Mallas Adaptativas Para la Solución de Ecuaciones Diferenciales Parciales Hiperbólicas

Yuber Alejandro Galeano Traslaviña

Universidad Industrial de Santander
Facultad de Ciencias
Escuela de Física
Maestría en Matemática Aplicada
Bucaramanga
2018

Mallas Adaptativas Para la Solución de Ecuaciones Diferenciales Parciales Hiperbólicas

Yuber Alejandro Galeano Traslaviña

Tesis para optar por el título de
Magíster en Matemática Aplicada

Director:
Fabio Duvan Lora Clavijo
Ph.D. Física

Codirectora:
Anamaría Navarro Noguera
M.Sc. Física

Universidad Industrial de Santander
Facultad de Ciencias
Escuela de Física
Maestría en Matemática Aplicada
Bucaramanga
2018

RESUMEN

TITULO: Mallas adaptativas para la solución de ecuaciones diferenciales parciales hiperbólicas[1]

AUTOR: Yuber Alejandro Galeano Traslaviña[2]

PALABRAS CLAVE: Mallas refinadas adaptativas, Error numérico, Ecuaciones diferenciales parciales hiperbólicas, Ecuación de onda.

RESUMEN: El refinamiento adaptativo de mallas AMR (por sus siglas en ingles), es una técnica utilizada para resolver problemas numéricos de forma eficiente, de tal manera que adapta la resolución numérica de la solución a las condiciones particulares del problema. En este sentido, se utiliza en la solución problemas descritos por ecuaciones diferenciales parciales hiperbólicas, los cuales, muchas veces requieren una precisión mayor en regiones específicas del dominio. Ejemplos de este tipo de problemas son la propagación de ondas o las ecuaciones de la hidrodinámica. El AMR optimiza los recursos computacionales de manera que se puede encontrar una solución logrando resultados mejores en términos de precisión con un número de operaciones inferior. En este trabajo se plantea la implementación de un algoritmo AMR basado en el trabajo de Berger and Oliger (1984), con el fin resolver sistemas de ecuaciones diferenciales parciales hiperbólicas, considerando a la ecuación de onda en una y dos dimensiones como ejemplo representativo de las mismas. Se realiza una descripción detallada del algoritmo AMR implementado, enfatizando en cada una de las partes que lo componen y en la forma en que estas se acoplan y se realizan las pruebas para corroborar la validez de los resultados numéricos obtenidos, tanto en términos de convergencia del error numérico como de eficiencia. Este desarrollo se plantea como un primer paso para implementar la técnica AMR en los algoritmos presentados en (Navarro et al., 2017) y (Lora-Clavijo et al., 2015a), los cuales son usados en la actualidad dentro del grupo de investigación y tienen la limitación de un enmallado uniforme.

[1]Tesis para optar por el título de Magíster en Matemática Aplicada, Escuela de Física, Facultad de Ciencias, Universidad Industrial de Santander.

[2]Director: Fabio Duvan Lora Clavijo, Ph.D. Codirectora: Anamaría Navarro Noguera M.Sc

ABSTRACT

TITLE: Adaptive Mesh Refinement for the solution of hyperbolic partial differential equations

AUTHOR: Yuber Alejandro Galeano Traslaviña[2]

KEY WORDS: Adaptive mesh refinement, Numerical error, Hiperbolic partial differential equations, Wave equation.

ABSTRACT: Adaptive mesh refinement AMR is a technique used to solve numerical problems efficiently, adapting the resolution grid according to the problem to be solved. In this sense, it is important to solve problems described by hyperbolic partial differential equations, which often require greater precision in specific regions of the domain. Examples of this type of problems are propagation of waves or hydrodynamics equations. AMR optimizes the computational resources, so that a solution can be found achieving better results in terms of accuracy with a lower number of operations. In this work we propose implementation of an algorithm AMR based on the work of Berger and Oliger (1984), in order to solve systems of partial differential hyperbolic equations, considering the wave equation in one and two dimensions as a representative example. A detailed description of the proposed algorithm will be made, emphasizing in each one of the component parts and the way they are coupled and tests are carried out to corroborate the correct performance of the algorithms implemented in terms of numeric error convergence and efficiency. This development is proposed as a first step to implement the AMR technique in the algorithms presented in (Navarro et al., 2017) and (Lora-Clavijo et al., 2015a), which are currently used within the research group and have the limitation of a uniform mesh.

[2]Director: Fabio Duvan Lora Clavijo, Ph.D. Co-director: Anamaría Navarro Noguera M.Sc

Índice general

Índice de figuras

Introducción

Con el avance de la Física, cada día se plantean problemas más complejos, muchos de los cuales no tienen una solución analítica, por lo cual para resolverlos resulta necesario el Análisis numérico. En este contexto un método numérico convierte un problema muy complejo en un gran número de operaciones aritméticas, que al ser resueltas entregan una aproximación de la solución exacta (Nakamura, 1992). En la solución numérica de sistemas de ecuaciones diferenciales parciales se tiene que casi siempre a mayor resolución de una malla, la solución estará más cerca de la solución exacta del problema. En otras palabras, entre más pequeños sean el tamaño de los pasos de la malla, tanto espaciales como temporales, se obtiene una mejor descripción del fenómeno bajo estudio. Sin embargo, para la mayoría de problemas numéricos obtener una solución con un tamaño de paso muy pequeño genera un costo computacional muy grande, y más aún si se trabaja en dos o tres dimensiones, pues se deben realizar muchas más operaciones para completar dicho cálculo. Tomando en cuenta esto, para lograr un equilibrio entre obtener una solución con buena precisión y a su vez hacer que el número de operaciones no sea demasiado elevado, ha surgido la necesidad de usar refinamiento adaptativo de mallas, en donde se tiene una resolución mayor en algunas regiones localizadas de interés mientras que en las demás el tamaño de paso original se mantiene, obteniendo así resultados mejores en términos de precisión con un número de operaciones inferior (Huang and Russell, 2010).

En los últimos años se han desarrollado muchos trabajos para la implementación de este tipo de mallas, todos ellos pensados en aplicaciones específicas, que muchas veces son extensibles a un tipo particular de ecuaciones diferenciales parciales (EDPs). Los primeros planteamientos fueron realizados por Berger and Oliger (1984) y están basados en la idea de multiples mallas para la solución de ecuaciones diferenciales parciales hiperbólicas (EDPHs) usando técnicas de diferencias finitas, donde las mallas se crean o se eliminan siguiendo estimaciones del error numérico, esto con el objetivo de mantener una precisión dada con un mínimo de operaciones. En trabajos como el de Berger and Colella (1989) o el de Bell et al. (1994) se complementó esta idea adaptándola a sistemas conservativos hiperbólicos como las ecuaciones de Euler de la hidrodinámica. Por otro lado, existen en la literatura diferentes códigos numéricos que usan los métodos de mallas adaptativas. Unos de ellos es el código FLASH (Fryxell et al., 2000), el cual está diseñado para estudiar una amplia gama de fenómenos astrofísicos como lo son las emanaciones de rayos gamma (GRBs por sus siglas en inglés), las inestabilidades de Rayleigh-Taylor y Richtmyer-Meshkov y los frentes de llamas termonucleares, entre otros. En 2002 Teyssier (2002) presentó RAMSES, un código diseñado para estudiar la estructura y formación del universo con una alta resolución espacial. Por otro lado, también se desarrollaron códigos numéricos en los cuales son necesarios los algoritmos de mallas refinadas en subdominios fijos, esto con el fin de hacer simulaciones numéricas de colisiones de agujeros

negros y estrellas de neutrones en el régimen de la relatividad general. Entre estos está el código libre CACTUS (Löffler et al., 2012; Schnetter et al., 2004), el cual contiene una estructura modular, que permite interactuar con diferentes arquitecturas y entre diferentes grupos. Este código está diseñado para estudiar procesos astrofísicos en relatividad general, específicamente aquellos relacionados con agujeros negros y estrellas de neutrones[1]. En esta misma dirección, Pretorius and Choptuik (2006) muestran como acoplar diferentes tipos de EDPs, específicamente elípticas e hiperbólicas, realizando un enmallado dinámico. Estos trabajos son de gran interés, y motivan el uso de las mallas adapatativas, ya que están generando plantillas de ondas gravitacionales, las cuales son muy necesarias hoy en día para la observación ondas gravitacionales (Abbott et al., 2016, 2017a,b). Finalmente, cabe resaltar que existen códigos independientes como lo es BAM (Brügmann, 1999; Thierfelder et al., 2011), entre otros.

En desarrollos previos, algunos de los cuales se han llevados a cabo dentro del **Grupo de Investigación en Relatividad y Gravitación (GIRG)**, se han derivado códigos propios como el presentado en el trabajo de Navarro et al. (2017) (MAGNUS), en el cual se muestra un código numérico que resuelve las ecuaciones de la magneto-hidrodinámica, con el fin de estudiar los diferentes tipos de ondas que se propagan en la atmósfera solar, considerando efectos físicos como los son la resistividad eléctrica y la transferencia de calor. Por otra parte, también se cuenta con un código magneto-hidrodinámico relativista (CAFE) (Lora-Clavijo et al., 2015a), el cual está diseñado para el estudio de fluidos astrofísicos y fenómenos de altas energías, como lo son los GRBs y jets extra-galácticos. Además, CAFE también se ha usado para el estudio de procesos de acreción en agujeros negros rotantes Cruz-Osorio and Lora-Clavijo (2016); Cruz-Osorio et al. (2012, 2017); Lora-Clavijo et al. (2015b, 2014); Lora-Clavijo and Guzmán (2013); Lora-Clavijo et al. (2013). Sin embargo, estos códigos tienen la limitación de que el enmallado es uniforme, por lo cual para tener simulaciones con alto grado de precisión se requieren tiempos computacionales muy longevos. Por esta razón, este trabajo de investigación se enmarca en el contexto de dichos desarrollos numéricos, con el objetivo de complementarlos y llevar a cabo una primera aproximación al desarrollo e implementación de algoritmos de este tipo dentro del grupo de investigación, esto con el objetivo de crear herramientas computacionales que contengan mallas adaptativas que harán posible lograr mejores resultados tanto en términos de precisión como en uso de recursos computacionales.

En este proyecto de investigación, se plantea la implementación de un algoritmo basado en el refinamiento adaptativo de mallas tomando como base los trabajos de Berger and Oliger (1984) y Berger and Colella (1989) para resolver EDPs hiperbólicas. Se utilizan diferencias finitas y una estructura cartesiana con sub-dominios refinados, en donde éstos se desplazan en el tiempo siguiendo la dinámica del problema y evolucionan de forma acoplada con la malla base, es decir, se desarrolla una única malla con diferentes resoluciones en diferentes dominios, la cual se adapta a la dinámica del sistemas de ecuaciones a resolver. Todo esto se lleva a cabo tomando en cuenta la validez de los resultados numéricos, mediante pruebas de convergencia y auto-convergencia de la norma del error. Como problema representativo se resuelve la ecuación de onda en una y dos dimensiones con condiciones iniciales dadas y condiciones de contorno de onda saliente.

[1] https://einsteintoolkit.org/index.html

El trabajo está organizado de la siguiente manera. En el capítulo 1 se describe la clasificación de las EDPs, enfatizando en las EDPHs y en especial en la ecuación de onda como ejemplo representativo de estas. Además se descompone la ecuación de onda en su sistema equivalente de EDPHs de primer orden, donde se estudian los modos característicos y las velocidades de propagación, todo esto para una y dos dimensiones. En el capítulo 2 se realiza una breve introducción al método AMR y se describen los algoritmos desarrollados de manera detallada, enfatizando en el funcionamiento de los mismos y en las herramientas usadas en su implementación. En el capítulo 3 se muestran las pruebas para verificar la validez de los resultados numéricos obtenidos en términos de convergencia del error numérico. En el capítulo 4 se presentan los resultados obtenidos en una y dos dimensiones, tanto para la malla uniforme como para el algoritmo AMR, mostrando gráficos de cada una de las simulaciones, del error numérico de las mismas y de las pruebas de convergencia de los algoritmos. Finalmente se presentan las conclusiones obtenidas a partir el desarrollo del trabajo y los anexos necesarios para comprender mejor el trabajo.

Capítulo 1

Ecuaciones diferenciales parciales

Una ecuación diferencial parcial (EDP) es aquella cuyos elementos involucran una función matemática ϕ de varias variables independientes $x, y, z, t, ...$ (por lo menos dos) y las derivadas parciales de ϕ respecto de esas variables (un caso especial son las ecuaciones diferenciales ordinarias, las cuales contienen funciones de una sola variable y sus derivadas). Las ecuaciones diferenciales parciales se emplean en la formulación matemática de procesos de la Física y otras ciencias que suelen estar distribuidos en el espacio y el tiempo; tales como la propagación del sonido o del calor, la electrostática, la electrodinámica, la dinámica de fluidos, la elasticidad, la mecánica cuántica, entre otros.

Las ecuaciones diferenciales parciales EDPs se pueden clasificar en tres tipos principales: parabólicas, elípticas e hiperbólicas, cada una de la cuales tiene características particulares y describen diferentes tipos de fenómenos físicos. Por ejemplo, las EDPs hiperbólicas se utilizan para describir fenómenos como el transporte convectivo de materia y las ondas elásticas, acústicas y electromagnéticas; las EDPs parabólicas describen la difusión de particulas en movimiento y las EDPs elípticas describen la conducción de calor en los sólidos, entre otros. Para distinguir el tipo de las ecuaciones, se considera la siguiente forma general de una ecuación diferencial parcial EDP de segundo orden en dos variables independientes, el orden de la EDP está denotado por la derivada de mayor orden

$$A\partial_{x_1x_1}\phi + B\partial_{x_1x_2}\phi + C\partial_{x_2x_2}\phi + D\partial_{x_1}\phi + E\partial_{x_2}\phi + F\phi + G = 0, \tag{1.1}$$

donde x_1 y x_2 son las variables independientes, las cuales pueden ser tanto espaciales como temporales, $\partial_{x_1x_2}\phi$ es la derivada parcial de ϕ respecto a x_1 y x_2, y A, B, C, D, E, F y G son constantes dadas. Esta EDP se puede clasificar como hiperbólica, parabólica o elíptica dependiendo de los coeficientes que acompañan los términos de esta, de la siguiente forma:

$$\begin{aligned} B^2 - 4AC > 0 \quad & \text{Hiperbólica,} \\ B^2 - 4AC = 0 \quad & \text{Parabólica,} \\ B^2 - 4AC < 0 \quad & \text{Elíptica.} \end{aligned}$$

Ahora, si se tienen n variables independientes la ecuación diferencial parcial de segundo orden en la variable independiente ϕ tiene la forma,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j}\partial_{x_ix_j}\phi + \text{terminos de orden menor} = 0,$$

donde los coeficientes $a_{i,j}$ son números reales. La clasificación de esta ecuación diferencial depende del signo de los valores propios de la matriz $A = (a_{i,j})$. Es hiperbólica si uno de los valores propios es negativo y los demás son positivos o si uno es positivo y los demás son negativos; es parabólica si todos los valores propios son positivos o negativos salvo uno que es cero; es elíptica si todos los valores propios son positivos o todos son negativos (Evans, 2010).

1.1. Ecuación de onda

Uno de los ejemplos más representativos de EDPs hiperbólicas y además importante en el marco de este trabajo es la ecuación de onda, la cual en su forma más elemental n-dimensional hace referencia a una función $\phi(x_1, x_2, ..., x_n, t)$ que satisface:

$$\partial_{tt}\phi - v^2\Delta\phi = 0, \tag{1.2}$$

donde $\Delta = \nabla^2$ es el operador laplaciano, v es una constante equivalente a la velocidad de propagación de la onda y t la coordenada temporal (Evans, 2010). Para un análisis numérico en casos como este, donde las ecuaciones son de orden mayor o igual a dos, resulta conveniente convertirlas a sistemas de EDPs de primer orden.

El problema de condiciones iniciales (CI) y condiciones en la frontera (CF) para la ecuación de onda en **una dimensión** en coordenadas cartesianas $\phi(x,t)$ es,

$$\begin{array}{ll} \text{EDP} & \begin{cases} \partial_{tt}\phi - v^2\partial_{xx}\phi = 0, & x \in \Omega, t > 0 \end{cases} \\ \text{CI} & \begin{cases} \phi(x,0) = \phi_0(x), & x \in \Omega \\ \phi'(x,0) = \phi_0'(x), \end{cases} \\ \text{CF} & \begin{cases} \phi(x,t) = f(x,t), & x \in \partial\Omega \end{cases} \end{array} \tag{1.3}$$

donde Ω es el dominio definido en $\mathbb{R}$, $\partial\Omega$ pertenece a la frontera de Ω, f es una función dada por las condiciones de frontera. Se puede plantear un sistema de EDPs de primer orden equivalente, considerando $u_1(x,t) = \partial_t\phi(x,t)$, $u_2(x,t) = v\partial_x\phi(x,t)$, el cual resulta ser,

$$\begin{array}{ll} \text{EDPs} & \begin{cases} \partial_t u_1 = v\partial_x u_2, & x \in \Omega, t > 0 \\ \partial_t u_2 = v\partial_x u_1, \end{cases} \\ \text{CI} & \begin{cases} u_1(x,0) = u_{10}(x), & x \in \Omega \\ u_2(x,0) = u_{20}(x), \end{cases} \\ \text{CF} & \begin{cases} u_1(x,t) = g(x,t), & x \in \partial\Omega \\ u_2(x,t) = h(x,t), \end{cases} \end{array} \tag{1.4}$$

donde g y f son funciones dadas por las condiciones de frontera.

Para encontrar las **direcciones características** de esta ecuación diferencial, es decir, las direcciones locales de propagación en el plano $x-t$, se debe tener en cuenta la forma matricial del sistema de EDPs,

$$\partial_t \mathbf{U} - \mathbf{A}\partial_x \mathbf{U} = 0, \tag{1.5}$$

con $\mathbf{U} = (u_1, u_2)^T$ y $\mathbf{A} = \begin{pmatrix} 0 & v \\ v & 0 \end{pmatrix}$, primero se calculan los valores propios de la matriz $\mathbf{A}$, resolviendo la ecuación $|\mathbf{A} - \mathbf{I}\lambda| = 0$, donde $\mathbf{I}$ es la matriz identidad de orden 2, entonces,

$$\lambda_\pm = \pm v, \tag{1.6}$$

los cuales están asociados, respectivamente, los vectores propios $\mathbf{v}_+ = (1,1)^T$ y $\mathbf{v}_- = (1,-1)^T$. Entonces la matriz que diagonaliza $\mathbf{A}$ y su inversa son,

$$\mathbf{P} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad \mathbf{P}^{-1} = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{1.7}$$

donde la matriz $\mathbf{A}$ puede representarse en la forma $\mathbf{A} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1}$, con $\mathbf{\Lambda} = \begin{pmatrix} \lambda_+ & 0 \\ 0 & \lambda_- \end{pmatrix}$. Multiplicando (1.5) por $\mathbf{P}^{-1}$, se tiene que

$$\begin{aligned} \mathbf{P}^{-1}\partial_t \mathbf{U} - \mathbf{P}^{-1}\mathbf{A}\partial_x \mathbf{U} &= 0, \\ \partial_t \mathbf{w} - \mathbf{\Lambda}\partial_x \mathbf{w} &= 0, \end{aligned} \tag{1.8}$$

donde

$$\mathbf{w} = \mathbf{P}^{-1}\mathbf{U} = \frac{1}{2}\begin{pmatrix} u_1 + u_2 \\ u_1 - u_2 \end{pmatrix} = \begin{pmatrix} \phi_i \\ \phi_d \end{pmatrix}, \tag{1.9}$$

es el vector cuyos componentes ϕ_d y ϕ_i son las variables características. De este modo, el sistema de ecuaciones (1.5) se desacopla y la dinámica del campo escalar queda descompuesta en un modo

$$\phi_i = \frac{u_1 + u_2}{2}, \tag{1.10}$$

que se mueve a la izquierda y otro modo

$$\phi_d = \frac{u_1 - u_2}{2}, \tag{1.11}$$

que se mueve a la derecha. En la figura 1.1 se puede observar gráficamente el uso de estas expresiones para realizar la descomposición de una función que es solución exacta de la ecuación de onda en una dimensión,

$$\phi(x,t) = 1/2\left(e^{-(x-x_0+t)^2/\sigma^2} + e^{-(x-x_0-t)^2/\sigma^2}\right),$$

en sus modos fundamentales, donde se puede observar la onda y sus modos fundamentales en un instante de tiempo, en este caso se tiene $x_0 = 0$, $\sigma = 0.1$, x definido en $[-1,1]$ y $t = 0.5$. Estos modos son útiles al desarrollar un algoritmo numérico, dado que sirven para imponer

las condiciones en las fronteras, ya que estas son las ondas que realmente viajan en tales direcciones (Guzmán, 2010).

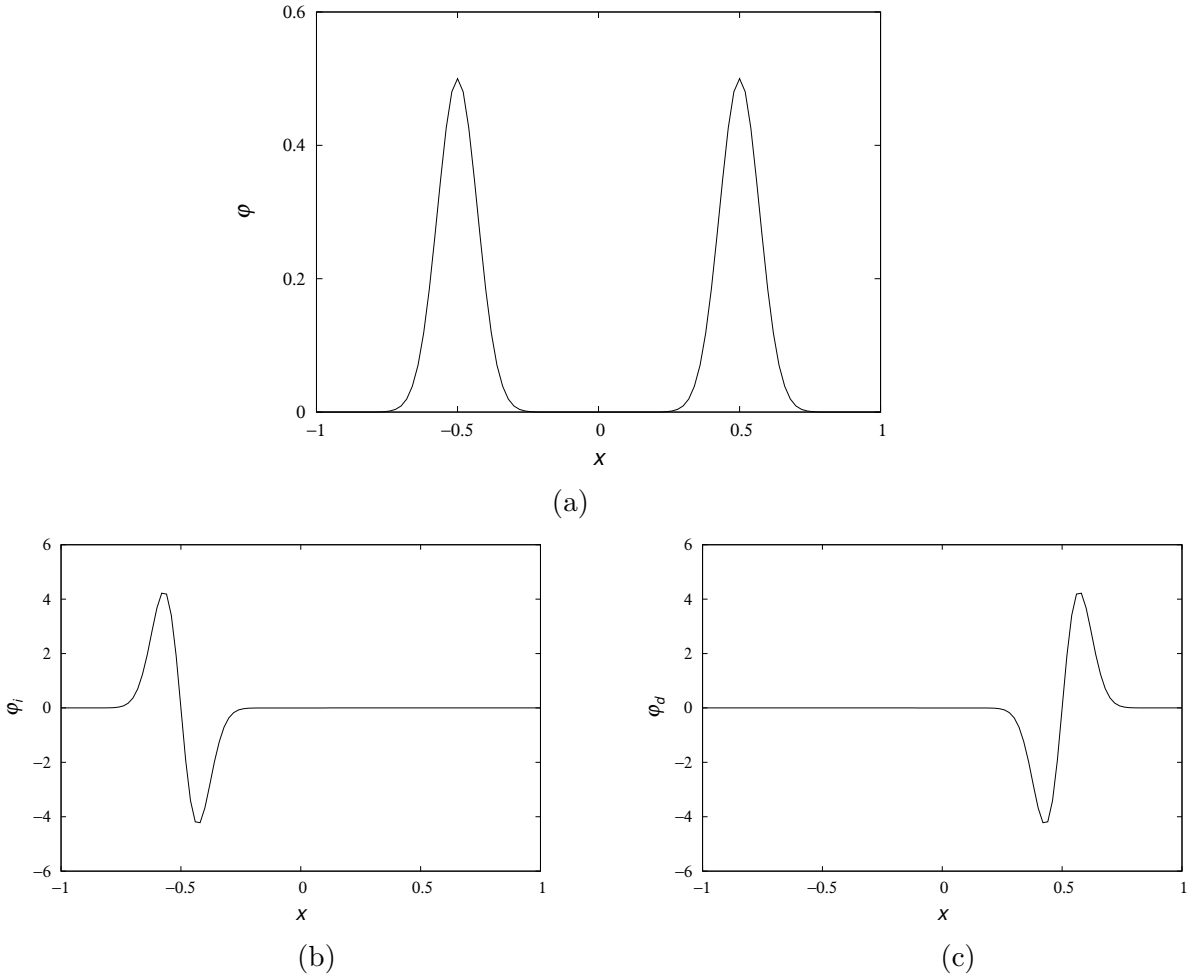


Figura 1.1. Modos fundamentales de una solución exacta de la ecuación de onda en una dimensión. En (a) se muestra la onda en un instante de tiempo y en (b) y (c) se muestran los modos ϕ_i y ϕ_d que se propagan a izquierda y derecha, respectivamente, en el mismo instante de tiempo, acorde con (1.11) y (1.10).

El problema de condiciones iniciales (CI) y condiciones en la frontera (CF) para la ecuación de onda en **dos dimensión** en coordenadas cartesianas $\phi(x, y, t)$ es,

$$\begin{array}{ll} \text{EDP} & \left\{ \begin{array}{ll} \partial_{tt}\phi - v^2(\partial_{xx}\phi + \partial_{yy}\phi) = 0, & x, y \in \Omega; t > 0 \\ \\ \phi(x, y, 0) = \phi_0(x, y), & x, y \in \Omega \\ \phi'(x, y, 0) = \phi_0'(x, y), & \\ \\ \phi(x, y, t) = f(x, y, t), & x \in \partial\Omega \end{array} \right. \\ \text{CI} & \\ \text{CF} & \end{array} \tag{1.12}$$

donde Ω es el dominio definido en $\mathbb{R}^2$, $\partial\Omega$ pertenece a la frontera de Ω, f es una función dada por las condiciones de frontera. Se plantea un sistema de EDPs de primer orden equivalente, considerando $u_1 = \partial_t\phi$, $u_2 = v\partial_x\phi$ y $u_3 = v\partial_y\phi$, el cual resulta ser,

$$\begin{array}{ll} \text{EDPs} & \begin{cases} \partial_t u_1 = v\partial_x u_2, & x, y \in \Omega, t > 0 \\ \partial_t u_2 = v\partial_x u_1, & \\ \partial_t u_3 = v\partial_y u_1, & \end{cases} \\ \text{CI} & \begin{cases} u_1(x, y, 0) = u_{10}(x, y), & x, y \in \Omega \\ u_2(x, y, 0) = u_{20}(x, y), & \\ u_3(x, y, 0) = u_{30}(x, y), & \end{cases} \\ \text{CF} & \begin{cases} u_1(x, y, t) = g(x, y, t), & x, y \in \partial\Omega \\ u_2(x, y, t) = h(x, y, t), & \\ u_3(x, y, t) = l(x, y, t), & \end{cases} \end{array} \tag{1.13}$$

donde g, h y l son funciones dadas por las condiciones de frontera.

