UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS
"MAT. LUIS MANUEL RIVERA GUTIÉRREZ "

PLANTILLA PARALELIZADA CON MPI PARA LA SOLUCIÓN DE PROBLEMAS DE VALORES INICIALES EN 3D

T E S I S

QUE PARA OBTENER EL GRADO DE:

LICENCIADO EN CIENCIAS FÍSICO MATEMÁTICAS

PRESENTA:

MIGUEL ANGEL CEJA MORALES

DIRECTOR DE TESIS:

DR. FRANCISCO S. GUZMÁN MURILLO

MORELIA MICHOACÁN AGOSTO DE 2017

A mis padres
mis hermanos
mi esposa e hijo
lo mejor de mi vida.

AGRADECIMIENTOS

La elaboración de esta tesis ha sido un tanto complicada, dada a una variedad de adversidades que se presentaban, sin embargo, al final se logró la meta y es para mi un honor darles, en este espacio, algo del agradecimiento que se merecen.

Debo agradecer de manera muy especial y sincera al Dr. Francisco S. Guzmán Murillo, por su apoyo, confianza y por darme un ejemplo académico a seguir. Le agradezco también el haberme facilitado siempre los medios necesarios para llevar a cabo el desarrollo de muchos otros trabajos y en especial el de esta tesis.

Debo agradecer también a Juan, Miguel y Francisco el tiempo y dedicación que se tomaron al revisar mi trabajo y en las ideas que me dieron a lo largo de su desarrollo. A Vene, Sofy e Itza por el tiempo que compartimos juntos resolviendo dudas de los códigos y de tareas.

A mis sinodales, Dr. Hector Igor Peréz, Dr. Francisco D. Mota, Dr. Petr Zhevandrov, Dra. Karina Figueroa, los cuales a pesar del tiempo se tomaron la molestia de revisar mi trabajo.

A los profesores que fueron muy importantes en mi desarrollo durante el transcurso de la carrera, Dr. Francisco S. Guzmán, Dr. Hector Igor Peréz, Dr. Mauricio Ortiz, Dr. Jose Antonio Gonzalez, Dr. Abdón E. Choque, Dr. Rigoberto Vera, Dr. Jorge Luis Lṕez

Quiero agradecer también a Misha, Gaby, Itza y lalito, mis mejores amigos, que siempre estuvieron a mi lado desde un principio y hasta el final, brindándome apoyo tanto en académico como personal, con los cuales compartí momentos inolvidables (los quiero mucho). A Karen, que siempre estuvo ahí como un ejemplo a seguir. A Chucho, que siempre me estaba ayudando, aconsejando y asesorando cuando tenía dudas. A Trino, Jorch, Esteban, Lety, Karlita, Juanito, Iza, Rubén y los Hawaiianus, con lo cuales también compartí muchos buenos momentos.

A mis wonejos: Vale, Mire, Alfred, Danny y Rose, con los cuales últimamente hemos pasado muchos momentos agradables.

Y sobre todo, agradecerle a mi familia, que siempre me ha apoyado en mis decisiones, mis padres, Maria Luisa Morales y Enrique Ceja Martínez, sin los cuales no hubiera sido capaz de cumplir esta meta. Mi hermano, Oscar el adoptado, que siempre estuvo apoyándome a pesar de la distancia y desacuerdos. Mi hermana, Araceli, la cual también siempre me ha apoyado junto a mi cuñis Maricela, las cuales son un ejemplo de que si se puede. Mi sue-

gra Felisa García y mis cuñados, Carlitos, Pedrito, Toñito e Irene, por estar con mi esposa y mi hijo cuando yo no podía hacerlo.

Y lo mejor para el final... para mi esposa, Maria Gabriela Simón García (mi princesa) y mi hijo Miguel Gabriel Ceja Simón (el bebé wonejo), sobran palabras para expresar lo infinitamente agradecido que estoy con ustedes, a pesar de tantos momentos (buenos y malos), se logró el objetivo, el cual es uno de muchos que viviremos juntos, gracias por apoyarme y estar conmigo siempre. Les pido una disculpa por los malos momentos que les he hecho pasar y el tiempo perdido que se ha ido. Sin embargo, este es el objetivo que hacia falta y ya está cumplido, ahora sí, a darle con todo a lo que sigue (los amo).

RESUMEN

En este trabajo se presentan los conceptos necesarios para resolver problemas de valores iniciales usando métodos numéricos y programación en paralelo. Para ilustrar esto, se resuelve la ecuación de onda en una y en tres dimensiones numéricamente, se emplean diferencias finitas, el método de líneas y un integrador Runge-Kutta de tercer orden, mediante el uso de la biblioteca Interfaz de Paso de Mensaje (MPI).

Se realizan pruebas de escalamiento, comparando el tiempo de ejecución durante el cálculo de la solución numérica de la ecuación de onda, en función del número de procesadores, verificando la convergencia de las soluciones.

Palabras clave: Programación en paralelo, MPI, ecuación de onda, solución numérica de ecuaciones diferenciales parciales, problemas de valores iniciales.

ABSTRACT

In this work are show the necessary concepts to solve initial value problems using numerical methods and parallel computing. For this, the one-dimensional wave equation and three-dimensional wave equation are solved numerically using finite differences, the method of lines and a Runge-Kutta of third order integrator by using the library Message Passing Interface (MPI).

Scaling tests are performed to compare the execution time during the numerical calculations for the wave equation as a function of the number of processors, verifyng the solution convergence.

Keywords: Parallel computing, MPI, wave equation, numerical solutions of partial differential equations, initial value problems.

Índice general

Capítulo 1

Introducción

Un problema de valor inicial (PVI) es una ecuación diferencial parcial junto con uno o varios valor especificos, llamados condiciones iniciales. Los problemas de valores iniciales (PVIs) son fundamentales en la física, ya que modelan fenómemos de la naturaleza. Entre los más destacables, están las ecuaciones de Einstein, las ecuaciones de Euler, las ecuaciones de Navier-Stokes, las ecuaciones de Maxwell, la Ecuación de Schroödinger, entre otras. Cuando se trabaja con este tipo de problemas matemáticos o físicos no siempre es posible encontrar una solución exacta. Para tales casos, se requiere la ayuda de una computadora, de modo que se requieren aplicaciones o programas computacionales tales que, ayuden a facilitar o aminorar el trabajo. Sin embargo, a pesar del increible trabajo que desempeñan las computadoras, existen problemas computacionales prohibitivos, problemas que se vuelven increiblemente costosos a la hora de calcular una solución, es decir, para poder resolver el problema se necesita una gran cantidad de cálculos y de recursos computacionales, con los cuales una computadora convencional no cuenta.

En esta tesis se resolverán PVIs, los cuales también pueden volverse problemas prohibitivos. Un caso más específico, que se verá más adelante, es la solución de la ecuación de onda en tres dimensiones, ya que para resolverla de forma numérica, es necesario tener muchos números de doble precisión, donde cada número ocupa una memoria de 64 bytes. Para estimar cuántos números de doble precisión se necesitan para calcular la solución de la ecuación de onda, se debe de tener presente lo siguiente: 3 coordenadas espaciales, 5 variables de evolución, 2 niveles de tiempo y el número de puntos de la partición del dominio N^3, ya que es un dominio tridimensional. Por tanto, se necesitan $(3 \times 5 \times 2 \times N^3 = 30N^3)$ números de doble precisión los cuales ocupan $30N^3 \times 64\ bytes = 1920N^3 bytes$ de memoria RAM. Si se trabaja con una discretización de tal modo que $N = 100$, se tienen $1{,}92 \times 10^9 = 1{,}92Gb$, cuya memoria aún puede ser alojada en una computadora convencional. Sin embargo, para un problema más grande, con $N = 256$, se requiere $32, 212, 254, 720\ bytes \approx 32Gb$, cuya memoria ya no puede ser alojada en una computadora convencional. Para tratar con este tipo de problemas, existe una solución que es llamada cómputo de alto rendimiento, ya sea trabajando con clusters de computadoras o con una supercomputadora

mediante la programación en paralelo.

Una supercomputadora es básicamente una computadora mucho más grande, que cuenta con más recursos, como más procesadores y más memoria. Mientras que un cluster de computadoras es un conjunto de computadoras conectadas entre sí por tarjetas de red, que permiten contar con más recursos, como más memoria y más procesadores [1]. Cada procesador realiza una parte del trabajo de cómputo.

Una característica del cómputo de alto rendimiento, es la forma de comunicación. Por un lado, las computadoras trabajan con memoria compartida, es decir, tienen una memoria accesible para todos los procesadores de esa computadora como lo muestra la Fig. 1.1. Por otro lado, un cluster tiene memoria distribuida, como lo muestra la Fig. 1.2, donde cada máquina cuenta con su propia memoria, las cuales a su vez tienen memoria compartida. Estas se unen por medio de tarjetas de red, teniendo así acceso a más procesadores y más memoria [2].

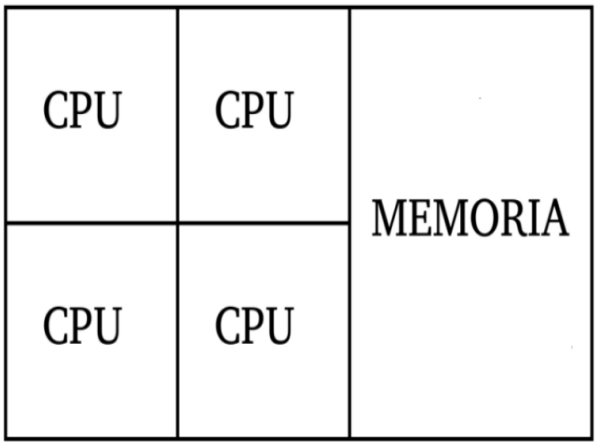


Figura 1.1: Esquema del concepto de memoria compartida, una computadora cuenta con una memoria y cada procesador tiene acceso a ella.

Para trabajar con supercomputadoras o clusters de computadoras, se emplea la programación en paralelo. Este tipo de programación es una forma de cómputo que trabaja con una serie de instrucciones que son repartidas entre varios procesadores, las cuales pueden ser resueltas simultáneamente.

En esta tesis se muestra una forma de resolver PVIs trabajando con una computadora más grande, usando su memoria compartida mediante la programación en paralelo. A modo ilustrativo, los PVIs que se resuelve numéricamente es la ecuación de onda en una dimensión y la ecuación de onda en tres dimensiones.

Se optó por usar la ecuación de onda como paradigma para resolver PVIs, ya que el estudio de las ondas constituye uno de los temas más importantes en el área de la Física, como por ejemplo, las ondas sonoras, las ondas de la luz, las ondas generadas sobre el agua, ondas gravitacionales y la función de onda en la mecánica cuántica; todas estas se

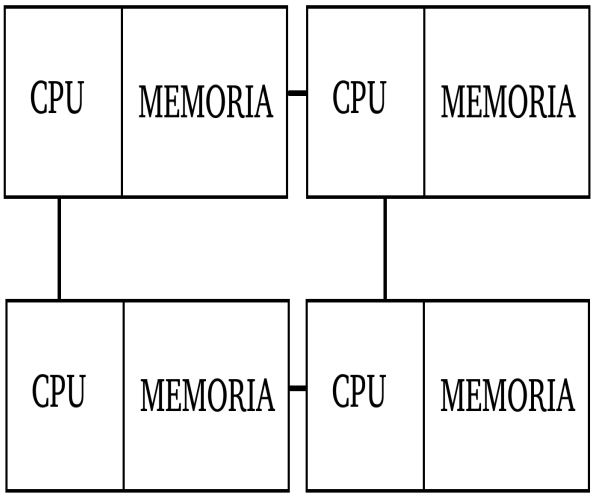


Figura 1.2: Desarrollo de la memoria distribuida para un cluster de computadoras con 4 CPU's. Cada una cuenta con su propia memoria compartida.

rigen por medio de ecuaciones diferenciales parciales (EDP's) dependientes del tiempo y por ello, para resolverlas aquí, se emplea el método de líneas, junto con el método de diferencias finitas, que constituyen métodos estándar en la solución de PVIs.

Los PVI se definen en un dominio de espacio y tiempo, por ejemplo en una dimensión el dominio espacial es un segmento de la recta real y en tres dimensiones el dominio puede ser un paralelepípedo. La paralelización para este tipo de problemas es directa, simplemente los problemas en una dimensión se resuelven dividiendo el dominio espacial en subsegmentos de la recta real, en los que cada procesador resolverá una parte del problema. En los problemas de tres dimensiones el dominio podría segmentarse en (por ejemplo) octantes y usar ocho procesadores, cada uno de ellos resolviendo una parte del PVI.

Además de obtener la solución numérica de dichas ecuaciones en forma paralela, para validar los resultados se realizan pruebas de convergencia. Las pruebas de convergencia indican si la solución numérica converge a la solución exacta, esto se observa al aumentar la resolución del dominio discreto.

Otra cosa que se desea al resolver estos PVIs usando cómputo en paralelo, es ilustrar y comparar la eficiencia que se tiene al trabajar desde 1, hasta 12 procesadores. Para ello, es de suma importancia la elaboración de pruebas de escalamiento, las cuales comparan el tiempo de ejecución de la solución, de la ecuación de onda, en función del número de procesadores. Es preciso mencionar que se trabaja con hasta 12 procesadores, ya que esta es la capacidad del equipo de cómputo con el que se desarrolló este trabajo.

Capítulo 2

Métodos Numéricos

Los métodos numéricos ofrecen alternativas para los cálculos complicados, son herramientas que permiten obtener una solución aproximada a problemas matemáticos o físicos mediante operaciones lógicas y matemáticas básicas [3]. Al tratarse de soluciones aproximadas, es preciso cuantificar los errores cometidos en dichas aproximaciones, los cuales a su vez son capaces de indicar qué tan precisa es la solución, para ello, se estudia la convergencia.

El problema computacional de esta tesis está centrado en la solución de PVIs dependientes del tiempo, más concretamente, el problema se centra en resolver la ecuación de onda en una dimensión para ilustrar los métodos numéricos y la paralelización, y en resolver la ecuación de onda en tres dimensiones en los que la memoria RAM es ahora un factor importante. Para resolver estas ecuaciones de forma numérica, se explotará el uso de los métodos numéricos y de la programación en paralelo. Además, es posible calcular la solución exacta de cada una de ellas y poder así desarrollar pruebas de convergencia.

Para resolver una ecuación diferencial parcial (EDP) de forma numérica se define un dominio discreto, que sea subconjunto del dominio donde se define la EDP. Para trabajar de forma paralelizada, se parte dicho dominio en dominios más pequeños que resuelve cada procesador. Esto se realiza con la finalidad de resolver problemas prohibitivos, es decir problemas que no pueden ser resueltos de forma serial porque necesitan mucha memoria RAM, o problemas que realizan muchos cálculos y requieren reducir el tiempo de trabajo. Se procede usando diferencias finitas, el método de líneas y un integrador Runge-Kutta de tercer orden.

2.1. Diferencias finitas

La solución de una ecuación diferencial parcial usando aproximaciones en diferencias finitas (DF) consiste en definir una versión discreta de la EDP y en calcular una solución sobre cada uno de los puntos del dominio discreto que esté contenido en el dominio continuo

de la EDP [4].

Supóngase que se tiene una EDP para la función f en un dominio discreto, que es un subconjunto del dominio donde se busca la solución de la EDP. Suponiendo que dicho dominio tiene dos coordenadas t y x (el tiempo y una coordenada espacial) $t \in [t^0, t^{N_t}]$, $x \in [x_0, x_{N_x}]$, la manera de discretizar la EDP consiste en considerar que las variables y funciones de la EDP están definidas en un conjunto de puntos en el tiempo y en el espacio dados por $t^n = n\Delta t$ y $x_j = j\Delta x$, donde $n = 0, \cdots N_t$ y $j = 0, \cdots, N_x$ son números enteros que etiquetan los puntos donde se define la ecuación discretizada. De este modo, la función f queda bien definida solamente en los puntos del dominio discreto (t^n, x_j), con $n = 0, \cdots N_t$ y $j = 0, \cdots, N_x$, y se denotan esos valores de la función por f_j^n. De modo que los valores t^0 y t^{N_t} corresponden a los valores que delimitan el dominio temporal de t, asimismo x_0 y x_{N_x} delimitan el dominio espacial de x.

Este procedimiento de discretización define una malla en un dominio similar al de la Fig. 2.1, en cuyos nodos (puntos de la malla) quedan definidas las funciones de la EDP. La manera más simple de definir una malla es cuando los nodos se encuentran igualmente espaciados [4]. Esto es, para la parte temporal $\Delta t = (t^{N_t} - t^0)/N_t$ y para la parte espacial $\Delta x = (x_{N_x} - x_0)/N_x$.

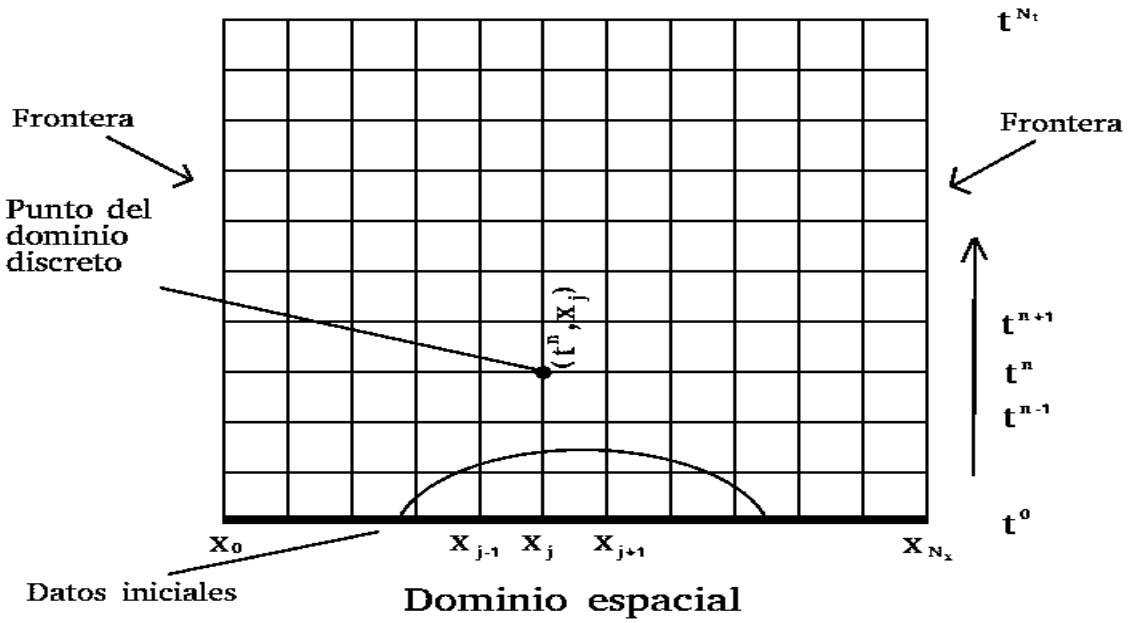


Figura 2.1: Dominio discreto de un PVI con condiciones iniciales y condiciones en la frontera. La versión discreta de la EDP queda definida solamente en los nodos de esta malla [4].

Una vez que es definido el dominio discreto de la EDP, es necesario aproximar la ecuación diferencial original mediante operadores diferenciales, para hacer esto se usan expansiones en serie de Taylor.

Para el caso continuo: Sea f una función analítica en el campo real, es posible calcular una expansión en serie de Taylor en torno a un punto x_0 de la siguiente manera:

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k, \tag{2.1}$$

donde $k = 0, 1, 2, \cdots$, denota el índice de la k-ésima derivada de la función f con respecto de x. Si ahora se trunca esta serie para $k = 3$ y se define $\Delta x = (x - x_0)$, se tiene:

$$f(x) = f(x_0) + f^{(1)}(x_0)\Delta x + f^{(2)}(x_0)\frac{\Delta x^2}{2} + O(\Delta x^3), \tag{2.2}$$

donde $O(\Delta x^3)$, es un término que denota el orden del error de truncamiento.

Para el caso discreto: Sea f_j^n una función definida en el dominio discreto definido anteriormente. Para este caso se mantendrá fija la etiqueta temporal n. Es posible calcular valores aproximados de la función en los puntos adyacentes a x_j a partir de una expansión en serie de Taylor en torno al punto x_j, con $f_j = f(x_j)$, de la siguiente manera usando (2.2):

$$f(x_{j-1}) = f(x_j) - f'(x_j)\Delta x + f''(x_j)\frac{\Delta x^2}{2} - O(\Delta x^3), \tag{2.3}$$

$$f(x_{j+1}) = f(x_j) + f'(x_j)\Delta x + f''(x_j)\frac{\Delta x^2}{2} + O(\Delta x^3), \tag{2.4}$$

donde se usan estas 2 expansiones con un error de truncamiento $O(\Delta x^3)$, con las cuales es posible construir aproximaciones para diferentes operadores diferenciales de f. Por ejemplo, si se toma la expresión (2.4), se le resta la expresión (2.3) y después se divide entre $2\Delta x$, se obtiene una expresión para la primera derivada en el punto x_j con un error de segundo orden:

$$f'(x_j) = \frac{f(x_{j+1}) - f(x_{j-1})}{2\Delta x} + O(\Delta x^2). \tag{2.5}$$

Se observa claramente que para tener el valor de la derivada, $f'(x_j)$, es necesario conocer los valores de f_{j+1} y f_{j-1}, por lo que se dice que esta aproximación es una aproximación centrada.

También es posible realizar operaciones similares de tal forma que la aproximación sea no centrada, como son las aproximaciones hacia atrás, es decir para obtener la derivada de f_j se necesitan conocer algunos valores anteriores como f_j, f_{j-1}, f_{j-2}, f_{j-3}, f_{j-4}, etcétera. De manera similar existen diferencias hacia adelante, donde para obtener la derivada, se necesitan conocer algunos de los valores posteriores como f_j, f_{j+1}, f_{j+2}, f_{j+3}, f_{j+4}, etcétera. Un ejemplo de estas aproximaciones son:

$$f'(x_j) = \frac{f(x_j) - f(x_{j-1})}{\Delta x} + O(\Delta x), \tag{2.6}$$

$$f'(x_j) = \frac{f(x_{j+1}) - f(x_j)}{\Delta x} + O(\Delta x), \tag{2.7}$$

las cuales son aproximaciones hacia atrás y hacia adelante respectivamente, con un error de primer orden.

