UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

PARALELIZACIÓN DE UN ALGORITMO DE DIFERENCIAS FINITAS 3D PARA LA GENERACIÓN DE SISMOGRAMAS SINTÉTICOS

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

EDUARDO CÉSAR CABRERA FLORES

DIRECTOR: DR. MARIO CHÁVEZ GONZÁLEZ

OCTUBRE DE 2008

DEDICATORIA.

A mi mamá y a mi papá por su paciencia, apoyo y sabiduría.

AGRADECIMIENTOS.

A mis hermanos por su compañía y aliento.

A mi familia.
A mis amigos todos por su afecto y complicidad.
Al Dr. Mario Chávez por el apoyo otorgado en la realización de este trabajo.
Al Departamento de Supercómputo de la DGSCA-UNAM, por los recursos de supercómputo de *KanBalam*.

"Cuando el hombre habla se divierte, sólo al escribir razona", Rousseau.

II

Índice general

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS.

Capítulo 1

INTRODUCCIÓN.

1.1. ANTECEDENTES.

México está situado en una de las zonas sísmicas con mayor actividad del orbe, la cual se ubica a lo largo de las costas del Océano Pacífico. Los sismos en México se deben a la interacción entre las cinco placas tectónicas: la de Cocos, la Norteamericana, la del Pacífico, la del Caribe y de Rivera mostradas en la Fig. 1.1. El 19/09/1985 ocurrió un sismo de subducción de magnitud en la escala de *Richter* $Ms = 8.1$, el cual originó perdidas humanas del orden de 30,000 –en particular en la Ciudad de México- así como económicas del orden de 6 mil millones de dólares estadounidenses.

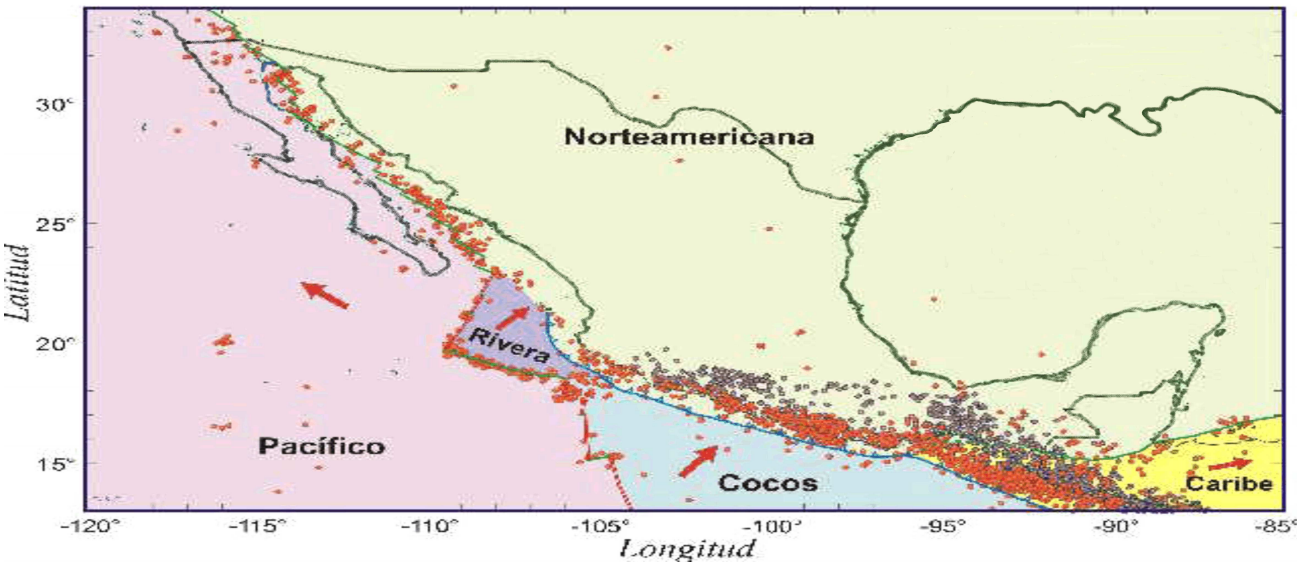


Figura 1.1: Ubicación de las placas tectónicas en México (Servicio Sismológico Nacional, 2002).

En la Fig. 1.2 se presenta el mapa de intensidades de *Mercalli* modificadas (Apéndice A) del mencionado sismo. Un aspecto relevante de dicha figura es que las intensidades máximas del sismo ocurrieron, como es habitual, en la zona epicentral y además en la Ciudad de México donde alcanzó intensidades entre IX y X. Dado que el período de recurrencia de este tipo de

sismos puede ser de sólo algunas decenas de años, así como, a que se ha detectado una laguna sísmica en la zona del estado de Guerrero, por lo cual, existe un interés sismológico, ingenieril y socioeconómico por estudiar los posibles movimientos del terreno al ocurrir sismos mayores al de 1985.

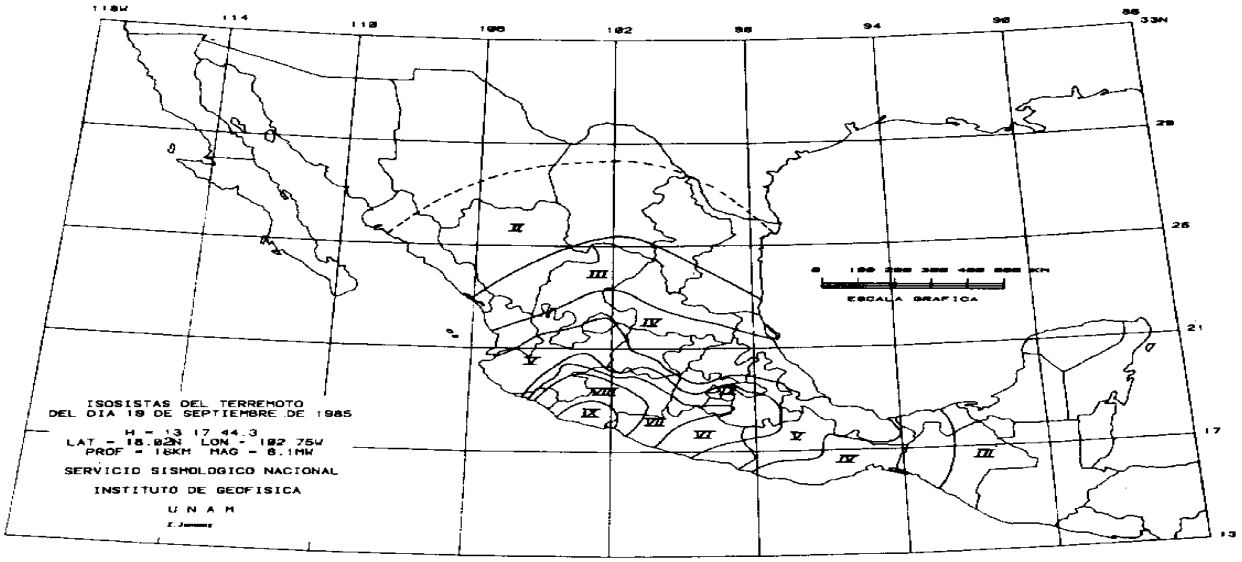


Figura 1.2: Intensidades e Isosistas del sismo del 19 de septiembre de 1985 (Servicio Sismológico Nacional, 1985).

Con el fin de obtener los denominados sismogramas sintéticos, asociados a eventos sísmicos específicos, en los últimos años se han propuesto modelos matemáticos computacionales para tal fin, como el que se desarrolló y aplicó para el sismo de subducción de Colima-Jalisco ocurrido el 09/10/1995 de $Ms = 7.6$ [1]. Dicho modelo incluye el modelado en bajas frecuencias ($f \leq 0.3$ Hz) a partir de un código serial de diferencias finitas alternadas (*staggered*) en 3D **DFA3D** de un sistema de ecuaciones en derivadas parciales que representa la propagación de ondas sísmicas en tres dimensiones.

1.2. OBJETIVO.

El objetivo de esta tesis es la paralelización del código serial de **DFA3D** respectivo y su aplicación para la obtención de sismogramas sintéticos del sismo $Ms = 8.1$ de 1985.

La tesis está organizada de la siguiente forma. En el capítulo 2 se describe la elastodinámica del problema de propagación de ondas sísmicas en un medio 3D y el método de diferencias finitas alternado (*staggered*) empleado. La implementación paralela del código serial realizada con paralelismo de datos, en conjunto con *MPI* (interfaz de envío de mensajes, por sus siglas en inglés), así como el estudio de la eficiencia del programa se presentan en el capítulo 3. La generación de los sismogramas sintéticos del sismo de 1985 y las mediciones del rendimiento del programa en paralelo son presentados en el capítulo 4. Las conclusiones y recomendaciones del trabajo son expuestas en el capítulo 5.

Capítulo 2

ELASTODINÁMICA Y MÉTODO DE DIFERENCIAS FINITAS ALTERNADAS (*STAGGERED*) 3D.

Al propagarse una onda sísmica, por un medio cualquiera, éste sufre modificaciones debidas al paso de la onda. La cantidad y el tipo de estas modificaciones están en función, principalmente, del contenido de energía de la onda y de las características físicas del medio. Los principales cambios experimentados por el medio debidos al paso de una onda sísmica son la modificación de la forma geométrica y la redistribución de las fuerzas internas. La elastodinámica aborda el análisis de estos efectos.

En este capítulo se explicará brevemente la teoría de la elastodinámica, que involucra análisis de esfuerzos, de deformaciones y de propagación de ondas. También se abordarán los conceptos de métodos numéricos, su importancia, validez y pertinencia en la solución de problemas físicos. Así mismo, se describirá el método de diferencias finitas alternadas empleado en este trabajo. A continuación se sintetizan los aspectos más relevantes de la propagación de ondas sísmicas [2].

2.1. ELASTODINÁMICA.

En esta sección se explican los conceptos básicos de la elastodinámica, como son las nociones del vector de desplazamientos, de esfuerzo y de deformaciones en un medio continuo, los cuales nos llevan a la ecuación de *Navier* o ecuación de movimiento, que expresa las condiciones de equilibrio en función de los componentes del vector de desplazamiento.

La mecánica de medios continuos estudia el comportamiento de la materia sin considerar su granularidad; es decir, su estructura atómica. Esta aproximación es válida mientras las condiciones experimentales no alcancen los límites, en los cuales aparecen los efectos cuánticos.

2.1.1. TEORÍA DE DEFORMACIONES.

El concepto de deformación es fundamental en la elastodinámica. Describe el cambio de las posiciones relativas entre dos puntos cualesquiera -entre un estado inicial y un estado final- que

componen un sólido en un medio continuo.

Las deformaciones pueden ser *elásticas* (las partículas regresan a su posiciones originales al retirarse el agente deformante); *plásticas* (las partículas adquieren nuevas posiciones) o *fractura* (el sólido pierde su unidad estructural).

Cuando un medio se deforma, cada partícula adquiere una posición nueva, que es descrita por unas coordenadas actuales, distintas a las originales. Dado un sólido y una partícula $\boldsymbol{P}$ en su interior. Sea $\vec{r}_a$ su ubicación antes de la deformación y $\vec{r'_a}$ la ubicación posterior a la deformación, Fig. 2.1. Entonces, el vector de desplazamiento $\vec{u}_a$ de la partícula se define como

$$\vec{u}_a = \vec{r'_a} - \vec{r} \tag{2.1}$$

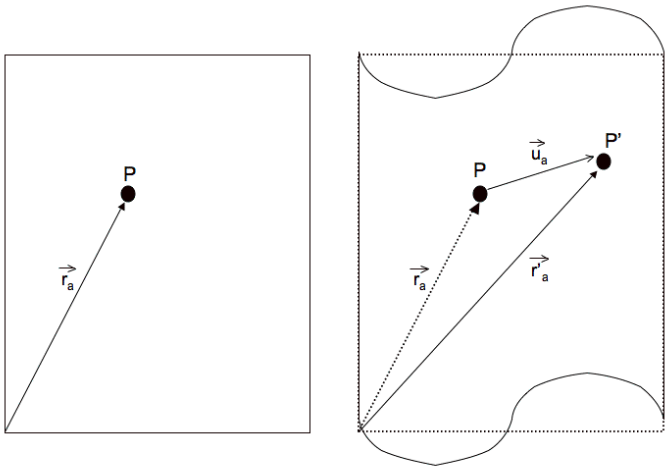


Figura 2.1: Vector de desplazamientos infinitesimales $\vec{u}_a$ de la partícula $\boldsymbol{P}$.

Si a cada partícula del sólido se le asignan coordenadas en un sistema fijo, el conjunto de los vectores $\vec{u}_a$ constituyen un campo vectorial y cuando la deformación es dinámica, además, está en función del tiempo

$$\overrightarrow{u} = \overrightarrow{u}\,(x, y, z, t) \tag{2.2}$$

A la ecuación (2.2) se le conoce como el campo de desplazamientos infinitesimales.

Una deformación homogénea se presenta cuando el sólido es deformado a lo largo de una sola dirección en el sistema tridimensional de referencia empleado. Dado que tenemos un sólido elástico lineal y una deformación en la dirección X

$$\frac{u_x}{x} = \frac{\Delta L}{L} \tag{2.3}$$

$$u_x = e_{xx} x \tag{2.4}$$

Cuando la deformación es no homogénea participan las otras direcciones del sistema de referencia; por lo tanto; $e_{xx} = e_{xx}(x, y, z)$; sin embargo, e_{xx} continua describiendo de manera local la deformación en la dirección X. Generalizando, tenemos

$$e_{xx} = \frac{\partial u_x}{\partial x} \ , \quad e_{yy} = \frac{\partial u_y}{\partial y}, \quad e_{zz} = \frac{\partial u_z}{\partial z} \tag{2.5}$$

Las ecuaciones (2.5) describen deformaciones del tipo de compresión, las cuales van a lo largo de una o más direcciones en el sistema de referencia. Resta por describir las deformaciones tipo cizalla, que describen cambios de posición diagonales y en los cuales existen ángulos involucrados. En estos casos, la deformación en X es proporcional a la posición en Y y viceversa

$$u_x = \frac{\varphi}{2} y \ , \quad u_y = \frac{\varphi}{2} x \tag{2.6}$$

; donde $\varphi/2$ representa el ángulo. Podemos representar la ecuación (2.6) de forma general como

$$e_{xy} = \frac{\partial u_x}{\partial y} \quad , \quad e_{yx} = \frac{\partial u_y}{\partial x} \tag{2.7}$$

De forma similar, podemos definir las deformaciones con respecto a las otras direcciones del sistema de referencia como una matriz de nueve componentes

$$\begin{bmatrix} e_{xx} & e_{xy} & e_{xz} \\ e_{yx} & e_{yy} & e_{yz} \\ e_{zx} & e_{zy} & e_{zz} \end{bmatrix} \tag{2.8}$$

La matriz (2.8) es simétrica; por lo cual, los terminos $e_{xy} = e_{yx}$, $e_{zx} = e_{xz}$, $e_{zy} = e_{yz}$. Podemos escribir la matriz (2.8) de forma general como el tensor de deformaciones

$$e_{ij} = \frac{1}{2} \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right) \tag{2.9}$$

; donde $i = j = x, y \ \vee z$

2.1.2. TEORÍA DE ESFUERZOS.

Sea un cuerpo $\boldsymbol{V}$ ocupando un volumen en una cierta ubicación en el espacio en un momento dado y una superficie cerrada $\boldsymbol{S}$ dentro de $\boldsymbol{V}$. Nos interesa conocer la interacción entre el material al interior y exterior de la superficie cerrada. Designamos sobre $\boldsymbol{V}$ un pequeño elemento diferencial de área ΔS y su vector normal η positivo hacia fuera del volumen de control $\boldsymbol{V}$. Ello nos permite diferenciar las partes del material en ambos lados de ΔS con respecto al sentido de la normal. El material exterior (positivo) de $\boldsymbol{V}$ ejerce sobre el material interior (negativo) una fuerza ΔF que es función del área y orientación de la superficie considerada Fig. 2.2.

$$\Delta F = \Delta F \left(\Delta S, \eta \right) \tag{2.10}$$

Si ΔS tiende a cero, el límite existe y es finito y el momento de las fuerzas que actúan sobre la superficie ΔS en cualquier punto dentro del área desaparece en dicho límite. Definimos el vector de esfuerzos como

$$\lim_{\Delta S \to 0} \frac{\Delta F}{\Delta S} = \frac{dF}{dS} = T_\eta \tag{2.11}$$

Este vector representa la fuerza por unidad de área que actúa sobre la superficie y es, además, función del vector normal η; es decir, de la orientación de la superficie.

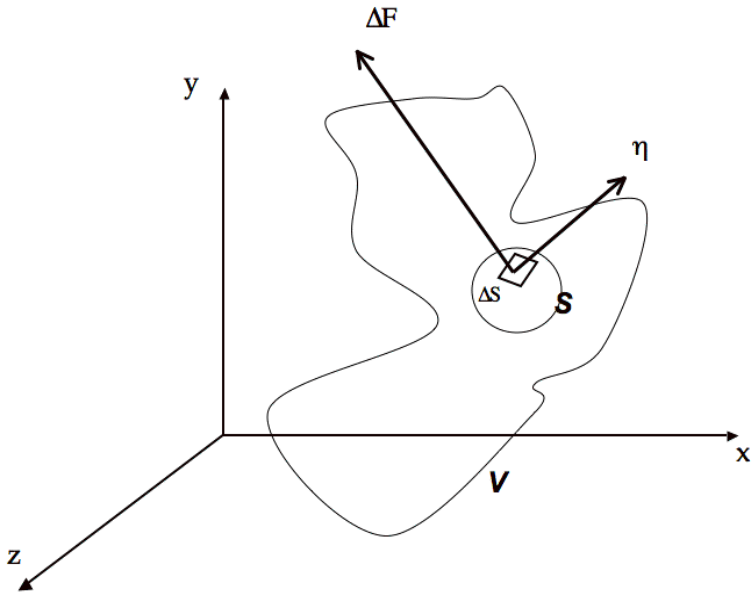


Figura 2.2: Dada una fuerza ΔF aplicada sobre un área ΔS tenemos el tensor de esfuerzos dado por el límite del cociente $\Delta F/\Delta S$ y su dirección por el vector normal η.

Los tipos de fuerza que pueden actuar sobre cualquier sólido son dos: *fuerzas de cuerpo* y *fuerzas de superficie o de contacto*. Las primeras son de gran alcance y actúan sobre todas las partículas del sólido y son proporcionales a su masa o volumen (v.gr. la fuerza de gravedad), en tanto que las segundas actúan sobre los bordes o las fronteras del sólido y su efecto no se produce a un mismo tiempo sobre todas las partículas, sino que es transmitido progresivamente a todas las partículas mediante las fuerzas de las partículas vecinas (esfuerzos). En un sólido toda fuerza interna tiene componentes de compresión y de cizalla; por lo tanto, hay que tomar en cuenta sus componentes a lo largo de todos los ejes. Dado una superficie $\Delta y \Delta z$ perpendicular al eje x. Una fuerza ΔF aplicada a dicha área se descompone en sus tres componentes $\Delta F_x, \Delta F_y, \Delta F_z$ Fig. 2.3. Se definen, por lo tanto, las siguientes expresiones:

$$\sigma_{xx} = \frac{\Delta F_x}{\Delta y \Delta z} \quad , \quad \sigma_{yx} = \frac{\Delta F_y}{\Delta y \Delta z} \quad , \quad \sigma_{zx} = \frac{\Delta F_z}{\Delta y \Delta z} \tag{2.12}$$

De igual forma, se puede tener una superficie de área $\Delta x \Delta y$ perpendicular al eje z o una superficie de área $\Delta x \Delta y$ perpendicular al eje y. Para ambos casos es posible descomponer las fuerzas –como se hizo previamente, ecuación (2.12), y definir los esfuerzos restantes:

$$\begin{aligned} \sigma_{xz} &= \frac{\Delta F_x}{\Delta x \Delta y}, \quad \sigma_{yz} = \frac{\Delta F_y}{\Delta x \Delta y}, \quad \sigma_{zz} = \frac{\Delta F_z}{\Delta x \Delta y} \\ \sigma_{xy} &= \frac{\Delta F_x}{\Delta x \Delta z}, \quad \sigma_{yy} = \frac{\Delta F_y}{\Delta x \Delta z}, \quad \sigma_{zy} = \frac{\Delta F_z}{\Delta x \Delta z} \end{aligned} \tag{2.13}$$

;donde en cada expresión σ_{ij} el primer índice i indica la dirección normal a la superficie sobre la cual actúa, mientras que el segundo índice j denota las componentes de la fuerza. Los com-

ponentes $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}$ se denominan *esfuerzos normales*. El resto de las componentes se conocen como *esfuerzos cortantes*. Podemos escribir en una notación matricial el tensor de esfuerzos, ecuación (2.14).

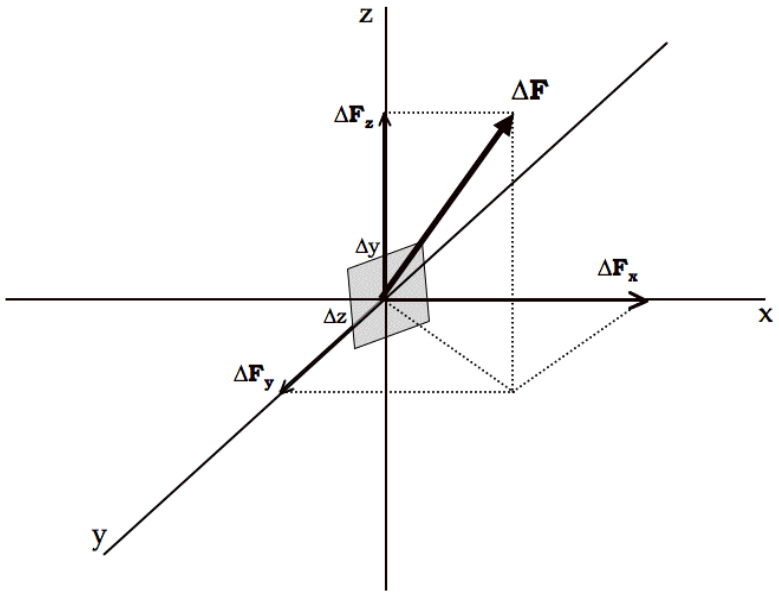


Figura 2.3: Una fuerza ΔF aplicada sobre una superficie $\Delta y \Delta z$ perpendicular al eje x, con sus tres componentes vectoriales $\Delta F_x, \Delta F_y, \Delta F_z$.

$$\sigma_{ij} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \tag{2.14}$$

La cual es una matriz simétrica; por lo tanto, $\sigma_{xy} = \sigma_{yx}$, $\sigma_{xz} = \sigma_{zx}$, $\sigma_{zy} = \sigma_{yz}$.

2.1.3. ECUACIONES CONSTITUTIVAS.

La ecuación constitutiva de un material es una ecuación que describe el comportamiento de dicho material en función de sus propiedades. La relación *deformación-esfuerzo* describe las propiedades mecánicas de un material y es, por consiguiente, una ecuación constitutiva. De hecho, existen tres relaciones deformación-esfuerzo que proporcionan una buena descripción sobre las propiedades mecánicas de la mayoría de los materiales*: fluido no viscoso*; *fluido viscoso Newtoniano* y *el sólido perfectamente elástico.*

Nos enfocaremos exclusivamente al sólido perfectamente elástico o de *Hooke*, ya que en sismología la idealización de un medio lineal, homogéneo y elástico es aceptada como una buena aproximación de los materiales reales.

2.1.3.1. SÓLIDO ELÁSTICO DE *HOOKE*.

Un sólido perfectamente elástico o de *Hooke* es un material que está regido por la ley de *Hooke*. Esta ley define que en cada punto las deformaciones locales son linealmente proporcionales a los esfuerzos locales. Podemos escribir esta ley en su forma general como

$$\sigma_{ij} = C_{ijkl} e_{kl} \tag{2.15}$$

;donde σ_{ij} representa el tensor de esfuerzos, e_{kj} es el tensor de deformaciones y C_{ijkl} es el tensor de elasticidad, que es independiente tanto del esfuerzo como de la deformación. Los coeficientes de C_{ijkl} se denominan constantes elásticas y constituyen un tensor de cuarto orden, que describe las propiedades elásticas del medio. De manera física, estas constantes elásticas proporcionan toda la información necesaria para entender cómo se deformará el medio debido a una tracción (compresión o cizalla) aplicada en alguna dirección.

De forma general, un tensor de cuarto orden tiene $3^4 = 81$ componentes independientes; sin embargo, el tensor de elasticidad posee algunas simetrías que reducen sustancialmente dicha cantidad. Mediante la ley de *Hooke* y la simetría del tensor de deformaciones implican que el tensor de elasticidad debe ser simétrico en su primer par de índices $C_{ijkl} = C_{jikl}$, el segundo par de índices del tensor de elasticidad se contrae con los del tensor de deformaciones, el cual es simétrico, lo cual nos garantiza que el tensor de elasticidad debe de ser, también, simétrico en estos índices $C_{ijkl} = C_{ijlk}$. El tensor de elasticidad, empleando un argumento termodinámico general [3], podemos definirlo como simétrico con respecto del primer y segundo par de índices $C_{ijkl} = C_{klij}$. Por lo tanto, la cantidad de componentes independientes del tensor de elasticidad disminuye a 21, correspondientes al número de componentes de una matriz simétrica de $6 \times 6 \quad \left(n = \frac{6^2+6}{2}\right)$.

2.1.3.2. ECUACIONES DE *NAVIER*.

Hasta ahora, hemos definido cuatro elementos fundamentales de la mecánica de medios continuos: el vector de desplazamientos infinitesimales $\vec{u}_a$; el tensor de esfuerzos σ_{ij}; el tensor de deformaciones e_{ij} y el tensor de elasticidad C_{ijkl}. Ahora, definiremos las ecuaciones de movimiento que relacionan esfuerzos, fuerzas de cuerpo y aceleración. En un medio vibrante con una superficie δS y volumen δV el equilibrio de las fuerzas totales y la aceleración está dado por (segunda ley de *Newton*)

$$\int_{\delta S} \sigma \cdot \hat{n} dS \; + \int_{\delta V} \vec{F} dV = \int_{\delta V} \rho \frac{\partial^2 \vec{u}}{\partial t^2} dV \tag{2.16}$$

;donde σ representa las fuerzas de tracción ejercidas sobre la superficie por las partículas vecinas (esfuerzos), $\vec{F}$ son las fuerzas de cuerpo y ρ indica la densidad del medio.

Dado un pequeño volumen, las integrales de volumen pueden ser consideradas constantes y, por ende, se puede escribir la ecuación (2.16) como

$$\frac{\int_{\delta S} \sigma \cdot \hat{n} dS}{\delta V} = \rho \frac{\partial^2 \vec{u}}{\partial t^2} - \vec{F} \tag{2.17}$$

del lado izquierdo de la ecuación (2.17) tenemos por definición, en el límite del lado, que corresponde a la divergencia

$$\lim_{\delta V \to 0} \frac{\int_{\delta S} \sigma \cdot \hat{n} dS}{\delta V} \equiv \nabla \cdot \sigma \tag{2.18}$$

y empleando coordenadas cartesianas definimos a la divergencia de $\nabla \cdot \sigma$ como

$$\nabla \cdot \sigma = \hat{i}\left(\frac{\partial}{\partial x}\sigma_{xx} + \frac{\partial}{\partial y}\sigma_{xy} + \frac{\partial}{\partial z}\sigma_{xz}\right) + \hat{j}\left(\frac{\partial}{\partial x}\sigma_{yx} + \frac{\partial}{\partial y}\sigma_{yy} + \frac{\partial}{\partial z}\sigma_{yz}\right) + \hat{k}\left(\frac{\partial}{\partial x}\sigma_{zx} + \frac{\partial}{\partial y}\sigma_{zy} + \frac{\partial}{\partial z}\sigma_{zz}\right) \tag{2.19}$$

reescribiendo la ecuación (2.19) tenemos

$$\rho \frac{\partial^2 \vec{u}}{\partial t^2} = \nabla \cdot \sigma - \vec{F} \tag{2.20}$$

y ahora empleando una notación más compacta de la ecuación (2.20) por medio de subíndices

$$\rho \partial_t^2 u_j = \partial_i \left(\sigma_{ij}\right) - F_j \tag{2.21}$$

por lo tanto, la ecuación (2.21) queda como sigue

$$\begin{array}{rcl} \rho \frac{\partial^2 u_x}{\partial t^2} & = & \left[\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z}\right] - F_x \\ \rho \frac{\partial^2 u_y}{\partial t^2} & = & \left[\frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{zy}}{\partial z}\right] - F_y \\ \rho \frac{\partial^2 u_z}{\partial t^2} & = & \left[\frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z}\right] - F_z \end{array} \tag{2.22}$$

Las ecuaciones (2.22) se denominan ecuaciones de *Navier*, que supone un sólido perfectamente elástico, homogéneo e isótropo, en el que las deformaciones y los esfuerzos son infinitesimales.

