten que los grupos desarrollen soluciones por separado integrándolas con mayor facilidad. Los módulos interactúan entre ellos y pueden ser transportados a otros proyectos en caso de necesidad. La reutilización es uno de los pilares del modelo orientado a objetos.

## Encapsulamiento

El encapsulamiento es el ocultamiento de la información de forma tal que sólo esté disponible para interactuar con un objeto sin la necesidad de conocer cómo se comporta internamente. Este factor hace que los objetos sean fácilmente reutilizables. Además, junto con los conceptos anteriores, crea la estructura precisa para una reducción de la complejidad. Muchas veces el encapsulamiento se le relaciona con el término *caja negra*. Un ejemplo sencillo ocurre cuando se es capaz de manejar algún tipo de artefacto sin saber exactamente qué es lo que ocurre internamente. En tal caso, el ocultamiento de información facilita la interacción.

## El objeto como base

Lógicamente, la orientación a objetos presenta muchas características comunes a otros tipos de prácticas. Lo que la distingue es el énfasis en definir y caracterizar de forma clara los componentes del sistema, dotándolo de sus capacidades. En el **objeto** se unen los datos y los algoritmos y es así en como el objeto se transforma en la pieza fundamental de esta estructura.

Un objeto debe de poseer cualidades que lo definan en esencia, es decir, tendrá propiedades invariantes que lo caracterizan a él y a su forma de actuar. Por ejemplo, no se puede imaginar un objeto *automóvil* sin su capacidad para desplazarse o un objeto *perro* sin su facultad de ladrar. Estas características forman parte intrínseca del objeto y deben ser representadas en el modelo.

## Programación Orientada a Objetos

Para concluir, si se quiere definir qué es la programación orientada a objetos se puede tomar la definición de un famoso ingeniero de software llamado **Grady Booch**: “*La programación orientada a objetos es un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de alguna clase, y cuyas clases son, todas ellas, miembros de una jerarquía de clases unidas mediante relaciones de herencia*”, véase [22]

### 4.4. Biblioteca gmm++

La biblioteca *gmm++* (<http://download.gna.org/getfem/html/homepage/gmm/>) provee algunos tipos básicos de matrices (densas y dispersas) y vectores, así como, operaciones genéricas (suma, resta, multiplicación, etc.) para operar con estas. Además contiene resolvedores para sistemas de ecuaciones tales como el gradiente conjugado, gmres, LU, QR, etc.

La instalación en sistemas basados en *debian* es

```
~# aptitude install libgmm++-dev
```

## Matrices y Vectores

Para declarar matrices y vectores en esta biblioteca basta escribir

```
Vector V(n)
Matrix M(n,m)
```

la convención es que cualquier tipo de vector o matriz puede ser inicializado con estos constructores. Acceder a los valores del vector o matriz

```
a = V[i]
V[i] = b
a = M(i, j)
M(i, j) = b
```

Gmm++ provee tipos de vectores y matrices densas

```
std::vector<T>
gmm::dense_matrix<T>
```

así como dispersas

```
std::rsvector<T>
gmm::csr_matrix<T>
gmm::csc_matrix<T>
```

Donde el tipo *csr* representa el tipo *compressed sparse row matrix* mientras que *csc* al tipo *compressed sparse column matrix*.

### Resolvedores Iterativos

Gmm++ contiene una buena cantidad de resolvedores iterativos que podemos usar facilmente para resolver los sistemas lineales que surgen de discretizar las ecuaciones diferenciales parciales. La lista de resolvedores es la siguiente:

```
// The matrix
gmm::row_matrix< std::vector<double> > A(10, 10);
// Right hand side
std::vector<double> B(10);
// Unknown
std::vector<double> X(10);
// Optional scalar product for cg
gmm::identity_matrix PS;
// Optional preconditioner
gmm::identity_matrix PR;
// Iteration object with the max residu
gmm::iteration iter(10E-9);
// restart parameter for GMRES
size_t restart = 50;
// Conjugate gradient
gmm::cg(A, X, B, PS, PR, iter);
// BICGSTAB BiConjugate Gradient Stabilized
gmm::bicgstab(A, X, B, PR, iter);
// GMRES generalized minimum residual
gmm::gmres(A, X, B, PR, restart, iter)
// Quasi-Minimal Residual method
gmm::qmr(A, X, B, PR, iter)
// unpreconditioned least square CG
gmm::least_squares_cg(A, X, B, iter)
```

## 4.5. Método de Descomposición de Dominio de Subestructuración

La descomposición de dominio generalmente se refiere a la separación de una EDP o una aproximación a ella dentro de problemas acoplados sobre subdominios pequeños formando una partición del dominio original. Esta descomposición puede hacerse a nivel continuo, donde diferentes modelos físicos, pueden ser usados en diferentes regiones, o a nivel discreto, donde puede ser conveniente el empleo de diferentes métodos de aproximación en diferentes regiones, o en la solución del sistema algebraico asociado a la aproximación de la EDP.

Los métodos de descomposición de dominio (Domain Decomposition Methods, DDM) se basan en la suposición de que dado un dominio  $\Omega \subset \mathbb{R}^n$ , se puede particionar en  $E$  subdominios  $\Omega_i$  con  $i = 1, 2, \dots, E$  tales que  $\Omega = \left( \bigcup_{i=1}^E \Omega_i \right)$ , entre los cuales puede existir o no un traslape. Entonces, el problema es reformulado en

términos de cada subdominio (mediante el uso de algún método de discretización, por ejemplo, diferencias finitas) obteniendo una familia de subproblemas de tamaño reducido e independientes entre sí, y que están acoplados a través de la solución en la interfase de los subdominios.

De esta manera, se puede clasificar de forma burda a los métodos de descomposición de dominio, como aquellos en que: existe traslape entre los subdominios y en los que no existe traslape. A la primera clase pertenece el método de Schwarz (en el cual el tamaño del traslape es importante en la convergencia del método) y a los de segunda clase pertenecen los métodos del tipo subestructuración (en el cual los subdominios sólo tienen en común a los nodos de interfase o frontera interior).

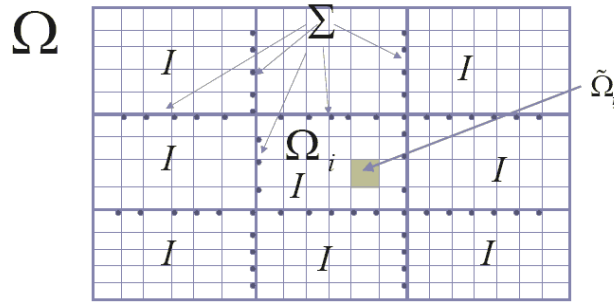


Figura 4.5.1: Dominios

Por otra parte, dado un sistema lineal  $Mx = w$  que proviene de la discretización de un método tipo diferencias finitas, elemento finito o volumen finito, siempre es posible reacomodarlo como

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$$

donde la matriz  $A$  es invertible, entonces

$$(D - CA^{-1}B)v = b - CA^{-1}a$$

y

$$u = A^{-1}(a - Bv)$$

De esta manera se ha transformado el sistema  $Mx = w$  en otro equivalente

$$Nv = g$$

pero con menor cantidad de grados de libertad, donde

$$\begin{aligned} N &= (D - CA^{-1}B) \\ g &= b - CA^{-1}a \end{aligned}$$

a esta descomposición matricial se le conoce como la *descomposición de Schur*. Por ello se han desarrollado varios métodos basados en esta descomposición, el más básico de ellos es el método de descomposición de dominio de subestructuración. A partir de este, se han generado múltiples variantes usando preconditionadores a priori, para acelerar la convergencia del sistema. Véase *apéndice B*.

Sin pérdida de generalidad, sea  $\Omega$  un dominio en  $\mathbb{R}^2$  el cual es descompuesto en una malla gruesa de  $n \times m$  dando un total de  $E$  subdominios; donde cada subdominio es descompuesto a su vez en  $p \times q$  elementos, entonces es necesario resolver los nodos  $u_\Gamma$  de la frontera interior  $\Gamma$  mediante la resolución del sistema virtual lineal asociado a método del complemento de Schur, véase *Apéndice B*.

$$Su_\Gamma = b$$

equivalente al sistema

$$\left[ \sum_{i=1}^E S_i \right] u_{\Gamma} = \sum_{i=1}^E b_i$$

está definido por

$$S_i = A_i^{\Gamma\Gamma} - A_i^{\Gamma I} (A_i^{II})^{-1} A_i^{I\Gamma}$$

y

$$b_i = b_{\Gamma_i} - A_i^{\Gamma I} (A_i^{II})^{-1} b_{I_i}$$

donde  $A_i^{\Gamma\Gamma}$ ,  $A_i^{I\Gamma}$ ,  $A_i^{II}$ ,  $A_i^{\Gamma I}$  son las matrices de carga asociadas a los nodos de FronteraInterior-FronteraInterior, FronteraInterior-Interiores, Interiores-Interiores, Interiores-FronteraInterior respectivamente de cada subdominio (las cuales son matrices bandadas), mientras que  $b_{\Gamma_i}$  y  $b_{I_i}$  son los vectores asociados al vector de carga de la frontera interior e interiores respectivamente.

Una vez resuelto el sistema  $Su_{\Gamma} = b$  de los nodos de la frontera interior, se procede a resolver los nodos interiores mediante la resolución de

$$u_I = (A_i^{II})^{-1} (b_I - A_i^{\Gamma I} u)$$

Notese que la dimensión del sistema  $Su_{\Gamma} = b$  es mucho menor que la del sistema original sin descomposición de dominio  $Ax = b$ . Además del cálculo de  $A_i^{\Gamma\Gamma} - A_i^{\Gamma I} (A_i^{II})^{-1} A_i^{I\Gamma}$  está desacoplado para cada subdominio  $i = 1, 2, \dots, E$ . Por ello es práctico implementar su solución en equipos con memoria compartida o dispersa, ya que sólo están acoplados por los valores en la frontera interior, que es un vector de dimensión pequeña que es fácil pasar entre los procesos que corren en un mismo procesador o múltiples cores.

# Capítulo 5

## Aplicaciones y Pruebas

### 5.1. Ejemplos

Para ejemplificar el método de diferencias finitas desarrollado, se mostrará cómo resolver algunas ecuaciones clásicas como Poisson, el Calor, la Onda, de Advección. Usando lo desarrollado en el capítulo dos, se pueden resolver ecuaciones diferenciales parciales de segundo orden en espacio y tiempo.

Dada la ecuación diferencial parcial, para poder utilizar el *Método de Diferencias Finitas* se debe de:

- Generar una malla del dominio, es decir, un conjunto de puntos en los cuales se buscará la solución aproximada a la ecuación diferencial parcial.
- Sustituir las derivadas correspondientes con alguna de las fórmulas de diferencias finitas centradas, para obtener un sistema algebraico de ecuaciones  $Ax = b$ .
- Resolver el sistema de ecuaciones para obtener la solución aproximada en cada punto de la malla.

#### 5.1.1. Ecuación de Poisson

La ecuación de Poisson

$$\Delta u = f$$

es el caso general de la ecuación de Laplace.

Donde

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f$$

en  $\Omega = [a, b] \times [c, d]$

con condiciones  $u(x, y) = g(x, y)$  en  $\partial\Omega$ .

Entonces, según el método de diferencias finitas debemos de sustituir las derivadas por su expresión en diferencia finita, por tanto

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} &= \frac{u_{i-1} - 2u_i + u_{i+1}}{h_1^2} \\ \frac{\partial^2 u}{\partial y^2} &= \frac{u_{i-n} - 2u_i + u_{i+n}}{h_2^2} \end{aligned}$$

por tanto la representación en diferencias finitas de la ecuación de laplace queda como

$$\frac{u_{i-1} - 2u_i + u_{i+1}}{h_1^2} + \frac{u_{i-n} - 2u_i + u_{i+n}}{h_2^2} = f_i$$

agrupando los términos se obtiene

$$\frac{1}{h_2^2}u_{i-n} + \frac{1}{h_1^2}u_{i-1} - 2\left(\frac{1}{h^2} + \frac{1}{h_2^2}\right)u_i + \frac{1}{h_1^2}u_{i+1} + \frac{1}{h_2^2}u_{i+n} = f_i$$

este último resultado es el estencil para el *i-ésimo* nodo de la malla que se utiliza para aproximar la solución de la ecuación de Poisson mediante el método de diferencias finitas, con un error de truncamiento  $O(h_1^2 + h_2^2)$ .

**Ejemplo**

Considérese la ecuación de Poisson en el dominio  $\Omega = [0, 0.5]$

$$\Delta u = -\pi^2 \cos(\pi x)$$

con la condición de frontera Dirichlet y Neumann

$$\begin{aligned} u(0) &= 1 \\ u(0.5) &= -\pi \end{aligned}$$

respectivamente.

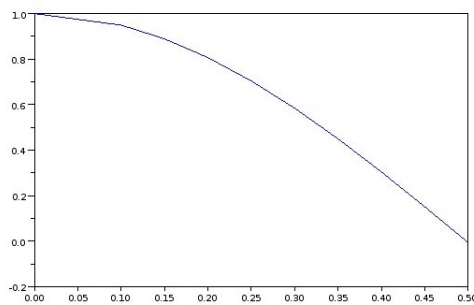


Figura 5.1.1: Solución de la ecuación de Poisson con condiciones Dirichlet y Neumann

El análisis de la partición de la malla contra el error en norma infinito

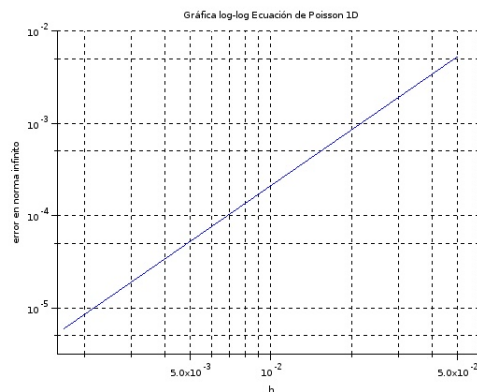


Figura 5.1.2: Gráfica del análisis del refinamiento usando diferencias finitas de la ecuación de Poisson, usando el gráfico de log error vs. log h, en donde se muestra la región de pendiente aproximada a -2 acorde a lo teóricamente esperado.

**Ejemplo**

Considérese la ecuación de Poisson para el dominio  $\Omega = [-1, 1] \times [-1, 1]$

$$\Delta u = 2n^2\pi^2 \text{sen}(n\pi x) \text{sen}(n\pi y)$$

con condiciones de frontera  $g_{\partial\Omega} = 0$ .

Cabe señalar que existe la solución analítica para este problema y es  $u(x, y) = \text{sen}(n\pi x) \text{sen}(n\pi y)$ . Se toma para el ejemplo el valor de  $n = 4$ . La gráfica de la solución se muestra a continuación.

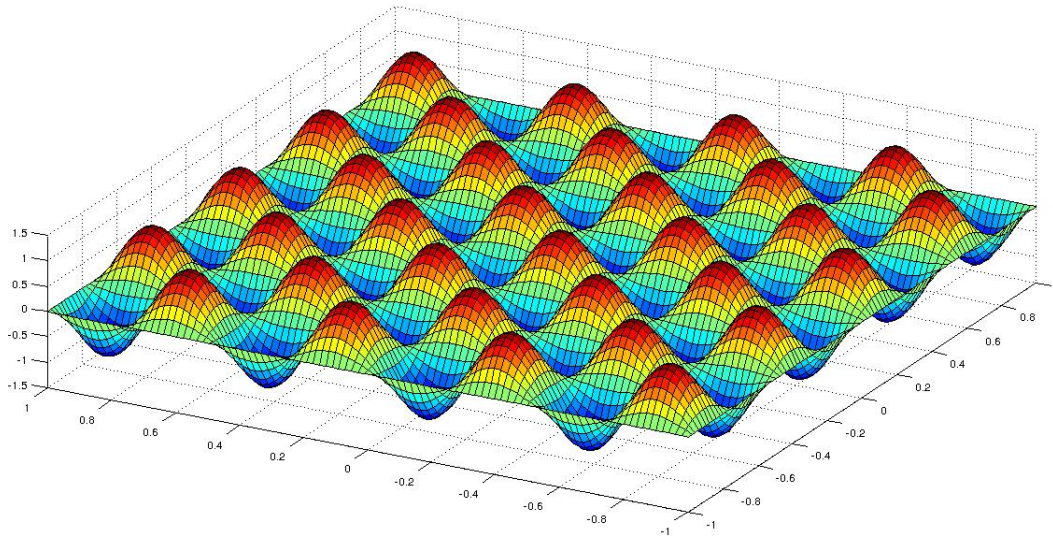


Figura 5.1.3: Solución de la ecuación de Poisson

El análisis de la partición de la malla contra el error en norma infinito

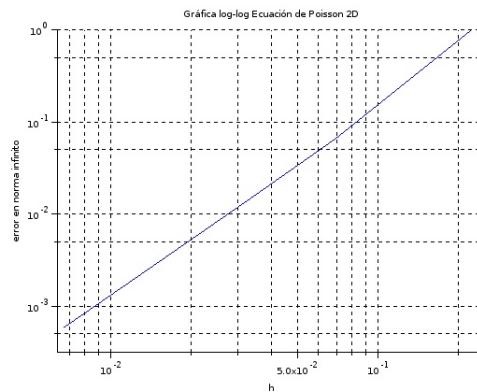


Figura 5.1.4: Gráfica del análisis del refinamiento usando diferencias finitas de la ecuación de Poisson, usando el gráfico de log error vs. log h, en donde se muestra la región de pendiente aproximada a -2 acorde a lo teóricamente esperado.