Las direcciones características de esta ecuación diferencial se pueden desacoplar de forma que la dinámica del campo escalar quede descompuesta en 4 modos fundamentales, los cuales se desplazan a izquierda, derecha, arriba y abajo. Para el análisis de los modos izquierda y derecha se supone $\partial_y\phi = 0$ de tal forma que:

$$\partial_t\mathbf{U} - \mathbf{A}\partial_x\mathbf{U} = 0, \tag{1.14}$$

en donde $\mathbf{U} = (u_1, u_2)^T$ y $\mathbf{A} = \begin{pmatrix} 0 & v \\ v & 0 \end{pmatrix}$. Al realizar la diagonalización de A se tiene:

$$\partial_t\mathbf{w} - \mathbf{\Lambda}\partial_x\mathbf{w} = 0, \tag{1.15}$$

con $\mathbf{\Lambda} = \begin{pmatrix} v & 0 \\ 0 & -v \end{pmatrix}$ y

$$\mathbf{w} = \frac{1}{2}\begin{pmatrix} u_1 + u_2 \\ u_1 - u_2 \end{pmatrix} = \begin{pmatrix} \phi_i \\ \phi_d \end{pmatrix}, \tag{1.16}$$

donde, el modo que se mueve a la izquierda es

$$\phi_i = \frac{u_1 + u_2}{2}. \tag{1.17}$$

y el que se mueve a la derecha es

$$\phi_d = \frac{u_1 - u_2}{2}, \tag{1.18}$$

Para el análisis de los modos arriba y abajo se supone $\partial_x\phi = 0$, de tal forma que:

$$\partial_t\mathbf{U} - \mathbf{A}\partial_y\mathbf{U} = 0, \tag{1.19}$$

en donde $\mathbf{U} = (u_1, u_3)^T$ y $\mathbf{A} = \begin{pmatrix} 0 & v \\ v & 0 \end{pmatrix}$. Al realizar al diagonalización de $\mathbf{A}$ se tiene:

$$\partial_t \mathbf{w} - \mathbf{\Lambda} \partial_y \mathbf{w} = 0, \tag{1.20}$$

con $\mathbf{\Lambda} = \begin{pmatrix} v & 0 \\ 0 & -v \end{pmatrix}$ y

$$\mathbf{w} = \frac{1}{2} \begin{pmatrix} u_1 + u_3 \\ u_1 - u_3 \end{pmatrix} = \begin{pmatrix} \phi_{ab} \\ \phi_{ar} \end{pmatrix}, \tag{1.21}$$

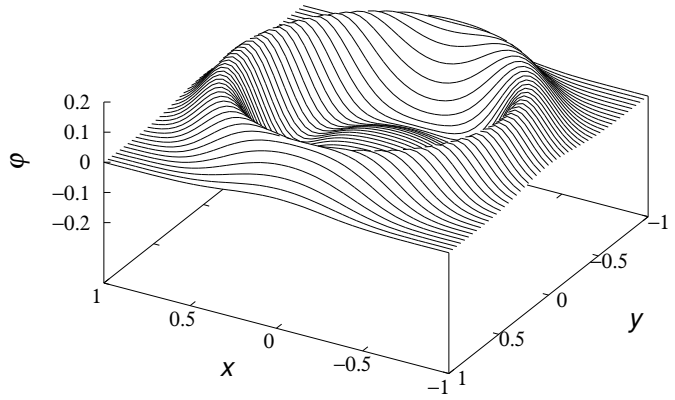
donde, el modo que se mueve hacia abajo es

$$\phi_{ab} = \frac{u_1 + u_3}{2}. \tag{1.22}$$

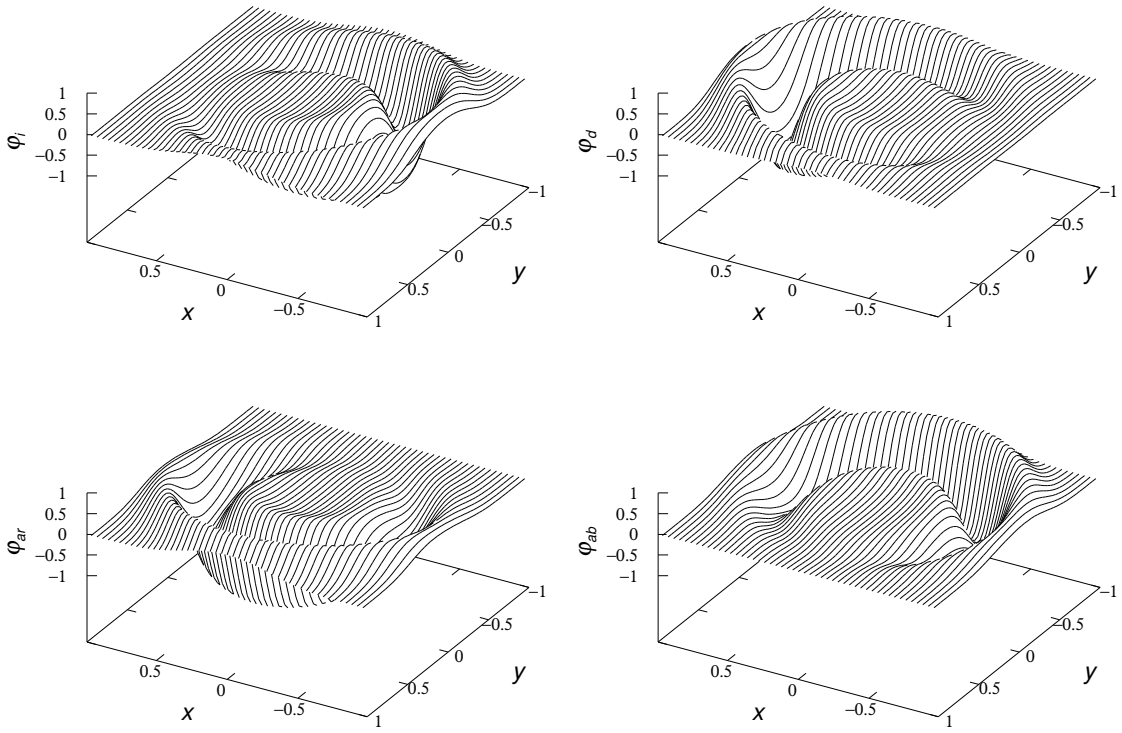
y el que se mueve hacia arriba es

$$\phi_{ar} = \frac{u_1 - u_3}{2} \tag{1.23}$$

En la figura 1.2 se puede observar gráficamente el uso de estas ecuaciones para realizar la descomposición en los modos fundamentales de una solución numérica de la ecuación de onda en dos dimensiones. De nuevo, se puede decir que estos modos son muy útiles para imponer condiciones de frontera en un algoritmo numérico. Para el caso en 3 dimensiones el procedimiento es el mismo pero el sistema resultante está compuesto por 4 EDPs y 6 modos fundamentales.



(a)



(b)

Figura 1.2. Modos fundamentales de la solución numérica de la ecuación de onda en dos dimensiones. En (a) se muestra la onda en un instante de tiempo y en (b) los modos ϕ_i, ϕ_d, ϕ_{ar} y ϕ_{ab} que se propagan a izquierda, derecha, arriba y abajo respectivamente en el mismo instante de tiempo, acorde con (1.18), (1.17), (1.23) y (1.22), una descripción más detallada de esta evolución numérica se encuentra en el capitulo 4.

Capítulo 2

Mallas refinadas adaptativas

Al momento de resolver numéricamente sistemas de EDPs, como los descritos anteriormente, surgen dinámicas que requieren una resolución alta en regiones específicas, lo cual genera la necesidad de crear mallas refinadas (ya sean fijas o adaptativas) en estas. La idea de este método consiste en resolver un problema utilizando diferentes dominios, cada uno con una resolución numérica particular, en donde hay dominios que están anidados dentro de otros, pero todos evolucionan de forma acoplada, es decir, como si se tratara de una sola malla. Este método no solo refina regiones específicas, sino que a medida que el problema evoluciona adapta estas a las nuevas dinámicas, desplazándolas, y modificándolas en forma y en tamaño siguiendo criterios matemáticos o físicos. Además, este método es recursivo en el sentido de que los dominios refinados pueden contener a su vez subdominios aún más refinados. La principal ventaja que tiene este método es que logra resultados numéricos mejores requiriendo un número menor de operaciones. En esta sección se plantea y se describe brevemente un algoritmo de refinamiento adaptativo de mallas basado en el trabajo de Berger and Oliger (1984) como base para su posterior implementación.

2.1. Descripción

El algoritmo comienza con todo el dominio espacial del problema cubierto con una malla base, la cual se encuentra en el nivel l_0. Entonces, a medida que avanza el cálculo se etiquetan las celdas seleccionadas para el refinamiento, las cuales se van a encontrar en el nivel l_1, y son subdominios de la malla base, esta selección se basa en identificar la región que presenta el error máximo. El nivel l_1 de la misma forma puede contener regiones refinadas en niveles más altos como l_2 o l_3, esto dado que el algoritmo es recurrente, es decir, ejecuta la misma acción en diferentes niveles, lo cual se ilustra mejor en las figuras 2.1 y 2.2. En el desarrollo de este trabajo se etiqueta como malla base a la situada en el nivel l_0, como malla hijo a la situada en l_1, como malla nieto en l_2 y así sucesivamente. El refinamiento obedece a las necesidades y a la naturaleza del problema, en el sentido en que este presenta valores del error numérico diferentes en distintas regiones. Todas las mallas tienen una estructura cartesiana espaciada, tanto en el espacio como en el tiempo, esto con el objetivo de hacer más simple el algoritmo.

Los tamaños de paso espaciales y temporales están relacionados por una constante $\mathbf{C}$[1] de la forma $\Delta t = \mathbf{C}\Delta x/v$, en este trabajo se usa $v = 1$, entonces $\Delta t = \mathbf{C}\Delta x$. Cada uno de los niveles se discretiza con tamaños específicos de paso, los cuales están relacionados por una constante α, en decir, se usan Δx y Δt para el nivel l_0, se usan $\Delta x/\alpha$ y $\Delta t/\alpha$ para el nivel l_1, se usan $\Delta x/\alpha^2$ y $\Delta t/\alpha^2$ para el nivel l_2, y así sucesivamente para niveles superiores. La constante α es un número entero que selecciona teniendo en cuenta que tanto se desea refinar la malla, usualmente tiene valores bajos, también se debe decir que la discretización se asume igual en todos los ejes, es decir, $\Delta x = \Delta y = \Delta z$.

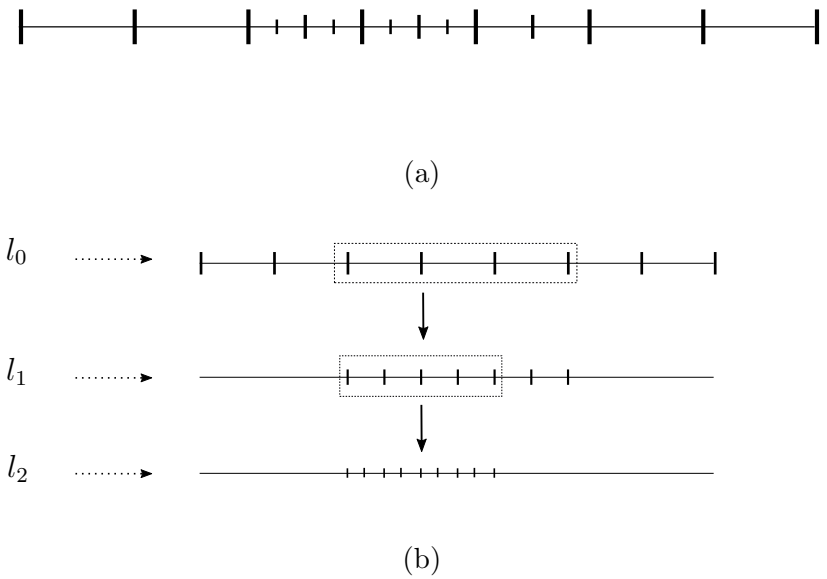
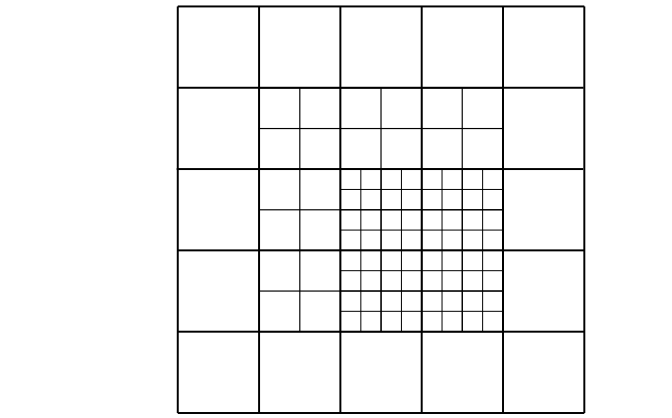
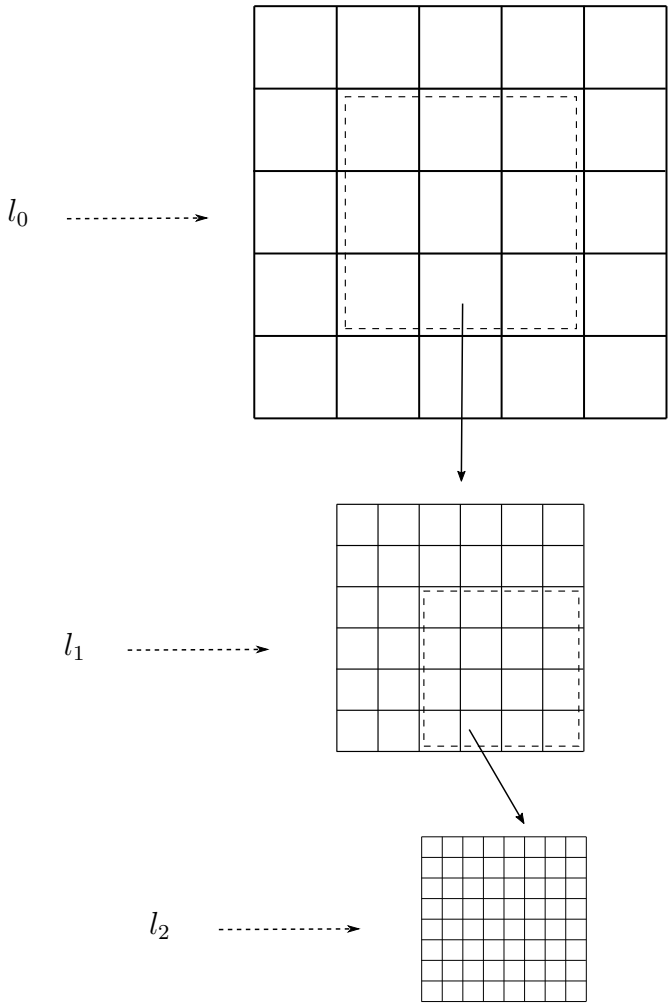


Figura 2.1. Refinamiento en 3 niveles l_0, l_1 y l_2 por un factor $\alpha = 2$ para el caso en una dimensión, en (a) se muestra la discretización completa y en (b) la misma discretización descompuesta por niveles.

[1] $\mathbf{C} = v\Delta t/\Delta x$, se le conoce como factor de Courant Friedrichs Lewy debido a que dichos autores lo definieron para la ecuación de advección (Courant et al., 1967)



(a)



(b)

Figura 2.2. Refinamiento en 3 niveles l_0, l_1 y l_2 por un factor $\alpha = 2$ para el caso en dos dimensiones. En (a) se muestra la discretización completa y en (b) la misma discretización descompuesta por niveles. (Fuente: Debreu et al. (2008)).

Existen dos consideraciones importantes al momento de definir una **evolución acoplada** de las mallas, es decir, que la evolución no se realice de forma totalmente independiente para cada malla, sino que por el contrario corresponda a una única evolución con diferentes resoluciones en diferentes dominios. La primera es que los valores de la malla padre son reemplazados por los de la malla hijo para cada paso de tiempo de la malla base, esto garantiza que la malla padre no evolucione al siguiente paso con su error numérico, que en este caso sería mayor. La segunda son las fronteras de la malla hijo, en este caso la malla hijo evoluciona de forma independiente a excepción de los puntos en las fronteras, los cuales son calculados utilizando interpolación de la malla padre en cada paso de tiempo, usualmente estos puntos son llamados, puntos fantasma (por que se agregan en los extremos de la malla refinada como puntos auxiliares, pero no pertenecen a esta). El número de puntos interpolados en cada extremo depende del orden del algoritmo de integración utilizado, en el caso del Runge Kutta 4 son necesarios 4 puntos. Estas consideraciones se ilustran en las figuras 2.3 y 2.4 para los casos en una y dos dimensiones, en donde $\Delta x = \Delta y$, el paso de tiempo de la malla base es $\Delta t = \mathbf{C} * \Delta x$ y el de la malla refinada $\Delta t_c = \mathbf{C} * \Delta x_c = \mathbf{C} * (1/\alpha) * \Delta x = \mathbf{C} * (1/2) * \Delta x = (1/2)\Delta t$.

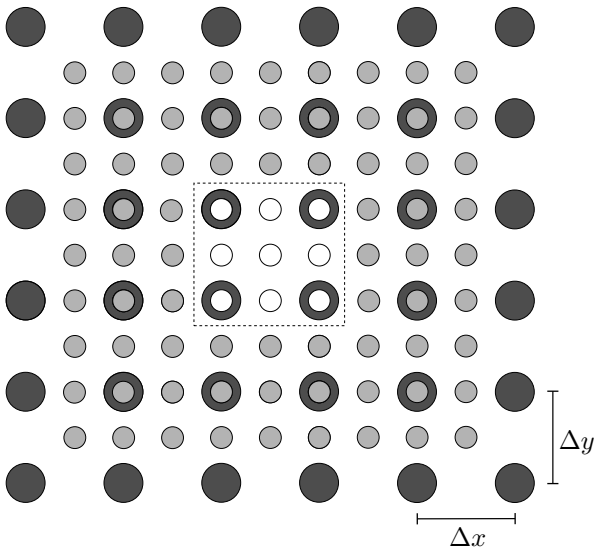
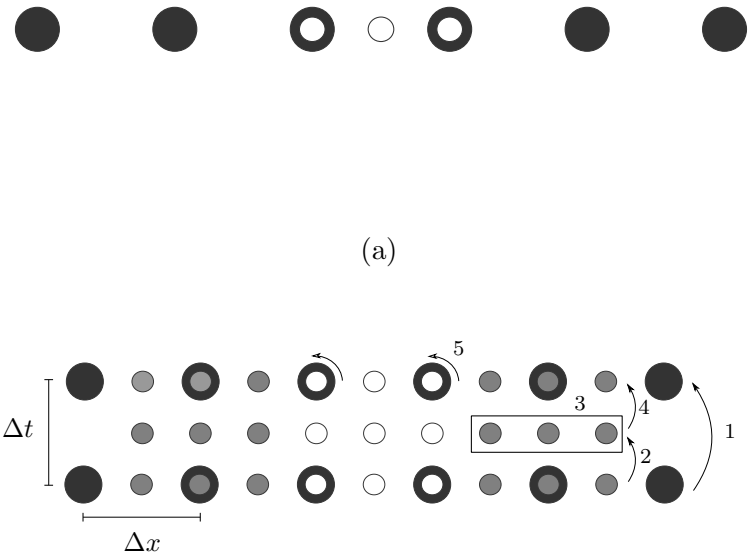


Figura 2.4. Malla en dos dimensiones con un subdominio refinado por $\alpha = 2$, en un instante de tiempo. Los puntos negros corresponden a la malla base, los blancos a la malla refinada y los grises son puntos de la malla refinada calculados utilizando interpolación de la malla base (puntos fantasma). La evolución temporal es igual al caso en una dimensión.

Finalmente, en el algoritmo existe recurrecia, es decir, el procedimiento para la evolución entre dos dominios anidados consecutivos, ya sea el dominio base y el hijo o el dominio hijo y el nieto, siempre es el mismo indefinidamente. Por esta razón no es necesario explicar el caso de múltiples dominios anidados, dado que es simplemente el mismo procedimiento ejecutado de forma recurrente.



(a)

(b)

Figura 2.3. Evolución de una malla en una dimensión con un subdominio refinado por $\alpha = 2$. En (a) se muestra la discretización inicial de los dominios. En (b) está la evolución de un paso de tiempo de la malla base, los puntos negros corresponden a la malla base, los blancos a la malla refinada y los grises son puntos de la malla refinada calculados utilizando interpolación de la malla base (puntos fantasma), cuando el punto blanco está dentro del punto negro quiere decir que el valor de la malla refinada en ese punto remplaza o es el mismo al de la malla base. Los pasos que se llevan a cabo son: **1**) evolución en el tiempo de la malla base, **2**) evolución en el tiempo de la malla hijo, **3**) interpolación de los puntos en las fronteras, **4**) evolución en el tiempo de la malla hijo, **5**) remplazo de la malla hijo en la malla padre.

2.2. Herramientas utilizadas

Al momento de resolver numéricamente un problema mediante mallas adaptativas, se hace necesario utilizar una serie de herramientas en cada una de las etapas del algoritmo. Estas herramientas van desde las utilizadas para la discretización, que en este caso son las diferencias finitas, las usadas para la evolución temporal, las cuales son el método de líneas y los métodos de integración, las utilizadas para aproximar el valor de la función en ciertos puntos a partir de valores en otros puntos, como son interpolaciones lineales multidimensionales e interpolación de lagrange. A continuación se presenta una breve descripción de cada una de estas herramientas enfatizando en las utilizadas específicamente en la implementación del algoritmo desarrollado.

2.2.1 Discretización por diferencias finitas. La solución aproximada de una EDP mediante el método de diferencias finitas (MDF), consiste en definir una versión discreta de la EDP, y además, en calcular una solución a dicha aproximación sobre un dominio. Discretizar

una EDP consiste básicamente en aproximar los diferentes operadores diferenciales en cada punto de la malla numérica, dichos operadores diferenciales son aproximados mediante el uso de la representación de Taylor de orden n de ϕ en x_i

$$\phi(x_i + \Delta x) = \phi(x_i) + \frac{\phi^{'}(x_i)}{1!}\Delta x + \frac{\phi^{''}(x_i)}{2!}\Delta x^2 + \cdots + \frac{\phi^{(n)}(x_i)}{n!}\Delta x^n + \mathcal{O}(\Delta x^{n+1}), \tag{2.1}$$

donde $\phi^{(n)}(x_i)$ representa la derivada de orden n respecto a x evaluada en x_i. En este caso la función $\phi : \mathbb{R} \to \mathbb{R}$ debe ser $n+1$ veces derivable donde $\mathcal{O}(\Delta x^{n+1}) \approx (\phi^{(n+1)}(x_i)/(n+1)!)\Delta x^{n+1} = g_n(\Delta x)\Delta x^n$ es llamado residuo o resto, denotando la diferencia entre el polinomio de Taylor de orden n y la función original; además se garantiza que existe una función $g_n : \mathbb{R} \to \mathbb{R}$ tal que $\lim_{\Delta x \to 0}(g_n) = 0$.

Para deducir la expresión de la primera derivada se denota $x_{i+1} = x_i + \Delta x$ y $x_{i-1} = x_i - \Delta x$, tal que

$$\begin{aligned} \phi(x_{i-1}) &= \phi(x_i) - \Delta x\phi^{'}(x_i) + \frac{\Delta x^2}{2}\phi^{''}(x_i) + \mathcal{O}(\Delta x^3), \\ \phi(x_{i+1}) &= \phi(x_i) + \Delta x\phi^{'}(x_i) + \frac{\Delta x^2}{2}\phi^{''}(x_i) + \mathcal{O}(\Delta x^3), \end{aligned} \tag{2.2}$$

donde al restar la primera a la segunda expresión y dividiendo por $2\Delta x$, se obtiene la expresión para la primera derivada en el punto x_i con un error de segundo orden

$$\phi^{'}(x_i) = \frac{\phi(x_{i+1}) - \phi(x_{i-1})}{2\Delta x} + \mathcal{O}(\Delta x^2). \tag{2.3}$$

A esta aproximación se le llama diferencia centrada, dado que es necesario conocer los valores de $\phi(x_{i-1})$ y $\phi(x_{i+1})$ para calcular la derivada en x_i. En este sentido es posible calcular aproximaciones no centradas o desbalanceadas, en donde se calcula el valor de la derivada en x_i con los valores de $\phi(x_{i+1})$ y $\phi(x_{i+2})$ o con los valores de $\phi(x_{i-1})$ y $\phi(x_{i-2})$, cambiando los coeficientes que acompañan estos términos como se muestra en la tabla 2.1.

Para calcular operadores con un orden de aproximación mayor basta con generalizar el procedimiento ilustrado anteriormente. Si se quiere calcular la primera derivada con un error de cuarto orden se considera el siguiente conjunto de series truncadas:

$$\phi(x_{i-2}) = \phi(x_i) - 2\Delta x\phi^{'}(x_i) + \frac{4\Delta x^2}{2}\phi^{''}(x_i) - \frac{8\Delta x^3}{6}\phi^{'''}(x_i) + \frac{16\Delta x^2}{24}\phi^{''''}(x_i) + \mathcal{O}(\Delta x^5),$$

$$\phi(x_{i-1}) = \phi(x_i) - \Delta x\phi^{'}(x_i) + \frac{\Delta x^2}{2}\phi^{''}(x_i) - \frac{\Delta x^3}{6}\phi^{'''}(x_i) + \frac{\Delta x^2}{24}\phi^{''''}(x_i) + \mathcal{O}(\Delta x^5),$$

$$\phi(x_{i+1}) = \phi(x_i) + \Delta x\phi^{'}(x_i) + \frac{\Delta x^2}{2}\phi^{''}(x_i) + \frac{\Delta x^3}{6}\phi^{'''}(x_i) + \frac{\Delta x^2}{24}\phi^{''''}(x_i) + \mathcal{O}(\Delta x^5),$$

$$\phi(x_{i+2}) = \phi(x_i) + 2\Delta x\phi^{'}(x_i) + \frac{4\Delta x^2}{2}\phi^{''}(x_i) + \frac{8\Delta x^3}{6}\phi^{'''}(x_i) + \frac{16\Delta x^2}{24}\phi^{''''}(x_i) + \mathcal{O}(\Delta x^5),$$

donde para calcular la primera derivada de ϕ en x_i es necesario encontrar la combinación correcta del tipo $a\phi(x_{i-2}) + b\phi(x_{i-1}) + c\phi(x_i) + d\phi(x_{i+1}) + e\phi(x_{i+2})$, tal que los coeficientes de $\phi(x_i)$ y las segundas y terceras derivadas sean cero. En el caso centrado la expresión para la aproximación de la primera derivada a cuarto orden toma la forma

$$\phi^{'}(x_i) = \frac{1}{12\Delta x}[\phi(x_{i-2}) - 8\phi(x_{i-1}) + 8\phi(x_{i+1}) - \phi(x_{i+2})] + \mathcal{O}(\Delta x^4). \tag{2.4}$$

Como en el caso anterior, para combinaciones desbalanceadas (expresiones para la derivada en donde se tienen más términos de un lado que de otro) se procede usando las expansiones truncadas para distintos puntos vecinos de x_i; los resultados de varias elecciones de puntos vecinos se resumen en la tabla 2.1. Para aproximaciones de la misma derivada con orden de aproximación mayor o derivadas de orden superior, se pueden calcular utilizando la misma dinámica, ver (Olver, 2016; Toro, 2013).

	x_{i-4}	x_{i-3}	x_{i-2}	x_{i-1}	x_i	x_{i+1}	x_{i+2}	x_{i+3}	x_{i+4}
			1	-4	3				
Segundo orden				-1	0	1			
					-3	4	-1		
	3	-16	36	-48	25				
		-1	6	-18	10	3			
Cuarto orden			1	-8	0	8	-1		
				-3	-10	18	-6	1	
					-25	48	-36	16	-3

Tabla 2.1: Coeficientes de la aproximación de la primera derivada usando distintas combinaciones de puntos vecinos. Las aproximaciones con segundo orden de precisión significan $\phi^{'} = 1/(2\Delta x)[...] + \mathcal{O}(\Delta x^2)$ y para cuarto orden $\phi^{'} = 1/(12\Delta x)[...] + \mathcal{O}(\Delta x^4)$. Las ecuaciones de diferencias (2.3) y (2.4) pueden servir como referencia para entender el uso de la tabla. Fuente: Guzmán (2010).