Para obtener una expresión para la segunda derivada con una aproximación de segundo orden para la función f_j en torno a un punto x_j se procede de forma análoga mediante combinaciones de las expansiones en serie de Taylor en distintos puntos vecinos de x_j. Si se desarrollan las expresiones (2.3) y (2.4) hasta un orden de error $O(\Delta x)^4$ se tiene:

$$\begin{aligned} f(x_{j-1}) &= f(x_j) - f'(x_j)\Delta x + f''(x_j)\frac{(\Delta x)^2}{2} - f'''(x_j)\frac{(\Delta x)^3}{6} + O(\Delta x^4) \\ f(x_{j+1}) &= f(x_j) + f'(x_j)\Delta x + f''(x_j)\frac{(\Delta x)^2}{2} + f'''(x_j)\frac{(\Delta x)^3}{6} + O(\Delta x^4), \end{aligned} \tag{2.8}$$

de estas últimas se hace la siguiente combinación lineal, $f(x_{j-1}) - 2f(x_j) + f(x_{j+1})$ y se divide entre Δx^2, teniendo así:

$$f''(x_j) = \frac{f(x_{j-1}) - 2f(x_j) + f(x_{j+1})}{\Delta x^2} + O(\Delta x^2). \tag{2.9}$$

la cual también es una aproximación centrada, y de manera similar a la primera derivada se pueden crear distintas combinaciones lineales para obtener aproximaciones no centradas, como por ejemplo:

$$f''(x_j) = \frac{f(x_{j-2}) - 2f(x_{j-1}) + f(x_j)}{\Delta x^2} + O(\Delta x), \tag{2.10}$$

$$f''(x_j) = \frac{-f(x_{j+2}) + 2f(x_{j+1}) - f(x_j)}{\Delta x^2} + O(\Delta x), \tag{2.11}$$

las cuales son aproximaciones hacia atrás y hacia adelante respectivamente, con un error de primer orden.

En general, se procederá a realizar los cálculos con aproximaciones centradas. Aunque, en ocasiones cuando se trabaja en las fronteras del dominio es útil realizar aproximaciones no centradas, ya que estas solo cuentan con vecinos por una sola parte, por ejemplo, supongase que se tiene un problema en una dimensión para $x = 0, 1, 2, 3, \cdots$, la parte de la frontera izquierda, $x = 0$, solo tiene vecinos en la derecha, $x = 1$, $x = 2, \cdots$,etc, por lo tanto, solo es posible realizar una aproximación hacia adelante.

Una vez hecho lo anterior, se tiene una forma de construir una versión discreta de una EDP definida en un dominio espacio-temporal, que es un subconjunto del dominio original de la EDP. El siguiente paso para resolver la EDP, es emplear el método de líneas.

2.2. Método de líneas

El método de líneas (MoL) permite resolver numéricamente ecuaciones diferenciales parciales dependientes del tiempo, como un sistema discreto de ecuaciones diferenciales ordinarias (EDO's) en el tiempo. El método funciona para ecuaciones de primer orden en el tiempo, por ello, cuando se tiene una EDP de segundo orden en el tiempo, como la ecuación de onda, es necesario definir variables que reducen el problema a ecuaciones de primer orden en el tiempo.

Sea la EDP $\partial_t u = L(u)$, con $L(u)$ un operador arbitrario. El MoL consiste en definir una versión semidiscreta de la ecuación en cada punto del dominio discreto (t^n, x_j):

$$\partial_t u = [L(u)]^n_j, \tag{2.12}$$

donde $L(u)^n_j$ es una versión discreta del operador del lado derecho. Se tiene entonces una EDO en t^n para cada j. Para resolverlas, se integra para cada j de t^n a t^{n+1} usando un integrador numérico, en este caso se usa un integrador Runge Kutta de tercer orden.

Para ilustrar la manera de utilizar el MoL: sea $f = f(t, x)$ una función definida en un dominio temporal $t \in [t^0, t^{N_t}]$ y un dominio espacial $x \in [x_0, x_{N_x}]$, suponga que dicha función satisface la siguiente EDP

$$\partial_t f = \partial_x f, \tag{2.13}$$

con condición inicial $f(t^0, x) = g(x)$ y condiciones de frontera $f(t, x_0) = h_1(t)$ y $f(t, x_{N_x}) = h_2(t)$. Para encontrar una solución, se discretiza el dominio en una malla discreta como la que se muestra en la Fig. 2.1. Se discretiza la parte espacial de la ecuación original mediante una aproximación centrada de la siguiente forma para $j = 1, \cdots, N_x - 1$:

$$(\partial_t f)^n_j = \frac{f^n_{j+1} - f^n_{j-1}}{2\Delta x} + O(\Delta x^2), \tag{2.14}$$

de modo que dicha ecuación está definida en el dominio discreto espacial. Esta ecuación es llamada versión semidiscreta, ya que solamente la parte derecha está discretizada (usualmente suele llamarse $rhs(f)$), mientras que la parte izquierda queda intacta. La ecuación (2.14) es una EDO en t para cada valor de j, es decir, se tienen $N_x - 1$ EDO's en t de f, las cuales se resuelven de t^n a t^{n+1} usando un integrador Runge-kutta de tercer orden. Para esto es posible fijarse en un punto de la malla, el punto (t^n, x_j) y por medio de la ecuación (2.14), es posible calcular el valor de f^{n+1}_j, en función de los valores f^n_{j+1} y f^n_{j-1}, para todo $j = 1, \cdots, N_x - 1$, teniendo en cuenta que para las fronteras, $j = 0$, $j = N_x$, se emplean las condiciones de frontera del problema en particular. La Fig. 2.2, se llama molécula de evolución, muestra el comportamiento de la ecuación (2.14). El punto central blanco, muestra el punto donde es evaluada la función f, es decir f^n_j, mientras que los círculos blancos de las orillas muestran los puntos donde se calcula f_{j-1} y f_{j+1} al tiempo n, el círculo negro muestra el punto donde f se calcula en función de los otros dos, es decir f_j al tiempo $n + 1$.

Sin embargo, este paso no se realiza de forma directa, para llegar a este paso de tiempo, se emplea el integrador Runge-Kutta de tercer orden, el cual realiza tres pasos en el tiempo antes de poder calcular el valor de f_j^{n+1}, esto se ilustra mediante los puntos azules y los puntos verdes, los cuales muestran los puntos con los cuales se calculan f_{j+1}, f_{j-1} a $t^{(n+\frac{1}{3})}$ y f_{j+1}, f_{j-1} a $t^{(n+\frac{2}{3})}$ respectivamente.

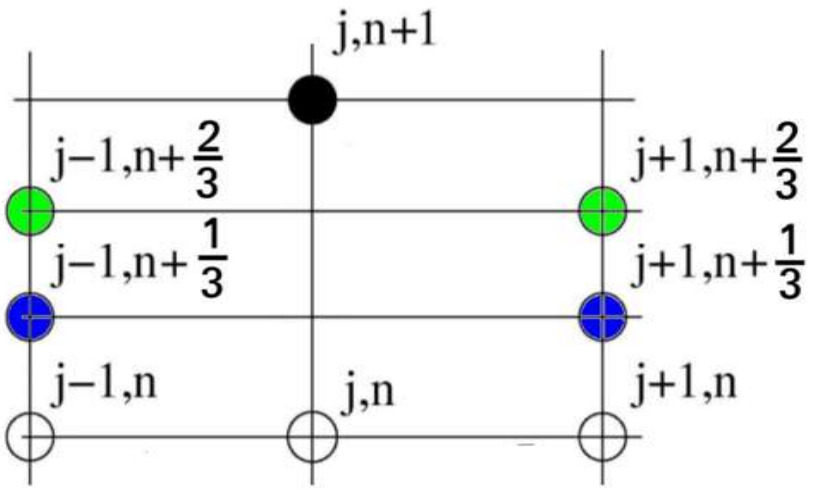


Figura 2.2: Molécula correspondiente al algoritmo de evolución de la ecuación (2.14), para la construcción de t^{n+1} a partir de t^n, usando un integrador Runge Kutta de tercer orden.

A continuación se presenta el integrador numérico empleado para resolver el sistema de EDO's en el tiempo.

2.3. Método de integración Runge-Kutta

Sea $\frac{dy}{dt} = f(t, y)$ una EDO, definida para un dominio en la variable independiente $t = t_0, t_1, t_2, \cdots$, y la variable dependiente $y = y_0, y_1, y_2, \cdots$, con $y(t_0) = y_0$. Es posible construir una solución numérica de esta EDO mediante el integrador Runge-Kutta (RK).

Ya que se trabaja en un dominio discreto, la EDO definida en un punto k de la malla es:

$$y_{k+1} = f(t_k, y_k), \tag{2.15}$$

con $k = 0, 1, 2, \cdots$. El método de integración RK consiste en proponer que y_{k+1} de la siguiente manera:

$$y_{k+1} = y_k + (a_1\kappa_1 + a_2\kappa_2 + \cdots + a_n\kappa_n)\,\Delta t \tag{2.16}$$

donde a_j son constantes y las κ_j los valores de $(\frac{dy}{dx})$ evaluadas en distintos puntos del lapso entre t_k y t_{k+1}:

$$\kappa_{i+1} = f(t_k + p_i\Delta t, y_k + \Delta t \sum_{j=1}^{N} q_{ij}\kappa_j), \quad i = 1, \ldots, m,$$

donde p_i, $q_{i,j}$ son constantes, las cuales sirven para determinar el tipo y precisión del RK y donde N determina el orden del RK [3]. Se trabajará con $N = 3$, es decir un Runge-Kutta de tercer orden (RK3), para el cual se usaran las siguientes expresiones.

$$y_{k+1} = y_k + \frac{1}{6}(\kappa_1 + 4\kappa_2 + \kappa_3)\Delta t, \tag{2.17}$$

con

$$\kappa_1 = f(t_k, y_k),$$
$$\kappa_2 = f(t_k + \frac{1}{2}\Delta t, y_k + \frac{1}{2}\kappa_1\Delta t),$$
$$\kappa_3 = f(t_k + \Delta t, y_k - \kappa_1\Delta t + 2\kappa_2\Delta t),$$

donde las constantes fueron determinadas de una manera muy particular, pues el sistema que obedece está sub-determinado. Un análisis más detallado de la manera en que se construye este RK3 y como se determinan dichas constantes, se muestra en el *Apéndice* A.

Es posible aplicar estos criterios para la discretización temporal del problema de la ecuación diferencial parcial del tipo (2.12) en cada punto x_j, de modo que una vez teniendo en cuenta todo lo anterior se puede resumir, a grandes rasgos, que si se desea encontrar una solución numérica de un PVI, bastará con resolver una serie de $N_x - 1$ EDO's para pasar del tiempo t^n al t^{n+1}, las cuales se resuelven mediante el método de líneas y la ayuda de un integrador numérico.

2.4. Estudio de errores

Cuando se trabaja con soluciones aproximadas se debe tener en cuenta los errores asociados a la solución numérica. Gracias al cálculo de los errores, es posible determinar si la solución aproximada es correcta o no.

Los errores que se abordarán en este trabajo son los errores de truncamiento, se dividen en dos tipos, error local y error global. Para ilustrarlo, considérese el problema $\frac{dy}{dt} = f(t, y)$ como en la sección anterior. *El error local* de una solución es la diferencia entre la solución exacta evaluada en t_k y el valor aproximado de la solución en t_k, esto es:

$$E_k^{loc} = y(t_k) - y_k, \tag{2.18}$$

donde $y(t_k)$ es la solución exacta evaluada en t_k, mientras que y_k es la solución numérica en el punto t_k la cual se construye a partir de $y(t_{k-1})$.

Por otro lado, *el error global* de una solución en el punto t_k es el error local acumulado en la solución aproximada desde t_0 hasta t_k. Se estima de la siguiente forma:

$$E_k^{glob} \approx (k\text{-pasos}) \times E_k^{loc} \approx \frac{t_k - t_0}{\Delta t} E_k^{loc} \tag{2.19}$$

donde E^{glob} y E^{loc}, son el error global y el error local, y el valor para k puede ser representado en términos de $(t_k - t_0)/\Delta t$, además, se observa que el error global es un grado menor que el error local.

Para el caso del integrador RK3 se tiene un error local de cuarto orden, mientras que el error global es de tercer orden. El argumento del porqué es preferible usar un método de integración RK3, sobre otros métodos como el método de Forward Euler, Backward Euler, Trapecio o inclusive los Métodos RK2, es porque el orden del error asociado del método RK3 es mayor, lo cual implica que el error es menor que el de los otros métodos.

Los métodos numéricos usados involucran errores espaciales de orden $O(\Delta x^2)$ como lo ilustra el ejemplo de la ecuación (2.14), por su parte, el integrador RK3 involucra error de orden $O(\Delta t^3)$. Se define una relación entre las resoluciones espaciales y temporales, $\Delta t = c\Delta x$ por lo tanto, el error del RK3 es $O(\Delta x^3)$, donde c es una constante llamada factor de Courant Friedrich Lewy (CFL). En el caso particular de la ecuación (2.14), el error dominante será de orden $O(\Delta x^2)$ y es con el que se trabajará de ahora en adelante. Es natural, que cuanto mayor es la resolución con que se construye una malla, menor será el error con que se están aproximando los operadores diferenciales a sus valores en el continuo, al igual que en la solución de EDO's.

Dado que se resuelve una versión aproximada de una EDP, es necesario verificar si dicha solución converge a la solución de la ecuación en el continuo.

Convergencia

En los casos que ilustra esta tesis de cómputo cientifico, se han elegido PVI's cuya solución exacta es conocida, lo que facilita el cálculo de los errores. Para mostrar un criterio de convergencia, puesto que se asume que el error dominante es de $O(\Delta x^2)$, la solución numérica debe converger con orden 2. Para observar esto, se escribe el error global de la siguiente manera:

$$E_1^{glob} = O(\Delta x_1^2) = E(x_j)\Delta x^2, \tag{2.20}$$

con $E(x_j)$ un coeficiente de error en la ecuación continua. Si se define una nueva resolución $\Delta x_2 = \Delta x/2$, se tiene que

$$E_2^{glob} = O(\Delta x_2^2) = E(x_j)\left(\frac{\Delta x}{2}\right)^2. \tag{2.21}$$

Si se toma la ecuación (2.20) y (2.21) se tiene la siguiente relación

$$\frac{E_1^{glob}}{E_2^{glob}} = 2^2, \tag{2.22}$$

el número $2^2 = 4$ en la ecuación (2.22) se llama factor de convergencia y debe ser evaluado en cada punto de la malla donde se ha calculado la solución numérica. Entonces, si la resolución aumenta por un factor de 2, la solución numérica se aproxima a la solución continua de manera cuadrática con cada aumento de la resolución. El hecho de que sea cuadrática es consecuencia de que la aproximación que se construyó involucra errores de segundo orden [3]. De manera análoga, es posible mostrar que si la aproximación tiene error global de cuarto orden, el factor de convergencia será $16 = 2^4$. De forma más general, si se tiene una resolución $\Delta x_\alpha = \Delta x/\alpha$, con un error global de orden β, se tiene entonces que el factor de convergencia es: $E_1/E_\alpha = \alpha^\beta$, donde α es el factor de refinamiento.

Calcular el factor de convergencia usando las pruebas de convergencia, es verificar que efectivamente el error global, de la resolución Δx sea igual a 4 veces el error global de la solución numérica usando la resolución $\Delta x/2$, y a su vez, sea igual a 16 veces el error de la solución numérica usando la resolución $\Delta x/4$. Esto se hace comparando la gráficas del error como se mostrará más adelante. De modo que mientras se cumplan estas condiciones, es posible afirmar que la solución numérica converge a la solución exacta en el continuo.

Sin embargo, existe un resultado que expresa lo anterior de manera más general, para ello, se usan las siguiente definiciones dadas por [5]:

Definición 1 *Sea $\partial_t u = L(u)$ una ecuación diferencial parcial, con $u = u(x,t)$, cuya solución exacta es $u_{ex}(x,t)$. Se dice que el sistema discreto $[\partial_t u]_j^n = [L(u)]_j^n$, con u_j^n la solución numérica, converge punto a punto sí, para cualquier x y t como $(j\Delta x, n\Delta t) \to (x,t)$, u_j^n converge a $u_{ex}(x,t)$ cuando Δx y Δt convegen a 0.*

Definición 2 *Sea $\partial_t u = L(u)$ una ecuación diferencial parcial, como antes. Se dice que el sistema discreto $[\partial_t u]_j^n = [L(u)]_j^n$ converge en el tiempo t sí, dado $t^n = n\Delta t$*

$$\| u^{n+1} - u_{ex}^{n+1} \| \to 0,$$

cuando $\Delta x \to 0$ y $\Delta t \to 0$.

Donde $\| \cdot \|$, es una norma, que en general puede ser la norma L_1, L_2, L_∞, etc.

Definición 3 *Sea $\partial_t u = L(u)$ una ecuación diferencial parcial. Se dice que el sistema discreto $[\partial_t u]_j^n = [L(u)]_j^n$, converge a orden (p, q), sí para cualquier t, como $t^n = n\Delta t$*

$$|| \, u^{n+1} - u_{ex}^{n+1} \, ||= O(\Delta x^p) + O(\Delta t^q)$$

cuando Δx y Δt convergen a 0.

Una explicación más profunda de estas definiciones se encuentra en [5], donde por el momento es suficiente con saber lo que expresan estas definiciones. Es decir, basta con demostrar que si una de las normas del error en t^n converge, la solución también converge en cada t^n. Esto permite calcular un escalar (la norma del error) como función del tiempo y verificar la convergencia durante la solución de la EDP. Para este caso, se calculan las normas $L_1(e)$ y la norma $L_2(e)$ del vector error que se definen como:

$$L_1 = \int |e_j| dx = \sum_{j=0}^{N} |e_j| \Delta x, \tag{2.23}$$

$$L_2 = \sqrt{\int e_j^2 dx} = \sqrt{\sum_{j=0}^{N} |e_j|^2 \Delta x}, \tag{2.24}$$

donde $e_j = u_{ex}(x_j) - u_j$ es el error en el punto x_j del dominio espacial, con $e_{ex}(x_j)$ la solución exacta de la EDP y u_j el valor de la solución numérica en x_j y N es el número total de puntos del dominio espacial. Las integrales se calculan usando la regla del trapecio, L_1 y L_2 son escalares en cada t^n, que se puede calcular durante toda la evolución. Con ellos es posible (usando las definiciones) verificar la convergencia de la solución durante la evolución y así saber si la solución numérica es correcta.

Capítulo 3

Programación en MPI

A pesar del increible desarrollo de la tecnología computacional, existen problemas muy costosos que necesitan muchos recursos computacionales, incluso prohibitivos. Por ejemplo, si se desea resolver una EDP con un dominio que contiene un billón de puntos, se vuelve un problema prohibitivo ya que una computadora convencional no tiene la memoria RAM suficiente para alojar un billón de números de doble precisión y por tanto no es posible usarla para resolver el problema. Tales problemas pueden ser resueltos empleando cómputo en paralelo para sumar la memoria RAM manejada por muchas CPU's que suelen estar concatenadas en clusters o supercomputadoras.

En esta tesis, se presenta la programación en paralelo para resolver PVIs, ya que el dominio espacial de estos puede dividirse en varios subdominios y en cada subdominio es posible resolver la EDP con un procesador mediante el MoL.

La implementación del MoL (2.12) se basa en la definición de un dominio espacio-temporal discreto. Tal dominio se define en una dimensión como: $[x_0, x_{N_x}] \times [0, t_{N_t}]$. Dicho dominio se discretiza como $x_i = x_0 + i\Delta x$, $t^n = n\Delta t$ respectivamente, donde $i = 0, 1, \cdots, N_x$, $n = 0, 1, \cdots, N_t$ son enteros que etiquetan los puntos donde estará definida la EDO, donde $\Delta x = (x_{N_x} - x_0)/N_x$ y Δt se determina en función de Δx, por medio del factor de Courant ($\Delta t = c\Delta x$).

Para la discretización en tres dimensiones se tiene: $[x_0, x_{N_x}] \times [y_0, y_{N_y}] \times [z_0, z_{N_z}] \times [0, t_{N_t}]$. Dicho dominio se discretiza como $x_i = x_0 + i\Delta x$, $y_j = y_0 + j\Delta y$, $z_k = z_0 + k\Delta z$ y $t^n = n\Delta t$ respectivamente, donde $i = 0, \cdots, N_x$, $j = 0, \cdots, N_y$, $k = 0, \cdots, N_z$, $n = 0, \cdots, N_t$ son enteros que etiquetan los puntos donde estará definida la EDP, donde $\Delta x = (x_{N_x} - x_0)/N_x$, $\Delta y = (y_{N_y} - y_0)/N_y$, $\Delta z = (z_{N_z} - z_0)/N_z$ y para Δt se usa el factor de Courant, $\Delta t = c\ min(\Delta x, \Delta y, \Delta z)$, pero aquí se usa $\Delta x = \Delta y = \Delta z$, entonces se tiene $\Delta t = c\ (\Delta x)$,

Cuando N_x, N_y, N_z son grandes, la memoria requerida es enorme. Por ejemplo para resolver la ecuación de onda en tres dimensiones, con $N_x = N_y = N_z = 100$, se tiene un cubo de dimensión 100^3, el cual ocupa aproximadamente $2Gb$ de memoria RAM para alojar números de doble precisión, los cuales aun se pueden alojar en una computadora convencional. Pero si se duplica el número de puntos para cada eje espacial, se estará mul-

tiplicando la memoria RAM por un factor de 2^3, tienendo un cubo de dimensión 200^3, que requiere aproximadamente $15Gb$ de memoria RAM para alojar números de doble precisión, los cuales no pueden alojarse en una computadora convencional ya que estas no cuentan con suficiente memoria RAM y por ello, se convierte en un problema prohibitivo.