**5.1.2. Ejemplos de Advección-Difusión**

Los siguientes tres ejemplos no tienen solución analítica, sin embargo, los resultados son conformes a la literatura, por ejemplo, véase [14]

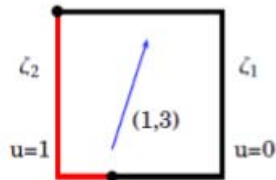
**Ejemplo**

Para el primer ejemplo, la ecuación utilizada es

$$-v\Delta u + \underline{b} \cdot \nabla u = 0$$

con  $\Omega = [0, 1] \times [0, 1]$ , con las condiciones de frontera

$$u(x, y) = \begin{cases} 0 & (x, y) \in \zeta_1 \\ 1 & (x, y) \in \zeta_2 \end{cases}$$



con  $\underline{b} = (1, 3)$  y  $v = 0.01$ . Se utilizó una discretización de  $512 \times 512$  nodos.

Para poder resolver esta ecuación (y las siguientes) en el programa desarrollado, se debe de hacer un poco de álgebra, ya que el programa recibe términos tipo  $au_{xx} + bu_{yy}$ , es decir, dada

$$\begin{aligned} -v\Delta u + \underline{b} \cdot \nabla u &= 0 \\ -0.01(u_{xx} + u_{yy}) + (1, 3) \cdot (u_x, u_y) &= 0 \\ -0.01u_{xx} - 0.01u_{yy} + u_x + 3u_y &= 0 \end{aligned}$$

de esta manera ya se puede introducir cada coeficiente de la derivada parcial dentro del programa que resuelve el problema.

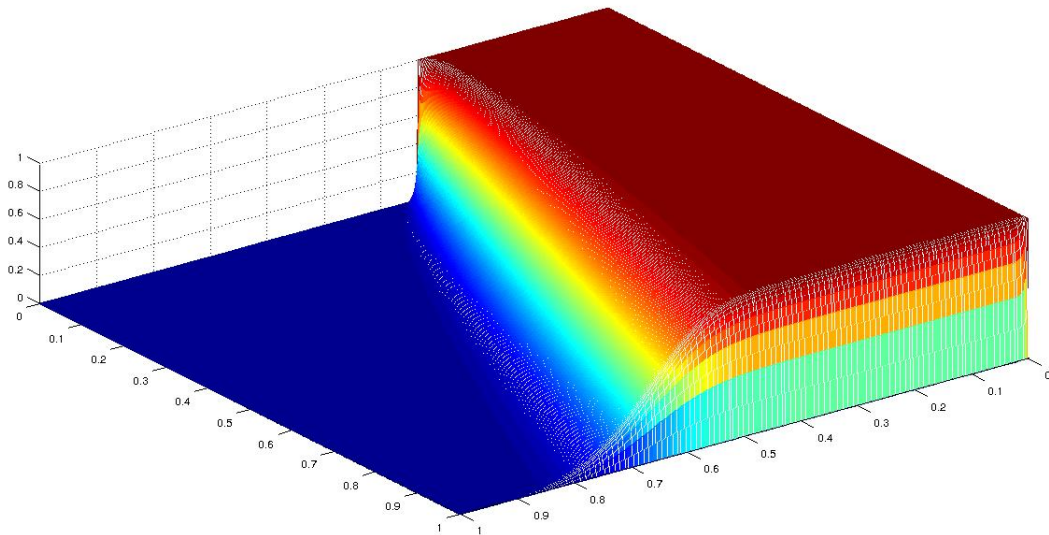


Figura 5.1.5: Solución del ejemplo

**Ejemplo**

Para este segundo ejemplo la ecuación a resolver es

$$-v\Delta u + \underline{b} \cdot \nabla u + cu = 0$$

con  $\Omega = [-1, 1] \times [-1, 1]$ , con las condiciones de frontera

$$u(x, y) = \begin{cases} y = -1 & 0 < x < 1 \\ y = 1 & 0 < x \leq 1 \\ x = 1 & -1 \leq y \leq 1 \end{cases}$$

$$u(x, y) = 0 \quad \text{para otro caso}$$

el coeficiente advectivo  $\underline{b} = (y, -x)$ , el valor de  $c = 10^{-4}$  y  $v = 0.01$ . Se utilizó una discretización de  $256 \times 256$  nodos.

Desarrollando la ecuación se tiene que

$$-v\Delta u + \underline{b} \cdot \nabla u + cu = 0$$

$$-0.01(u_{xx} + u_{yy}) + (y, -x) \cdot (u_x, u_y) + 10^{-4}u = 0$$

$$-0.01u_{xx} - 0.01u_{yy} + yu_x - xu_y + 10^{-4}u = 0$$

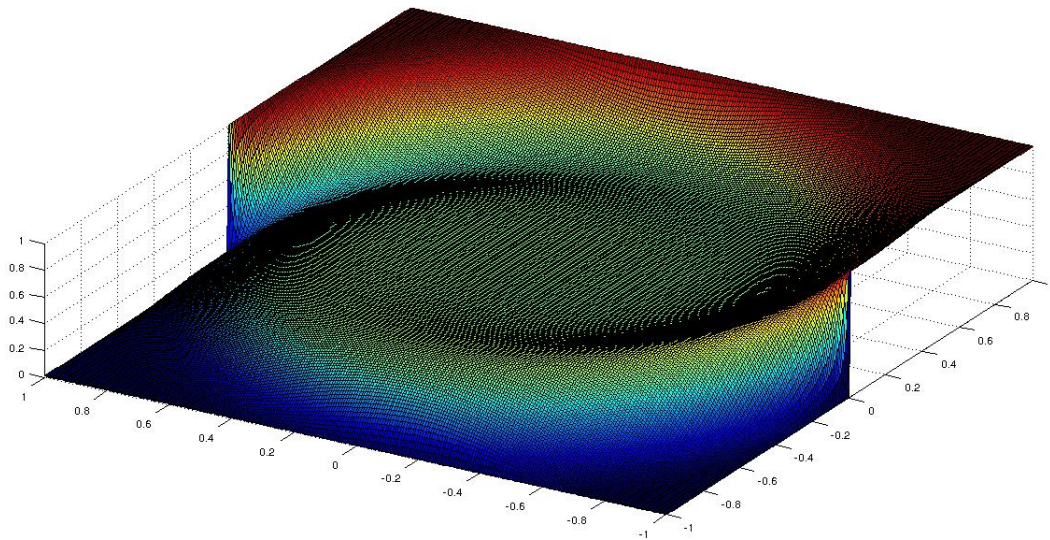


Figura 5.1.6: Solución del ejemplo

### Ejemplo

Para el tercer ejemplo la ecuación a resolver es

$$-v\Delta u + \underline{b} \cdot \nabla u + cu = 0$$

con  $\Omega = [-1, 1] \times [-1, 1]$ , con las condiciones de frontera

$$u(x, y) = \begin{cases} x = -1 & -1 < y < 1 \\ y = 1 & -1 \leq x \leq 1 \end{cases}$$

$$u(x, y) = 0 \quad \left\{ \begin{array}{l} y = -1 \\ -1 \leq x \leq 1 \end{array} \right.$$

$$u(x, y) = \frac{1+y}{2} \quad \left\{ \begin{array}{l} x = 1 \\ -1 \leq y \leq 1 \end{array} \right.$$

el coeficiente advectivo  $\underline{b} = \left(\frac{1+y}{2}, 0\right)$ , el valor de  $c = 10^{-4}$  y  $v = 0.01$ . Se utilizó una discretización de  $128 \times 128$  nodos.

Desarrollando la ecuación se tiene que

$$\begin{aligned}
 -v\Delta u + \underline{b} \cdot \nabla u + cu &= 0 \\
 -0.01(u_{xx} + u_{yy}) + \left(\frac{1+y}{2}, 0\right) \cdot (u_x, u_y) + 10^{-4}u &= 0 \\
 -0.01u_{xx} - 0.01u_{yy} + \left(\frac{1+y}{2}\right)u_x + 10^{-4}u &= 0
 \end{aligned}$$

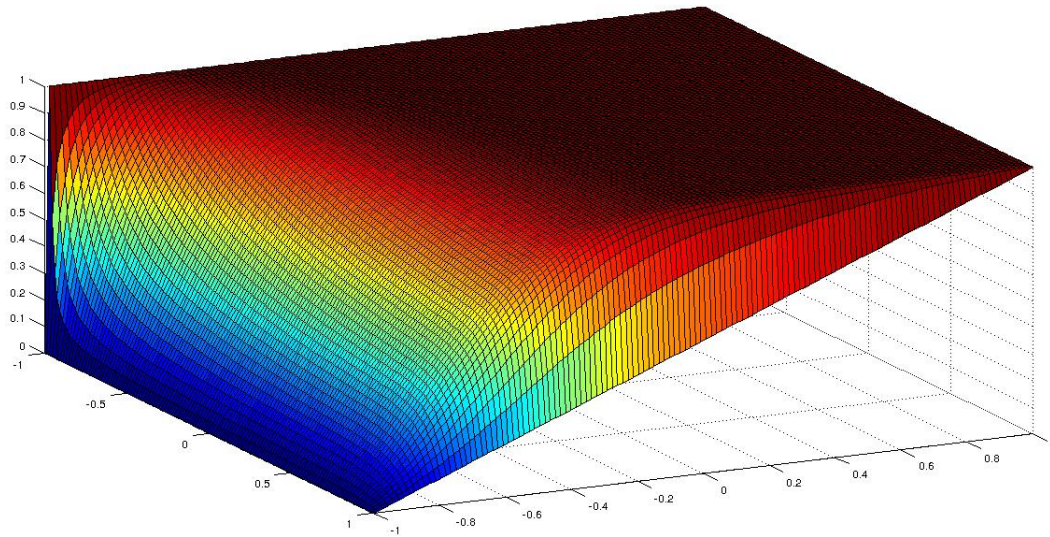


Figura 5.1.7: Solución del ejemplo

### 5.1.3. Ecuación de Calor o Difusión

La ecuación diferencial parcial parabólica que se resolverá es la de calor o difusión

$$u_t = \alpha u_{xx}$$

con  $0 < x < l$  y  $t > 0$  sujeta a las condiciones

$$\begin{aligned}
 u(0, t) = u(l, t) &= 0 \\
 u(x, 0) &= f(x)
 \end{aligned}$$

entonces, comenzando con la discretización del tiempo, se tiene que

$$u_t \approx \frac{u_i^{j+1} - u_i^j}{k}$$

donde  $k$  es la partición del tiempo  $t$  en intervalos. La parte espacial se obtiene como

$$\alpha u_{xx} \approx \frac{\alpha}{2} \left[ \frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h^2} + \frac{u_{i-1}^{j+1} - 2u_i^{j+1} + u_{i+1}^{j+1}}{h^2} \right]$$

entonces

$$\frac{u_i^{j+1} - u_i^j}{k} = \frac{\alpha}{2} \left[ \frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h^2} + \frac{u_{i-1}^{j+1} - 2u_i^{j+1} + u_{i+1}^{j+1}}{h^2} \right]$$

ahora, como ya se sabe, basta agrupar los términos  $j + 1$  del lado izquierdo de la igualdad y los términos  $j$  del lado derecho, para así obtener el sistema de ecuaciones que se resolverá  $Au^{j+1} = Bu^j$  en cada intervalo de tiempo, obteniendo el estencil siguiente

$$-\frac{\alpha}{2h^2}u_{i-1}^{j+1} + \left(\frac{1}{k} + \frac{\alpha}{h^2}\right)u_i^{j+1} - \frac{\alpha}{2h^2}u_{i+1}^{j+1} = \frac{\alpha}{2h^2}u_{i-1}^j + \left(\frac{1}{k} - \frac{\alpha}{h^2}\right)u_i^j + \frac{\alpha}{2h^2}u_{i+1}^j$$

### Ejemplo

Considérese la ecuación del calor en el dominio  $\Omega = [0, 1]$

$$u_t = \Delta u$$

con condición inicial

$$u(x, 0) = \text{sen}(\pi x)$$

y condiciones en la frontera

$$u(x, 0, t) = 0$$

$$u(x, 1, t) = 0$$

Se resolvió utilizando 21 nodos con  $dt = 0.025$  con 21 pasos de tiempo, es decir, el tiempo final fue de 0.5. La solución al problema se muestra en las siguientes gráficas para algunos tiempos

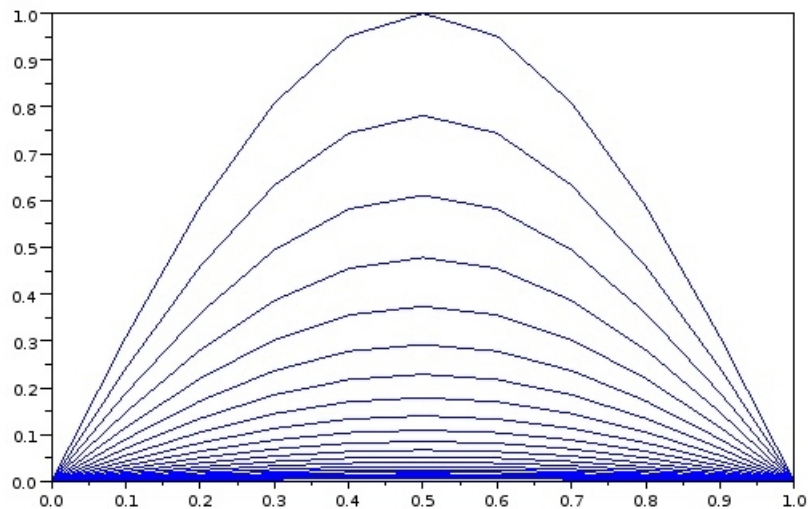


Figura 5.1.8: Solución de la ecuación de calor

El análisis de la ecuación de calor se da en la gráfica *loglog*

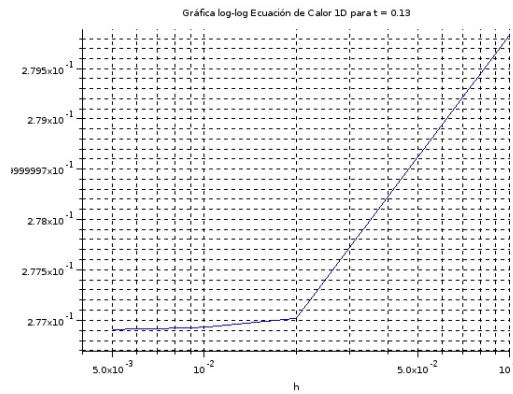


Figura 5.1.9: Gráfica del análisis del refinamiento usando diferencias finitas de la ecuación de calor, usando el gráfico de log error vs. log h, en donde se muestra la región de pendiente aproximada a -2 acorde a lo teóricamente esperado.

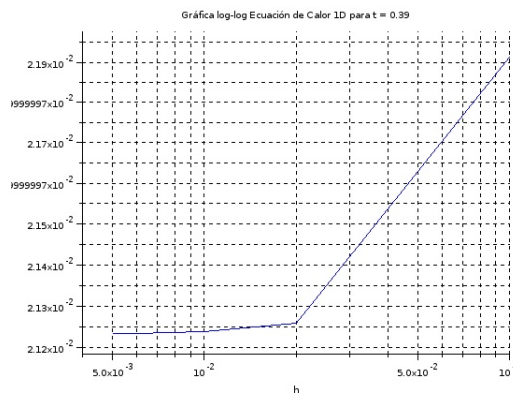


Figura 5.1.10: Gráfica del análisis del refinamiento usando diferencias finitas de la ecuación de calor, usando el gráfico de log error vs. log h, en donde se muestra la región de pendiente aproximada a -2 acorde a lo teóricamente esperado.

**Ejemplo**

Se resolverá la ecuación de calor en dos dimensiones en el dominio  $\Omega = [0, 1] \times [0, 1]$

$$u_t = \Delta u$$

con condición inicial

$$u(x, 0) = \text{sen}(\pi x) \text{sen}(2\pi y)$$

y condiciones en la frontera

$$\begin{aligned} u(x, 0, t) &= 0 \\ u(x, 1, t) &= 0 \\ u(0, y, t) &= 0 \\ u(1, y, t) &= 0 \end{aligned}$$

El problema se corrió con una malla de  $21 \times 21$  nodos, un paso de tiempo de  $dt = 0.0025$  durante 40 pasos de tiempo.

La solución al problema se muestra en las siguientes gráficas para algunos tiempos

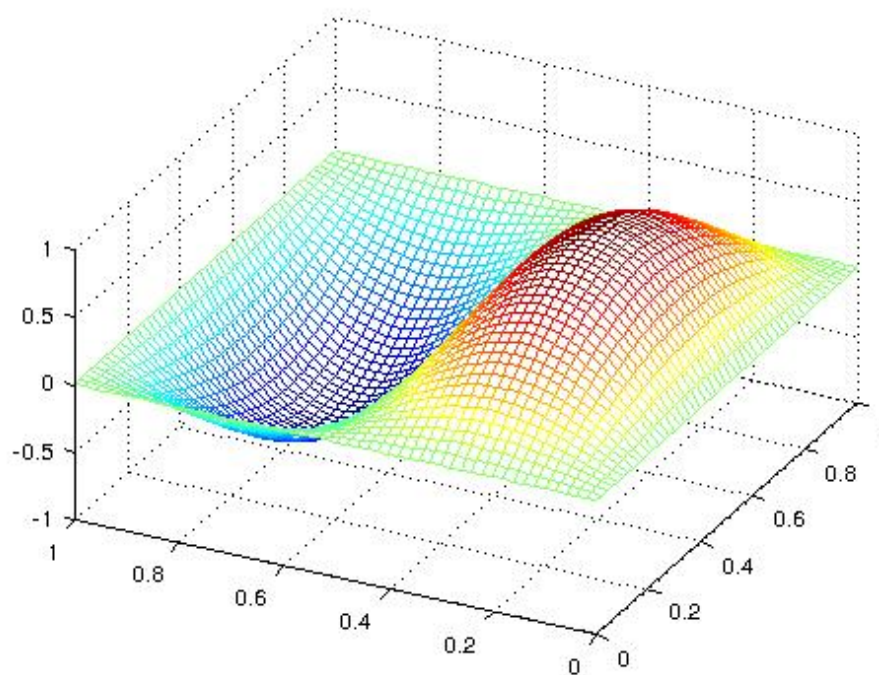


Figura 5.1.11: Solución de la ecuación de calor. Condición inicial.