2.2.2 Método de líneas. La integración en el tiempo de los sistemas de EDPHs de primer orden se lleva a cabo mediante el uso de un método en el cual se separan los operadores diferenciales espaciales de los temporales, llamado el método de líneas (Kreiss and Scherer, 1992), el cual consiste en discretizar únicamente la parte espacial del sistema de ecuaciones diferenciales parciales,

$$\frac{d\phi}{dt} = \zeta(\phi), \tag{2.5}$$

donde ζ es el operador espacial, de tal forma que el sistema queda redefinido como un conjunto de ecuaciones diferenciales ordinarias de primer orden de la forma

$$\frac{d\phi(x_i, t_i)}{dt} = v\frac{\phi(x_{i+1}, t_i) - \phi(x_{i-1}, t_i)}{2\Delta x} + \mathcal{O}(\Delta x^2), \tag{2.6}$$

el cual posteriormente se evoluciona en el tiempo utilizando un integrador estándar de ecuaciones diferenciales ordinarias; en este trabajo se utiliza el método Runge Kutta de cuarto orden (ver apéndice A). Cabe aclarar que el método resultante es estable.

2.2.3 Interpolación. La interpolación es muy usada en el desarrollo de este código, dado que en muchos casos hay que calcular el valor de las variables en ciertos puntos en función de la información de estas mismas variables en otros puntos. En este sentido, se usa la interpolación lineal multidimensional, es decir interpolación bilineal, trilineal o n-lineal y la interpolación polinómica de Lagrange. Cuando se evoluciona la malla refinada, las fronteras de ésta deben extraer información de la malla base mediante interpolación, como se mostró en las figuras 2.3 y 2.4; en este caso se calcula el valor de algunos puntos de la malla refinada a partir de la malla base. Para la malla en una dimensión se utiliza interpolación bilineal y para la malla en dos dimensiones interpolación trilineal. Por otra parte, los puntos en las fronteras de la malla base se calculan por medio de interpolación polinómica de Lagrange discretizada a partir de sus puntos vecinos.

La **interpolación bilineal** es una extensión de la interpolación lineal tradicional a dos dimensiones. Así que suponiendo que se tiene una función aproximadamente lineal $\phi(x,t)$ y el valor de esta en 4 puntos que definen un rectángulo como el mostrado en la figura (2.5)

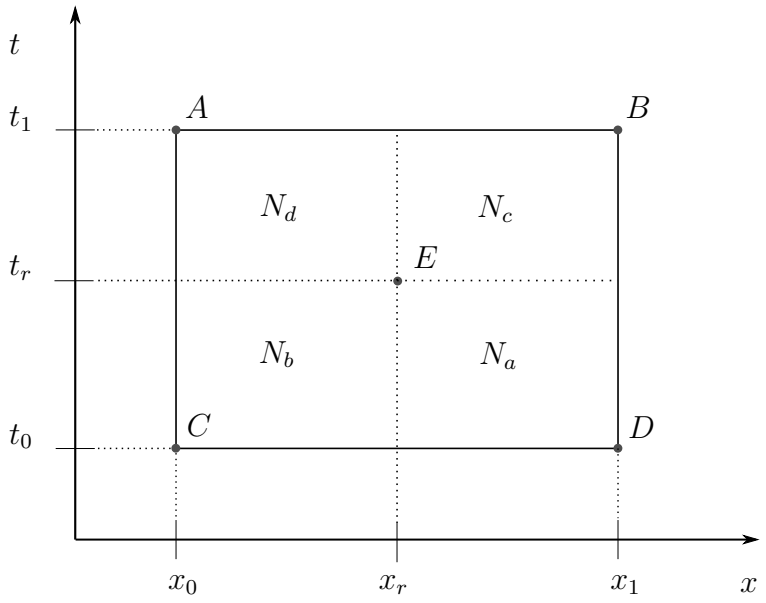


Figura 2.5. Interpolación bilineal. En los puntos A, B, C y D con coordenadas (x_0,t_1), (x_1,t_1), (x_0,t_0) y (x_1,t_0) se tienen los valores de la función $\phi(x,t)$, ϕ_0, ϕ_1, ϕ_2 y ϕ_3 y se pretende hallar un valor aproximado de esta en el punto interior E (x_r,t_r).

se divide el rectángulo $ABCD$ en cuatro partes mediante las líneas $x = x_r$ y $t = t_r$. Para interpolar el valor ϕ_r, donde $\phi(x_r,t_r) \approx \phi_r$, primero se encuentran las áreas normalizadas de las particiones, las cuales se normalizan al dividirlas por el área del rectángulo $ABCD$. Las cuatro áreas normalizadas N_a, N_b, N_c y N_d cada una opuesta al vértice que lleva la misma

letra están dadas por,

$$N_a = \frac{(x_1 - x_r)(t_r - t_0)}{(x_1 - x_0)(t_1 - t_0)}, \qquad N_b = \frac{(x_r - x_0)(t_r - t_0)}{(x_1 - x_0)(t_1 - t_0)},$$
$$N_c = \frac{(x_1 - x_r)(t_1 - t_r)}{(x_1 - x_0)(t_1 - t_0)}, \qquad N_d = \frac{(x_r - x_0)(t_1 - t_r)}{(x_1 - x_0)(t_1 - t_0)}, \tag{2.7}$$

donde ϕ_r se calcula por la ecuación:

$$\phi_r = \phi_0 N_a + \phi_1 N_b + \phi_2 N_c + \phi_3 N_d. \tag{2.8}$$

Por otra parte, la **interpolación trilineal** es una extensión de la interpolación lineal tradicional a tres dimensiones. Suponiendo que se tiene la función aproximadamente lineal ecuación $\phi(x, y, t)$ y se tienen ocho puntos que definen un paralelepípedo rectangular como el mostrado en la figura 2.6,

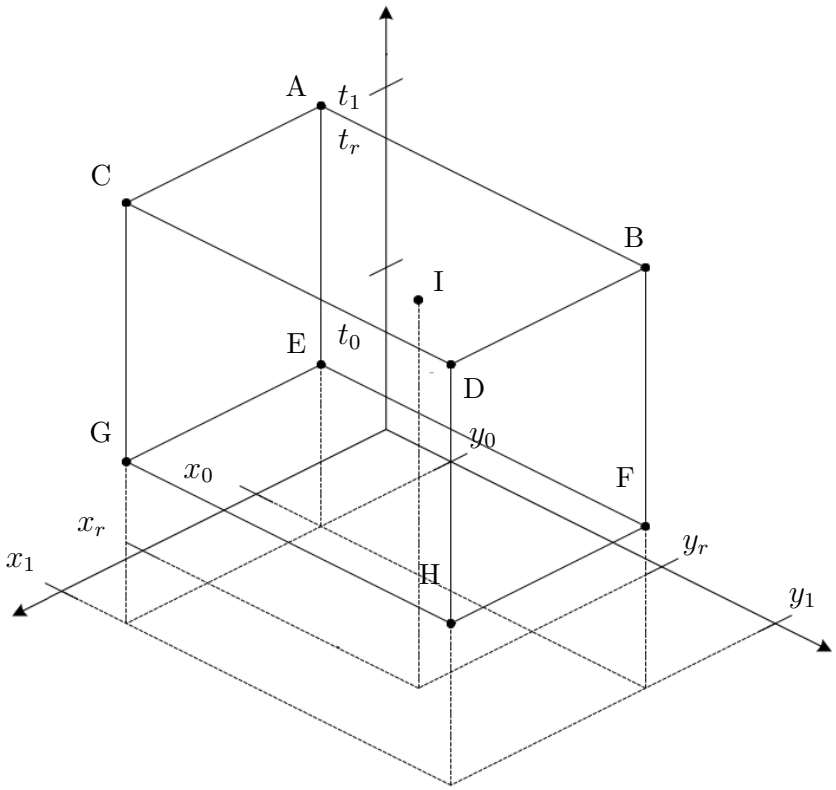


Figura 2.6. Interpolación Trilineal. Se tienen los valores de la función $V(x, y, t)$ en los puntos A, B, C, D, E, F, G y H y se busca hallar un valor aproximado de esta en el punto interior I.

se divide el paralelepípedo en 8 volúmenes por los planos $x = x_r$, $y = y_r$ y $t = t_r$. Para interpolar el valor V_r (es decir $V(x_r, y_r, t_r)$), primero se encuentran los volúmenes normalizados de las particiones, estos se normalizan al dividirlos por el volumen total del prisma. Los ocho volúmenes normalizados N_a, N_b, N_c, N_d, N_e, N_f, N_g y N_h, cada uno diagonalmente opuesto al vértice que lleva la misma letra, están dados por,

$$
\begin{aligned}
N_a &= \frac{(x_1 - x_r)(y_1 - y_r)(t_r - t_0)}{(x_1 - x_0)(y_1 - y_0)(t_1 - t_0)}, \qquad N_b = \frac{(x_1 - x_r)(y_r - y_0)(t_r - t_0)}{(x_1 - x_0)(y_1 - y_0)(t_1 - t_0)}, \\
N_c &= \frac{(x_r - x_0)(y_1 - y_r)(t_r - t_0)}{(x_1 - x_0)(y_1 - y_0)(t_1 - t_0)}, \qquad N_d = \frac{(x_r - x_0)(y_r - y_0)(t_r - t_0)}{(x_1 - x_0)(y_1 - y_0)(t_1 - t_0)}, \\
N_e &= \frac{(x_1 - x_r)(y_1 - y_r)(t_1 - t_r)}{(x_1 - x_0)(y_1 - y_0)(t_1 - t_0)}, \qquad N_f = \frac{(x_1 - x_r)(y_r - y_0)(t_1 - t_r)}{(x_1 - x_0)(y_1 - y_0)(t_1 - t_0)}, \\
N_g &= \frac{(x_r - x_0)(y_1 - y_r)(t_1 - t_r)}{(x_1 - x_0)(y_1 - y_0)(t_1 - t_0)}, \qquad N_h = \frac{(x_r - x_0)(y_r - y_0)(t_1 - t_r)}{(x_1 - x_0)(y_1 - y_0)(t_1 - t_0)},
\end{aligned}
\tag{2.9}
$$

de tal forma que ϕ_r se calcula a partir de la siguiente, expresión

$$
\phi_r = \phi_0 N_a + \phi_1 N_b + \phi_2 N_c + \phi_3 N_d + \phi_4 N_e + \phi_5 N_f + \phi_6 N_g + \phi_7 N_h. \tag{2.10}
$$

Finalmente, para la **interpolación de Lagrange discretizada** para $n+1$ puntos $(x_0, \phi_0) \ldots, (x_n, \phi_n)$, en donde los x_j se asumen distintos, se tiene el polinomio interpolador en la forma de Lagrange, el cual es la combinación lineal,

$$
L(x) = \sum_{j=0}^{n} \phi_j l_j(x), \tag{2.11}
$$

de bases polinómicas de Lagrange,

$$
l_j(x) = \prod_{i=0; i \neq j}^{n} \frac{x - x_i}{x_j - x_i} = \frac{x - x_0}{x_j - x_0} \cdots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdots \frac{x - x_n}{x_j - x_n}, \tag{2.12}
$$

donde, como un caso particular, se toman 3 puntos equidistantes en x, $(\Delta x, \phi_1)$, $(2\Delta x, \phi_2)$ y $(3\Delta x, \phi_3)$, para calcular el valor ϕ_0 correspondiente a $x = 0$, como se muestra en la figura 2.7. Teniendo en cuenta esto último, la expresión que se obtiene es

$$
\phi_0 = 3\phi_1 - 3\phi_2 + \phi_3. \tag{2.13}
$$

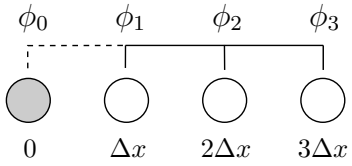


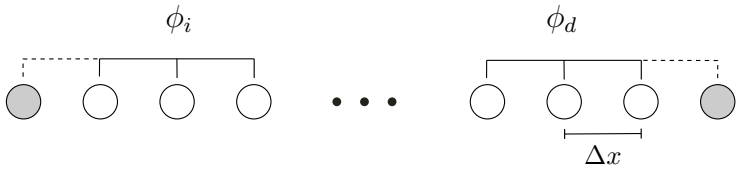
Figura 2.7. Interpolación discretizada de Lagrange para tres puntos.

Ahora, para el caso de 4, 5, 6 y 7 puntos se tienen las expresiones

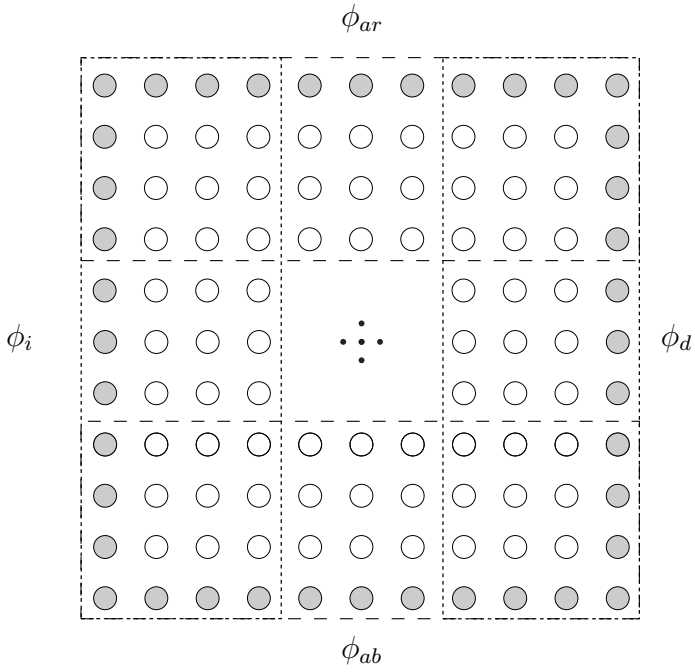
$$\begin{aligned}
\phi_0 &= 4\phi_1 - 6\phi_2 + 4\phi_3 - \phi_4, \\
\phi_0 &= 5\phi_1 - 10\phi_2 + 10\phi_3 - 5\phi_4 + \phi_5, \\
\phi_0 &= 6\phi_1 - 15\phi_2 + 20\phi_3 - 15\phi_4 + 6\phi_5 - \phi_6, \\
\phi_0 &= 7\phi_1 - 21\phi_2 + 35\phi_3 - 35\phi_4 + 21\phi_5 - 7\phi_6 + \phi_7,
\end{aligned} \tag{2.14}$$

donde se puede ver que cuanto mayor sea el número de puntos que se tomen más preciso será el cálculo del valor hallado; además menos propenso a errores locales, ya que la interpolación se realiza sobre un intervalo mayor (Alfio Quarteroni, 2012).

2.2.4 Condiciones de frontera. Para calcular el valor de ϕ en los puntos límite del dominio numérico de la malla base se utilizaron condiciones de frontera de onda saliente, es decir, que permiten el flujo hacia afuera de las fronteras definidas pero no hacia adentro. En este caso se separa la onda en sus modos fundamentales de propagación, es decir, el modo que se propaga a la izquierda y a la derecha para el caso en una dimensión, como se muestra en (1.11) y (1.10) y en la figura 1.1, o en los modos que se propagan a izquierda, derecha, arriba y abajo para el caso en dos dimensiones, como se muestra en (1.18), (1.17), (1.23) y (1.22), y en la figura 1.2. De tal manera que si se quiere calcular el valor de un punto en la frontera, solo se utilice el modo que sale por esa frontera, cancelando todos los demás. Posteriormente se interpolan los puntos, es decir, se utilizan (2.13) y (2.14) para calcular el valor de la función en el punto en la frontera a partir del valor de la misma función en sus puntos vecinos. Esto se ilustra mejor en la figura 2.8 para los casos en una y dos dimensiones.



(a)



(b)

Figura 2.8. En (a) se presenta el calculo de los puntos en las fronteras de la malla base para una dimensión y en (b) para dos dimensiones. Se utiliza interpolación de Lagrange discretizada, donde se calculan los puntos (grises) a partir de los tres puntos vecinos (blancos) del modo fundamental correspondiente, por medio de (2.13). En este caso, los puntos pequeños negros representan todos los puntos internos de la malla.

2.2.5 Movimiento de las mallas. En el caso de la ecuación de onda se pueden establecer tanto las trayectorias como la velocidades de los puntos que tienen el error máximo, y además se pueden utilizar estos datos para ubicar las mallas refinadas. Para que las mallas refinadas sigan estas trayectorias optimas se efectúa un desplazamiento espacial cada cierto número de pasos de tiempo n, lo cual define una velocidad v_m y teniendo en cuenta la constante de Courant $\mathbf{C} = v\Delta x/\Delta t$, donde v es la velocidad de la onda,

$$v_m = \frac{\Delta x}{n\Delta t} = \frac{v}{n * \mathbf{C}}, \tag{2.15}$$

para que la velocidad de propagación de la malla v_m sea la misma que la velocidad de la onda v, $n * \mathbf{C} = 1$. Para la malla en dos dimensiones, donde $\Delta x = \Delta y$,

$$\vec{v_d} = \frac{\Delta x}{n_x \Delta t}\hat{\imath} + \frac{\Delta y}{n_y \Delta t}\hat{\jmath} = \frac{v_x}{n_x * \mathbf{C}}\hat{\imath} + \frac{v_y}{n_y * \mathbf{C}}\hat{\jmath}, \tag{2.16}$$

y $n_x * \mathbf{C} = n_y * \mathbf{C} = 1$ para que la malla tenga la misma velocidad de la onda $\vec{v_m} = v_x\hat{\imath} + v_y\hat{\jmath}$. Esto se ve en forma más clara en las figuras 2.9 y 2.10, en donde la malla refinada se desplaza con velocidades específicas en una y dos dimensiones, se puede ver como se completan los puntos faltantes de la malla que se mueve utilizando interpolación lineal de la malla base (por simplicidad en esta figura no se tienen en cuenta los puntos fantasma de la malla refinada).

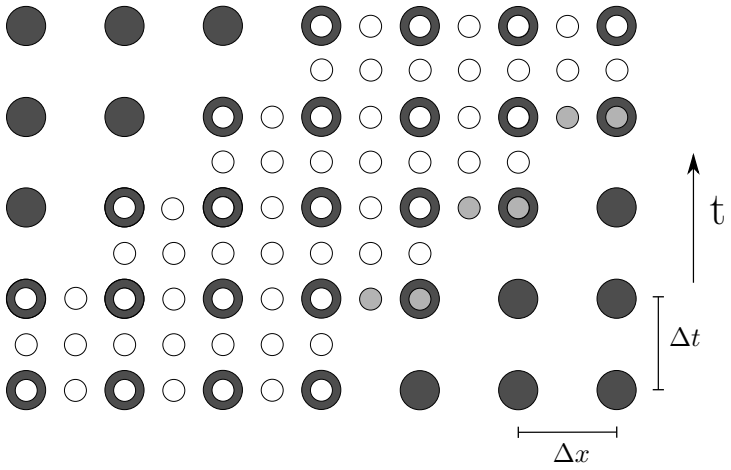


Figura 2.9. Desplazamiento de un subdominio refinado por un factor $\alpha = 2$ en una dimensión, a una velocidad v_m. Los puntos negros corresponden a la malla base, los puntos blancos a la malla refinada y los grises también a la malla refinada pero se calculan mediante interpolación de la malla base. La figura corresponde a un intervalo de tiempo.

2.3. Algoritmos

A continuación se presentan los algoritmos utilizados durante el desarrollo del proyecto, es decir, la secuencia de pasos lógicos implementados en los códigos de malla única y AMR. Para mostrarlos de la forma más simple se utiliza el pseudocódigo de cada uno de estos. El algoritmo 1 corresponde a la integración usando el método Runge Kutta 4 (RK4), en donde se ilustra la evolución de un paso de tiempo para una malla genérica; posteriormente se presenta el algoritmo 2 el cual es usado para llevar a cabo la evolución utilizando malla única, donde se tiene una malla que evoluciona con el mismo paso espacial y temporal para todo el dominio. Finalmente, el algoritmo 3 es utilizado para llevar a cabo la evolución usando el método AMR, el cual en este caso tiene un único subdominio refinado.

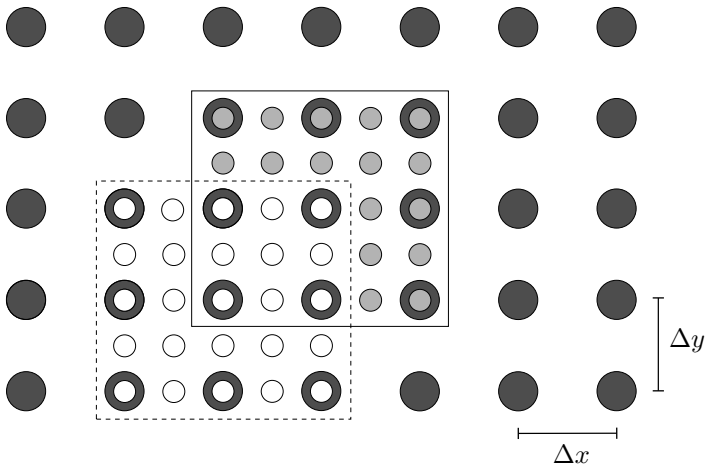


Figura 2.10. Desplazamiento de un subdominio refinado por un factor $\alpha = 2$ en dos dimensiones, a una velocidad $\vec{v_d}$. Los puntos negros corresponden a la malla base, los puntos blancos a la malla refinada y los grises también a la malla refinada pero se calculan mediante interpolación de la malla base. La figura corresponde al instante de tiempo en el que se efectúa el movimiento de la malla

Algoritmo 1 Runge Kutta 4, algoritmo para la evolución numérica de un arreglo de variables $U = [u, u_1, u_2, ...]$ un paso en el tiempo. Una explicación más detallada se encuentra en el apéndice A.

Entrada: Arreglo de variables $U(x, t_i)$
Salida: Arreglo de variables $U(x, t_{i+1})$
1: **for** $n = 1$ a $n = 4$ **do**
2: Calcular: k_n
3: Interpolar puntos en las fronteras
4: **end for**
5: $m = (k_1 + 2k_2 + 2k_3 + k_4)/6$
6: $U(x, t_{i+1}) = U(x, t_i) + m\Delta t$

Algoritmo 2 Malla única, algoritmo para la evolución temporal del arreglo de variables $U = [u, u_1, u_2, ...]$ con condición inicial $U(x, 0)$, utilizando una malla uniforme en todo el dominio, donde N representa el número de pasos que esta evoluciona en el tiempo.

Entrada: Dato Inicial $U(x, 0)$
Salida: Arreglo de variables $U(x, t_N)$ en un tiempo $t = t_N$
1: **for** $i = 0$ a $i = N - 1$ **do**
2: Aplicar **RK4** a $U(x, t_i)$
3: **end for**

Algoritmo 3 AMR. algoritmo para la evolución temporal de un problema usando AMR, $U_1(x,t_i)$ y $U_2(x,t_i)$ corresponden a los arreglos de variables para la malla base y la malla hijo, α es el factor de refinamiento del sub-dominio y N es el número de pasos que evoluciona la malla base en el tiempo. Cada n pasos de tiempo el arreglo $U_2(i)$ se desplaza Δx, lo cual corresponde a una velocidad de desplazamiento $v = \Delta x/(n\Delta t)$

Entrada: Datos Iniciales $U_1(x,0)$ y $U_2(x,0)$
Salida: Arreglo de variables $U_1(x,t_n)$ y $U_2(x,t_n)$ en un tiempo $t = t_n$

```
for i = 0 a i = N − 1 do
    if n es divisor de i then
        Desplazar U_2 Δx en x
        Interpolar puntos faltantes
    end if
    Aplicar RK4 a U_1(x, t_i)
    for l = 0 a l = α − 1 do
        Interpolar las fronteras de la malla U_2(x, t_i + lΔt/α) a partir de U_1(x, t_i) y U_1(x, t_{i+1})
        Aplicar RK4 a U_2(x, t_i + lΔt/α)
    end for
    Reemplazar U_2(x, t_{i+1}) en U_1(x, t_{i+1})
end for
```

2.4. Costo Computacional

Con el objetivo de explicar el incremento en el costo computacional asociado al refinamiento de una malla, se va a llamar a la evolución temporal de un punto (aplicar el RK4 sobre un punto) como una "operación", y dado que un porcentaje muy alto del trabajo requerido para realizar la evolución de un problema está dado por estas operaciones, se va a realizar el análisis en términos de ellas. Teniendo en cuenta que cuando se realiza un refinamiento se lleva a cabo tanto en el espacio como en el tiempo, debido a la relación $\Delta t = \mathbf{C}\Delta x$, cuando se refina una malla por un factor α, en una dimensión, se necesitan aproximadamente α^2 veces el número original de operaciones para llevar a cabo la misma evolución. Esto se puede ver de forma más clara en la figura 2.11, en donde, para evolucionar el mismo problema con un refinamiento $\alpha = 2$ se requieren $\alpha N_t(\alpha N_x - 1)$ operaciones, en contraste con el costo original que era $N_x N_t$ operaciones, acá N_x es el número de puntos de la malla original y N_t el número de pasos temporales que esta evoluciona. Para el caso en dos dimensiones, se necesitan aproximadamente α^3 veces el número original de operaciones para llevar a cabo la misma evolución (si se supone una región rectangular con parámetros N_x, N_y y N_t la expresión exacta sería $\alpha N_t(\alpha N_x - 1)(\alpha N_y - 1)$) y para el caso en tres dimensiones se necesitan aproximadamente α^4 veces el número original de operaciones para llevar a cabo la misma evolución (si se supone una región en forma de prisma de base rectangular con parámetros N_x, N_y, N_z y N_t la expresión exacta sería $\alpha N_t(\alpha N_x - 1)(\alpha N_y - 1)(\alpha N_z - 1)$), es decir, el número de operaciones crece de forma considerable cuando se quiere una resolución alta y más si las simulaciones se llevan a cabo en varias dimensiones. Según lo expuesto anteriormente, y con la intensión de ilustrar este incremento en el costo computacional, cuando se quiere refinar una

evolución en tres dimensiones por un factor $\alpha = 8$ se necesitará aproximadamente 4000 veces más trabajo que la evolución original. En este sentido el AMR ofrece una buena alternativa a refinar todo el dominio, dado que el refinamiento solo se aplica en los subdominios en los que es necesario, reduciendo considerablemente el costo computacional.