Para desarrollar la programación en paralelo, se usa la interfaz MPI, que es una biblioteca de paso de mensaje, la cual proporciona una serie de funciones para los lenguajes de programación *C*, *C++* y para este caso, Fortran90.

3.1. Funciones básicas

Antes de continuar, se presenta la estructura del programa de cómputo usando la biblioteca MPI:

Inicializar programa Fortran90

Inicializar MPI

Averiguar el número total de procesadores y etiquetar a cada uno.

-Instrucciones de un programa serial

- Cada procesador ejecuta las instrucciones
- Comienza la comunicación entre procesadores

- Finalizan instrucciones y se obtienen resultados
- Se imprimen los resultados

Finalizar MPI

Finalizar programa

Para inicializar MPI en el lenguaje fortran90 se usa el siguiente encabezado

- include 'mpi.h',

el cual sirve para que el programa reconozca todas las funciones de la biblioteca MPI.

Otras funciónes fundamentales son:

- MPI_INIT(ierr),
- MPI_COMM_SIZE(MPI_COMM_WORLD, num_procs, ierr),
- MPI_COMM_RANK(MPI_COMM_WORLD, proc_id, ierr),

La primera función inicializa la biblioteca de MPI, donde el argumento (ierr) es un número entero al cual se le asocia un valor que denotará un tipo de error. Este es un argumento que todos los comandos de MPI usan, así, si existe algún error se aborta la instrucción deteniendo el programa y se le asocia un número que caracteriza algún tipo de error. Por otra parte MPI_COMM_SIZE y MPI_COMM_RANK son funciones que leen el número total de procesadores que se reciben como instrucción desde la línea de ejecución y determinan una etiqueta $0, 1, 2, \ldots$, para cada uno de los procesadores respectivamente, ambos argumentos deben ser números enteros. El argumento MPI_COMM_WORLD es para informarle al programa que se estará comunicando con varios procesadores.
Otras funciones que se usarán son las siguientes:

- MPI_WTIME(ierr),

regresa el tiempo computacional del procesador en segundos desde que comenzó a trabajar, hasta que se manda llamar.

- MPI_BCAST(buffer, contador, MPI_datatype, MASTER, MPI_COMM_WORLD,ierr),

recibe varios argumentos, el argumento buffer inicializa una dirección temporal de almacenamiento de información la cual sirve para poder transferir parámetros, como el número de puntos de cada procesador. Después se tiene un contador que determina cuantos datos se almacenan, MPI_datatype determina de qué tipo son. Por último, el argumento MASTER es el procesador 0, el cual se define como un procesador global, sirve para dar las instrucciones a los demás procesadores y también para guardar los resultados.

- MPI_SEND(buffer, contador, MPI_datatype, proc_id, tag, MPI_COMM_WORLD, status, ierr)
- MPI_RECV(buffer, contador, MPI_datatype, proc_id, tag, MPI_COMM_WORLD, status, ierr)
- MPI_ISEND(buffer, contador, MPI_datatype, proc_id, tag, MPI_COMM_WORLD, status, ierr)
- MPI_IRECV(buffer, contador, MPI_datatype, proc_id, tag, MPI_COMM_WORLD, status, ierr)
- MPI_WAIT(request, status, ierr)

Las funciones Send y Recv, son dos funciones fundamentales en MPI, los cuales sirven para enviar y recibir información entre procesadores. Existen muchas variantes de estas funciones, pero estos cuatro son con los que se trabajará aquí. Se usan estas, ya que por una

parte MPI_SEND es una función de envio con bloqueo, es decir, al momento de enviar alguna información previamente almacenada en un buffer se bloquea, hasta que lo que se haya enviado, sea recibido. Mientras que MPI_ISEND es una función de envío con no-bloqueo, el cual permite enviar información sin ninguna restricción, similarmente para MPI_RECV y MPI_IRECV, con la diferencia que estas funciones no envían información, sino que estos se encargan de recibirla. El primer argumento de estas funciones, es un buffer que sirve para almacenar información relevante. El segundo argumento es un contador que se encarga de determinar cuantos parámetros van a ser enviados o recibidos. El tercer argumento representa el tipo de contador, es decir, entero, doble precisión, entero corto, punto flotante, etcétera. El cuarto argumento se refiere al destino a donde va a ser enviado, o de donde va a ser recibido. El quinto argumento es una etiqueta, la cual tiene que ser la misma para el SEND y para el RECV, puesto que etiqueta la información que se esta comunicando para compartir la información.

La función WAIT se usa acompañada de alguna de las funciones Send o Recv, sirve para restringir a los procesadores, haciendo que esperar a que terminen de realizar una instrucción para poder hacer la siguiente. Por ejemplo, si se desea enviar desde el procesador 1 y el procesador 2, 10 datos al procesador MASTER cada uno, donde este último los procesará y a su vez tiene que regresarle los 10 datos ya procesados a ambos, al usar el comando WAIT lo que sucede es que el procesador MASTER inicialmente recibe los 10 primeros datos del procesador 1, los procesa y los envía de regreso, haciendo que el procesador 2 espere. Una vez que el procesador 1 termina, entonces el procesador 2 envia los 10 datos al MASTER, este los procesa y los envía de regreso. Para esto sirve el argumento 'request', el cual es una operación de identificación y se le impone que hasta que una instrucción esté completa, realice la otra. Esta última función es muy similar al envio con bloqueo, ya que bloquea el paso de información hasta que el otro procesador reciba la información anterior.

3.2. Comunicación entre procesadores

Ahora se dará una breve descripción de la forma en que se comunican los procesadores. En primer lugar, se definen los vecinos izquierdo y derecho de cada uno de los procesadores, esto se hace en una sección dentro del código de Fortran90, donde simplemente se da una serie de condiciones para definirlos. La idea de los vecinos izquierdo y derecho, es ideal a la hora de trabajar con EDP, puesto que permite tener una comunicación entre las fronteras de cada procesador. En la Fig. 3.1 se muestra un ejemplo con 5 procesadores, donde las flechas indican en donde esta la comunicación entre los procesadores, tanto a la izquierda como a la derecha.

A continuación se presenta la manera en que se distribuye el dominio con el que se desea trabajar, tanto en una dimensión, como en tres dimensiones.

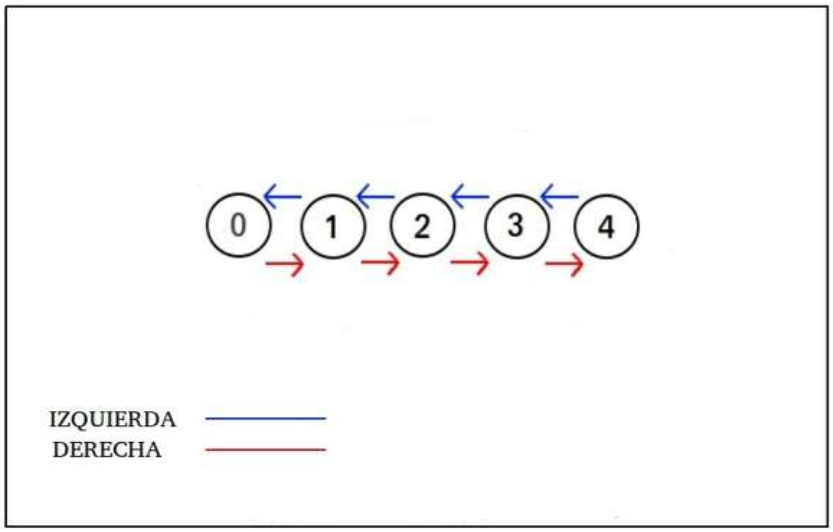


Figura 3.1: Ejemplo con 5 procesadores, se denota cada uno por el número de cada procesador: 0, 1, 2, 3 y 4, los cuales identifican a su vecino izquierdo y vecino derecho, por ejemplo el vecino izquierdo del procesador 3 será el 2 y el derecho será el 4.

Distribución en una dimensión

La forma de distribuir el dominio en una dimensión conviene ilustrarla con otro ejemplo. Suponga que se tiene un dominio espacial unidimensional como el antes mencionado, que es llamado dominio total. Supóngase también, que se trabajará con 4 procesadores. Se compara la forma en que se distribuye el dominio en un procesador y con 4 procesadores, los cuales serán: 0, 1, 2, 3. El primer caso, se presenta en la Fig. 3.2, donde se muestra la discretización del dominio con un solo procesador. Para este caso de programación serial, una vez discretizado el dominio, se procede a resolver el problema de ecuaciones diferenciales parciales dependiente del tiempo de forma directa, mediante el MoL.

Por otra parte, al trabajar con 4 procesadores, lo primero que se hace es discretizar el dominio igual que en el caso de un procesador, enseguida se realiza la repartición del dominio entre los 4 procesadores, es decir, el dominio se reparte en 4 secciones lo más parecido posible, donde la repartición dependerá del número de procesadores y del número de puntos de la discretización del dominio total. La forma de obtener reparticiones del dominio lo más parecido posible, se ilustra mediante el siguiente ejemplo: Supóngase que se trabaja con 7 procesadores y con 100 puntos, de modo que a cada procesador le tocaran $100/7 = 14$ puntos, sobrando 2, los cuales se reparten en los primeros 2 procesadores, teniendo así una distribución lo más uniformemente posible. La Fig. 3.3, muestra la redis-

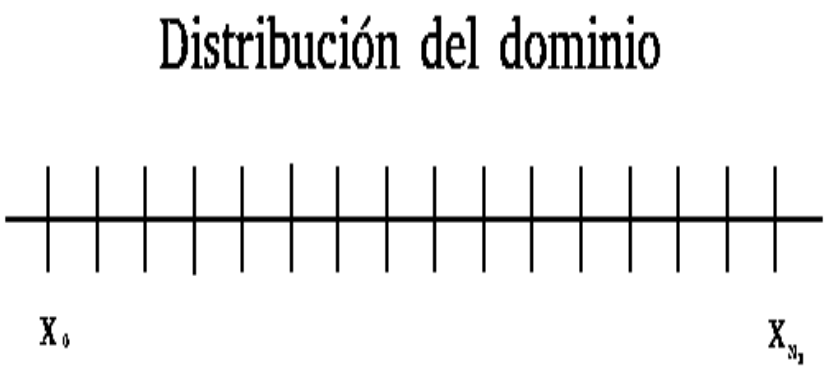


Figura 3.2: Discretización del dominio en una dimensión para un sólo procesador, se discretiza todo el dominio en una malla unidimensional con la distancia de separación de cada uno de los puntos $\Delta x = (x_{N_x} - x_0)/N_x$.

tribución del dominio entre los 4 procesadores, donde el procesador 0 tiene N_{x_0} puntos, el procesador 1 tiene N_{x_1} puntos, el procesador 2 tiene N_{x_2} puntos y el procesador 3 tiene N_{x_3} puntos, tales que $N_{x_0} + N_{x_1} + N_{x_2} + N_{x_3} = N_x$, el dominio total.

Para construir la distribución del dominio de forma general, supóngase que se trabaja en un dominio que tiene N_x puntos, y que se pretende trabajar con M procesadores, tales que $m = 0, 1, 2, \cdots, M - 1$, etiqueta al procesador m. Entonces, el dominio se reparte usando la función módulo: $N_x mod(M) = r$, donde r es el residuo de la función módulo, el cual se reparte entre los primeros m procesadores. Teniendo entonces que los primeros procesadores tendran un punto más que los otros y se espera que estos primeros tardarán un poco más en realizar el trabajo, ya que comunicarán más información y tendran que realizar más cálculos.

Además, si el número de puntos de cada procesador se determina de tal manera que el procesador m tiene N_{x_m} puntos, entonces $N_x = \sum_0^{M-1} N_{x_m}$.

Una vez que se tiene el dominio distribuido en cada procesador, se procede a resolver la EDP, para ello se aplica el MoL. Dado que el dominio está repartido entre varios procesadores, cada uno de ellos resuelve la ED en su dominio espacial de t^n a t^{n+1}. Sin embargo, para poder construir la solución al tiempo siguiente, cada procesador necesita comunicarse con sus vecinos en cada paso de tiempo mediante las funciones SEND y RECV. Para mostrar esta comunicación, es conveniente retomar la ecuación (2.14), tomando en cuenta que para este caso, esta ecuación está definida en cada procesador. De modo que para calcular

Figura 3.3: Forma en que se distribuye el dominio de la Fig. 3.2 en cuatro procesadores. Los números denotan los procesadores 0,1,2,3. Se define la distribución, de tal manera que el procesador 0 cuente con N_{x_0} puntos, el procesador 1 con N_{x_1} puntos, el procesador 2 con N_{x_2} puntos y por último, el procesador 3 con N_{x_3} puntos, tales que $N_{x_0} + N_{x_1} + N_{x_2} + N_{x_3} = N_x$.

la solución de esta ecuación en el procesador 0, se tiene:

$$[\partial_t f]^n_j = \frac{f^n_{j+1} - f^n_{j-1}}{2\Delta x}, \tag{3.1}$$

donde $j = 1, \cdots, N_{x_0}$. Para el procesador 1, la ecuación a resolver es:

$$[\partial_t f]^n_j = \frac{f^n_{j+1} - f^n_{j-1}}{2\Delta x}, \tag{3.2}$$

donde $j = 1, \cdots, N_{x_1}$. De manera análoga se calcula la solución para los demás procesadores, cambiando el dominio de la etiqueta j. Sin embargo, se tiene un problema en los extremos, para el procesador 0 en el extremo izquierdo no hay problema, ya que el valor de $[\partial_t f]^n_0$ es determinado por las condiciones de frontera. Pero por otro lado, no es posible calcular el valor de $[\partial_t f]^n_{N_{x_0}}$, ya que se tiene que resolver

$$[\partial_t f]^n_{N_{x_0}} = \frac{f^n_{N_{x_0}+1} - f^n_{N_{x_0}-1}}{2\Delta x}, \tag{3.3}$$

donde el valor de f^n en el punto $N_{x_0} + 1$ no esta definido en el dominio del procesador 0. Pero por construcción, el punto N_{x_0+1} del procesador 0, es el punto 1 del procesador 1 como lo ilustra la Fig. 3.4. Por lo tanto, se realiza una comunicación entre estos 2 procesadores, de modo que se comunica el valor de la función f^n_1 del procesador 1 que equivale al valor

de la función $f^n_{N_{x_0}+1}$ del procesador 0.

De manera similar, para la ecuación (3.2) no es posible calcular el valor de la función en el extremo izquierdo, f^n_1 del procesador 1, ya que se tiene:

$$[\partial_t f]^n_1 = \frac{f^n_2 - f^n_0}{2\Delta x}, \tag{3.4}$$

donde f^n en el punto 0 no esta determinado en el dominio del procesador 1, pero usando el mismo argumento, el punto 0 del procesador 1, es el punto N_{x_0} del procesador 0, como se observa en la Fig. 3.4, de modo que se comunica el valor de la función $f^n_{N_{x_0}}$ del procesador 0 con su equivalente, el valor de la función f^n_0 del procesador 1.

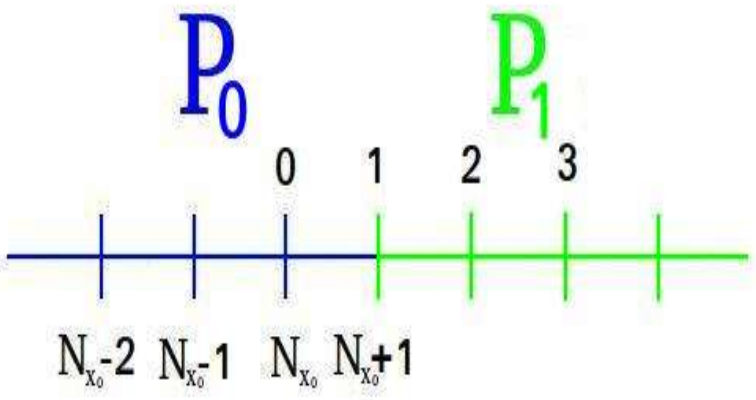


Figura 3.4: Comunicación entre el procesador 0, y el procesador 1. Las etiquetas de abajo denotan los puntos del procesador 0, las de arriba los del procesador 1. Se realiza una comunicación para intercambiar valores de los extremos, para ello se impone la condición de que el valor de $f^n_{N_{x_0}+1}$ del procesador 0 sea el valor de f^n_1 del procesador 1. De igual manera, se impone que el valor de f^n_0 del procesador 1 sea el valor de $f^n_{N_{x_0}}$ del procesador 0. De este modo se comunica la información requerida entre los procesadores 0 y 1.

Se usa esta convención para comunicar las fronteras de los procesadores y poder calcular f^{n+1}_j en todo el dominio.

Distribución en tres dimensiones

La distribución del dominio en tres dimensiones del problema en EDP's se presenta mediante un ejemplo. Supóngase que se tiene un dominio tridimensional definido por: $\{[x_0, x_{N_x}] \times [y_0, y_{N_y}] \times [z_0, z_{N_z}]\}$, que es discretizado como antes, mediante una malla uniforme tridimensional. Al igual que en el caso de una dimensión, en la malla discretizada

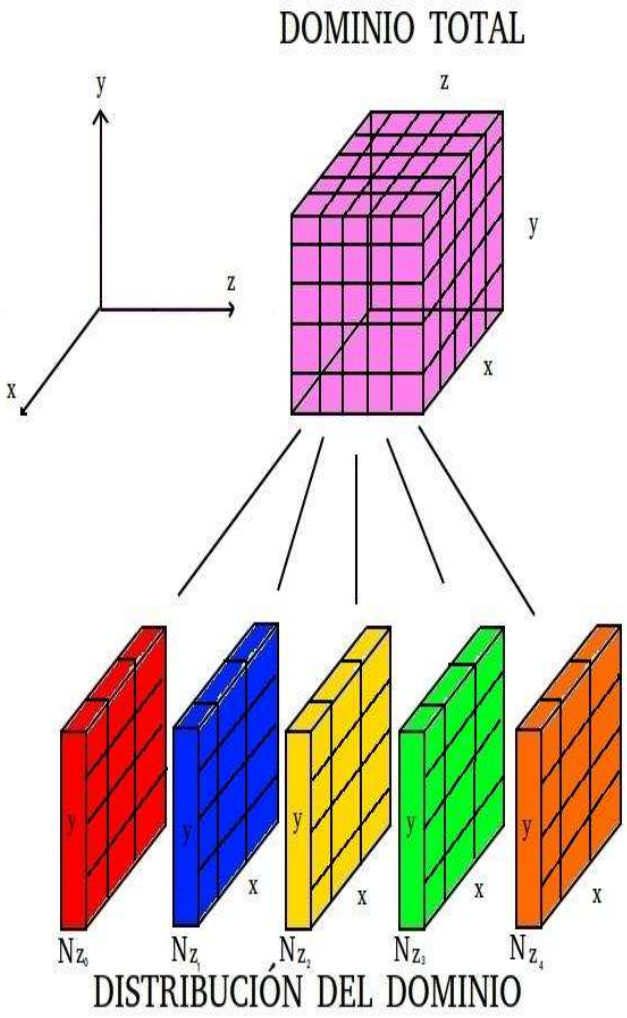


Figura 3.5: Discretización del dominio tridimensional: $x_i = x_0 + i\Delta x$, $y_j = y_0 + j\Delta y$, $z_k = z_0 + k\Delta z$, con $i = 0, \cdots, N_x$, $j = 0, \cdots, N_y$, $k = 0, \cdots, N_z$, donde $\Delta x = (x_{N_x} - x_0)/N_x$, $\Delta y = (y_{N_y} - y_0)/N_y$, $\Delta z = (z_{N_z} - z_0)/N_z$. Se trabaja con cinco procesadores y el dominio se reparte a lo largo del eje z, obteniendo 5 subdominios paralelepípedos, tales que el número de puntos a lo largo del eje z de cada subdominio, es el siguiente: N_{z_0} para el primer segmento, N_{z_1} para el segundo, N_{z_2} para el tercero, N_{z_3} para el cuarto y N_{z_4} para el quinto.

se realizan particiones del dominio dependiendo del número de procesadores. Supóngase también que se trabajará con 5 procesadores. La Fig. 3.5 muestra un cubo que ilustra el dominio tridimensional y la distribución del dominio en 5 subdominios que se asignarán a 5 procesadores, para esto, se toman bloques de datos a lo largo del eje z, es decir,se realizan particiones del dominio a lo largo de este eje teniendo así 5 prismas rectangulares con el mismo número de puntos tanto en x como en y pero para el eje z se toma en cuenta que algunos prismas rectangulares pueden ser más largos que otros, lo cual dependerá del número de procesadores y del número de puntos con los que se este trabajando en este eje. Para ello, se usa nuevamente la función $N_z mod(M) = r$, con N_z el número de puntos del eje z, M el número de procesadores y r el residuo de la función *mod*, el cual se reparte entre los primeros procesadores.

Una vez que se tiene el dominio distribuido en cada procesador, se procede a resolver la EDP. Dado que el dominio esta repartido entre varios procesadores, cada uno resuelve la ED en su dominio espacial de t^n a t^{n+1}. Teniendo en cuenta que en cada paso de tiempo necesita comunicarse cada procesador con sus vecinos para poder construir la solución a un tiempo posterior. Para ilustrarlo, supóngase que se desea resolver la siguiente EDP:

$$\partial_t f = \partial_x f + \partial_y f + \partial_z f, \tag{3.5}$$

con $f = f(t, x, y, z)$, definida en el domino tridimensional como el de la Fig. 3.5, cuya versión semidiscreta es:

$$[\partial_t f]^n_{i,j,k} = \frac{f^n_{i+1,j,k} - f^n_{i-1,j,k}}{2\Delta x} + \frac{f^n_{i,j+1,k} - f^n_{i,j-1,k}}{2\Delta y} + \frac{f^n_{i,j,k+1} - f^n_{i,j,k-1}}{2\Delta z}, \tag{3.6}$$

donde $i = 0, \cdots, N_x$, $j = 0, \cdots, N_y$, $k = 0, \cdots, N_z$, $n = 0, \cdots, N_t$ son enteros que etiquetan los puntos donde esta definida la EDP. Al realizar las reparticiones del dominio, se tiene un prisma rectangular para cada uno de los procesadores, donde N_x y N_y son iguales para todos los procesadores, pero en el eje z se tienen, N_{z0} puntos para el procesador 0, N_{z1} puntos para el procesador 1, N_{z2} puntos para el procesador 2, N_{z3} puntos para el procesador 3 y N_{z4} puntos para el procesador 4, de modo que $N_{z_0} + N_{z_1} + N_{z_2} + N_{z_3} + N_{z_4} = N_z$.