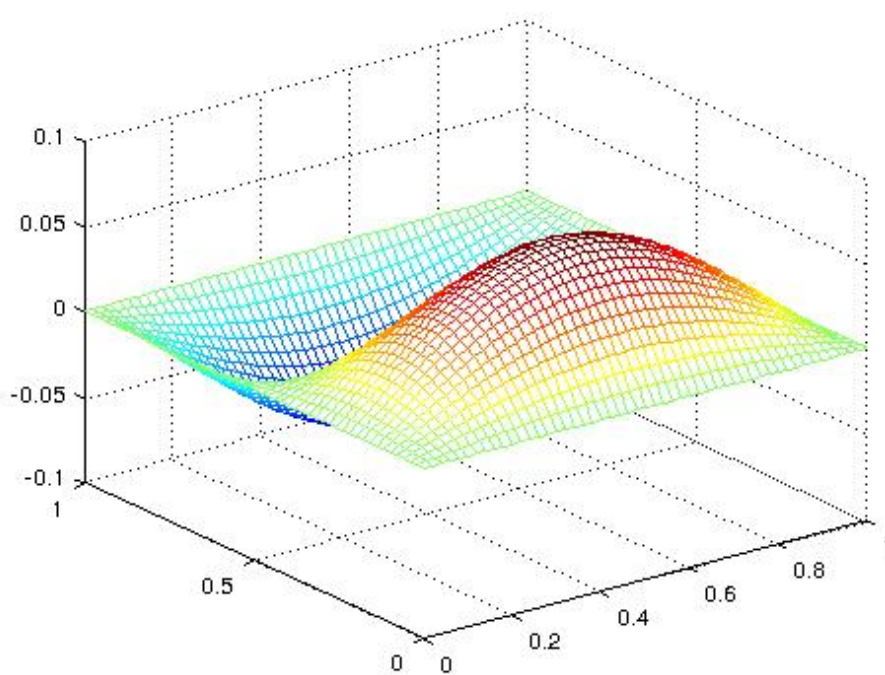


Figura 5.1.12: Solución de la ecuación de calor.

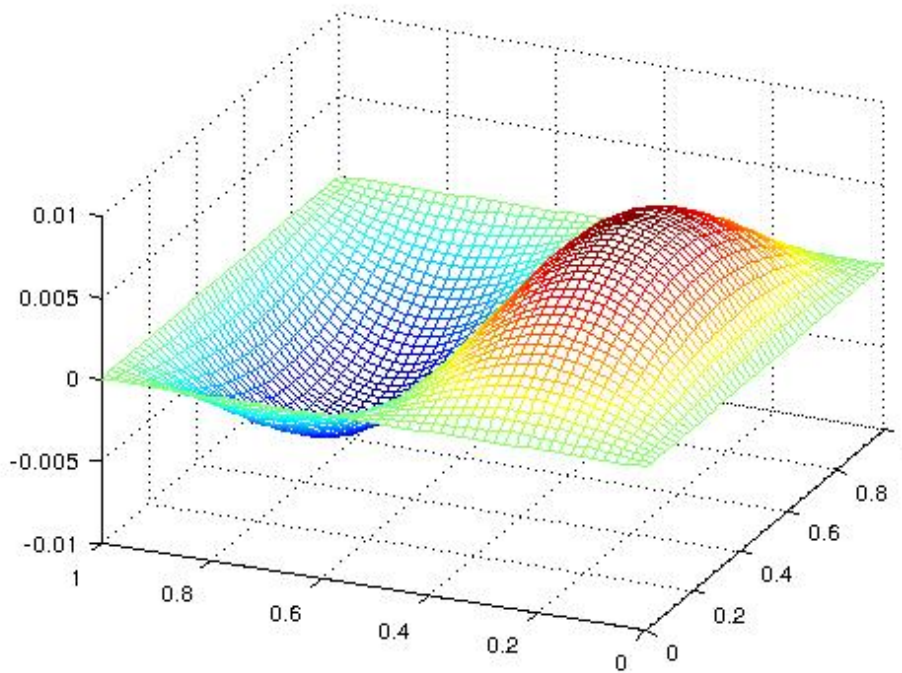


Figura 5.1.13: Solución de la ecuación de calor. Tiempo final.

Este ejemplo tiene solución mediante series de Fourier, por lo tanto no se realiza el análisis, ya que, Fourier también es un método aproximativo, sin embargo, la solución coincide con lo reportado en la literatura.

#### 5.1.4. Ecuación de Onda en 1D

La ecuación diferencial parcial que se resolverá es la ecuación de onda, que es el ejemplo de una ecuación hiperbólica. La ecuación de onda esta dada como

$$u_{tt} = \alpha u_{xx}$$

para  $0 < x < l$  y  $t > 0$ , sujeta a las condiciones

$$\begin{aligned} u(0, t) = u(l, t) &= 0 \\ u(x, 0) &= f(x) \\ u_x(x, 0) &= g(x) \end{aligned}$$

donde  $\alpha$  es una constante. Se comienza reemplazando las derivadas parciales por un cociente de diferencias finitas

$$\begin{aligned} u_{tt} &\approx \frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{k^2} \\ u_{xx} &\approx \frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h^2} \end{aligned}$$

por tanto, se tiene que el estencil para resolver esta ecuación es

$$u_i^{j+1} = 2u_i^j - u_i^{j-1} + \frac{\alpha k^2}{h^2} u_{i-1}^j - \frac{2\alpha k^2}{h^2} u_i^j + \frac{\alpha k^2}{h^2} u_{i+1}^j$$

**Ejemplo**

Consideremos la ecuación de onda en el dominio  $\Omega = [-1, 1]$

$$u_{tt} = 4u_{xx}$$

con condiciones iniciales

$$\begin{aligned} u(x, 0) &= \text{sen}(\pi x) \\ u_t(x, 0) &= 0 \end{aligned}$$

y condiciones en la frontera

$$\begin{aligned} u(0, t) &= 0 \\ u(1, t) &= 0 \end{aligned}$$

El problema se corrió en una malla de 1001 nodos y un paso de tiempo  $dt = 0.001$  durante 500 pasos de tiempo.

La solución al problema se muestra en la siguiente gráfica para algunos tiempos

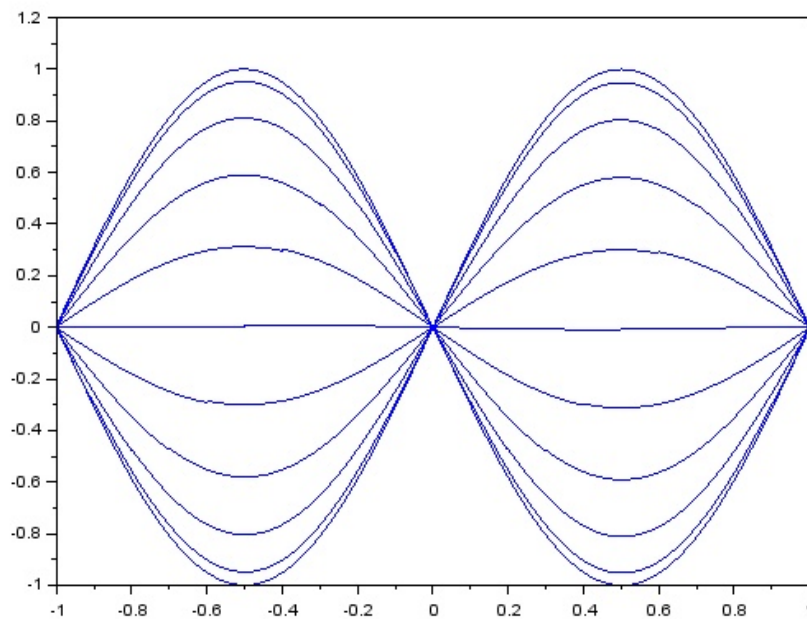


Figura 5.1.14: Solución de la ecuación de onda

El análisis de la ecuación de onda, mediante la gráfica *loglog* es el siguiente

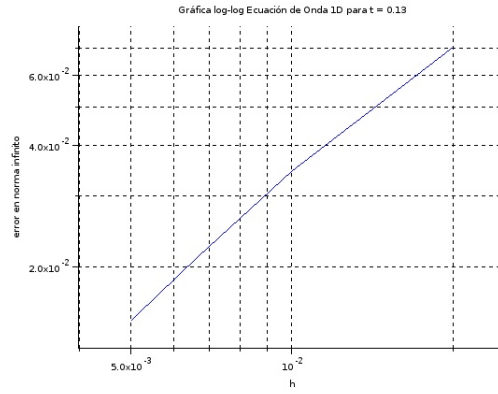


Figura 5.1.15: Gráfica del análisis del refinamiento usando diferencias finitas de la ecuación de onda, usando el gráfico de log error vs. log h, en donde se muestra la región de pendiente aproximada a -2 acorde a lo teóricamente esperado.

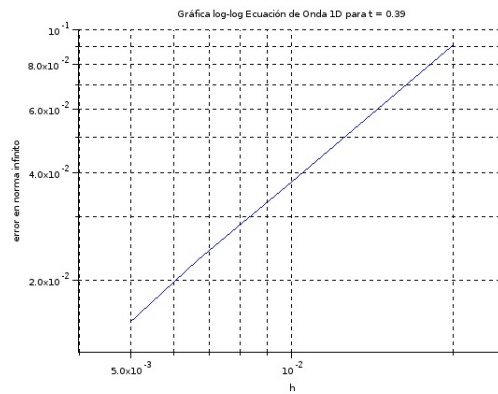


Figura 5.1.16: Gráfica del análisis del refinamiento usando diferencias finitas de la ecuación de onda, usando el gráfico de log error vs. log h, en donde se muestra la región de pendiente aproximada a -2 acorde a lo teóricamente esperado.

### 5.1.5. Ecuación de Advección con Upwind

La ecuación de advección

$$u_t + au_x = 0$$

se resolverá utilizando el esquema upwind.

Sí  $a \geq 0$  el stencil es

$$u_i^{j+1} = \frac{ak}{h}u_{i-1}^j + \left(1 - \frac{ak}{h}\right)u_i^j$$

en el caso que  $a < 0$  se tiene que

$$u_i^{j+1} = \left(1 + \frac{ak}{h}\right)u_i^j - \frac{ak}{h}u_{i+1}^j$$

donde  $k = \Delta t$  es el paso del tiempo y  $h = \Delta x$  es la longitud de los intervalos.

#### Ejemplo

Tomemos la ecuación

$$u_t + 2u_x = 0$$

en el dominio  $\Omega = [-2, 8]$  y condición inicial

$$u(x, 0) = e^{-(x-0.2)^2}$$

Esta ecuación tiene la solución analítica  $u(x, t) = u(x - at, 0)$ .

El problema se corrió en 401 nodos a un paso de tiempo  $dt = 0.012$  por 360 pasos de tiempo. El factor para que converja debe de cumplir  $\mu = \frac{a*dt}{h} < 1$ , se noto en este tipo de problemas que  $\mu$  debe de ser muy cercano a 1 para conseguir la mejor aproximación.

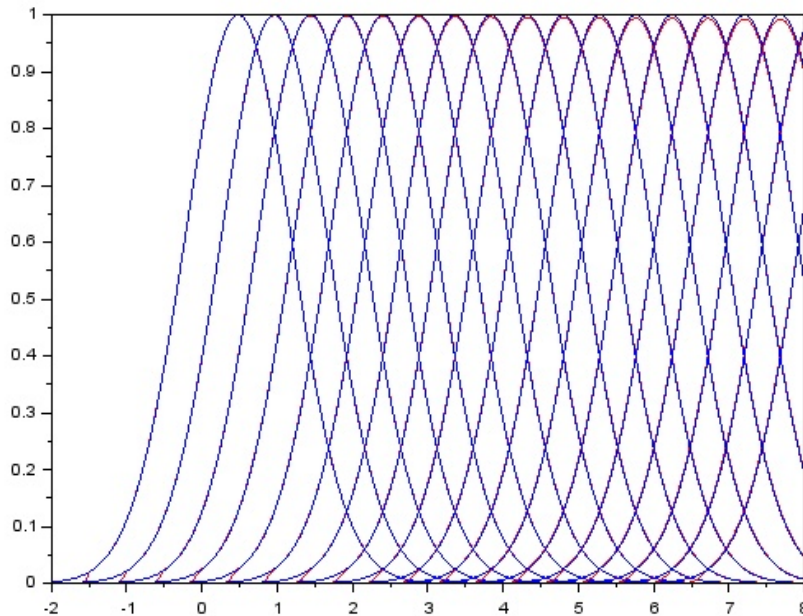


Figura 5.1.17: Solución de la ecuación de advección utilizando el esquema upwind

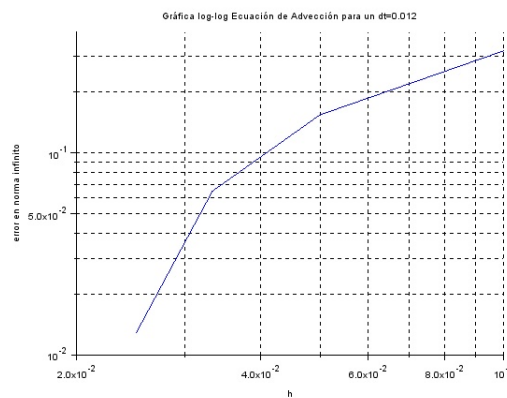


Figura 5.1.18: Gráfica del análisis del refinamiento usando diferencias finitas de la ecuación de advección, usando el gráfico de log error vs. log h, en donde se muestra la región de pendiente aproximada a -2 acorde a lo teóricamente esperado.

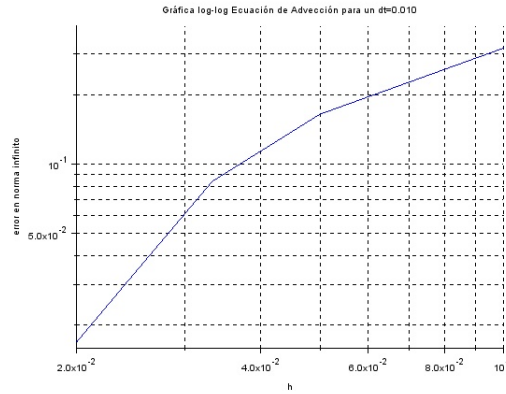


Figura 5.1.19: Gráfica del análisis del refinamiento usando diferencias finitas de la ecuación de advección, usando el gráfico de log error vs. log h, en donde se muestra la región de pendiente aproximada a -2 acorde a lo teóricamente esperado.

### 5.1.6. Ecuación de Helmholtz<sup>12</sup>

La ecuación de Helmholtz  $(\Delta + k^2)u = 0$  nombrada así por Hermann von Helmholtz, describe la propagación de ondas y la solución de dicha ecuación es requerida para resolver problemas que se presentan en aeroacustica, dispersión de ondas, problemas en geofísica etc.

En una dimensión, el problema está dado por

$$u_{xx} + k^2u = 0$$

donde  $k = \frac{\omega}{c}$  y  $\omega$  es la frecuencia de la propagación de la onda y  $c$  es la velocidad del sonido.

Sustituyendo  $u_{xx}$  por su representación en diferencias finitas

$$u_{xx} \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}$$

se obtiene

$$\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} + k^2u_i = 0$$

agrupando los nodos

$$\frac{1}{h^2}u_{i-1} + \left[ k^2 - \frac{2}{h^2} \right] u_i + \frac{1}{h^2}u_{i+1} = 0$$

donde  $h = \Delta x$ .

#### Ejemplo

Sea la ecuación

$$-u_{xx} - k^2u = 0$$

en el dominio  $\Omega = [0, 1]$  y condiciones de frontera

$$\begin{aligned} u(0) &= 1 \\ u'(1) &= iku(1) \end{aligned}$$

Dirichlet y Neumann respectivamente.

<sup>1</sup>Basado en el trabajo [21]

<sup>2</sup>Más detalles en el apéndice D

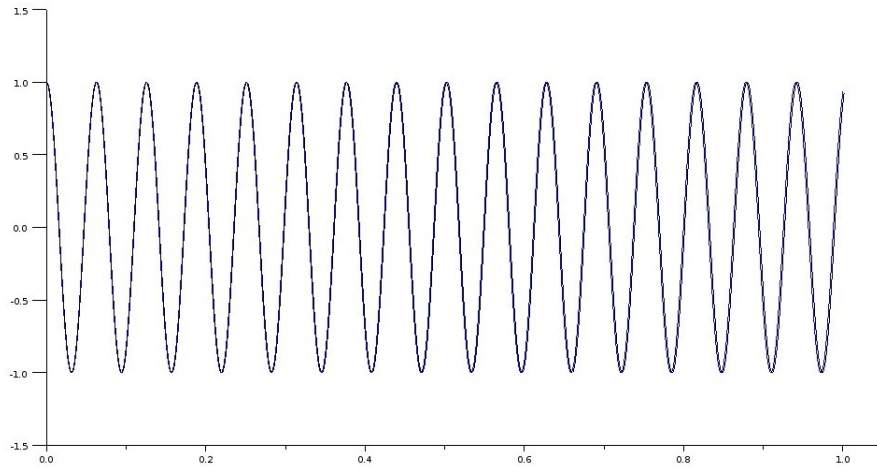


Figura 5.1.20: Solución de la ecuación de Helmholtz

El análisis de los resultados, se muestran en la gráfica *loglog*

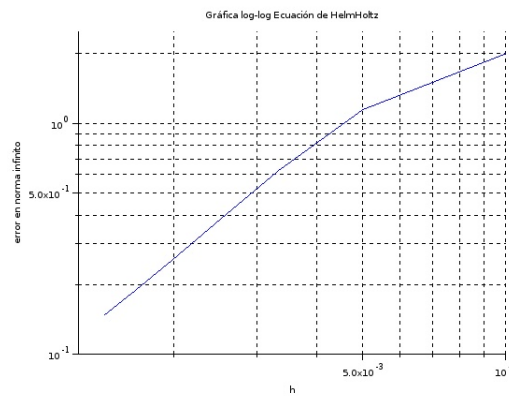


Figura 5.1.21: Gráfica *loglog* de la ecuación de Helmholtz

## 5.2. Método de Descomposición de Dominio de Schur o Subestructuración

En el trabajo [14], se considera como modelo matemático el problema de valor en la frontera (BVP) asociado con la ecuación de Poisson, con condiciones de frontera Dirichlet, definido en como:

$$\begin{aligned} -\nabla^2 u &= f_{\Omega} & \text{en } \Omega \\ u &= g_{\partial\Omega} & \text{en } \partial\Omega \end{aligned}$$

este ejemplo, gobierna los modelos de muchos sistemas de la ingeniería y de la ciencia, entre ellos el flujo de agua subterránea a través de un acuífero isotrópico, homogéneo bajo condiciones de equilibrio y es muy usado en múltiples ramas de la física, por ejemplo, gobierna la ecuación de la conducción de calor en un sólido bajo condiciones de equilibrio.