Para el caso de interés de esta tesis, que es el de la propagación de ondas sísmicas en un medio 3D homogéneo, isótropo y elástico es conveniente expresar las ecuaciones (2.22) mediante un sistema de 9 ecuaciones en derivadas parciales, para los tres componentes del vector de velocidad $v_i\left(\vec{x}, t\right)$ y los seis componentes independientes del tensor de esfuerzos $\sigma_{ij}\left(\vec{x}, t\right)$. Donde i,j =1,2,3 y el tensor de esfuerzo es simétrico; es decir, $\sigma_{ij} = \sigma_{ji}$. Las ecuaciones 3D de velocidad-esfuerzo están expresadas en las ecuaciones (2.23) y (2.24) [4]

$$\frac{\partial v_i\left(\overrightarrow{x}, t\right)}{\partial t} - b\left(\overrightarrow{x}\right) \frac{\partial \sigma_{ij}\left(\overrightarrow{x}, t\right)}{\partial x_j} = b\left(\overrightarrow{x}\right)\left[f_i\left(\overrightarrow{x}, t\right) + \frac{\partial m_{ij}^a\left(\overrightarrow{x}, t\right)}{\partial x_j}\right] \tag{2.23}$$

$$\frac{\partial \sigma_{ij}\left(\overrightarrow{x}, t\right)}{\partial t} - \lambda\left(\overrightarrow{x}\right) \frac{\partial v_k\left(\overrightarrow{x}, t\right)}{\partial x_k}\delta_{ij} - \mu\left(\overrightarrow{x}\right)\left[\frac{\partial v_i\left(\overrightarrow{x}, t\right)}{\partial x_j} + \frac{\partial v_j\left(\overrightarrow{x}, t\right)}{\partial x_i}\right] = \frac{\partial m_{ij}^s\left(\overrightarrow{x}, t\right)}{\partial t} \tag{2.24}$$

;donde $\overrightarrow{x} \in R^3$, $b = 1/\rho$ es la masa y λ y μ son las constantes de *Lamé*. Las fuentes de las ondas sísmicas están definidas por f_i (fuente de la fuerza) o m_{ij} (fuente del momento). Las componentes simétricas y antisimétricas del tensor de densidad de momento están descritas mediante las ecuaciones (2.25) y (2.26), respectivamente

$$m_{ij}^{s}\left(\overrightarrow{x},t\right)=1/2\left[m_{ij}\left(\overrightarrow{x},t\right)+m_{ji}\left(\overrightarrow{x},t\right)\right] \tag{2.25}$$

$$m_{ij}^{a}\left(\overrightarrow{x},t\right)=1/2\left[m_{ij}\left(\overrightarrow{x},t\right)-m_{ji}\left(\overrightarrow{x},t\right)\right] \tag{2.26}$$

La condición de frontera de tracción (o la componente normal de esfuerzo) debe de satisfacer $\sigma_{ij}\left(\overrightarrow{x},t\right)n_j\left(\overrightarrow{x}\right)=t_i\left(\overrightarrow{x},t\right)$; con $\overrightarrow{x}$ en $\boldsymbol{S}$, donde $t_i\left(\overrightarrow{x},t\right)$ son los componentes del vector de tracción superficiales de variación temporal y $n_i\left(\overrightarrow{x}\right)$ son los elementos del vector unitario externos normales a $\boldsymbol{S}$. Las condiciones iniciales sobre las variables dependientes están especificadas en todas las partes de $\boldsymbol{V}$ ecuación (2.27) y $\boldsymbol{S}$ ecuación (2.28) en el tiempo $t=t_0$ como

$$v_i\left(\overrightarrow{x},t_0\right)=v_i^0\left(\overrightarrow{x}\right) \tag{2.27}$$

$$\sigma_{ij}\left(\overrightarrow{x},t_0\right)=\sigma_{ij}^0\left(\overrightarrow{x}\right) \tag{2.28}$$

2.2. MÉTODOS NUMÉRICOS.

Resolver una ecuación de forma analítica es difícil en la mayoría de los casos, a menos que la ecuación sea extremadamente sencilla. Las dificultades encontradas al buscar soluciones analíticas se pueden clasificar dentro de los siguientes casos [5]:

1. El algoritmo es simple, pero el orden de la ecuación o la cantidad de cálculos a realizar es muy grande.
2. La ecuación es multidimensional.
3. La geometría es compleja.
4. No se conoce una solución o un procedimiento analítico.
5. Una solución analítica es posible, pero es complicada y costosa.

Los ingenieros y científicos han optado por aproximaciones experimentales para la mayoría de los complicados sistemas reales. Aunque, obviamente existen severas limitantes de estas aproximaciones, tales como errores experimentales y la naturaleza burda de los resultados. Actualmente es imposible separar la utilización de la computadora en el diseño, análisis y simulación de sistemas en las tecnologías más avanzadas.

Los métodos numéricos son los procedimientos matemáticos basados en operaciones aritméticas para los cuales las computadoras calculan la solución de las ecuaciones matemáticas. Dada la naturaleza digital de las computaciones existen varias diferencias entre los métodos numéricos y las aproximaciones analíticas. La principal diferencia es que los espacios continuos no pueden ser representados por una memoria principal finita; es decir, se requieren elevados requerimientos de recursos de cómputo.

De forma genérica, los problemas físicos que involucran ecuaciones diferenciales parciales cuya solución puede obtenerse mediante métodos numéricos computacionales se clasifican en

tres clases: *(a)* problemas de valores iniciales, cuya solución describe la evolución temporal de las variables involucradas a partir de los valores iniciales de las funciones en el tiempo $t = 0$; *(b)* problemas de valores en las fronteras, que son problemas estáticos en los que la solución debe verificar, en particular, los valores de las funciones en cierta parte del dominio; es decir, en las fronteras de éste y *(c)* problemas de valores iniciales y con valores en las fronteras, como es el caso del problema de esta tesis. Tradicionalmente, la solución numérica de ecuaciones diferenciales parciales es realizada mediante los siguientes métodos: de diferencias finitas, de elemento finito y volumen finito. Sólo el primero de ellos será descrito, ya que es el empleado en este trabajo.

2.2.1. ESQUEMA DE DIFERENCIAS FINITAS.

Los métodos numéricos en diferencias finitas están basados en el concepto básico de la derivada, cuya forma analítica es

$$\frac{du\,(x)}{dx} = \lim_{\Delta x \to 0} \frac{u\,(x + \Delta x) - u\,(x)}{\Delta x} \tag{2.29}$$

En la ecuación (2.29) se supone que la función u es continua y Δx, el denominador, tiende arbitrariamente a cero según la teoría de límites.

En los esquemas numéricos en diferencias finitas, la diferenciación pasa a ser un proceso discreto, en el que se aproxima de forma general como

$$\frac{du\,(x)}{dx} \approx \frac{u\,(x + \Delta x) - u\,(x)}{\Delta x} \tag{2.30}$$

;donde Δx es un parámetro numérico fijo que puede ajustarse bajo ciertas condiciones. La función u es evaluada en dos o mas puntos de una malla discreta, que proporciona soporte a la solución numérica con un orden de convergencia algebraico.

Comúnmente, la función u es multivariable de dos, tres o más variables espaciales y una temporal, pero sin soslayar dicha cantidad, el dominio de las variables físicas asociadas al problema se discretiza en un número variable de puntos, que denominaremos celdas, generando la malla antes citada. Al aplicar el esquema de diferencias finitas sobre la región de estudio nos proporciona como resultado conocer el valor de la función incógnita en cada celda.

Para el caso de una dimensión, las variables espacial y temporal son discretizadas como se indica

$$\begin{array}{ll} x_i = x_0 + i\Delta x & i = 0, 1, 2, ..., I \\ t_n = t_0 + n\Delta t & j = 0, 1, 2, ..., N \end{array} \tag{2.31}$$

Entonces, cualquier punto de la malla, celda, es descrito por sus coordenadas (j, n) y el valor de la función u en esa celda se escribe como u_j^n. Por lo tanto, las derivadas de u con respecto a x y t podemos escribirlas como

$$\begin{array}{l} \frac{\partial u}{\partial x}\big|_{j,n} = \frac{u_j^{n+1} - u_j^n}{\Delta x} \\ \frac{\partial u}{\partial t}\big|_{j,n} = \frac{u_j^{n+1} - u_j^n}{\Delta t} \end{array} \tag{2.32}$$

Las derivadas (2.32) pueden ser calculadas empleando expresiones distintas, pero ello afectaría el desempeño del esquema referente a, entre otros criterios, la estabilidad y convergencia. Si elegimos, por ejemplo, el esquema "*forward time centered space*" (*FTCS*, espacialmente centrada y temporalmente recorrido hacia adelante; por sus siglas en inglés), existen otros como hacia delante, atrás o centrales, tendríamos las siguientes ecuaciones

$$\begin{array}{l} \frac{\partial u}{\partial x}|_{j,n} = \frac{u^n_{j+1} - u^n_{j-1}}{\Delta x} \\ \frac{\partial u}{\partial t}|_{j,n} = \frac{u^{n+1}_j - u^n_j}{\Delta t} \end{array} \tag{2.33}$$

Además, existen otros criterios, de los ya citados, como por ejemplo para evaluar si el esquema es siempre estable, siempre inestable o condicionalmente estable. Tenemos, también esquemas explícitos e implícitos. Un esquema explícito es aquel cuyo término u^n_j en una celda se calcula directamente a partir de valores de dicho término calculado previamente en otras celdas de la malla; mientras que en el esquema implícito el valor de u^n_j se calcula a partir de sistemas de ecuaciones que involucran varios términos u para valores diferentes de n y/o j. La implementación del esquema explícito es más sencillo, relativamente, que los implícitos –en 3D en particular- y no es necesario resolver sistemas de ecuaciones en cada paso de cálculo; sin embargo, requiere atención especial con el criterio de estabilidad, que implica pasos pequeños de cálculo y revisión de condiciones críticas periódicamente.

2.2.2. ESQUEMA DE DIFERENCIAS FINITAS ALTERNADAS.

El algoritmo de diferencias finitas utilizado en este trabajo es un esquema explícito de segundo orden de precisión en el tiempo y de cuarto orden en el espacio. El almacenamiento alternado de las celdas permite que las derivadas parciales sean aproximadas mediante diferencias finitas centrales sin duplicar la extensión espacial de los operadores, proporcionando así mayor precisión [6]. Aplicamos el esquema de diferencias finitas a las ecuaciones (2.23) y (2.24) después de eliminar las unidades de medidas. Si elegimos una discretización espacial 3D de la malla de tal manera que

$$\begin{array}{c} x_i = x_0 + (i-1)\, h_x \ , \quad y_j = y_0 + (j-1)\, h_y \ , \quad z_k = z_0 + (k-1)\, h_z \\ i = 1, 2, ...I \quad , \quad j = 1, 2, ..., J \quad , \quad k = 1, 2, ..., K \end{array} \tag{2.34}$$

;donde x_0, y_0, z_0 son los valores mínimos de la malla y h_x, h_y, h_z proporcionan la distancia entre los puntos en cada uno de los tres ejes coordenados (x, y, z). Los valores de los dos parámetros de *Lamé* y la masa en la celda están dados por $\lambda(x_i, y_j, z_k)$, $\mu(x_i, y_j, z_k)$ y $b(x_i, y_j, z_k)$, respectivamente. La discretización en el tiempo se define como $t_l = t_0 + (l-1)h_t$, para l=1, 2, 3,. . . ,L; donde t_0 es el tiempo mínimo y h_t es el incremento en el tiempo.

Subconjuntos de las variables dependientes son almacenados en distintas mallas espaciales y temporales. En particular, los componentes de la diagonal del tensor de esfuerzos son almacenados en los puntos de la malla antes definidos:

$$\sigma_{xx}\,(x_i, y_j, z_k, t_l)\,, \quad \sigma_{yy}\,(x_i, y_j, z_k, t_l)\,, \quad \sigma_{zz}\,(x_i, y_j, z_k, t_l) \tag{2.35}$$

Los componentes no diagonales del tensor de esfuerzos son desplazados en el espacio medio intervalo de la malla a lo largo de dos ejes coordenados mediante la ecuación (2.36)

$$\begin{array}{c} \sigma_{xy}\left(x_i+h_x/2,y_j+h_y/2,z_k,t_l\right), \quad \sigma_{yz}\left(x_i,y_j+h_y/2,z_k+h_z/2,t_l\right), \\ \sigma_{xz}\left(x_i+h_x/2,y_j,z_k+h_z/2,t_l\right) \end{array} \tag{2.36}$$

Por último, las tres componentes del vector de velocidad son almacenadas en los puntos de la malla que son desplazados, tanto en el espacio como en el tiempo, medio intervalo de la malla, ecuación (2.37)

$$v_x\left(x_i+h_x/2,y_j,z_k,t_l+h_l/2\right), \quad v_y\left(x_i,y_j+h_y/2,z_k,t_l+h_t/2\right), \quad v_z\left(x_i,y_j,z_k+h_z/2,t_l+h_t/2\right) \tag{2.37}$$

Podemos definir, también, las siguientes cantidades escalares frecuentemente utilizadas

$$\begin{array}{c} p_x=\frac{c_1h_tS_w}{h_x}, p_y=\frac{c_1h_tS_w}{h_y}, p_z=\frac{c_1h_tS_w}{h_z} \\ q_x=\frac{c_2h_tS_w}{h_x}, q_y=\frac{c_2h_tS_w}{h_y}, q_z=\frac{c_2h_tS_w}{h_z} \end{array} \tag{2.38}$$

;donde S_w es la unidad característica de medida de la velocidad de onda. De igual forma, podemos definir S_r, S_v y S_s como unidades características de medida de la densidad de masa, del vector de velocidad y de los componentes del tensor de esfuerzos, respectivamente. Como ejemplo, podríamos elegir S_w = 6,000 m/s y S_r = 2,100 kg/m^3 como valores normalizados adecuados y $c_1 = 9/8$ y $c_2 = -1/24$ como coeficientes en el operador de diferencias finitas de cuarto orden.

Entonces, las fórmulas de actualización en el tiempo para las tres variables representativas no conocidas: velocidad, los componentes de esfuerzo de la diagonal y los componentes de esfuerzo fuera de la diagonal están expresadas por las ecuaciones (2.39), (2.40) y (2.41), para $v_x,\ \sigma_{xx},\ \sigma_{xy}$, respectivamente. Expresiones similares se obtienen para las velocidades $v_y,\ v_z$ en las direcciones $y,\ z$; así como para los esfuerzos $\sigma_{xz},\ \sigma_{yy},\ \sigma_{yz},\ \sigma_{zz}$.

En la ecuación (2.39) la masa (b) entre las celdas de la malla es obtenida mediante interpolación. En cambio, no se requiere interpolación para los parámetros de *Lamé* en la ecuación (2.40). El modulo de corte o rigidez (μ) de la ecuación (2.41) entre las celdas de la malla es, también, obtenido mediante interpolación.

$$\begin{array}{l} v_x\left(x_i+h_x/2,y_j,z_k,t_l+h_t/2\right)=v_x\left(x_i+h_x/2,y_j,z_k,t_l-h_t/2\right) \\ +\, b\left(x_i+h_x/2,y_j,z_k\right)\{p_x\left[\sigma_{xx}\left(x_i+h_x,y_j,z_k,t_l\right)-\sigma_{xx}\left(x_i,y_j,z_k,t_l\right)\right] \\ +\, q_x\left[\sigma_{xx}\left(x_i+2h_x,y_j,z_k,t_l\right)-\sigma_{xx}\left(x_i-h_x,y_j,z_k,t_1\right)\right] \\ +\, p_y\left[\sigma_{xy}\left(x_i+h_x/2,y_j+h_y/2,z_k,t_l\right)-\sigma_{xy}\left(x_i+h_x/2,y_j-h_y/2,z_k,t_1\right)\right] \\ +\, q_y\left[\sigma_{xy}\left(x_i+h_x/2,y_j+3h_y/2,z_k,t_l\right)-\sigma_{xy}\left(x_i+h_x/2,y_j-3h_y/2,z_k,t_1\right)\right] \\ +\, p_z\left[\sigma_{xz}\left(x_i+h_x/2,y_j,z_k+h_z/2,t_l\right)-\sigma_{xz}\left(x_i+h_x/2,y_j,z_k-h_z/2,t_1\right)\right] \\ +\, q_z\left[\sigma_{xz}\left(x_i+h_x/2,y_j,z_k+3h_z/2,t_l\right)-\sigma_{xz}\left(x_i+h_x/2,y_j,z_k-3h_y/2,t_1\right)\right]\} \\ +\, b\left(x_i+h_x/2,y_j,z_k\right)\{\frac{h_t}{S_\rho S_v}f_x\left(x_i+h_x/2,y_j,z_k,t_l\right) \\ +\, p_y\left[m^a_{xy}\left(x_i+h_x/2,y_j+h_y/2,z_k,t_l\right)-m^a_{xy}\left(x_i+h_x/2,y_j-h_y/2,z_k,t_l\right)\right] \\ +\, q_y\left[m^a_{xy}\left(x_i+h_x/2,y_j+3h_y/2,z_k,t_l\right)-m^a_{xy}\left(x_i+h_x/2,y_j-3h_y/2,z_k,t_l\right)\right] \\ +\, p_z\left[m^a_{xz}\left(x_i+h_x/2,y_j,z_k+h_z/2,t_l\right)-m^a_{xz}\left(x_i+h_x/2,y_j,z_k-h_z/2,t_l\right)\right] \\ +\, q_z\left[m^a_{xz}\left(x_i+h_x/2,y_j,z_k+3h_z/2,t_l\right)-m^a_{xz}\left(x_i+h_x/2,y_j,z_k-3h_z/2,t_l\right)\right]\} \end{array} \tag{2.39}$$

$$
\begin{aligned}
&\sigma_{xx}\left(x_i, y_j, z_k, t_l + h_t\right) = \sigma_{xx}\left(x_i, y_j, z_k, t_l\right) + \left[\lambda\left(x_i, y_j, z_k\right) + 2\mu\left(x_i, y_j, z_k\right)\right] \\
&\times \left\{p_x\left[v_x\left(x_i + h_x/2, y_j, z_k, t_l + h_t/2\right) - v_x\left(x_i - h_x/2, y_j, z_k, t_l + h_t/2\right)\right]\right. \\
&\left. + q_x\left[v_x\left(x_i + 3h_x/2, y_j, z_k, t_l + h_t/2\right) - v_x\left(x_i - 3h_x/2, y_j, z_k, t_l + h_t/2\right)\right]\right\} \\
&+ \lambda\left(x_i, y_j, z_k\right)\left\{p_y\left[v_y\left(x_i, y_j + h_y/2, z_k, t_l + h_t/2\right) - v_y\left(x_i, y_j - h_y/2, z_k, t_l + h_t/2\right)\right]\right. \\
&+ q_y\left[v_y\left(x_i, y_j + 3h_y/2, z_k, t_l + h_t/2\right) - v_y\left(x_i, y_j - 3h_y/2, z_k, t_l + h_t/2\right)\right] \\
&+ p_z\left[v_z\left(x_i, y_j, z_k + h_z/2, t_l + h_t/2\right) - v_z\left(x_i, y_j, z_k - h_z/2, t_l + h_t/2\right)\right] \\
&\left. + q_z\left[v_z\left(x_i, y_j, z_k + 3h_z/2, t_l + h_t/2\right) - v_z\left(x_i, y_j, z_k - 3h_z/2, t_l + h_t/2\right)\right]\right\} \\
&+ \left[m^s_{xx}\left(x_i, y_j, z_k, t_l + h_t\right) - m^s_{xx}\left(x_i, y_j, z_k, t_l\right)\right]
\end{aligned}
\tag{2.40}
$$

Finalmente, podemos representar de forma gráfica el esquema de diferencias finitas alternadas (*staggered*) en la que podemos apreciar la ubicación de los elementos utilizados para realizar la actualización de velocidades y esfuerzos, en particular, para las ecuaciones (2.39), (2.40) y (2.41) en las Figs. 2.4, 2.5 y 2.6, respectivamente.

$$
\begin{aligned}
&\sigma_{xy}\left(x_i + h_x/2, y_j + h_y/2, z_k, t_l + h_t\right) = \sigma_{xy}\left(x_i + h_x/2, y_j + h_y/2, z_k, t_l\right) \\
&+ \mu\left(x_i + h_x/2, y_j + h_y/2, z_k\right) \\
&\times \left\{p_x\left[v_y\left(x_i + h_x, y_j + h_y/2, z_k, t_l + h_t/2\right) - v_y\left(x_i, y_j + h_y/2, z_k, t_l + h_t/2\right)\right]\right. \\
&\left. + q_x\left[v_y\left(x_i + 2h_x, y_j + h_y/2, z_k, t_l + h_t/2\right) - v_y\left(x_i - h_x, y_j + h_y/2, z_k, t_l + h_t/2\right)\right]\right. \\
&\left. + p_y\left[v_x\left(x_i + h_x/2, y_j + h_y, z_k, t_l + h_t/2\right) - v_x\left(x_i - h_x/2, y_j, z_k, t_l + h_t/2\right)\right]\right. \\
&\left. + q_y\left[v_x\left(x_i + h_x/2, y_j + 2h_y, z_k, t_l + h_t/2\right) - v_x\left(x_i + h_x/2, y_j - h_y, z_k, t_l + h_t/2\right)\right]\right\} \\
&+ \left[m^s_{xy}\left(x_i + h_x/2, y_j + h_y/2, z_k, t_l + h_t\right) - m^s_{xy}\left(x_i + h_x/2, y_j + h_y/2, z_k, t_l\right)\right]
\end{aligned}
\tag{2.41}
$$

En el capítulo 3 se presenta la implementación paralela del código serial correspondiente a la solución del sistema de ecuaciones (2.23) y (2.24) expresadas por las ecuaciones (2.39)–(2.41) y semejantes.

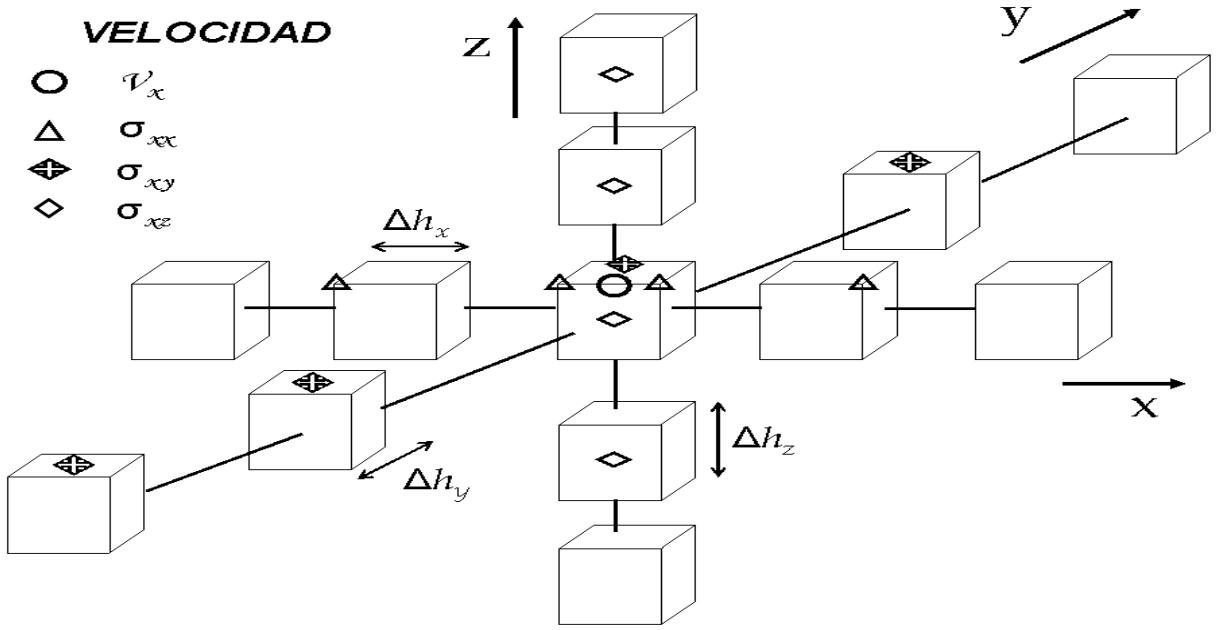


Figura 2.4: Se muestra el esquema de **DFA3D** para actualizar el valor de velocidad V_x en un tiempo dado y todos los esfuerzos asociados en dicho cálculo.

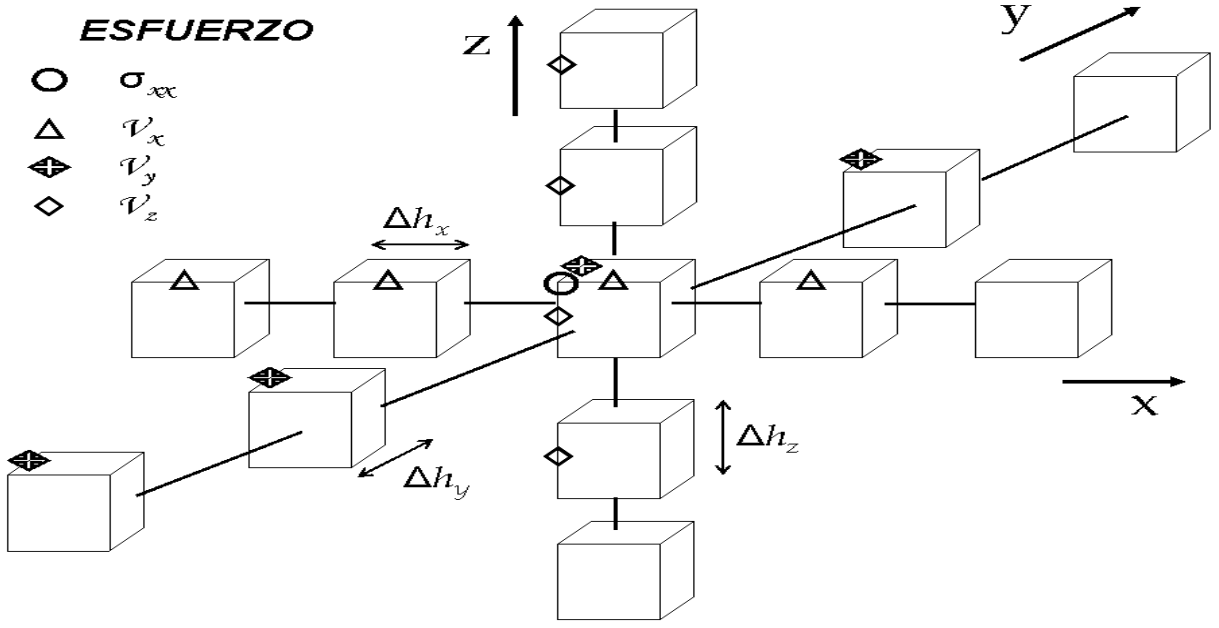


Figura 2.5: Representación gráfica del esquema de **DFA3D** para actualizar el valor de esfuerzo σ_{xx} en un tiempo determinado y todas las velocidades involucradas en dicha actualización.

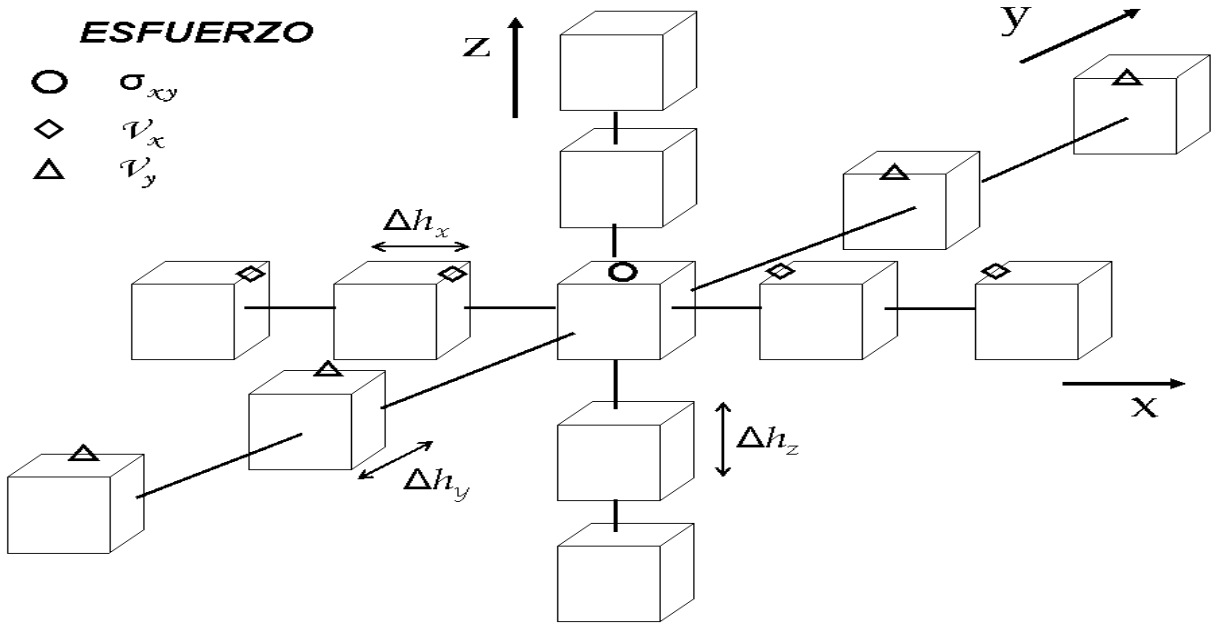


Figura 2.6: Esquema visual de **DFA3D** para actualizar el valor de esfuerzo σ_{xy} en un tiempo determinado y todas las velocidades asociadas en dicha actualización.

2.3. CONCLUSIONES DEL CAPÍTULO.

Se mencionan las conclusiones más relevantes de este capítulo:

1. Se discutió de manera sucinta la teoría de la elastodinámica por medio de la teoría de deformaciones, de desplazamientos y de esfuerzos hasta llegar a las ecuaciones de *Navier* y las ecuaciones constitutivas de la propagación de ondas sísmicas en medios sólidos, elásticos, homogéneos e isótropos.

2. Se discutió, también, la importancia de los métodos numéricos y los problemas físicos que involucran sistemas de ecuaciones diferenciales parciales que pueden resolverse de forma numérica. Dichos problemas generalmente se clasifican en tres tipos: problemas de valores iniciales, problemas de valores en las fronteras y problemas de valores iniciales con valores en las fronteras.

3. Se expuso el método de diferencias finitas alternadas, que es una variante al de diferencias finitas, de segundo orden en el tiempo y cuarto orden en el espacio que permite optimizar el uso de la memoria al no duplicar la extensión espacial de los operadores, proporcionando así mayor precisión.

4. Se incluyen las expresiones que permiten el cálculo de las velocidades y los esfuerzos asociados a la propagación de ondas sísmicas en un medio elástico, homogéneo e isótropo discretizado mediante el método de diferencias finitas alternadas.