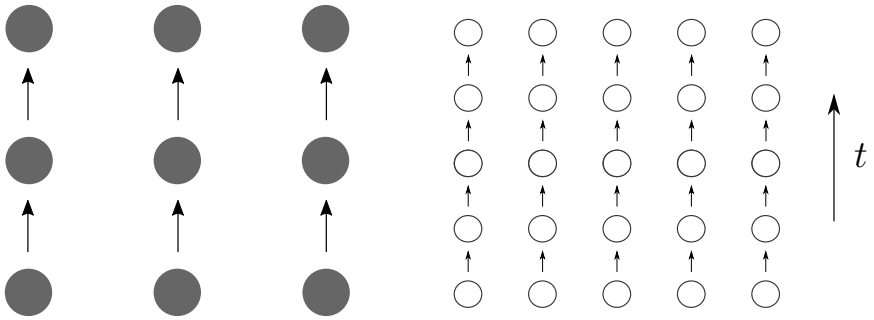


Figura 2.11. Incremento en el número de operaciones (flechas) asociado al refinamiento de la malla en una dimensión, por un factor $\alpha = 2$. En la figura de la izquierda está la malla original con un número de puntos $Nx = 3$ que evolucionan $N_t = 2$ pasos de tiempo, lo que equivale a $N_x * N_t = 6$ operaciones. En la figura de la derecha está la malla refinada en la cual se llevan a cabo $\alpha N_t(\alpha N_x - 1) = 20$ operaciones, para desarrollar la misma evolución.

Capítulo 3

Convergencia

Al usar un método numérico hay que tener claro que la discretización en diferencias finitas y la integración numérica proveen de una versión aproximada de la ecuación en el continuo, por lo cual la ecuación que se resolverá en adelante será solamente una aproximación de la solución analítica del problema que se quiere abordar. Es preciso entonces establecer criterios que indiquen que cuando se resuelve la versión discreta de una EDP, al menos en el límite continuo ($\Delta x \to 0$, $\Delta t \to 0$), se está calculando la solución correcta.

3.1. Orden de convergencia

Para obtener el orden de convergencia de un algoritmo es preciso examinar cada uno de los elementos de éste que impliquen una aproximación. En este caso se tienen tres, la interpolación, la discretización por diferencias finitas y el método de integración, aunque solo los dos últimos influyen de forma determinante. Para las **diferencias finitas** se considera la expansión de Taylor mostrada en (2.1), en donde, si se realiza una aproximación de segundo orden espacial y temporal como la mostrada en (2.3), se obtienen los residuos mostrados a continuación

$$\begin{aligned}\mathcal{O}(\Delta x^2) =& \frac{\phi^{(2)}(x_0)}{2!}\Delta x^2 + \cdots + \frac{\phi^{(n)}(x_0)}{n!}\Delta x^n + \mathcal{O}(\Delta x^{n+1}),\\ \mathcal{O}(\Delta t^2) =& \frac{\phi^{(2)}(t_0)}{2!}\Delta t^2 + \cdots + \frac{\phi^{(n)}(t_0)}{n!}\Delta t^n + \mathcal{O}(\Delta t^{n+1}),\end{aligned} \tag{3.1}$$

tomando en cuenta únicamente los términos principales, que son claramente mayores a todos los demás. En este caso, se puede decir que el error es proporcional a $(\Delta t)^2$ y a $(\Delta x)^2$; además, dado que $\Delta x = \mathbf{C}\Delta t$, siendo $\mathbf{C}$ una constante se tiene que el error numérico está dado por

$$Error_1(x,t) = \mathcal{O}(\Delta x^2, \Delta t^2) = E(x,t)\Delta x^2, \tag{3.2}$$

donde $E(x,t)$ corresponde a los coeficientes de error en los diferentes puntos para la aproximación en diferencias finitas. Ahora, si se define una nueva resolución $\Delta x_\beta = \Delta x/\beta$ con $\beta > 1$ se tiene que

$$Error_\beta(x,t) = E(x,t)\left(\frac{\Delta x}{\beta}\right)^2, \tag{3.3}$$

de donde

$$\frac{Error_1(x,t)}{Error_\beta(x,t)} = \beta^2, \tag{3.4}$$

es decir, si la resolución aumenta por un factor β, la solución de la versión discreta se aproxima a la versión continua de manera cuadrática, o también que el error respecto a la solución exacta se reduce por un factor de β^2. El hecho de que sea cuadrática es consecuencia de que se utilizó una aproximación en diferencias finitas de orden dos (Thomas, 2013).

Para el caso de la **integración usando RK4**, en su planteamiento se utiliza una aproximación por series de Taylor de orden 4 (Ascher and Petzold, 1998), por lo tanto converge a ese mismo orden. En este punto tiene que haber un equilibrio entre el orden de convergencia dado por las diferencias finitas y el orden dado por el método de integración, donde en general se espera que el algoritmo converja al menor de los dos, en este caso a segundo orden. En general, si se cumple el criterio de convergencia, la solución numérica se aproxima a la solución exacta del problema a resolver en el límite en que el tamaño del paso Δx tiende a cero.

En los casos en donde no se tiene una solución exacta, el orden de convergencia de un resultado numérico se calcula utilizando **autoconvergencia**. Para tal fin se utilizan las soluciones numéricas del problema a tratar con 3 tamaños de paso diferentes, en este caso Δx para $\phi_1(x,t)$, $\Delta x/\beta$ para $\phi_2(x,t)$ y $\Delta x/\beta^2$ para $\phi_3(x,t)$. Entonces, teniendo en cuenta que la solución numérica $\phi(x,t)$ es aproximadamente igual a la solución exacta $\phi_{exact}(x,t)$ más el error numérico dado por (3.2), se tiene que

$$\begin{aligned}
\phi_1(x,t) &\approx \phi_{exact}(x,t) + E_1(x,t)\Delta x^2, \\
\phi_2(x,t) &\approx \phi_{exact}(x,t) + E_2(x,t)(\Delta x/\beta)^2, \\
\phi_3(x,t) &\approx \phi_{exact}(x,t) + E_3(x,t)(\Delta x/\beta^2)^2,
\end{aligned} \tag{3.5}$$

donde $E_1(x,t) \approx E_2(x,t) \approx E_3(x,t)$ son denotados como $E(x,t)$. Finalmente, para hallar el orden de convergencia se procede de la siguiente forma

$$\begin{aligned}
\phi_1(x,t) - \phi_2(x,t) &\approx E(x,t)\Delta x^2\left(\frac{\beta^2-1}{\beta^2}\right), \\
\phi_2(x,t) - \phi_3(x,t) &\approx E(x,t)\Delta x^2\left(\frac{\beta^2-1}{\beta^4}\right),
\end{aligned} \tag{3.6}$$

obteniendo así que el orden de convergencia es

$$\text{orden} \approx \frac{\ln\left(\frac{\phi_1(x,t)-\phi_2(x,t)}{\phi_2(x,t)-\phi_3(x,t)}\right)}{\ln(\beta)} = \frac{\ln(\beta^2)}{\ln(\beta)} = 2. \tag{3.7}$$

Este resultado es igual a 2 dado a que se utilizó la aproximación en diferencias finitas de segundo orden como referencia para este desarrollo.

3.2. Error numérico y normas del error

La forma más simple para calcular el error numérico al resolver un sistema de EDPHs es utilizando la solución exacta como referencia, calculando el error como la resta entre la solución numérica obtenida y la solución exacta. Ahora, si no se tiene solución exacta del problema (lo cual es lo más común), el error se calcula utilizando la resta de dos soluciones numéricas. Tomando (3.5) y al restar dos soluciones con tamaños de paso Δx y $\Delta x/\beta$ se tiene que:

$$\phi_1(x,t) - \phi_2(x,t) \approx E(x,t)\Delta x^2 \left(1 - \frac{1}{\beta^2}\right), \tag{3.8}$$

el cual es un valor aproximado del error numérico escalado por una constante que se conoce.

Dado que cada punto de una malla tiene un error numérico asociado y casi siempre el número de puntos es muy grande (más si se trabaja en dos o tres dimensiones), las **normas del error** son parámetros usados para cuantificar el error en un cálculo numérico de una forma más simple, en donde se tiene un único valor de error que corresponde a todos los puntos de una malla para un instante de tiempo determinado. La norma L_1 discreta corresponde a la media aritmética, es decir

$$L_1(t) = \frac{1}{N}\sum_{i=1}^{N} Error(x_i,t), \tag{3.9}$$

la norma L_2 discreta es la media cuadrática

$$L_2(t) = \left(\frac{1}{N}\sum_{i=1}^{N} Error(x_i,t)^2\right)^{1/2}, \tag{3.10}$$

y en general se puede definir la norma L_n como

$$L_n(t) = \left(\frac{1}{N}\sum_{i=1}^{N} Error(x_i,t)^n\right)^{1/n}, \tag{3.11}$$

donde $Error(x_i,t)$ es el error numérico en el punto x_i utilizando un tamaño de paso Δx, para un instante de tiempo t y N es el número de puntos que tiene la malla. Para verificar la convergencia en el tiempo de un algoritmo es más fácil usar una de estas normas.

A continuación se presenta el análisis del valor de una norma general L_n para dos resoluciones diferentes, demostrándose que cualquier norma de este tipo cumple el criterio de convergencia anteriormente expuesto (3.4), dado que,

$$\begin{aligned} L_n(t)[\Delta x] &= \left[\frac{1}{N}\sum_{i=1}^{N}\left(E(x_i,t)\Delta x^2\right)^n\right]^{\frac{1}{n}} = \left[\frac{1}{N}\sum_{i=1}^{N} E(x_i,t)^n\right]^{\frac{1}{n}} \Delta x^2, \\ L_n(t)[\Delta x/\beta] &= \left[\frac{1}{N}\sum_{i=1}^{N}\left(E(x_i,t)\left(\frac{\Delta x}{\beta}\right)^2\right)^n\right]^{\frac{1}{n}} = \left[\frac{1}{N}\sum_{i=1}^{N} E(x_i)^n\right]^{\frac{1}{n}} \frac{\Delta x^2}{\beta^2}, \end{aligned} \tag{3.12}$$

se tiene que,

$$\frac{L_n(t)[\Delta x]}{L_n(t)[\Delta x/\beta]} = \beta^2, \tag{3.13}$$

lo cual muestra que converge de forma cuadrática dado que si se reduce Δx por un factor β la norma del error se reduce por un factor β^2, esto debido a que la deducción parte de una aproximación en diferencias finitas de segundo orden.

Capítulo 4

Resultados

4.1. Una dimensión

4.1.1 Malla única. Antes de desarrollar el código AMR es importante llevar a cabo la implementación de un código numérico de malla única, esto debido a que este permite analizar de forma clara el comportamiento del error numérico, y es un paso intermedio en el desarrollo del código AMR. Las características del código están dadas por una discretización con el método de diferencias finitas de orden 2, método de líneas, un integrador Runge Kutta de orden 4 y condiciones de frontera de onda saliente como las descritas en la sección 2.2.4. Los algoritmos 1 y 2 muestran tanto la evolución temporal como el funcionamiento global del código. Partiendo del problema planteado en (1.4), el cual corresponde a la ecuación de onda en una dimensión en coordenadas cartesianas descompuesta en un sistema de EDPHs de primer orden,

$$\begin{array}{ll} \text{EDPs} & \begin{cases} \partial_t u_1 = v\partial_x u_2, & x \in \Omega, t > 0 \\ \partial_t u_2 = v\partial_x u_1, & \end{cases} \\ \text{CI} & \begin{cases} u_1(x,0) = u_{10}(x), & x \in \Omega \\ u_2(x,0) = u_{20}(x), & \end{cases} \\ \text{CF} & \begin{cases} u_1(x,t) = g(x,t), & x \in \partial\Omega \\ u_2(x,t) = h(x,t), & \end{cases} \end{array}$$

donde $u_1 = \partial_t\phi$, $u_2 = \partial_x\phi$, $v = 1$, Ω es el dominio del problema, $\partial\Omega$ pertenece a la frontera de Ω, condiciones iniciales,

$$u_1(x,0) = \partial_t\phi(x,0) = \frac{1}{\sigma^2}A(x-x_0)\left(e^{\frac{-(x-x_0)^2}{\sigma^2}} - e^{\frac{-(x-x_0)^2}{\sigma^2}}\right), \tag{4.1}$$
$$u_2(x,0) = \partial_x\phi(x,0) = -A\left(\frac{1}{\sigma^2}\right)(x-x_0)\left(e^{\frac{-(x-x_0)^2}{\sigma^2}} + e^{\frac{-(x-x_0)^2}{\sigma^2}}\right),$$

con la condición auxiliar,

$$\phi(x,0) = \frac{1}{2}A\left(e^{\frac{-(x-x_0)^2}{\sigma^2}} + e^{\frac{-(x-x_0)^2}{\sigma^2}}\right),$$

y condiciones de contorno,

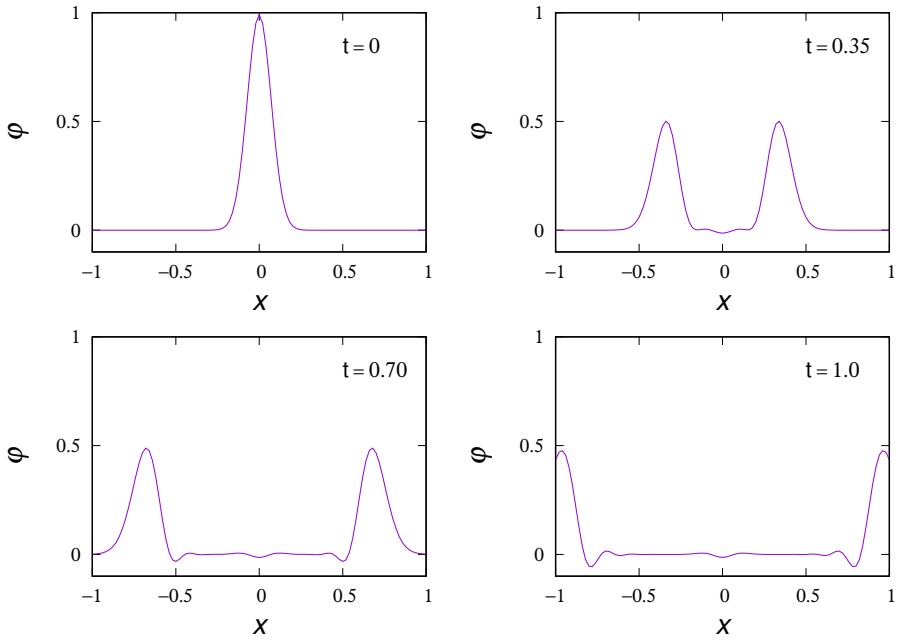
$$\begin{array}{ll} u_1(x_i,t) = \phi_i(x_i,t) & u_2(x_i,t) = \phi_i(x_i,t) \\ u_1(x_f,t) = -\phi_d(x_f,t) & u_2(x_f,t) = \phi_d(x_f,t) \end{array} \tag{4.2}$$

siendo x_i y x_f las fronteras del dominio Ω. Estas condiciones están mejor explicadas en la sección 2.2.4. Las condiciones iniciales corresponden a la función,

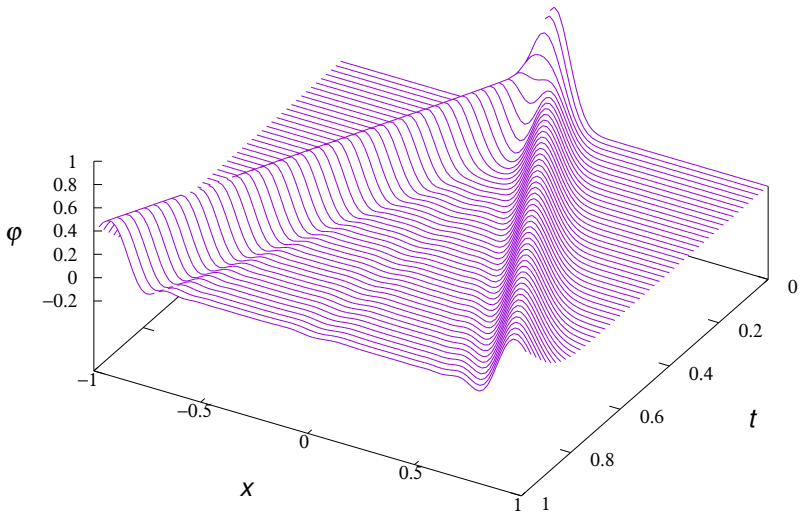
$$\phi(x,t) = \frac{1}{2}A\left(e^{\frac{-(x-x_0+t)^2}{\sigma^2}} + e^{\frac{-(x-x_0-t)^2}{\sigma^2}}\right), \tag{4.3}$$

y a sus derivadas evaluadas en $t = 0$, donde A es la amplitud inicial de la onda, σ define el ancho de la función gaussiana y x_0 es la posición inicial de la misma, esta función es solución exacta de la ecuación de onda en una dimensión, y además va a ser usada para calcular el error en la solución numérica y para verificar la validez de los resultados del código en términos de convergencia.

En la figura 4.1 se presenta la evolución numérica de este problema, en donde, en (a) se muestra la función de onda $\phi(x,t)$ en diferentes instantes de tiempo y en (b) el diagrama espacio-temporal para un intervalo de tiempo, lo cual está acorde con la solución exacta, es decir, hay dos pulsos que se propagan a derecha e izquierda con una velocidad $v = 1$. El pulso residual ubicado en $x = 0$ se debe al error numérico, igual que otras pequeñas oscilaciones que se verán mejor al realizar la comparación con la solución exacta.



(a)

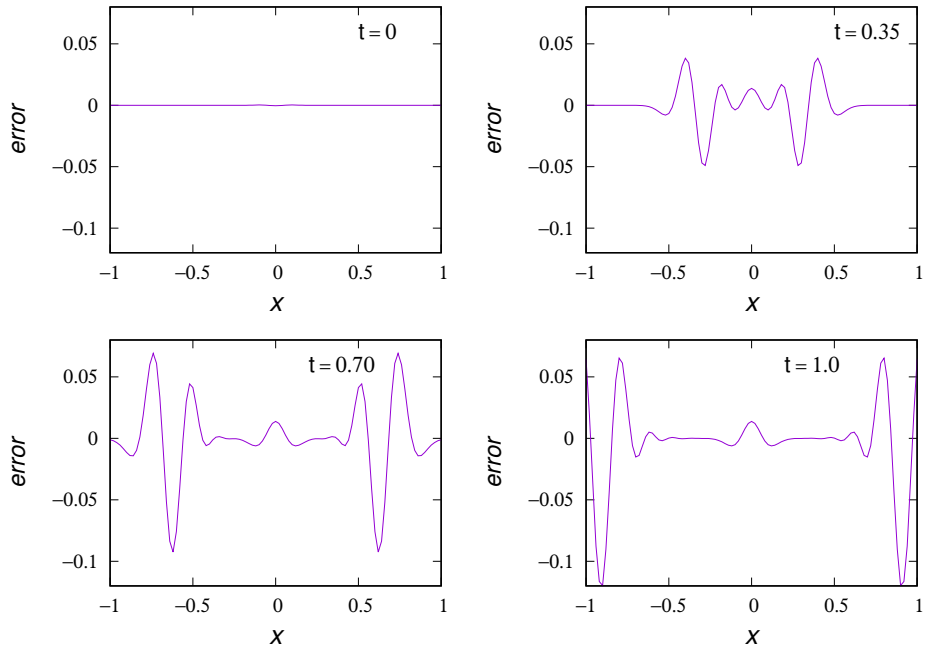


(b)

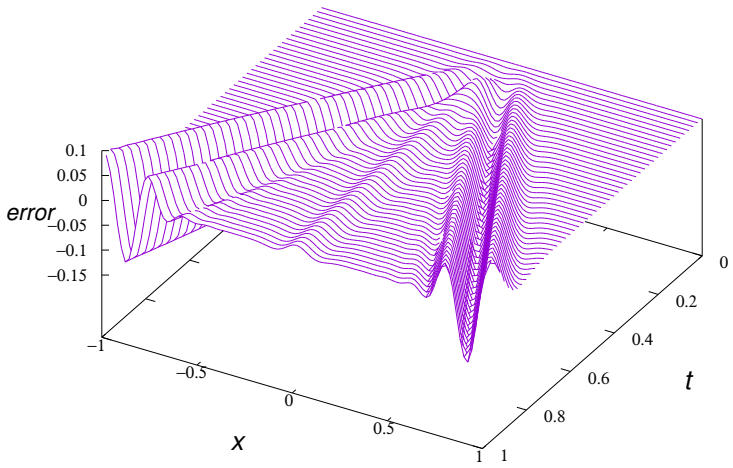
Figura 4.1. Evolución numérica de la ecuación de onda en una dimensión con malla única, para un dominio espacial $[-1, 1]$ y temporal $[0, 1]$, con un $\Delta x = 0.02$ y un $\mathbf{C} = 0.25$, para las condiciones iniciales dadas por (4.1) donde $A = 1$, $\sigma = 0.1$, $x_0 = 0$. En (a) se puede ver la forma de la onda en distintos instantes de tiempo y en (b) el diagrama espacio temporal para el intervalo de tiempo.

Posteriormente, se procede a analizar el error numérico que se genera a medida que se propaga la onda. En este caso, considerando el error numérico como la diferencia entre la solución exacta mostrada en (4.3) y la solución numérica. En la figura 4.2 se presenta la

evolución del error numérico para la solución numérica mostrada en la figura 4.1, al igual que en la figura anterior se presenta el error para cuatro instantes de tiempo y una gráfica espacio temporal en tres dimensiones para un intervalo de tiempo, en donde como es de esperarse se puede ver el pulso residual en $x = 0$ mencionado anteriormente y dos pulsos que viajan con la onda y que crecen a medida que esta se propaga, esto debido a que el error numérico se va acumulando.



(a)



(b)

Figura 4.2. Evolución del error numérico, al restar la solución exacta en (4.3) a la solución numérica mostrada en la figura 4.1. En (a) se puede ver la forma y la magnitud del error en distintos instantes de tiempo y en (b) un diagrama espacio temporal para un intervalo de tiempo.

Para calcular la solución numérica del problema es necesario hacer uso de los modos característicos de propagación de esta, en la figura 4.3 se presentan los resultados numéricos de estos para una dimensión, en donde en las figuras 4.3(a), 4.3(b) y 4.3(c) se presentan, respectivamente, la onda ϕ y los modos que se desplazan a izquierda ϕ_i y derecha ϕ_d en un instante de tiempo, acorde con (1.11) y (1.10).