En particular se considera el problema con  $\Omega = [-1, 1] \times [-1, 1]$ , donde

$$\begin{aligned} f_{\Omega} &= 2n^2\pi^2 \text{sen}(n\pi x) \text{sen}(n\pi y) \\ g_{\partial\Omega} &= 0 \end{aligned}$$

cuya solución analítica está dada por

$$u(x, y) = \text{sen}(n\pi x) \text{sen}(n\pi y)$$

Por ejemplo para  $n = 4$ , la solución está dada por la ecuación  $u(x, y) = \text{sen}(4\pi x) \text{sen}(4\pi y)$ , cuya gráfica es

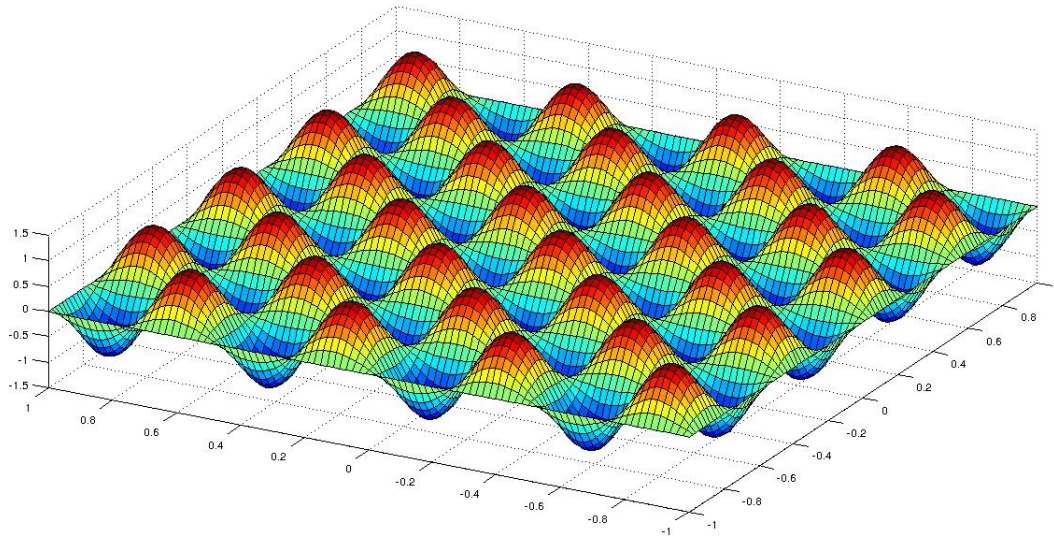


Figura 5.2.1: Solución de la ecuación de Poisson

Para las pruebas de rendimiento en las cuales se evalúa el desempeño de los programas realizados se usa  $n = 1006$ , pero es posible hacerlo con  $n \in \mathbb{N}$  más grande.

Las pruebas realizadas se hicieron usando desde 32 hasta 1024 Cores en el cluster Kan Balam, para las descomposiciones de  $31 \times 33$  y  $150 \times 150$ ,  $31 \times 33$  y  $200 \times 200$  y  $31 \times 33$  y  $250 \times 250$ , con lo que se obtiene los siguientes resultados

Subdominios	32 cores	64 cores	128 cores	256 cores	512 cores	1024 cores
$31 \times 33$ y $150 \times 150$	7315s	4016s	2619s	1941s	1541s	1298s
$31 \times 33$ y $200 \times 200$	ND	16037s	4916s	3166s	2688s	2295s
$31 \times 33$ y $250 \times 250$	ND	ND	26587s	8716s	6388s	ND

En donde si usamos las métricas de aceleración y eficiencia relativas (véase *Ápndice A*) obtenemos los siguientes resultados

Subdominio	Aceleración(Speed Up)	Aceleración Esperada	Eficiencia
$31 \times 33$ y $150 \times 150$	$S_{1024}^{32} = 5.6$	$S_{1024}^{32} = 32$	$E_{1024}^{32} = 0.2$
$31 \times 33$ y $200 \times 200$	$S_{1024}^{64} = 5.9$	$S_{1024}^{64} = 8$	$E_{1024}^{64} = 0.7$
$31 \times 33$ y $250 \times 250$	$S_{1024}^{128} = 4.1$	$S_{1024}^{128} = 4$	$E_{1024}^{128} = 1.0$

Donde  $S$  es el factor de aceleración, definido como  $S = \frac{T(1)}{T(p)}$ , que es el cociente del tiempo que se tarda en completar el cómputo utilizando un solo procesador entre el tiempo que se necesita para realizarlo en  $p$  procesadores trabajando en paralelo. La  $E$  se define como factor de eficiencia, y está dado por  $E = \frac{T(1)}{pT(p)} = \frac{s}{p}$ , el cual es el cociente del tiempo que se tarda en completar el cómputo usando un sólo core del procesador entre el número de procesadores multiplicado por el tiempo que necesita para realizarlo en  $p$  procesadores trabajando en paralelo.  $E$  será cercano a la unidad cuando el hardware se utiliza de manera eficiente.

De esta última tabla, se desprende que los algoritmos desarrollados son altamente escalables en equipos paralelos, ya que es posible obtener una alta eficiencia al encontrar descomposiciones adecuadas al Hardware. Que la eficiencia sea igual a la unidad implica que el hardware se está usando de la manera más eficiente posible.

## Capítulo 6

# Comentarios Finales y Conclusiones

Los modelos matemáticos de muchos sistemas de interés en Ciencias e Ingenierías, incluyendo una gran cantidad de sistemas importantes de Ciencias de la Tierra conducen a una gran variedad de ecuaciones diferenciales parciales, cuyos métodos de resolución están basados en el procesamiento de sistemas algebraicos de gran escala. Además, la increíble expansión experimentada por el hardware computacional existente, ha hecho posible el efectivo tratamiento de problemas de un cada vez mayor incremento de diversidad y complejidad que poseen las aplicaciones Científicas y de Ingenierías.

Los métodos de aproximación analítica a la solución de Ecuaciones Diferenciales Parciales, proporcionan frecuentemente información útil acerca del comportamiento de la solución en valores críticos de la variable dependiente, pero tienden a ser más difíciles de aplicar que los métodos numéricos. Entre las consideraciones que justifican el uso de los métodos numéricos para solucionar ecuaciones diferenciales parciales se encuentran:

- Los datos de los problemas reales presentan siempre errores de medición, y el trabajo aritmético para la solución está limitado a un número finito de cifras significativas que resultan en errores de redondeo. Por lo tanto, incluso los métodos analíticos proporcionan aproximaciones.
- La evaluación numérica de las soluciones analíticas es a menudo una tarea laboriosa y computacionalmente ineficiente, mientras que los métodos numéricos generalmente proporcionan soluciones numéricas adecuadas.

es por ello que el uso de métodos numéricos como el Método de Diferencias Finitas se han convertido en una excelente herramienta para la resolución de ecuaciones diferenciales parciales.

### 6.1. Conclusiones

A lo largo de este trabajo se desarrolló de manera minuciosa el Método de Diferencias Finitas en mallas estructuradas, para poder aplicarlo a la resolución de ecuaciones diferenciales parciales de segundo orden, en una, dos y tres dimensiones espaciales, así como también en tiempo. Con esto es posible resolver una gran variedad de ecuaciones parciales que aparecen en muchos problemas, como lo son: la ecuación de Laplace, la ecuación de Poisson, la ecuación del calor, la ecuación de la onda, la ecuación de advección, etc. La programación se realizó de manera general, de tal manera que el mismo programa pueda resolver distintos casos de las ecuaciones de segundo grado, esto se logra simplemente cambiando los valores de los coeficientes.

Los logros de este trabajo se pueden enumerar como:

- Desarrollé de manera general, del método de diferencias finitas para resolver ecuaciones diferenciales parciales de segundo grado en una, dos y tres dimensiones con mallas estructuradas. En el capítulo dos, a partir de la serie de Taylor se muestra cómo encontrar las derivadas que serán sustituidas en las ecuaciones diferenciales que se intentan resolver. Se muestra cómo se realiza el proceso en una, dos y tres dimensiones, para la primera y segunda derivada.

- Desarrollé los métodos de diferencias finitas para resolver problemas con primera y segunda derivada. En el capítulo dos, en la sección 2.6, se dan ejemplos de cómo resolver ecuaciones diferenciales parciales con el método de diferencias finitas. Se muestra cómo empezando de la ecuación a resolver, se discretiza y como se llega a obtener un sistema  $Ax = b$ , el cual al resolverlo se tiene la aproximación numérica de la ecuación en cada punto de la malla, todo esto, con distintas condiciones de frontera (Dirichlet, Neumann o Robin).
- Desarrollé los algoritmos para la solución de ecuaciones diferenciales parciales en espacio y tiempo. En el capítulo dos, secciones 2.6 y 2.7 se muestran los algoritmos para poder resolver las ecuaciones diferenciales parciales, tanto en espacio como en tiempo. Se muestra el esquema theta para ecuaciones con una derivada temporal, como la ecuación de calor. También se desarrolla el algoritmo para resolver ecuaciones con segunda derivada temporal, por ejemplo, para resolver la ecuación de ondas. Por último, se describe y se muestra el esquema upwind, para resolver ecuaciones como la advección.
- Desarrollé de los programas para resolver las ecuaciones diferenciales parciales tanto en espacio como en tiempo. Se desarrollaron los algoritmos mostrados en el capítulo dos. Se desarrollaron en lenguaje C++ para una, dos y tres dimensiones, así como en los apéndices se mostró la manera de la implementarlos con scilab y matlab(octave). El código se encuentra en la página <http://mmc2.geofisica.unam.mx/omb>.

Los métodos numéricos para resolver ecuaciones diferenciales parciales son rápidos y confiables dentro de sus rangos de aplicación, nos permiten tener una idea más clara de la solución de la ecuación diferencial en todo el dominio de interés y es relativamente sencilla su implementación. Existen situaciones en las que es preferible el uso de métodos numéricos aún cuando existe una solución analítica o semi-analítica, por ejemplo, la ecuación de calor tiene solución en series de Fourier, sin embargo los métodos numéricos como el de diferencias finitas son más eficientes para estos casos. La mayor parte de los problemas concretos son en general, complejos, y donde los métodos matemáticos para solucionarlos no son suficientes para resolverlos.

## 6.2. Trabajo Futuro

En este trabajo se mostro cómo resolver ecuaciones diferenciales parciales lineales con el Método de Diferencias Finitas. Así mismo, el método puede ser aplicado para resolver ecuaciones diferenciales parciales no lineales. Es posible también, adaptar la metodología de este trabajo a otros métodos numéricos que resuelven ecuaciones parciales, como: Volumen Finito y Elemento Finito.

Un trabajo futuro puede abarcar:

- Desarrollar el Método de Diferencias Finitas para resolver Ecuaciones Parciales No-lineales. Una vez que podemos resolver ecuaciones lineales de segundo grado, el esfuerzo se enfocaría en el siguiente grupo de ecuaciones, las no lineales. Así como las lineales, las no lineales también aparecen en problemas de ciencias e ingeniería, ejemplos de éste tipo de ecuaciones:
  - La ecuación de onda no lineal  $u_t + uu_x = 0$ .
  - La ecuación eikonal  $(u_x)^2 + (u_y)^2 + (u_z)^2 = c^2 F$ , la cual es una ecuación que se encuentra en los problemas de propagación de ondas.
- Desarrollar la misma metodología con otros métodos, como lo son: *Elemento Finito* y *Volumen Finito*. Lo que se realizaría, sería resolver ecuaciones diferenciales parciales, mostrando el cómo realizarlo (dependiendo del método), su deducción matemática y su implementación computacional en una, dos y tres dimensiones.
- Trabajar en la definición del problema en mallas no estructuradas.
- Utilizar metodologías óptimas para hacer la paralelización acorde al hardware disponible al usuario.

Así se podría tener un grupo de herramientas para poder resolver gran cantidad de problemas, de los cuales podemos escoger el que ofrezca mayores ventajas a nivel computacional o el que mejor se adapte al problema a trabajar.

# Apéndice A

## Cómputo Paralelo

Los sistemas de cómputo con procesamiento en paralelo surgen de la necesidad de resolver problemas complejos en un tiempo más o menos razonable, utilizando ventajas de memoria, velocidad de los procesadores, formas de interconexión de estos y distribución de la(s) tarea(s), a los que en su conjunto se denomina arquitectura en paralelo. Se entiende por una arquitectura en paralelo a un conjunto de procesadores interconectados que son capaces de cooperar en la solución de un problema. Así, para resolver un problema en particular, se usa una o combinación de múltiples arquitecturas (topologías), ya que cada una ofrece ventajas y desventajas que tienen que tomarse en cuenta antes de implementar la solución del problema en una arquitectura en particular, *véase*

### Arquitecturas

Se explicarán las dos clasificaciones de computadoras más conocidas de la actualidad. La primera es la clasificación de Flynn, en la cual se tienen en cuenta sistemas con uno o varios cores. La segunda, más moderna, sólo se tienen en cuanto los sistemas con más de un core.

### Clasificación de Flynn

Es una clasificación clásica de aruitecturas de computadoras, en la cual se toman en cuenta uno o más procesadores.

Se basa en el flujo que siguen los datos dentro de la máquina y de las instrucciones sobre los datos. Se define como *flujo de intrucciones* al conjunto de instrucciones secuenciales que son ejecutadas por un único core y como *flujo de datos* al flujo secuencial de datos requeridos por el flujo de instrucciones.

Dado lo anterior, Flynn clasifica los sistemas en cuatro categorías:

#### Single Instruction stream, Single Data stream (SISD)

Los sistemas de este tipo se caracterizan por tener un único flujo de instrucciones sobre un único flujo de datos, es decir, se ejecuta una instrucción detrás de otra. Este es el concepto de arquitectura serie de Von Neumann donde, en cualquier momento, sólo se ejecuta una única instrucción, un ejemplo de estos sistemas son las máquinas secuenciales convencionales.

#### Single Instruction stream, Multiple Data stream (SIMD)

Estos sistemas tienen un único flujo de instrucciones que operan sobre múltiples flujos de datos. El procesamiento es síncrono, la ejecución de las instrucciones sigue siendo secuencial como en el caso anterior, todos los elementos realizan una misma instrucción pero sobre una gran cantidad de datos. Por este motivo existirá concurrencia de operación, es decir, esta clasificación es el origen de la máquina paralela. El funcionamiento de este tipo de sistemas es el siguiente. La unidad de control manda una misma instrucción a todas las unidades de proceso (ALUs). Las unidades de proceso operan sobre datos diferentes pero con la misma instrucción recibida.

**Multiple Instruction stream, Single Data stream (MISD)**

Sistemas con múltiples instrucciones que operan sobre un único flujo de datos.

**Multiple Instruction stream, Multiple Data stream (MIMD)**

Sistemas con un flujo de múltiples instrucciones que operan sobre múltiples datos. Estos sistemas empezaron a utilizarse antes de la década de los 80s. Son sistemas con memoria compartida que permiten ejecutar varios procesos simultáneamente (sistema multiprocesador).

Cuando las unidades de proceso reciben datos de una memoria no compartida estos sistemas reciben el nombre de MULTIPLE SISD (MSISD). En arquitecturas con varias unidades de control (MISD Y MIMD), existe otro nivel superior con una unidad de control que se encarga de controlar todas las unidades de control del sistema (ejemplo de estos sistemas son las máquinas paralelas actuales).

**Computadoras Paralelas**

La clasificación moderna de computadoras paralelas hace alusión única y exclusivamente a los sistemas que tienen más de un core (i.e máquinas paralelas). Existen dos tipos de sistemas teniendo en cuenta su acoplamiento:

- Los sistemas fuertemente acoplados. Son aquellos en los que los procesadores dependen unos de otros. Los sistemas fuertemente acoplados se comunican a través de una memoria común.
- Los sistemas débilmente acoplados. Son aquellos en los que existe poca interacción entre los diferentes procesadores que forman el sistema.

Atendiendo ésta y a otras características, la clasificación moderna divide a los sistemas en dos tipos:

- Sistemas multiprocesador (fuertemente acoplados)
- Sistemas multicomputadoras (débilmente acoplados)

**Equipo Paralelo de Memoria Compartida**

Un multiprocesador o multicore puede verse como una computadora paralela compuesta por varios cores interconectados que comparten un mismo sistema de memoria. Los sistemas multiprocesadores o multicores son arquitecturas MIMD(Multiple Instruction Stream, Multiple Data Stream) con memoria compartida. Tienen un único espacio de direcciones para todos los procesadores y los mecanismos de comunicación se basan en el paso de mensajes desde el punto de vista del programador. Dado que los procesadores comparten diferentes módulos de memoria, pudiendo acceder a un mismo módulo varios cores, a los multiprocesadores también se les llama sistemas de memoria compartida.