Hay que mecionar, que al igual que en el caso de una dimensión, se presenta un problema al tratar de calcular directamente la solución de la EDP. Para ilustrarlo, basta con intentar calcular el valor de $[\partial_t f]^n_{i,j,N_{z_0}}$ para el procesador 0:

$$[\partial_t f]^n_{i,j,N_{z_0}} = \frac{f_{i+1,j,N_{z_0}} - f_{i-1,j,N_{z_0}}}{2\Delta x} + \frac{f_{i,j+1,N_{z_0}} - f_{i,j-1,N_{z_0}}}{2\Delta y} + \frac{f_{i,j,N_{z_0}+1} - f_{i,j,N_{z_0}-1}}{2\Delta z}, \tag{3.7}$$

donde los punto de $(i, j, N_{z_0} + 1)$, para todo i, j, no estan definido en el dominio de este procesador. Sin embargo, por construcción, los puntos $(i, j, N_{z_0} + 1)$ del procesador 0, son los puntos $(i, j, 1)$ del procesador 1. De tal manera que los valores de la función $f_{i,j,N_{z_0}+1}$ en el procesador 0 son los valores de la función $f_{i,j,1}$ del procesador 1.

Teniendo entonces que la comunicación se realiza mediante las fronteras en el eje z de los prismas rectangulares de cada procesador. Como se observa en la Fig. 3.5, cada frontera esta discretizada y tiene N_x por N_y puntos. Entonces, no se realiza la comunicación punto a punto como en una dimensión, sino que se realiza mediante planos de N_xN_y puntos, comunicando $N_x \times N_y$ números de doble precisión para calcular las derivadas a lo largo del eje z, ya que en estas es donde existe un problema. De tal modo que en cada frontera se realiza este mismo procedimiento para lograr construir la solución al tiempo posterior.

Una vez que se sabe como calcular la solución para cada procesador, el siguiente paso es salvar los datos. Hay dos formas para esto:

Caso 1.
: El procesador MASTER recolecta, de cada uno de los procesadores, las soluciones en cada paso de tiempo, guardandolas en un archivo de datos, después, simplemente lo imprime en la pantalla.

Caso 2.
: Cada procesador salva la solución en cada paso de tiempo, guardandola en un archivo de datos, después se imprime en la pantalla la solución de cada procesador en su dominio.

Para los problemas que se abordarán en esta tesis, bastará con los conceptos y funciones básicas que hasta el momento se han presentado usando [2] y [6].

3.3. Escalamiento

Ya que se usa una interfaz que permite trabajar con varios procesadores, es preciso tener una forma de medir y comparar los resultados que se obtienen al calcular la solución en serie, es decir con un solo procesador y calcularla con 2, o más procesadores, en este caso es posible realizar dichos cálculos hasta un total de 12 procesadores, los cuales son la capacidad máxima del equipo de cómputo donde se realiza este trabajo. Para hacer estas comparaciones se realizan pruebas de escalamiento, que consisten en medir el tiempo que tarda el código en resolver un PVI. Para hacer esto, se manda llamar a la función MPI_WTIME, que se caracteriza por devolver el tiempo en segundos cuando se manda a llamar, el cual representa el tiempo de cómputo transcurrido. Técnicamente se usa el nombre de wall_time, en español, tiempo de trabajo.

Por medio de las pruebas de escalamiento se compara la eficacia de trabajar con varios procesadores, para verlo claramente se hacen gráficas de los tiempos de duración de las corridas contra el número de procesadores que se están usando. En este trabajo se realizarán estas pruebas con 1, 2, 4, 8 y 12 procesadores. Para tener un valor de tiempo de ejecución confiable, es conveniente hacer varias corridas del programa para cada valor del número de procesadores con los que se trabajó, y con estas corridas se calcula un promedio.

Aprovechando el hecho de que se trata de una prueba de escalamiento, es decir, solo interesa medir el tiempo de trabajo, se toman dos mediciones distintas, unas guardando los datos de las soluciones cada cierto número de iteraciones y otras al guardar solo los datos inicial y final de las soluciones. Se realizan estas dos pruebas, ya que cuando se paraleliza, cada procesador tiene acceso al disco duro y cada procesador lo hace en su momento. De modo que si se mandan salvar datos y despúes se realiza una comunicación, cada procesador debe esperar a que los datos se hayan guardado previamente, para poder comunicarse.

Capítulo 4

La ecuación de onda

4.1. La ecuación de onda en una dimensión

La ecuación de onda en una dimensión espacial, se define de la siguiente manera

$$\frac{\partial^2 \varphi(t,x)}{\partial t^2} = c^2 \frac{\partial^2 \varphi(t,x)}{\partial x^2}, \tag{4.1}$$

donde c es la velocidad de la onda y con $\varphi(0,x)$ y $\dot{\varphi}(0,x)$ condiciones iniciales. Se resuelve esta ecuación de forma numérica, y se pretende ilustrar cómo se trabaja de forma paralelizada. Además, esta ecuación tiene solución exacta, con la cual es posible comparar las soluciones obtenidas y calcular errores de la solución numérica. Se trabaja con $c = 1$, teniendo entonces:

$$\frac{\partial^2 \varphi(t,x)}{\partial t^2} = \frac{\partial^2 \varphi(t,x)}{\partial x^2}, \tag{4.2}$$

en un dominio $x \in [x_0, x_{N_x}]$, $t \geq 0$. Con condiciones iniciales $\varphi(0,x) = \varphi_0(x)$, $\frac{d}{dt}\varphi(t,x)|_{t=0} = \dot{\varphi}_0(x)$, condiciones de frontera $\varphi(t,x_0) = g_1(t)$ y $\varphi(t,x_{N_x}) = g_2(t)$. La solución general de la ecuación de onda es:

$$\varphi(t,x) = f(x+t) + g(x-t), \tag{4.3}$$

para comprobar que cumple con la ecuación (4.2) se hace el siguiente cambio de variable, $u = x + t$ y $w = x - t$, de este modo la ecuación (4.3) se transforma en:

$$\varphi(u,w) = f(u) + g(w), \tag{4.4}$$

ahora se calcula $\frac{\partial^2 \varphi}{\partial t^2}$, $\frac{\partial^2 \varphi}{\partial x^2}$:

$$\frac{\partial \varphi}{\partial t} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial t} + \frac{\partial g}{\partial w}\frac{\partial w}{\partial t} = \frac{\partial f}{\partial u} - \frac{\partial g}{\partial w} \Rightarrow \frac{\partial^2 \varphi}{\partial t^2} = \frac{\partial^2 f}{\partial u^2} + \frac{\partial^2 g}{\partial w^2},$$

$$\frac{\partial \varphi}{\partial x} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial g}{\partial w}\frac{\partial w}{\partial x} = \frac{\partial f}{\partial u} + \frac{\partial g}{\partial w} \Rightarrow \frac{\partial^2 \varphi}{\partial t^2} = \frac{\partial^2 f}{\partial u^2} + \frac{\partial^2 g}{\partial w^2},$$

lo que indica que se satisface $\frac{\partial^2 \varphi}{\partial t^2} = \frac{\partial^2 \varphi}{\partial x^2}$.

Ahora que se sabe cómo deben de ser las soluciones, se propone como solución exacta un pulso gaussiano de la forma:

$$\varphi(t, x) = \frac{A}{2}\left(e^{-\frac{(x-t)^2}{\sigma^2}} + e^{-\frac{(x+t)^2}{\sigma^2}}\right), \tag{4.5}$$

donde A es la amplitud de la onda y σ el ancho del pulso, constantes que son características del pulso y que determinan las condiciones iniciales $\varphi_0(x) = Ae^{-\frac{x^2}{\sigma^2}}$, $\dot{\varphi_0}(x) = -2\frac{x}{\sigma^2}\varphi_0(x)$. Se usa este tipo de solución ya que es suave, y se comporta como una gaussiana en el origen a $t = 0$, y que al transcurrir el tiempo se comporta como dos ondas que se van desplazando una hacia la izquierda y otra hacia la derecha con velocidad 1. Como condiciones de frontera se usan condiciones de onda saliente. Las condiciones de onda saliente se imponen para modelar fronteras abiertas que permiten el flujo hacia afuera de las soluciones pero no hacia adentro [4]. De la ecuación (4.3), se tiene que las funciones f y g arbitrarias, corresponden al desplazamiento del pulso hacia la izquierda y hacia la derecha respectivamente a lo largo de las líneas características con $t/x = cte$. Además, dichas funciones son soluciones de las ecuaciones

$$(\partial_t - \partial_x)f = 0, \tag{4.6}$$

$$(\partial_x + \partial_t)g = 0, \tag{4.7}$$

por lo tanto, la condición de onda saliente se reduce a imponer en la frontera izquierda x_0 y en la frontera derecha x_{N_x}, la condición (4.6) y la condición (4.7).

Una vez que es conocida la solución exacta (4.5), se procede a calcular la solución numérica mediante la ayuda de MPI y el MoL. Se discretiza el dominio en una malla simple, como la de la Fig. 2.1. Se realizan particiones del dominio dependiendo del número de procesadores con los que se deseé trabajar, donde cada procesador resolvera, en su dominio, una parte del problema.

Para resolver la ecuación de onda se definen nuevas variables, $\pi = \partial_t \varphi$, $\psi = \partial_x \varphi$, cuya ecuación de evolución puede construirse mediante:

$$\partial_t \psi = \partial_t(\partial_x \varphi) = \partial_x \partial_t \varphi = \partial_x \pi,$$

$$\partial_t \pi = \partial_t^2 \varphi = \partial_x^2 \varphi = \partial_x \psi,$$

teniendo en cuenta esto, la EDP original se convierte en un sistema de EDP's de primer orden en el tiempo para las nuevas variables:

$$\partial_t \varphi = \pi, \tag{4.8}$$

$$\partial_t \psi = \partial_x \pi, \tag{4.9}$$

$$\partial_t \pi = \partial_x \psi. \tag{4.10}$$

con condiciones iniciales:

$$\varphi(0, x) = A e^{-\frac{x^2}{\sigma^2}}, \tag{4.11}$$

$$\psi(0, x) = -2\frac{x}{\sigma^2}\varphi(0, x), \tag{4.12}$$

$$\pi(0, x) = 0. \tag{4.13}$$

Las condiciones de frontera se implementan usando las ecuaciones (4.9) y (4.10) que se descomponen en un modo que se mueve a la derecha ($R = 1/2(\pi - \psi)$) y otro modo que se mueve a la izquierda ($L = 1/2(\pi + \psi)$), de tal manera que para la frontera izquierda se impone la condición de la eliminación del modo que viaja hacia la derecha ($R = 0$) y para la frontera derecha se impone la condición de eliminar el modo que viaja a la izquierda ($L = 0$). Explícitamente, esto se reduce a la siguiente expresión para la frontera izquierda $x = x_0$:

$$\frac{1}{2}(\pi(t, x_0) + \psi(t, x_0)) = L_0, \tag{4.14}$$

$$\frac{1}{2}(\pi(t, x_0) - \psi(t, x_0)) = R_0 = 0, \tag{4.15}$$

cuya solución es $\pi(t, x_0) = \psi(t, x_0)) = L_0$. De manera similar, para la condición de frontera derecha $x = x_{N_x}$:

$$\frac{1}{2}(\pi(t, x_{N_x}) + \psi(t, x_{N_x})) = L_N = 0, \tag{4.16}$$

$$\frac{1}{2}(\pi(t, x_{N_x}) - \psi(t, x_{N_x})) = R_N, \tag{4.17}$$

cuya solución es $\pi(t, x_{N_x}) = R_N$, $\psi(t, x_{N_x}) = -R_N$. Por lo tanto, las condiciones de frontera son $\pi(t, x_0) = \psi(t, x_0) = L_0$ y $\pi(t, N_x) = -\psi(t, N_x) = R_N$ y el problema se reduce a calcular L_0, R_0, L_N y R_n. Para conocer estos valores se realiza una extrapolación usando los puntos internos de la malla.

El sistema (4.8) - (4.10) es apropiado para usar el MoL, es decir, se trata de un sistema de EDP de primer orden en el tiempo. La versión semidiscreta de este sistema es:

$$\partial_t \varphi = \pi_j, \tag{4.18}$$

$$\partial_t \psi = \frac{\pi_{j+1} - \pi_{j-1}}{2\Delta x} + O(\Delta x^2), \tag{4.19}$$

$$\partial_t \pi = \frac{\psi_{j+1} - \psi_{j-1}}{2\Delta x} + O(\Delta x^2), \tag{4.20}$$

donde la ecuación (4.19) y (4.20) se obtienen usando aproximaciones centradas. Se resuelve este sistema en el tiempo de forma discreta en el dominio de cada procesador, para

ello se usa el MoL, con el cual se calcula una solución para algún tiempo t^{n+1}, en términos del tiempo t^n. Se calcula la solución para cada uno de los procesadores a cada paso de tiempo, teniendo en cuenta que en las fronteras de cada procesador debe existir una comunicación, ya que de lo contrario, no es posible calcular la solución al tiempo t^{n+1}. Por ejemplo, para el procesador 0 se tiene un problema en el extremo derecho, puesto que para encontrar $\psi^{n+1}_{N_{x_0}}$ usando la expresión (4.19) es necesario conocer los valores de la función $\pi^n_{N_{x_0}+1}$ y $\pi^n_{N_{x_0}-1}$, pero no es posible calcular el valor de $\pi^n_{N_{x_0+1}}$ usando este procesador. Para resolver esto, se usa la Fig. 3.4, por medio de la cual se observa la forma en que se debe comunicar el procesador 0 y el procesador 1. Esto es, gracias a que el punto $N_{x_0} + 1$ del procesador 0, es el punto 1 del procesador 1, comunicando entonces el valor de la función de π^n_1 del procesador 1 como el valor de la función $\pi^n_{N_{x_0}+1}$ del procesador 0. Hecho este análisis, es posible calcular el valor de $\psi^{n+1}_{N_{x_0}}$, y de manera similar, se emplea esto para las fronteras de cada procesador teniendo así una manera de construir la solución a un tiempo posterior. Además, hay que tener presente que para los casos donde no exista un vecino izquierdo ni derecho, se usan las condiciones de frontera.

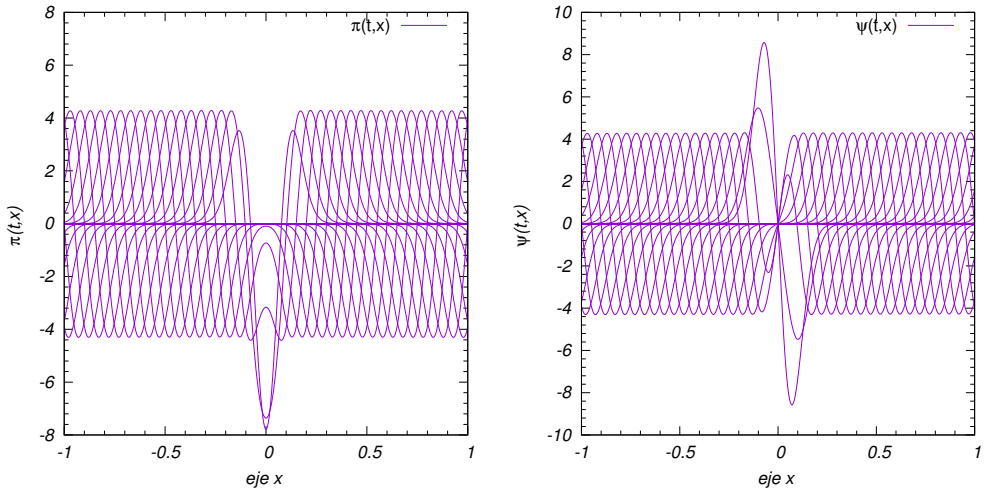
Solución de la ecuación de onda en una dimensión

Se presenta la solución de la ecuación de onda en una dimensión, la cual se resolvió usando los siguientes parámetros: $A = 1$, la amplitud de la onda, $\sigma = 0{,}1$ el ancho del pulso, $c = (\Delta t/\Delta x) = 0{,}25$ el factor de Courant. En un dominio espacial de $x \in [-1, 1]$ cubierto con 1000 puntos y un dominio temporal $t \in [0, 2]$.

Para resolver la ecuación de onda se definieron nuevas variables, con las cuales se reescribió y se obtuvó el sistema (4.18) - (4.20) por medio del cual, es posible encontrar soluciones para esas variables π y ψ como lo muestra la Fig. 4.1, donde se calcularon usando 12 procesadores. Sin embargo, la solución de la EDP original es φ, por ello, la Fig. 4.2 muestra la solución de φ, la cual también se resolvió con 12 procesadores. Se observa que al transcurrir un cierto intervalo de tiempo, la gaussiana original se transforma en dos ondas con amplitud un medio de la original, las cuales comienzan a desplazarse hacia la derecha y hacia la izquierda conforme transcurre el tiempo.

No solo se calculó la solución para este número de procesadores, sin embargo, se pudo observar que al calcular la solución con un distinto número de procesadores, esta se mantenia similar a la solución exacta, lo cual se verifica con las pruebas de convergencia.

La Fig. 4.3, muestra la solución para φ calculada con 4 procesadores, donde se puede observar la parte que cada uno de los procesadores resolvió por separado. Además, se presenta la solución para 4 valores de t, donde se aprecia mejor la evolución de la onda.



(a) $\pi(t, x)$ (b) $\psi(t, x)$

Figura 4.1: En (a) se muestra la solución de $\pi(t, x)$ para varios valores de tiempo. En (b) se muestra la solución de $\psi(t, x)$ para varios valores de tiempo, en el dominio $x \in [-1, 1]$. La solución se propaga desde el centro $x = 0$, hacia las fronteras $x = -1$ y $x = 1$, con $t = 2$, tiempo para el cual la onda ya salió del dominio espacial.

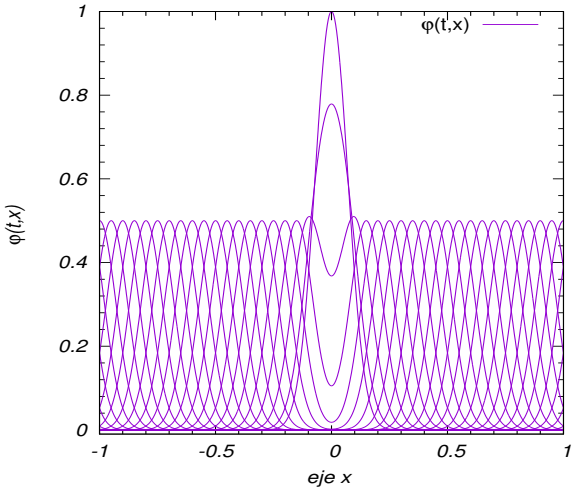
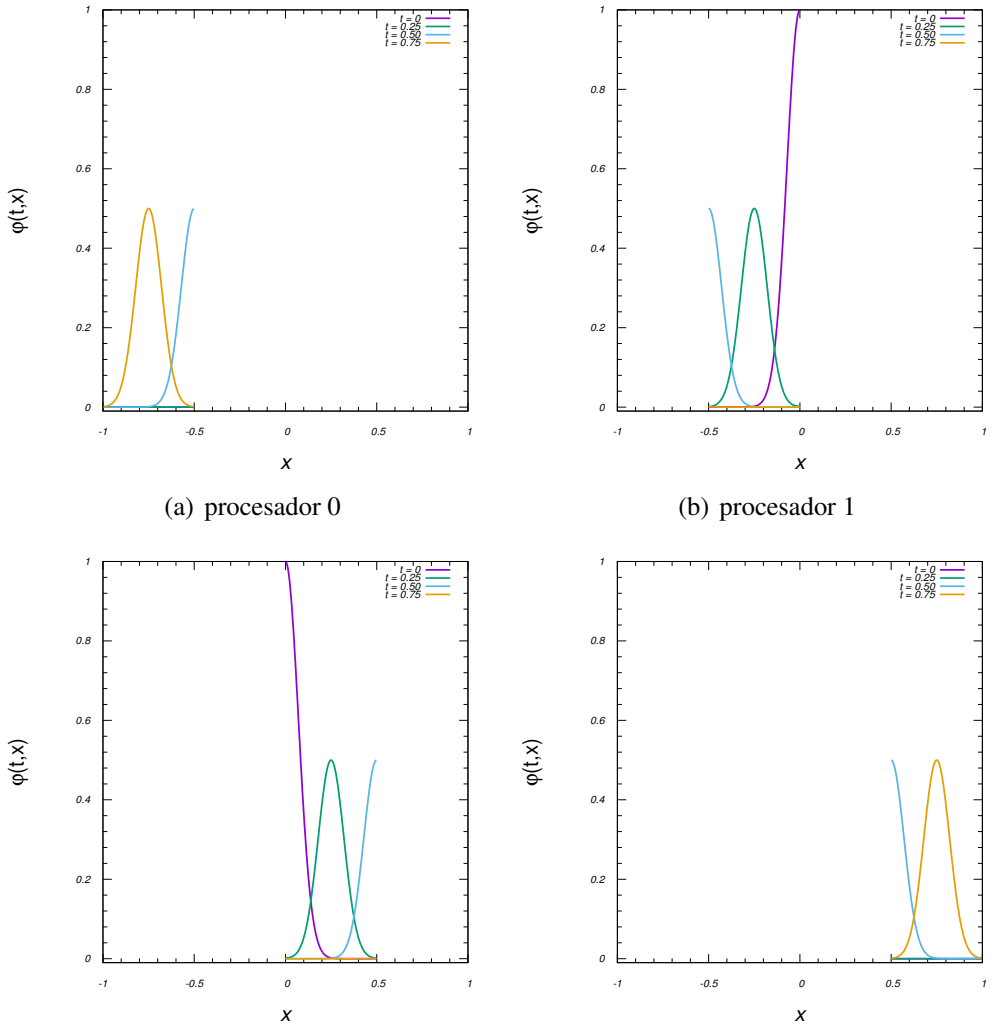


Figura 4.2: Solución de la ecuación de onda en una dimensión $\varphi(t, x)$, para varios tiempos a la vez, en el dominio espacial $x \in [-1, 1]$ y hasta $t = 2$. Al tiempo inicial la Gaussiana esta centrada en el origen y con el paso del tiempo se descompone en dos gaussianas de amplitud un medio de la original, que viajan una hacia la izquierda y otra hacia la derecha.