Capítulo 3

PARALELIZACIÓN DEL PROGRAMA SERIAL.

La implementación paralela de un programa o algoritmo puede hacerse a partir de una versión serial, como es el caso de este trabajo, o desde su concepción se diseña e implementa una versión paralela del algoritmo, sin necesidad de crear la versión serial intermedia.

La paralelización de un algoritmo tiene dos objetivos primarios: 1) incrementar el tamaño del dominio del problema, permitiendo con esto obtener resultados más confiables y precisos, al aumentar la resolución del problema y 2) disminuir los tiempos de ejecución de los programas, permitiendo realizar más simulaciones en menos tiempo.

Además de estos objetivos específicos, los recursos disponibles de los equipos de Cómputo de Alto Rendimiento (*CAR*) deben de emplearse de forma óptima, en particular, los de la supercomputadora adquirida por la *UNAM* e inaugurada en 2007, *KanBalam* [7], la cual es empleada para la obtención de sismogramas sintéticos (capítulo 4) en este trabajo.

En este capítulo se comentarán sucintamente las arquitecturas de computadoras; algunas formas de paralelización, las cuales están en función de la arquitectura en la cual se ejecutará el programa; se introducirán los conceptos y las métricas, que nos permiten evaluar el desempeño de los programas paralelos y, finalmente, se explicará el método empleado en la paralelización del algoritmo para la generación de sismogramas sintéticos.

3.1. ARQUITECTURAS DE COMPUTADORAS.

Es importante el estudio de los dispositivos que componen a las computadoras y supercomputadoras [8, 9] para entender lo que sucede con el flujo de los datos e instrucciones entre la memoria y las unidades de procesamiento. De otra manera se tendría dificultad en entender las técnicas de optimizaciones paralelas y vectoriales y la información proporcionada por las herramientas y compiladores que realizan algunas optimizaciones automáticas de los programas.

3.1.1. CLASIFICACIÓN DE *FLYNN.*

Esta clasificación está basada en el flujo que tienen los datos dentro de la computadora y de las instrucciones sobre dichos datos; es decir, en el número de flujos de instrucciones y de datos que pueden ser simultáneamente procesados. Esta clasificación es útil por su simplicidad y amplio uso. *Flynn* [10] clasifica los sistemas de cómputo en cuatro categorías, como se muestra en la Fig. 3.1:

- *Single Instruction, Single Data (SISD*, una instrucción, un dato; por sus siglas en inglés).

  Esta categoría corresponde a la definición clásica de computadora, según la arquitectura *Von Neumann*, en la cual se ejecuta una sola instrucción sobre un único dato, Fig. 3.1a).

- *Single Instruction, Multiple Data (SIMD*, una instrucción, múltiples datos; por sus siglas en inglés).

  Estos sistemas tienen un único flujo de instrucciones que operan sobre múltiples datos. El procesamiento es síncrono, pero la ejecución de las instrucciones continúa siendo secuencial, todos los procesadores ejecutan la misma instrucción sobre grandes cantidades de datos. Esquemáticamente representada en la Fig. 3.1b). Las máquinas vectoriales son ejemplos de este tipo de sistema.

- *Multiple Instruction, Single Data (MISD*, múltiples instrucciones, un dato; por sus siglas en inglés).

  No existen producciones masivas. Los arreglos sistólicos son ejemplos de esta categoría, Fig. 3.1c).

- *Multiple Instruction, Multiple Data (MIMD*, múltiples instrucciones, múltiples datos; por sus siglas en inglés).

  Estos sistemas tienen múltiples flujos de instrucciones que operan sobre múltiples datos. Abarca todo el espectro de sistemas de multiprocesadores, que involucra un vasto número de topologías de interconexión entre los distintos elementos de los sistemas, de procesadores y arquitecturas. El factor importante en los sistemas *MIMD* es que cada procesador opera de forma independiente de los otros ejecutando, potencialmente, programas diferentes. Los procesadores se comunican a través de una red de alta velocidad, que permite que los procesadores compartan datos y sincronicen los cálculos, Fig. 3.1d).

La categoría *MIMD*, a su vez, es usualmente subdividida en un par de clases:

Single Program, Multiple Data (SPMD, un programa, múltiples datos; por sus siglas en inglés). Múltiples procesadores autónomos ejecutan de forma simultánea el mismo programa –en lugares independientes del código- sobre distintos datos. También es conocido como *Single Process, Multiple Data* (un proceso, múltiples datos; por sus siglas en inglés). Este es el paradigma de programación paralela más común.

Multiple Program, Multiple Data (MPMD, múltiples programas, múltiples datos; por sus siglas en inglés). Múltiples procesadores autónomos simultáneamente ejecutan, al menos, dos programas independientes entre sí.

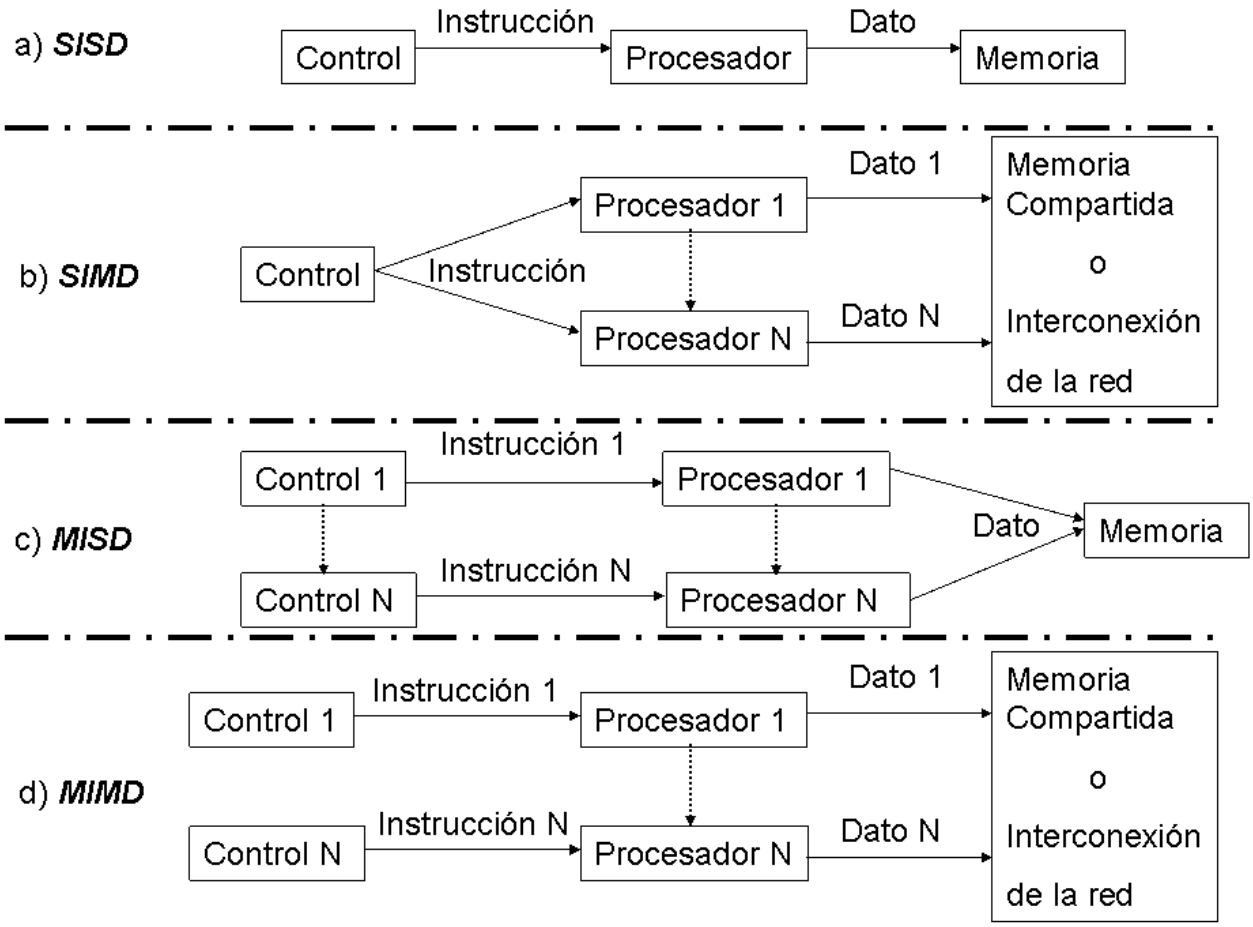


Figura 3.1: Taxonomía de *Flynn* dividida en sus cuatro categorías. Tanto en a) como en b) una unidad de control regula la ejecución de la instrucción que opera sobre uno o múltiples datos de manera síncrona. En c) y d) varias unidades de control regulan las instrucciones realizadas sobre múltiples datos, su ejecución puede ser tanto síncrona como asíncrona y de uno o más programas.

3.1.2. CÓMPUTO DE ALTO RENDIMIENTO.

Para realizar simulaciones de *CAR* es necesario emplear computadoras con múltiples procesadores, los cuales trabajan coordinadamente en la solución de un algoritmo. Estas computadoras pueden ser desde cúmulos de algunos procesadores hasta supercomputadoras –término utilizado para definir a las computadoras con vastos recursos de procesamiento, memoria principal y almacenamiento interconectados entre sí todos estos recursos con redes de interconexión de gran ancho de banda– de cientos de miles de procesadores. Éstas pueden clasificarse según el

tipo de procesador empleado, la organización física de la memoria, el esquema de paralelización soportado y la manera en que se comunican los múltiples procesadores.

El *CAR* se caracteriza por una gran demanda de procesamiento de datos empleando masivamente los recursos de *hardware* (*HW*) disponibles. Los problemas que el *CAR* se enfoca en resolver principalmente son los problemas de *Gran Reto* (término acuñado por *Wilson* [11]), los cuales *"son problemas fundamentales en ciencia e ingeniería que tienen un amplio impacto científico, social, político y económico, cuyas soluciones pueden ser desarrolladas aplicando técnicas y recursos de CAR*"; estos problemas son dinámicos. Los grandes retos del *CAR* son aquellos proyectos que son demasiado difíciles de investigar y que no pueden ser resueltos en una cantidad de tiempo razonable aun con las computadoras más rápidas y eficientes. Las características de estos problemas son, entre otras:

- Necesidad de soluciones de mayor complejidad.
  - Gran cantidad de variables, elevados requerimientos de memoria y más capacidad de procesamiento.
  - Conjuntos de datos grandes requieren un post-procesamiento considerable y/o visualización en 2D y 3D.
- El tipo de escala involucrada en la solución del problema.
  - Escalas demasiado grandes o pequeñas, las cuales son difíciles de comparar de manera experimental; v.gr.: la simulación de nanopartículas o problemas astronómicos:

Ejemplos de problemas de gran reto podemos citar:

- Dinámica molecular.
- Modelación y predicción del clima.
- Modelación del genoma.
- Astrofísica.
- Estructura electrónica de materiales.
- Farmacología.
- Diseño aeroespacial.
- Dinámica de fluidos.
- Modelación sísmica.
- Visualización biomédica y biomecánica.
- Reconocimiento de lenguaje.
- Modelación de la combustión.

3.2. PROCESADORES.

El procesador es el elemento más importante del equipo de cómputo. Éstos han evolucionado desde su aparición, pasando de poder hacer sólo unas cuantas instrucciones por segundo hasta realizar millones. Han pasado de ser procesadores sencillos, RISC, CISC, vectoriales, escalares y superescalares [8, 9] llegando a la actualidad donde los microprocesadores conjuntan múltiples núcleos –*cores*, procesadores incrustados en un solo circuito integrado (*chip*)– y se utilizan, también, procesadores gráficos para realizar cómputo.

3.2.1. MULTINÚCLEOS (*MULTICORES*).

Actualmente, la tendencia presentada por la tecnología de semiconductores caracterizada, en particular, por la ley de *Moore* [12, 13] –la capacidad de procesamiento de los equipos se duplica cada dos años- impone desafíos significativos al procesador, al sistema y a los diseñadores de *software* (*SW*). Un estado crítico se ha alcanzado en la explotación de la complejidad de la lógica en el diseño del procesador, que aunado con los problemas relacionados al consumo de energía ha forzado a la industria a la estrategia de integrar múltiples núcleos (*cores*) –procesadores incrustados en un solo circuito integrado (*chip*). Esta tecnología multinúcleo implica un cambio dramático de prácticas convencionales previas e involucra nuevos desafíos generados para el uso eficaz de ésta en el *CAR*.

3.2.2. *FPGA*.

El *FPGA* (arreglo de compuertas programables por campo, por sus siglas en inglés) es un dispositivo semiconductor que contiene bloques lógicos cuya interconexión y funcionalidad puede programarse. Los *FPGA*s son el resultado de la convergencia de dos tecnologías diferentes: los dispositivos lógicos programables (*PLD*, por sus siglas en inglés) y los circuitos integrados de aplicación específica (*ASIC*, por sus siglas en inglés). Entre las características más relevantes podemos citar:

- consume menos potencia que el procesador convencional.
- al utilizarlos como aceleradores pueden incrementar de manera significativa la capacidad de cómputo.
- pueden proporcionar un incremento importante en el rendimiento de ciertas aplicaciones.

El reto que encaran los programadores de *FPGA*s es asegurarse que los algoritmos y la representación numérica empleada utilizan al máximo sus recursos. Ello implica, en la mayoría de los casos, que se deba reescribir el programa por completo. Tradicionalmente, los *FPGA*s son programados mediante *HDL* (lenguaje de descripción de *HW*, por sus siglas en inglés) tal como *VHDL* (lenguaje para descripción y modelado de circuitos, por sus siglas en inglés), acrónimo de VHSIC (circuito integrado de muy alta velocidad, por sus siglas en inglés) y *HDL* [14].

3.2.3. *GPU.*

Un *GPU* (unidad de procesamiento gráfico, por sus siglas en inglés) es un dispositivo dedicado a generar gráficos para computadoras, estaciones de trabajo o consolas de videojuegos. Éstos pueden superar en rendimiento a los procesadores convencionales en ciertas aplicaciones dado su paralelización intrínseca del dispositivo. Además, puede ejecutar una sóla instrucción sobre muchos datos al mismo tiempo utilizando el esquema *SIMD*. Esta paralelización inherente es el resultado de su arquitectura. Las tarjetas gráficas están constituidas por un arreglo de multiprocesadores, cada uno posee su propia sección de ancho de banda. Hasta hace poco, un gran obstáculo en el desarrollo de aplicaciones de propósito general para *GPU*s era la complejidad de su programación. La única manera de acceder al dispositivo era mediante paquetes gráficos como *OpenGL* [15, 16] o utilizando un lenguaje ensamblador especial para la tarjeta. Sin embargo, debido al potencial en cómputo de los *GPU*s para realizar cómputo númerico se han desarrollado paquetes como *CUDA* (arquitectura de dispositivo de cómputo unificada, por sus siglas en inglés) [17] y rutinas de álgebra lineal como la biblioteca *CUBLAS* (subprogramas de álgebra lineal básica de cómputo unificado) [18].

3.2.4. *CELL.*

El microprocesador *CELL* (arquitectura de motor *Cell* de banda ancha, por sus siglas en inglés) consiste de un núcleo del procesador *PowerPC* [19, 20], de propósito general, conectado a ocho núcleos *DSP* [21], de propósito especial, llamados *unidades sinergísticas de procesamiento* (*SPE*), que son el corazón del concepto de este procesador. *CELL* está diseñado para cubrir el espacio existente entre los microprocesadores convencionales de las computadoras personales y los procesadores especializados de alto rendimiento como los *GPU*s y *FPGA*s y optimizado para realizar cómputo numérico de punto flotante de precisión simple. Los ocho *SPE*s constituyen esencialmente una computadora vectorial mediante la simplificación de la arquitectura del microprocesador, trasladando las técnicas para ocultar la latencia e incrementar el paralelismo a nivel de instrucciones y su complejidad al *SW*.

La consola de videojuegos *PlayStation3* de *Sony* contiene la primera aplicación comercial de este microprocesador. La supercomputadora clasificada número uno en la lista más reciente (junio 2008) de las 500 supercomputadoras más rápidas del orbe (*top500*) [22] es una supercomputadora híbrida con procesadores *Cell* trabajando en conjunto con procesadores *x86 AMD* capaz de alcanzar un rendimiento de *Petaflops* [1] [23, 24, 25, 26, 27].

3.3. COMPUTADORAS VECTORIALES, PARALELAS Y CÚMULOS.

3.3.0.1. COMPUTADORAS VECTORIALES.

Una computadora vectorial o procesador vectorial contiene un conjunto de unidades aritméticas especiales conocidas como “unidades funcionales” o “encadenamientos aritméticos” (regis-

[1] *flops* significa operaciones de punto flotante por segundo

tros *pipelined*). Estas unidades son utilizadas para procesar los elementos de vectores o arreglos eficientemente, intercalando la ejecución de diferentes partes (etapas) de una operación aritmética sobre elementos distintos del vector, a medida que recorren dicha unidad funcional. Estos procesadores contienen registros vectoriales para mantener varios elementos -a la vez- de un vector y operar sobre ellos en un solo ciclo de reloj, en lugar de hacerlo en varios ciclos como con las operaciones escalares. Un ejemplo de este tipo de computadoras es la *CRAY-YMP*, la primer supercomputadora de la *UNAM*, que llegó en 1991. En la más reciente lista del (*top500*) [22] solamente hay 2 supercomputadoras de este tipo que representan el 0.40 % de dicha lista con un rendimiento máximo de 59,293 *Gigaflops* y 6,134 procesadores.

3.3.0.2. COMPUTADORAS PARALELAS.

Son computadoras con dos o más procesadores que pueden trabajar simultánea y coordinadamente para resolver un problema de cómputo. Una computadora de este tipo es también llamada multiprocesador. Éstas pueden ser subdivididas en dos tipos principalmente: *MIMD* y *SIMD*, según la clasificación de *Flynn*. En otras palabras, los procesadores de un multiprocesador *MIMD* pueden actuar independientemente uno de otro con distintas instrucciones sobre diferentes datos, mientras que los procesadores de una computadora *SIMD* ejecutan la misma instrucción al mismo tiempo pero con datos distintos. La *Origin2000* [28], la segunda supercomputadora de la *UNAM* es un ejemplo de tipo *MIMD* de computadora paralela. Empleaba procesadores superescalares por el gran rendimiento que proporcionaban.

3.3.0.3. CÚMULOS DE COMPUTADORAS.

Son un conjunto de computadoras de arquitecturas homogéneas o heterogéneas conectadas en red -bajo cualquier topología- que resuelven problemas de cómputo distribuido o paralelo y funcionan como una sola computadora. El principal problema estriba en las conexiones de la red, que no está exenta a que se vea afectada por diversos factores, incluido el tipo del problema a resolver. Sin embargo, recientemente han ocurrido avances importantes hechos para incluir a los cúmulos de computadoras como una de las más importantes formas de hacer *CAR* y más aún con los adelantos en las tecnologías de redes de gran ancho de banda (la cantidad de datos que se pueden transmitir en una unidad de tiempo) y baja latencia (costo de inicio, en tiempo, para que un paquete de información se transfiera) no propietarias y de costo medianamente asequible. La supercomputadora *KanBalam* es un ejemplo de esta categoría. Los cúmulos constituyen el 80 % del *top500* con 400 sistemas instalados y un rendimiento de 7,438,172 *Gigaflops* y 1,260,698 procesadores, por mucho, el sistema dominante en la actualidad.

3.3.1. ORGANIZACIÓN DE LA MEMORIA.

La categoría de multiprocesadores *MIMD* puede dividirse, también, en dos nuevos subtipos, según la organización física de la memoria, la cual impone, entre otras cosas, el esquema de paralelización a emplear para obtener un buen rendimiento. Dichos tipos son:
memoria compartida y memoria distribuida.

3.3.1.1. MEMORIA COMPARTIDA.

Un multiprocesador *MIMD* de memoria compartida tiene una sola memoria que puede ser accesada por todos los procesadores. Un problema importante en este tipo de computadoras es la escalabilidad; ésta es la facilidad para incrementar el número de procesadores y otros elementos de *HW* con un correspondiente incremento en el rendimiento. Conforme se incrementa el número de procesadores de igual manera aumenta el tráfico en el *bus* que conecta a los procesadores con la memoria compartida. Incrementando el ancho de banda de la red de interconexión se soluciona parcialmente el problema. Ejemplos de este tipo de computadoras son las *SMP* (multiprocesamiento simétrico; por sus siglas en inglés), las cuales son simétricas desde el punto de vista del tiempo de acceso de los múltiples procesadores a la memoria compartida.

3.3.1.2. MEMORIA DISTRIBUIDA.

Cada uno de los procesadores individuales de un multiprocesador de memoria distribuida tiene asociado directamente a él una unidad de memoria; esto es, cada procesador tiene su propia memoria local. Un procesador o conjunto de procesadores, junto con su memoria es llamado "*nodo*" –el cual es esencialmente una computadora independiente con sistema operativo propio– y está conectado a los otros nodos mediante algún tipo de red. Estas redes pueden ser de algunos de los tipos mostrados en la Fig. 3.2.

A esta categoría de computadoras de les conoce comúnmente *MPP* (máquinas masivamente paralelas, por sus siglas en inglés) que tienen decenas de miles de procesadores. En la lista actual del *top500* el número de estas computadoras asciende a 98 que representa el 19.60 % de las computadoras de dicha lista, con un rendimiento máximo total de 5,238,473 *Gigaflops* y 1,135,873 procesadores.

3.4. MODELOS Y PARADIGMAS DE PARALELIZACIÓN.

Existen distintos esquemas de programación paralela, los cuales están en función tanto de las arquitecturas de las computadoras como de los lenguajes de programación utilizados, aunque en algunas arquitecturas es posible implementar más de un método de programación.

3.4.1. MODELOS DE PARALELIZACIÓN.

3.4.1.1. PARALELISMO DE DATOS.

En el paralelismo de datos (*Data Parallelism*), o partición de datos, el dominio del problema es distribuido entre los procesadores, de tal manera que cada procesador resuelve su propio subdominio del problema, independientemente de los otros procesadores. Las mismas instrucciones son ejecutadas en distintos procesadores, comúnmente llamado *SIMD* o *SPMD*.

La descomposición en subdominios puede hacerse en una, dos o tres dimensiones. Las bibliotecas de intercambio de mensajes son los paradigmas usualmente empleados, ejemplos de éstas son, entre otras: *PVM* (máquina virtual paralela; por sus siglas en inglés) [29] y *MPI* (interfaz

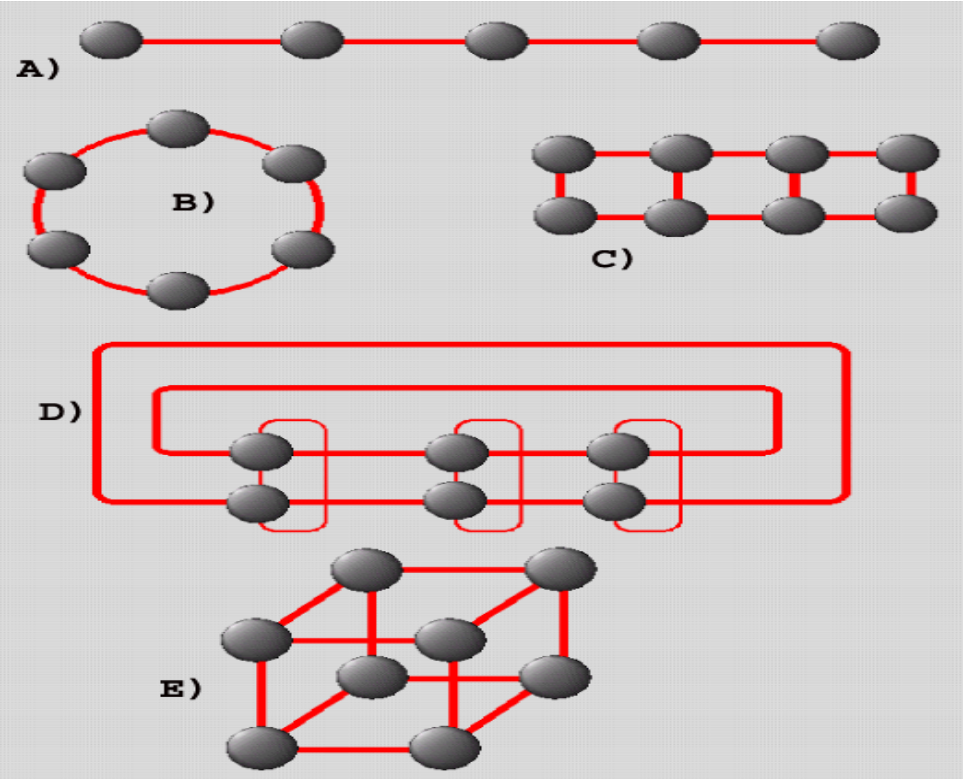


Figura 3.2: Distintas topologías de interconexión entre los procesadores o nodos de una computadora: A) *Bus*, B) Anillo, C) Malla, D) Torus y E) Hipercubo.

de envío de mensajes; por sus siglas en inglés) [30, 31]. Esta última es el estándar *de facto*, con una implementación propietaria (hecha por los fabricantes) en prácticamente todas las arquitecturas, las cuales aprovechan todas las características del *HW* de dichas computadoras y además es portátil. Este paradigma de paralelización es bastante escalable y el más empleado. Ilustrado gráficamente en la Fig. 3.3.

3.4.1.2. PARALELISMO DE FUNCIONES.

Consiste en dividir una aplicación en diferentes tareas que puedan ser ejecutadas de manera independiente o de forma asíncrona. En este paradigma, el enfoque se centra en el cómputo que es realizado más que en los datos manipulados por aquel. A este tipo de paralelismo se le conoce, también, como descomposición funcional. Esquemáticamente, este tipo de paralelización es mostrado en la Fig. 3.4.

3.4.1.3. MEMORIA COMPARTIDA.

En el modelo de memoria compartida las tareas comparten un espacio de direcciones común, en el cual leen y escriben dichas tareas de forma asíncrona. Existen varios mecanismos que controlan el acceso al espacio común de memoria compartida, los cuales, entre otros, son: los

candados, semáforos y barreras.

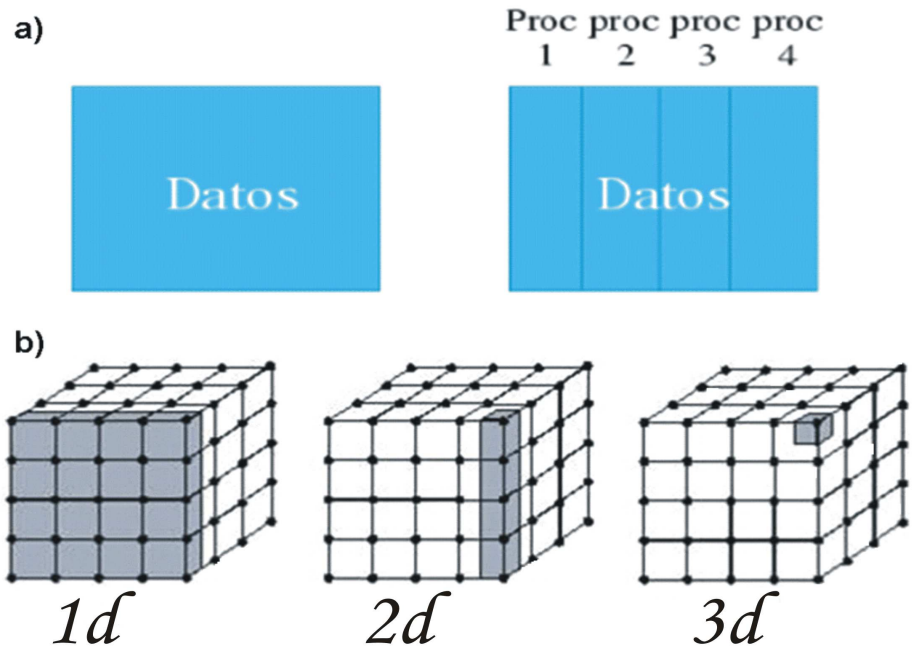


Figura 3.3: Ilustrado por incisos el dominio bidimensional del problema a) se divide de manera vertical en 4 procesadores *1d*. Dado un dominio de problema tridimensional b) la paralelización de los datos se puede realizar en *1d* si sólo se realiza la partición con respecto a uno de los ejes coordenados de un plano cartesiano de tres dimensiones; *2d* si la partición es realizada en un par de ejes coordenados y *3d* si la división del dominio es con respecto a los tres ejes coordenados. La parte sombreada representa los datos relacionados a una sola tarea.

La implementación de programas empleando este esquema puede ser relativamente sencilla. Las hebras o *threads* son una implementación de este paradigma de programación paralela, las más populares son:

- *Hebras POSIX*, basado en bibliotecas. Es una interfaz de programación de aplicaciones; conocido comúnmente como *Pthreads* [32]; especificado por el estándar *IEEE POSIX 1003.1.* Implica paralelismo explícito.
- *OpenMP (Open Multi-Processing)* [33], basado en directivas al compilador. Es una interfaz de programación de aplicaciones, portátil y multiplataforma. Su empleo puede ser relativamente fácil.

Ambas implementaciones tienen desventajas importantes, entre otras, que no son altamente escalables y que la localidad de los datos es difícil de entenderse y programarse.