Para hacer uso de la memoria compartida por más de un core, se requiere hacer uso de técnicas de semáforos que mantienen la integridad de la memoria; esta arquitectura no puede crecer mucho en el número de procesadores interconectados por la saturación rápida del bus o del medio de interconexión. Dependiendo de la forma en que los procesadores comparten la memoria, se clasifican en sistemas multiprocesador UMA, NUMA, COMA y Pipeline.

**Equipo Paralelo de Memoria Distribuida**

Los sistemas multicomputadoras se pueden ver como una computadora paralela en el cual cada core tiene su propia memoria local. En estos sistemas la memoria se encuentra distribuida y no compartida como en los sistemas multiprocesador. Los procesadores se comunican a través de paso de mensajes, ya que éstos sólo tienen acceso directo a su memoria local y no a las memorias del resto de los procesadores.

La transferencia de los datos se realiza a través de la red de interconexión que conecta un subconjunto de procesadores con otro subconjunto. La transferencia de unos procesadores a otros se realiza por múltiples transferencias entre procesadores conectados dependiendo del establecimiento de dicha red.

Dado que la memoria está distribuida entre los diferentes elementos de proceso, estos sistemas reciben el nombre de distribuidos. Por otra parte, estos sistemas son débilmente acoplados, ya que los módulos funcionan de forma casi independiente unos de otros. Este tipo de memoria distribuida es de acceso lento por ser peticiones a través de la red, pero es una forma muy efectiva de tener acceso a un gran volumen de memoria.

### Equipo Paralelo de Memoria Compartida-Distribuida

La tendencia actual en las máquinas paralelas es de aprovechar las facilidades de programación que ofrecen los ambientes de memoria compartida y la escalabilidad de las ambientes de memoria distribuida. En este modelo se conectan entre sí módulos de multiprocesadores, pero se mantiene la visión global de la memoria a pesar de que es distribuida. El desarrollo de sistemas operativos y compiladores del dominio público (Linux y software GNU), estándares para el pase de mensajes (MPI), conexión universal a periféricos (PCI), etc. han hecho posible tomar ventaja de los económicos recursos computacionales de producción masiva (CPU, discos, redes). La principal desventaja que presenta a los proveedores de multicomputadoras es que deben satisfacer una amplia gama de usuarios, es decir, deben ser generales. Esto aumenta los costos de diseños y producción de equipos, así como los costos de desarrollo de software que va con ellos: sistema operativo, compiladores y aplicaciones. Todos estos costos deben ser añadidos cuando se hace una venta. Por supuesto alguien que sólo necesita procesadores y un mecanismo de pase de mensajes no debería pagar por todos estos añadidos que nunca usará.

Los cluster se pueden clasificar en dos tipos según sus características físicas:

- Cluster homogéneo si todos los procesadores y/o nodos participantes en el equipo paralelo son iguales en capacidad de cómputo (en la cual es permitido variar la cantidad de memoria o disco duro en cada core).
- Cluster heterogéneo es aquel en que al menos uno de los procesadores y/o nodos participantes en el equipo paralelo son de distinta capacidad de cómputo.

Los cluster pueden formarse de diversos equipos; los más comunes son los de computadoras personales, pero es creciente el uso de computadoras multiprocesador de más de un core de memoria compartida interconectados por red con los demás nodos del mismo tipo, incluso el uso de computadoras multiprocesador de procesadores vectoriales Pipeline. Los cluster armados con la configuración anterior tienen grandes ventajas para procesamiento paralelo:

- La reciente explosión en redes implica que la mayoría de los componentes necesarios para construir un cluster son vendidos en altos volúmenes y por lo tanto son económicos. Ahorros adicionales se pueden obtener debido a que sólo se necesitará una tarjeta de vídeo, un monitor y un teclado por cluster. El mercado de los multiprocesadores es más reducido y más costoso.
- Remplazar un componente defectuoso en un cluster es relativamente trivial comparado con hacerlo en un multiprocesador, permitiendo una mayor disponibilidad de clusters cuidadosamente diseñados.

Desventajas del uso de clusters de computadoras personales para procesamiento paralelo:

- Con raras excepciones, los equipos de redes generales producidos masivamente no están diseñados para procesamiento paralelo y típicamente su latencia es alta y los anchos de banda pequeños comparados con multiprocesadores. Dado que los clusters explotan tecnología que sea económica, los enlaces en el sistema no son veloces implicando que la comunicación entre componentes debe pasar por un proceso de protocolos de negociación lentos, incrementando seriamente la latencia. En muchos y en el mejor de los casos (debido a costos) se recurre a una red tipo Fast Ethernet restringimiento la escalabilidad del cluster.

- Hay poco soporte de software para manejar un cluster como un sistema integrado.
- Los procesadores no son tan eficientes como los procesadores usados en los multiprocesadores para manejar múltiples usuarios y/o procesos. Esto hace que el rendimiento de los clusters se degrade con relativamente pocos usuarios y/o procesos.
- Muchas aplicaciones importantes disponibles en multiprocesadores y optimizadas para ciertas arquitecturas, no lo están en clusters.

**Tipos de Cluster**

Básicamente existen tres tipo de clusters, cada uno de ellos ofrece ventajas y desventajas, el tipo más adecuado para el cómputo científico de alto rendimiento, pero existen aplicaciones científicas que pueden usar más de un tipo al mismo tiempo.

- Alta-disponibilidad (Fail-over o High-Availability): este tipo de cluster esta diseñado para mantener uno o varios servicios disponibles incluso a costa de rendimiento, ya que su función principal es que el servicio jamás tenga interrupciones como por ejemplo un servicio de bases de datos.
- Alto-rendimiento (HPC o High Performance Computing): este tipo de cluster está diseñado para obtener el máximo rendimiento de la aplicación utilizada incluso a costa de la disponibilidad del sistema, es decir el cluster puede sufrir caídas, este tipo de configuración está orientada a procesos que requieran mucha capacidad de cálculo.
- Balanceo de Carga (Load-balancing): este tipo de cluster esta diseñado para balancear la carga de trabajo entre varios servidores, lo que permite tener, por ejemplo, un servicio de cálculo intensivo multiusuarios que detecte tiempos muertos del proceso de un usuario para ejecutar en dichos tiempos procesos de otros usuarios.

**Métricas de Desempeño**

Las métricas de desempeño del procesamiento de alguna tarea en paralelo es un factor importante para medir la eficiencia y consumo de recursos al resolver una tarea con un número determinado de procesadores y recursos relacionados de la interconexión de éstos.

Entre las métricas para medir desempeño en las cuales como premisa se mantiene fijo el tamaño del problema, destacan las siguientes: Factor de aceleración, eficiencia y fracción serial. Cada una de ellas mide algo en particular y sólo la combinación de estas dan un panorama general del desempeño del procesamiento en paralelo de un problema en particular en una arquitectura determinada al ser comparada con otras.

**Factor de Aceleración(Speed Up)**

Se define como el cociente del tiempo que se tarda en completar el cómputo de la tarea usando un sólo core entre el tiempo que necesita para realizarlo en  $p$  procesadores trabajando en paralelo

$$s = \frac{T(1)}{T(p)} \tag{A.0.1}$$

en ambos casos se asume que se usará el mejor algoritmo tanto para un solo core como para  $p$  procesadores. Esta métrica en el caso ideal debería de aumentar de forma lineal al aumento del número de procesadores.

**Eficiencia**

Se define como el cociente del tiempo que se tarda en completar el cómputo de la tarea usando un solo core entre el número de procesadores multiplicado por el tiempo que necesita para realizarlo en  $p$  procesadores trabajando en paralelo

$$e = \frac{T(1)}{pT(p)} = \frac{s}{p} \tag{A.0.2}$$

Este valor será cercano a la unidad cuando el hardware se esté usando de manera eficiente, en caso contrario el hardware será desaprovechado.

### Fracción Serial

Se define como el cociente del tiempo que se tarda en completar el cómputo de la parte secuencial de una tarea entre el tiempo que se tarda en completar el cómputo de la tarea usando un solo core

$$f = \frac{T_s}{T(1)} \quad (\text{A.0.3})$$

usando la ley de Amdahl

$$T(p) = T_s + \frac{T_p}{p} \quad (\text{A.0.4})$$

y rescribiéndola en términos de factor de aceleración, obtenemos la forma operativa del cálculo de la fracción serial que adquiere la forma siguiente

$$f = \frac{\frac{1}{s} - \frac{1}{p}}{1 - \frac{1}{p}} \quad (\text{A.0.5})$$

Esta métrica permite ver las inconsistencias en el balance de cargas, ya que su valor debiera de tender a cero en el caso ideal, por ello un incremento en el valor de  $f$  es un aviso de granularidad fina con la correspondiente sobrecarga en los procesos de comunicación.

## Apéndice B

# Método de Descomposición de Dominio de Subestructuración<sup>1</sup>

La solución numérica por los esquemas tradicionales de discretización tipo deiferencias finitas generan una discretización del problema, la cual es usada para generar un sistema de ecuaciones  $Ax = b$ .

Esta única matriz puede ser paralelizada mediante el uso de memoria compartida y/o distribuida. En la memoria compartida, la mejor opción es *openmp*, véase[23], en el caso de memoria distribuida se utiliza *mpi*, véase [24]. Existen otras formas de realizar la paralelización, la más eficiente es la técnica que se aplica a la discretización del problema mediante el método de descomposición de dominio de subestructuración, esta genera una descomposición del sistema lineal  $Ax = b$  en múltiples sistemas lineales (véase sección 4.5) acoplados entre si, sólo por los nodos que comparten en la discretización de los subdominios.

Sin pérdida de generalidad se considerarán problemas con valor en la frontera (VBVP) de la forma

$$\begin{aligned}\mathcal{L}u &= f \\ u &= g\end{aligned}\tag{B.0.1}$$

donde

$$\mathcal{L}u = -\nabla \cdot a \cdot \nabla u + cu$$

donde  $a$  es una matriz positiva definida y simétrica, con  $c > 0$ . Como caso particular del operador elíptico de orden 2 y para ejemplificar se toma un dominio  $\Omega \subset \mathbb{R}^2$  con fronteras poligonales, es decir,  $\Omega$  es un conjunto abierto acotado y conexa tal que su frontera  $\partial\Omega$  es la unión de un número finito de polígonos.

Si se multiplica la ecuación  $-\nabla \cdot a \cdot \nabla u + cu = f_\Omega$  por  $v \in V = H_0^1(\Omega)$  se obtiene

$$-v(-\nabla \cdot a \cdot \nabla u + cu) = vf_\Omega$$

entonces, aplicando el Teorema de Green

$$\int_{\Omega} (\nabla v \cdot a \cdot \nabla u + cuv) dx = \int_{\Omega} vf_\Omega dx$$

Definiendo el operador bilineal

$$a(u, v) = \int_{\Omega} (\nabla v \cdot a \cdot \nabla u + cuv) dx$$

y la funcional lineal

$$l(v) = \langle f, v \rangle = \int_{\Omega} vf_\Omega dx$$

entonces se puede reescribir el problema en forma variacional, haciendo uso de la forma bilineal  $a(\cdot, \cdot)$  y la función lineal  $l(\cdot)$ .

---

<sup>1</sup>Este apéndice está basado en el trabajo doctoral del Dr. Antonio Carrillo Ledesma [14].

Donde las funciones lineales definidas por tramos en  $\Omega_E$  en este caso serán polinomios de orden uno en cada variable de forma separada y cuya restricción de  $\phi_i$  a  $\Omega_E$  es  $\phi_i^{(E)}$ . Para simplificar los cálculos, se supone que la matriz  $\underline{a} = a \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ , entonces se tiene que la integral del lado izquierdo queda escrita como

$$\int_{\Omega} (a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j) dx dy = \int_{\Omega} f_{\Omega} \phi_j dx dy$$

donde

$$\begin{aligned} K_{ij} &= \int_{\Omega} (a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j) dx dy \\ &= \sum_{e=1}^E \int_{\Omega_e} (a \nabla \phi_i^e \cdot \nabla \phi_j^e + c \phi_i^e \phi_j^e) dx dy \\ &= \sum_{e=1}^E \int_{\Omega_e} \left( a \left[ \frac{\partial \phi_i^e}{\partial x} \frac{\partial \phi_j^e}{\partial x} + \frac{\partial \phi_i^e}{\partial y} \frac{\partial \phi_j^e}{\partial y} \right] + c \phi_i^e \phi_j^e \right) dx dy \end{aligned}$$

y el lado derecho como

$$\begin{aligned} F_j &= \int_{\Omega} f_{\Omega} \phi_j dx dy \\ &= \sum_{e=1}^E \int_{\Omega} f_{\Omega} \phi_j dx dy \end{aligned}$$

Si se considera el problema dado en la ecuación B.0.1 en el dominio  $\Omega$ , el cual es subdividido en  $E$  subdominios  $\Omega_i$  con  $i = 1, 2, \dots, E$  sin traslape, también conocida como malla gruesa  $T_H$ , es decir

$$\Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j \quad \bar{\Omega} = \bigcup_{i=1}^n \bar{\Omega}_i$$

y al conjunto

$$\Gamma = \bigcup_{i=1}^E \Gamma_i \quad \text{si } \Gamma_i = \partial \Omega_i / \partial \Omega$$

se le llama la frontera interior del dominio  $\Omega$ .

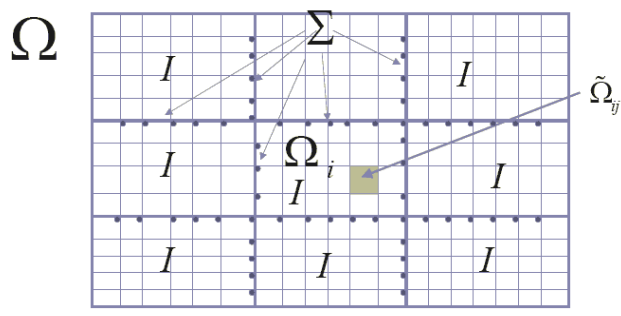


Figura B.0.1: Dominios

Un ejemplo de un dominio  $\Omega$  y su descomposición en subdominios  $\Omega_i$  y cada  $\Omega_i$  a su vez descompuesto en  $\Omega_E$  subdominios. Sin pérdida de generalidad se toma  $g = 0$  en  $\partial \Omega$ , nótese que siempre es posible poner el problema de la B.0.1 como uno de condiciones de frontera Dirichlet que se nulifiquen mediante la adecuada manipulación del término del lado derecho de la ecuación.

Primeramente sea  $D \subset H_0^1(\Omega)$  un espacio lineal de funciones de dimensión finita  $N$ , en el cual esté definido un producto interior denotado para cada  $u, v \in D$  por

$$u \cdot v = \langle u, v \rangle$$

Considerando la existencia de los subconjuntos linealmente independientes

$$B \subset \widetilde{D}, B_I \subset \widetilde{D}_I, B_\Gamma \subset \widetilde{D}_\Gamma$$

$$B_\Gamma \subset \widetilde{D}_\Gamma, B_{\Gamma J} \subset \widetilde{D}_{\Gamma 1}, B_{\Gamma M} \subset \widetilde{D}_{\Gamma 2}$$

los cuales satisfacen

$$B = B_I \cup B_\Gamma$$

y

$$\overline{B_\Gamma} = B_{\Gamma J} \cup B_{\Gamma M}$$

el espacio generado por cada uno de los subconjuntos  $B_\Gamma$  y  $\overline{B_\Gamma}$  es  $\widetilde{D}_\Gamma$ , sin embargo, nótese la propiedad de que los miembros de  $B_\Gamma$  tienen soporte local. Se definen las bases

$$\begin{aligned} B_I &= \{w_I^1, \dots, w_I^{N_I}\} \\ B_{\Gamma M} &= \{w_M^1, \dots, w_M^{N_\Gamma}\} \\ B_{\Gamma J} &= \{w_J^1, \dots, w_J^{N_\Gamma}\} \end{aligned}$$

de las funcionales lineales  $\phi_i$  en  $\Omega$ .

Entonces definiendo para toda  $\delta = 1, \dots, K$ , la matriz  $N_\delta \times N_\delta$

$$A \equiv [\langle w_I^i, w_I^j \rangle]$$

que sólo esta definida en cada subespacio (subdominio  $\Omega_\delta$ ). Entonces, la matriz virtual  $A_{II}$  está dada por la matriz diagonal de la forma

$$A_{II} \equiv \begin{pmatrix} A_{II}^1 & & & \\ & A_{II}^2 & & \\ & & \ddots & \\ & & & A_{II}^E \end{pmatrix}$$

donde el resto de la matriz fuera de la diagonal son bloques es cero.