(a) procesador 0

(b) procesador 1

(c) procesador 2

(d) procesador 3

Figura 4.3: Solución de la ecuación de onda en una dimensión, $\varphi(t, x)$, para cuatro valores del tiempo, $t = 0$, $t = 0{,}25$, $t = 0{,}5$, $t = 0{,}75$, calculada con 4 procesadores. Donde la gráfica a) muestra la parte que resolvió el procesador 0. La gráfica b) muestra la parte que resolvió el procesador 1. La gráfica c) muestra la parte que resolvió el procesador 2. Por útlimo, la gráfica d) muestra la parte que resolvió el procesador 3.

Pruebas de convergencia

Dado que se resuelve una versión aproximada de una EDP, es necesario verificar si dicha solución converge a la solución de la ecuación en el continuo. Para esto se necesitan realizar las respectivas pruebas de convergencia usando el error global:

$$E^{glob} = \varphi_{ex} - \varphi_{num}, \tag{4.21}$$

donde φ_{ex} es la solución exacta y φ_{num} es la solución numérica de la EDP. Para saber si la solución numérica converge a la solución exacta, se hace uso de la expresión (2.22). De este modo se ejecuta el código con tres resoluciones distintas para mostrar efectivamente la convergencia a segundo orden del error. En las Figs. 4.4 y 4.5, se muestra dicho error usando como solución exacta la ecuación (4.5). Cada una de estas gráficas se elaboró usando dos distintos números de procesadores, 1 y 12, respectivamente. De igual manera para cada número de procesadores, se muestra el error para 4 distintas soluciones en el tiempo: $t = 0{,}2$, $t = 0{,}6$, $t = 0{,}8$ y $t = 1{,}0$.

También se realizaron pruebas de convergencia para la norma $L_1(e)$ y la norma $L_2(e)$ mediante las ecuaciones (2.23) y (2.24), donde dichas funciones escalares del error se monitorean como funciones del tiempo. En las Figs. 4.6 y 4.7, se muestran dichas normas para distintos números de procesadores, 1, 2, 4 y 8. Además, ambas normas al igual que el error tienen convergencia de segundo orden. Como ya se mencionó, dada la convergencia de una norma, se tiene la convergencia de la solución numérica.

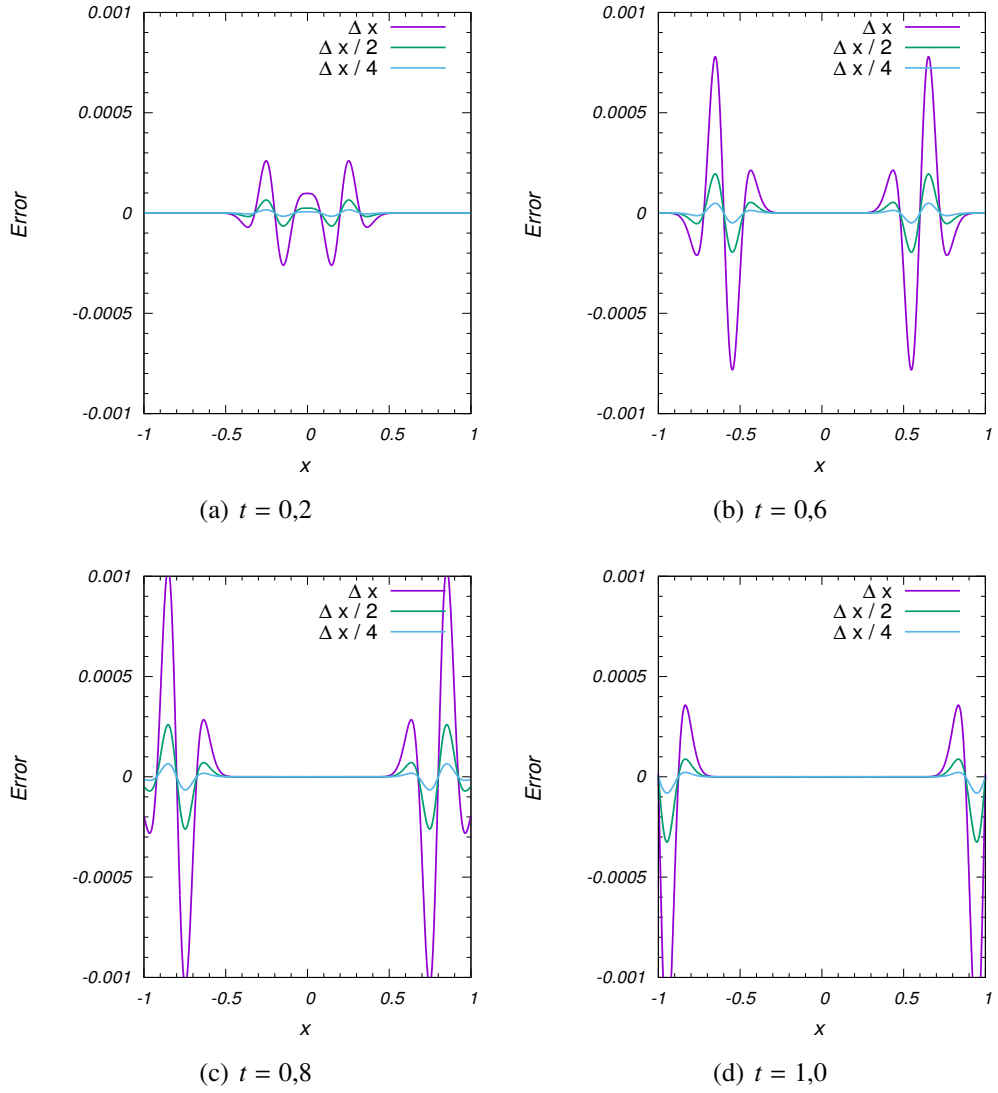
Una vez que se sabe que la solución aproximada converge a la solución exacta, mediante las pruebas correspondientes, se procede a realizar otras pruebas, las pruebas de escalamiento, que son las que se presentan a continuación.

Pruebas de escalamiento

Las pruebas de escalamiento consisten en medir el tiempo de trabajo del cálculo de las soluciones. Se hicieron 2 pruebas de escalamiento, unas salvando datos y otras sin salvar datos de las soluciones durante la ejecución del código.

Prueba 1: Sin salvar datos durante la ejecución.

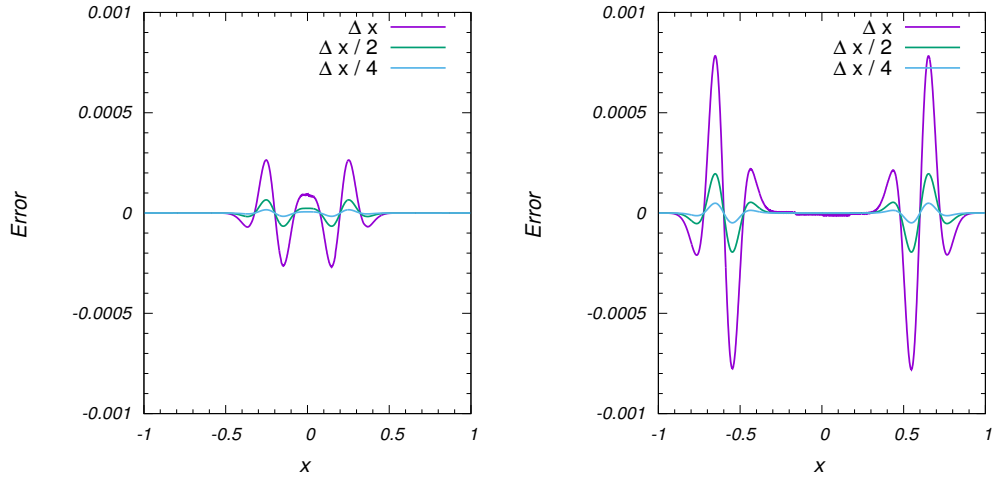
- Se hicieron las corridas para 1, 2, 4, 8 y 12 procesadores.
- Se hicieron 3 corridas para cada número de procesadores, tomando solo el tiempo de trabajo.
- Se calculó un promedio del tiempo de las 3 corridas.
- Se salvaron los datos solo para $t = 0$ y para $t = 2$.



(a) $t = 0{,}2$ (b) $t = 0{,}6$

(c) $t = 0{,}8$ (d) $t = 1{,}0$

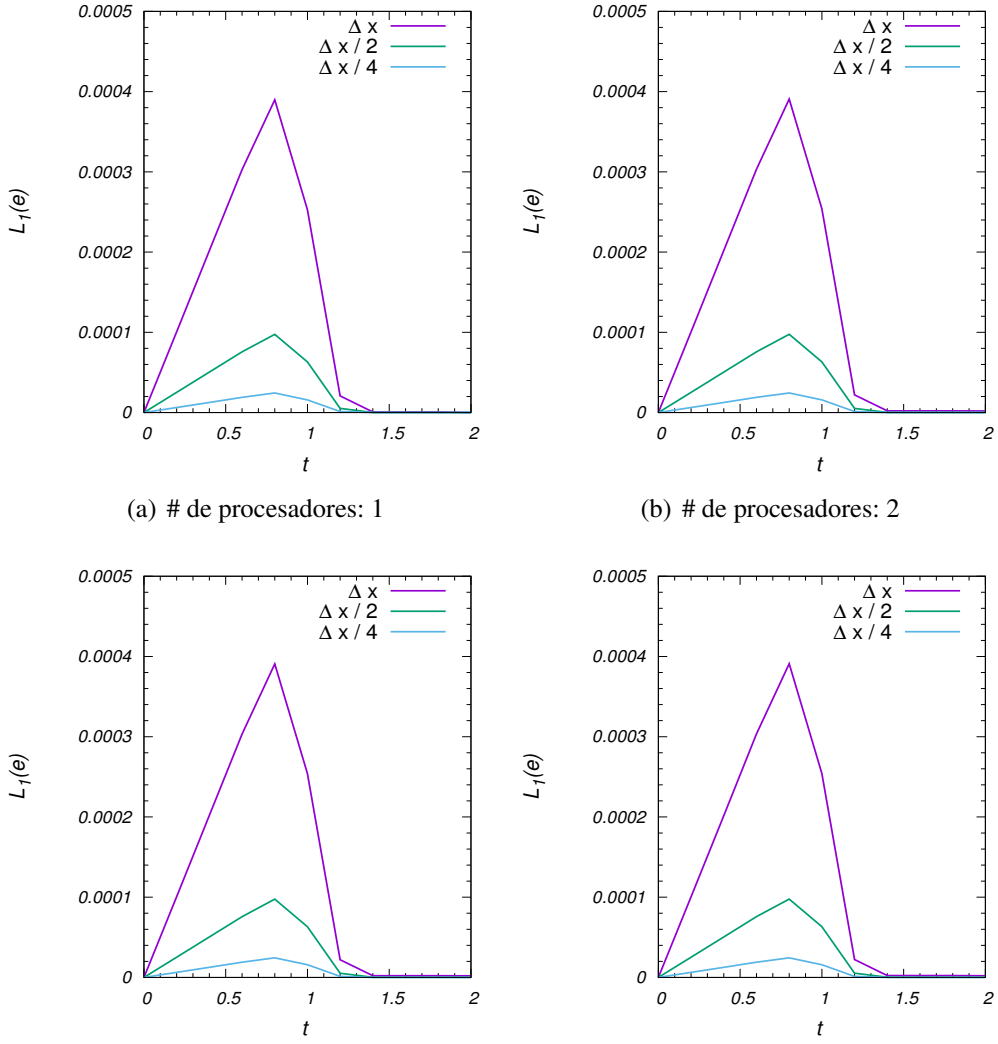
Figura 4.4: Gráficas del error al trabajar con 1 procesador para diferentes valores de tiempo. Se calcula el error con 3 resoluciones diferentes, la curva (morada) muestra el error cuando se usa una resolución de $\Delta x = 0{,}002$, la curva (verde) corresponde a $\Delta x/2$ y por último, la curva (azul) ilustra el caso para una resolución $\Delta x/4$. Se verifica que las curvas se superponen cuando se multiplica la curva (verde) por un factor de 4 y la curva (azul) por un factor de 16, por tal motivo es posible afirmar, que se tiene convergencia de segundo orden trabajando con 1 procesador.



(a) $t = 0{,}2$ (b) $t = 0{,}6$

(c) $t = 0{,}8$ (d) $t = 1{,}0$

Figura 4.5: Gráficas del error al trabajar con 12 procesador para diferentes valores de tiempo. Se calcula el error con 3 resoluciones diferentes, la curva (morada) muestra el error cuando se usa una resolución de $\Delta x = 0{,}002$, la curva (verde) corresponde a $\Delta x/2$ y por último, la curva (azul) ilustra el caso para una resolución $\Delta x/4$. Se verifica que las curvas se superponen cuando se multiplica la curva (verde) por un factor de 4 y la curva (azul) por un factor de 16, por tal motivo es posible afirmar, que se tiene convergencia de segundo orden trabajando con 12 procesadores.



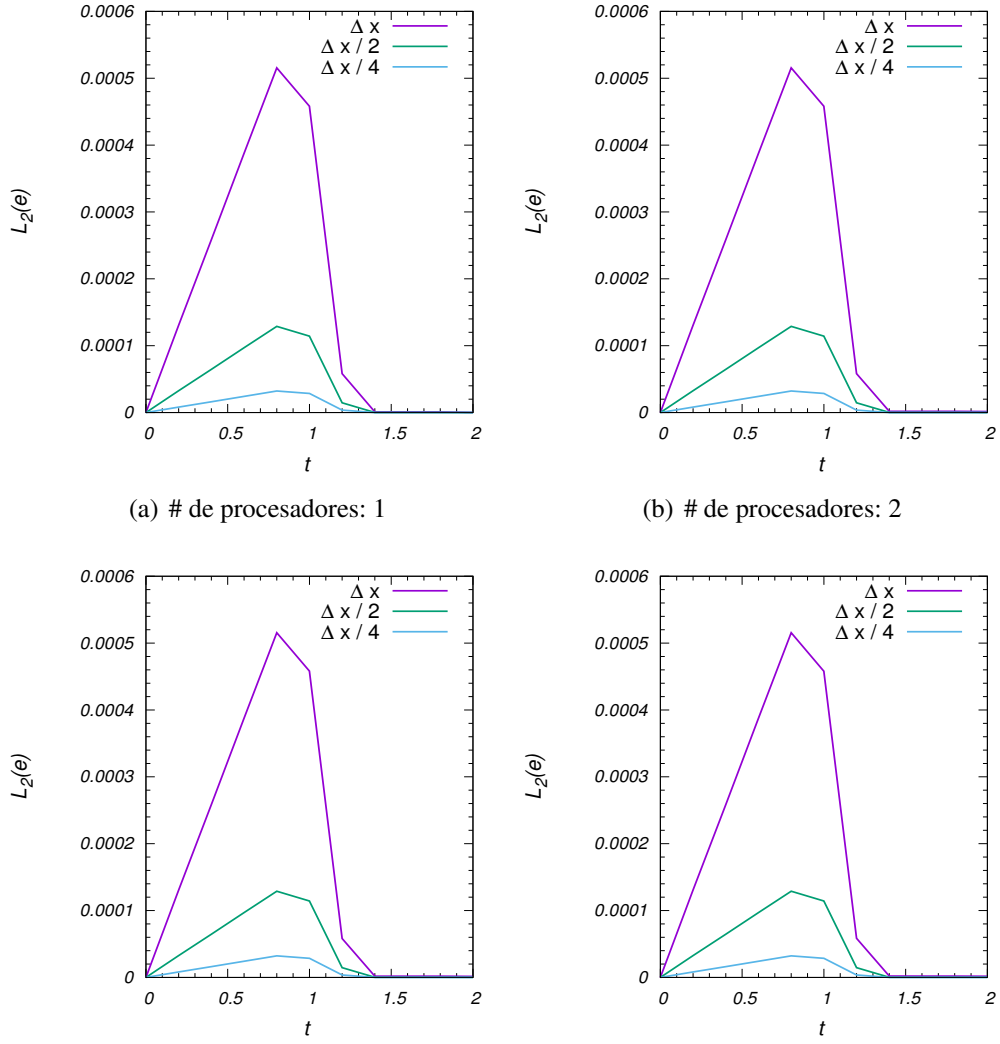
(a) # de procesadores: 1

(b) # de procesadores: 2

(c) # de procesadores: 4

(d) # de procesadores: 8

Figura 4.6: La norma L_1 del error calculada para todo el dominio temporal, $t \in [0, 2]$. Se calcula dicha norma con 1, 2, 4 y 8 procesadores. Se calcula cada gráfica del error con 3 resoluciones diferentes, la curva (morada) muestra la norma del error cuando se usa una resolución de $\Delta x = 0{,}002$, la curva (verde) corresponde a $\Delta x/2$ y la curva (azul) con una resolución $\Delta x/4$. Se verifica que las curvas se superponen cuando se multiplica la curva (verde) por un factor de 4 y (azul) por un factor de 16, por tal motivo es posible afirmar, que se tiene convergencia de segundo orden para la norma L_1.



(a) # de procesadores: 1

(b) # de procesadores: 2

(c) # de procesadores: 4

(d) # de procesadores: 8

Figura 4.7: La norma L_2 del error calculada para todo el dominio temporal, $t \in [0,2]$. Se calcula dicha norma con 1, 2, 4 y 8 procesadores. Se calcula cada gráfica del error con 3 resoluciones diferentes, la curva (morada) muestra la norma del error cuando se usa una resolución de $\Delta x = 0{,}002$, la curva (verde) corresponde a $\Delta x/2$ y la curva (azul) con una resolución $\Delta x/4$. Se verifica que las curvas se superponen cuando se multiplica la curva (verde) por un factor de 4 y la curva (azul) por un factor de 16, por tal motivo es posible afirmar, que se tiene convergencia de segundo orden para la norma L_2.

Prueba 2: Salvando datos durante la ejecución:

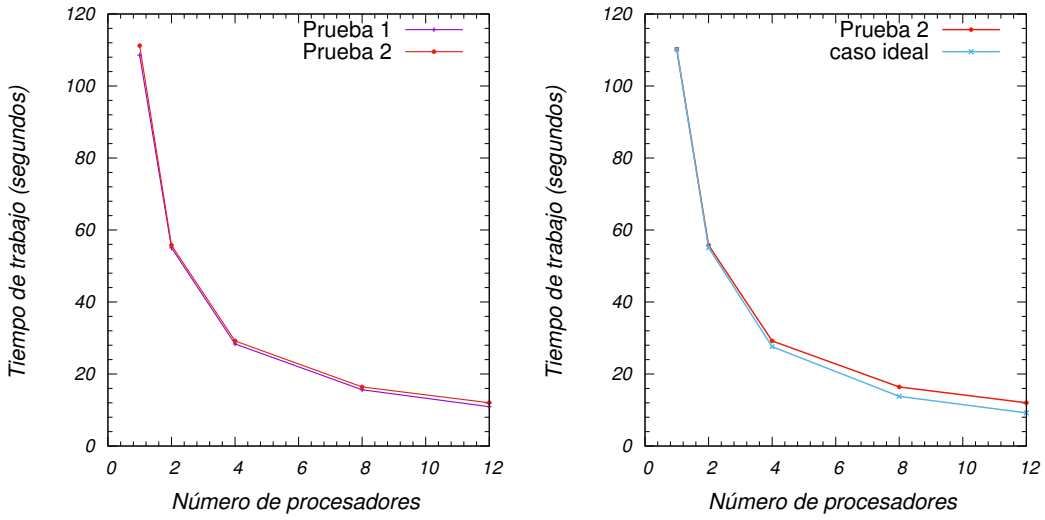
- Se hicieron las corridas para 1, 2, 4, 8 y 12 procesadores.
- Se hicieron 3 corridas para cada número de procesadores, tomando el tiempo de trabajo.
- Se calculó un promedio del tiempo de las 3 corridas.
- Se salvaron los datos cada 4000 iteraciones de 40000 en total.

Para tomar el tiempo se usa el comando MPI_WTIME, el cual arroja el valor del tiempo en segundos. Se trabaja en el dominio espacial $x \in [-1, 1]$, cubierto con 10000 puntos y se cubre el dominio temporal con 40000 iteraciones, teniendo entonces $\Delta x = 0{,}0002$ y $\Delta t = c\Delta x = 0{,}00005$. La Fig. 4.8 (a), muestra la prueba 1 y la prueba 2, de donde comparando el tiempo al trabajar con 1 procesador y con 2 procesadores, se observa que este último se reduce a la mitad. De igual manera, al comparar el tiempo trabajando con 2 procesadores y el tiempo trabajando con 4 procesadores, también se observa que el tiempo se reduce aproximadamente a la mitad. Siguiendo este comportamiento, el caso ideal sería que al trabajar con 8 procesadores, el tiempo se redujera nuevamente a la mitad. El cuadro 4.1 muestra los valores de la prueba 2 y los valores del caso ideal, donde además se puede observar la diferencia de ambos. La Fig 4.8 (b) muestra estos dos casos, la prueba 2 y el caso ideal.