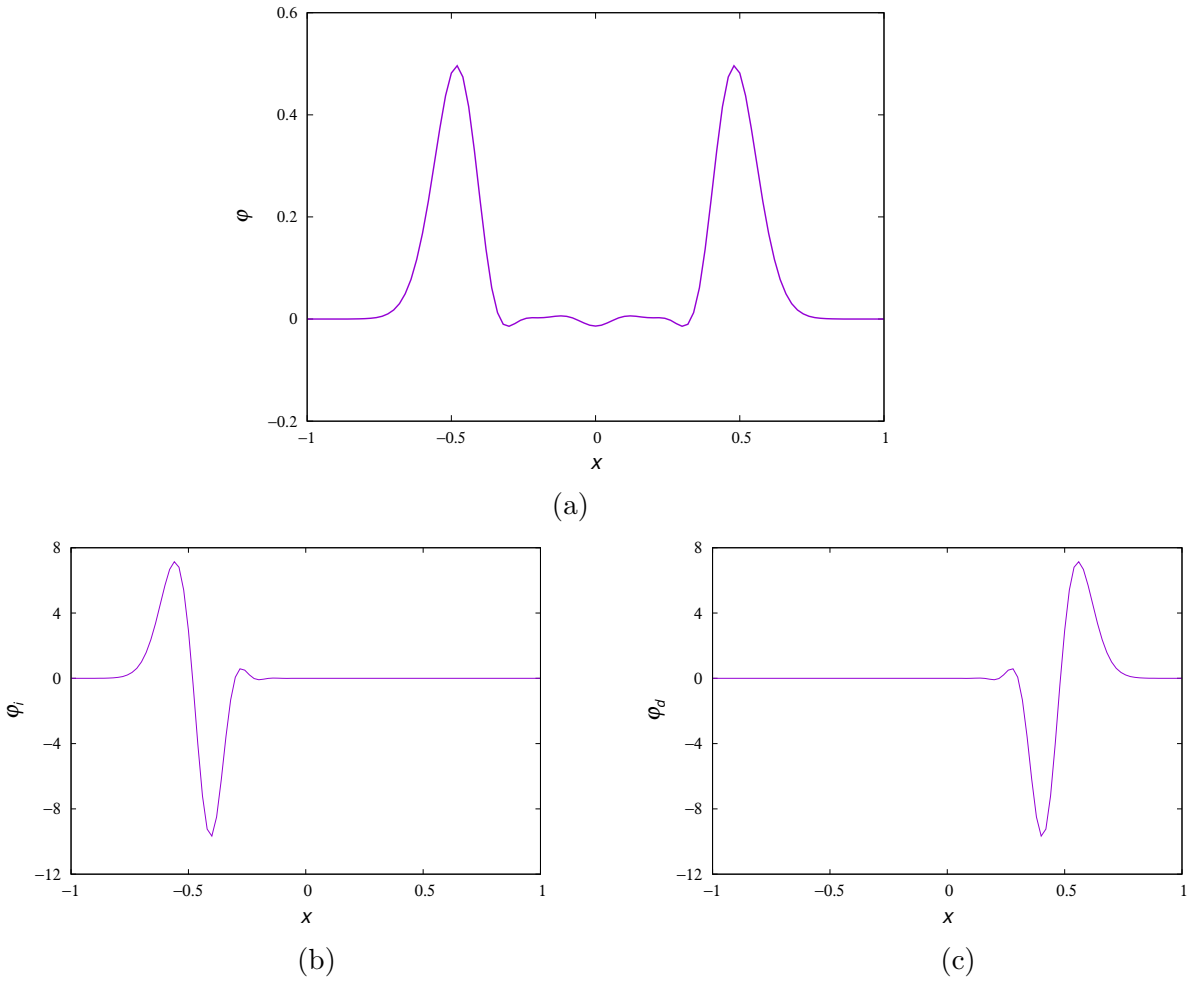
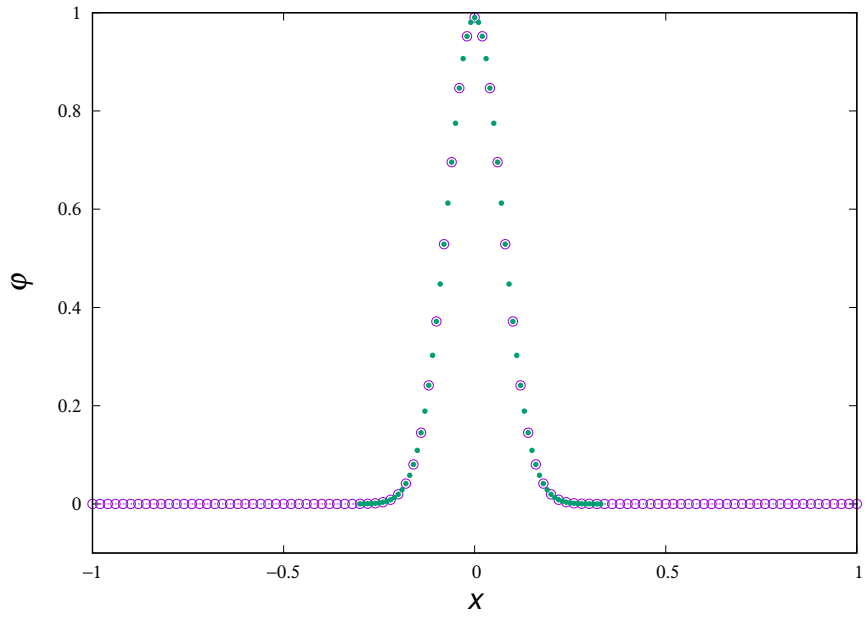


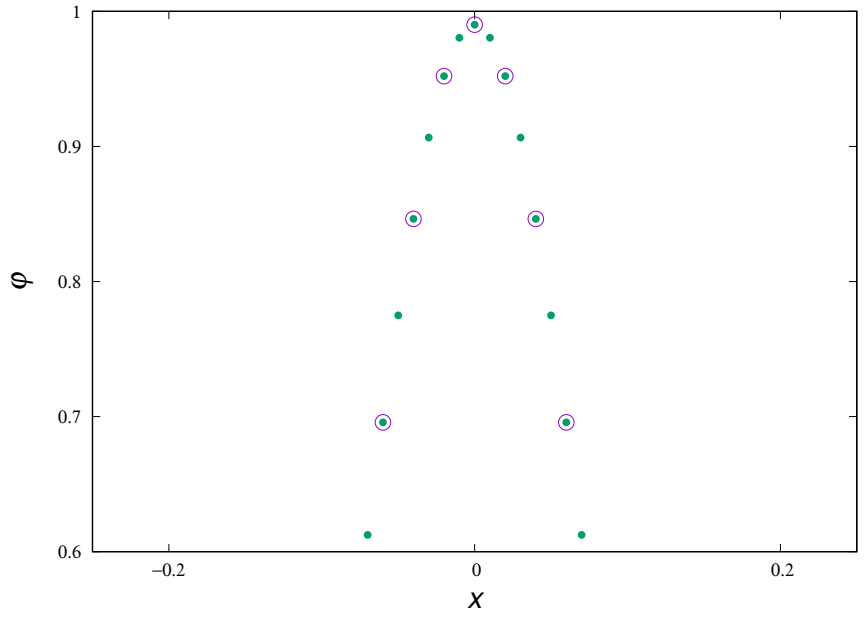
Figura 4.3. Descomposición de la solución numérica mostrada en la figura 4.1 en sus modos fundamentales. En (a) se muestra la onda en un instante de tiempo determinado, en (b) y (c) los modos ϕ_i y ϕ_d que se propagan a izquierda y derecha respectivamente en el mismo instante de tiempo.

Refinamiento y error numérico. Posterior a la implementación de los algoritmos con malla única se procedió a implementar el algoritmo 3, el cual adapta la malla a las condiciones del problema, refinando las regiones que presentan el máximo error, se usaron las mismas condiciones iniciales utilizadas anteriormente para realizar la evolución numérica. En las figuras 4.4 y 4.5 se muestra como se realiza la discretización para dos mallas (padre e hijo), cada una con una resolución y un dominio diferente, en este caso la relación entre el tamaño de paso es $\alpha = 2$ y $\alpha = 8$ respectivamente. Se puede ver claramente como la región que tiene la malla refinada está descrita de forma más precisa, y además, que a medida que se aumenta el factor de refinamiento α aumenta la precisión. Las figuras mencionadas corresponden a

un instante de tiempo; sin embargo, las regiones refinadas espacialmente se refinan en igual medida temporalmente, dado que Δx y Δt están relacionados por una constante $\mathbf{C}$, esto se mostró en la figura 2.3.

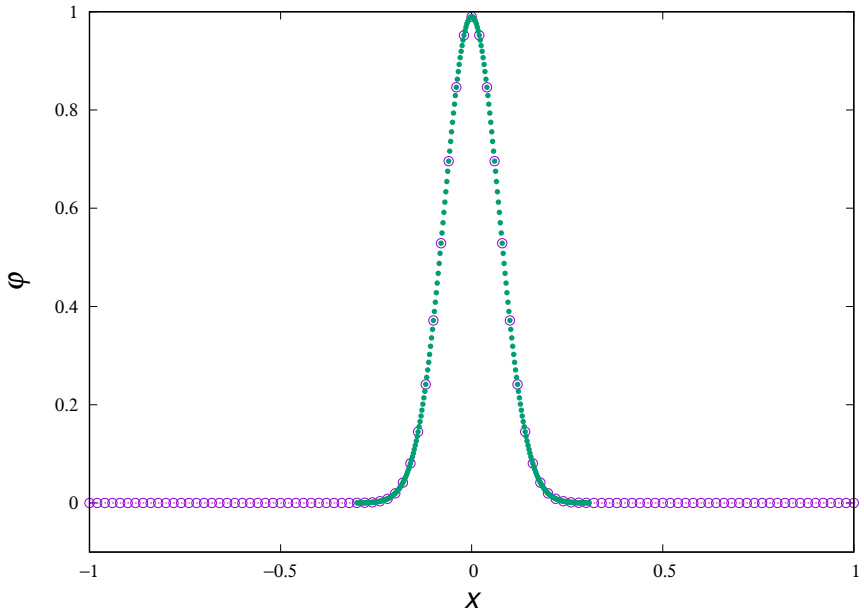


(a)

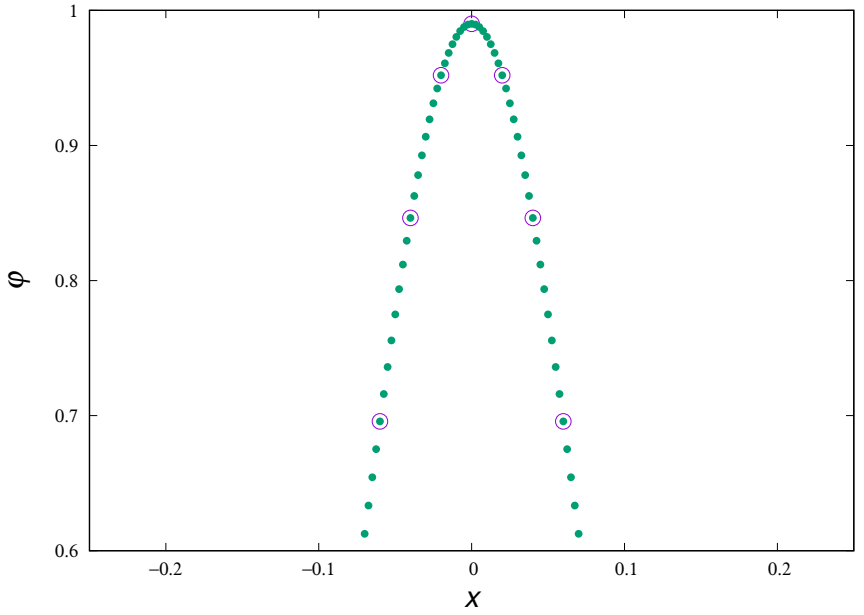


(b)

Figura 4.4. Discretización de dos dominios en dos niveles diferentes utilizando el algoritmo AMR para un instante de tiempo $t = 0.05s$, el dominio padre (l_0) es $[-1, 1]$ y el subdominio hijo (l_1) es $[-0.5, 0.5]$. La relación del tamaño del paso entre dominios está dada por el factor $\alpha = 2$. En (a) se muestra el dominio completo, y en (b) un acercamiento para observar mejor la discretización.



(a)

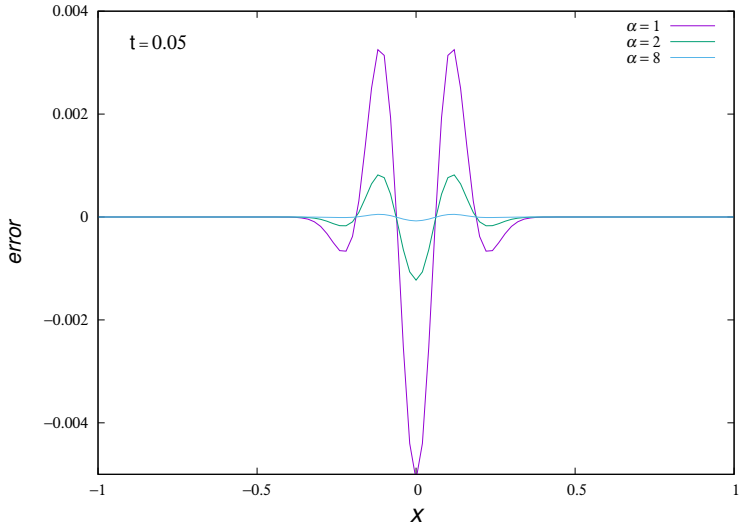


(b)

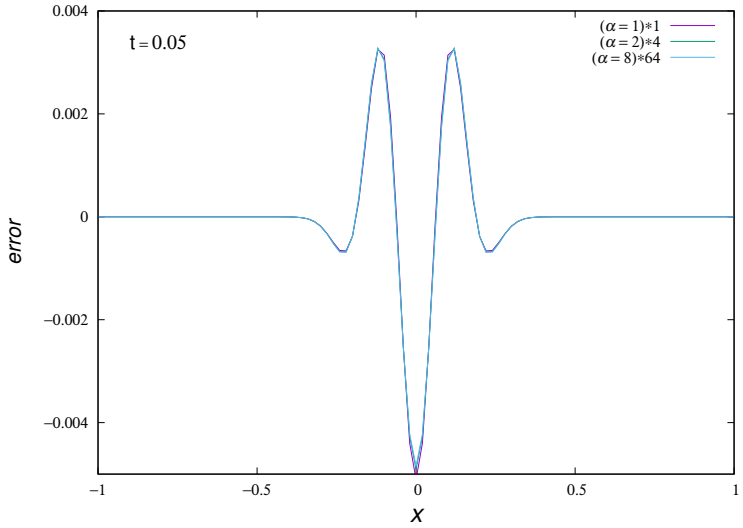
Figura 4.5. Discretización de dos dominios en dos niveles diferentes l_0 y l_1 utilizando el algoritmo AMR para un instante de tiempo $t = 0.05s$, el dominio padre (l_0) es $[-1, 1]$ y el subdominio hijo (l_1) es $[-0.5, 0.5]$, La relación del tamaño del paso entre dominios está dada por el factor $\alpha = 8$. En (a) se muestra el dominio completo, y en (b) un acercamiento para observar mejor la discretización.

Por otra parte, en la figura 4.6 se puede ver una comparación del error numérico en el mismo instante de tiempo del algoritmo de malla única y el error de aquellos que tienen un subdominio refinado por un factor α, en donde se muestra un buen comportamiento en términos convergencia del error numérico. Al multiplicar cada uno de los errores por su respectivo factor de convergencia α^2 (como se muestra en (3.4)), se puede ver que su comportamiento

obedece a la teoría y que solo hubo necesidad de refinar la región en donde estaba el máximo del error, obteniéndose resultados equiparables a haber resuelto el mismo problema con todo el dominio refinado con la resolución más alta, pero en el caso del refinamiento adaptativo con un número de operaciones muy inferior.



(a)

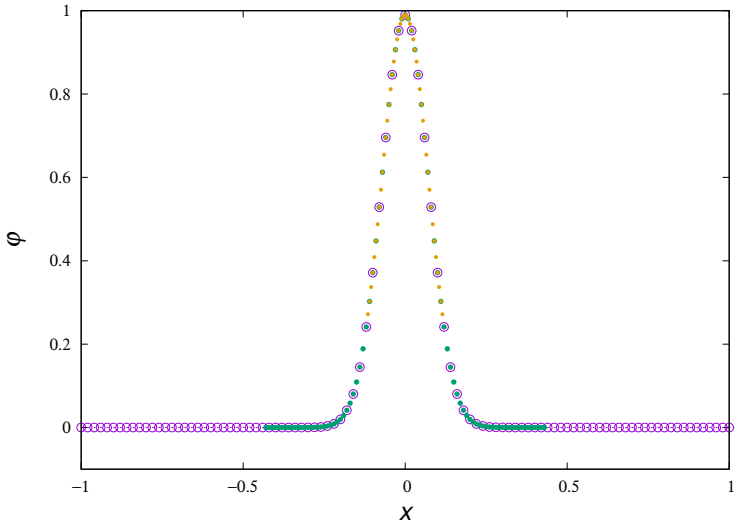


(b)

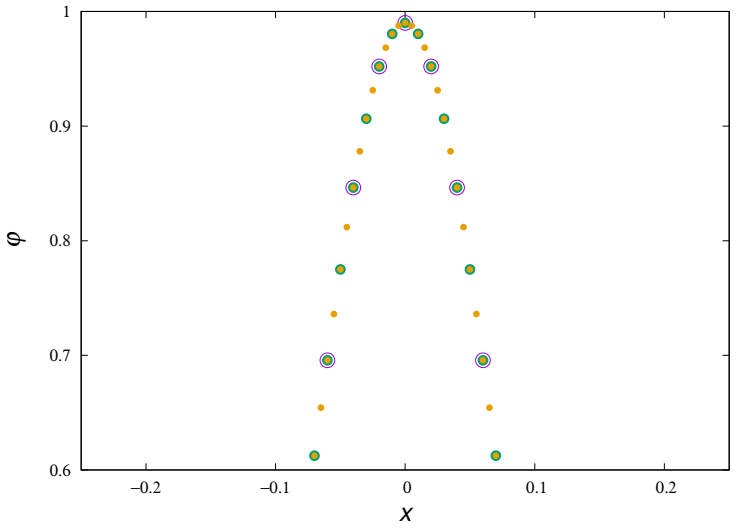
Figura 4.6. Comparación del error numérico en un instante de tiempo $t = 0.05s$ para resoluciones con $\alpha = 1$ como en la figura 4.1, $\alpha = 2$ como en la figura 4.4 y $\alpha = 8$ como en la figura 4.5. En (a) está el error numérico para las 3 resoluciones manteniendo las magnitudes originales y en (b) se multiplica por 4 el error de la malla que fue refinada por un factor $\alpha = 2$ y por 64 el error de la que fue refinada por $\alpha = 8$, esto acorde con el criterio de convergencia de (3.4).

Ahora, como se mencionó anteriormente el algoritmo AMR puede refinar por niveles, es decir, se puede refinar subdominios dentro de otros que ya hayan sido refinados, como se muestra en la figura 2.1, y este procedimiento es recurrente, es decir, se ejecuta un único procedimiento entre niveles como el mostrado en la figura 2.3. En la figura 4.7 se presentan

los resultados de este caso, en donde hay tres niveles, cada uno con una resolución particular y un dominio específico, se establece la relación del tamaño del paso entre dominios por el factor $\alpha = 2$. Esta forma de refinar una malla es muy usada cuando se necesita tener una resolución muy alta en un sector específico pero a la vez no resulta adecuado un cambio brusco de resolución, por tanto se llega a resoluciones muy altas refinando gradualmente. Por otra parte, es bueno aclarar que el algoritmo AMR puede seguir incluyendo dominios anidados indefinidamente.



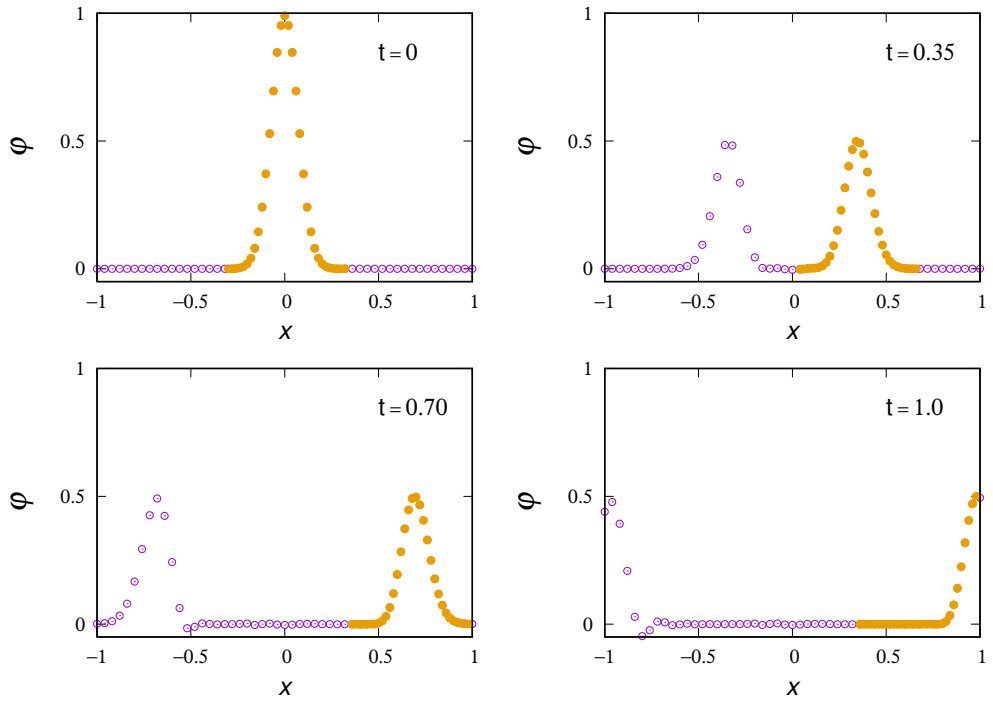
(a)



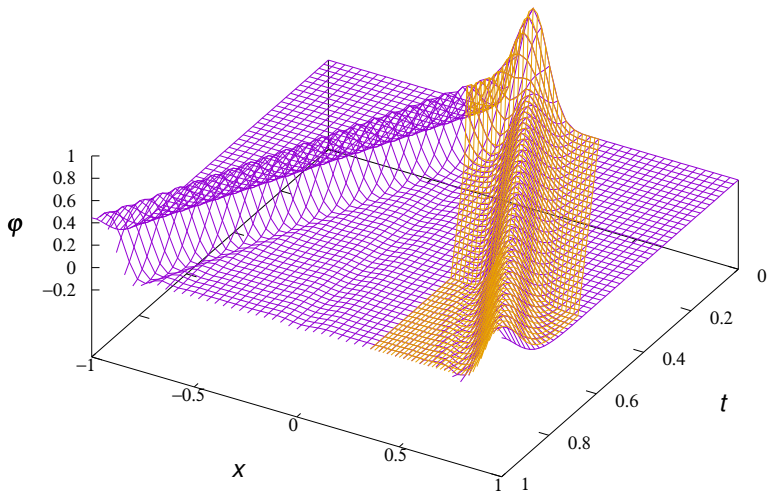
(b)

Figura 4.7. Discretización de tres dominios en tres niveles diferentes (l_0, l_1 y l_2) utlizando el algoritmo AMR para un instante de tiempo $t = 0.05s$, el dominio padre (l_0) es $[-1, 1]$, el subdominio hijo (l_1) es $[-0.5, 0.5]$ y el subdominio nieto (l_2) es $[-0.25, 0.25]$. La relación del tamaño del paso entre dominios está dada por el factor $\alpha = 2$. En (a) se muestra el dominio completo, y en (b) un acercamiento para observar mejor la discretización.

4.1.2 Malla Refinada Adaptativa (AMR). En la sección anterior se mostraron únicamente capturas de tiempo del AMR, esto con el objetivo de ilustrar como se lleva a cabo la discretización. Ahora, se va a mostrar como estos dominios refinados se adaptan a problemas específicos viajando con la región que presenta el error máximo. En la figura 4.8 se puede ver el mismo problema resuelto anteriormente pero ahora con una malla refinada que se adapta a la dinámica del problema, desplazándose a la misma velocidad de uno de los pulsos, esto teniendo en cuenta que la región en la que se encuentra el pulso es la que tiene la magnitud máxima del error numérico como se vio en la figura 4.2. Este desplazamiento se efectúa por medio de (2.15), ajustando la velocidad de la malla v_d (dándole un valor a n) con la velocidad de propagación de la onda (la cual viene dada por la EDP). El factor de refinamiento es $\alpha = 2$ y se refina únicamente uno de los pulsos, esto con el objetivo de ver más adelante el contraste entre el error numérico de los dos pulsos.



(a)

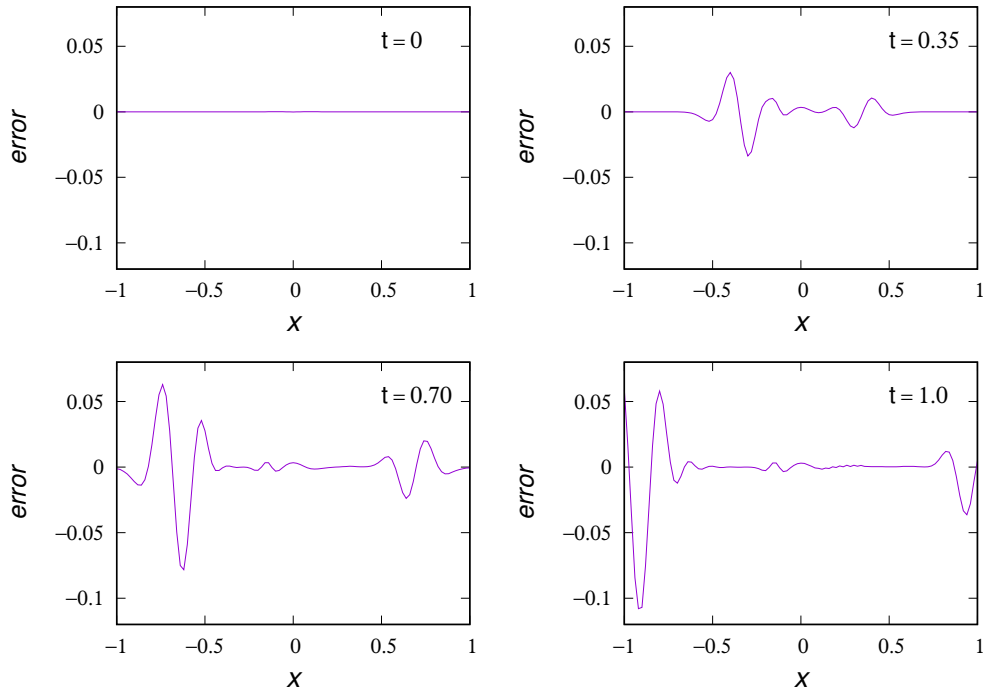


(b)

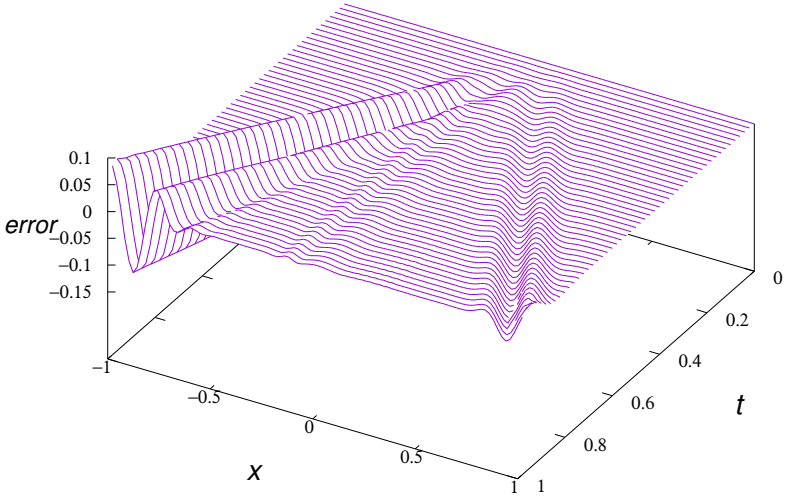
Figura 4.8. AMR, Malla refinada con un dominio inicial $[-0.3, 0.3]$ que se desplaza a la misma velocidad de uno de los pulsos dentro del dominio de la malla base $[-1.0, 1.0]$, para un intervalo temporal $[0, 1.0]$. Las condiciones iniciales dadas por (4.1) donde $A = 1$, $\sigma = 0.1$, $x_0 = 0$. En (a) se presentan capturas para diferentes instantes de tiempo de los dominios discretizados y en (b) se presenta un intervalo de tiempo completo.

En la figura 4.9 se muestran los resultados del error numérico, calculado a partir de la

solución numérica mostrada en la figura 4.8, la figura 4.9(a) presenta el error para cuatro instantes de tiempo diferentes y la figura 4.9(b) el diagrama espacio temporal del error para un intervalo de tiempo, donde se puede ver que el pulso que viaja con una malla refinada presenta una disminución del error, la cual está dada por la nueva resolución.