De forma similar se definen

$$\begin{aligned} A_{I\Gamma}^\delta &\equiv [\langle w_I^i, w_\Gamma^\alpha \rangle] \\ A_{\Gamma I}^\delta &\equiv [\langle w_\Gamma^\alpha, w_I^i \rangle] \\ A_{\Gamma\Gamma}^\delta &\equiv [\langle w_\Gamma^\alpha, w_\Gamma^\alpha \rangle] \end{aligned}$$

para toda  $\delta = 1, \dots, E$ , obserérvase que como  $\overline{B_\Gamma} = B_{\Gamma J} \cup B_{\Gamma M}$  entonces

$$A_{\Gamma\Gamma}^\delta = [\langle w_\Gamma^\alpha, w_\Gamma^\alpha \rangle] = [\langle w_{\Gamma J}^\alpha, w_{\Gamma J}^\alpha \rangle] + [\langle w_{\Gamma M}^\alpha, w_{\Gamma M}^\alpha \rangle]$$

también que  $A_{I\Gamma}^\delta = (A_{\Gamma I}^\delta)^T$ . Entonces las matrices virtuales  $A_{\Gamma I}$ ,  $A_{I\Gamma}$ , y  $A_{\Gamma\Gamma}$  quedarían definidas como

$$\begin{aligned} A_{\Gamma I} &= [A_{\Gamma I}^1 \quad A_{\Gamma I}^2 \quad \dots \quad A_{\Gamma I}^E] \\ A_{I\Gamma} &= \begin{bmatrix} A_{I\Gamma}^1 \\ A_{I\Gamma}^2 \\ \vdots \\ A_{I\Gamma}^E \end{bmatrix} \\ A_{\Gamma\Gamma} &= \left[ \sum_{i=1}^E A_{\Gamma\Gamma}^i \right] \end{aligned}$$

donde  $\left[\sum_{i=1}^E A_{\Gamma\Gamma}^i\right]$  es construida sumando las  $A_{\Gamma\Gamma}^i$  según el orden de los nodos globales versus los nodos locales.

También se considera al vector  $u = (u_1, \dots, u_E)$  el cual se escribe como  $u = (u_I, u_\Gamma)$  donde  $u_I = (u_1, \dots, N_I)$  y  $u_\Gamma = (u_1, \dots, N_\Gamma)$ .

Así, el sistema virtual

$$\begin{aligned} A_{II}u_I + A_{I\Gamma}u_\Gamma &= b_I \\ A_{\Gamma I}u_I + A_{\Gamma\Gamma}u_\Gamma &= b_\Gamma \end{aligned} \quad (\text{B.0.2})$$

queda expresado como

$$\begin{aligned} \begin{pmatrix} A_{II}^1 & & \\ & \ddots & \\ & & A_{II}^E \end{pmatrix} \begin{pmatrix} u_{I_1} \\ \vdots \\ u_{I_E} \end{pmatrix} + \begin{pmatrix} A_{I\Gamma}^1 \\ \vdots \\ A_{I\Gamma}^E \end{pmatrix} \begin{pmatrix} u_{\Gamma_1} \\ \vdots \\ u_{\Gamma_E} \end{pmatrix} &= \begin{pmatrix} b_{I_1} \\ \vdots \\ b_{I_E} \end{pmatrix} \\ (A_{\Gamma I}^1 \ \dots \ A_{\Gamma I}^E) \begin{pmatrix} u_{I_1} \\ \vdots \\ u_{I_E} \end{pmatrix} + (A^{\Gamma\Gamma}) \begin{pmatrix} u_{\Gamma_1} \\ \vdots \\ u_{\Gamma_E} \end{pmatrix} &= \begin{pmatrix} b_{\Gamma_1} \\ \vdots \\ b_{\Gamma_E} \end{pmatrix} \end{aligned}$$

o de manera compacta  $Au = b$ , nótese que las matrices  $A_{\Gamma\Gamma}^i$ ,  $A_{\Gamma I}^i$ ,  $A_{I\Gamma}^i$  y  $A_{II}^i$  son matrices bandadas. Si ahora tratamos de obtener  $u_I$  de la primera ecuación del sistema B.0.2 se obtiene

$$u_I = (A_{II})^{-1} (b_I - A_{I\Gamma}u_\Gamma)$$

sustituyendo en la segunda ecuación del mismo sistema se obtiene

$$(A_{\Gamma\Gamma} - A_{\Gamma I} (A_{II})^{-1} A_{I\Gamma}) u_\Gamma = b_\Gamma - A_{\Gamma I} (A_{II})^{-1} b_I \quad (\text{B.0.3})$$

en la cual los nodos interiores no figuran en la ecuación y todo queda en función de los nodos de la frontera interior  $u_\Gamma$ .

A la matriz formada por  $A_{\Gamma\Gamma} - A_{\Gamma I} (A_{II})^{-1} A_{I\Gamma}$  se le conoce como el complemento de Schur global y se le denota como

$$S = A_{\Gamma\Gamma} - A_{\Gamma I} (A_{II})^{-1} A_{I\Gamma}$$

En este caso, como se está planteando en términos de subdominios  $\Omega_i$ , con  $i = 1, \dots, E$ , entonces las matrices  $A_{\Gamma\Gamma}^i$ ,  $A_{\Gamma I}^i$ ,  $A_{I\Gamma}^i$  y  $A_{II}^i$  quedan definidas de manera local, así que se procede a definir el complemento de Schur local como

$$S_i = A_{\Gamma\Gamma}^i - A_{\Gamma I}^i (A_{II}^i)^{-1} A_{I\Gamma}^i$$

de forma adicional se define

$$b_i = b_{\Gamma_i} - A_{\Gamma I}^i (A_{II}^i)^{-1} b_{I_i}$$

El sistema dado por B.0.3 se escribe como

$$Su_\Gamma = b \quad (\text{B.0.4})$$

y queda definido de manera virtual a partir de

$$\left[ \sum_{i=1}^E S_i \right] u_\Gamma = \left[ \sum_{i=1}^E b_i \right] \quad (\text{B.0.5})$$

donde  $\left[\sum_{i=1}^E S_i\right]$  y  $\left[\sum_{i=1}^E b_i\right]$  podrían ser construidas sumando las  $S_i$  y  $b_i$  respectivamente según el orden de los nodos globales versus los nodos locales.

El sistema lineal virtual obtenido de B.0.4 se resuelve (dependiendo de si es o no simétrico) eficientemente usando el método de Gradiente Conjugado o alguna variante de GMRES, para ello no es necesario construir

la matriz  $S$  con las contribuciones de cada  $S_i$  correspondientes al subdominio  $i$ , lo que se hace es pasar a cada subdominio el vector  $u_{\Gamma_i}$  correspondiente a la  $i$ -ésima iteración del método iterativo usado, para que en cada subdominio se evalúe  $\tilde{u}_{\Gamma}^i = S_i u_{\Gamma_i}$  localmente y con el resultado se forma el vector  $\tilde{u}_{\Gamma} = \sum_{i=1}^E \tilde{u}_{\Gamma_i}$  y se continúe con los demás pasos del método. Esto es ideal para una implementación en paralelo del método de Gradiente Conjugado o variante de GMRES.

Una vez resuelto el sistema B.0.5 en el que se ha encontrado la solución para los nodos de la frontera interior  $u_{\Gamma}$ , entonces se debe resolver localmente los  $u_{I_i}$ , correspondientes a los nodos interiores para cada subespacio  $\Omega_i$ , para poder realizar esto se emplea

$$u_{I_i} = (A_{II}^i)^{-1} (b_{I_i} - A_{I\Gamma}^i u_{\Gamma_i})$$

para cada  $i = 1, \dots, E$ , quedando así resuelto el problema  $Au = b$  tanto en los nodos interiores  $u_{I_i}$  como en los de la frontera interior  $U_{\Gamma_i}$  correspondientes a cada subespacio  $\Omega_i$ .

**Observación 1.** Las matrices locales  $S_i$  y  $(A_{II}^i)^{-1}$  no se construyen, ya que estas serían matrices densas y su construcción es computacionalmente costosa, y como sólo interesa el producto  $S y_{\Gamma}$ , o más precisamente  $\left[ \sum_{i=1}^E S_i \right] y_{\Gamma}$ , entonces si  $y_{\Gamma}$  es el vector correspondiente al subdominio  $i$ , se obtiene

$$\tilde{u}_{\Gamma_i} = \left( A_{\Gamma\Gamma}^i - A_{\Gamma I}^i (A_{II}^i)^{-1} A_{I\Gamma}^i \right) y_{\Gamma_i}$$

Para evaluar de forma eficiente esta expresión, se realizan las siguientes operaciones equivalentes

$$\begin{aligned} x_1 &= A_{\Gamma\Gamma}^i y_{\Gamma_i} \\ x_2 &= \left( A_{\Gamma I}^i (A_{II}^i)^{-1} A_{I\Gamma}^i \right) y_{\Gamma_i} \\ \tilde{u}_{\Gamma_i} &= x_1 - x_2 \end{aligned}$$

la primera y tercera expresión no tienen ningún problema en su evaluación, para la segunda expresión se debe evaluar

$$x_3 = A_{II}^i y_{\Gamma_i}$$

con este resultado intermedio se debe calcular

$$x_4 = (A_{II}^i)^{-1} x_3$$

pero al no contarse con  $(A_{II}^i)^{-1}$ , entonces se multiplica la expresión por  $A_{II}^i$  obteniendo

$$A_{II}^i x_4 = A_{II}^i (A_{II}^i)^{-1} x_3$$

simplificando

$$A_{II}^i x_4 = x_3$$

Esta última expresión puede ser resuelta usando *Factorización LU*, *Gradiente Conjugado* o alguna variante de *GMRES* –cada una de estas opciones tiene ventajas y desventajas computacionales que deben ser evaluadas al momento de implementar el código para un problema particular–. Una vez obtenido  $x_4$ , se puede calcular

$$x_2 = A_{\Gamma I}^i x_4$$

así

$$\tilde{u}_{\Gamma_i} = x_1 - x_2$$

completando la secuencia de operaciones necesarias para obtener  $S_i y_{\Gamma_i}$ .

**Observación 2.** En el caso del cálculo

$$b_i = b_{\Gamma_i} - A_{\Gamma I}^i (A_{II}^i)^{-1} b_{I_i}$$

algo análogo a lo anterior deberá de hacerse, ya que de nuevo está involucrado  $(A_{II}^i)^{-1}$ , por ello se debe usar el siguiente procedimiento para evaluar de forma eficiente esta expresión, realizando las operaciones equivalentes

$$y_1 = (A_{II}^i)^{-1} b_{I_i}$$

multiplicando por  $A_{II}^i$  se obtiene

$$A_{II}^i y_1 = A_{II}^i (A_{II}^i)^{-1} b_{I_i}$$

simplificando

$$A_{II}^i y_1 = b_{I_i}$$

donde esta última expresión puede ser resuelta usando *Factorización LU*, *Gradiente Conjugado* o alguna variante de *GMRES*, luego se hace

$$y_2 = A_{\Gamma I}^i y_1$$

y para finalizar, se obtiene

$$b_i = b_{\Gamma_i} - y_2$$

**Observación 3.** En la evaluación de

$$u_{I_i} = (A_{II}^i)^{-1} (b_{I_i} - A_{\Gamma I}^i u_{\Gamma_i})$$

está nuevamente involucrado el término  $(A_{II}^i)^{-1}$ , por esto se usa el siguiente procedimiento para evaluar de forma eficiente esta expresión, realizando las operaciones equivalentes

$$\begin{aligned} x_4 &= b_{I_i} - A_{\Gamma I}^i u_{\Gamma_i} \\ u_{I_i} &= (A_{II}^i)^{-1} x_4 \end{aligned}$$

multiplicando por  $A_{II}^i$  a la última expresión se obtiene

$$A_{II}^i u_{I_i} = A_{II}^i (A_{II}^i)^{-1} x_4$$

simplificando

$$A_{II}^i u_{I_i} = x_4$$

donde esta última evaluación puede ser resuelta nuevamente usando *Factorización LU*, *Cholesky*, *Gradiente Conjugado* o alguna variante de *GMRES*.

Como se indico en las últimas observaciones, para resolver el sistema  $A_{II}^i x = b$  puede usar *Factorización LU*, *Factorización Cholesky*, *Gradiente Conjugado*, alguna variante de *GMRES* o cualquier otro método para resolver sistemas lineales, pero deberá de usarse aquel que proporcione la mayor velocidad en el cálculo o que consuma la menor cantidad de memoria (ambas condicionantes son mutuamente excluyentes), por ello la decisión de que método usar deberá de tomarse al momento de tener que resolver un problema particular en un equipo dado y básicamente el condicionante es el tamaño de la matriz  $A_{II}^i$ .

Para usar el método de *Factorización LU*, se deberá primeramente de factorizar la matriz bandada  $A_{II}^i$  en una matriz  $LU$ , la cual es bandada pero incrementa el tamaño de la banda a más del doble, pero esta operación sólo se deberá de realizar una vez en cada subdominio, y para solucionar los diversos sistemas lineales  $A_{II}^i x = b$  sólo será necesario evaluar los sistemas

$$\begin{aligned} Ly &= b \\ Ux &= y \end{aligned}$$

Esto proporciona una manera eficiente de evaluar el sistema lineal pero el consumo en memoria para un problema particular puede ser excesivo.

Por ello, si el problema involucra una gran cantidad de nodos interiores y el equipo en el que se implantará la ejecución del programa tiene una cantidad de memoria limitada, es recomendable usar el método de *Gradiente Conjugado* o alguna variante de *GMRES*, este consume una cantidad de memoria adicional pequeña, pero puede consumir muchas iteraciones con el consecuente aumento de tiempo de cómputo para alcanzar la precisión de la Factorización LU.

De esta forma, es posible adaptar el código para tomar en cuenta desde la implementación hasta al equipo de cómputo al que se tenga acceso y poder sacar el máximo provecho al método de subestructuración en la resolución de problemas de gran envergadura.

Para más detalle, véase [14].

## Apéndice C

# Crank-Nicolson 1D Para la Ecuación de Calor Scilab, C++ y MatLab

En este apéndice se mostrará cómo resolver una ecuación diferencial parcial en una dimensión con el esquema de *Crank-Nicolson*, por ejemplo, la ecuación de calor.

Consideremos el siguiente problema, que consiste en la ecuación

$$u_t = u_{xx}$$

con  $\Omega = [0, 1]$ , sujeta a las condiciones

$$u(0, t) = u(1, t) = 0$$

y

$$u(x, 0) = \text{sen}(\pi x)$$

El código del ejemplo, en C++, Scilab y MatLab puede obtenerse de la página:

<http://mmc2.geofisica.unam.mx/omb>.

### C.1. Scilab

El siguiente programa para el entorno SciLab (<http://www.scilab.org/>) es el encargado de resolver el problema de la ecuación de calor por diferencias finitas. Scilab es un software de cómputo numérico libre, el cual incluye un lenguaje de programación de alto nivel para poder crear *scripts* que se ejecutan en el entorno. Scilab es multiplataforma (\*nix, mac, windows) y posee cientos de funciones matemáticas ya definidas, listas para poder ser usadas por el desarrollador.

```
1 // Facultad de Ciencias
2 // Universidad Nacional Autonoma de Mexico
3 // Mexico D.F.
4 //
5 // Autor: Omar Jonathan Mendoza Bernal
6 // Página: mmc2.geofisica.unam.mx/omb
7 //
8 // Programa que forma parte del trabajo de Tesis
9 // "Resolución de Ecuaciones Diferenciales Parciales
10 // Mediante el Método de Diferencias Finitas y su Paralelización"
11 //
12 // Código liberado bajo la licencia GPL ver. 2.0
13
14
15 // Crank-Nicolson
16 // Para una EDP de segundo orden
17 //  $u_t + a(x)u_{xx} + b(x)u_x + c(x)u = f$ 
18
19 // Dominio
```

```

20 // l0<x<l
21 // 0<t<T
22
23 // Condiciones de frontera Dirichlet
24 // u(0,t) = u(l,t) = constante 0 < t < T cond de frontera
25
26 // Condición inicial
27 // u(x,0) = g(x) l0 <= x <= l
28
29 // Datos de entrada
30 // intervalo [l0, l]
31 // entero m>=3
32 // entero N>=1
33
34 // Salida
35 // aproximaciones w_ij a u(x_i, t_j)
36
37
38 //
39 // Funciones de los coeficientes
40 function y = a(x)
41     y = -1;
42 endfunction
43
44 function y = b(x)
45     y = 0;
46 endfunction
47
48 function y = c(x)
49     y = 0;
50 endfunction
51
52 function y = f(x)
53     y = 0;
54 endfunction
55
56 //
57 // función de la condición inicial
58 function y = condicion_inicial(x)
59     y = sin(x * %pi);
60 endfunction
61
62 // Condiciones de frontera
63 // Sólo Dirichlet
64 cond_izq = 0;
65 cond_der = 0;
66
67
68 // Datos de entrada
69 l0 = 0;
70 l = 1; // intervalo [0,1]
71 m = 11; // Numero de nodos
72 M = m - 2; // Nodos interiores
73
74 // División del espacio y del tiempo
75 h = (l - l0)/(m-1);
76 k = 0.025; // Paso del tiempo
77 N = 21; // Numero de iteraciones
78
79 // Aw^(j+1) = Bw^j + f^j
80 // creo el vector w donde se guardará la solución para cada tiempo
81 // A matriz del lado izquierdo
82 // B matriz del lado derecho
83 // B_prima para guardar el resultado de Bw^j
84 // ff que es el vector de los valores de f en cada nodo
85 w = zeros(M,1);
86 ff = zeros(M,1);
87 A = zeros(M,M);
88 B = zeros(M,M);
89 w_sol = zeros(m,m)
90
91 // Se crea el espacio de la solución o malla
92 espacio = zeros(m,1)
93 for i = 1:m
94     xx = l0 + (i-1)*h;
95     espacio(i) = xx;

```

```

96 end
97 disp(espacio , "Espacio")
98
99 // Condición inicial
100 for i=1:M
101     w(i) = condicion_inicial(espacio(i+1));
102 end
103
104 w_sol(1) = cond_izq;
105 for kk = 1:M
106     w_sol(kk + 1) = w(kk);
107 end
108 w_sol(m) = cond_der;
109 plot(espacio , w_sol);
110
111 disp(w, "Condiciones_iniciales")
112
113
114 // primer renglon de cada matriz
115 A(1,1) = 1/k - a(10 + h)/(h*h);
116 A(1,2) = a(10 + h)/(2*h*h) + b(10 + h)/(4*h) ;
117
118 B(1,1) = 1/k + a(10 + h)/(h*h) - c(10 + h);
119 B(1,2) = - a(10 + h)/(2*h*h) - b(10 + h)/(4*h);
120
121 ff(1) = f(10 + h) - cond_izq;
122
123 // se completa las matrices desde el renglon 2 hasta el m-2
124 for i = 2:M-1
125     A(i, i-1) = a(10 + i*h)/(2*h*h) - b(10 + i*h)/(4*h) ;
126     A(i,i) = 1/k - a(10 + i*h)/(h*h);
127     A(i,i+1) = a(10 + i*h)/(2*h*h) + b(10 + i*h)/(4*h) ;
128
129     B(i, i-1) = - a(10 + i*h)/(2*h*h) + b(10 + i*h)/(4*h);
130     B(i,i) = 1/k + a(10 + i*h)/(h*h) - c(10 + i*h);
131     B(i,i+1) = - a(10 + i*h)/(2*h*h) - b(10 + i*h)/(4*h);
132 end
133
134 // último renglon de cada matriz
135 A(M,M-1) = a(1 - h)/(2*h*h) - b(1-h)/(4*h) ;
136 A(M,M) = 1/k - a(1 - h)/(h*h);
137
138 B(M,M-1) = - a(1-h)/(2*h*h) + b(1-h)/(4*h);
139 B(M,M) = 1/k + a(1-h)/(h*h) - c(1-h);
140
141 ff(M) = f(1 - h) - cond_der;
142
143 // Resolvemos el sistema iterativamente
144 for j=1:N
145     t = j*k;
146     B_prima = B * w + ff;
147     w = inv(A) * B_prima;
148     disp(t, "tiempo")
149     disp(w, "Sol")
150
151     w_sol(1) = cond_izq;
152     for kk = 1:M
153         w_sol(kk + 1) = w(kk);
154     end
155     w_sol(m) = cond_izq;
156
157     plot(espacio , w_sol);
158 end

```

## C.2. C++

El mismo problema, ahora utilizando el lenguaje C++ con la librería gmm++. C++ es uno de los lenguajes más utilizados cuando se necesita eficiencia en el uso de los recursos de la(s) computadora(s). Gmm++ es una biblioteca libre, la cual nos permite de una manera sencilla, crear y manipular vectores y matrices, así como, ofrece los resolvedores de sistemas  $Ax = b$  más comunes y eficientes. Además de por su facilidad de uso, se opto por utilizarla en este trabajo, y, a continuación se muestra parte de su uso para resolver la ecuación de calor.