3.4.1.4. HÍBRIDOS.

Este esquema es utilizado en computadoras que tienen tanto memoria compartida como memoria distribuida en una sola plataforma; es decir, nodos con varios procesadores que comparten la memoria, e interconectados con otros nodos similares. Este modelo no ha sido muy

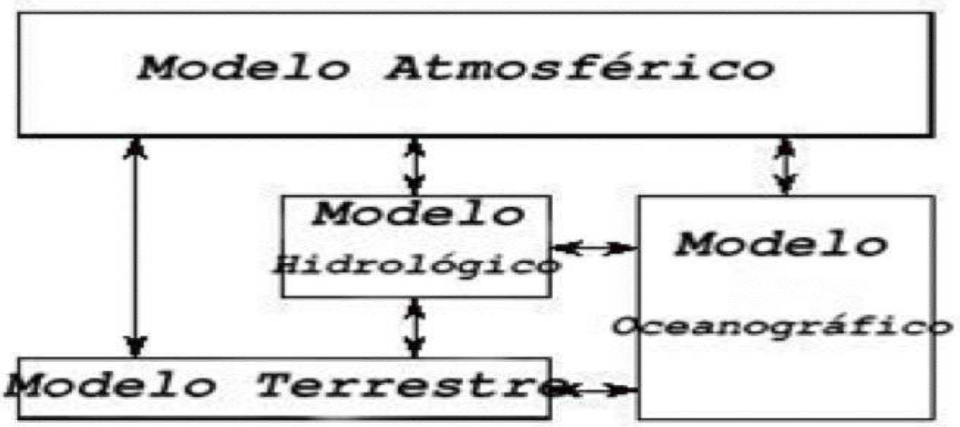


Figura 3.4: Representación de un modelo terrestre climático. Cada modelo o tarea puede ser representado como una tarea independiente: modelo atmosférico, modelo terrestre, modelo hidrológico y modelo oceanográfico. Los intercambios de datos entre estas tareas o modelos están representados por flechas

utilizado hasta ahora –ya que implica un esfuerzo notable su implementación y el rendimiento alcanzado es comparable a los modelos antes descritos, pero esto podría cambiar porque con la producción de computadoras masivamente paralelas que utilizan la tecnología multinúcleo los retos, dificultades y rendimientos se modificaran drásticamente. Ejemplos de este tipo puede ser utilizar *MPI* en combinación de ambos tipos de hebras: *MPI + OpenMP* o *MPI + Pthreads*.

3.4.1.5. ORIENTADO A OBJETOS Y SERVICIOS.

Se emplean los lenguajes orientados a objetos, tales como *C++, Java, Eiffel, Ada, Smalltalk*, entre otros, y su propósito es hacer cómputo distribuido en áreas muy amplias y están enfocados básicamente en áreas de servicios, tales como, los servicios de la *web*. Ejemplos de estos modelos son: *CORBA* [34, 35] y *DCOM* [36].

3.4.2. PARADIGMAS DE PARALELIZACIÓN.

Se pueden citar varios ejemplos de implementar un programa paralelo [37], los cuales están en función de la naturaleza del problema. Sin embargo, en algunos de estos problemas es viable aplicar más de uno, y la elección dependerá, entonces, tanto de la arquitectura del equipo de cómputo disponible así como del lenguaje de programación elegido o *a priori* establecido.

3.4.2.1. *MAESTRO-ESCLAVO.*

En este ejemplo, un proceso principal o *maestro* genera y reparte la carga de trabajo en M procesos *esclavos*. Cada uno de ellos, recibe trabajo a realizar, ejecuta el cómputo y regresa los

resultados al proceso *maestro*. Por lo tanto, el proceso *maestro* es el encargado de repartir la carga de trabajo, recibir y analizar los resultados suministrados por los procesos *esclavos*, pero también podría realizar cómputo; es decir, tendría doble función ser *maestro* y *esclavo* a la vez.

Este paradigma es adecuado, generalmente, para arquitecturas tanto de memoria compartida como memoria distribuida (sección 3.1.4). Sin embargo, el principal problema de este paradigma radica en que el proceso *maestro* puede convertirse en un "cuello de botella". Lo cual podría ocurrir si la carga de trabajo asignada a cada *esclavo* es muy pequeña generando muchas comunicaciones con el *maestro* o bien si la cantidad de datos que deba procesar y escribir/leer es demasiada. Es decir, se debe buscar un equilibrio entre la cantidad de cómputo realizado tanto por el *maestro* como por los *esclavos* y la cantidad de comunicaciones que dicha carga de trabajo genera. El tamaño de dicha carga de trabajo; es decir, la cantidad de cómputo asignada a *maestro* y *esclavos* es conocida como granularidad.

La interacción entre el *maestro* y los *esclavos* es de forma asíncrona, concepto de comunicación explicado más adelante.

3.4.2.2. SPMD (UN PROGRAMA, MÚLTIPLES DATOS).

Como ya se describió, este modelo es el más ampliamente utilizado. Cada proceso ejecuta el mismo programa en distintos datos, los cuales fueron repartidos entre todos los procesos (sección 3.4.1.1 y Fig. 3.3).

3.4.2.3. SEGMENTACIÓN DE DATOS.

Este ejemplo es el adecuado para aplicaciones que involucran múltiples etapas de ejecución. También es conocido como descomposición funcional o paralelismo de funciones (sección 3.4.1.2 y Fig. 3.4).

3.5. MÉTRICAS DE RENDIMIENTO

La discusión acerca de las métricas de rendimiento es bastante útil, porque ellas nos permiten de una manera simple entender las características del poder de cómputo de las supercomputadoras, las cuales son las computadoras con la máxima capacidad de procesamiento, memoria, almacenamiento y velocidad de comunicación que existen actualmente. El rendimiento de las supercomputadoras es el atributo principal que hace a éstas superiores sobre otros equipos. El rendimiento depende en gran parte de la naturaleza del problema que se esté resolviendo. Por rendimiento de una computadora se entiende la efectividad del desempeño de ésta, cualquiera que sea, en una aplicación o *benchmark* de referencia específico. Esta noción de rendimiento incluye: velocidad, eficiencia y costo de comunicaciones, entre otras.

3.5.1. FACTOR DE ACELERACIÓN (*SPEED-UP*).

El denominado factor de aceleración o (*speed-up*) *Sp* es una de las principales métricas de rendimiento cuando se trata de evaluar el desempeño de un programa paralelo. El factor de aceleración es la relación entre los tiempos serial y paralelo de ejecución del programa; es

decir, está definido como el cociente del tiempo que se tarda en completar el cómputo de la tarea usando un solo entre el tiempo requerido para hacerlo en M procesadores trabajando en paralelo o cuanto más rápido se ejecutará el programa en M procesadores que en un solo procesador. Esta métrica puede ser evaluada de dos maneras: la primera es fijar el tamaño del problema resuelto y variar la cantidad de procesadores –problema de tamaño fijo; la segunda es que la cantidad de procesadores empleados para resolver el problema se incremente a medida que se aumenta el tamaño del problema –problema de tamaño variable.
Para un problema de tamaño fijo el factor de aceleración se define como

$$Sp \equiv \frac{T_1(N)}{T_M(N)} \leq p \tag{3.1}$$

; donde T_1 es el tiempo de la ejecución serial, T_M es el tiempo de ejecución en paralelo en M procesadores en un problema de tamaño fijo N y p es el factor de aceleración ideal [38].
Para un problema de tamaño variable el factor de aceleración está definido mediante

$$Sp \equiv \frac{MT_1(N/M)}{T_M(N)} \tag{3.2}$$

Idealmente, en un programa perfectamente paralelo –con un balanceo de carga excelso y sin sobrecarga debido a la paralelización– el factor de aceleración es una función lineal de M:

$$Sp \equiv \frac{T_1(N)}{T_M(N)} = \frac{T_1(N)}{T_1(N)/M} = M \tag{3.3}$$

El factor de aceleración está limitado por la ley de *Amdahl* [38], cuando el programa no es total o perfectamente paralelo. La ley de *Amdahl* expone que un programa está compuesto por una parte paralela T_M , que puede ser repartida entre los M procesadores involucrados y una parte serial T_1 , cuya carga de trabajo es independiente de los procesadores de la parte paralela; por lo tanto, el factor de aceleración lo podemos definir como:

$$Sp \equiv \frac{T_1(N)}{T_M(N)} = \frac{T_1}{1-\alpha+\frac{\alpha}{M}} = \frac{T_1}{S+\frac{\alpha}{M}} \leq p \tag{3.4}$$

; donde α es igual a la fracción paralela; S es la fracción serial; M es el número de procesadores, en un problema de tamaño N y p es igual al factor de aceleración ideal, según *Amdahl*. Por lo tanto, el factor de aceleración está en función de la fracción serial del programa; es decir, $S_P = \frac{T_1}{1-\alpha} = \frac{T_1}{S}$ cuando $M \to \infty$, lo cual podemos observar en los siguientes ejemplos representados en la Tabla 3.1:

- Con 2 procesadores y el 98 % del código paralelo, el *factor de aceleración* es cercano a 2.
- Con 8 procesadores, el factor de aceleración es 7.

Desafortunadamente, la ley de *Amdahl* a pesar de su sencillez y utilidad no es suficiente, puesto que no involucra todos los factores de un programa paralelo, en particular, el referente a las comunicaciones y sincronización de los M procesos. Para involucrar las comunicaciones y el cómputo podemos estimar estos términos mediante la ecuación (3.5)

Tabla 3.1: Se muestra el factor de aceleración en dos ejemplos utilizando dos y ochos procesadores y distintos porcentajes de código paralelo. Mientras mayor es el porcentaje del código paralelo el factor de aceleración es más cercano al número de procesadores empleados.

Dos procesadores		Ocho procesadores	
% Paralelo	Factor de aceleración	% Paralelo	Factor de aceleración
80	1.6	80	3.3
90	1.8	90	4.7
95	1.9	95	5.9
98	1.96	98	7.0

$$T\left(N,M\right)=\tau_{comp}\left(N,M\right)+\tau_{comm}\left(N,M\right) \tag{3.5}$$

;donde τ_{comp} es el costo de computación y τ_{comm} representa el costo de comunicación, los cuales pueden calcularse a partir del tamaño del problema N, y de la cantidad de procesadores utilizados M. Podemos reformular la ecuación de la ley *Amdahl* para involucrar estos costos, quedando de la siguiente forma para problemas de tamaño fijo y variable, ecuaciones (3.1) y (3.2) respectivamente:

$$Sp\equiv\frac{T_1(N)}{T_M(N)}=\frac{T_1}{\tau_{comp}\left(N,M\right)+\tau_{comm}\left(N,M\right)} \tag{3.6}$$

$$Sp\equiv\frac{T_1(N)}{T_M(N)}=\frac{T_1}{\tau_{comp}\left(N,M\right)+\tau_{comm}\left(N,M\right)}M \tag{3.7}$$

3.5.2. FACTOR DE EFICIENCIA.

La eficiencia E es la métrica que nos permite determinar la forma en que los recursos han sido utilizados y está definida como el cociente del tiempo que se tarda en completar el cómputo de la tarea empleando un solo procesador en un problema de tamaño N entre el número de procesadores multiplicado por el tiempo de ejecución necesario para ejecutarlo en M procesadores en el mismo problema de tamaño N fijo

$$E=\tfrac{T_1(N)}{MT_M(N)},\quad 0\leq E\leq 1 \tag{3.8}$$

En el caso de un problema de tamaño variable la eficiencia se define como

$$E\equiv\frac{T_1\left(N/M\right)}{T_M\left(N\right)} \tag{3.9}$$

Idealmente, el valor de la eficiencia es igual a 1 cuando todo el tiempo es empleado en operaciones de cómputo; es decir, no existiese comunicación ni sincronización alguna entre los M procesadores involucrados en la solución de un problema.

3.6. METODOLOGÍA.

A continuación se presenta la metodología aplicada en este trabajo para realizar la paralelización del algoritmo de **DFA3D**, sintetizado en el capítulo 2. Primero se describen las denominadas precondiciones de *Pancake* [39], seguido del método propuesto por *Foster* [40] para transformar el código de su versión serial a paralela.

3.6.1. PRECONDICIONES DE *PANCAKE*.

Para realizar la paralelización de un programa a partir de su versión serial, *Pancake* [39] propone verificar, entre otras recomendaciones, tres precondiciones, las cuales deberían validarse de forma positiva, éstas son: frecuencia de uso, tiempo de ejecución y resolución.

La frecuencia de uso se refiere a la cantidad de veces que se ejecuta el programa antes de realizar un cambio al mismo y recompilar; es decir, si en la solución de un problema el programa requiere ser modificado en varias ocasiones entre algunas ejecuciones, no es recomendable paralelizar el programa y la precondición es negativa. En cambio, si el número de veces que se corre el programa es muy alto, comparado con la cantidad de veces que se realizan los cambios, se puede decir que el programa es susceptible de ser paralelizado; por lo tanto, se valida como positiva esta precondición. Si la frecuencia de uso es intermedia entre los dos casos antes citados se puede decir que la precondición es posible. Dicho de otra manera, el programa no debe estar en su fase de desarrollo, sino en el de producción.

Si el tiempo de ejecución de un programa es de sólo algunos minutos no es necesario, ni vale la pena el esfuerzo de realizar la paralelización o intentar hacerlo, que implica que esta precondición sea negativa. Dicha precondición es posible, cuando el tiempo de ejecución involucra algunas horas. Si el programa tarda en ejecutarse varios días bien vale la pena intentar una paralelización, disminuyendo el tiempo de espera en la obtención de resultados, haciendo que la precondición se valide como positiva. En aplicaciones de tiempo real esta precondición resulta subjetiva, puesto que se espera que el programa responda prácticamente de forma instantánea.

Por último, si la resolución o tamaño actual del dominio del problema satisface los requerimientos de éste, la precondición es negativa y no debería intentarse paralelizar el programa. Esta precondición adquiere un valor posible si la resolución fuese deseable aumentarla, pero si es imperativo incrementar la resolución o el tamaño del dominio del problema porque ya no satisface la complejidad o las necesidades del problema se le asigna a la precondición un valor positivo. Esquemáticamente pueden representarse dichas precondiciones como se indica en la Fig. 3.5.

Según *Pancake*, si, al menos, una de las citadas precondiciones es negativa no deberíamos llevar a cabo una paralelización del programa (aunque, como ya se observo, si la aplicación es de tiempo real, el tiempo de ejecución es una condición no suficiente).

Por otro lado, si las tres precondiciones son positivas, o el programa es utilizado en múltiples ocasiones antes de modificarlo; su tiempo de ejecución es de varios días y la resolución es insuficiente, entonces se puede intentar una paralelización. Sin embargo, la validación positiva de las tres precondiciones no garantiza el éxito en la implementación paralela del programa o algoritmo ni tampoco el fracaso si algunas o todas las precondiciones son posibles o negativas.

Sin ser una receta, la verificación de estos requisitos orienta y proporciona una pauta en

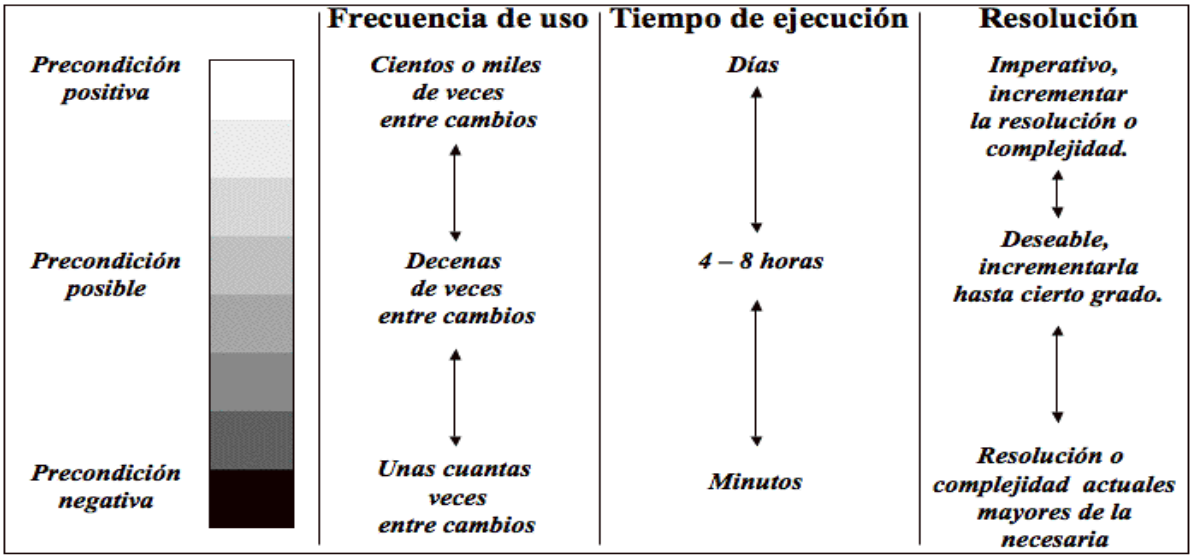


Figura 3.5: Esquema que representa la validación: positiva, posible o negativa de las tres precondiciones, propuestas por *Pancake* [39]: frecuencia de uso, tiempo de ejecución y resolución. Si éstas son positivas; es decir, se ejecuta cientos de veces antes de recompilar; el tiempo total de ejecución del proceso es de varios días y la resolución actual es insuficiente para modelar el problema, entonces es recomendable –y vale la pena el tiempo y esfuerzo- realizar la paralelización.

la paralelización de un programa, de tal manera que no se debería llevar a cabo ésta si los resultados, los tiempos de ejecución y la resolución son suficientes o satisfactorios, a menos que, sea con fines didácticos.

3.6.2. MÉTODO DE FOSTER.

El método empleado en la transformación del código de interés a su versión paralela es el propuesto por *Foster* [40], el cual está integrado por cuatro etapas, representado en la Fig. 3.6

1. Partición : Dividir el problema en pequeñas partes o tareas; es decir, distribuir los datos de una gran carga computacional entre varios procesadores. Cada parte tendrá su propio conjunto de datos y de operaciones que se realizarán sobre los primeros. Si la división del problema es muy grande –cuantitativamente- se le conoce como "granularidad fina", lo cual implicará muchas comunicaciones entre las tareas, aunque este tipo de descomposición de grano fino proporciona la mejor flexibilidad; por el contrario, si la división es pequeña

"granularidad gruesa" no existirán muchas comunicaciones entra las tareas, pero éstas serían de gran tamaño, y probablemente los recursos de un nodo o procesador no sean suficientes para su ejecución. Por lo tanto, se deberá buscar un equilibrio entre la cantidad de particiones o tareas y la cantidad de recursos de *HW*, lo cual se realizará en las siguientes etapas.

2. Comunicación : Las comunicaciones indispensables para la coordinación de las tareas en la ejecución del programa son determinadas en esta fase, así como también las estructuras de comunicación adecuadas (Fig. 3.6). Éstas pueden ser de varios tipos:

   - Local, entre los vecinos de la tarea y que son ejecutados en un mismo procesador; es decir, no requiere comunicaciones vía red.
   - Global, en la cual todos las tareas están involucradas y las comunicaciones son realizadas mediante la red de interconexión, lo cual impacta el rendimiento del programa.

   - Estructurada, el dominio del problema es dividido en forma regular, de tal manera que cada tarea resuelve una proporción similar del problema.
   - No estructurada, cuando el dominio del problema es irregular, por lo cual cada tarea resolverá diferentes tamaños del dominio del problema.

   - Estática, ya que desde el inicio hasta la conclusión de la ejecución del problema existe la misma cantidad de tareas; es decir, la cardinalidad de las tareas es conocido de antemano y fijo.
   - Dinámica, en este esquema las tareas son generadas y eliminadas durante la ejecución del proceso, y las comunicaciones se incrementan o disminuyen conforme se generan o destruyen tareas, respectivamente.

   - Síncrona, ya que las tareas se ejecutan en un tiempo determinado y coordinadamente se comunican los datos.
   - Asíncrona, las comunicaciones entre las tareas son realizadas en cualquier momento; es decir, no existe correspondencia temporal y; por lo tanto, no requiere una sincronización entre las tareas.

3. Aglomeración : Las pequeñas tareas, si fuese necesario, son agrupadas en tareas más grandes para incrementar el rendimiento del programa (Fig. 3.6). Al llevar a cabo la aglomeración de las tareas se tiene por objetivo reducir las comunicaciones entre las tareas, lo cual implica que el cómputo se incrementa porque el procesador o el sistema no están ejecutando múltiples comunicaciones, de tal manera que el desempeño del programa aumenta, aproximándose al ideal.

4. Mapeo : Cada tarea es asignada a sólo un procesador de la computadora de tal manera que se optimice el uso de los procesadores minimizando las comunicaciones involucradas (Fig. 3.6). Los encargados de realizar esta asignación son el sistema operativo, elementos

de *HW*, calendarizadores o sistemas de atención de colas. Como se cito al inicio de este capítulo, uno de los dos objetivos primordiales de la paralelización es reducir el tiempo de ejecución de un proceso y el mapeo mediante un par de estrategias permite alcanzar este propósito; dichas estrategias son:

- Incrementar la concurrencia –término empleado en la descripción del número de acciones o tareas que pueden ser realizadas al mismo tiempo o de forma simultánea– al ubicar las tareas que son capaces de ejecutarse al mismo tiempo en procesadores diferentes.
- Aumentar la localidad –que se refiere a una alta proporción de acceso a los datos a la memoria local con respecto a los datos que son accesados en memoria remota y que son suministrados mediante comunicaciones. El concepto de localidad es primordial en la obtención de alto rendimiento en arquitecturas tipo *MISD* y *MIMD* de multiprocesadores o multinúcleo– cuando son asignados al mismo procesador las tareas que se comunican constantemente.

En el mapeo de tareas a procesadores se deberá buscar tener un equilibrio entre los recursos requeridos para una distribución de carga equitativa de las tareas del problema y bajos costos de comunicación; dado que ambas estrategias, podrían en algunos casos ser excluyentes u originar conflictos. El problema del mapeo o asignación es un problema *NP-completo*; es decir, que no existe un algoritmo tratable que en tiempo polinómico pueda evaluar estos equilibrios de manera general [41].

En síntesis, las etapas de partición y comunicación del método propuesto por *Foster* [40] (Fig. 3.6) para la transformación del código serial a paralelo, se enfocan en la concurrencia y la escalabilidad –que es la capacidad de un algoritmo de aprovechar el aumento en la cantidad de procesadores; es decir, que el factor de aceleración *Sp* se incremente de forma proporcional al número de procesadores empleados- de los procesos, mientras que en las etapas de aglomeración y mapeo el enfoque está orientado a la localidad de los procesos y sus datos.

Es conveniente mencionar que el factor de aceleración *Sp*;,la eficiencia *E*, la concurrencia; la escalabilidad y la localidad dependen del *HW* y *SW* disponibles para la ejecución en paralelo.

3.7. DISEÑO DEL PROGRAMA PARALELO.

En esta sección se describe el algoritmo serial de diferencias finitas alternadas 3D **DFA3D** (capítulo 2), su diseño en paralelo, de acuerdo al método propuesto por *Foster* [40] descrito en 3.4.2. y su implementación utilizando *MPI*, así como una breve explicación de esta biblioteca.

3.7.1. DESCRIPCIÓN DEL ALGORITMO SERIAL.

En la Fig. 3.7 se muestra el diagrama de flujo en bloques del algoritmo de **DFA3D** para la obtención de sismogramas, en su versión serial.

El método de **DFA3D** utilizado para modelar el problema en este trabajo resuelve un conjunto de ecuaciones diferenciales parciales (capítulo 2), que describen la propagación de ondas

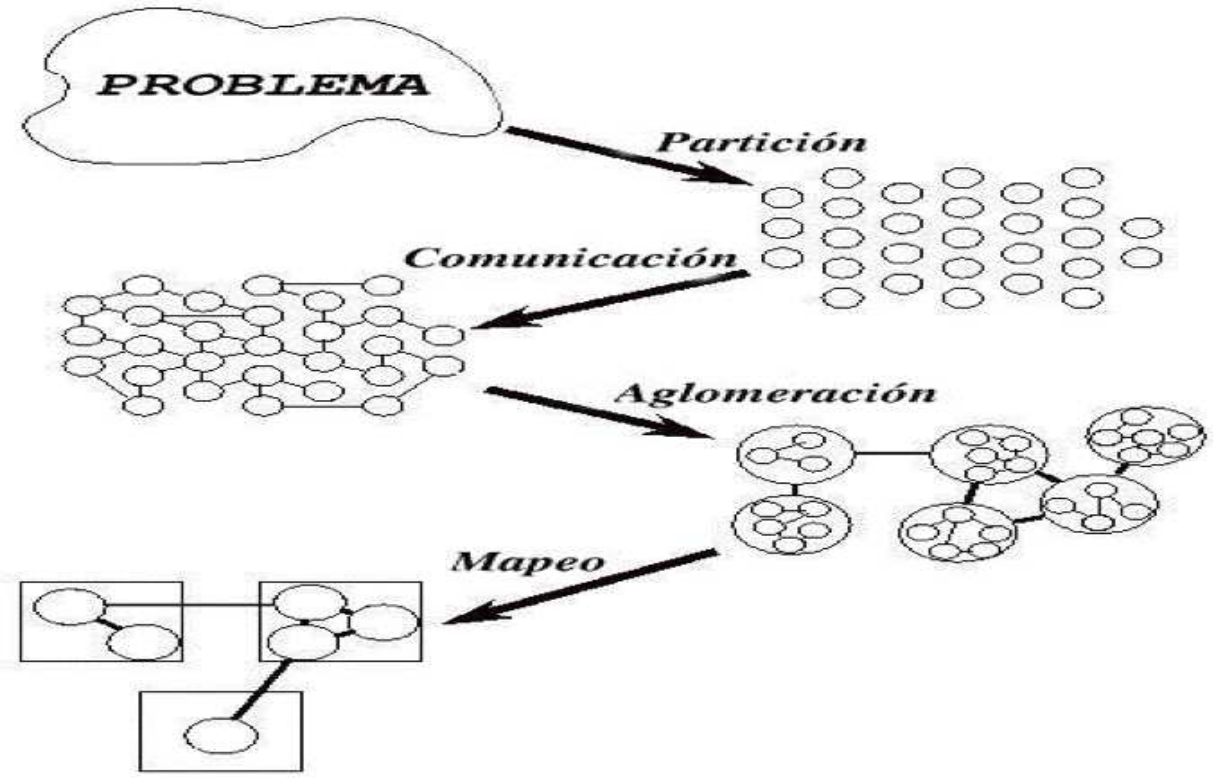


Figura 3.6: A partir de un problema dado, se llevan a cabo las siguientes tareas de forma secuencial: la partición del dominio del problema en tantas tareas como se requiera o sea posible; después se establecen las comunicaciones entre las tareas; posteriormente se lleva a cabo la aglomeración de las tareas para reducir comunicaciones e incrementar el cómputo y, por último, se realiza el mapeo que consiste en ubicar las tareas en los procesadores.

sísmicas asociadas a una descripción cinemática de la ruptura de la fuente sísmica, expresado en términos de sus velocidades y esfuerzos en el dominio del tiempo y espacio.

La descripción, en bloques Fig. 3.7, del programa serial se describe a continuación:

1. Inicio.
2. Declaración de variables y parámetros de configuración del modelo.
3. Inicialización de las variables y parámetros.
4. Reserva memoria dinámicamente para las tres variables de velocidad V_x, V_y, V_z; de densidad ρ; parámetros de Lamé μ, λ y seis variables de esfuerzos $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz}$.
5. Lee la estructura del modelo.
6. Crea los archivos de salida (sismogramas).

7. Abre archivo de la fuente (zona de ruptura) y reserva memoria dinámica para almacenar los datos de la fuente.

   - Ciclo para la simulación de los pasos de tiempo ntst $= 2,$ nt.
     a) Calcula las velocidades V_x, V_y, V_z con **DFA3D** de cuarto orden en el espacio [*DVEL()*].
     b) Calcula velocidades con condiciones de absorción en las fronteras [*ABC()*].
     c) Calcula las velocidades con **DFA3D** de segundo orden en las fronteras [*BND2D()*].
     d) Calcula las velocidades con **DFA3D** en superficies libres de fronteras [*FUVW()*].
     - Verifica si $ntst \leq idur$ (duración en pasos de tiempo de la ruptura).
       - Verdadero.
         1) Lee los datos de la fuente (zona de ruptura).
         2) Adiciona fuente gaussiana *wavelet* [*ADDB3D()*].
       - Falso.
         1) Cierra archivo de la fuente.

     e) Calcula los esfuerzos $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz}$ con **DFA3D** de cuarto orden en el espacio [*DSTRES()*].
     f) Calcula los esfuerzos con **DFA3D** de segundo orden en las fronteras [*STRBND1()*].
     g) Calcula los esfuerzos con **DFA3D** en superficies libres de fronteras [*FRES()*].
     h) Escribe archivo de salida (sismogramas).

8. Libera memoria dinámica.

9. Termina.

En la Fig. 3.8 se ilustra el esquema general del método numérico de diferencias finitas alternadas utilizado en este trabajo. Este método es computacionalmente intensivo, ya que para calcular los valores de las velocidades y esfuerzos para cada punto del medio discretizado se requieren trece valores de estas variables en nodos vecinos, y como se indicó en el capítulo 2, la solución de las ecuaciones (2.23) y (2.24) en un medio continuo, Fig. 3.9a) es aproximado por su solución en un conjunto finito de puntos con un espaciamiento regular en dicho dominio. Estos puntos son ubicados en un paralelepípedo o malla rectangular en discretizaciones en una, dos y tres dimensiones sobre los ejes Z; Z, Y y Z, Y, X, respectivamente Figs. 3.9b), c), d).