De manera similar, se realizó otro cálculo de estas pruebas con mayor demanda de memoria RAM. Para ello, se aumentó el número de puntos del dominio a 100000, con $\Delta x = 2 \times 10^{-5}$, teniendo entonces $\Delta t = c\Delta x = 5 \times 10^{-5}$ y ($N_t = t/\Delta t = 400000$) el número de iteraciones. Se realizaron la prueba 1 y prueba 2, salvando para esta última los datos cada 40000 iteraciones. La Fig. 4.9 muestra estas pruebas, donde comparando el tiempo al trabajar con 1 procesador con el tiempo al trabajar con 2 procesadores, se observa que este último se reduce casi a la mitad. Sin embargo, esta prueba muestran rápidamente que no se tiene un comportamiento lineal como el esperado y esto se observa al comparar la gráfica de 2 y 4 procesadores. Para el caso ideal se muestra el cuadro 4.2, que ilustra también una diferencia.

Conviene mencionar el comparativo entre la prueba 1 y la prueba 2, lo primero es observar que los tiempos de la prueba 1 son menores que los de la prueba 2, otra cosa es que las curvas se comportan de manera similar, es decir, comparando los tiempos al trabajar con 1 y con 2 procesadores, se observa que el tiempo de trabajo se redice a la mitad, sin embargo al seguir observando el comportamiento de la curva, se muestra que esta relación ya no se reduce a la mitad al trabajar con el doble número de procesadores.

Esto se debe, a que al trabajar con un mayor número de procesadores, también se tiene una mayor comunicación, lo cual implica un gasto de tiempo al comunicar muchos procesadores.

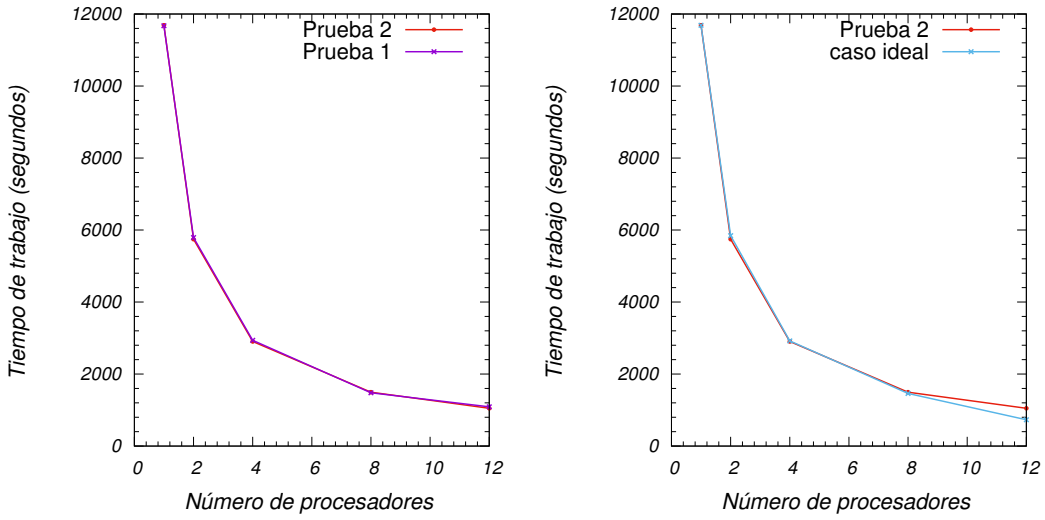


(a) Prueba 1 - Prueba 2 (b) Prueba 2 - Caso Ideal

Figura 4.8: Pruebas de escalamiento para un dominio de $x \in [-1, 1]$ con $\Delta x = 0{,}0002$, cubierto con 10000 puntos, usando 40000 iteraciones ($t = 2$). (a) La curva (morada) muestra el tiempo que tarda la ejecución de la prueba 1 en función del número de procesadores, mientras que la curva (roja) muestra el tiempo que tarda la ejecución de la prueba 2 en función del número de procesadores. (b) La curva (roja) muestra el tiempo que tarda la ejecución de la prueba 2 y la curva (azul) ilustra el tiempo que debe tardar para el caso ideal.

Cuadro 4.1: Prueba de escalamiento para la ecuación de onda en una dimensión cubierta de 10000 puntos, se comparan los tiempos de escalamiento con el caso ideal y se estima una diferencia.

Número de procesadores	Tiempo real	Tiempo ideal	Diferencia
1 procesador	$t = 110{,}2s$	$t = 110{,}2s$	$t = 0s$
2 procesadores	$t = 55{,}8s$	$t = 55{,}1s$	$t = 0{,}7s$
4 procesadores	$t = 29{,}2s$	$t = 27{,}6s$	$t = 1{,}6s$
8 procesadores	$t = 16{,}4s$	$t = 13{,}8s$	$t = 2{,}6s$
12 procesadores	$t = 12{,}0s$	$t = 9{,}2s$	$t = 2{,}8s$



(a) Prueba 1 - Prueba 2 (b) Prueba 2 - Caso Ideal

Figura 4.9: Pruebas de escalamiento para el dominio $x \in [-1, 1]$ con $\Delta x = 2 \times 10^{-5}$, cubierto con 100000 puntos, para 400000 iteraciones ($t = 2$). En (a) la curva color rojo muestra el tiempo que tarda la ejecución de la prueba 2 en función del número de procesadores y la curva (morada) muestra el tiempo que tarda la ejecución de la prueba 1 en función del número de procesadores. (b) La curva (roja) muestra el tiempo que tarda la ejecución de la prueba 2, mientras que la curva (azul) muestra el tiempo para el caso ideal del cuadro 4.2.

Cuadro 4.2: Prueba de escalamiento para la ecuación de onda en una dimensión cubierta de 100000 puntos, donde se comparan los tiempos de escalamiento con el caso ideal y se estima un error comparativo.

Número de procesadores	Tiempo real	Tiempo ideal	Diferencia
1 procesador	$t = 11687s$	$t = 11687s$	$t = 0s$
2 procesadores	$t = 5745{,}8s$	$t = 5843{,}5$	$t = 97{,}7s$
4 procesadores	$t = 2905{,}1s$	$t = 2921{,}8s$	$t = 635{,}3s$
8 procesadores	$t = 1296{,}2s$	$t = 1460{,}9s$	$t = 163{,}8s$
12 procesadores	$t = 1049{,}8s$	$t = 730{,}45s$	$t = 319{,}4s$

4.2. La ecuación de onda en tres dimensiones

Una vez resuelta la ecuación de onda unidimensional, es momento de pasar al siguiente problema, el cual es resolver la ecuación de onda para 3 dimensiones espaciales. La razón es que problemas en tres dimensiones requieren de uso de cómputo en paralelo, ya que estos problemas trabajan con un número N^3 de puntos, los cuales necesitan de más memoria RAM. La ecuación de onda en tres dimensiones es:

$$\frac{\partial^2\varphi(t,x,y,z)}{\partial t^2} = \nabla^2\varphi(t,x,y,z), \tag{4.22}$$

donde el operador ∇^2 es el operador Laplaciano. Se resolverá la ecuación de onda en un dominio espacial $[x_0, x_{N_x}] \times [y_0, y_{N_y}] \times [z_0, z_{N_z}]$ y un dominio temporal, $t \in [0, t^{N_t}]$. Con condiciones iniciales que se propagan como onda esférica, esto es:

$$\varphi(0,x,y,z) = \frac{A}{2}e^{-\left(\frac{x^2+y^2+z^2}{\sigma^2}\right)}, \tag{4.23}$$

$$\dot{\varphi}(0,x,y,z) = 0, \tag{4.24}$$

con A la amplitud de la onda y σ el ancho del pulso. Como condición de frontera se usa lo siguiente:

$$\varphi(t,x_0,y,z) = \varphi(t,x_1,y,z), \varphi(t,x_{N_x},y,z) = \varphi(t,x_{N_x-1},y,z),$$
$$\varphi(t,x,y_0,z) = \varphi(t,x,y_1,z), \varphi(t,x,y_{N_y},z) = \varphi(t,y_{N_y-1},z),$$
$$\varphi(t,x,y,z_0) = \varphi(t,x,y,z_1), \varphi(t,x,y,z_{N_z}) = \varphi(t,x,y,z_{N_x-1}),$$

es decir, los valores de las fronteras se reemplazan por valores de los puntos vecinos.

Antes de resolver el problema numérico veamos la solución exacta, para ello, se usa el operador laplaciano en coordenadas esféricas:

$$\nabla^2\varphi = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\varphi}{\partial r}\right) + \frac{1}{r^2\text{sen}\theta}\frac{\partial}{\partial\theta}\left(\text{sen}\theta\frac{\partial\varphi}{\partial\theta}\right) + \frac{1}{r^2\text{sen}^2\theta}\frac{\partial^2\varphi}{\partial\phi^2}. \tag{4.25}$$

Como se trata de un problema esféricamente simétrico, la dependencia en φ y θ se elimina y solamente se considera la parte radial:

$$\nabla^2\varphi = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\varphi}{\partial r}\right). \tag{4.26}$$

Para simplificar las cosas se hace un cambio de variable $\psi = r\varphi$, con $r \neq 0$ y se obtiene

$$\begin{aligned}
\nabla^2\left(\frac{\psi}{r}\right) &= \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\left(\frac{\psi}{r}\right)}{\partial r}\right) \\
&= \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\left(\frac{1}{r}\frac{\partial\psi}{\partial r}\right) + \psi\left(-\frac{1}{r^2}\right)\right) \\
&= \frac{1}{r^2}\frac{\partial}{\partial r}\left(r\frac{\partial\psi}{\partial r} - \psi\right) \\
&= \frac{1}{r^2}\left(\frac{\partial\psi}{\partial r} + r\frac{\partial^2\psi}{\partial r^2} - \frac{\partial\psi}{\partial r}\right) \\
&= \frac{1}{r}\left(\frac{\partial^2\psi}{\partial r^2}\right),
\end{aligned}$$

entonces se tiene

$$\nabla^2\left(\frac{\psi}{r}\right) = \frac{1}{r}\left(\frac{\partial^2\psi}{\partial r^2}\right), \tag{4.27}$$

sustituyendo en la ecuación (4.22)

$$\frac{\partial^2\left(\frac{\psi}{r}\right)}{\partial t^2} = \nabla^2\left(\frac{\psi}{r}\right) = \frac{1}{r}\left(\frac{\partial^2\psi}{\partial r^2}\right), \tag{4.28}$$

y al final se obtiene:

$$\frac{\partial^2\psi}{\partial t^2} = \frac{\partial^2\psi}{\partial r^2}, \tag{4.29}$$

que es igual al caso de una dimensión. Por tal motivo y con ayuda de la ecuación (4.3) se sabe que la solución es de la forma

$$\psi(t, r) = f(r + t) + g(r - t), \tag{4.30}$$

para datos iniciales que se propagan como una onda esférica, se propone una solución de tipo pulso gaussiano

$$\psi(t, r) = \frac{A}{2}\left((r - t)e^{-\frac{(r-t)^2}{\sigma^2}} + (r + t)e^{-\frac{(r+t)^2}{\sigma^2}}\right), \tag{4.31}$$

de modo que para $\varphi = \psi/r$, se tiene:

$$\varphi(t, r) = \frac{A}{2r}\left((r - t)e^{-\frac{(r-t)^2}{\sigma^2}} + (r + t)e^{-\frac{(r+t)^2}{\sigma^2}}\right), \tag{4.32}$$

con $r \neq 0$, la cual satisface la ecuación (4.22), con A y σ la amplitud y el ancho de la gaussiana respectivamente. Se usa este tipo de solución por el comportamiento suave de la solución. Sin embargo, la ecuación (4.32) no esta definida para $r = 0$, por lo tanto, para construir la solución en dicho punto, se hace una expansión en serie de Taylor de la funcion exponencial en torno a este punto y se obtiene:

$$\begin{aligned}
e^{-\frac{(r-t)^2}{\sigma^2}} &= \left[e^{-\frac{(r_0-t)^2}{\sigma^2}}\right]_{r_0=0} - \left[2(r-r_0)\frac{(r_0-t)}{\sigma^2}e^{-\frac{(r_0-t)^2}{\sigma^2}}\right]_{r_0=0} \\
&+ \left[4\frac{(r-r_0)^2}{2!}\frac{(r_0-t)^2}{\sigma^4}e^{-\frac{(r_0-t)^2}{\sigma^2}}\right]_{r_0=0} \\
&- \left[\frac{2}{\sigma^2}\frac{(r-r_0)^2}{2!}e^{-\frac{(r_0-t)^2}{\sigma^2}}\right]_{r_0=0} + O((r-r_0)^3) \\
&\approx e^{-\frac{t^2}{\sigma^2}} - 2(r)\frac{(-t)}{\sigma^2}e^{-\frac{t^2}{\sigma^2}} + 4\frac{r^2}{2}\frac{t^2}{\sigma^4}e^{-\frac{t^2}{\sigma^2}} - \frac{2}{\sigma^2}\frac{r^2}{2!}e^{-\frac{t^2}{\sigma^2}},
\end{aligned}$$

$$\begin{aligned}
e^{-\frac{(r+t)^2}{\sigma^2}} &= \left[e^{-\frac{(r_0+t)^2}{\sigma^2}}\right]_{r_0=0} - \left[2(r-r_0)\frac{(r_0+t)}{\sigma^2}e^{-\frac{(r_0+t)^2}{\sigma^2}}\right]_{r_0=0} \\
&+ \left[4\frac{(r-r_0)^2}{2!}\frac{(r_0+t)^2}{\sigma^4}e^{-\frac{(r_0+t)^2}{\sigma^2}}\right]_{r_0=0} \\
&- \left[\frac{2}{\sigma^2}\frac{(r-r_0)^2}{2!}e^{-\frac{(r_0+t)^2}{\sigma^2}}\right]_{r_0=0} + O((r-r_0)^3) \\
&\approx e^{-\frac{t^2}{\sigma^2}} - 2(r)\frac{t}{\sigma^2}e^{-\frac{t^2}{\sigma^2}} + 4\frac{r^2}{2}\frac{t^2}{\sigma^4}e^{-\frac{t^2}{\sigma^2}} - \frac{2}{\sigma^2}\frac{r^2}{2!}e^{-\frac{t^2}{\sigma^2}},
\end{aligned}$$

sustituyendo en φ, se tiene:

$$\begin{aligned}
\varphi(t,r) &= \frac{A}{2r}(r-t)\left(e^{-\frac{t^2}{\sigma^2}} + \frac{2rt}{\sigma^2}e^{-\frac{t^2}{\sigma^2}} + 4\frac{r^2}{2}\frac{t^2}{\sigma^4}e^{-\frac{t^2}{\sigma^2}} - \frac{2}{\sigma^2}\frac{r^2}{2!}e^{-\frac{t^2}{\sigma^2}}\right) \\
&+ \frac{A}{2r}(r+t)\left(e^{-\frac{t^2}{\sigma^2}} - \frac{2rt}{\sigma^2}e^{-\frac{t^2}{\sigma^2}} + 4\frac{r^2}{2}\frac{t^2}{\sigma^4}e^{-\frac{t^2}{\sigma^2}} - \frac{2}{\sigma^2}\frac{r^2}{2!}e^{-\frac{t^2}{\sigma^2}}\right) \\
&= \frac{A}{r}re^{-\frac{t^2}{\sigma^2}}\left(1 + 4\frac{r^2}{2}\frac{t^2}{\sigma^4} - \frac{2}{\sigma^2}\frac{r^2}{2!}\right) - \frac{2rt}{\sigma^2}\frac{A}{r}te^{-\frac{t^2}{\sigma^2}} \\
&= Ae^{-\frac{t^2}{\sigma^2}}\left(1 + 4\frac{r^2}{2}\frac{t^2}{\sigma^4} - \frac{2}{\sigma^2}\frac{r^2}{2!} - \frac{2t^2}{\sigma^2}\right)
\end{aligned}$$

Por lo tanto, para valores de φ cercanos a $r = 0$ la solución de la ecuación de onda es:

$$\varphi = Ae^{-\frac{t^2}{\sigma^2}} - \frac{2At^2}{\sigma^2}e^{-\frac{t^2}{\sigma^2}}. \tag{4.33}$$

Una vez calculada la solución exacta de la ecuación de onda, se procede a calcular la solución numérica usando MPI y el MoL. Para hacer esto se define un dominio discreto como el de la Fig. 3.5, con $x_i = x_0 + i\Delta x$, $y_j = y_0 + j\Delta y$, $z_k = z_0 + k\Delta z$, $\Delta x = (x_{N_x} - x_0)/N_x$, $\Delta y = (y_{N_y} - y_0)/N_y$, $\Delta x = (z_{N_z} - z_0)/N_z$, con $i = 0, N_x$, $j = 0, N_y$, $k = 0, N_z$. De igual modo, para trabajar de forma paralelizada, se realizan particiones del dominio en función del número de procesadores.

Es conveniente regresar a la ecuación (4.22) y simplificar la notación de la siguiente manera:

$$\partial_t^2\varphi = \nabla^2\varphi = \partial_x^2\varphi + \partial_y^2\varphi + \partial_z^2\varphi. \tag{4.34}$$

Para resolverla se definen nuevas variables: $\pi = \partial_t\varphi$, $\psi_x = \partial_x\varphi$, $\psi_y = \partial_y\varphi$, $\psi_z = \partial_z\varphi$, de modo que las ecuaciones de evolución para φ, π, ψ_x, ψ_y, ψ_z son

$$\partial_t\varphi = \pi, \tag{4.35}$$

$$\partial_t\pi = \partial_t^2\varphi = \partial_x^2\varphi + \partial_y^2\varphi + \partial_z^2\varphi = \partial_x\psi_x + \partial_y\psi_y + \partial_z\psi_z, \tag{4.36}$$

$$\partial_t\psi_x = \partial_t\partial_x\varphi = \partial_x\partial_t\varphi = \partial_x\pi, \tag{4.37}$$

$$\partial_t\psi_y = \partial_t\partial_y\varphi = \partial_y\partial_t\varphi = \partial_y\pi, \tag{4.38}$$

$$\partial_t\psi_z = \partial_t\partial_z\varphi = \partial_z\partial_t\varphi = \partial_z\pi. \tag{4.39}$$

Ahora, este conjunto de 5 ecuaciones de evolución, tiene las condiciones iniciales

$$\varphi(0, x, y, z) = Ae^{-\frac{x^2}{\sigma^2}-\frac{y^2}{\sigma^2}-\frac{z^2}{\sigma^2}}, \tag{4.40}$$

$$\varphi_x(0, x, y, z) = \partial_x\varphi_0 = 2A\frac{x}{\sigma^2}e^{-\frac{x^2}{\sigma^2}-\frac{y^2}{\sigma^2},-\frac{z^2}{\sigma^2}}, \tag{4.41}$$

$$\varphi_y(0, x, y, z) = \partial_y\varphi_0 = 2A\frac{y}{\sigma^2}e^{-\frac{x^2}{\sigma^2}-\frac{y^2}{\sigma^2},-\frac{z^2}{\sigma^2}}, \tag{4.42}$$

$$\varphi_z(0, x, y, z) = \partial_z\varphi_0 = 2A\frac{z}{\sigma^2}e^{-\frac{x^2}{\sigma^2}-\frac{y^2}{\sigma^2},-\frac{z^2}{\sigma^2}}, \tag{4.43}$$

$$\pi(0, x, y, z) = 0, \tag{4.44}$$

cuya versión semidiscreta del sistema es:

$$[\partial_t\varphi]^n_{i,j,k} = \pi^n_{i,j,k}, \tag{4.45}$$

$$[\partial_t\pi]^n_{i,j,k} = \frac{\psi_{x_{i+1,j,k}} - \psi_{x_{i-1,j,k}}}{2\Delta x} + \frac{\psi_{y_{i,j+1,k}} - \psi_{y_{i,j-1,k}}}{2\Delta y} + \frac{\psi_{z_{i,j,k+1}} - \psi_{z_{i,j,k-1}}}{2\Delta z}, \tag{4.46}$$

$$[\partial_t \psi_x]^n_{i,j,k} = \frac{\pi_{i+1,j,k} - \pi_{i-1,j,k}}{2\Delta x}, \tag{4.47}$$

$$[\partial_t \psi_y]^n_{i,j,k} = \frac{\pi_{i,j+1,k} - \pi_{i,j-1,k}}{2\Delta y}, \tag{4.48}$$

$$[\partial_t \psi_z]^n_{i,j,k} = \frac{\pi_{i,j,k+1} - \pi_{i,j,k-1}}{2\Delta z}, \tag{4.49}$$

el cual es un conjunto de ecuaciones diferenciales en el tiempo en forma discreta que se resuelve con ayuda del MoL. Dado que se trabaja de forma paralelizada y que la partición del dominio total se realizó a lo largo del eje z, entonces solo los operadores que tienen derivada respecto a z serán los que se estaran comunicando, que son el tercer término del lado derecho de la ecuación (4.46) y el término derecho de la ecuación (4.49), donde la comunicación se realiza por medio de las fronteras del eje z de cada uno de los procesadores, teniendo en cuenta que la comunicación no es punto a punto, sino que se comunican planos de $N_x N_y$ puntos.

Solución de la ecuación de onda en tres dimensiones

Se presenta la solución de la ecuación de onda para tres dimensiones espaciales, la cual es resuelta para los siguientes parámetros: la amplitud de la onda, $A = 1$, el ancho del pulso, $\sigma = 0{,}1$ y el factor de Courant, $c = \Delta t/min(\Delta x, \Delta y, \Delta z) = 0{,}50$. Se trabaja para un dominio espacial $x \in [-1, 1]$, $y \in [-1, 1]$, $z \in [-1, 1]$, cubierto con 256 puntos para cada eje, es decir 256^3 y usando un dominio temporal $t \in [0, 1]$. Al resolver esta ecuación con un diferente número de procesadores, se observó que la solución no depende del número de procesadores, por tanto, se muestra el comportamiento trabajando con 12 procesadores. A su vez, dado que el sistema es un sistema acoplado, es posible tener soluciones para π, ψ_x, ψ_y, ψ_z y φ, sin embargo, la solución de interes es φ, puesto que esta es la variable de la EDP original. La Fig. 4.10, muestra la condición inicial, $t = 0$, mientras que en la Fig. 4.11 se observa cómo va evolucionando la onda a $t = 0{,}125$ de igual manera la Fig. 4.12 muestra la solución a $t = 0{,}25$, y por último, la Fig. 4.13 muestra la evolución de la onda a $t = 0{,}50$. Se observa cómo comienza la onda a propagarse, y ya que se calcularon estas soluciones con 12 procesadores, es posible observar la parte que cada uno de los procesadores resuelve.