La librería gmm++ puede ser descargada desde (<http://download.gna.org/getfem/html/homepage/gmm/>).

```

1 // Facultad de Ciencias
2 // Universidad Nacional Autonoma de Mexico
3 // Mexico D.F.
4 //
5 // Autor: Omar Jonathan Mendoza Bernal
6 // Página: mmc2.geofisica.unam.mx/omb
7 //
8 // Programa que forma parte del trabajo de Tesis
9 // "Resolución de Ecuaciones Diferenciales Parciales
10 // Mediante el Método de Diferencias Finitas y su Paralelización"
11 //
12 // Código liberado bajo la licencia GPL ver. 2.0
13
14 // Crank-Nicolson
15 // Para una EDP de segundo orden
16 //  $u_t + a(x)u_{xx} + b(x)u_x + c(x)u = f$ 
17
18 // Dominio
19 //  $l_0 < x < l$ 
20 //  $0 < t < T$ 
21
22 // Condiciones de frontera Dirichlet
23 //  $u(0,t) = u(l,t) = \text{constante}$   $0 < t < T$  cond de frontera
24
25 // Condición inicial
26 //  $u(x,0) = g(x)$   $l_0 \leq x \leq l$ 
27
28 // Datos de entrada
29 // intervalo  $[l_0, l]$ 
30 // entero  $m \geq 3$ 
31 // entero  $N \geq 1$ 
32
33 // Salida
34 // aproximaciones  $w_{ij}$  a  $u(x_i, t_j)$ 
35 #include <gmm/gmm.h>
36
37 #include <iostream>
38 #include <cmath>
39
40 #define PI 3.14159
41 #define SIMETRICA 1
42
43 //
44 // Funciones de los coeficientes
45 double a(double x) {
46     return -1;
47 }
48
49 double b(double x) {
50     return 0;
51 }
52
53 double c(double x) {
54     return 0;
55 }
56
57 double f(double x) {
58     return 0;
59 }
60
61 // Condiciones de frontera
62 double cond_izq () {
63     return 0;
64 }
65
66 double cond_der () {
67     return 0;
68 }
69
70 // Condicion inicial
71 double condicion_inicial (double x) {
72
73     double val = sin(x*PI);
74
75     return val;

```

```

76 }
77
78 int main(int argc, const char *argv[]) {
79
80     // Intervalo
81     const double l0 = 0;
82     const double l = 1;
83
84     const int m = 11; // Numero de nodos
85     const int M = m - 2; // Numero de nodos interiores
86
87     // Division del espacio y tiempo
88     double h = (l - l0) / (m - 1);
89     double k = 0.01; // paso del tiempo
90     int N = 10; // Numero de iteraciones
91
92     //  $Aw^{(j+1)} = Bw^{(j)} + f^{(j)}$ 
93     // creo el vector w donde se guardará la solución para cada tiempo
94     // A matriz del lado izquierdo
95     // B matriz del lado derecho
96     // ff que es el vector de los valores de f en cada nodo
97     gmm::row_matrix< gmm::rsvector<double> > A(M, M);
98     gmm::row_matrix< gmm::rsvector<double> > B(M, M);
99     std::vector<double> sol(m); // Para la solución total del sistema (con fronteras)
100    std::vector<double> B_p(M);
101    std::vector<double> w(M);
102    std::vector<double> w2(M);
103    std::vector<double> ff(M);
104
105    // Contiene las coordenadas de los nodos internos
106    std::vector<double> espacio(M);
107
108    std::cout << "h_" << h << std::endl;
109    for (int i = 0; i < M; i++) {
110        double xx = l0 + (i+1)*h;
111        espacio[i] = xx;
112    }
113
114    std::cout << "Espacio\n" << espacio << std::endl;
115
116    // Condiciones iniciales
117    for (int i = 0; i < M; i++) {
118        w2[i] = condicion_inicial(espacio[i]);
119    }
120
121    sol[0] = cond_izq();
122    for (int k = 0; k < M; k++)
123        sol[k+1] = w2[k];
124    sol[M-1] = cond_der();
125
126    std::cout << "Condiciones_iniciales\n" << sol << std::endl;
127
128    // primer renglon de cada matriz
129    A(0,0) = 1/k - a(l0 + h)/(h*h);
130    A(0,1) = a(l0 + h)/(2*h*h) + b(l0 + h)/(4*h) ;
131
132    B(0,0) = 1/k + a(l0 + h)/(h*h) - c(l0 + h);
133    B(0,1) = - a(l0 + h)/(2*h*h) - b(l0 + h)/(4*h);
134
135    ff[0] = f(l0 + h) - cond_izq();
136
137    // se completa las matrices desde el renglon 2 hasta el m-2
138    for (int i = 1; i < M-1; i++) {
139        A(i, i-1) = a(l0 + i*h)/(2*h*h) - b(l0 + i*h)/(4*h) ;
140        A(i, i) = 1/k - a(l0 + i*h)/(h*h);
141        A(i, i+1) = a(l0 + i*h)/(2*h*h) + b(l0 + i*h)/(4*h) ;
142
143        B(i, i-1) = - a(l0 + i*h)/(2*h*h) + b(l0 + i*h)/(4*h);
144        B(i, i) = 1/k + a(l0 + i*h)/(h*h) - c(l0 + i*h);
145        B(i, i+1) = - a(l0 + i*h)/(2*h*h) - b(l0 + i*h)/(4*h);
146    }
147
148    // último renglon de cada matriz
149    A(M-1,M-2) = a(l - h)/(2*h*h) - b(l-h)/(4*h) ;
150    A(M-1,M-1) = 1/k - a(l - h)/(h*h);
151

```

```

152     B(M-1,M-2) = - a(1-h)/(2*h*h) + b(1-h)/(4*h);
153     B(M-1,M-1) = 1/k + a(1-h)/(h*h) - c(1-h);
154
155     ff[M-1] = f(1 - h) - cond_der();
156
157     // Empieza la iteración de la solución
158     double t;
159     for (int i = 1; i < N; i++) {
160         t = i * k;
161
162         gmm::mult(B,w2,B_p);
163         gmm::add(ff ,B_p);
164
165         if (SIMETRICA) {
166             gmm::identity_matrix PS; // Optional scalar product for cg
167             gmm::identity_matrix PR; // Optional preconditioner
168             gmm::iteration iter(10E-6); // Iteration object with the max residu
169
170             gmm::cg(A, w, B_p, PS, PR, iter); // Conjugate gradient
171
172             w2 = w;
173
174             sol[0] = cond_izq();
175             for (int k = 0; k < M; k++)
176                 sol[k+1] = w[k];
177             sol[m-1] = cond_der();
178
179             std::cout << "Tiempo_" << t << std::endl;
180             std::cout << "Sol_CGM\n" << sol << std::endl;
181
182         } else {
183             size_t restart = 50; // restart parameter for GMRES
184             gmm::iteration iter(10E-6); // Iteration object with the max residu
185             gmm::ilut_precond<gmm::row_matrix<gmm::rsvector<double>>> Pre(A, 10, 1e-4);
186             gmm::mult(B,w2,B_p);
187             gmm::add(ff ,B_p);
188             gmm::gmres(A, w, B_p, Pre, restart , iter); // execute the GMRES algorithm
189             w2 = w;
190
191             sol[0] = cond_izq();
192             for (int k = 0; k < M; k++)
193                 sol[k+1] = w[k];
194             sol[m-1] = cond_der();
195
196             std::cout << "Tiempo_" << t << std::endl;
197             std::cout << "Sol_GMRES_precod_ILUT\n" << sol << std::endl;
198
199         }
200     }
201
202     return 0;
203 }

```

### C.3. MatLab

MatLab (<http://www.mathworks.com/products/matlab/>) es un entorno para cómputo científico no libre, sumamente usado en el mundo. Ofrece un lenguaje de programación de alto nivel. Contiene enorme cantidad de funciones ya predefinidas que el desarrollador puede utilizar al instante. Una de las ventajas de MatLab es su enorme capacidad de ser ampliado y adecuarlo a las necesidades del usuario mediante los llamados *toolboxes*, que entre otras cosas ofrecen la capacidad de realizar: procesamiento de imágenes, análisis de datos, optimización, etc. La opción de software libre de MatLab es Octave (<http://www.gnu.org/software/octave/>). A continuación se muestra el programa para resolver la ecuación de calor en el entorno, mediante el uso de su lenguaje de programación.

```

1  % // Facultad de Ciencias
2  % // Universidad Nacional Autonoma de Mexico
3  % // Mexico D.F.
4  % //
5  % // Autor: Omar Jonathan Mendoza Bernal
6  % // Página: mmc2.geofisica.unam.mx/omb
7  % //

```

```

8  %% Programa que forma parte del trabajo de Tesis
9  %% "Resolución de Ecuaciones Diferenciales Parciales
10 %% Mediante el Método de Diferencias Finitas y su Paralelización"
11 %%
12 %% Código liberado bajo la licencia GPL ver. 2.0
13
14 %% Crank-Nicolson
15 %% Para una EDP de segundo orden
16 %%  $u_t + a(x)u_{xx} + b(x)u_x + c(x)u = f$ 
17 %%
18 %% Dominio
19 %%  $l_0 < x < l$ 
20 %%  $0 < t < T$ 
21 %%
22 %% Condiciones de frontera Dirichlet
23 %%  $u(0,t) = u(l,t) = \text{constante}$   $0 < t < T$  cond de frontera
24 %%
25 %% Condición inicial
26 %%  $u(x,0) = g(x)$   $l_0 \leq x \leq l$ 
27 %%
28 %% Datos de entrada
29 %% intervalo  $[l_0, l]$ 
30 %% entero  $m \geq 3$ 
31 %% entero  $N \geq 1$ 
32 %%
33 %% Salida
34 %% aproximaciones  $w_{ij}$  a  $u(x_i, t_j)$ 
35 %%
36 %%
37
38 function CN(m)
39 %% Condiciones de frontera
40 %% Sólo Dirichlet
41 cond_izq = 0;
42 cond_der = 0;
43
44
45 %% Datos de entrada
46 l0 = 0;
47 l = 1; %% intervalo  $[0,1]$ 
48 m = 11; %% Numero de nodos
49 M = m - 2; %% Nodos interiores
50
51 %% División del espacio y del tiempo
52 h = (l - l0)/(m-1);
53 k = 0.025; %% Paso del tiempo
54 N = 21; %% Numero de iteraciones
55
56 %%  $Aw^{(j+1)} = Bw^{(j)} + f^{(j)}$ 
57 %% creo el vector w donde se guardará la solución para cada tiempo
58 %% A matriz del lado izquierdo
59 %% B matriz del lado derecho
60 %% ff que es el vector de los valores de f en cada nodo
61 w = zeros(M,1);
62 ff = zeros(M,1);
63 A = zeros(M,M);
64 B = zeros(M,M);
65 w_sol = zeros(m,m);
66
67 %% Se crea el espacio de la solución o malla
68 espacio = zeros(M,1);
69 for i = 1:m
70     xx = l0 + (i-1)*h;
71     espacio(i) = xx;
72 end
73 disp(espacio, 'Espacio')
74
75 %% Condición inicial
76 for i = 1:M
77     w(i) = condicion_inicial(espacio(i+1));
78 end
79
80 w_sol(1) = cond_izq;
81 for kk = 1:M
82     w_sol(kk + 1) = w(kk);
83 end

```

```

84 w_sol(m) = cond_der;
85
86 plot(espacio , w_sol);
87 hold on
88
89 %disp(w, 'Condiciones iniciales ')
90
91
92 %// primer renglon de cada matriz
93 A(1,1) = 1/k - a(10 + h)/(h*h);
94 A(1,2) = a(10 + h)/(2*h*h) + b(10 + h)/(4*h) ;
95
96 B(1,1) = 1/k + a(10 + h)/(h*h) - c(10 + h);
97 B(1,2) = - a(10 + h)/(2*h*h) - b(10 + h)/(4*h);
98
99 ff(1) = f(10 + h) - cond_izq;
100
101 %// se completa las matrices desde el renglon 2 hasta el m-2
102 for i = 2:M-1
103     A(i, i-1) = a(10 + i*h)/(2*h*h) - b(10 + i*h)/(4*h) ;
104     A(i, i) = 1/k - a(10 + i*h)/(h*h);
105     A(i, i+1) = a(10 + i*h)/(2*h*h) + b(10 + i*h)/(4*h) ;
106
107     B(i, i-1) = - a(10 + i*h)/(2*h*h) + b(10 + i*h)/(4*h);
108     B(i, i) = 1/k + a(10 + i*h)/(h*h) - c(10 + i*h);
109     B(i, i+1) = - a(10 + i*h)/(2*h*h) - b(10 + i*h)/(4*h);
110 end
111
112 %// último renglon de cada matriz
113 A(M,M-1) = a(1 - h)/(2*h*h) - b(1-h)/(4*h) ;
114 A(M,M) = 1/k - a(1 - h)/(h*h);
115
116 B(M,M-1) = - a(1-h)/(2*h*h) + b(1-h)/(4*h);
117 B(M,M) = 1/k + a(1-h)/(h*h) - c(1-h);
118
119 ff(M) = f(1 - h) - cond_der;
120
121 %// muestro las matrices
122 %//printf("Matriz A\n");
123 %//disp(A);
124 %//printf("Matriz B\n");
125 %//disp(B);
126
127
128 %// B_prima para guardar el resultado de Bw^j
129 %// Resolvemos el sistema iterativamente
130 for j=1:N
131     t = j*k;
132     B_prima = B * w + ff;
133     w = inv(A) * B_prima;
134     %disp(t, 'tiempo ')
135     %disp(w, 'Sol ')
136
137     w_sol(1) = cond_izq;
138     for kk = 1:M
139         w_sol(kk + 1) = w(kk);
140     end
141     w_sol(m) = cond_izq;
142     %figure(j);
143     plot(espacio , w_sol);
144     hold on
145
146 end
147 return
148
149
150 %//
151 %// Funciones de los coeficientes
152
153 function y = a(x)
154     y = -1;
155 return
156
157 function y = b(x)
158     y = 0;
159 return

```

```
160
161 function y = c(x)
162     y = 0;
163 return
164
165 function y = f(x)
166     y = 0;
167 return
168
169 function y = condicion_inicial(x)
170 y = sin(x * pi);
171 return
```

## Apéndice D

# Ecuación de Advección con upwind en python

### D.1. Python

Python ([www.python.org](http://www.python.org)) es un lenguaje moderno y multiparadigma de aprendizaje fácil y rápido por su sintaxis clara y sencilla, se trata de un lenguaje interpretado de tipado dinámico. NumPy (Numerical Python, [www.numpy.org](http://www.numpy.org)) es una biblioteca open source para computo científico. NumPy permite trabajar con vectores y matrices de una manera más natural. La biblioteca contiene una basta cantidad de funciones matemáticas útiles, incluyendo funciones de álgebra lineal, transformadas de Fourier, generadores de números aleatorios, etc. Matplotlib ([www.matplotlib.org](http://www.matplotlib.org)) es una biblioteca de generación de gráficos para python, produce tales gráficos a partir de listas de python o vectores y matrices definidos en numpy.

### D.2. Ejemplo

Tomemos la ecuación

$$u_t + 2u_x = 0$$

en el dominio  $\Omega = [-2, 8]$  y condición inicial

$$u(x, 0) = e^{-(x-0.2)^2}$$

Esta ecuación tiene la solución analítica  $u(x, t) = u(x - at, 0)$ .