(a)

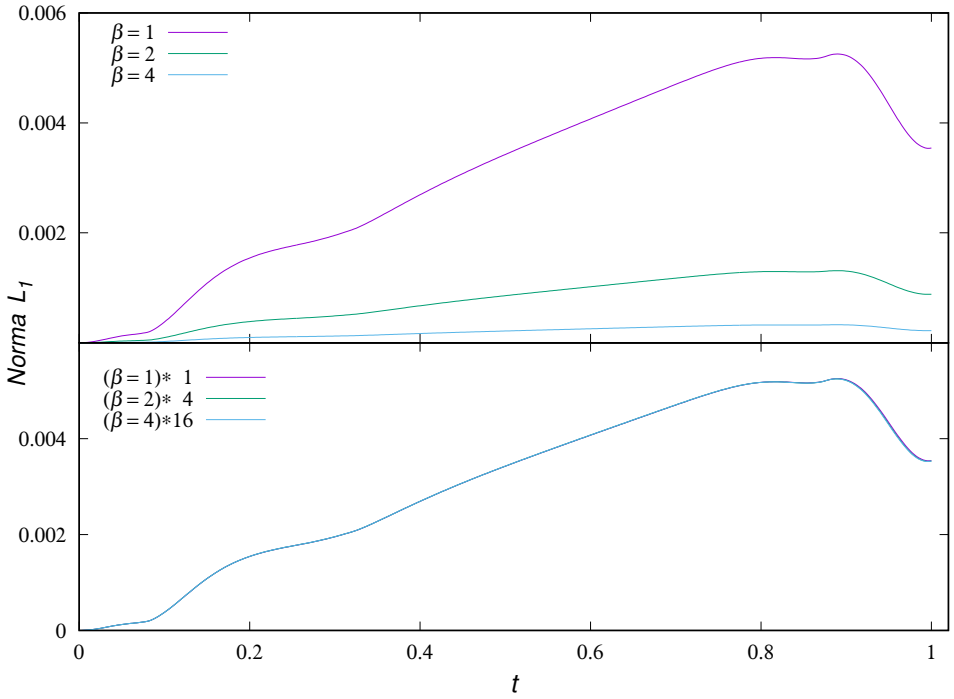


(b)

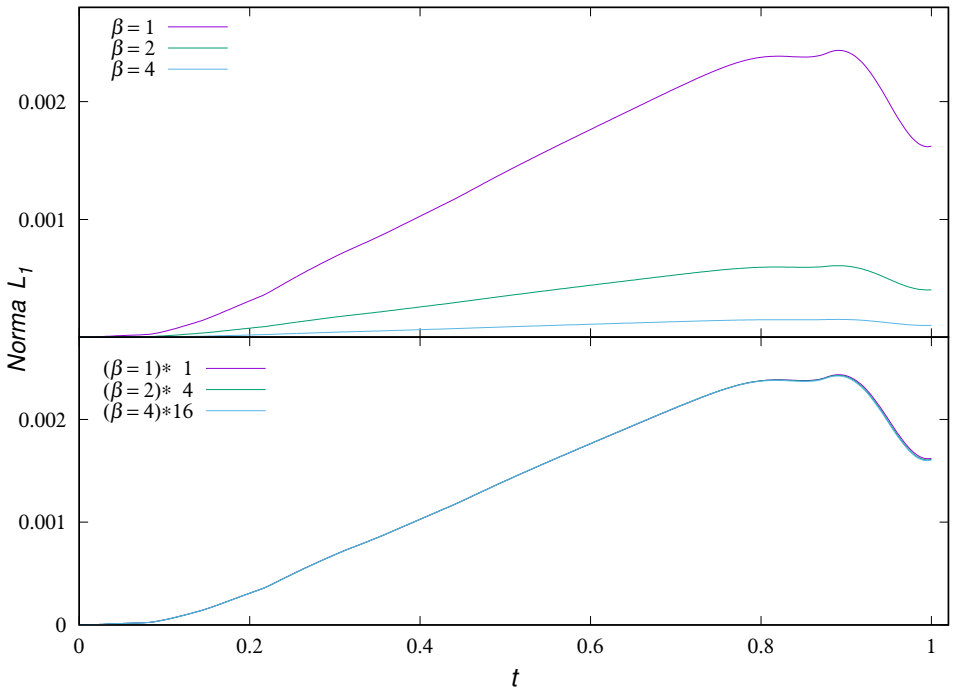
Figura 4.9. Evolución del error numérico usando AMR en uno solo de los pulsos de la onda. El error es calculado al restar la solución exacta en (4.3) a la solución numérica mostrada en 4.8. En la figura (a) se puede ver la forma y la magnitud del error en distintos instantes de tiempo y en (b) la evolución completa de este para un intervalo de tiempo.

4.1.3 Pruebas de convergencia. En general para corroborar el buen funcionamiento de un algoritmo hay que verificar que su solución numérica converge a la solución exacta en el

límite en que la discretización tanto temporal como espacial tienden a cero, es decir, cuando $\Delta x \to 0$ y $\Delta t \to 0$. Para esto se utiliza alguna de las normas del error L_n, las cuales están definidas en (3.11), en donde, para cada instante de tiempo se va a tener un único valor del error correspondiente a todos los puntos de la malla, lo cual permite verificar cómo cambia este valor cuando cambia la resolución base de la malla. Como una garantía de que los algoritmos implementados en una dimensión funcionan bien, es decir, que sus resultados numéricos son válidos, a continuación se presentan las pruebas de convergencia para cada uno de ellos. Si se tienen los datos del error numérico en un intervalo de tiempo para simulaciones que tienen diferente tamaño de paso base (es decir el nivel l_0 tiene diferente resolución), se puede verificar la convergencia de la norma mediante (3.13), donde β sería la relación entre el tamaño de paso para las diferentes simulaciones y n el tipo de norma a utilizar. En este caso se utilizó la norma L_1 descrita por (3.9), y dadas las características de los códigos descritos anteriormente estos tiene orden de convergencia 2. En la figura 4.10 se presentan las gráficas de convergencia para estos dos algoritmos, en este caso la norma L_1 se calcula para 3 resoluciones base, es decir, $\beta = 1$, $\beta = 2$ y $\beta = 4$, siendo los respectivos factores de convergencia $\beta^2 = 1$, $\beta^2 = 4$ y $\beta^2 = 16$. En la figura 4.10(a) está la prueba de convergencia correspondiente al algoritmo de malla única y en la figura 4.10(b) la correspondiente al algoritmo AMR, donde se puede ver que los algoritmos convergen adecuadamente acorde con la teoría. También al observar la magnitud del error global se puede ver como este disminuye en el caso del algoritmo que usó AMR.



(a) Malla única



(b) AMR

Figura 4.10. Convergencia de la norma L_1 para los algoritmos malla única y AMR. La norma L_1 se calcula para 3 resoluciones base, en este caso la relación entre resoluciones es $\beta = 2$ (verde) y $\beta = 4$ (celeste) y los respectivos factores de convergencia son $\beta^2 = 4$ (verde) y $\beta^2 = 16$ (celeste). En (a) se presenta la prueba de convergencia para el algoritmo con malla única, en la parte de arriba se tienen las normas originales y en la de abajo las mismas multiplicadas por sus respectivos factores de convergencia, la prueba de convergencia correspondiente al algoritmo AMR se presenta en (b).

4.2. Dos dimensiones

4.2.1 Malla única. Al igual que en el caso en una dimensión, antes de desarrollar código AMR se llevó a cabo la implementación de un código numérico de malla única para la solución de la ecuación de onda en dos dimensiones. Las características del código en dos dimensiones son iguales a las descritas anteriormente para una dimensión. En los algoritmos 1 y 2 se presenta tanto de evolución como el funcionamiento global del código. Partiendo del problema planteado en (1.13), el cual corresponde a la ecuación de onda en dos dimensión en coordenadas cartesianas descompuesta en un sistema de EDPHs de primer orden,

$$\begin{array}{cl}
\text{EDPs} & \begin{cases} \partial_t u_1 = v\partial_x u_2, & x,y \in \mathbb{R}, t > 0 \\ \partial_t u_2 = v\partial_x u_1, & \\ \partial_t u_3 = v\partial_y u_1, & \end{cases} \\
\text{CI} & \begin{cases} u_1(x,y,0) = u_{10}(x,y), & x,y \in \mathbb{R} \\ u_2(x,y,0) = u_{20}(x,y), & \\ u_3(x,y,0) = u_{30}(x,y), & \end{cases} \\
\text{CF} & \begin{cases} u_1(x,y,t) = g(x,y,t), & x,y \in \partial\Omega \\ u_2(x,y,t) = h(x,y,t), & \\ u_3(x,y,t) = l(x,y,t), & \end{cases}
\end{array}$$

donde $u_1 = \partial_t\phi$, $u_2 = v\partial_x\phi$, $u_3 = v\partial\phi$, y $v = 1$, Ω es el dominio del problema, $\partial\Omega$ pertenece a la frontera de Ω, condiciones iniciales,

$$\begin{aligned}
u_3(x,y,0) &= \partial_y\phi(x,y,0) = -2A\left(\frac{1}{\sigma^2}\right)(y-y_0)e^{-\frac{(x-x_0)^2+(y-y_0)^2}{\sigma^2}}, \\
u_1(x,y,0) &= \partial_t\phi(x,y,0) = 0 \\
u_2(x,y,0) &= \partial_x\phi(x,y,0) = -2A\left(\frac{1}{\sigma^2}\right)(x-x_0)e^{-\frac{(x-x_0)^2+(y-y_0)^2}{\sigma^2}},
\end{aligned} \tag{4.4}$$

con la condición auxiliar,

$$\phi(x,y,0) = A\left(e^{\frac{-(x-x_0)^2+(y-y_0)^2}{\sigma^2}}\right), \tag{4.5}$$

y condiciones de contorno,

$$\begin{array}{ll}
u_1(x_i,y,t) = \phi_i(x_i,y,t) & u_2(x_i,y,t) = \phi_i(x_i,y,t) \\
u_1(x_f,y,t) = \phi_d(x_f,y,t) & u_2(x_d,y,t) = -\phi_d(x_f,y,t) \\
u_1(x,y_i,t) = \phi_{ab}(x,y_i,t) & u_3(x,y_i,t) = -\phi_{ab}(x,y_i,t) \\
u_1(x,y_f,t) = \phi_{ar}(x,y_f,t) & u_3(x,y_f,t) = \phi_{ar}(x,y_f,t)
\end{array} \tag{4.6}$$

siendo x_i, x_d, y_i y y_f las fronteras del dominio Ω. Estas condiciones están mejor explicadas en la sección 2.2.4. Las condiciones iniciales corresponden a la función,

$$\phi(x,y,t) = \frac{1}{2}A\left(e^{-\frac{\left(\sqrt{(x-x_0)^2+(y-y_0)^2}+t\right)^2}{\sigma^2}} + e^{-\frac{\left(\sqrt{(x-x_0)^2+(y-y_0)^2}-t\right)^2}{\sigma^2}}\right) \tag{4.7}$$

y sus derivadas evaluadas en $t = 0$, en donde A define la amplitud inicial de la onda, σ el ancho y x_0 y y_0 la posición de la onda en $t = 0$. Esta función no es solución exacta del problema, por lo que para calcular el error de la solución numérica y para verificar la validez de los resultados del código en términos de convergencia, se hará uso de ecuaciones diferentes a las usadas en la sección 4.1.1.

En la figura 4.11 se presenta la evolución numérica de este problema, en donde se muestra la onda en diferentes instantes de tiempo y se pueden observar la amplitud y la forma como se propaga, lo cual está acorde con EDP planteada; es decir, la onda se propaga en todas la direcciones con la misma velocidad $v = 1$ y con una amplitud decreciente en el tiempo. El pulso residual ubicado en el origen $x = 0$ y $y = 0$ se debe al error numérico, al igual que otras pequeñas oscilaciones que viajan con la onda, las cuales se verán mejor más adelante.

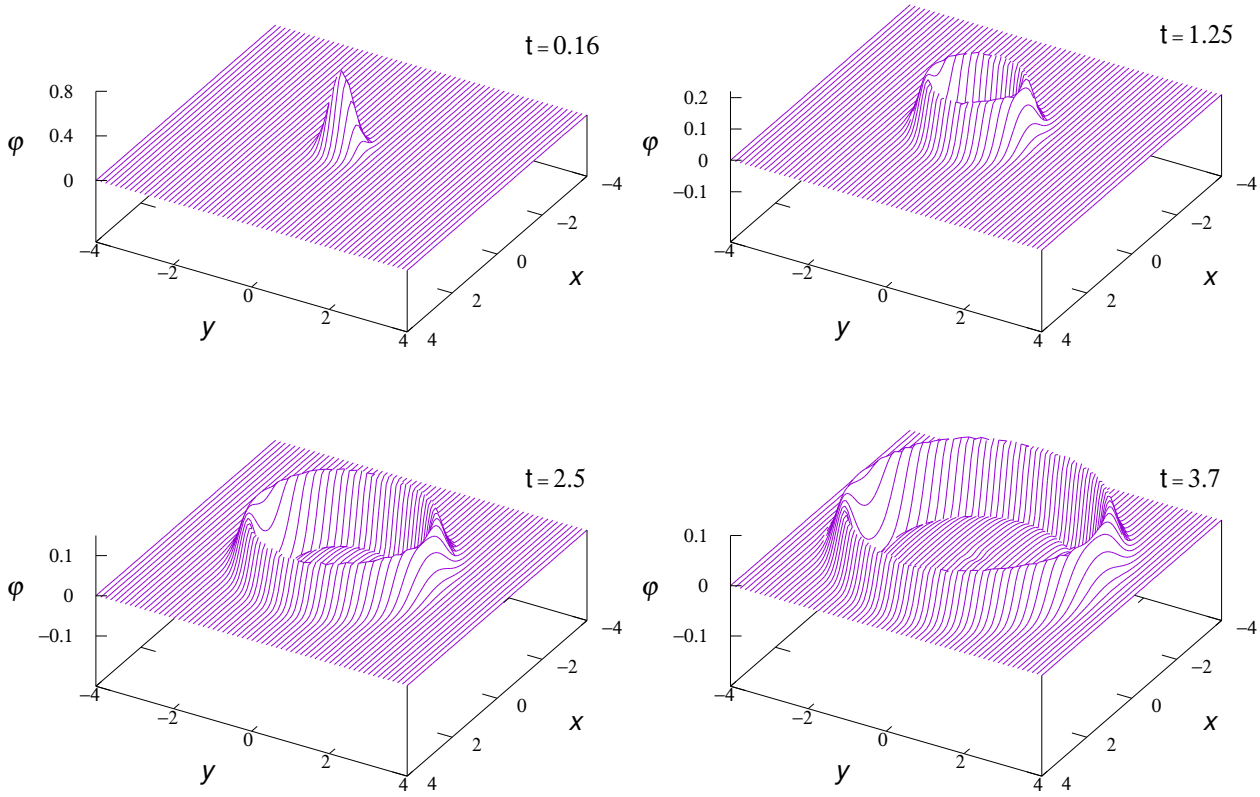


Figura 4.11. Evolución numérica de la ecuación de onda en dos dimensiones utilizando malla única. Las condiciones iniciales están dadas por (4.7) donde $A = 1$, $\sigma = 0.1$, $x_0 = 0$ y $y_0 = 0$. El dominio espacial es $[-4, 4]$ en x y $[-4, 4]$ en y, el dominio temporal es $[0, 4.2]$, el tamaño de paso es $\Delta x = 0.02$, $CFL = 0.25$ y se utilizaron condiciones de frontera de onda saliente.

En este caso para calcular el error numérico que se genera a medida que se propaga la onda se tiene en cuenta (3.8), la cual se usa para hallar el error numérico de un problema que no tiene solución exacta, mediante la resta de dos soluciones numéricas. En la figura 4.12 se presenta la evolución del error numérico para la solución mostrada en la figura 4.11. Se ve claramente el pulso residual en el origen mencionado anteriormente y un pulso que viaja radialmente acompañando la onda, este último crece en el tiempo dado que el error se va acumulando a medida que se propaga.

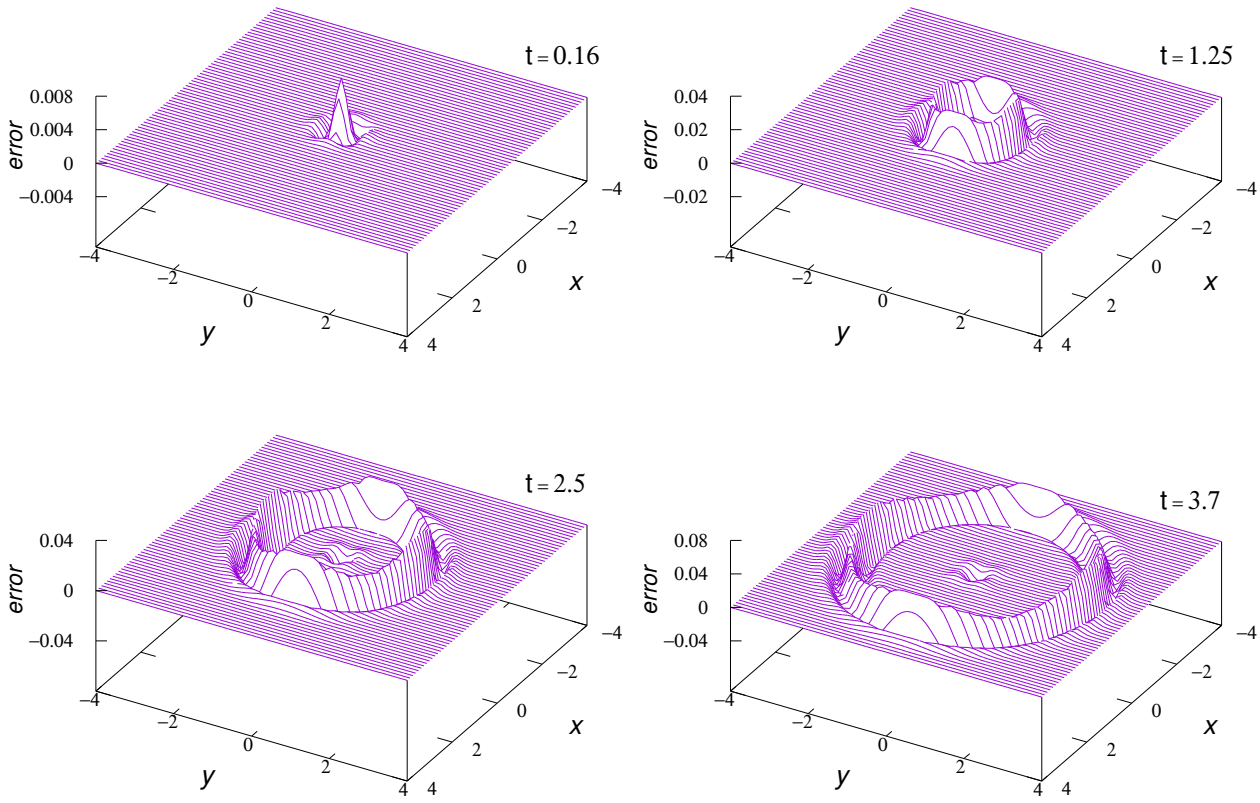
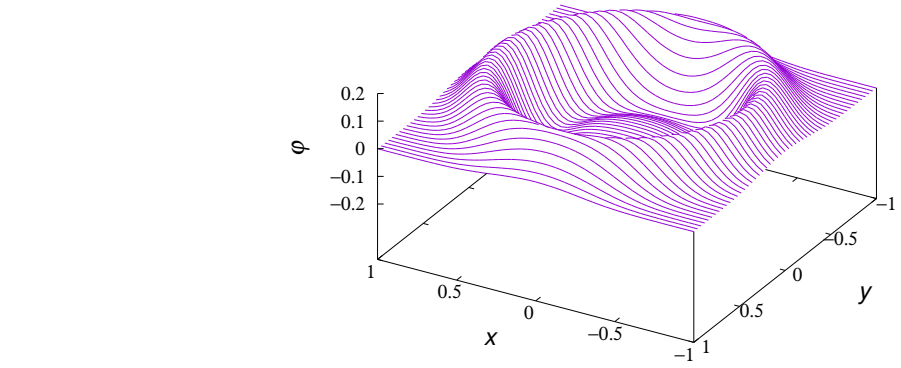
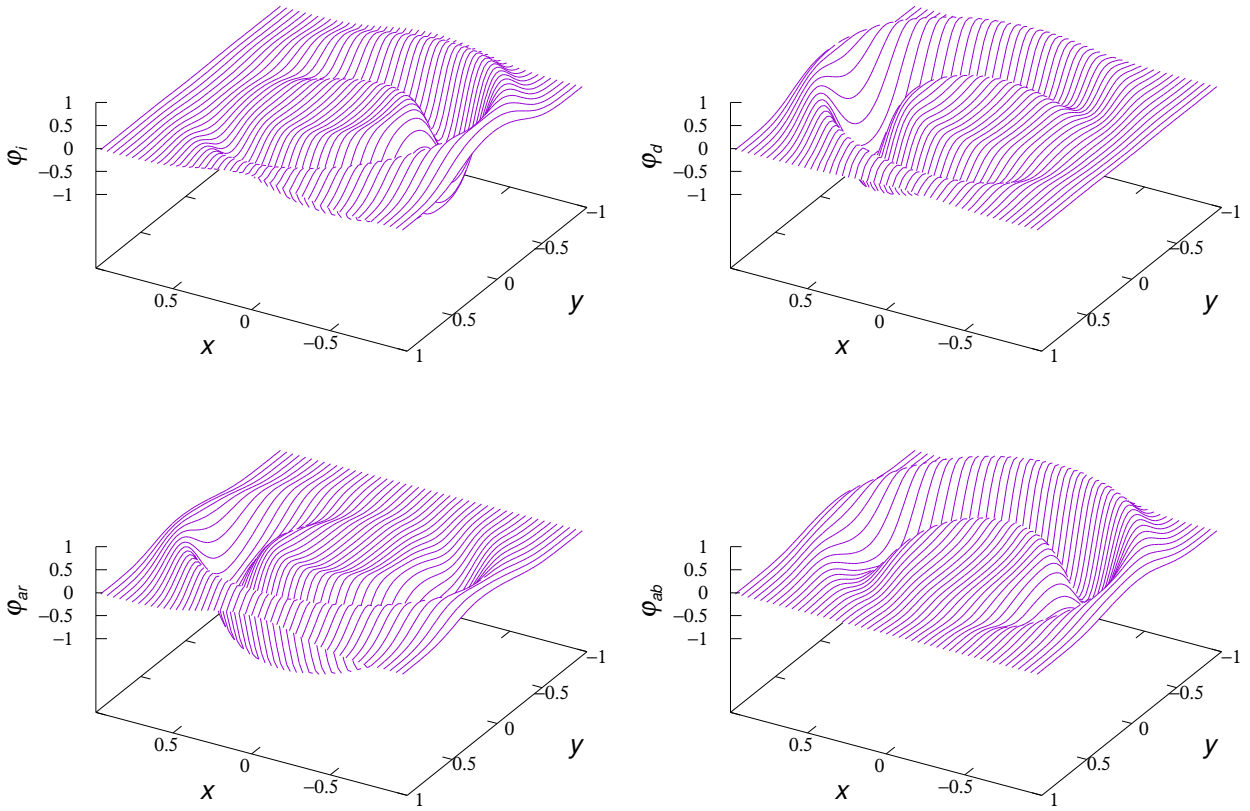


Figura 4.12. Evolución del error numérico para la solución mostrada en la figura 4.11, calculado utilizando (3.8). En la figura se puede ver la forma y la magnitud del error numérico para diferentes instantes de tiempo.

Partiendo de la evolución numérica del problema, se puede ver la forma en que la onda se desacopla en sus diferentes modos característicos de propagación, en la figura 4.13 se presentan los resultados numéricos de estos en dos dimensiones, en donde se pueden ver la onda ϕ, y los modos que se desplazan a izquierda ϕ_i, derecha ϕ_d, arriba ϕ_{ar} y abajo ϕ_{ab}, acorde con (1.18), (1.17), (1.23) y (1.22) para un mismo instante de tiempo.



(a)



(b)

Figura 4.13. Descomposición de la solución numérica de una onda propagándose en dos dimensiones en sus modos fundamentales. En (a) se muestra la onda ϕ en un instante de tiempo determinado y en (b) los modos que se propagan a izquierda ϕ_i, derecha ϕ_d, arriba ϕ_{ar} y abajo ϕ_{ab}, acorde con (1.18), (1.17), (1.23) y (1.22).

4.2.2 Malla Refinada Adaptativa (AMR). Posterior a la implementación con malla única, en las figuras 4.14 y 4.15 se puede ver el mismo problema pero ahora con un subdominio refinado, el cual se adapta a la dinámica del problema, desplazándose a la misma velocidad en la que se propaga la onda. Hay que tener en cuenta que la región en la que se encuentra el máximo de la onda es la que tiene la magnitud máxima del error numérico como se vio en la figura 4.12. El desplazamiento se realiza como se describe en (2.16) ajustando la velocidad $\vec{v_d}$ de la malla refinada (ajustando n_x y n_y) a la velocidad de propagación de la onda, que viene dada por la EDP. El factor de refinamiento es $\alpha = 3$ y se refina únicamente una región rectangular, lo cual más adelante mostrará el contraste entre el error numérico en la región que fue refinada respecto a las demás regiones.