A modo de esclarecer el comportamiento de la onda, en la Fig. 4.14, se muestra el comportamiento de φ a lo largo del eje y. Donde a $t = 0$, se observa un comportamient similar al caso unidimensional, sin embargo, en este caso, la amplitud de la onda decrece en función del tiempo como se puede observar de la solución exacta (4.31).

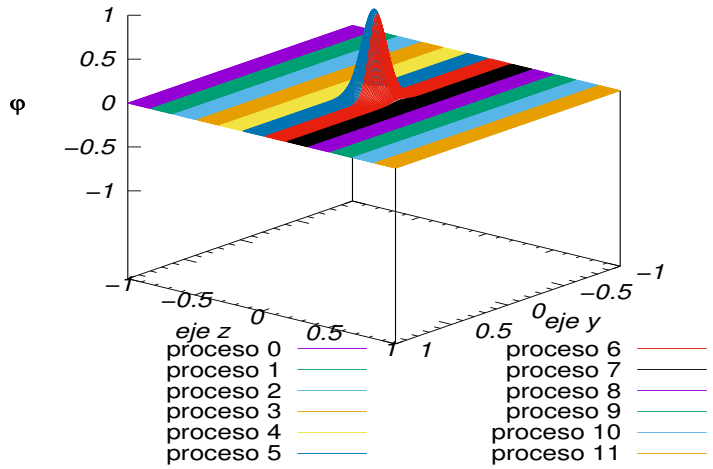


Figura 4.10: Condición inicial de la solución de la ecuación de onda en tres dimensiones proyectada sobre el plano yz, para un dominio de $[-1,1]\times[-1,1]\times[-1,1]$, cubierto con 256^3 puntos. Los colores representan la parte que calculó cada uno de los procesadores para $t = 0$.

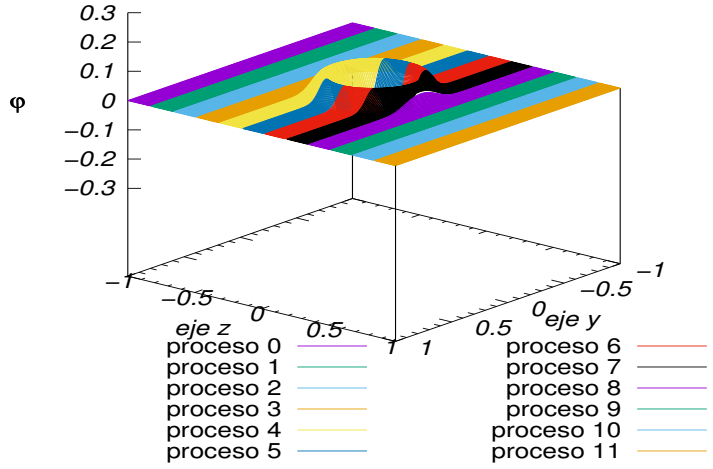


Figura 4.11: Solución de la ecuación de onda en tres dimensiones proyectada sobre el plano yz, para un dominio de $[-1,1]\times[-1,1]\times[-1,1]$, cubierto con 256^3 puntos. Los colores representan la parte que cálculo cada uno de los procesadores para $t = 0{,}25$.

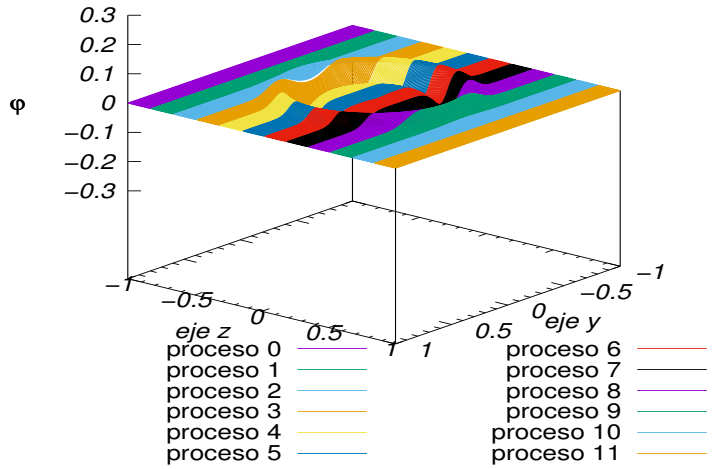


Figura 4.12: Solución de la ecuación de onda en tres dimensiones proyectada sobre el plano yz, para un dominio de $[-1,1]\times[-1,1]\times[-1,1]$, cubierto con 256^3 puntos. Los colores representan la parte que cálculo cada uno de los procesadores para $t = 0{,}50$.

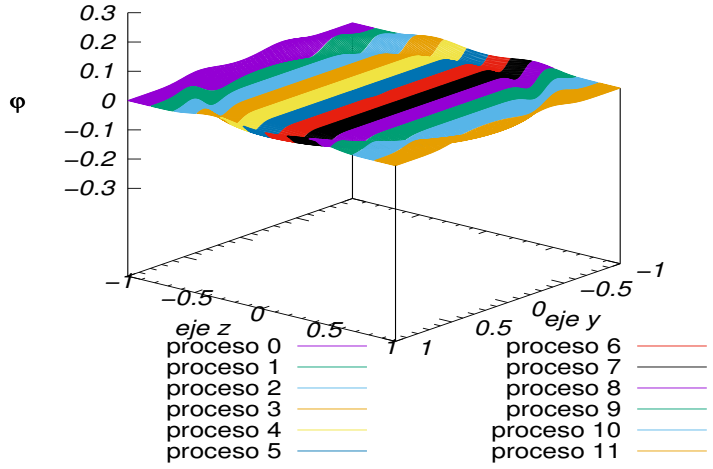
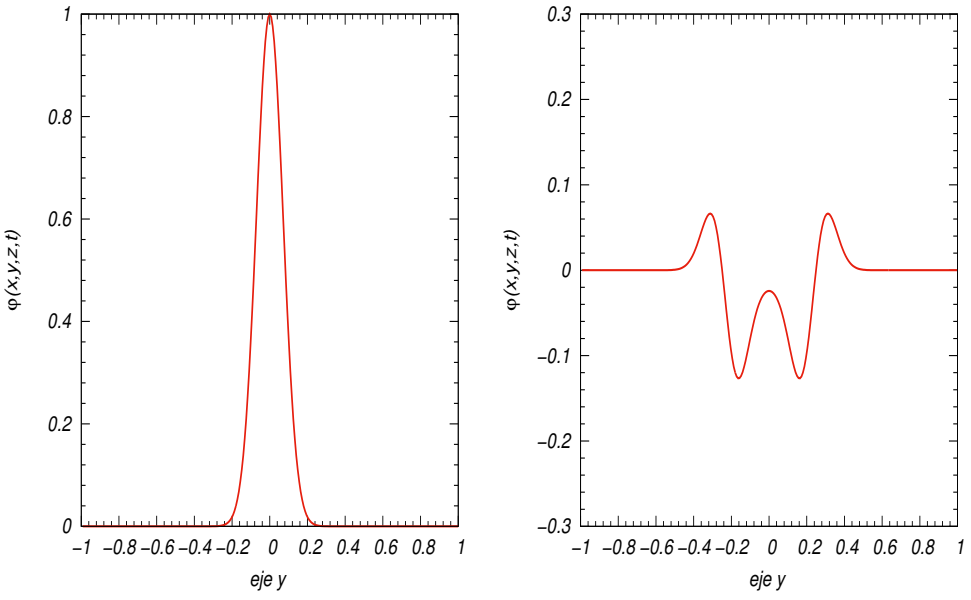
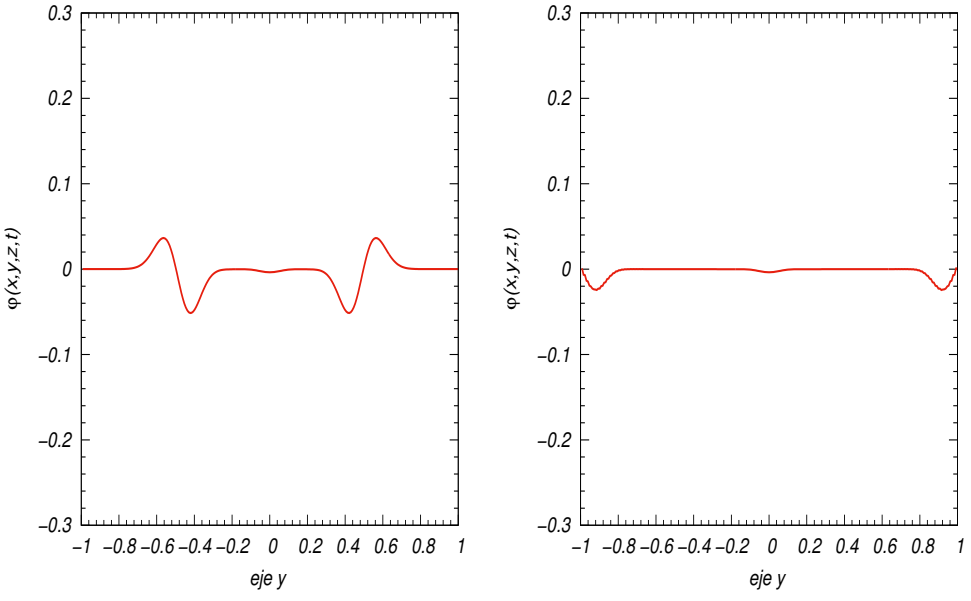


Figura 4.13: Solución de la ecuación de onda en tres dimensiones proyectada sobre el plano yz, para un dominio de $[-1,1]\times[-1,1]\times[-1,1]$, cubierto con 256^3 puntos. Los colores representan la parte que cálculo cada uno de los procesadores para $t = 1{,}0$.



(a) $t = 0$

(b) $t = 0{,}25$



(c) $t = 0{,}50$

(d) $t = 1{,}0$

Figura 4.14: Solución de la ecuación de onda a lo largo del eje y, donde se puede observar la evolución de la onda para 4 valores en el tiempo. a) $t = 0$, b) $t = 0{,}25$, c) $t = 0{,}50$, d) $t = 1{,}0$.

Pruebas de convergencia

Una vez calculada la solución de la ecuación de onda tridimensional, se procede a realizar las pruebas de convergencia las cuales se realizan de una forma análoga al caso unidimensional, calculando el error dado por la expresión (4.21):

$$Error = \varphi_{ex} - \varphi_{num}.$$

Para saber si la solución aproximada converge a la solución exacta, se hace uso de la expresión (2.22). De este modo se ejecuta el código con 2 resoluciones distintas para mostrar efectivamente la convergencia a segundo orden del error. Para este caso, se consideran las mismas dimensiones espaciales y temporales que se usan para calcular la solución de la ecuación de la onda, se usan 128^3 puntos teniendo entonces una resolución $\Delta x = \Delta y = \Delta z = 2/128$. Para apreciar mejor la convergencia, se presenta solo el error a lo largo del eje x. La Fig. 4.15 muestra dicho error trabajando con 12 procesadores y usando como solución exacta la ecuación (4.31) y la ecuación (4.33), se presentan 4 casos, $t = 0{,}25$, $t = 0{,}50$, $t = 0{,}75$ y $t = 1{,}0$, donde para cada uno es posible verificar la convergencia.

Pruebas de escalamiento

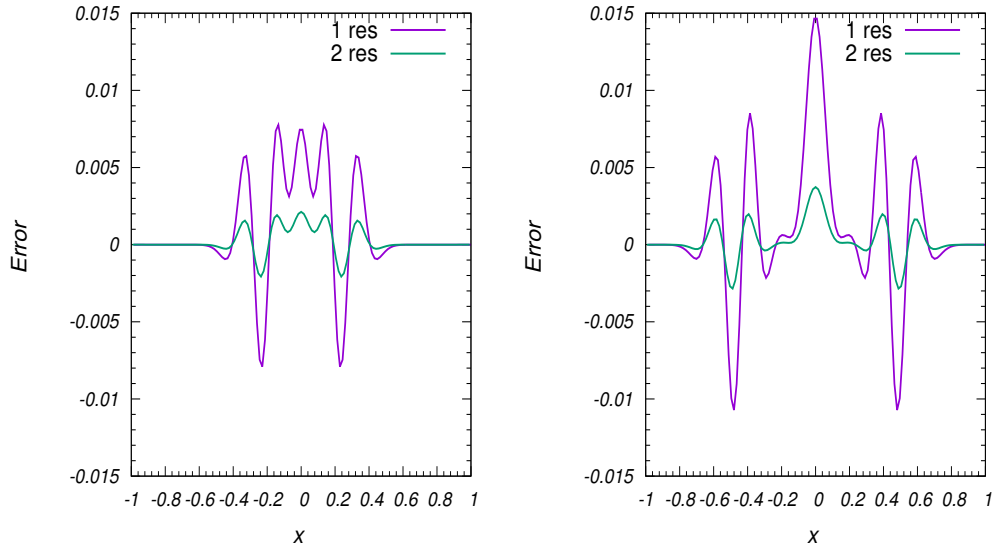
Al igual que en el caso unidimensional, se hicieron 2 pruebas de escalamiento, unas salvando datos y otras sin salvar datos de las soluciones durante la ejecución del código computacional.

Prueba 1: Sin salvar datos durante la ejecución.

- Se hizo para 1, 2, 4, 8 y 12 procesadores.
- Se hicieron 3 corridas para cada número de procesadores, tomando solo el tiempo de trabajo.
- Se calculó un promedio del tiempo de las 3 corridas.
- Se salvaron los datos solo para $t = 0$ y para $t = 1$.

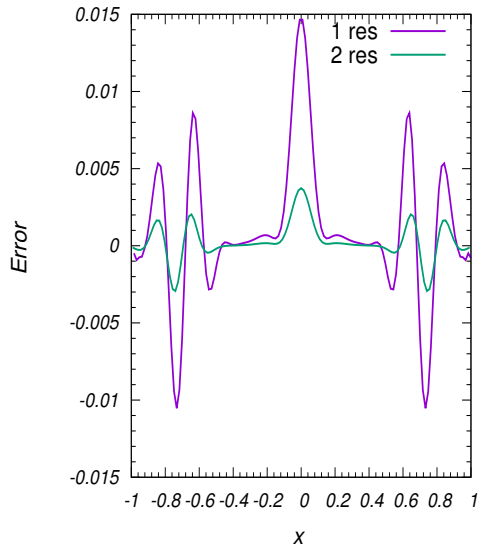
Prueba 2: Salvando datos durante la ejecución:

- Se hizo para 1, 2, 4, 8 y 12 procesadores.
- Se hicieron 3 corridas para cada número de procesadores, tomando el tiempo de trabajo.
- Se calculó un promedio del tiempo de las 3 corridas.

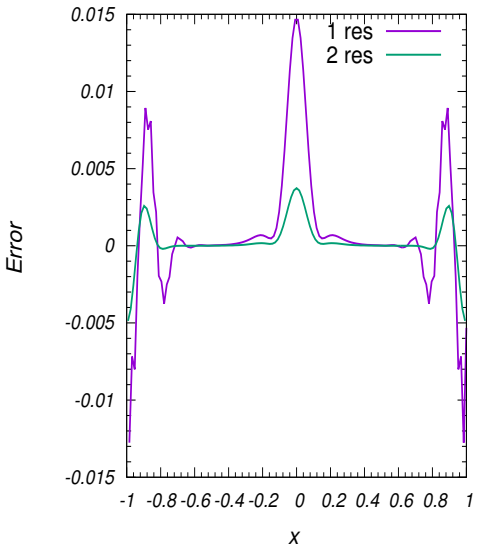


(a) $t = 0{,}25$

(b) $t = 0{,}50$



(c) $t = 0{,}75$



(d) $t = 1{,}0$

Figura 4.15: Pruebas de convergencia trabajando con 12 procesadores y para 4 valores distintos del tiempo. Se calcula el error con 2 resoluciones diferentes, la curva (morada) muestra el error usando una resolución $\Delta x = \Delta y = \Delta z = 2/128$, mientras que la curva (verde) corresponde a $\Delta x/2 = \Delta y/2 = \Delta z/2 = 2/256$. Se verifica que las curvas se superponen cuando se multiplica la curva (verde) por un factor de 4 por tal motivo se tiene una convergencia de segundo orden trabajando con 12 procesador.

- Los datos se salvaron de dos formas:
  - A cada iteración, para observar las gráficas a lo largo del eje x, eje y, o el eje z.
  - Cada 4 iteraciones, para observar las gráficas a lo largo del plano yz.

Se realizaron estas 2 pruebas para 3 distintos números de puntos, y se trabajan en un dominio espacial $x \in [-1,1]$, $y \in [-1,1]$, $z \in [-1,1]$. El primer caso se presenta en la Fig. 4.16, donde se muestra la prueba 1 y preba 2 cubiertas con 64^3 puntos, realizando 64 iteraciones, donde es posible apreciar, que no siempre la mejor opción es trabajar con un gran número de procesadores, ya que al calcular la solución de una EDP con pocos puntos esta tarda más tiempo comunicando datos entre muchos procesadores, que comunicando datos con 4 o hasta 2 procesadores.

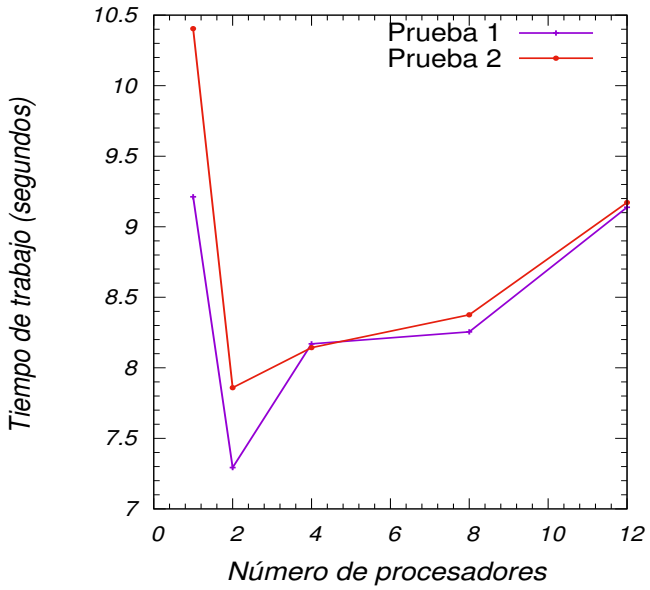
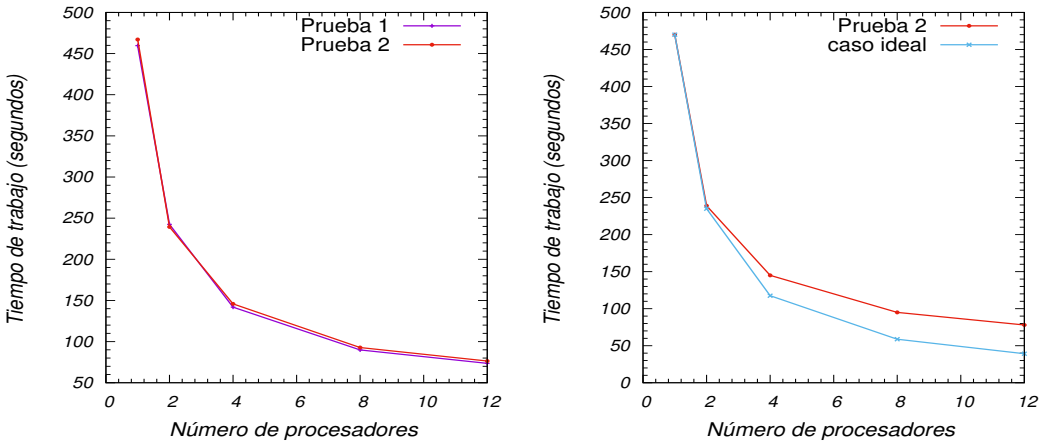


Figura 4.16: Pruebas de escalamiento en un dominio $[-1,1] \times [-1,1] \times [-1,1]$, cubierto con 64^3 puntos, con $c = \Delta t/min(\Delta x, \Delta y, \Delta z) = 0{,}5$ y para $t = 1$, usando ($N_t = t/\Delta t = 64$) iteraciones. La curva color (moreado) muestra el tiempo que tarda la ejecución de la prueba 1 en función del número de procesadores. La curva (morada) muestra el tiempo que tarda la ejecución de la prueba 2 en función del número de procesadores.

Por otro lado, la Fig. 4.17 (a) muestra la prueba 1 y prueba 2 cubiertas con 128^3 puntos, y realizando 128 iteraciones, o lo que es equivalente, multiplicando por un factor de 2^3 la memoria RAM. Se observa del comportamiento de la pendiente de la curva, que cada vez que se trabaja con el doble numéro de procesadores, el tiempo de trabajo se reduce casi a la mitad. El cuadro 4.3, muestra el tiempo real del cálculo de la solución (prueba 2) y el tiempo del caso ideal, los cuales se ilustra en la Fig. 4.17 (b).