El problema se corrió en 401 nodos a un paso de tiempo  $dt = 0.012$  por 360 pasos de tiempo. El factor para que converja debe de cumplir  $\mu = \frac{a*dt}{h} < 1$ , se noto en este tipo de problemas que  $\mu$  debe de ser muy cercano a 1 para conseguir la mejor aproximación.

```
# -*- coding: utf-8 -*-
"""
@author: omar jonathan mendoza bernal

Upwind scheme for
u_t + a(x) u_x = 0

Ejemplo
u_t + a u_x = 0
Si u(x,0) = f(x) es la condición inicial
la solución analitica es
u(x,t) = f(x-at)
"""
```

```

from math import exp
from scipy import sparse
import numpy as np
import matplotlib.pyplot as plt

# Ecuación
#  $u_t + 2 u_x = 0$ 
coef_a = 2

def condicion_inicial (x):
    """
    Condición inicial de la ecuación
    """
    y = exp (-(x - 0.2)*(x - 0.02))
    return y

def sol_analitica (x, t):
    """
    Solución analítica de la ecuación
    """
    y = exp(-(x-2*t)*(x-2*t))

    return y

#####
# Dominio
#####
a = -2.0
b = 8.0

# Partición del dominio
nNodos = 401
h = (b - a)/(nNodos - 1)

# Intervalo de tiempo
dt = 0.012

# creo el vector w donde se guardará la solución para cada tiempo
# B matriz del lado derecho
w = np.zeros((nNodos,1))
B = np.zeros((nNodos, nNodos))
espacio = np.zeros((nNodos,1))

for i in xrange(nNodos):
    xx_ = a + i*h
    espacio[i] = xx_
    w[i] = condicion_inicial(xx_)

print "Espacio"
print espacio

print "Condición Inicial"

```

```

print w

mu = coef_a * dt / h

# Para una buena aproximación
# mu debe ser muy cercano a 1
if mu <= 1:
    print "mu ", mu
    print "Buena aproximación"
else:
    print "mu ", mu
    print "Mala aproximación"

if coef_a >= 0:
    B[0,0] = 1 - mu
    for i in xrange(1, nNodos):
        B[i,i-1] = mu
        B[i,i] = 1 - mu
else:
    B[0,0] = 1 + mu;
    B[0,1] = -mu;

    # se completa las matrices desde el renglon 2 hasta el m-2
    for i in xrange(1, nNodos-1):
        B[i,i] = 1 + mu;
        B[i, i+1] = -mu;

    B[nNodos-1,nNodos-1] = 1 + mu

# para guardar la solución analítica
xx = np.zeros((nNodos,1));

iteraciones = 201

# Matriz sparse csr
Bs = sparse.csr_matrix(B);

# Resolvemos el sistema iterativamente
for j in xrange(1, iteraciones):
    t = j*dt;
    w = Bs*w;

    # Imprimir cada 20 iteraciones
    if j%20 == 0:
        print "t", t
        print "w", w
        plt.plot(espacio, w)

```

Gráfica de solución del problema

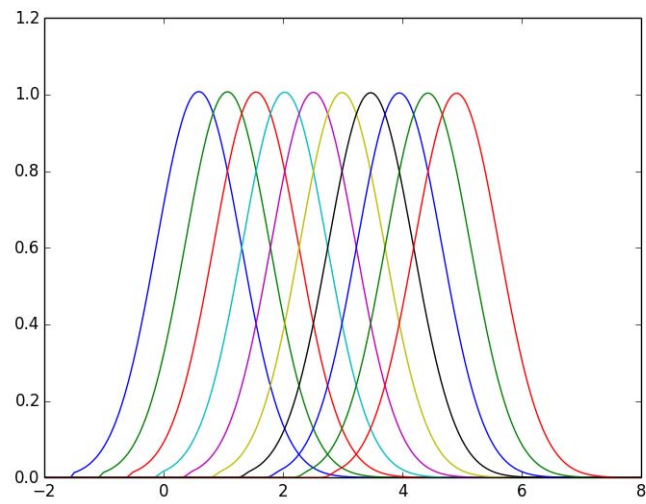


Figura D.2.1: Solución de la ecuación de advección

## Apéndice E

# Ecuación de Helmholtz<sup>1</sup>

El estudio de los fenómenos de onda es importante en muchas áreas de las ciencias y de las ingenierías. La ecuación de Helmholtz nombrada así por Hermann von Helmholtz, surge del estudio de la propagación de ondas y la solución de dicha ecuación es requerida para resolver problemas que se presentan en aeroacústica, dispersión de ondas, problemas en geofísica etc.

La formulación matemática en un medio homogéneo está dada por

$$\Delta u + k^2 u = 0$$

con  $k = \frac{\omega}{c}$  y donde  $\omega$  es la frecuencia de la propagación de la onda y  $c$  es la velocidad del sonido.

Si bien ha habido un tremendo avance en las áreas de métodos computacionales para resolver ecuaciones diferenciales parciales, resolver la ecuación lineal de Helmholtz con número de onda alto ( $k$ ) de manera numérica es uno de las tareas más complicadas para el cómputo científico. A altos valores de  $k$  la ecuación se vuelve altamente oscilatoria. Por ejemplo, supóngase que se tiene una malla con distancia entre nodos igual a  $h$ , para aproximar el comportamiento oscilatorio de la ecuación debe de pasar que  $kh$  sea pequeño, dado que se requiere la solución para  $k$  grandes, implica que  $h$  debe ser muy pequeña. Simulaciones numéricas y estudios teóricos han confirmado que aunque  $kh$  esté fija y sea una buena aproximación para cierto problema, la aproximación numérica se dispersa rápidamente conforme se va incrementando el valor de  $k$ . A esto se le conoce como el efecto *pollución* (pollution effect). El pollution effect solo se ha eliminado para problemas unidimensionales de la ecuación, no así para problemas en dos y tres dimensiones. Más aún, para asegurar una buena aproximación numérica, es necesario forzar la condición  $k^2 h < 1$ . Esto implica que el sistema de ecuaciones es proporcional a  $k^3$ , véase [21].

El desarrollo de aproximaciones numéricas eficientes para la solución de la ecuación de Helmholtz con  $k$  grandes, es un área activa de la investigación actual.

### E.1. Ejemplo Numérico

Sea la ecuación

$$-u_{xx} - k^2 u = 0$$

en  $\Omega = (0, 1)$  y condiciones de frontera

$$\begin{aligned} u(0) &= 1 \\ u'(1) &= iku(1) \end{aligned}$$

Discretizando la ecuación, se obtiene

$$-\left(\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}\right) - k^2 u_i = 0$$

---

<sup>1</sup>Con base en el trabajo [21]

agrupando términos

$$\left(-\frac{1}{h^2}\right)u_{i-1} + \left(\frac{2}{h^2} - k^2\right)u_i + \left(-\frac{1}{h^2}\right)u_{i+1} = 0$$

El manejo de la frontera Neumann

$$u'(1) = iku(1)$$

queda de la siguiente forma, sí

$$\alpha u'(b) + \beta u(b) = \gamma$$

sustituyendo la derivada que aparece por su representación en diferencia finita

$$\alpha \frac{u_{n+1} - u_{n-1}}{2h} + \beta u(b) = \gamma$$

despejando el término  $u_{n+1}$

$$u_{n+1} = u_{n-1} - \frac{2h\beta}{\alpha}u_n + \frac{2h\gamma}{\alpha}$$

sustituyendo el termino  $u_{n+1}$  en

$$\frac{u_{n-1} - 2u_n + u_{n+1}}{h^2} = f_n$$

se tiene que

$$\left(\frac{1}{h^2}\right)u_{n-1} + \left(-\frac{1}{h^2} - \frac{\beta}{\alpha h}\right)u_n = \frac{f_n}{2} - \frac{\gamma}{\alpha h} \tag{E.1.1}$$

que es la forma en que se trabajan las fronteras Neumann. La frontera del problema está dada por

$$u'(1) = iku(1)$$

que es igual a

$$u'(1) - iku(1) = 0$$

sustituyendo los valores en E.1.1, es decir,  $\alpha = 1$ ,  $\beta = -ik$  y  $\gamma = 0$ , por lo que se obtiene

$$\left(\frac{1}{h^2}\right)u_{n-1} - \left(\frac{1}{h^2} + \frac{ik}{h}\right)u_n = \frac{f(1)}{2}$$

y de esta manera queda representada la condición de frontera Neumann en el lado derecho del dominio  $\Omega$ .

El siguiente programa desarrollado en Scilab implementa la solución a la ecuación de Helmholtz que se acaba de discutir.

```

1 // Facultad de Ciencias
2 // Universidad Nacional Autonoma de Mexico
3 // Mexico D.F.
4 //
5 // Autores: Antonio Carrillo y Omar Jonathan Mendoza Bernal
6 // Página: mmc2.geofisica.unam.mx/omb
7 //
8 // Programa que forma parte del trabajo de Tesis
9 // "Resolución de Ecuaciones Diferenciales Parciales
10 // Mediante el Método de Diferencias Finitas y su Paralelización"
11 //
12 // Código liberado bajo la licencia GPL ver. 2.0
13
14 // Ejemplo de una ecuación diferencial parcial en 1D
15 // Ecuación de Helmholtz
16 // -Uxx-k^2U=0
17 // 0<=U<=1
18 // U(0)=1 y Ux(1)=ikU(1)
19
20
21 function y=LadoDerecho(x)
22     y = 0.0;
23 endfunction
24

```

```

25 function y=SolucionAnalitica(x, k)
26     y = exp(%i*k*x);
27 endfunction
28
29
30 K = 100;
31 KK = K*K;
32
33 a = 0;           // Inicio dominio
34 c = 1;           // Fin dominio
35 M = 601;         // Partición
36 N = M-1;         // Nodos interiores
37 h = (c-a)/(M-1); // Incremento en la malla
38 Y0 = 1;          // Condición Dirchlet inicial en el inicio del dominio
39 Y1 = %i*K;        // Condición Neumann inicial en el fin del dominio
40 A = zeros(N,N); // Matriz A
41 b = zeros(N);   // Vector b
42
43
44
45 R = -1/(h^2);
46 P = 2/(h^2)-KK;
47 Q = -1/(h^2);
48
49
50 // Primer renglon de la matriz A y vector b
51 A(1,1) = P;
52 A(1,2) = Q;
53 b(1) = LadoDerecho(a)-Y0*R; // Frontera dirichlet
54
55 // Renglones intermedios de la matriz A y vector b
56 for i=2:N-1
57     A(i,i-1) = R;
58     A(i,i) = P;
59     A(i,i+1) = Q;
60     b(i) = LadoDerecho(a+h*(i-1));
61 end
62
63 // Relgion final de la matriz A y vector b
64 A(N,N-1) = 1/(h^2);
65 A(N,N) = -1/(h^2) + Y1/h;
66 b(N) = LadoDerecho(c)/2;
67
68
69
70 // Resuleve el sistema lineal Ax=b
71 x=inv(A)*b;
72
73 ESC = 5;
74 xxx=zeros(M*ESC,1);
75 zzz=zeros(M*ESC,1);
76 for i=1:M*ESC
77     xxx(i)=a+h/ESC*(i-1);
78     zzz(i)=SolucionAnalitica(xxx(i),K);
79 end
80
81 // Prepara la graficacion
82 xx=zeros(M,1);
83 zz=zeros(M,1);
84 for i=1:M
85     xx(i)=a+h*(i-1);
86     zz(i)=SolucionAnalitica(xx(i),K);
87 end
88
89 yy=zeros(M,1);
90 yy(1)=Y0; // Condicion inicial
91 for i=1:N
92     yy(i+1)=x(i);
93 end
94
95 // Grafica la solucion de la Ecuacion Diferencial Parcial en 1D
96 plot2d(xx,yy)
97 //plot2d(xxx,zzz)
98
99 //printf("yy ")
100 //disp(yy)

```

```
101 //printf("zz ")
102 //disp (zz)
103 //printf("zzz ")
104 //disp (zzz)
```

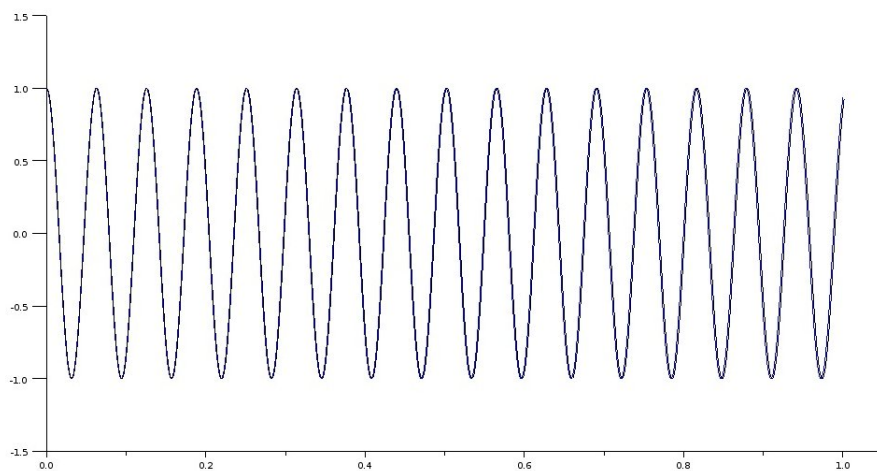


Figura E.1.1: Solución de la ecuación de Helmholtz

## Apéndice F

# Gráficos en MatLab

Las gráficas presentes en este trabajo, fueron realizadas en el entorno MatLab, para generarlas, lo más sencillo es moverse al directorio donde se encuentran los datos y después en la consola interactiva introducir los siguientes comandos:

```
x = importdata( 'EspacioX.dat' );  
y = importdata( 'EspacioY.dat' );  
z = importdata( 'Datos.dat' );
```

```
mesh(x, y, z)
```

Este ejemplo es para una gráfica en 3D, donde

- En 'x' se almacena los puntos de la partición del eje x
- En 'y' se almacena los puntos de la partición del eje y
- En 'z' el resultado de evaluar una función  $f(x, y)$  o en este caso la solución del método de diferencias finitas en cada punto  $(x, y)$  de la malla
- $mesh(x, y, z)$  se encarga de generar la gráfica

# Bibliografía

- [1] Carrillo Ledesma, Antonio, “Aplicaciones del Cómputo en Paralelo a la modelación de Sistemas Terrestres”. Tesis de Maestría, Instituto de Geofísica, UNAM, 2006.
- [2] De la Cruz Salas, Luis M. “Método de volumen finito para flujo en una fase”.
- [3] Lang, Serge, “Álgebra Lineal”, Ed. Addison-Wesley Iberoamericana. Segunda Edición, 1986.
- [4] Braess, Dietrich, “Finite Elements”. Cambridge University Press, Tercera Edición, 2007.
- [5] Haberman, Richard, “Ecuaciones en Derivadas Parciales con Series de Fourier y Problemas de Contorno”, Ed. Prentice-Hall, Tercera Edición, 2003.
- [6] Minzoni, Antonmaria, “Apuntes de Ecuaciones en Derivadas Parciales”, Serie FENOMECC-IIMAS, 2003.
- [7] Burden, Richard, “Análisis Numérico”. Ed. Thomson Learning, Séptima Edición, 2002.
- [8] Grossmann, Christian, et. al., “Numerical Treatment of Partial Differential Equations”, Ed. Springer, Univesitext.
- [9] Evance, Lawrence, “Partial Differential Equations”, American Mathematical Society, Graduate Studies in Mathematics Volume 19.
- [10] C. T. Kelley, “Iterative Methods for Linear and Nonlinear Equations”, SIAM, 1995.
- [11] Herrera, Ismael, “Mathematical Modeling in Science and Engineering: An Axiomatic Approach”, Ed. Wiley, 2012.
- [12] Leveque, Randall, “Finite Difference Methods for Ordinary and Partial Differential Equations”, SIAM, 2007.
- [13] Gurtin, Morton E. “An introduction to Continuum Mechanics”, Ed. Academic Press, 1981.
- [14] Carrillo Ledesma, Antonio, “Métodos de Descomposición de Dominio en el Espacio de Vectores Derivados y su Implementación Computacional en Paralelo”. Tesis Doctoral, Instituto de Geofísica, UNAM, 2013.
- [15] Braun, M. “Ecuaciones diferenciales y sus aplicaciones”, Grupo Editorial Iberoamérica, 1990.
- [16] Yoysef Saad, ”Iterative Methods for Sparce Linear Systems”, Second Edition, January 2000.
- [17] A.R. Mitchell and D.F. Griffiths, “The finite difference method in Partial Differential Equation”, John Wiley & Sons, 1980.
- [18] Richard Shewchuk, “An introduction to the Conjugate Gradient Method Without the Agonizing Pain”, School of Computer Science, Carnegie Mellon University, Agosto 1994.
- [19] Díaz Viera, Martín Alberto, “Desarrollo del Método de Colocación Trefftz-Herrera: Aplicaciones a Problemas a Transporte en las Geociencias”. Tesis Doctoral, Instituto de Geofísica, UNAM, 2001.
- [20] Lara Aparicio, Miguel, et. al., “Cálculo”, Compañía Editorial Continental, Segunda Edición, 1977.

- [21] Yau Shu wong, et. al., "Exact Finite Diference Schemes For Solving Helmholtz equation At Any Wave-number", International Journal Of Numerical Analsis and Modeling, Series B, páginas 91-108, 2011.
- [22] Booch, Grady, "Análisis y Diseño Orientado a Objetos", Pearson Educación, 1996.
- [23] Chapman, Barbara, et. al., "Using OpenMP", Scientific and Engineering Computation Series, MIT Press, 2007.
- [24] Groop, "Using MPI. Portable Parallel Programming with the Message Passing Interface", MIT Press, 1999.
- [25] Chen, Zhangxin et al, "Computational Methods for Multiphase Flows in Porous Media", SIAM.