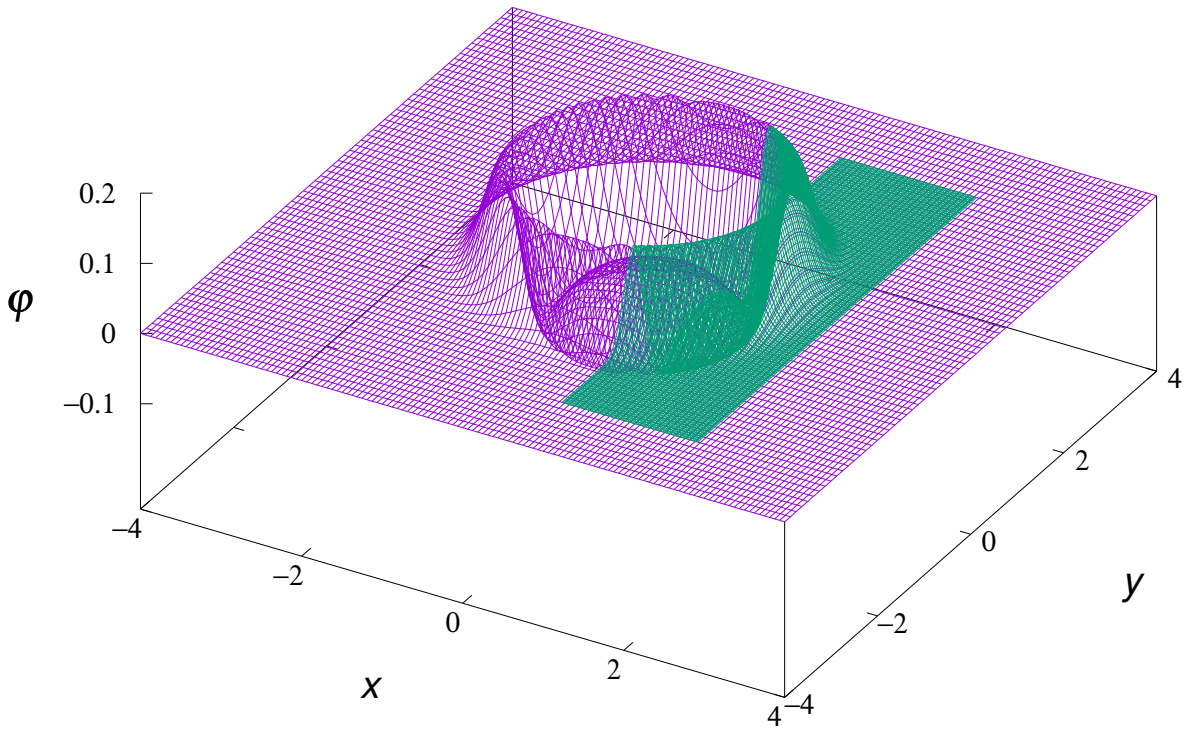


Figura 4.14. Refinamiento de un subdominio por una factor $\alpha = 3$ para la evolución numérica en dos dimensiones, la evolución completa de este se se muestra en la figura 4.15

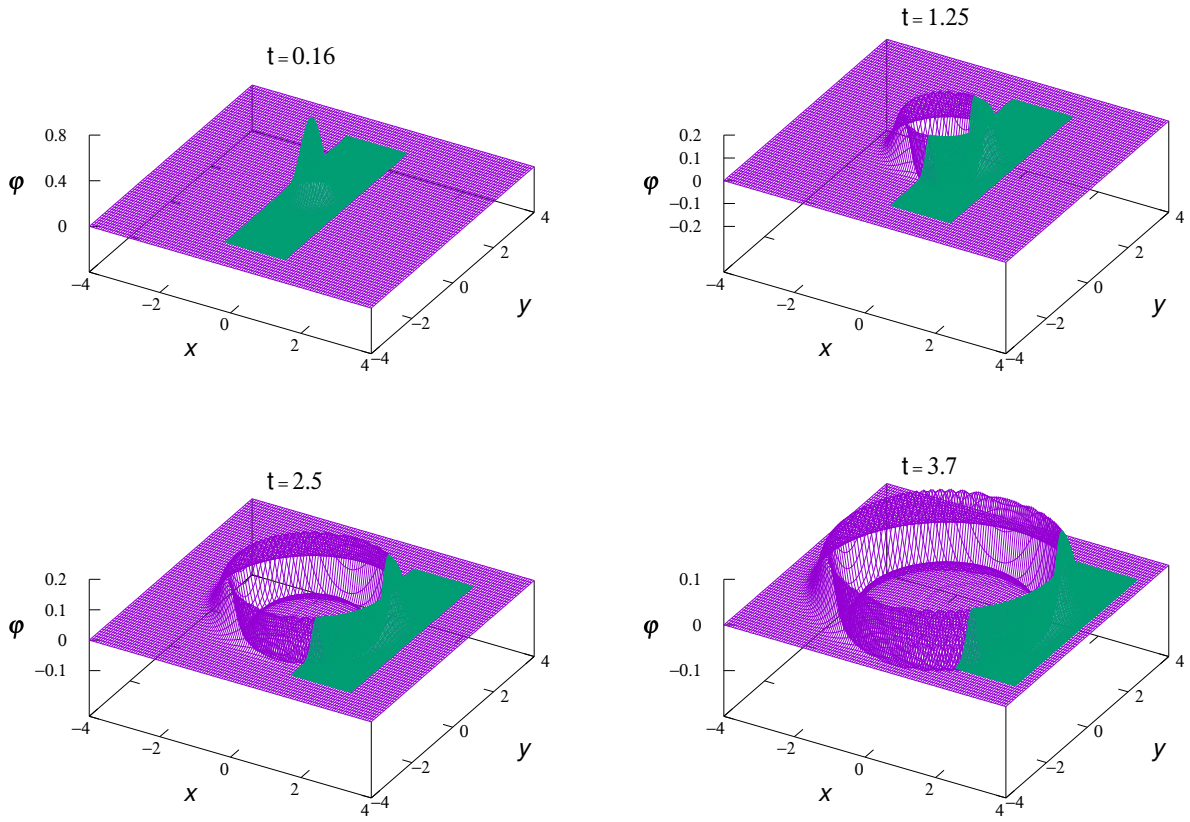


Figura 4.15. Evolución numérica de la ecuación de onda en dos dimensiones utilizando AMR. Las condiciones iniciales están dadas por (4.7) donde $A = 1$, $\sigma = 0.1$, $x_0 = 0$ y $y_0 = 0$. El dominio refinado en $t = 0$ es $[-1, 1]$ en x y $[-3, 3]$ en y, y viaja con una velocidad $v_d = 1$ en un intervalo de tiempo $[0, 4.2]$, el tamaño de paso de la malla base es $\Delta x = 0.02$, el de la malla hijo $\Delta x = 0.02/\alpha = 0.007$, $\mathbf{C} = 0.25$ y se utilizaron condiciones de frontera de onda saliente.

En las figuras 4.16 y 4.17 se presenta el error numérico correspondiente a esta evolución, siendo claro que la región que viaja con una malla refinada presenta una disminución considerable del error, en este caso dada por la nueva resolución de la malla.

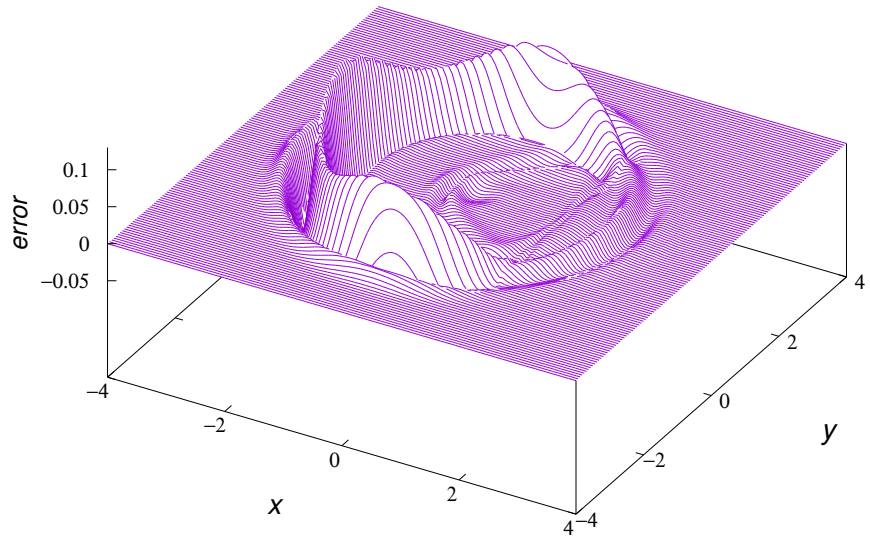


Figura 4.16. Error numérico correspondiente a la solución numérica mostrada en la figura 4.14

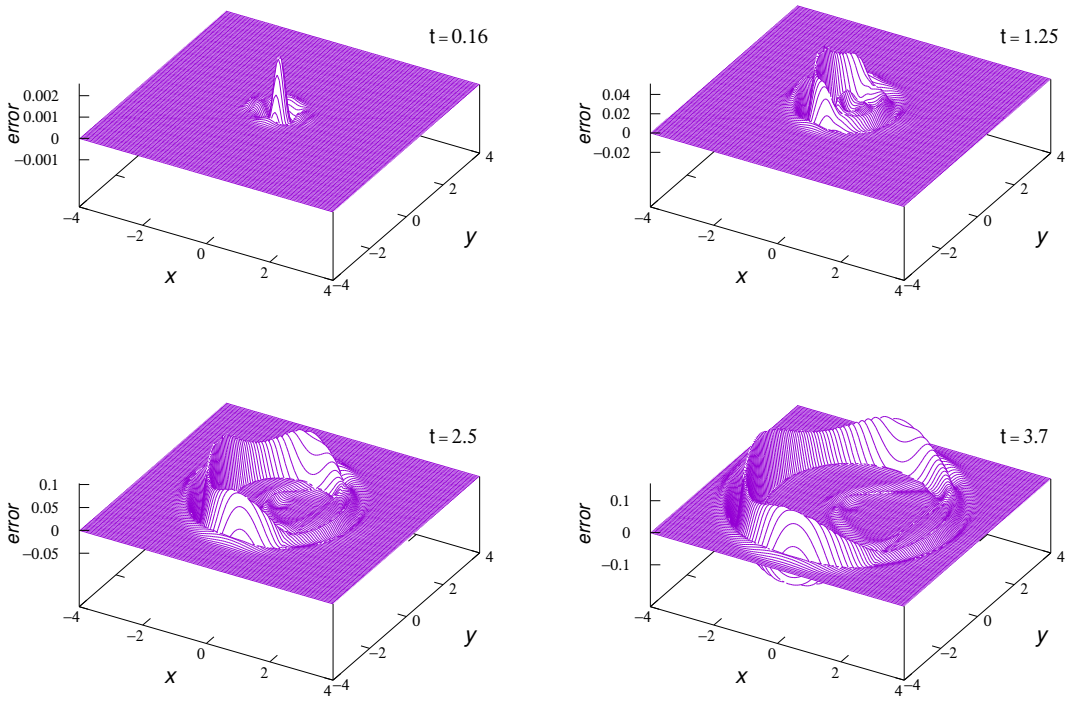
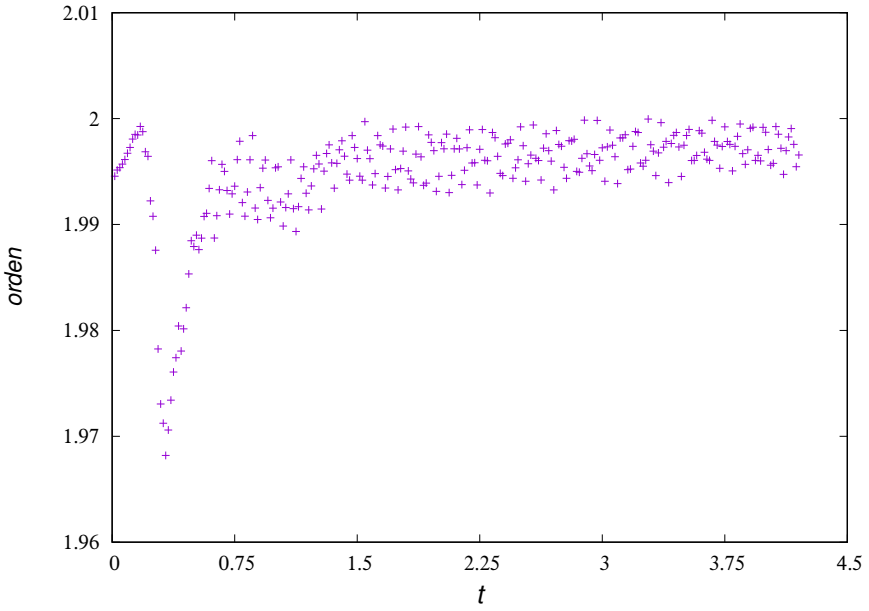
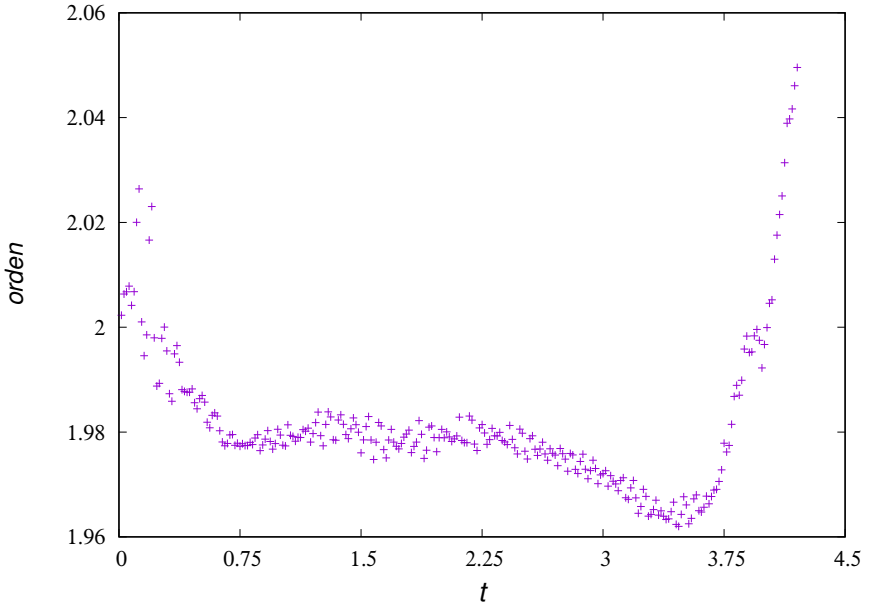


Figura 4.17. Evolución del error numérico para la solución mostrada en la figura 4.15, calculado utilizando (3.8). En la figura se puede ver la forma y la magnitud del error en diferentes instantes de tiempo.

4.2.3 Pruebas de convergencia. Como garantía de que los algoritmos implementados en dos dimensiones funcionan bien, es decir, que sus resultados numéricos son válidos, a continuación se presentan las pruebas de convergencia para ellos. En este caso no se tiene una solución exacta del problema, por lo tanto es necesario utilizar otra técnica para verificar la convergencia de la norma del error cuando $\Delta x \to 0$ y $\Delta t \to 0$. Esta técnica es la auto-convergencia y está descrita en (3.7) en donde se utilizan 3 soluciones numéricas con diferente discretización en la malla base (o el nivel l_0), es decir, la malla base para cada una de las simulaciones se discretiza con un paso diferente Δx, $\Delta x/\beta$ ó $\Delta x/\beta^2$. A partir de los resultados obtenidos con estas tres simulaciones se calcula el orden de convergencia para cada paso temporal lo cual se muestra en la figura 4.18, tanto para el algoritmo de malla única como para el AMR, es claro que no es una línea recta perfecta, pero los valores oscilan muy cerca de valor esperado, es decir dos, esto teniendo en cuenta que el algoritmo converge a segundo orden, dadas sus características.



(a)



(b)

Figura 4.18. Utilizando tres soluciones numéricas para tres resoluciones base diferentes se puede calcular el factor de convergencia para los algoritmos implementados en dos dimensiones, el calculo se realiza en cada paso de tiempo mediante (3.7). En (a) se puede ver el resultado para el algoritmo de malla única en un intervalo de tiempo y en (b) para el algoritmo AMR.

4.3. Otros problemas resueltos

4.3.1 Pulso en dos dimensiones.

Malla única. También se realizan las pruebas de los algoritmos con otras configuraciones de condiciones iniciales, en este caso una onda que viaja paralela al eje x en dos dimensiones, para la cual contamos con la solución exacta del problema, a diferencia del ejemplo en dos dimensiones mostrado en la sección 4.2. Partiendo del problema planteado en (1.13), el cual corresponde a la ecuación de onda en dos dimensiones en coordenadas cartesianas descompuesta en un sistema de EDPHs de primer orden, con condiciones iniciales dadas por,

$$\begin{aligned} u_1(x,0) &= \partial_t\phi(x,0) = -\frac{A\left(e^{-\frac{2x}{\sigma^2}}-1\right)e^{-\frac{x}{\sigma^2}}}{\sigma^2\left(e^{-\frac{2x}{\sigma^2}}+1\right)^2}, \\ u_2(x,0) &= \partial_x\phi(x,0) = -\,\partial_t\phi(x,0), \\ u_3(x,0) &= \partial_y\phi(x,0) = 0, \end{aligned} \tag{4.8}$$

con la condición auxiliar

$$\phi(x,0) = \frac{A}{e^{\frac{x}{\sigma^2}}+e^{-\frac{x}{\sigma^2}}}, \tag{4.9}$$

y condiciones de contorno,

$$\begin{aligned} u_1(x_i,y,t) &= \phi_i(x_i,y,t) \quad & u_2(x_i,y,t) &= \phi_i(x_i,y,t) \\ u_1(x_f,y,t) &= \phi_d(x_f,y,t) \quad & u_2(x_d,y,t) &= -\phi_d(x_f,y,t) \end{aligned} \tag{4.10}$$

siendo x_i, x_d, y_i y y_f las fronteras del dominio Ω. Las condiciones iniciales corresponden a la función,

$$\phi(x,t) = \frac{A}{e^{\frac{(x-t)}{\sigma^2}}+e^{-\frac{(x-t)}{\sigma^2}}}. \tag{4.11}$$

y a sus derivadas evaluadas en $t = 0$, donde A define la amplitud inicial de la onda y σ en ancho de la misma, esta función es solución exacta del problema.

La evolución numérica del problema se muestra en la figura 4.19 y la evolución del error numérico está en la figura 4.20. La onda viaja paralela al eje x con una velocidad $v = 1$. Además se puede ver que la zona que presenta el error máximo se propaga con la onda, creciendo en magnitud a medida que esta avanza.

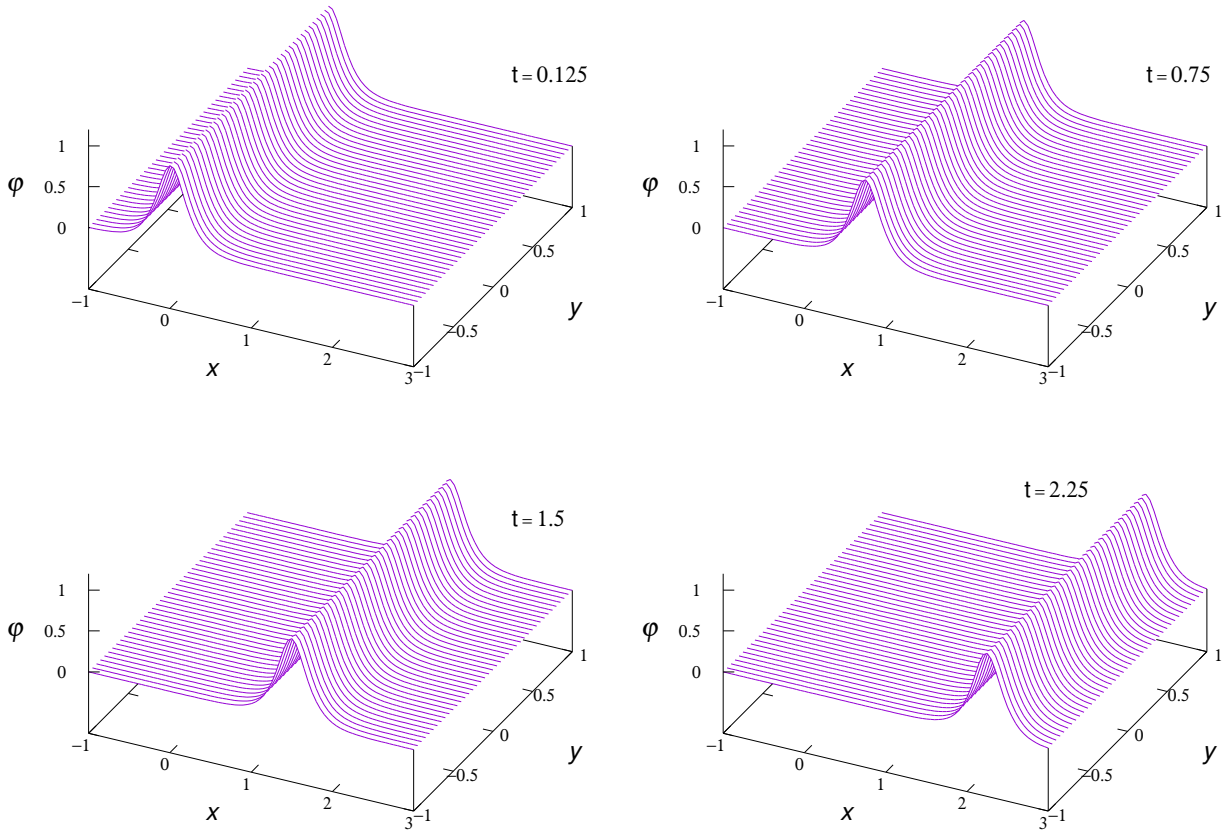


Figura 4.19. Evolución numérica de la ecuación de onda en dos dimensiones utilizando malla única. Las condiciones iniciales están dadas por (4.11), donde $A = 2$ y $\sigma = 0.3$. El dominio espacial es $[-1, 3]$ en x y $[-1, 1]$ en y, el dominio temporal es $[0, 3.25]$, el tamaño de paso es $\Delta x = 0.05$, $CFL = 0.25$ y se utilizaron condiciones de onda saliente para las fronteras. En la figura se puede ver la evolución para diferentes instantes de tiempo.

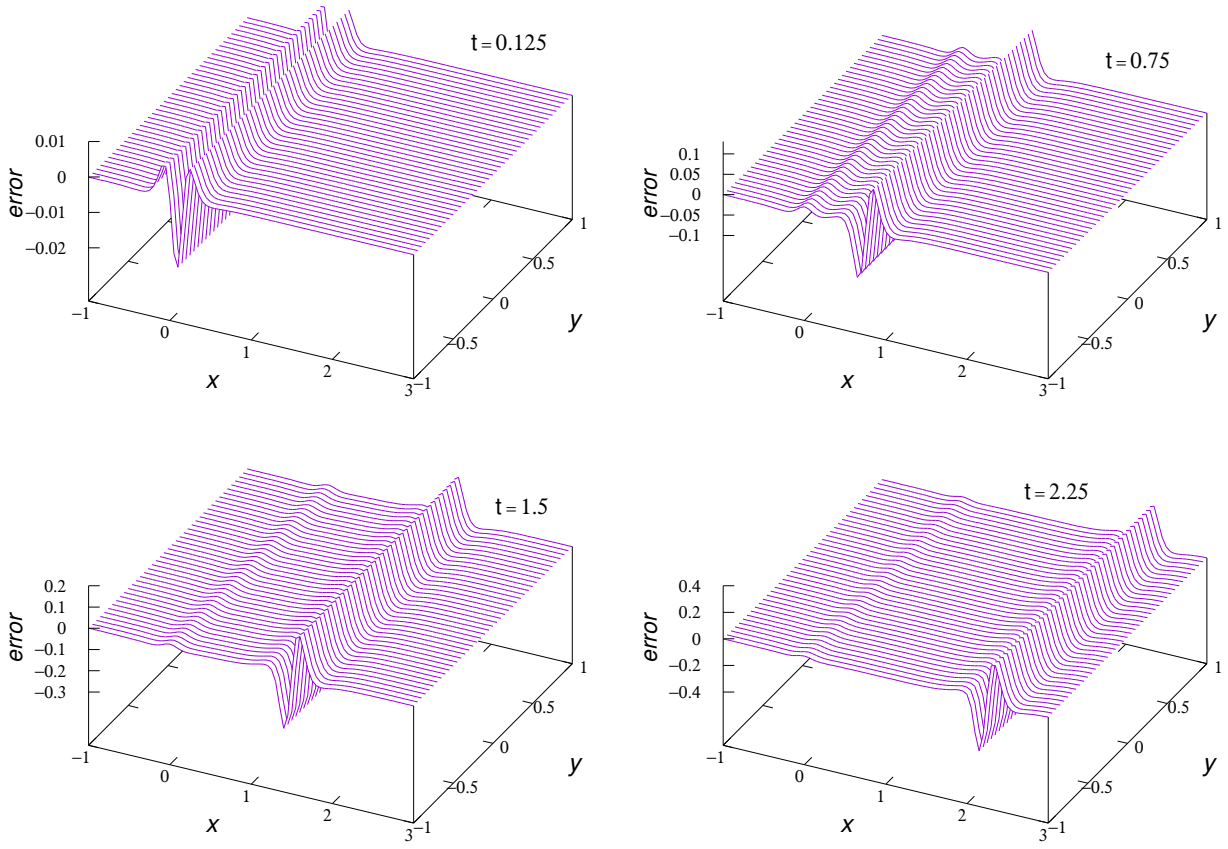


Figura 4.20. Evolución del error numérico para la solución mostrada en la figura 4.19, calculado al restar la solución exacta mostrada en (4.11) a la solución numérica. En la figura se puede ver la forma y la magnitud del error numérico para diferentes instantes de tiempo.

Malla Refinada Adaptativa (AMR). Posteriormente se procede a realizar la evolución usando AMR, es decir se implementa una malla refinada, la cual se adapta a la dinámica del problema. En las figuras 4.21 y 4.22 se puede ver el mismo problema resuelto anteriormente, en donde la malla refinada se desplaza a la misma velocidad en la que se propaga la onda. El factor de refinamiento es $\alpha = 3$ y se refina toda la región en la que se encuentra la onda, lo cual más adelante permitirá el contraste entre el error numérico utilizando malla única y AMR en la figura 4.23.

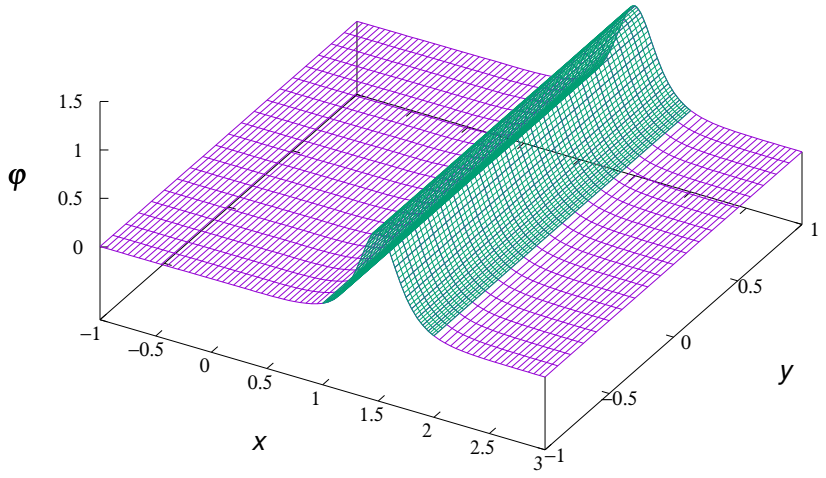


Figura 4.21. Refinamiento de un subdominio por una factor $\alpha = 3$ para una evolución numérica en dos dimensiones, la evolución completa de este se se muestra en la figura 4.23.