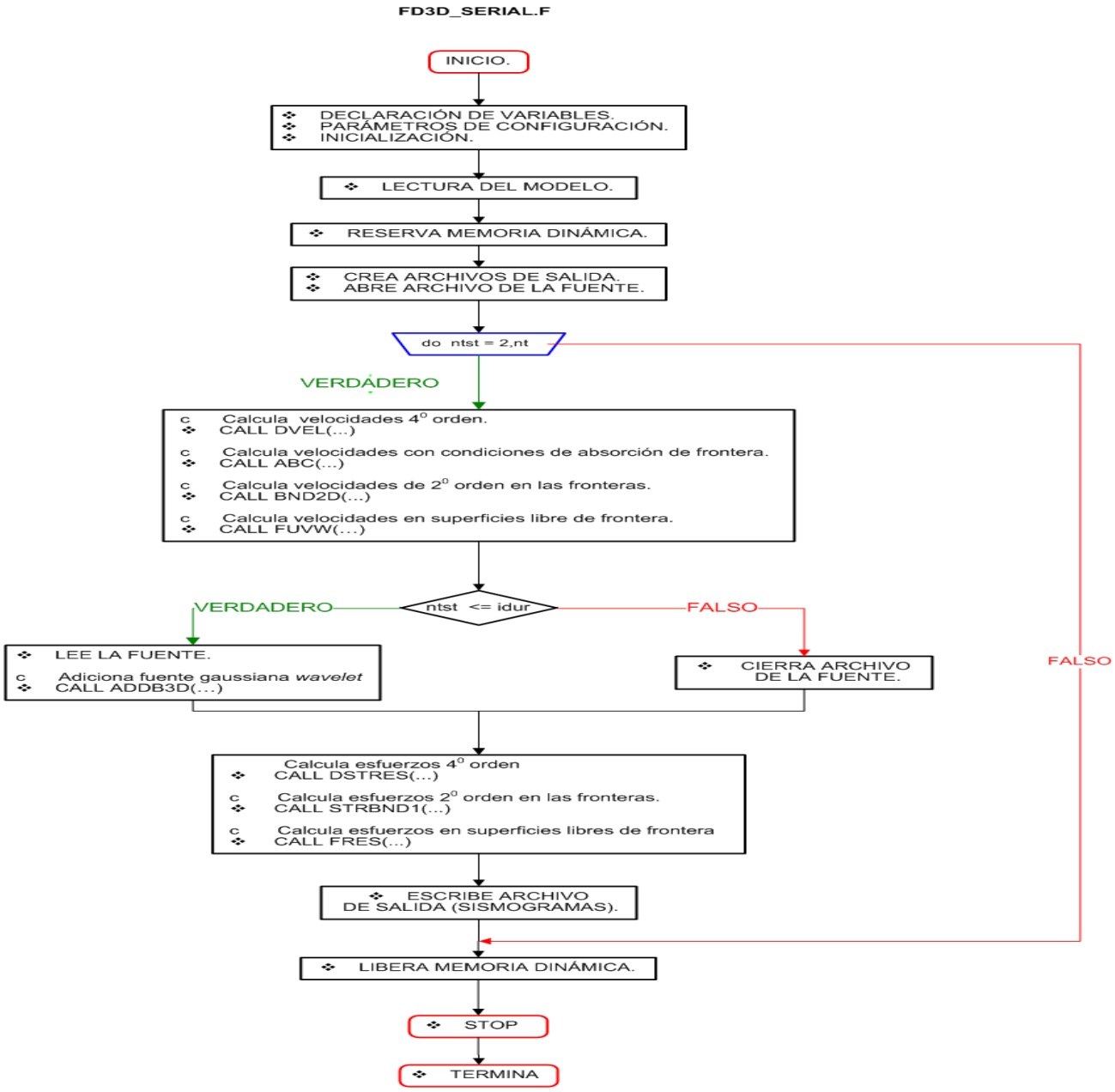


Figura 3.7: Diagrama de flujo por bloques de la versión serial del programa del algoritmo de **DFA3D**.

3.7.2. PARALELIZACIÓN.

De acuerdo al método propuesto por *Foster* en 3.6.2. [40] se describirá la paralelización del programa serial y posteriormente su implementación utilizando *MPI* [30, 31].

3.7.2.1. PARTICIÓN.

Se utilizó el paralelismo de datos o descomposición del dominio porque es escalable [42]. Los mejores programas paralelos son aquellos en los cuales se trata que cada procesador realice prácticamente la misma cantidad de cómputo minimizando las comunicaciones entre los procesos.

Este equilibrio, ya descrito, entre el balanceo de la carga y los costos de comunicación es conseguido utilizando descomposición del dominio regular aplicado al esquema explícito de diferencias finitas, como es el caso del programa serial. Al emplear este paradigma de paralelización el dominio es dividido en pequeñas tareas (subdominios) –de tamaños idénticos- y distribuidos entre todos los procesadores; por lo tanto, cada procesador resuelve su propio subdominio del problema.

En este trabajo se realizaron descomposiciones del dominio del problema en una, dos y tres dimensiones. El dominio tridimensional del problema, Fig. 3.9a), así como las descomposiciones del dominio en una dimensión *1d* z, Fig. 3.9b), dos dimensiones *2d* Y, Z, Fig. 3.9c) y tres dimensiones *3d* X, Y, Z, Fig. 3.9d) están ilustradas en las mallas de las correspondientes figuras. Si se sigue la táctica de obtener la máxima concurrencia posible, entonces, se realiza la descomposición más agresiva viable -en tres dimensiones *3d* X, Y, Z, Fig. 3.9c); por lo tanto, se tendría una tarea para cada punto de la malla, "granulado muy fino", que implica que cada punto del dominio del problema fuese resuelto de forma independiente por un solo procesador. De manera tal que se tendrían $N_x \times N_y \times N_z$ tareas, cada una con una complejidad $\mathrm{O}(1)$ [41] de cómputo y de datos para cada iteración en el tiempo.

3.7.2.2. COMUNICACIÓN.

A partir del esquema general del algoritmo de diferencias finitas, Fig. 3.8, que requiere trece puntos para la actualización de un valor cualquiera, el algoritmo de **DFA3D** utilizado para la propagación de la onda elástica en la ecuación de la forma velocidad-esfuerzo del problema que se aborda en este trabajo, que es de segundo orden en el dominio del tiempo y de cuarto orden en el dominio del espacio, utiliza nueve variables, tres de ellas de velocidad V_x, V_y, V_z y seis de esfuerzos $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz}$ (capítulo 2), de las cuales se muestran la velocidad V_x y los esfuerzos σ_{xx} y σ_{xy} en las Figs. 2.4, 2.5 y 2.6, respectivamente, así como las variables involucradas y la posición de éstas para su actualización. Las otras dos variables de velocidad y las cuatro restantes de esfuerzos son similares a las ilustradas en las citadas figuras.

Las comunicaciones del problema están distribuidas lo que implica que puedan realizarse concurrentemente o de forma síncrona, que implica que en un determinado momento todos los procesos deben de realizar las comunicaciones para que puedan seguir adelante con su trabajo. De tal manera que para realizar la actualización de las variables de velocidades y de esfuerzos para cada punto de la máxima partición obtenida en la sección previa (3.7.2.1), donde cada tarea se encarga de resolver un solo punto del dominio discretizado del problema, es necesario que cada tarea lleve a cabo entre 8 y 9 comunicaciones con sus vecinos para la actualización de una sola variable; es decir, para actualizar todas y cada una de las nueve variables físicas del modelo se requerirán 75 comunicaciones con sus vecinos. Más las comunicaciones para la generación de sismogramas y "*snapshots*" (imagen del dominio completo de la simulación en un instante dado) en cada iteración.

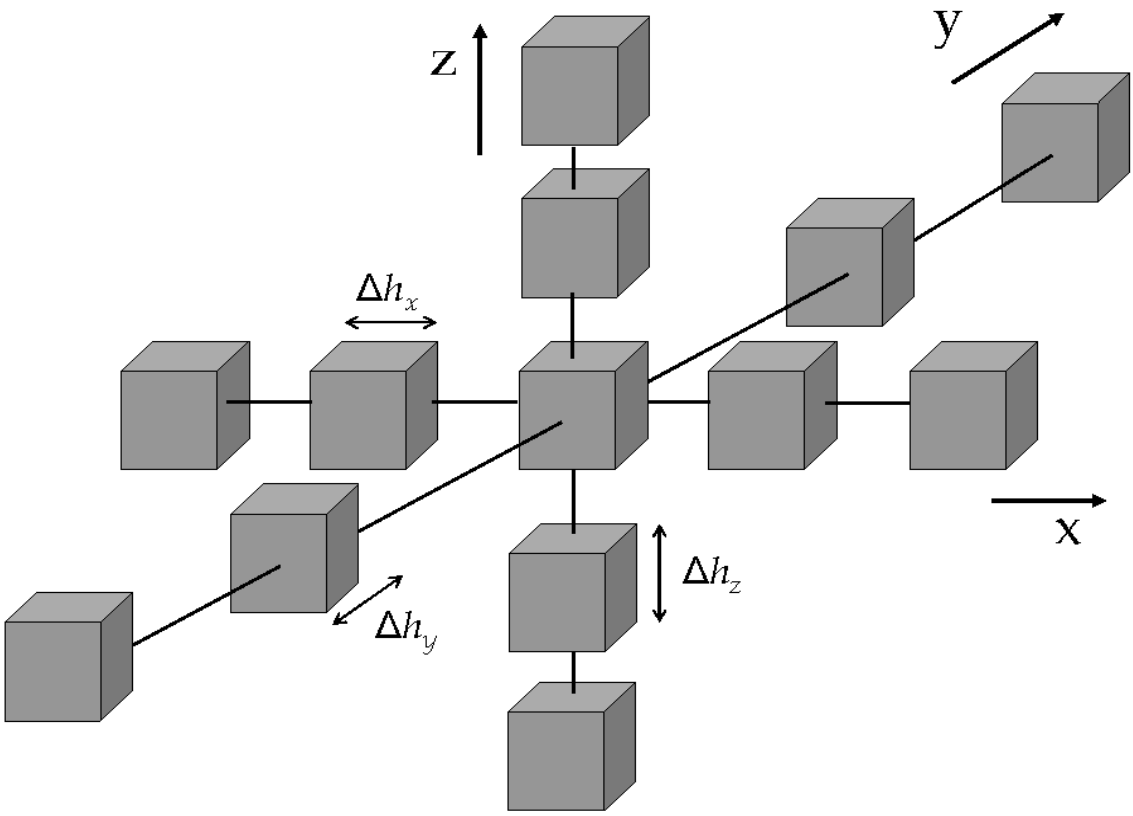


Figura 3.8: Esquema genérico del algoritmo de diferencias finitas 3D. Para un único punto del espacio tridimensional se muestran los trece puntos utilizados para la actualización tanto horizontal como vertical de las velocidades y esfuerzos; donde $\Delta x, \Delta y, \Delta z$ representan los espaciamientos espaciales en *x,y,z*, respectivamente.

3.7.2.3. AGLOMERACIÓN.

Al haber realizado las dos primeras etapas del método de paralelización propuesto por *Foster* [40], tenemos como resultado $N_x \times N_y \times N_z$ tareas, las cuales requieren 75 comunicaciones, sin tomar en cuenta las comunicaciones para la escritura de los sismogramas y "*snapshots*", con sus vecinos para llevar a cabo la actualización de las nueve variables físicas que describen el modelo. Dicha cantidad de tareas creadas, que depende del tamaño del problema, puede ser mucho mayor que cualquier cantidad de procesadores con que pueda contar un equipo de cómputo; por lo tanto, nuestra descomposición de grano fino obtenido en la sección 3.7.2.1 puede ser inviable por dos cuestiones: la gran cantidad de procesadores requeridos y de comunicaciones, de tal manera que es necesario llevar a cabo una aglomeración de las tareas para reducir dichas cantidades. La aglomeración utilizada está orientada, como ya se citó, a disminuir el número de procesadores y comunicaciones necesarios, al incrementar la localidad de los procesos y datos. Esta aglomeración se propone tanto en el plano horizontal como en el vertical de tamaño mínimo de $5 \times 5 \times 5$ puntos por tarea, que nos daría un total de 6 comunicaciones máximo, para los

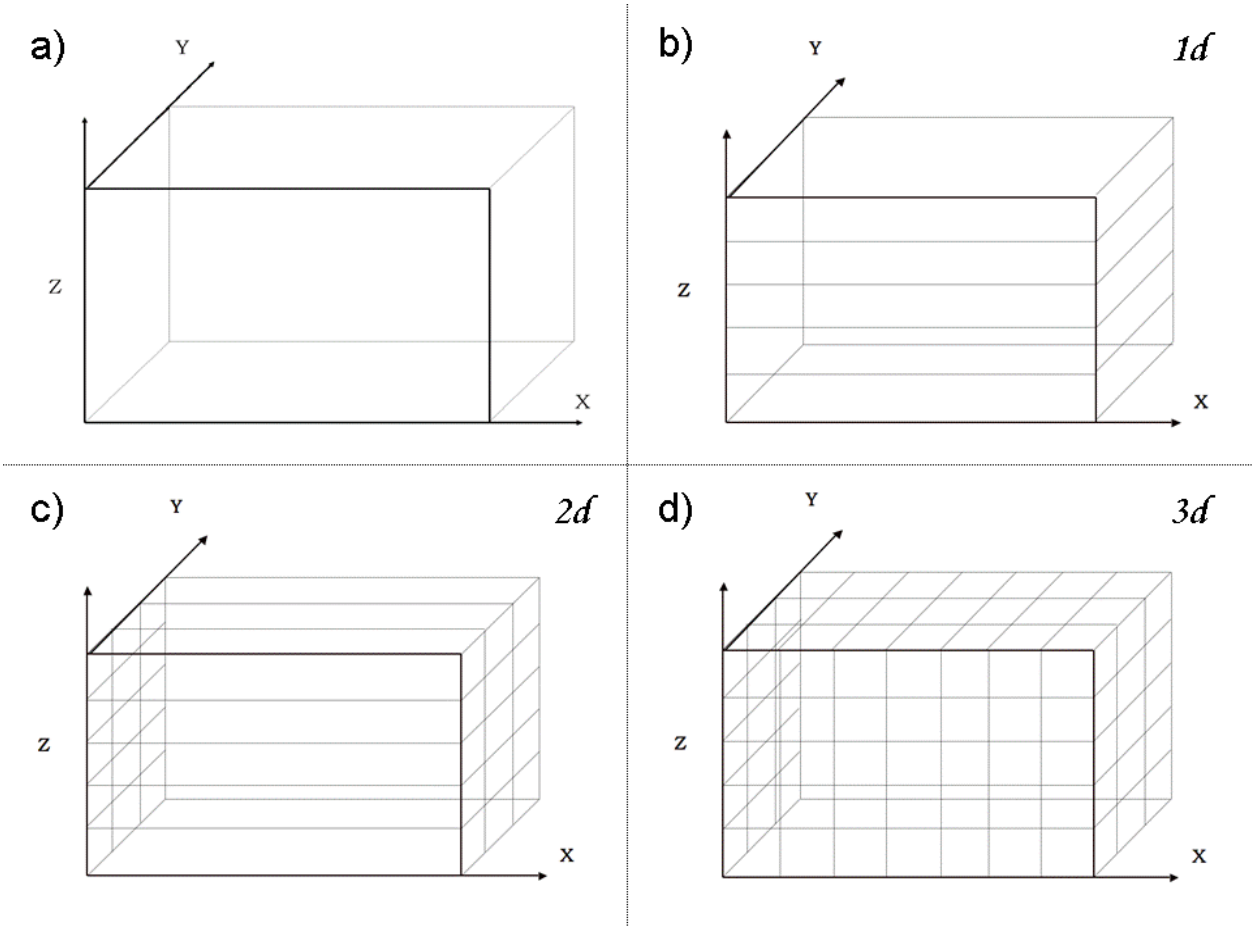


Figura 3.9: El dominio tridimensional del problema es mostrado en la subfigura a). Se describen las descomposiciones del dominio en *1d* (eje Z) $N_z = 5$ subfigura b); *2d* (ejes Y, Z) $N_y = 3, N_z = 5$ subfigura c) y en *3d* (ejes X, Y, Z) $N_x = 7, N_y = 3, N_z = 5$ subfigura d) .

puntos de la frontera de este subdominio, que contienen 12 datos.

3.7.2.4. MAPEO.

Dado que el dominio del problema se dividió de forma equitativa entre las tareas y las comunicaciones locales y globales son estructuradas y estáticas, el mapeo simple, cuya meta es minimizar las comunicaciones entre los procesadores, es útil y es el empleado en esta etapa. El cual consiste en permitir que el sistema operativo mediante calendarizadores o sistemas de atención de colas sea el encargado de asignar, automáticamente, un proceso a un determinado procesador; es decir, una relación biunívoca entre procesos y procesadores. La asignación de esta relación biunívoca puede ser en bloque o cíclico.

3.7.2.5. IMPLEMENTACIÓN USANDO *MPI*.

En esta sección se describirá sucintamente la biblioteca *MPI*, así como las transformaciones realizadas al programa serial para su paralelización.

3.7.2.6. *MPI* (INTERFAZ DE ENVÍO DE MENSAJES).

A principio de la década de 1990 aproximadamente 40 organizaciones, especialmente investigadores universitarios y de laboratorios gubernamentales y fabricantes de computadoras paralelas, de Europa y Estados Unidos integraron un comité para la creación de un estándar en el envío de mensajes. Dicho comité definió la sintaxis y semántica de un conjunto de instrucciones. Fruto de este foro es la versión *MPI 1.1*, que tomó las mejores características de sus predecesores, tales como: *PVM, PARMACS, Intel NX/2, Express* y *Zipcode* [43]. Se mejoró en 1997 generando *MPI 2*.

Podemos caracterizar a *MPI* y listar algunas de sus ventajas [44], que lo han convertido en el estándar *de facto*:

- Es un estándar, no una implementación específica. Un programa generado en una implementación particular de *MPI* debe funcionar en cualquier otra implementación sin modificación alguna.
- Es una biblioteca no un lenguaje de programación. En ella se especifica el nombre de la instrucción; la cantidad y el tipo de argumentos así como los valores y tipos de regreso de dichas instrucciones. Por lo tanto, se pueden escribir programas tanto en *Fortran, C y C++* en todas sus variantes.
- Es completamente portátil. El código de un programa *MPI* puede ser compilado y ejecutado en cualquier arquitectura sin requerir modificación alguna.
- Está formalmente especificado, por medio de un documento oficial que contiene todas las características que debe tener una implementación cualquiera.
- Es posible realizar comunicaciones totalmente asíncronas, permitiendo traslapar el cómputo y las comunicaciones.
- Realiza un adecuado manejo de los *buffers* (memoria intermedia) de los mensajes. Los datos son enviados y recibidos mediante estructuras de datos definidas por el usuario y no empleando estructuras definidas internamente por las bibliotecas de comunicación.
- Los grupos de procesos son estables y eficientes, los cuales son creados por medio de operaciones colectivas y la información de los integrantes del grupo es accesible a todos ellos.
- Permite una programación eficiente de alto rendimiento en todas las arquitecturas especificadas en la sección 3.3 de este capítulo.

- Permite realizar las siguientes instrucciones: comunicaciones punto a punto; operaciones colectivas; grupos de procesos; contextos de comunicación; formación de topologías; escritura y lectura en paralelo; creación dinámica de procesos, entre otras [45, 46, 47].

Existen implementaciones gratuitas disponibles. Las más populares son *MPICH* (del Laboratorio Nacional de *Argonne*) [48, 49] *LAM* (del Centro de Supercómputo de *Ohio*) [50] y *OpenMPI* [51].

La versión de *MPI 1.1* consta de 125 funciones mmientras que la versión *MPI 2* más de 200. Sin embargo, es posible realizar un programa paralelo empleando sólo 6 funciones, de las cuales las más importantes son las que involucran el intercambio de mensajes. Para llevar a cabo la operación de intercambio de datos son necesarias 2 funciones: envío y recepción de datos. Las cuales deben ser recíprocas, no necesariamente concurrentes y tener una relación biunívoca; es decir, para cada instrucción de envío debe haber una instrucción de recepción y viceversa. Dos más son imprescindibles para inicializar y finalizar el ámbito de *MPI* y las últimas dos funciones, de las 6 en cuestión, se utilizan para saber la cantidad de procesos creados y la identificación de cada uno de éstos.

3.7.2.7. IMPLEMENTACIÓN DE LA PARALELIZACIÓN DEL PROGRAMA SERIAL.

Antes de aplicar el diseño del programa paralelo descrito en las secciones 3.7.2.1 – 3.7.2.4 fue imperativo realizar la depuración de la versión serial del programa; es decir, identificar y eliminar las variables no utilizadas, así como identificar el tipo de las mismas, en particular, reales o enteras e inicializar todas las variables a cero –es primordial identificar cuáles son las variables que corresponden a las tres velocidades: V_x, V_y, V_z, así como también a las seis de esfuerzos $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz}$; realizar el reagrupamiento de las instrucciones; intercambiar las variables globales por locales; eliminar los bloques comunes mediante envío de parámetros de forma explícita por medio de rutinas o funciones e identificar los bloques susceptibles de paralelización –mediante un perfilado del programa serial; es decir, un muestreo estadístico del tiempo que cada función, rutina o sección del programa demora en ejecutarse, ilustrado en la Fig. 3.10. Las funciones que consumen el mayor porcentaje en tiempo de procesador durante la ejecución del programa serial son: *dstres, dvel, strbnd1, bnd2d y abc* y las candidatas a ser optimizadas. Dichas funciones actualizan las seis variables de esfuerzos y las tres variables de velocidad del algoritmo de **DFA3D** utilizado para la propagación de la onda elástica en la ecuación de la forma velocidad-esfuerzo del problema de interés abordado en este trabajo.

En el algoritmo serial de este trabajo las tres precondiciones, propuestas por *Pancake* [39] son validadas como positivas, ya que no se realizan cambios entre múltiples ejecuciones; el tiempo de ejecución del programa, dependiendo de la arquitectura, es de varias horas, y la resolución es insuficiente porque la discretización es muy gruesa (los datos son reportados en el capítulo 4). La versión serial de esta tesis está implementada en *Fortran77*, al realizar la paralelización se mantuvo el lenguaje de programación para simplificar su desarrollo, pero actualizándolo a *Fortran90*, porque este lenguaje sigue en desarrollo (con versiones recientes como la 90, 95 y 2003) y el rendimiento obtenido por otros lenguajes de programación de alto nivel, aún no es, del todo, comparable al obtenido con el citado lenguaje [52].

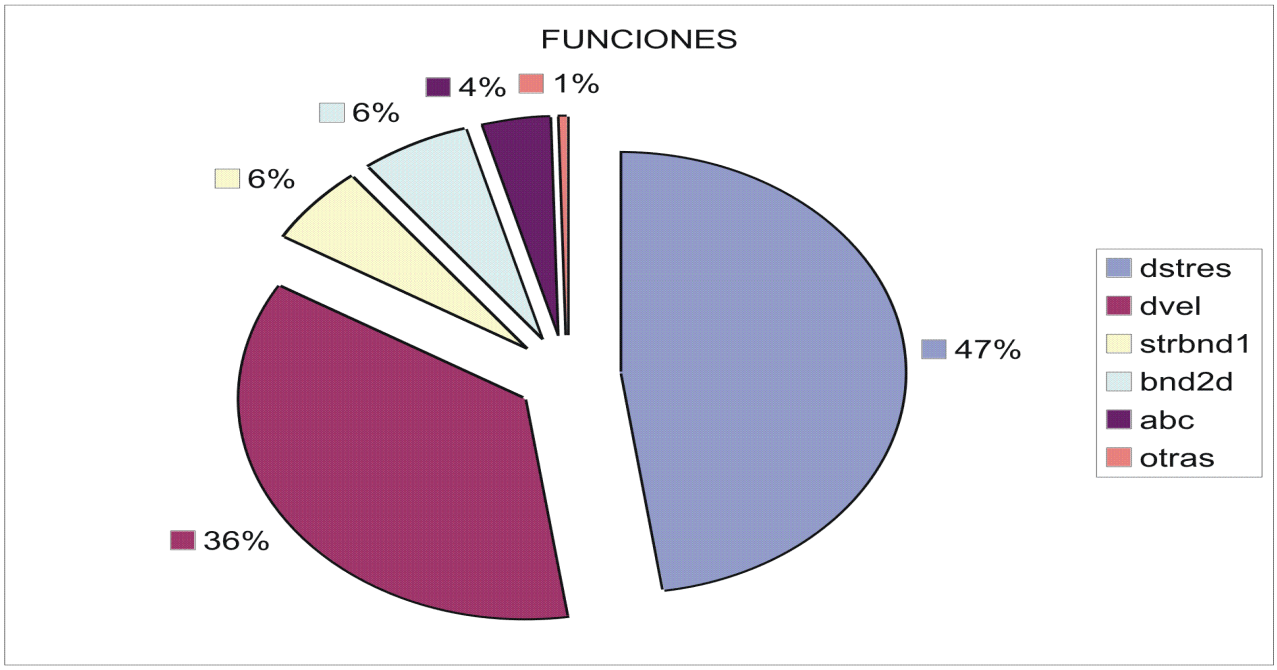


Figura 3.10: Perfilado del programa serial, en el cual se muestran las rutinas que consumen el mayor porcentaje del tiempo de ejecución: *dstres, dvel, strbnd1, bnd2d y abc.*

Se utilizó la biblioteca *MPI*, por las características listadas en la sección previa 3.7.2.6 y el paralelismo de datos (espacial o de dominio) –con el diseño propuesto por *Foster* [40], cuyas fases, aplicadas al programa serial, se describieron en la sección 3.6.2– por su característica de eficiencia, al ser escalable [42] y *ad hoc* para modelos grandes de propagación de onda elástica, que demandan demasiados recursos, los cuales imposibilitan que puedan caber en un solo procesador o nodo.

En la implementación del programa se realizó la paralelización más balanceada y agresiva; es decir, en *3d*, la cual es la más eficiente, aunque como se mostró en la etapa de aglomeración (sección 3.7.2.3), es necesario agrupar varias tareas pequeñas para obtener mayor localidad de los procesos y datos, y así mejorar la eficiencia, el factor de aceleración, y por ende el rendimiento del programa.

El esquema de diferencias finitas espacial de cuarto orden requiere dos planos de memoria adicionales en cada cara para calcular de forma completa y adecuada las soluciones de cada subdominio de forma independiente de los otros procesadores; por lo tanto, se reservaron subdominios de relleno en cada cara del cubo o paralelepípedo, Fig. 3.11, para asegurarnos, como se cito antes, del preciso funcionamiento del algoritmo de diferencias finitas alternado utilizado.

A continuación se describe la implementación paralela del algoritmo de **DFA3D** empleando *MPI*, empleando el esquema *maestro-esclavo* y paralelismo de datos.

La lectura de los parámetros básicos de la ejecución, que son datos escalares y reales, es realizada por el procesador *0* o *maestro* y transferido a los otros procesos o *esclavos* mediante

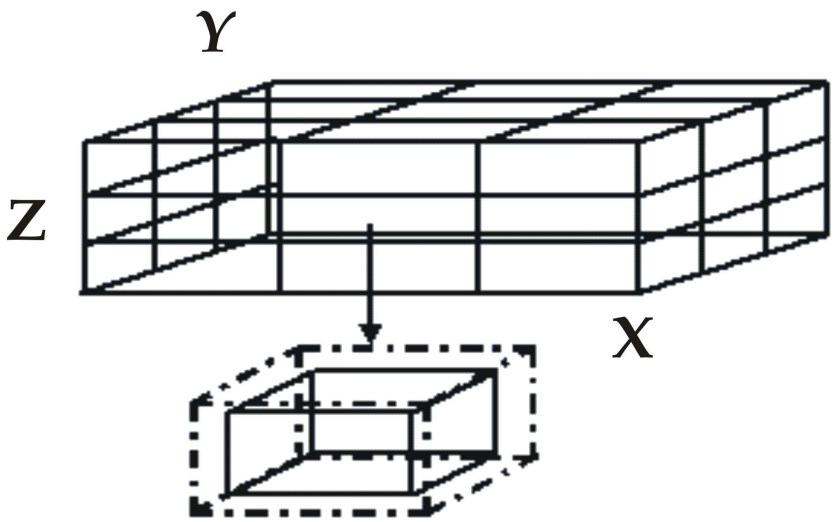


Figura 3.11: En la descomposición *3d* empleando paralelismo de datos se muestra un subdominio independiente con las celdas de relleno, representadas con líneas punteadas.

la instrucción de *MPI* de comunicaciones colectivas:

`MPI_BCAST`

La implementación de la etapa de partición es realizada mediante la creación de una topología virtual cartesiana con la instrucción de *MPI*

`MPI_CART_CREATE`

utilizada para representar cartesianamente el modelo del problema con la descomposición del dominio Fig. 3.9d).

La lectura, tanto de los datos básicos, como la de los datos que definen las características del volumen terrestre modelado (velocidades y densidades) es realizada en paralelo. Las instrucciones de *MPI* mediante las cuales se ejecutan estas tareas son las siguientes

`MPI_FILE_OPEN`

`MPI_FILE_READ_ORDERED`

Dentro del ciclo de simulación, en el cual se llevan a cabo las actualizaciones de las variables físicas del modelo, se realizan las comunicaciones necesarias para que cada proceso envíe la información a cada uno de sus procesos vecinos en *3d* (arriba, abajo, derecha, izquierda, atrás y adelante) Fig. 3.12 y con ello cada procesador calcule la solución de su propio subdominio de manera independiente. Dichas comunicaciones están implementadas por medio de las siguientes instrucciones de *MPI*:

`MPI_CART_SHIFT`

`MPI_SENDRECV`

la primera de ellas define -utilizando la topología cartesiana antes establecida- la operación de comunicación a lo largo de una dirección o eje cartesiano (x,y,z); la segunda realiza la comunicación -en una sola instrucción envía y recibe. Las rutinas involucradas en estas instrucciones son las que realizan las actualizaciones mediante diferencias finitas alternadas; las que aplican las condiciones de frontera y las rutinas de escritura de sismogramas y *snapshots*.