(a) Prueba 1 - Prueba 2 (b) Prueba 2 - Caso Ideal

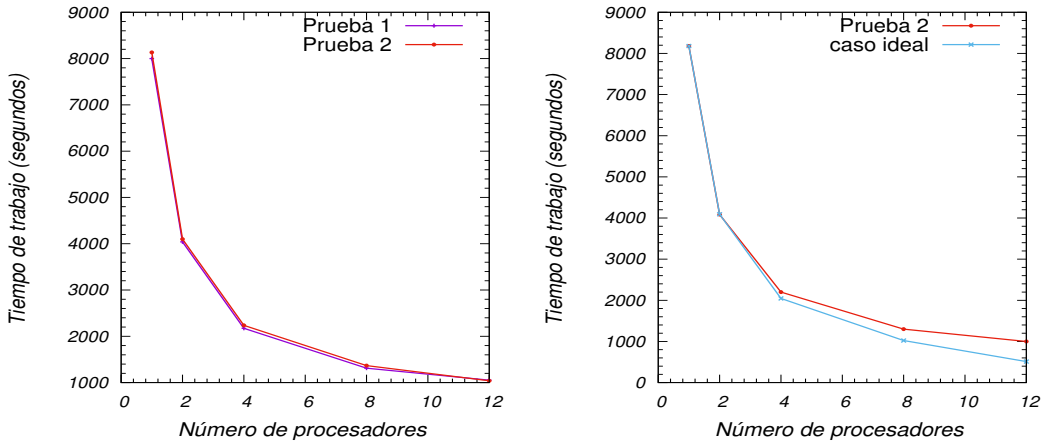
Figura 4.17: Pruebas de escalamiento en un dominio $[-1,1]\times[-1,1]\times[-1,1]$, cubierto con 128^3 puntos, con $c = \Delta t/min(\Delta x,\Delta y,\Delta z) = 0{,}5$, y con $t = 1$ usando ($N_t = t/\Delta t = 128$) iteraciones. En (a) la curva (morada) muestra el tiempo que tarda la ejecución de la prueba 1 en función del número de procesadores. La curva (roja) muestra el tiempo que tarda la ejecución de la prueba 2 en función del número de procesadores. (b) Muestra la curva (roja) de la prueba 2 y la curva (azul) asociada al caso ideal del cuadro 4.3.

Análogamente, la Fig. 4.18 (a) muestra la prueba 1 y prueba 2 multiplicando nuevamente la memoria RAM por 2^3, con 256^3 puntos, y para 256 iteraciones. Esta última parte es escencial para mostrar porqué es conveniente trabajar de forma paralelizada, puesto que conforme se aumenta por un factor de 2 el número de puntos de cada eje, la memoria RAM aumentara con un factor de 2^3, haciendo que un problema relativamente sencillo, se transforme en un problema prohibitivo. Además, ilustra que el escalamiento no es lineas, ya que al principio, al trabajar con 1 procesador y con 2 procesadores, se observa que el tiempo de trabajo se reduce a la mitad al momento que se duplica el número de procesadores con los que se este trabajando, pero al trabajar con 4 y con 8 procesadores, se pierde esta realación. Para apreciarlo mejor, se presenta el cuadro 4.4, el cual ilustra el caso ideal y el caso real (prueba 2), además, es posible estimar una diferencia entre esto dos. La Fig. 4.18 (b) ilustra los datos del cuadro 4.4, de donde se observa que la relación cambia. Esto depende del número de procesadores y de la comunicación que se esta llevando a cabo.

Donde hay que mecionar que el escenario perfecto para paralelizar es aquel que ocupa más tiempo calculando que comunicando.

Cuadro 4.3: Prueba de escalamiento cubierta con 128^3 puntos de la ecuación de onda en tres dimensiones. Se comparan los tiempos de escalamiento con el caso ideal y se estima una diferencia temporal.

Número de procesadores	Tiempo real	Tiempo ideal	Diferencia
1 procesador	$t = 470s$	$t = 470s$	$t = 0s$
2 procesadores	$t = 239s$	$t = 235s$	$t = 4s$
4 procesadores	$t = 145s$	$t = 117{,}5s$	$t = 27{,}5s$
8 procesadores	$t = 95s$	$t = 58{,}8$	$t = 9{,}2s$
12 procesadores	$t = 78s$	$t = 39{,}2s$	$t = 38{,}8s$



(a) Prueba 1 - Prueba 2 (b) Prueba 2 - Caso Ideal

Figura 4.18: Pruebas de escalamiento para un dominio de $[-1,1] \times [-1,1] \times [-1,1]$, con $\Delta x = 1/256$, $\Delta y = 1/256$, $\Delta z = 1/256$, con 256^3 puntos, para 256 iteraciones ($t = 1$) y con $c = 0{,}5$. En (a) la curva (morada) muestra el tiempo que tarda la ejecución de la prueba 1 en función del número de procesadores. La curva (roja) muestra el tiempo que tarda la ejecución de la prueba 2 en función del número de procesadores. En (b) se muestra la curva (roja) de la prueba 2 y la curva (azul) asociada al caso ideal del cuadro 4.4.

Cuadro 4.4: Prueba de escalamiento cubierta con 256^3 puntos de la ecuación de onda en tres dimensiones. Se comparan los tiempos de escalamiento con el caso ideal y se estima una diferencia temporal.

Número de procesadores	Tiempo real	Tiempo ideal	Diferencia
1 procesador	$t = 8180s$	$t = 8180s$	$t = 0s$
2 procesadores	$t = 4085s$	$t = 4090s$	$t = 5s$
4 procesadores	$t = 2200s$	$t = 2045s$	$t = 155s$
8 procesadores	$t = 1300s$	$t = 1022$	$t = 278s$
12 procesadores	$t = 1000s$	$t = 511s$	$t = 489s$

Capítulo 5

Conclusiones

Se construyó un programa de cómputo en paralelo, capaz de resolver PVIs, usando el lenguaje de programación Fortran90 y la biblioteca de MPI. Para la implementación del código, se usaron diferencias finitas, el método de líneas y un integrador Runge-Kutta de tercer orden.

Para probar el código se resolvió la ecuación de onda unidimensional y la ecuación de onda tridimensional, realizaron pruebas de convergencia las cuales indican que la solución numérica converge a la solución exacta en el continuo.

Se trabajó con 1, 2, 4, 8 y 12 procesadores, realizando pruebas de escalamiento, las cuales consisten en medir el tiempo de trabajo en función del número de procesadores. Al realizar estas pruebas se observó que el escalamiento no es lineal, ya que al trabajar con un mayor número de procesadores existe una mayor comunicación, lo que implica un mayor tiempo de trabajo.

Las pruebas de escalamiento se realizaron en ambos problemas, mediante estas se observó que en ocasiones, cuando se trabaja con un pequeño número de puntos, no es necesario trabajar con muchos procesadores. Pero al trabajar con problemas que requieran un gran número de puntos siempre la mejor opción es trabajar con más procesadores. Notando que al trabajar en problemas prohibitivos esto es obligatorio.

Otro punto importante que conviene mencionar, es la partición del dominio entre procesadores. Para el caso unidimensional, la partición espacial se hace dividiendo en segmentos la recta. Para el caso tridimensional se realizaron cortes a lo largo de un eje espacial, (el eje z). Sin embargo, también existe la posibilidad de dividir el dominio tridimensional en octantes, asociando a cada octante un procesador, donde cada procesador estará comunicando información con varios vecinos a la vez, calculando así la solución más rápido.

La estructura del código computacional empleado, sirve para una extensa gama de PVIs, para ello, basta con plantear bien el problema modificando las ecuaciones de evolución. Por ejemplo, la ecuación de Schrödinger en tres dimensiones, puede reescribirse como un sistema de dos ecuaciones diferenciales parciales, las cuales determinarán las nuevas ecuaciones de evolución para este problema, teniendo en cuentas las condiciones iniciales y de frontera. De este modo, es posible resolver un extenso número de PVIs.

Apéndice A

Anexo I: Método Runge-Kutta de tercer orden

Sea $\frac{dy}{dt} = f(t, y)$ una EDO, definida para un dominio en la variable independiente $t = t_0, t_1, t_2, \cdots$, y la variable dependiente $y = y_0, y_1, y_2, \cdots$, con $y(t_0) = y_0$. Es posible construir una solución numérica de esta EDO mediante el integrador Runge-Kutta (RK). Ya que se trabaja en un dominio discreto, la EDO definida en un punto k de la malla es:

$$y_{k+1} = f(t_k, y_k), \tag{A.1}$$

con $k = 0, 1, \cdots$.

El método de integración RK consiste en proponer que y_{k+1} de la forma

$$y_{k+1} = y_k + (a_1\kappa_1 + a_2\kappa_2 + \cdots + a_n\kappa_n)\,\Delta t \tag{A.2}$$

donde a_j son constantes y las κ_j los valores de $(\frac{dy}{dt})$ evaluadas en distintos puntos del lapso entre t_k y t_{k+1}:

$$\kappa_1 = f(t_k, y_k),$$

$$\kappa_2 = f(t_k + p_1\Delta t, y_k + q_{11}\kappa_1\Delta t),$$

$$\kappa_3 = f(t_k + p_2\Delta t, y_k + (q_{21}\kappa_1 + q_{22}\kappa_2)\Delta t),$$

$$\vdots$$

$$\kappa_{i+1} = f(t_k + p_i\Delta t, y_k + \Delta t \sum_{j=1}^{N} q_{ij}\kappa_j), \quad i = 1, \ldots, m,$$

$$\vdots$$

.

.

$$k_n = f(t_k + p_{n-1}\Delta t, y_k + (q_{(n-1)1}k_1 + q_{(n-1)2}k_2 + \cdots)\Delta t),$$

donde p_i, $q_{i,j}$ son constantes, las cuales sirven para determinar el tipo y precisión del método y donde N determina el orden del RK [3]. Se trabajará con $N = 3$, es decir un Runge-Kutta de tercer orden (RK3), para ello, Se supone una expansión solamente en κ_1, κ_2 y κ_3, o sea

$$y_{k+1} = y_k + (a_1\kappa_1 + a_2\kappa_2 + a_3\kappa_3)\,\Delta t, \tag{A.3}$$

con

$$\begin{aligned} \kappa_1 &= f(t_k, y_k), \\ \kappa_2 &= f(t_k + p_1\Delta t, y_k + q_{11}\kappa_1\Delta t), \\ \kappa_3 &= f(t_k + p_2\Delta t, y_k + (q_{21}\kappa_1 + q_{22}\kappa_2)\Delta t). \end{aligned} \tag{A.4}$$

Las constantes a_1, a_2, a_3, p_1, p_2, q_{11}, q_{21} y q_{22} se determinan comparando la forma de y_{k+1} con su expansión en serie de Taylor, esto es:

$$y_{k+1} = y_k + \left[\frac{dy}{dt}\right]_{t_k}\Delta t + \left[\frac{d^2y}{dt^2}\right]_{t_k}\frac{\Delta t^2}{2} + \left[\frac{d^3y}{dt^3}\right]_{t_k}\frac{\Delta t^3}{6} + O(\Delta t^4),$$

donde las derivadas tienen la forma:

$$\frac{dy}{dt} = f(t, y),$$

$$\frac{d^2y}{dt^2} = \frac{df(t,y)}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial t} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}f = f_t + f_y f,$$

$$\begin{aligned} \frac{d^3y}{dt^3} &= \frac{d}{dt}\left(\frac{d^2y}{dt^2}\right) = \frac{d}{dt}\left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}f\right) \\ &= f_{tt} + 2f_{yt}f + f_{yy}f^2 + f_t f_y + f_y^2 f. \end{aligned}$$

Sustituyendo los términos de las derivadas en la expansión de arriba, se obtiene:

$$\begin{aligned} y_{k+1} &= y_k + f\Delta t + (f_t + f_y f)\tfrac{\Delta t^2}{2} + (f_{tt} + 2f_{ty}f + f_{yy}f^2 + \\ &\quad f_t f_y + f_y^2 f)\tfrac{\Delta t^3}{6} + O(\Delta t^4). \end{aligned} \tag{A.5}$$

Por otra parte se evaluan κ_1, κ_2 y κ_3 para determinar la ecuación (A.3). Para ello, se requiere la expansión en serie de Taylor de una función de dos variables, la cual se formula de la siguiente manera. Sea f una función real de dos variables $f(t, y)$ infinitamente diferenciable en torno de un punto del espacio (a, b), la serie de Taylor para esta función

es:

$$f(t,y) = f(a,b) + f_t(a,b)(t-a) + f_y(a,b)(y-b) + \frac{1}{2}[f_{tt}(a,b)(t-a)^2 + 2f_{ty}(a,b)(t-a)(y-b) + f_{yy}(a,b)(y-b)^2] + O(\Delta t^3). \quad \text{(A.6)}$$

De modo que las $\kappa's$ se expresan de la siguiente forma:

$$\begin{aligned}
\kappa_1 &= f(t_k, y_k), \\
\kappa_2 &= f(t_k + p_1\Delta t, y_k + q_{11}\kappa_1\Delta t), \\
&= f + (f_t p_1 + f_y q_{11}\kappa_1)\Delta t + \tfrac{1}{2}[f_t p_1^2 + 2f_{ty}p_1 q_{11}\kappa_1 + f_{yy}(q_{11}k_1)^2]\Delta t^2 + O(\Delta t^3),
\end{aligned}$$

sustituyendo κ_1 se tiene

$$\begin{aligned}
\kappa_2 &= f + (f_t p_1 + f_y q_{11} f)\Delta t + \tfrac{1}{2}[f_{tt}p_1^2 + 2f_{ty}p_1 q_{11} f + f_{yy}q_{11}^2 f^2]\Delta t^2, \\
\kappa_3 &= f(t_k + p_2\Delta t, y_k + (q_{21}\kappa_1 + q_{22}\kappa_2)\Delta t) \\
&= f + (f_t p_2 + f_y(q_{21}\kappa_1 + q_{22}\kappa_2))\Delta t + \tfrac{1}{2}[f_{tt}p_2^2 + 2f_{ty}(p_2 q_{21}\kappa_1 + p_2 q_{22}k_2) + \\
&\quad f_{yy}(q_{21}^2\kappa_1^2 + 2q_{21}q_{22}\kappa_1\kappa_2 + q_{22}^2\kappa_2^2)]\Delta t^2 + O(\Delta t^3),
\end{aligned}$$

sustituyendo κ_1 y κ_2 se obtiene

$$\begin{aligned}
\kappa_3 &= f + (f_t p_2 + f_y q_{21} f + f_y q_{22}\{f + (f_t p_1 + f_y q_{11} f)\Delta t + \tfrac{1}{2}[f_{tt}p_1^2 + 2f_{ty}p_1 q_{11} f + \\
&\quad f_{yy}q_{11}^2 f^2]\Delta t^2\})\Delta t + \tfrac{1}{2}f_{tt}p_2^2\Delta t^2 + f_{ty}p_2 q_{21} f\Delta t^2 + f_{ty}p_2 q_{22}\{f + (f_t p_1 + \\
&\quad f_y q_{11} f)\Delta t + \tfrac{1}{2}[f_{tt}p_1^2 + 2f_{ty}p_1 q_{11} f + f_{yy}q_{11}^2 f^2]\Delta t^2\}\Delta t^2 + \tfrac{1}{2}f_{yy}q_{21}^2 f^2\Delta t^2 + \\
&\quad f_{yy}q_{21}q_{22} f\{f + (f_t p_1 + f_y q_{11} f)\Delta t + \tfrac{1}{2}[f_{tt}p_1^2 + 2f_{ty}p_1 q_{11} f]\Delta t^2\}\Delta t^2 + \\
&\quad \tfrac{1}{2}f_{yy}q_{22}^2\{f^2 + 2f(f_t p_1 + f_y q_{11} f)\Delta t + [f(f_{tt}p_1^2 + 2f_{ty}p_1 q_{11} + f_{yy}q_{11} f) + \\
&\quad (f_t p_1 + f_y q_{11} f)^2]\Delta t^2 + O(\Delta t^3)\}\Delta t^2 + O(\Delta t^3) \\
&= f + (f_t p_2 + f_y q_{21} f + f_y q_{22} f)\Delta t + [f_y f_t p_1 q_{22} + f_y^2 q_{11} q_{22} f + \tfrac{1}{2}f_{tt}p_2^2 + \\
&\quad f_{ty}f p_2 q_{21} + f_{ty}f p_2 q_{22} + \tfrac{1}{2}f_{yy}f^2 q_{21}^2 + f_{yy}f^2 q_{21}q_{22} + \tfrac{1}{2}f_{yy}2^2 q_{22}^2]\Delta t^2 + O(\Delta t^3),
\end{aligned}$$

simplificando se tiene:

$$\begin{aligned}
\kappa_3 &= f + (f_t p_2 + f_y q_{21} f + f_y q_{22} f)\Delta t + [f_y f_t p_1 q_{22} + f_y^2 f q_{11} q_{22} + f_{tt}\tfrac{1}{2}p_2^2 + \\
&\quad f_{ty}f(p_2 q_{21} + p_2 q_{22}) + f_{yy}f^2(\tfrac{1}{2}q_{21}^2 + q_{21}q_{22} + \tfrac{1}{2}q_{22}^2)]\Delta t^2 + O(\Delta t^3)
\end{aligned}$$

sustituyendo en (A.3) se tiene:

$$\begin{aligned} y_{k+1} &= y_k + (a_1 f + a_2\{f + (f_t p_1 + f_y f q_{11})\Delta t + \tfrac{1}{2}[f_{tt} p_1^2 + f_{ty} f 2 p_1 q_{11} + f_{yy} f^2 q_{11}^2] \\ &\quad \Delta t^2\} + a_3\{f + (f_t p_2 + f_y f q_{21} + f_y f q_{22})\Delta t + [f_y f_t p_1 q_{22} + f_y^2 f q_{11} q_{22} + \\ &\quad f_{tt}\tfrac{1}{2} p_2^2 + f_{ty} f(p_2 q_{21} + p_2 q_{22}) + f_{yy} f^2(\tfrac{1}{2} q_{21}^2 + q_{21} q_{22} + \tfrac{1}{2} q_{22}^2)]\Delta t^2 + \\ &\quad O(\Delta t^3)\})\Delta t. \end{aligned}$$

agrupando términos

$$\begin{aligned} y_{k+1} &= y_k + f(a_1 + a_2 + a_3)\Delta t + f_t(a_2 p_1 + a_3 p_2)\Delta t^2 + f_y f(a_2 q_{11} + a_3 q_{21} + \\ &\quad a_3 q_{22})\Delta t + \{f_{tt}(\tfrac{1}{2} a_2 p_1^2 + \tfrac{1}{2} a_3 p_2^2) + f_{ty} f(a_2 p_1 q_{11} + a_3 p_2 q_{21} + a_3 p_2 q_{22}) + \\ &\quad f_t f_y a_3 p_1 q_{22} + f_y^2 f a_3 q_{11} q_{22} + f_{yy} f^2(\tfrac{1}{2} a_2 q_{11}^2 + \tfrac{1}{2} a_3 q_{21}^2 + \tfrac{1}{2} a_3 q_{22}^2 + \\ &\quad a_3 q_{21} q_{22})\}\Delta t^3 + O(\Delta t^4), \end{aligned}$$

comparando con la ecuación (A.5) se tiene el siguiente sistema de ecuaciones:

$$\begin{aligned} a_1 + a_2 + a_3 &= 1 \\ a_2 p_1 + a_3 p_2 &= \tfrac{1}{2} \\ a_2 q_{11} + a_3 q_{21} + a_3 q_{22} &= \tfrac{1}{2} \\ \tfrac{1}{2} a_2 p_1^2 + \tfrac{1}{2} a_3 p_2^2 &= \tfrac{1}{6} \\ a_2 p_1 q_{11} + a_3 p_2 q_{21} + a_3 p_2 q_{22} &= \tfrac{1}{3} \\ a_3 p_1 q_{22} &= \tfrac{1}{6} \\ a_3 q_{11} q_{22} &= \tfrac{1}{6} \\ \tfrac{1}{2} a_2 q_{11}^2 + \tfrac{1}{2} a_3 q_{21}^2 + \tfrac{1}{2} a_3 q_{22}^2 + a_3 q_{21} q_{22} &= \tfrac{1}{6} \end{aligned}$$

el cual es un sistema de 8 ecuaciones con 8 incognitas y dado que existe una dependencia lineal entre algunas ecuaciones, es posible reducir el sistema a 6 ecuaciones con 8 incogni-

tas, lo cual quiere decir que existe una infinidad de soluciones. Para resolver este sistema basta con fijar dos de las constantes de modo que se obtenga un sistema de 6 ecuaciones con 6 incognitas. La solución más común de este sistema es dada por:

$$\begin{aligned} a_1 &= \tfrac{1}{6} \\ a_2 &= \tfrac{2}{3} \\ a_3 &= \tfrac{1}{6} \\ p_1 &= \tfrac{1}{2} \\ p_2 &= 1 \\ q_{11} &= \tfrac{1}{2} \\ q_{21} &= -1 \\ q_{22} &= 2 \end{aligned}$$

donde sustituyendo en la ecuación (A.3), κ_1, κ_2 y κ_3, se obtiene:

$$y_{k+1} = y_k + \frac{1}{6}\left(\kappa_1 + 4\kappa_2 + \kappa_3\right)\Delta t, \tag{A.7}$$

con

$$\begin{aligned} \kappa_1 &= f(t_k, y_k) \\ \kappa_2 &= f(t_k + \tfrac{1}{2}\Delta t, y_k + \tfrac{1}{2}\kappa_1 \Delta t) \\ \kappa_3 &= f(t_k + \Delta t, y_k - \kappa_1 \Delta t + 2\kappa_2 \Delta t) \end{aligned} \tag{A.8}$$

De este modo se construye el método Runge-Kutta 3 que se usa en este trabajo.

Bibliografía

[1] Diego H. Milone, Adrián A. Azar, Leonardo H. Rufiner, *Supercomputadoras basadas en clusters de PCs, 02 julio 2002.*

[2] https://computing.llnl.gov/tutorials/parallel_comp/

[3] Steven C. Chapra, Raymond P. Canale *Tufts University, University of Michigan, Métodos numéricos para ingenieros, 2007.*

[4] F. S. Guzmán, *Solución de la ecuación de onda como un problema de valores iniciales usando diferencias finitas, Rev. mex. de física 56 (1) 51-68.*

[5] J.W. Thomas, *Numerical Partial Differential Equations: finite difference methods, ("Texts in Applied Mathematics physics", Springer. 1995) Fort Collins, CO 80543 USA.*

[6] www.mpich.org