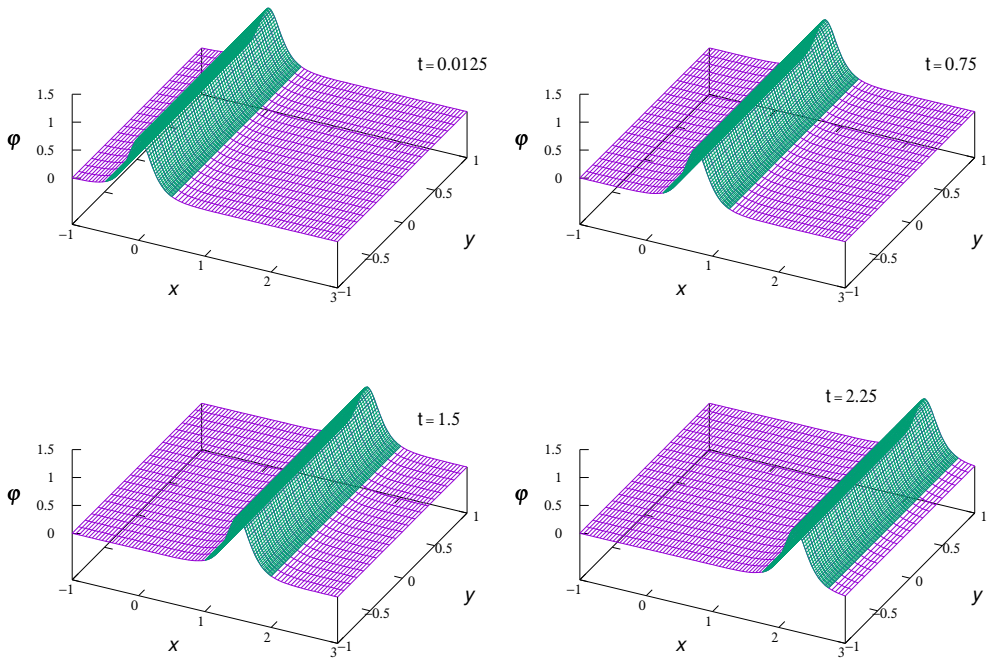
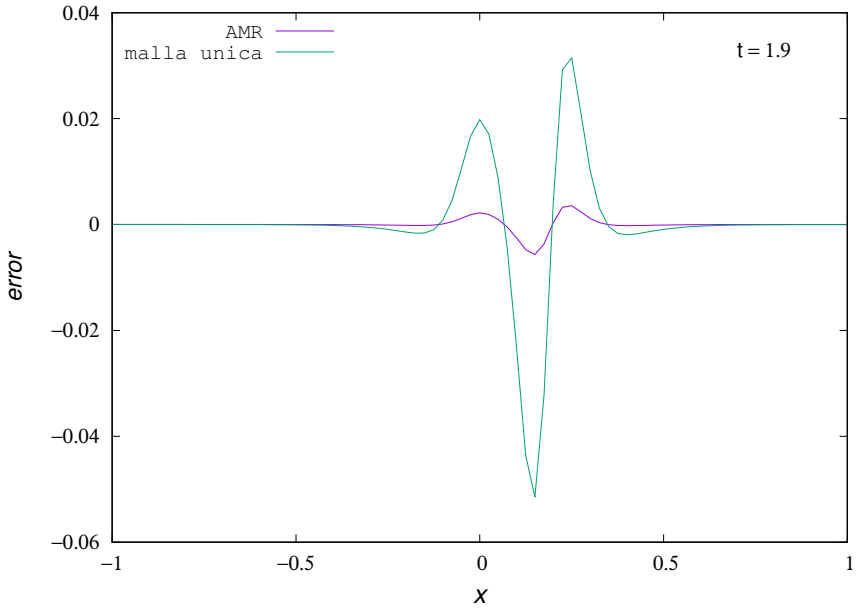
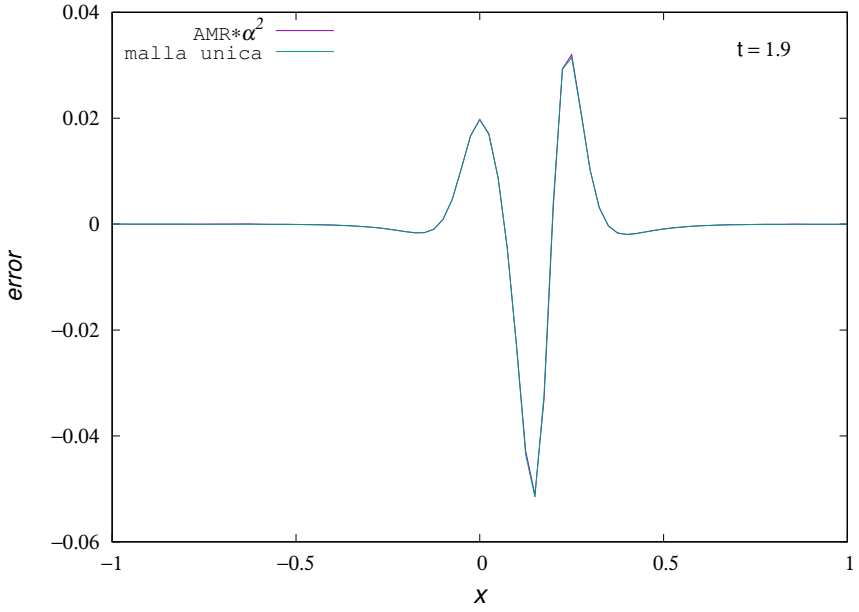


Figura 4.22. Evolución numérica de la ecuación de onda en dos dimensiones utilizando AMR. Las condiciones iniciales están dadas por (4.11), donde $A = 2$ y $\sigma = 0.3$. El dominio refinado en $t = 0$ es $[-0.8, 0.8]$ en x y $[-1, 1]$ en y, y viaja a una velocidad $v_d = 1$ en un intervalo de tiempo $[0, 2.5]$, el tamaño de paso es $\Delta x = 0.05$ para la malla base y $\Delta x = 0.05/\alpha = 0.017$ para la malla hijo, $CFL = 0.25$ y se utilizaron condiciones de onda saliente para las fronteras.



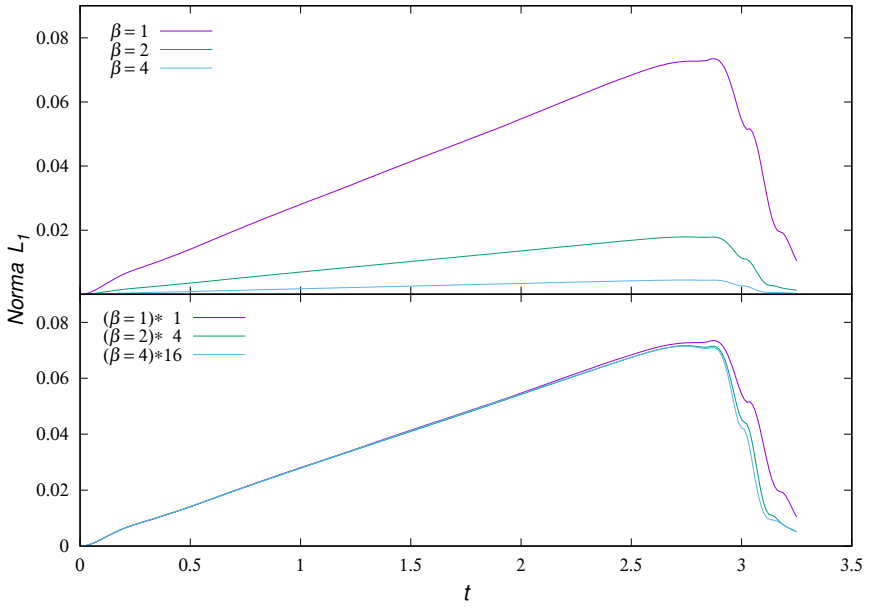
(a)



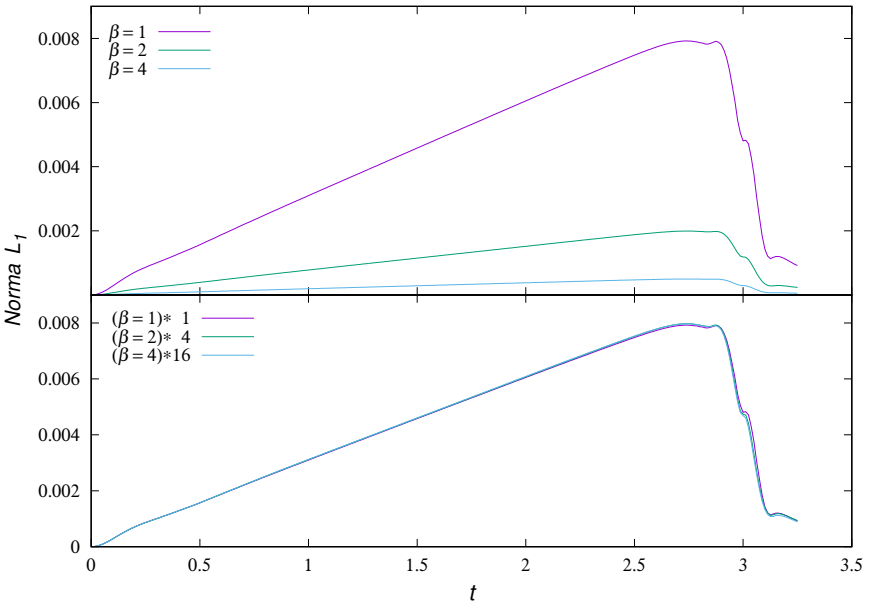
(b)

Figura 4.23. Captura de un instante de tiempo del error numérico de la onda utilizando malla única (verde) y utilizando AMR (morado), en (a) están las magnitudes originales y en (b) las mismas multiplicando las correspondiente al AMR por $\alpha^2 = 9$.

Pruebas de Convergencia. A continuación se presentan las pruebas de convergencia para este problema. En este caso se tiene una solución exacta, por lo que se puede utilizar esta para verificar la convergencia de la norma del error cuando $\Delta x \to 0$ y $\Delta t \to 0$. Se tienen los datos del error numérico en un intervalo de tiempo para simulaciones que tienen diferente tamaño de paso base (es decir el nivel l_0 tiene diferente Δx), se puede verificar la convergencia de la norma mediante (3.13), donde β sería la relación entre el tamaño de paso para las diferentes corridas, n el tipo de norma a utilizar y el código tiene orden de convergencia 2. En la figura 4.10 se presentan las gráficas para estos dos algoritmos, en este caso la norma L_1 se calcula para 3 resoluciones base, es decir, $\beta = 1$, $\beta = 2$ y $\beta = 4$ y los respectivos factores de convergencia son $\beta^2 = 1$, $\beta^2 = 4$ y $\beta^2 = 16$. En la figura 4.10(a) está la prueba correspondiente al algoritmo de malla única y en la figura 4.10(b) la correspondiente al algoritmo AMR, donde es claro que los algoritmos convergen adecuadamente acorde con la teoría. Además se puede ver que la magnitud del error global se reduce por un factor aproximado de 9 en el caso de usar AMR en relación a usar malla única.



(a)



(b)

Figura 4.24. Convergencia de la norma L_1 para los algoritmos malla única y AMR. La norma L_1 se calcula para 3 resoluciones base, en este caso la relación entre resoluciones es $\beta = 2$ (verde) y $\beta = 4$ (celeste) y los respectivos factores de convergencia son $\beta^2 = 4$ (verde) y $\beta^2 = 16$ (celeste). En (a) se presenta la prueba de convergencia para el algoritmo con malla única, en la parte de arriba se tienen las normas originales y en la de abajo las mismas multiplicadas por sus respectivos factores de convergencia, la prueba correspondiente a los algoritmos AMR se presenta en (b).

4.4. Costo Computacional

Con el objetivo de corroborar lo expuesto en la sección 2.4, se llevaron a cabo simulaciones con diferentes tamaño de paso Δx y se midieron los tiempos de cómputo de estas para una y dos dimensiones. Para las simulaciones en una dimensión se tienen los siguientes parámetros, $N_x = 1600$ y $N_t = 6400$, y estas se realizaron disminuyendo el tamaño del paso a $\Delta x/\alpha$, para diferentes valores de α. Los resultados se muestran en la tabla 4.1 y en la figura 4.25. La expresión para el tiempo de cómputo esperado (teórico) está dada por $t = 2.6 * \alpha N_t * (\alpha N_x - 1)/(N_x * N_t)$, donde $\alpha N_t * (\alpha N_x - 1)$ es el número de operaciones para la malla refinada por un factor α, $N_x * N_t$ es el número de operaciones para la malla original y $2.6s$ es el tiempo de cómputo para la malla original. El tiempo de cómputo experimental se mide al llevar a cabo las simulaciones.

α	Tiempo de Computo $[s]$
1	2.6
2	10.5
4	41.6
6	92.5
8	164.9
10	258.5
12	383.2
14	524.5
16	689.4

Tabla 4.1: Tiempos de Computo en una dimensión

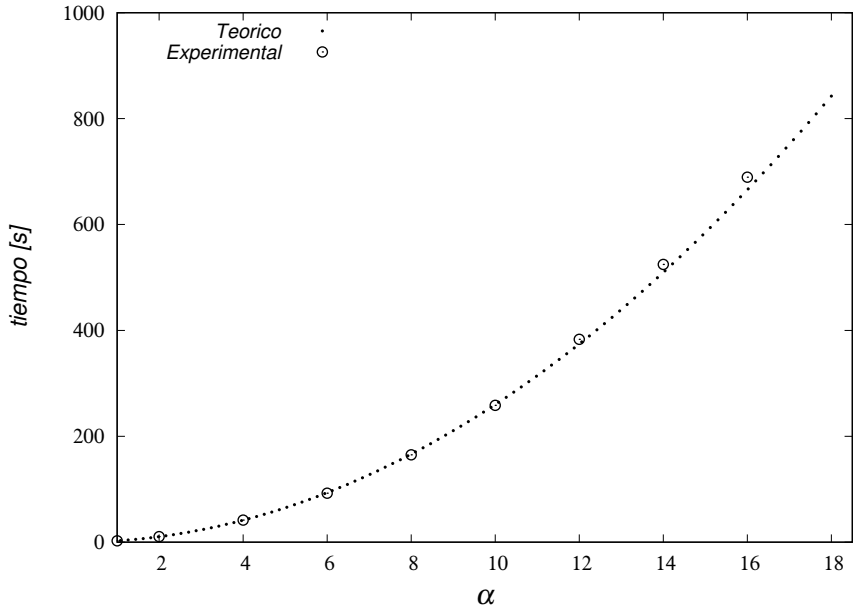


Figura 4.25. Comparación entre el tiempo de computo esperado (teórico) y el tiempo de computo obtenido (experimental) para simulaciones con diferente Δx en una dimensión.

Para el caso en dos dimensiones se tienen los parámetros $N_x = 100$, $N_y = 100$ y $N_t = 400$, y se realizaron simulaciones disminuyendo el tamaño del paso a $\Delta x/\alpha$, para diferentes valores de α. Los resultados se muestran en la tabla 4.2 y en la figura 4.26. La expresión para el tiempo de computo esperado (teórico) está dada por $t = 1.82 * \alpha N_t * (\alpha N_x - 1)(\alpha N_y - 1)/(N_x * N_y * N_t)$, donde $\alpha N_t * (\alpha N_x - 1)(\alpha N_y - 1)$ es el número de operaciones para la malla refinada por un factor α, $N_x * N_y * N_t$ es el número de operaciones para la malla original, y $1.82s$ es el tiempo de computo para la malla original. El tiempo de computo experimental se mide al llevar a cabo las simulaciones.

α	Tiempo de Computo $[s]$
1	1.82
2	13.2
4	115.1
6	390.6
8	898.8
10	1749.3
12	2992.0
14	4635.2
16	6952.0

Tabla 4.2: Tiempos de Computo en dos dimensiones

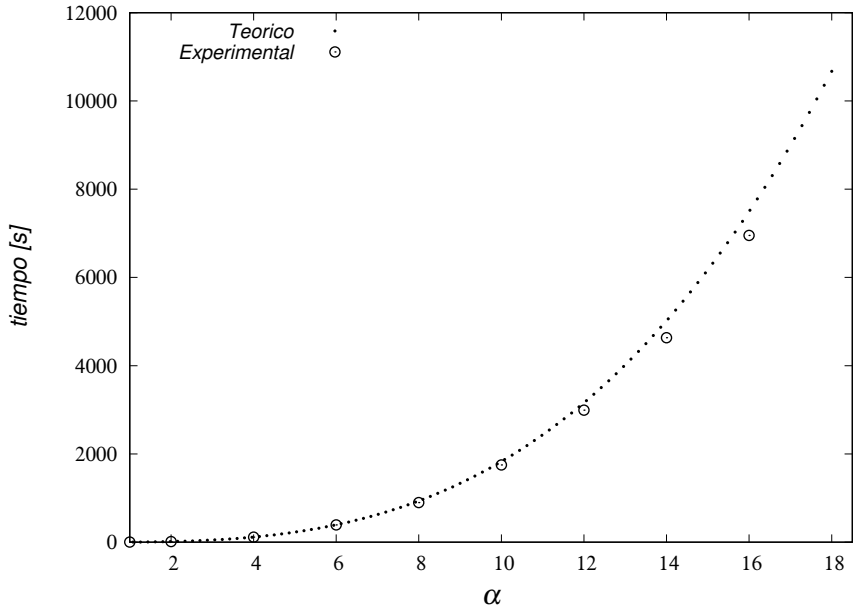


Figura 4.26. Comparación entre el tiempo de computo esperado (teórico) y el tiempo de computo obtenido (experimental) para simulaciones con diferente ancho de paso en dos dimensiones.

Es claro que las relaciones para el incremento del costo computacional para una y dos dimensiones, se cumplen con bastante precisión. Se puede deducir partiendo de estos resultados el ahorro que implica el uso de AMR al refinar únicamente ciertas regiones del dominio por un factor determinado, y además se puede calcular este trabajo como la suma del trabajo para la evolución de los diferentes dominios por separado.

Como una forma de ilustrar el ahorro de trabajo que implica el algoritmo AMR, se va a comparar el costo computacional de refinar todo el dominio por un factor $\alpha = 4$ y el de refinar únicamente el 10 % del dominio por el mismo factor, para los casos en una dos y tres dimensiones. Usando las ecuaciones descritas en la sección 2.4, se puede establecer que para el caso en una dimensión, el algoritmo con AMR realiza aproximadamente un $15,62\,\%$ del trabajo que realiza el algoritmo en el que se refina todo el dominio, para el caso en dos dimensiones esta cifra es del $11.41\,\%$ y para en tres dimensiones es $10.35\,\%$. Estos resultados fueron corroborados de forma experimental con bastante precisión para una y dos dimensiones.

Capítulo 5

Conclusiones

Se presenta el diseño, la implementación y los resultados numéricos de un algoritmo para la solución de EDPHs usando un refinamiento adaptativo, en este caso utilizando la ecuación de onda como ejemplo representativo. Se puede concluir que el método AMR resulta efectivo para reducir el costo computacional significativamente al resolver numéricamente este tipo de problemas, dando una resolución mayor únicamente en las regiones en la que se pretende reducir el error y de esta forma obteniendo resultados mejores en términos de precisión con un número de operaciones computacionales mucho menor. El hecho de refinar únicamente un sub-dominio específico funciona bien en el sentido que en ese dominio la reducción del error está asociada directamente con el nuevo tamaño del paso (Δx), y además en que los algoritmos convergen adecuadamente a orden dos (lo cual se demuestra con resultados numéricos), esto garantiza de forma clara que los resultados obtenidos a partir de estos códigos son válidos. Se pudo notar que un cambio abrupto en la resolución causa un aumento considerable en el error numérico, por lo tanto resulta conveniente refinar por niveles, y utilizando factores de refinamiento bajos (es posible profundizar en este sentido para encontrar las configuraciones óptimas). Los mismos principios en los que se basan los algoritmos para una y dos dimensiones son extensibles para el caso en tres dimensiones, por lo tanto queda pendiente esta implementación, la cual es de gran interés. Se pudo comprobar que el método resulta mucho más útil en simulaciones que involucran varias dimensiones y refinamientos altos, debido a que bajo estas condiciones el costo computacional crece mucho más rápido a medida que se refina. En el caso de la ecuación de onda el error numérico siempre es mayor en el punto máximo de los pulsos y crece a medida que estos se desplazan, por lo que en este caso las mallas adaptativas se deben mover con ellos.

Capítulo 6

Bibliografía

Abbott, B. P., R. Abbott, T. Abbott, M. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari, et al.
2016. Observation of gravitational waves from a binary black hole merger. *Physical review letters*, 116(6):061102.

Abbott, B. P., R. Abbott, T. Abbott, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari, V. Adya, et al.
2017a. Gw170814: A three-detector observation of gravitational waves from a binary black hole coalescence. *Physical review letters*, 119(14):141101.

Abbott, B. P., R. Abbott, T. Abbott, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari, V. Adya, et al.
2017b. Gw170817: observation of gravitational waves from a binary neutron star inspiral. *Physical Review Letters*, 119(16):161101.

Alfio Quarteroni, F. S.
2012. *Scientific Computing with MAT lAB*. Springer Science & Business Media.

Ascher, U. M. and L. R. Petzold
1998. *Computer methods for ordinary differential equations and differential-algebraic equations*, volume 61. Siam.

Bell, J., M. Berger, J. Saltzman, and M. Welcome
1994. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM Journal on Scientific Computing*, 15(1):127–138.

Berger, M. J. and P. Colella
1989. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84.

Berger, M. J. and J. Oliger
1984. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of computational Physics*, 53(3):484–512.

Brügmann, B.
1999. Binary black hole mergers in 3d numerical relativity. *International Journal of Modern Physics D*, 8(01):85–100.

Courant, R., K. Friedrichs, and H. Lewy
1967. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234.

Cruz-Osorio, A. and F. Lora-Clavijo
2016. Non-axisymmetric relativistic wind accretion with velocity gradients on to a rotating black hole. *Monthly Notices of the Royal Astronomical Society*, 460(3):3193–3201.

Cruz-Osorio, A., F. Lora-Clavijo, and F. Guzmán
2012. Is the flip-flop behaviour of accretion shock cones on to black holes an effect of coordinates? *Monthly Notices of the Royal Astronomical Society*, 426(1):732–738.

Cruz-Osorio, A., F. Sánchez-Salcedo, and F. Lora-Clavijo
2017. Relativistic bondi–hoyle–lyttleton accretion in the presence of small rigid bodies around a black hole. *Monthly Notices of the Royal Astronomical Society*, 471(3):3127–3134.

Debreu, L., C. Vouland, and E. Blayo
2008. Agrif: Adaptive grid refinement in fortran. *Computers & Geosciences*, 34(1):8–13.

Evans, L. C.
2010. *Partial Differential Equations*. American Mathematical Society; 2 edition.

Fryxell, B., K. Olson, P. Ricker, F. Timmes, M. Zingale, D. Lamb, P. MacNeice, R. Rosner, J. Truran, and H. Tufo
2000. Flash: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *The Astrophysical Journal Supplement Series*, 131(1):273.

Guzmán, F.
2010. Solución de la ecuación de onda como un problema de valores iniciales usando diferencias finitas. *Revista mexicana de física E*, 56(1):51–68.

Huang, W. and R. D. Russell
2010. *Adaptive moving mesh methods*, volume 174. Springer Science & Business Media.

Kreiss, H.-O. and G. Scherer
1992. Method of lines for hyperbolic differential equations. *SIAM Journal on Numerical Analysis*, 29(3):640–646.

Löffler, F., J. Faber, E. Bentivegna, T. Bode, P. Diener, R. Haas, I. Hinder, B. C. Mundim, C. D. Ott, E. Schnetter, et al.
2012. The einstein toolkit: a community computational infrastructure for relativistic astrophysics. *Classical and Quantum Gravity*, 29(11):115001.

Lora-Clavijo, F., A. Cruz-Osorio, and F. Guzmán
2015a. Cafe: a new relativistic mhd code. *The Astrophysical Journal Supplement Series*, 218(2):24.

Lora-Clavijo, F., A. Cruz-Osorio, and E. M. Méndez
2015b. Relativistic bondi–hoyle–lyttleton accretion onto a rotating black hole: density gradients. *The Astrophysical Journal Supplement Series*, 219(2):30.

Lora-Clavijo, F., M. Gracia-Linares, and F. Guzmán
2014. Horizon growth of supermassive black hole seeds fed with collisional dark matter. *Monthly Notices of the Royal Astronomical Society*, 443(3):2242–2251.

Lora-Clavijo, F. and F. Guzmán
2013. Axisymmetric bondi–hoyle accretion on to a schwarzschild black hole: shock cone vibrations. *Monthly Notices of the Royal Astronomical Society*, 429(4):3144–3154.

Lora-Clavijo, F., F. Guzmán, and A. Cruz-Osorio
2013. Pbh mass growth through radial accretion during the radiation dominated era. *Journal of Cosmology and Astroparticle Physics*, 2013(12):015.

Nakamura, S.
1992. *Metódos numericos aplicados con software.* Prentice-Hall Hispanoamericana.

Navarro, A., F. Lora-Clavijo, and G. A. González
2017. Magnus: A new resistive mhd code with heat flow terms. *The Astrophysical Journal*, 844(1):57.

Olver, P. J.
2016. *Introduction to partial differential equations*. Springer.

Pretorius, F. and M. W. Choptuik
2006. Adaptive mesh refinement for coupled elliptic-hyperbolic systems. *Journal of Computational Physics*, 218(1):246–274.

Schnetter, E., S. H. Hawley, and I. Hawke
2004. Evolutions in 3d numerical relativity using fixed mesh refinement. *Classical and quantum gravity*, 21(6):1465.

Teyssier, R.
2002. Cosmological hydrodynamics with adaptive mesh refinement-a new high resolution code called ramses. *Astronomy & Astrophysics*, 385(1):337–364.

Thierfelder, M., S. Bernuzzi, and B. Bruegmann
2011. Numerical relativity simulations of binary neutron stars. *Physical Review D*, 84(4):044012.

Thomas, J. W.
2013. *Numerical partial differential equations: finite difference methods*, volume 22. Springer Science & Business Media.

Toro, E. F.
2013. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media.

Apéndice A

Integración numérica

En el marco de este algoritmo se utiliza la integración numérica para realizar la evolución temporal de cada una de las variables, para tal fin el método escogido fue el **Runge Kutta 4** (RK4), dado que presenta características adecuadas en términos de convergencia y estabilidad. Este método evoluciona una variable a partir de la función para la derivada de esta y un dato inicial, como se muestra a continuación. Si se tiene un problema de valor inicial

$$\phi_i' = g(t_i, \phi_i), \quad \phi(t_0) = \phi_0, \tag{A.1}$$

el método RK4 para este problema está dado por la siguiente ecuación

$$\phi_{i+1} = \phi_i + \frac{1}{6}\Delta t(k_1 + 2k_2 + 2k_3 + k_4), \tag{A.2}$$

donde

$$\begin{aligned} k_1 &= g\left(t_i, \phi_i\right), \\ k_2 &= g\left(t_i + \frac{1}{2}\Delta t, \phi_i + \frac{1}{2}k_1\Delta t\right), \\ k_3 &= g\left(t_i + \frac{1}{2}\Delta t, \phi_i + \frac{1}{2}k_2\Delta t\right), \\ k_4 &= g\left(t_i + \Delta t, \phi_i + k_3\Delta t\right). \end{aligned} \tag{A.3}$$

Así, el valor ϕ_{i+1} es determinado por el presente valor ϕ_i mas el producto del tamaño del intervalo Δt por una pendiente estimada. Esta pendiente es un promedio ponderado de diferentes pendientes, donde k_1 es la pendiente al principio del intervalo, k_2 es la pendiente en el punto medio (usando k_1 para determinar el valor de g en el punto $t_i + \Delta t/2$ por el Método de Euler), k_3 es otra vez la pendiente del punto medio (usando k_2 para determinar el valor de g) y k_4 es la pendiente al final del intervalo (con el valor de g determinado por k_3). Promediando estas cuatro pendientes, se le asigna mayor peso a las pendientes en el punto medio

$$pendiente = \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}. \tag{A.4}$$

No obstante, es bueno aclarar que existen diferentes métodos cada uno con una precisión específica y un costo computacional asociado; para una documentación más amplia consultar los trabajos de Nakamura (1992) y Ascher and Petzold (1998).