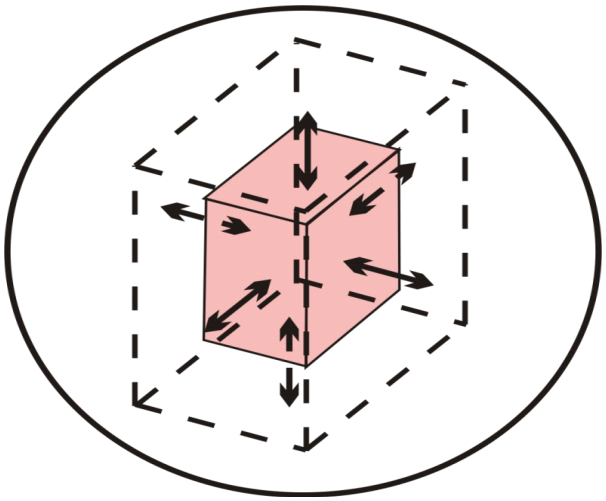


Figura 3.12: Esquema que representa la intercomunicación entre los subdominios para la actualización de los trece parámetros en el algoritmo de **DFA3D**. Un subdominio independiente en *3d* (sombreado) con las celdas de relleno (líneas punteadas) y las flechas que indican el sentido de la comunicación (arriba, abajo, derecha, izquierda, atrás, adelante).

Finalmente, la escritura de los resultados, los sismogramas y los planos de *snapshots*, es realizada de forma serial por el proceso *maestro*; por lo tanto, la instrucción colectiva de agrupación de los datos es requerida:

`MPI_GATHER`

con la cual, todos los procesos *esclavos* envían el contenido de su propio subdominio, de un plano o de los puntos de interés y el proceso *maestro* realiza la escritura de los datos. Todas estas instrucciones de *MPI* se muestran en el diagrama de flujo de las Figs. 3.13 y 3.14.

3.8. ARQUITECTURA DE KANBALAM.

Para poder determinar el análisis teórico del rendimiento es imprescindible conocer algunos parámetros del sistema. Por lo cual, se describe, de manera breve, la arquitectura de la supercomputadora de la *UNAM*, *KanBalam* [7] empleada para la simulación de este trabajo.

El sistema *HP Cluster Platform* 4000, *KanBalam* Fig. 3.15, es la supercomputadora paralela más poderosa de México y América Latina con una capacidad de procesamiento de 7.113 *Teraflops* (7.113 billones de operaciones aritméticas por segundo). Cuenta con 1,368 procesadores (núcleos *AMD Opteron* de 2.6 GHz) organizados en 342 nodos, cada nodo con 4 núcleos; una *RAM* total de 3 TB físicamente distribuida y un sistema de almacenamiento masivo de 160 TB.

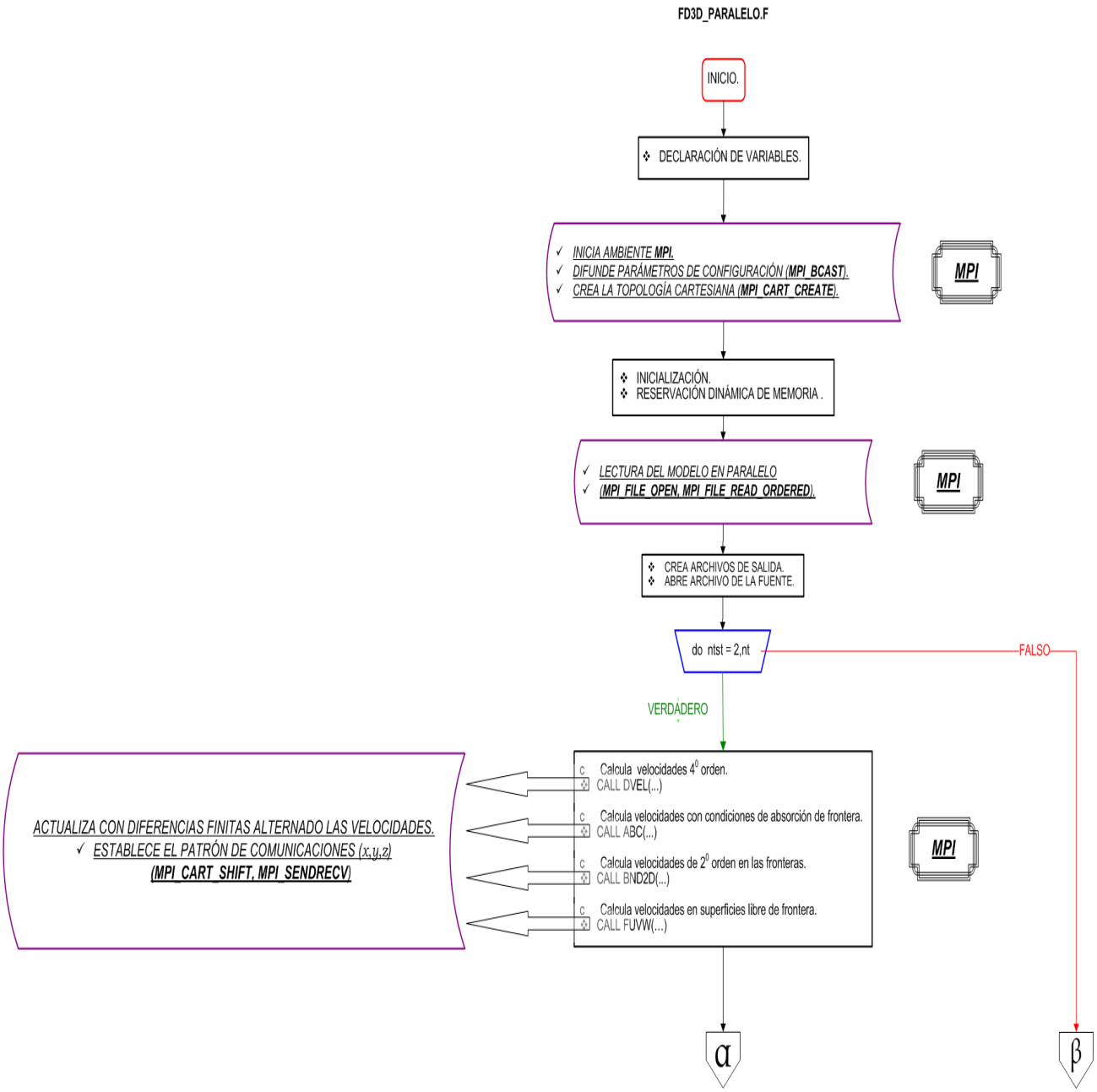


Figura 3.13: Diagrama de flujo en bloques del algoritmo de diferencias finitas alternado en paralelo con las instrucciones de *MPI* utilizadas.

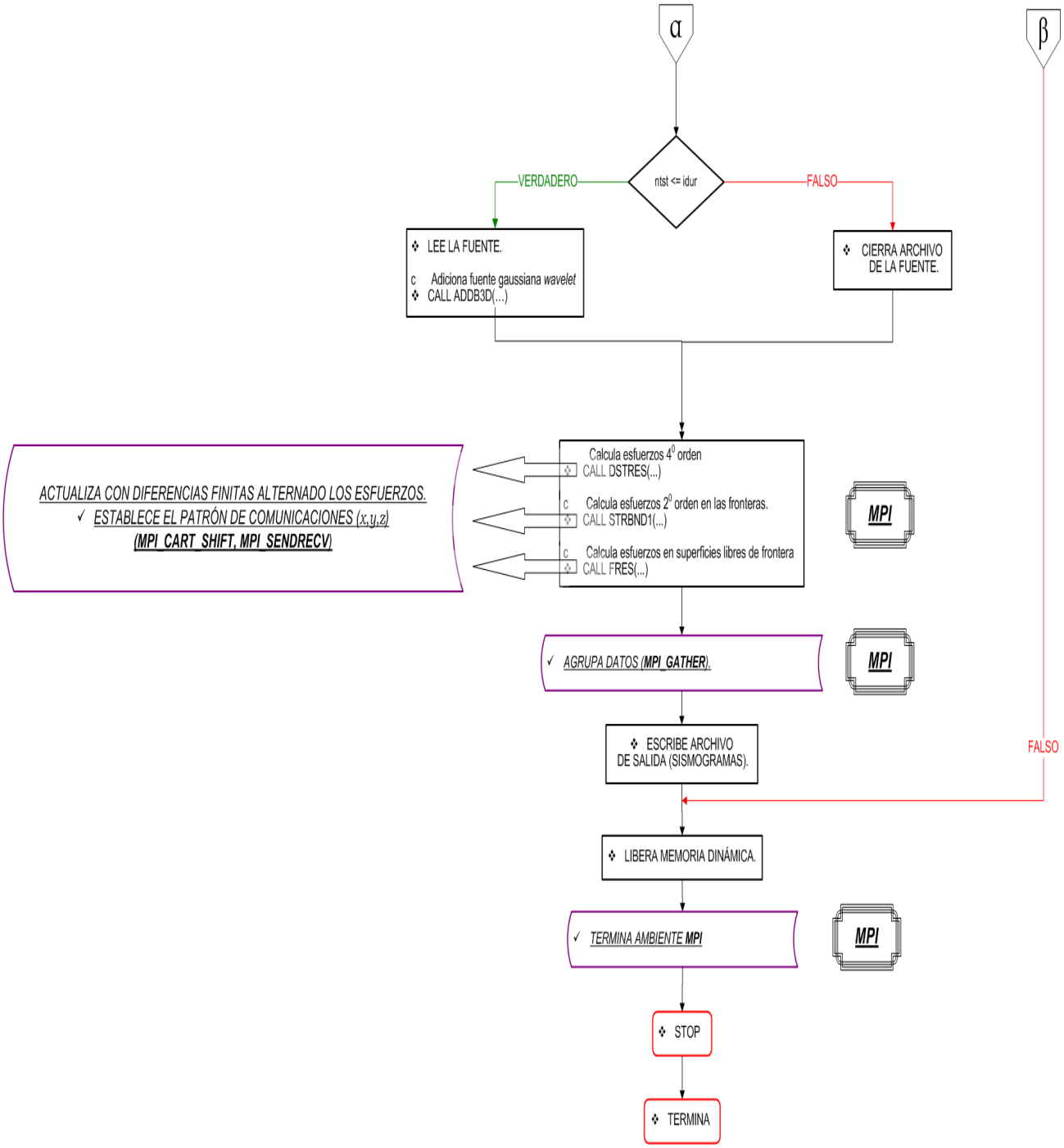


Figura 3.14: Diagrama de flujo en bloques del algoritmo de **DFA3D** en paralelo con las instrucciones de *MPI* utilizadas señaladas (continuación).

Cada nodo de cómputo de *KanBalam* está constituído por 2 procesadores duales *AMD Opteron* 285 de 2.6 GHz; es decir, tiene 4 núcleos por nodo. Con una memoria *RAM* total por nodo de 8GB y un rendimiento teórico pico de 5.2 *Gflops* por núcleo, para un total de 20.8 *Gflops* por nodo.

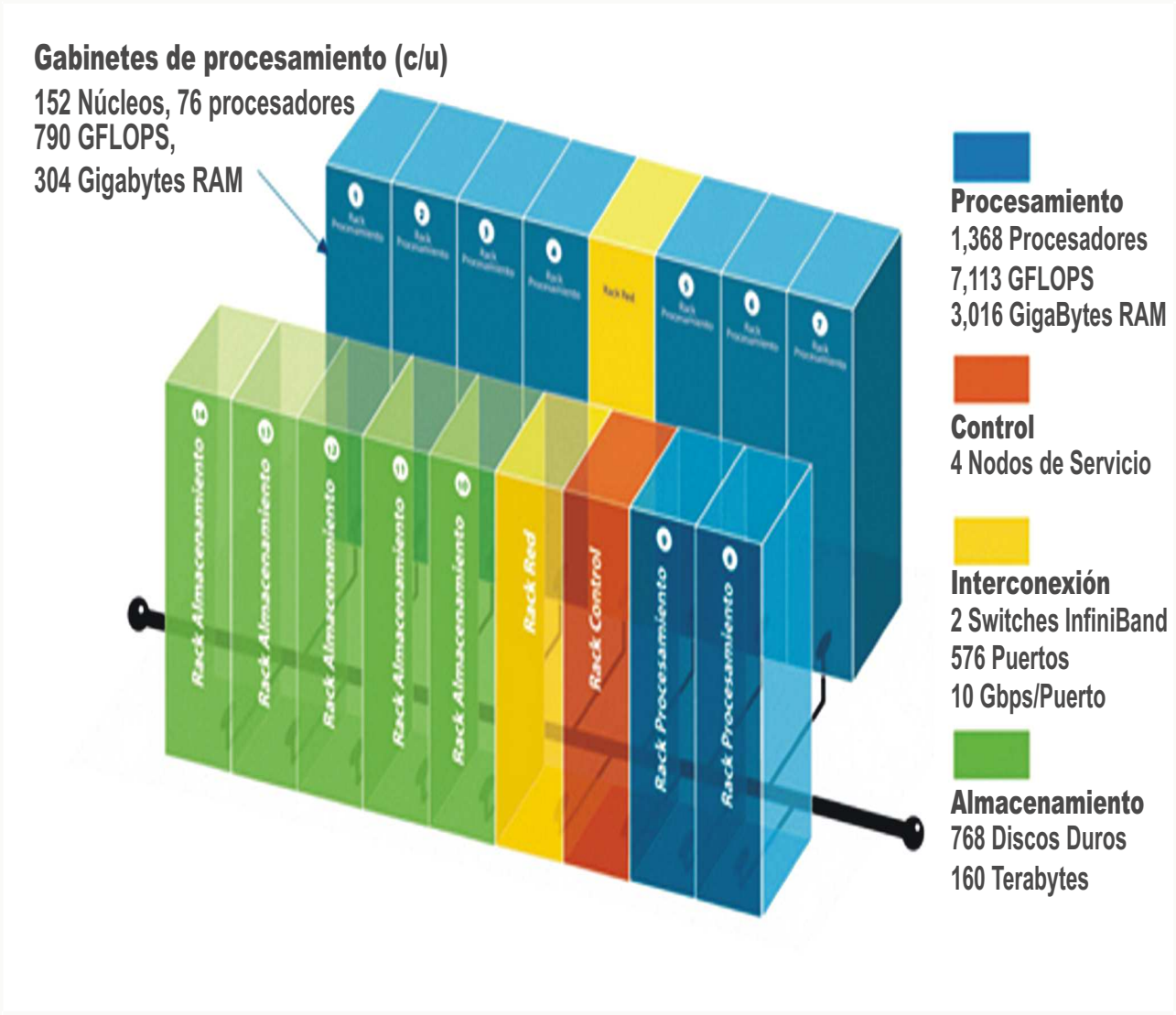


Figura 3.15: Esquema de *KanBalam* en bloques. Los bloques de procesamiento (1-9) representan los gabinetes de los nodos de cómputo (1,368 núcleos, 152 por cada bloque); el componente de control (4 nodos de servicio); las unidades más claras son los gabinetes de interconexión de la red de datos (2 *switches Infiniband* [53]) y los elementos del (10-14) constituyen los nodos de almacenamiento (160 TB) (Modificada del Departamento de Supercómputo [7]).

Además, cada nodo posee un disco duro de 160 GB Fig. 3.16. Un parámetro importante de la capacidad de cómputo de *KanBalam* es el tiempo de cómputo por *flop*, el cual es 1.9×10^{-9} s/*flop*.

El sistema de almacenamiento masivo de 160 TB es *Lustre* [54]. El cual es un sistema de archivos escalable, seguro, robusto y confiable para cúmulos.

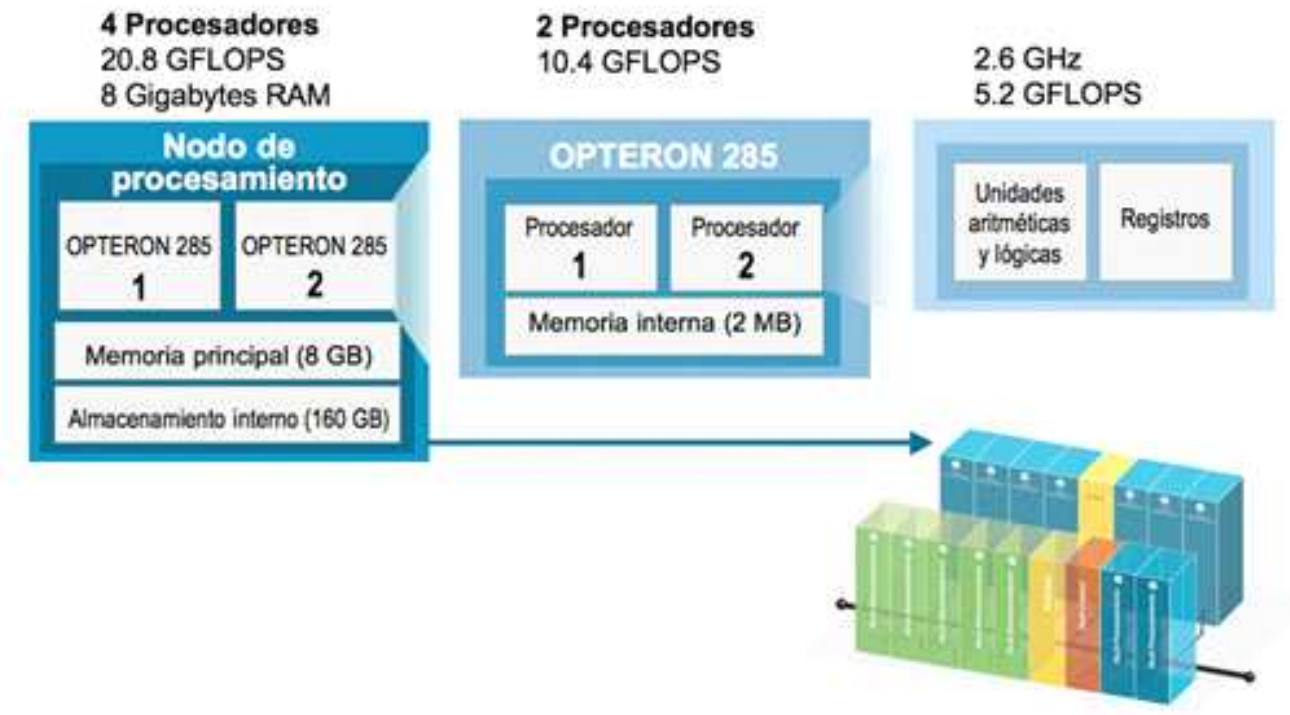


Figura 3.16: Organización de los nodos de cómputo de KanBalam. Cada uno de ellos contiene 2 procesadores duales *Opteron 285* (4 núcleos) de 2.6 GHz de frecuencia de reloj y un rendimiento teórico pico de 5.2 *Gflops*. Con una *RAM* total de 8 GB y un rendimiento por nodo de 2.8 *Gflops*. Incluido un disco duro local de 160 GB en cada nodo (Modificada del Departamento de Supercómputo [7]).

KanBalam tiene tres redes de interconexión, la principal es la red de datos, la cual está compuesta por dos *switches Infiniband* 4X de 288 puertos. La velocidad de cada conexión es de 10 Gigabits/s. La topología es *fat tree-half blocking*, de modo que cada nodo tiene una conexión a uno de los *switches* (192 nodos en cada *switch*), mientras que ambos *switches* tienen 96 conexiones entre sí, Fig. 3.17. Además, un parámetro importante es la latencia de la red, la cual es de 13×10^{-6}s. A esta red se conectan todos los elementos del cúmulo y se utiliza para la comunicación de datos entre los procesos y para las operaciones de E/S en el sistema de archivos principal.

Las otras dos redes son la red de administración y de consolas. La red administración tiene 10 *switches Gigabit ethernet* de 48 puertos y un ancho de banda de 1 Gigabit/s, con una topología de interconexión tipo estrella. Mediante esta red se realiza la configuración, organización y control del equipo en general.

La red de consolas tiene 10 *switches Fast ethernet* de 50 puertos y un ancho de banda de 100 Megabits/s, con una topología de interconexión tipo estrella, también. Permite revisar los sensores, realizar el apagado y encendido de nodos y es una red alterna cuando la red de administración no está disponible.

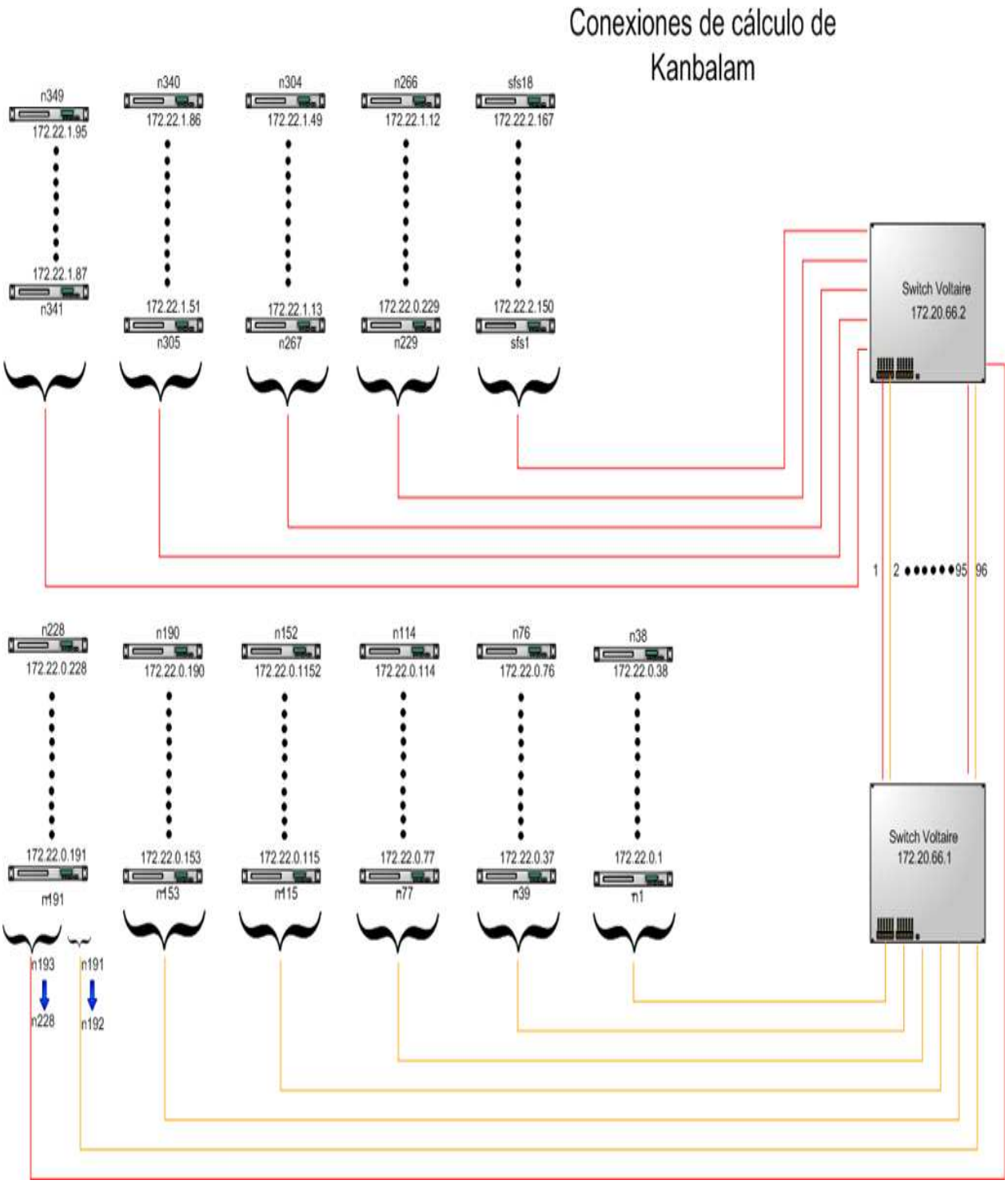


Figura 3.17: Red de interconexión de datos. La topología es *fat tree-half blocking*. Constituida por 2 *switches Infiniband* de 288 puertos. Con una velocidad de transferencia de 10 Gigabits/s (Modificada del Departamento de Supercómputo [7]).

3.9. ANÁLISIS DE RENDIMIENTO.

Como ya se mencionó antes, el factor de aceleración y la eficiencia, *Sp* y E, respectivamente, son las métricas más importantes al caracterizar el rendimiento de los programas paralelos.

En este trabajo se utiliza el esquema del problema de tamaño variable, el cual es el más adecuado para códigos de propagación de onda elástica en grandes problemas (vastos recursos utilizados tanto en memoria principal *RAM* (memoria de acceso aleatorio; por sus siglas en inglés) como en memoria secundaria o almacenamiento en disco duros y en tiempo de cómputo para que puedan caber en computadoras uniprocesador), los cuales son la motivación y esfuerzo de este trabajo, al llevar a cabo la paralelización. Por lo tanto, se deberá estimar tanto el factor de aceleración *Sp*, como la eficiencia E, dados por las ecuaciones (3.2) y (3.9), respectivamente.

3.9.1. COSTO TEÓRICO.

Podemos estimar el costo teórico del programa paralelizado en este trabajo de forma directa y relativamente sencilla, sin considerar los costos involucrados en las operaciones de entrada/salida (E/S), ya que el mayor tiempo de ejecución del algoritmo de interés, ocurre en la actualización de las velocidades y esfuerzos, derivadas de la propagación de las ondas sísmicas.

Para realizar la estimación del costo, debemos calcular los términos de comunicación y computación de la ecuación (3.5).

Existen dos parámetros principales de máquina que impactan en gran medida en la velocidad de las comunicaciones. El primero es el ancho de banda –que está en función del tamaño del mensaje– y es el recíproco de la transmisión tiempo/*byte*. El segundo es la latencia ι,–que es independiente del tamaño del mensaje. De tal manera que el costo de enviar un solo mensaje de longitud de datos χ es

$$\iota + \chi\beta \tag{3.10}$$

;donde β es 1/ancho de banda. Como se mencionó para *KanBalam* [7] su ancho de banda es de 10 Gbits/s; su latencia $\iota = 13 \times 10^{-6}$ s y tiempo de cómputo por *flop* $1.9 \times 10^{-9}\Gamma$ s/*flop*.

Por otro lado, el tamaño de los mensajes que se envían en el algoritmo de **DFA3D** depende de: la discretización *1d, 2d, 3d*, Figs. 3.9b), c), y d), respectivamente y de la cantidad de procesadores utilizados en cada caso; del tamaño de las discretizaciones $\Delta x, \Delta y, \Delta z$; de las dimensiones físicas del problema L_x, L_y, L_z y del orden del algoritmo de diferencias finitas, de cuarto orden espacial y de segundo orden temporal, utilizado en este trabajo. Así, para el caso *3d* los subdominios quedarían $N_x = L_x/(\Delta x \times n_x), \quad N_y = L_y/(\Delta y \times n_y), \quad N_z = L_z/(\Delta z \times n_z)$;donde $n_x \times n_y \times n_z = M$ procesadores; por lo tanto, el tamaño total de subdominios en que se discretiza el problema físico es: $R = N_x \times N_y \times N_z$ y el costo para ejecutar el cálculo del esquema de **DFA3D** es

$$\mathrm{A}\Gamma\mathrm{R} \tag{3.11}$$

;donde A es la cantidad de operaciones de punto flotante (reales) realizadas en el esquema de **DFA3D** (*velocidad-esfuerzo*, que consiste de nueve variables –6 esfuerzos y 3 velocidades– estrechamente acoplados que son actualizadas en cada iteración [capítulo 2]). Además, el esquema

de **DFA3D** requiere que se envíen 4 planos de datos a cada vecino en el cubo de *3d* en cada dirección.

Por lo tanto, el costo de comunicación para una descomposición de dominio *1d*, utilizando la ecuación (3.10) es, máximo

$$8\left(\iota + 4\beta \mathrm{R}_1\right) \tag{3.12}$$

;donde el 4, dentro del paréntesis, indica el tamaño de la palabra en *bytes*; es decir, que cada dato utiliza 4 *bytes* de memoria en *KanBalam* y $R_1 = N_i \times N_j$ con $i \neq j = x \oplus y \oplus z$, y diferente del eje sobre el cual se particiona. Por ejemplo, para una paralelización en Z, tenemos $8\left(\iota + 4\beta(N_x \times N_y)\right)$

Para una partición *2d* el costo de la comunicación está dada por

$$16\left(\iota + 4\beta R_2\right) \tag{3.13}$$

;donde $R_2 = (N_i \times N_j) + (N_i \times N_k)$ con $i \neq j \neq k = x \oplus y \oplus z$, y diferente del eje sobre el cual se particiona. Por ejemplo, para una paralelización en Y y Z, tenemos $16\left(\iota + 4\beta(N_x \times N_z + N_x \times N_y)\right)$

Por último, para una descomposición *3d* tenemos que el costo asociado a las comunicaciones es

$$24\left(\iota + 4\beta \mathrm{R}_3\right) \tag{3.14}$$

;donde $R_3 = (N_i \times N_j) + (N_i \times N_k) + (N_j \times N_k)$ con $i \neq j \neq k = x \oplus y \oplus z$. Por ejemplo, para una paralelización *3d* tenemos $24\left(\iota + 4\beta((N_x \times N_y) + (N_x \times N_z) + (N_y \times N_z))\right)$

En resumen, para una descomposición del dominio en *3d* tenemos que el tiempo de cómputo dado por la ecuación (3.11) y comunicaciones expresado en la ecuación (3.14) es

$$T\left(\mathrm{N}, \mathrm{M}\right) = \mathrm{A\Gamma R} + 24\left(\iota + 4\beta \mathrm{R}_3\right) \tag{3.15}$$

Entonces, utilizando la ecuación (3.15) podemos reescribir el factor de aceleración de la ecuación (3.6), quedando como:

$$Sp \equiv \frac{\mathrm{A\Gamma R}}{\mathrm{A\Gamma R} + 24\left(\iota + 4\beta R_3\right)} \tag{3.16}$$

Es importante hacer notar que las constantes 8, 16 y 24 de las ecuaciones (3.12), (3.13), (3.14), (3.15) y (3.16) podrían ser reemplazadas por unas constantes cualesquiera, pero no se hizo para enfatizar que el costo de comunicación está en función tanto del orden del esquema del algoritmo de diferencias finitas alternadas como del tipo de descomposición del dominio implementada, *1d*, *2d* o *3d*.

La ecuación (3.16) es una aproximación del rendimiento de un programa paralelo, la cual refleja las operaciones inherentes de cómputo, de comunicaciones y del algoritmo utilizado. Esta ecuación es más general al evaluar el desempeño del programa paralelo ya que expresa explícitamente los aspectos que impactan en mayor medida el rendimiento de los programas. El más importante de éstos son las comunicaciones, que estarán en función del particionamiento y del algoritmo empleados y del *HW* disponible.

La granularidad es otra característica importante que afecta el rendimiento de los programas paralelos, por ello, es imperativo encontrar un equilibrio entre la cantidad de particiones realizadas y las comunicaciones necesarias para intercomunicar estas particiones. Por un lado tenemos que a mayor granularidad, " *granularidad fina*", mayor resolución, y más comunicaciones, por el otro, a menor granularidad, " *granularidad gruesa*", menor resolución y menos comunicaciones, pero es muy probable que éste último no satisfaga nuestros requerimientos de mayor precisión de los resultados.

3.10. CONCLUSIONES DEL CAPÍTULO.

A continuación se enumeran las conclusiones más relevantes de este capítulo:

1. La paralelización de un algoritmo tiene dos objetivos primarios: *a)* incrementar el tamaño del dominio del problema, permitiendo con esto obtener resultados más confiables y precisos al aumentar la resolución del problema y *b)* disminuir los tiempos de ejecución de los programas, permitiendo realizar más simulaciones en menos tiempo.

2. Las arquitecturas de las computadoras pueden clasificarse en tres tipos: *a)* por el flujo de instrucciones y datos que se desean procesar, que empleando la taxonomía de *Flynn*, corresponde a cuatro clases más: *SISD*, *SIMD*, *MISD*, *MIMD*; más combinaciones de éstas, que abarcan *grosso modo* el amplio espectro actual de las computadoras permitiendo de una manera sencilla la caracterización de los distintos equipos de cómputo. *b)* de acuerdo a la disposición de la memoria: compartida o distribuida. La cual establece lineamientos a seguir en el tipo de paradigma de paralelización a utilizar, así como también la sencillez de su implementación y los beneficios y limitaciones al utilizarlos. *c)* al tipo de procesador utilizado: vectoriales, superescalares y microprocesadores –ampliamente utilizados en los cúmulos, que dominan por amplio margen el porcentaje de computadoras de este tipo que se encuentran entre las 500 más poderosas del orbe, reportados en el *top500*.

3. Se presentaron distintos esquemas y paradigmas de paralelización: paralelismo de datos o de funciones con hebras tanto en memoria compartida o distribuida; así como esquemas de paralelización híbridos que conjunta paralelismo de datos y funciones; segmentación de datos, un programa aplicado a múltiples datos y *maestro-esclavo*, como ejemplos de paradigmas, sus ventajas, limitaciones, áreas de utilización, tipos de lenguajes de programación empleados y cuáles son las arquitecturas *ad hoc* para su implementación. En las que destaca el paralelismo de datos *3d*, por ser la paralelización más agresiva pero a la vez la más compleja de implementar.

4. Se discutieron conceptos de paralelismo y métricas de rendimiento para la evaluación de los programas, como lo son los factores de aceleración y eficiencia, utilizados por la ley de *Amdahl*, que nos proporciona una buena aproximación del rendimiento del programa, con base en el porcentaje de código paralelizable. El factor de aceleración - que se define como el cociente del tiempo que se tarda en completar el cómputo de la tarea usando un solo procesador entre el tiempo requerido para hacerlo en múltiples procesadores trabajando en paralelo - que evalúa el desempeño del programa paralelo con respecto al tiempo; es decir, cuantas veces es más rápida la ejecución del mismo respecto a su versión serial. Teóricamente, debería ser lineal la aceleración con respecto al número de procesadores utilizados. El factor de eficiencia –que varía de 0 a 1, y es función de las comunicaciones, de las sincronizaciones y del *HW* de los equipos de cómputo empleados - indica que tan bien se han empleado los recursos, en particular, los procesadores. Idealmente, 1 es la eficiencia objetivo, que representa una óptima utilización del equipo por parte del programa.

5. Se utilizaron las precondiciones de *Pancake* para verificar si es viable o no llevar a cabo la paralelización de un programa serial. Éstas son: frecuencia de uso, tiempo de ejecución

y resolución con las cuales se obtienes pautas que indican la pertinencia o no de intentar una paralelización.

6. Se aplicó al programa de interés la metodología de *Foster* para el diseño del programa paralelo conformada por cuatros etapas: *a)* partición –que consiste en dividir el problema en pequeñas tareas; *b)* comunicación –mediante la cual se realiza el intercambio de datos necesarios para la coordinación de las tareas de la etapa previa; *c)* aglomeración –que permite conjuntar las tareas, para reducir la cantidad de comunicaciones y de esta manera incrementar el rendimiento del programa y *d)* el mapeo –que consiste en asignar a cada tarea o conjunto de ellas un procesador. La partición y la comunicación centran su atención en la concurrencia y escalabilidad; mientras que la aglomeración y el mapeo su interés está dirigido a la localidad de los datos en los procesadores asignados.

7. De la aplicación de los criterios de *Pancake* y *Foster* se determinó efectuar la paralelización del algoritmo serial en *3d*, la más balanceada y agresiva, en comparación con la paralelización *1d* y *2d*, dado que la *3d* es la más eficiente de las tres particiones realizadas. Se utilizó la biblioteca de envío de mensajes *MPI*, por sus características de ser escalable, portátil y eficiente, además de ser el estándar *de facto* en la programación paralela.

8. El programa de **DFA3D** paralelizado se implementó en la supercomputadora *KanBalam*, que es un cúmulo con 684 procesadores *Opteron* duales -1,368 núcleos- organizados en 342 nodos, cada uno de ellos con sistema operativo y memoria independiente, interconectados entre sí por una red de baja latencia 13×10^{-6} s y un ancho de banda de 10 Gigabits/s –*Infiniband*.

9. Por último, con base en los parámetros de *KanBalam* de latencia y ancho de banda de la red *Infiniband* y operaciones por segundo de los procesadores, así como la cantidad de memoria requerida por las distintas discretizaciones del problema de interés se realizó un análisis teórico del rendimiento del programa con base en la ley de *Amdahl*, pero incluyendo en el análisis los costos asociados tanto al cómputo como a las comunicaciones, que están en función de la granularidad de las tareas y el orden del algoritmo de diferencias finitas utilizado. Los resultados se presentan en el capítulo 4.

Capítulo 4

GENERACIÓN DE SISMOGRAMAS SINTÉTICOS PARA EL SISMO DE MICHOACÁN DEL 19/09/1985.

En este capítulo se describirá el modelado del sismo $M_s = 8.1$ del 19 de septiembre de 1985 y sus características. Se mostrarán y discutirán los resultados computacionales obtenidos, basados en los tópicos introducidos en el capítulo 3: factor de aceleración Sp y eficiencia E de la implementación paralela en comparación con la versión serial; así como los resultados sismológicos; es decir, los sismogramas sintéticos obtenidos, su validación con respecto de los observados e importancia. Por último, se presentan las conclusiones de este capítulo.

4.1. MODELACIÓN NUMÉRICA.

Como se mencionó en el capítulo 2, el modelado numérico es una herramienta indispensable en la actualidad en prácticamente todas las áreas de la ciencia e ingeniería. Involucra, al menos, tres etapas al emplearse metodológicamente [55]: 1) se debe formular una descripción matemática del fenómeno de interés; 2) deben concebirse técnicas efectivas para resolver las ecuaciones que rigen el fenómeno planteadas en la etapa previa, descripciones que usualmente son resultas utilizando programas de cómputo y 3) debe existir una retroalimentación de las dos primeras fases ya que se debe tener un entendimiento cualitativo del problema físico, de las ecuaciones matemáticas y sus correspondientes métodos numéricos empleados para garantizar que los números (resultados obtenidos) reflejen la física del problema.

Como se estableció en el capítulo 2, sección 2.2, de este trabajo, en ocasiones la única aproximación posible y práctica de simulación de un fenómeno es emplear modelos numéricos computacionales. Los cuales se han convertido en una poderosa aproximación para entender y predecir fenómenos naturales.

4.1.1. MODELO DEL SISMO DE 1985.

El 19 de septiembre de 1985 ocurrió un sismo de subducción superficial con epicentro en las costas michoacanas del Océano Pacífico, a unos 340 km de la Ciudad de México y cuya magnitud fue de $M_s = 8.1$. En la Fig. 4.1 se muestra la proyección superficial del área de ruptura (rectángulo interno) de este evento, de tamaño 180×140 km; también, se muestra la superficie total empleada en su modelación, cuyas dimensiones son 500×600 km (rectángulo externo). Las Figs. 4.1, 4.2 y 4.3 presentan las dimensiones físicas del evento simulado. Tomando en cuenta la Fig. 4.2 se decidió que un volumen de $500 \times 600 \times 124$ km, representase el dominio físico del problema.

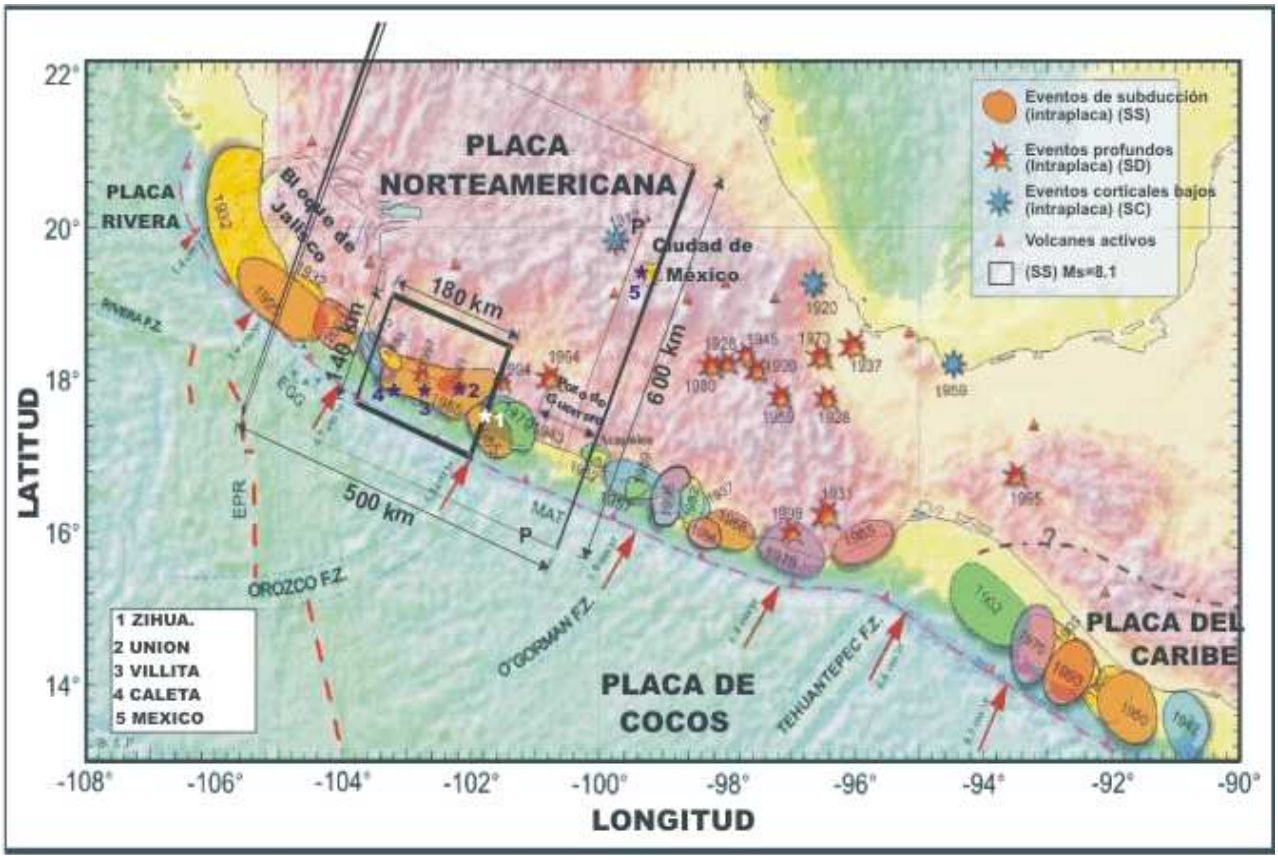


Figura 4.1: En esta imagen se ilustran las dimensiones utilizadas en la modelación física del sismo del 19/09/1985. El rectángulo externo indica 500 x 600 km superficiales (en planta) que incluye las zonas epicentral (180×140 km de superficie, rectángulo interno) y de la ciudad de México –incluyendo la ubicación de 4 puntos de los cuales se tienen observaciones del sismo. Se muestran, también, distintos eventos históricos, el tipo de ellos y la ubicación de latitud y longitud de la zona. (Modificada del Servicio Sismológico Nacional).

El modelo matemático; es decir, la descripción matemática de las ecuaciones que gobiernan la propagación de ondas sísmicas desde su hipocentro del sismo de 1985 hasta, en este caso, la ciudad de México fue especificado en la sección 2.1 del capítulo 2, así como también el método numérico, diferencias finitas alternadas, empleado en la simulación de este trabajo, sección 2.2.2.

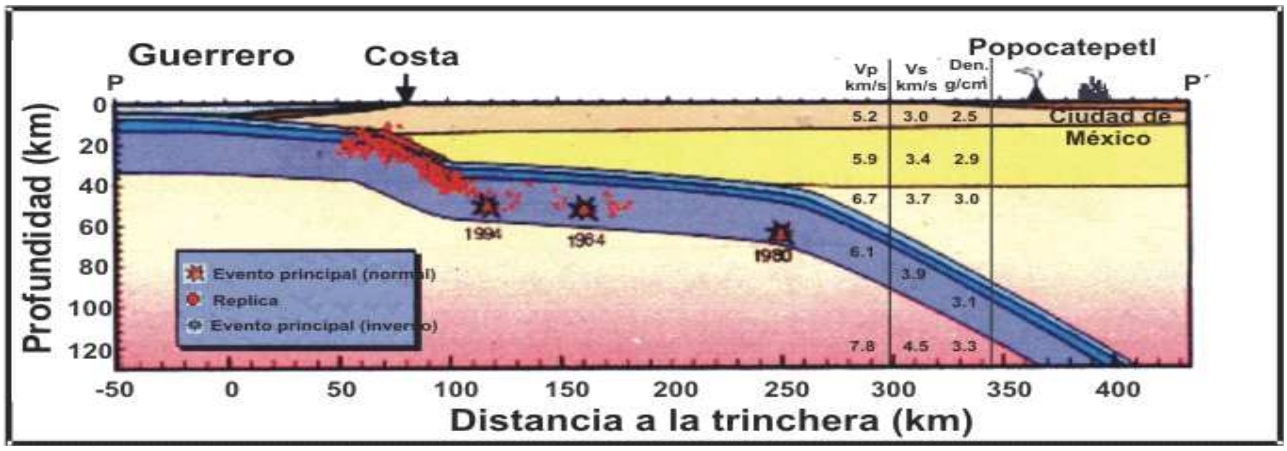


Figura 4.2: Perfil desde la costa del Océano Pacífico hasta la ciudad de México que muestra las placas tectónicas involucradas en la generación de sismos de subducción superficiales en México, así como las características de velocidades de propagación de las ondas $\boldsymbol{P}$ y $\boldsymbol{S}$ y las densidades del subsuelo (Servicio Sismológico Nacional).

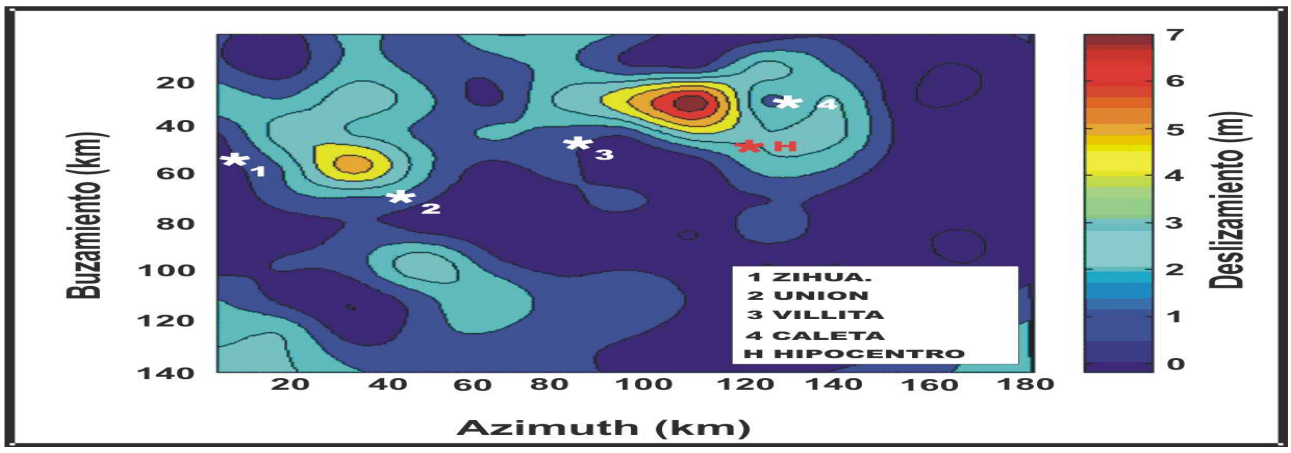


Figura 4.3: Esquema con la representación cinemática del desplazamiento promedio asociado al sismo –hasta 7m de deslizamiento entre las placas– en la zona epicentral. Tres sub-eventos participaron en la generación del sismo en cuestión. Con H está indicado el hipocentro y, en números del 1 al 4, las localidades en las cuales se obtuvieron registros. El tamaño de la zona de ruptura es de 180 × 140 km [56].

Como se discutió en el capítulo 3, el modelo computacional utilizado es la implementación paralela del código de **DFA3D**, a partir de su versión serial en *Fortran77*, empleando la biblioteca *MPI* [30, 31] (interfaz de envío de mensajes; por sus siglas en inglés) descrito en la sección 3.7, del capítulo 3. Esta versión paralela fue ejecutada en la supercomputadora HP CP 4000-*KanBalam* de la UNAM, que tiene 1,368 núcleos o procesadores *AMD Opteron*; 3 TB de *RAM* y un almacenamiento de 160 TB [7], descrita en la sección 3.8 del capítulo citado.

4.2. RESULTADOS COMPUTACIONALES.

Como se indicó en el mismo capítulo 3, se utilizó la paralelización más agresiva y más compleja, pero que proporciona mejor escalabilidad y rendimiento. Se realizaron distintas paralelizaciones en *1, 2 y 3d*. Se llevaron a cabo varias ejecuciones con distinta cantidad de procesadores en cada uno de los ejes coordenados del problema. Como se indicó en el capítulo anterior, nos enfocaremos en el problema de tamaño variable; es decir, donde la cantidad de procesadores utilizados se incrementa a medida que se utilizan discretizaciones espaciales Δh más pequeñas. En este caso, $\Delta h = 1.0, 0.5, 0.25$ y 0.125 km, como se discutirá más adelante, proporcionó más y mejor información de las características del sismo y sus repercusiones tanto en la zona de la fuente del mismo como en la ciudad de México, al emplear $\Delta h = 0.125$ km. En la tabla 4.1 se resumen 4 ejecuciones con distinto tamaño del problema y mayor discretización del mismo; la cantidad de procesadores utilizados; el tiempo total de ejecución; el factor de aceleración *Sp*) obtenido; la eficiencia E alcanzada y la cantidad de memoria empleada en cada subdominio de la partición.

Con base en los datos de la tabla 4.1 discutiremos los resultados observados. Para obtener las métricas de factor de aceleración y eficiencia debemos ejecutar las mejores versiones serial y paralelas. El experimento consistió en ejecutar la versión serial y obtener el tiempo de ejecución de referencia, así como la memoria empleada por este programa (2.08 GB), que está dentro de los límites de recursos que poseen los 337 nodos regulares de *KanBalam* [7], cuya arquitectura fue descrita en el capítulo 3. Posteriormente, se determinó emplear 1.042 GB de memoria por subdominio en cada una de las ejecuciones paralelas, para no saturar la memoria de cada núcleo (4 por nodo y teóricamente 2 GB de RAM por núcleo) y obtener un buen rendimiento.

Tabla 4.1: Incluye el tamaño del modelo y de la discretización empleados Δh especificadas en km; la cantidad total de procesadores M utilizados y los correspondientes a cada dimensión Mx, My, Mz; el tiempo total de ejecución en segundos; el factor de aceleración *Sp*; la eficiencia lograda E y la memoria utilizada por cada proceso Mps en *Gigabytes*. [57]

Tamaño y discretización del espacio estudiado (Δh, km)	M	Mx	My	Mz	Tiempo de ejecución (s)	Factor de Aceleración (*Sp*)	Eficiencia (E)	Mps (GB)
500x600x124 (1)	1	1	1	1	34,187.7	1	1	2.08
1000x1200x248 (0.5)	16	1	4	4	33,201.5	16.47	1.03	1.042
2000x2400x496 (0.25)	128	4	8	4	36,230.3	120.8	0.94	1.042
4000x4800x992 (0.125)	1,024	16	16	4	52,203.2	670	0.65	1.042

Para el primer caso paralelo, Fig. 4.4, con un $\Delta h = 0.5$ km, tenemos 1000 x 1200 x 248 elementos (el doble de la versión serial). Los cuales se dividieron en 16 tareas, 1 sobre el eje X, 4 sobre el eje coordenado Z y 4 más en el eje Y; es decir, en *2d*. Se hizo la división de esta forma para mostrar que también es posible lograr un buen rendimiento del programa, aunque

no tanta escalabilidad e ir gradualmente a la más agresiva y escalable. Una eficiencia superior a 1 es indicativo que se utiliza de forma óptima tanto la memoria *cache* de los procesadores así como la red de intercomunicación *Infiniband*. La cantidad de datos totales comunicados en cada iteración de la ejecución del problema es de 69,504,000 *bytes*; es decir, aproximadamente 70 MB en 48 comunicaciones; 24 comunicaciones tanto en el plano XZ como en el plano XY. Por lo tanto, obtenemos un muy buen factor de aceleración de 16.47, que indica un buen equilibrio entre la cantidad de particiones y comunicaciones realizadas, el cual supera la ley de *Amdahl* [38].

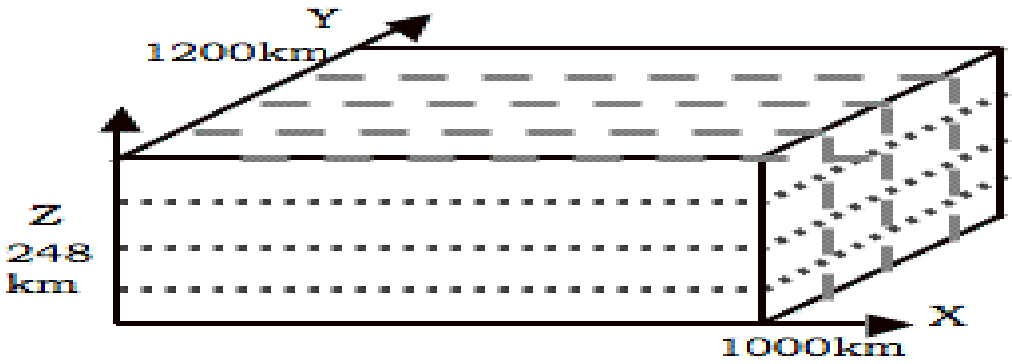


Figura 4.4: Representación de la partición *2d* que se implementó con una $\Delta h = 0.5$ km. Cuatro divisiones tanto en el eje Y como el Z; es decir, 16 paralepípedos cuyas dimensiones son de 1000 x 300 x 62 elementos.

En la segunda paralelización Fig. 4.5, con un $\Delta h = 0.25$ km y $2000 \times 2400 \times 496$ elementos. Se dividió el dominio en 128 subdominios, 4 en el eje X, 8 en el eje Y y 4 con respecto al eje Z (cada uno de estos subdominios con un tamaño de $500 \times 300 \times 124$ elementos en las direcciones X, Y, Z, respectivamente). En cada iteración se realizan 592 comunicaciones, 192 en los planos YZ y XY, 208 en el plano XZ- para un total de datos comunicados de 390,707,200 *bytes*, aproximadamente 390 MB. Se obtuvo una eficiencia cercana a la unidad 0.94, dado que la cantidad de comunicaciones antes citadas ya empieza a ser un factor, así como también el uso intensivo de memoria *cache* por cada proceso.

En el tercer y último caso Fig. 4.6, se utilizaron 1,024 procesadores en tantos subdominios de $250 \times 300 \times 248$ elementos– una partición en *3d* con 16 particiones tanto en los ejes X como en el eje Y y 4 en el eje Z y una $\Delta h = 0.125$ km. La cantidad de memoria total utilizada es 1.07 TB; cada uno de esos 1,024 subdominios emplea, como en los otros casos paralelos 1.042 GB de memoria RAM y un tamaño de $250 \times 300 \times 248$. Al tener una paralelización *3d* y con 1,024 tareas comunicando de forma constante tenemos como resultado un total de 3,584 comunicaciones y 3,016,704,000 *bytes*, aproximadamente 3 GB, por cada iteración; lo cual impacta, en gran medida, en el denominador de la ecuación (3.6) para obtener un factor de aceleración de 670 y, por ende, una eficiencia de 0.65; que es una aceptable eficiencia en programas masivamente paralelos. Los cuales se ven afectados por la gran cantidad de comunicaciones y volumen de datos

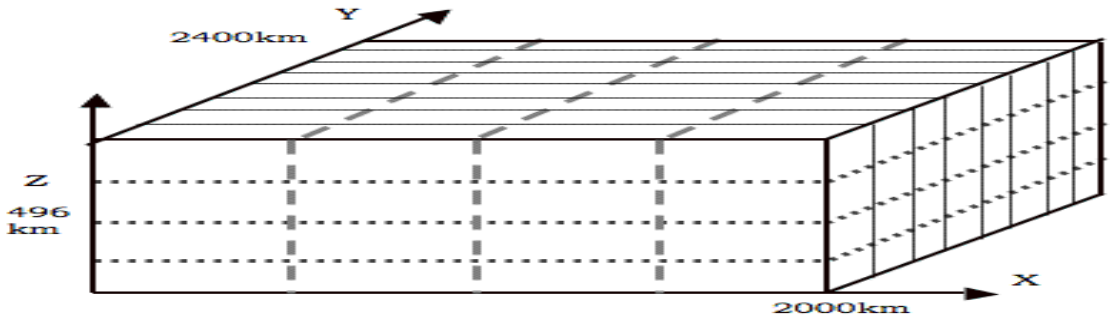


Figura 4.5: Partición *3d* utilizada con una $\Delta h = 0.25$ km. Cuatro divisiones en el eje X, 8 en el eje Y y 4 en el eje Z; es decir, 128 paralepípedos cuyas dimensiones son de $500 \times 300 \times 124$ elementos.

transmitidos, que saturan el HW utilizado, así como también el uso intensivo de la memoria *cache* compartida de los núcleos por procesador. En la Fig. 4.7, se comparan el Sp teórico con el obtenido en el experimento computacional recién descrito.

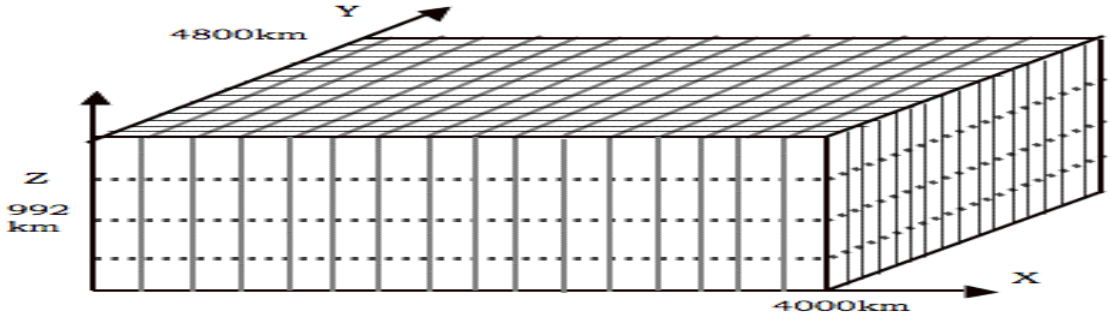


Figura 4.6: Esquema que ilustra la partición *3d* utilizada con una discretización de 0.125km. 16 divisiones en el eje X, 16 en el eje Y y 4 en el eje Z; es decir, 1,024 paralepípedos cuyas dimensiones son de 250 x 300 x 248 elementos.

La discretización de 0.125 km es la máxima que podemos realizar utilizando *KanBalam*, y los resultados sismológicos obtenidos de cada una de las cuatro ejecuciones ya discutidas y resumidas en la tabla 4.1 se mostrarán a continuación.

4.3. RESULTADOS SISMOLÓGICOS.

Los sismogramas, que son registros *tiempo-velocidad* del movimiento del suelo ocasionado por fuentes naturales, como los sismos, o artificiales, como explosiones, son registrados por

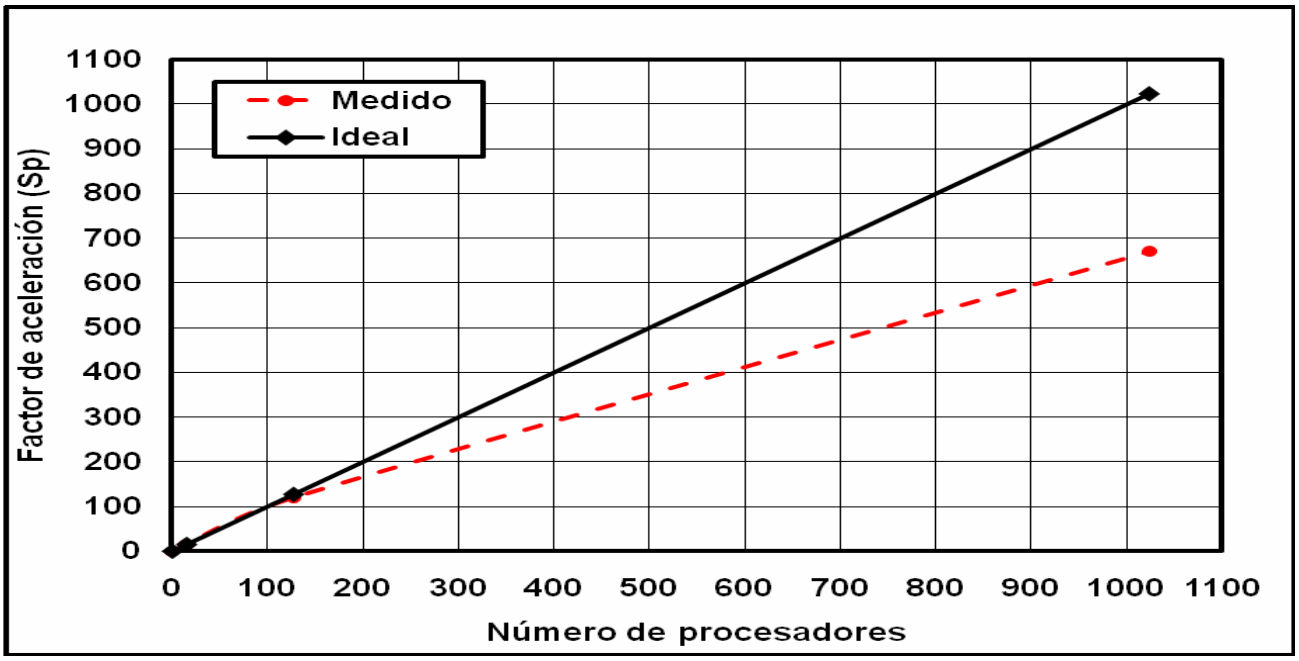


Figura 4.7: Gráfica que muestra los factores de aceleración *Sp* ideal, en línea continua, y el registrado en las simulaciones, línea discontinua, para 1, 16, 128 y 1,024 procesadores (Tabla 4.1).

sismógrafos. La duración y la amplitud máxima de las ondas son los datos captados en los registros del sismógrafo.

Durante el sismo de 1985 se obtuvieron en cinco estaciones distintas registros de aceleraciones y velocidades. Cuatro de ellos en la zona de la fuente Fig. 4.3, y el último corresponde a Tacubaya (México, D.F.) Todos estos sismogramas observados son las referencias para evaluar los resultados de los sintéticos obtenidos por las simulaciones, así como también la diferencia de emplear distintos tamaños de discretización espacial Δh. Las comparaciones de los sismogramas observados *vs* los sintéticos se realizan tanto en el dominio del tiempo como en el de la frecuencia. Estos últimos mediante los espectros de las amplitudes de *Fourier* de los sismogramas. En el dominio del tiempo las comparaciones de los sismogramas observados *vs* los sintéticos nos permiten apreciar diferencias en formas de las amplitudes de las ondas sísmicas; mientras que en el dominio de la frecuencia, las contribuciones de las frecuencias a las amplitudes máximas de las señales.

Las Figs. 4.8 muestran las aceleraciones observadas para el sismo del 19/09/1985 en las cinco estaciones ya descritas Fig. 4.3, tanto en el dominio del tiempo 4.8(a) como en la frecuencia 4.8(b), respectivamente. Con aceleraciones observadas de hasta $1.65 m/s^2$ en la zona de la fuente sísmica y amplitudes hasta de 1.3, alrededor de 1 Hz, en el dominio de la frecuencia. En la Fig. 4.8(a) las señales que se encuentran en la zona de la fuente terminan su registro en los segundos 50, 65, 63 y 73 aproximadamente, mientras que la de Tacubaya (México, D.F.) en el segundo 165 aproximadamente. Las escalas de tiempo y velocidad son distintas en la última señal, Tacubaya.

En las Figs. 4.9 se ilustran en el dominio del tiempo 4.9(a) y de la frecuencia 4.9(b), respectivamente, los registros de las velocidades en baja frecuencia (f $\leq$ 0.1 Hz) correspondientes a los acelerogramas de las Figs. 4.8.

En las Figs. 4.10 se ilustran el sismograma observado en Caleta (sitio más cercano al hipocentro –Fig. 4.3) y los sintéticos obtenidos en las simulaciones con distinto tamaño de discretización desde 1 km hasta 0.125 km. Como se puede apreciar, al emplear una discretización más fina la señal del sismograma sintético tiene menos ruido; es decir, es una señal mas limpia. Nótese que para $\Delta h = 1$ km su espectro de amplitudes de *Fourier* Fig. 4.10(b), incluye un máximo en 0.5 Hz, el cual desaparece para las discretizaciones $\Delta h = 0.5, 0.25$ y 0.125 km. La conclusión es que el máximo en 0.5 Hz para un $\Delta h = 1$ km es ruido numérico. Por lo tanto, todos los sintéticos mostrados en las figuras subsecuentes serán con la máxima discretización posible 0.125 km.

La siguientes Figs. 4.11, muestran los sismogramas observados y sintéticos de velocidades tanto en el dominio del tiempo como en el de la frecuencia para el sitio Caleta en la dirección norte-sur, N-S, la forma y amplitud del sismograma sintético es buena aproximación al registrado. En el dominio de la frecuencia la energía disminuye drásticamente después de 0.25 Hz, Fig. 4.11(b) lo cual puede mejorarse si se refinan las características de la fuente sintética Fig. 4.3, empleada.

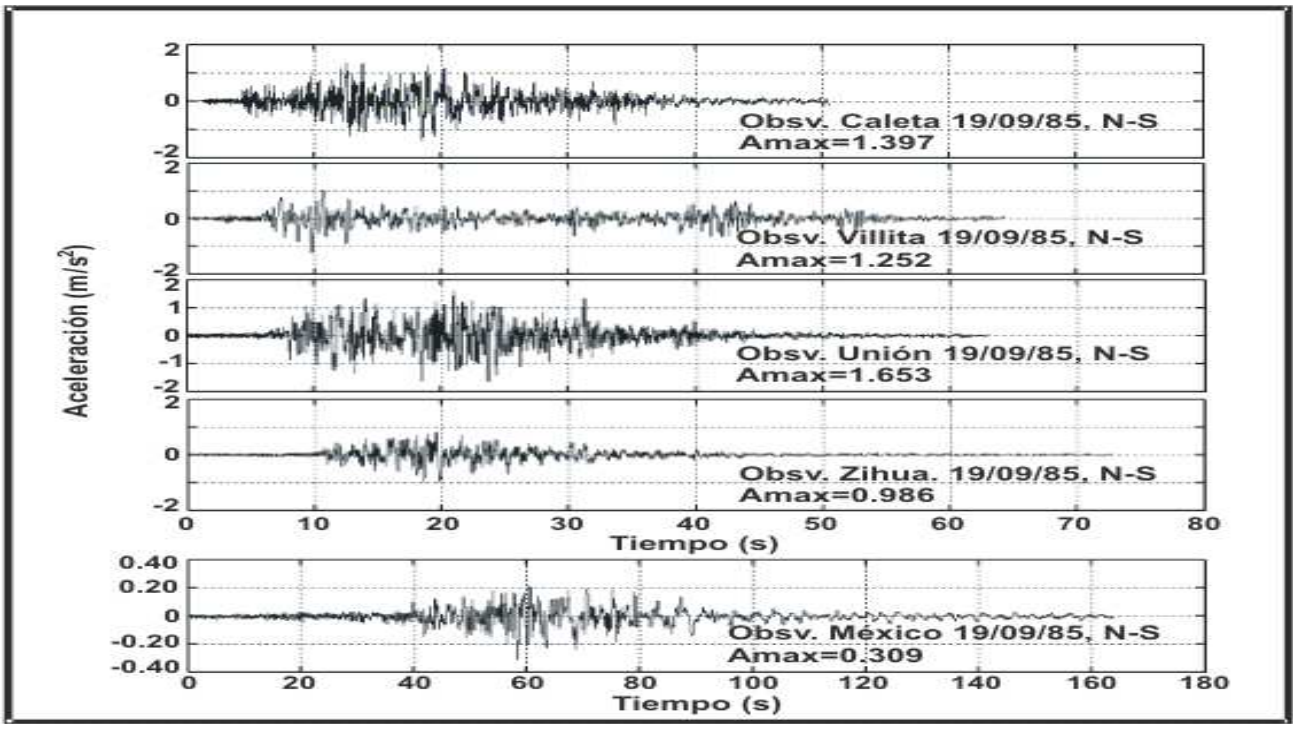
Las Figs. 4.12(a) y 4.12(b) corresponden a los sismogramas observados y sintéticos de velocidad en el dominio del tiempo y de la frecuencia, respectivamente, para el sitio Villita, con una orientación N-S e ilustran una buena forma y amplitud de los sintéticos con respecto a los observados en el dominio del tiempo y hasta el segundo 15, aproximadamente, de la simulación. Dicha aproximación es mejor apreciada en el dominio de la frecuencia Fig. 4.12(b).

Los sismogramas observados y sintéticos en el dominio del tiempo y la frecuencia para el sitio Unión se ilustran en la Figs. 4.13(a) y 4.13(b), respectivamente, con una orientación N-S. Tanto la forma como las amplitudes en ambos dominios es aceptable. En el dominio de la frecuencia el sintético es parecido al observado hasta 0.4 Hz Fig. 4.13(b).

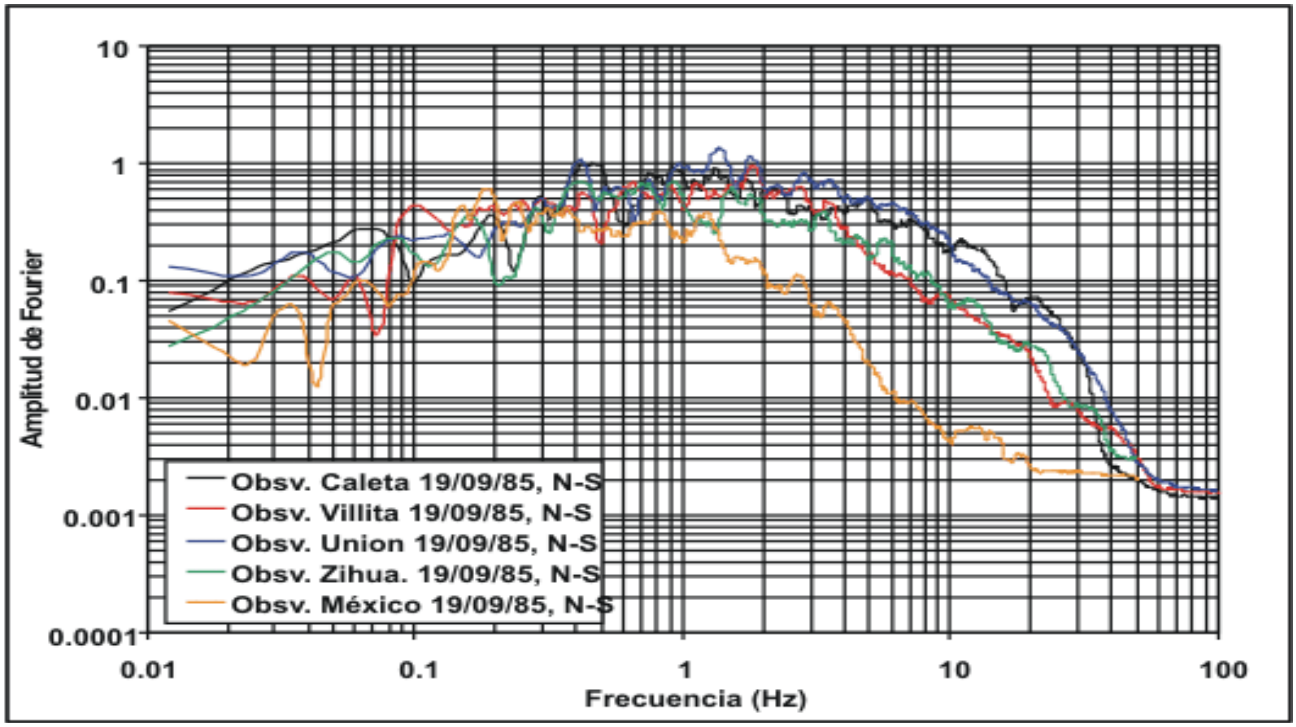
En las Figs. 4.14(a) y 4.14(b) se pueden observar los sismogramas observados y sintéticos en el dominio del tiempo y de la frecuencia para el sitio Zihuatanejo, respectivamente, en la dirección N-S. La forma en el dominio del tiempo es parecido, lo cual se aprecia mejor en el dominio de la frecuencia 4.14(b).

Los sismogramas observados y sintéticos en el dominio del tiempo y la frecuencia para el sitio Tacubaya (México, D.F.) con orientación N-S están ilustrados en las Figs. 4.15(a) y 4.15(b), respectivamente. La comparativa es buena y es mejor apreciada en el dominio de la frecuencia.

Las Figs. 4.16(a) y 4.16(b) muestran los sismogramas observados y sintéticos para el sitio Tacubaya (México, D.F.) en el dominio del tiempo la primera y en el dominio de la frecuencia la segunda, ambas con una orientación vertical. La aproximación es buena en ambos dominios tanto en la amplitud como en la forma de las señales.

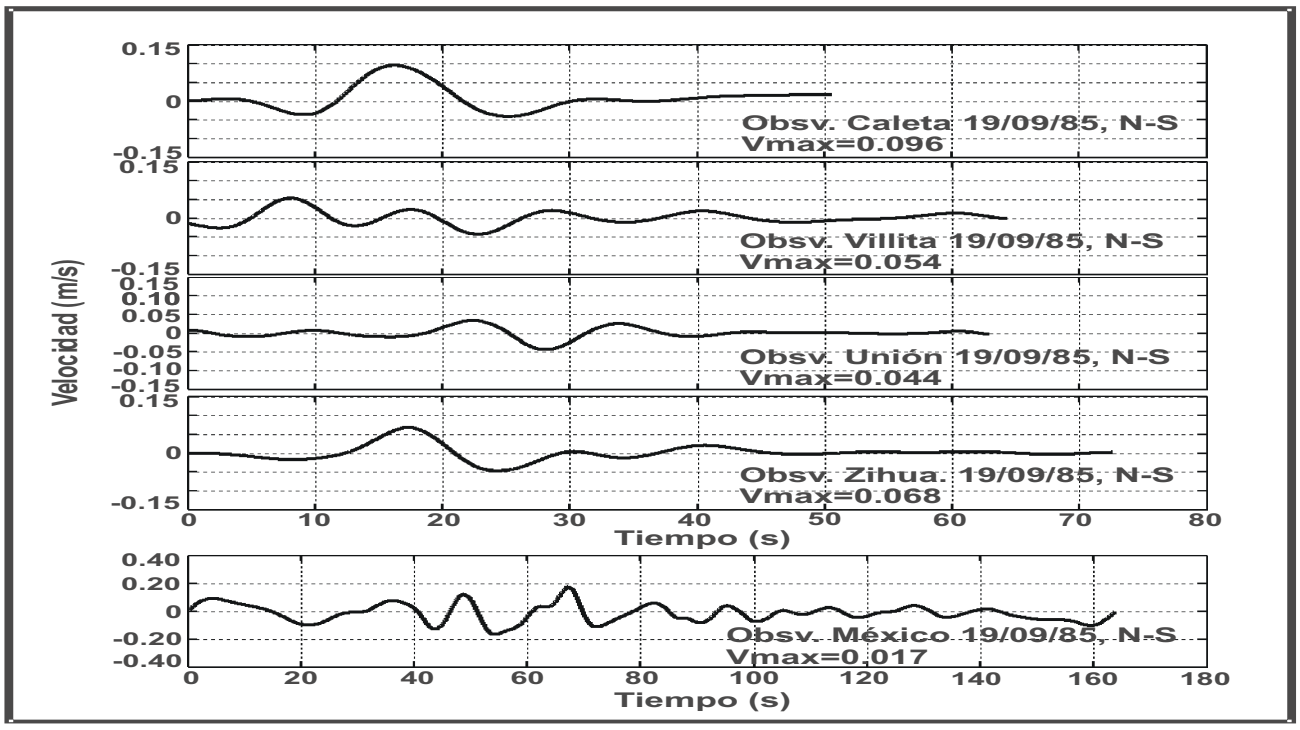


(a) Sismogramas (aceleraciones) observados en la dirección N-S en los sitios (de arriba hacia abajo): Caleta, Villita, Unión, Zihuatanejo y Tacubaya (México, D.F.).

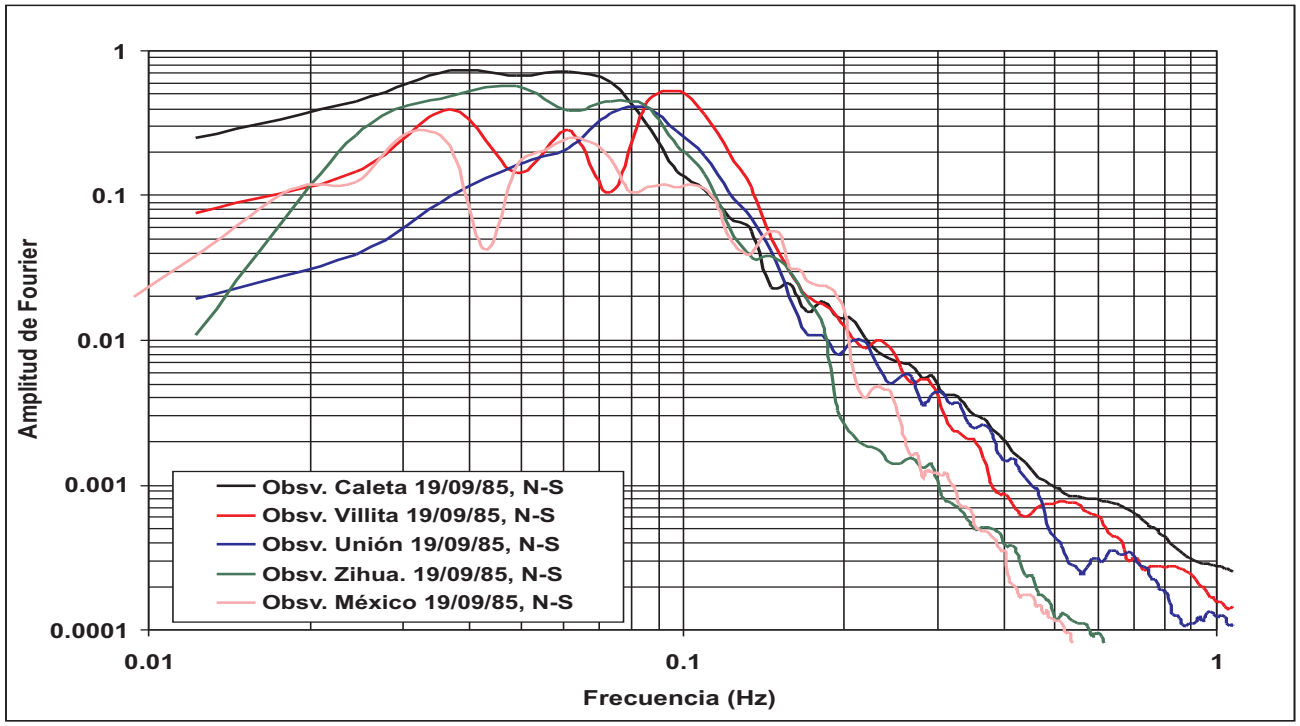


(b) Espectros de amplitudes de *Fourier* de los sismogramas (aceleraciones) observados en la dirección N-S en los sitios: Caleta, Villita, Unión, Zihuatanejo y Tacubaya (México, D.F.).

Figura 4.8: Figuras que ilustran los sismogramas de las aceleraciones en el dominio del tiempo 4.8(a) y la frecuencia 4.8(b), respectivamente, para el macrosismo de 1985.

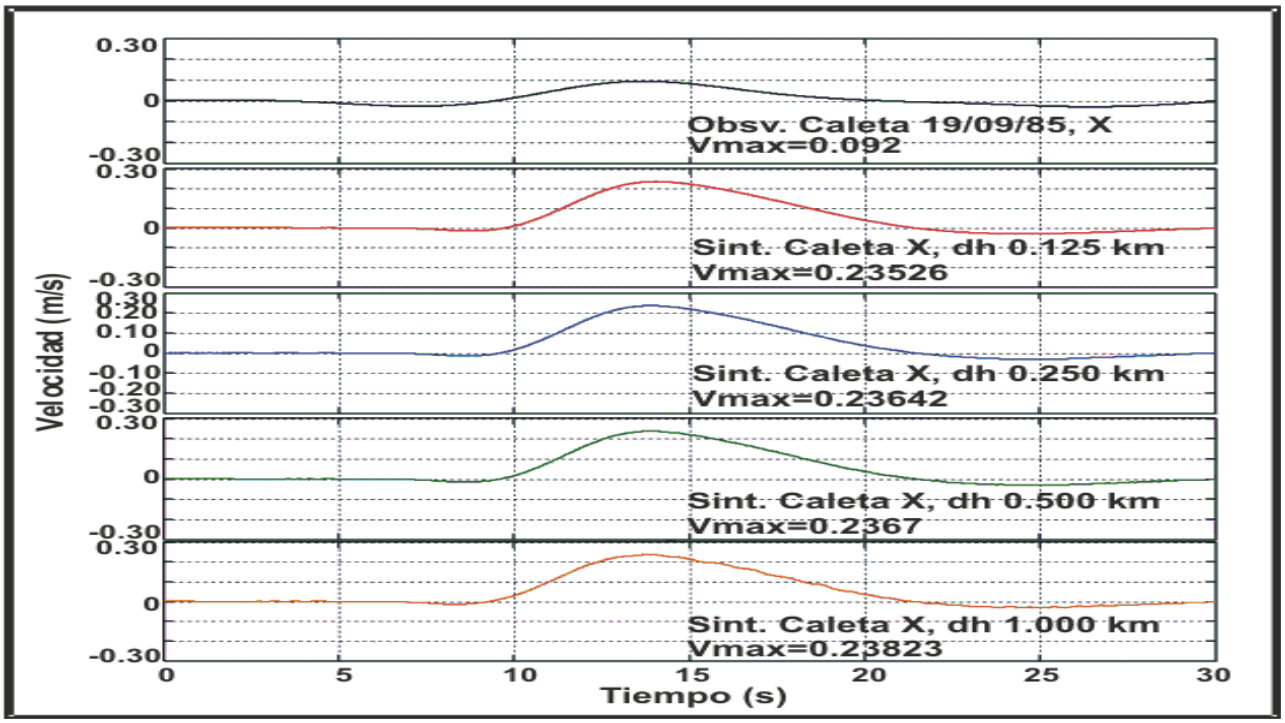


(a) Sismogramas (velocidades) observados en baja frecuencia (0.25 a 0.1Hz) en la dirección N-S en los sitios (de arriba hacia abajo): Caleta, Villita, Unión, Zihuatanejo y Tacubaya (México, D.F.).

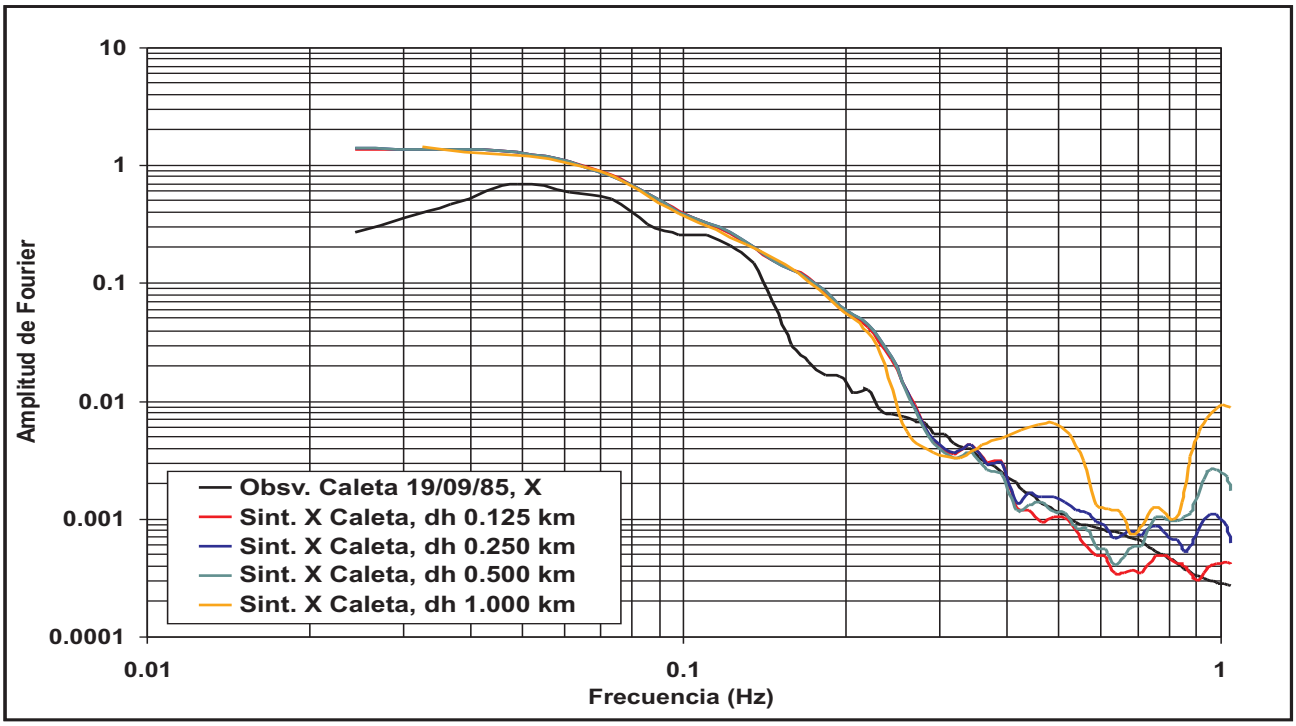


(b) Espectros de amplitudes de *Fourier* de los sismogramas (velocidades) observados en baja frecuencia (0.25 a 0.1 Hz) en la dirección N-S en los sitios: Caleta, Villita, Unión, Zihuatanejo y Tacubaya (México, D.F.).

Figura 4.9: Figuras que ilustran los sismogramas de las velocidades en el dominio del tiempo 4.9(a) y la frecuencia 4.9(b), respectivamente, para el macrosismo de 1985.

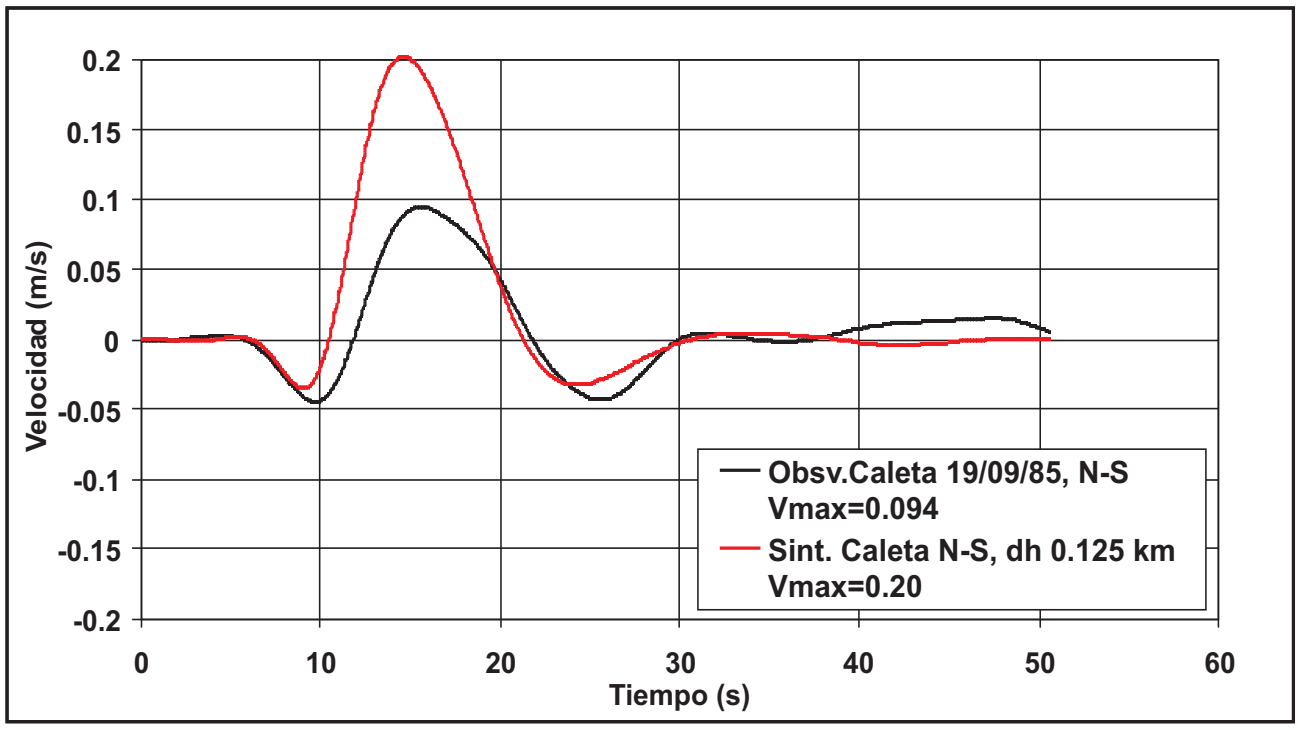


(a) Sismogramas (velocidades)) observados y sintéticos en el sitio Caleta en la dirección paralela a la trinchera Mesoamérica (*X*) con discretizaciones espaciales $\Delta h = 0.125, 0.250, 0.5$ y 1 km.

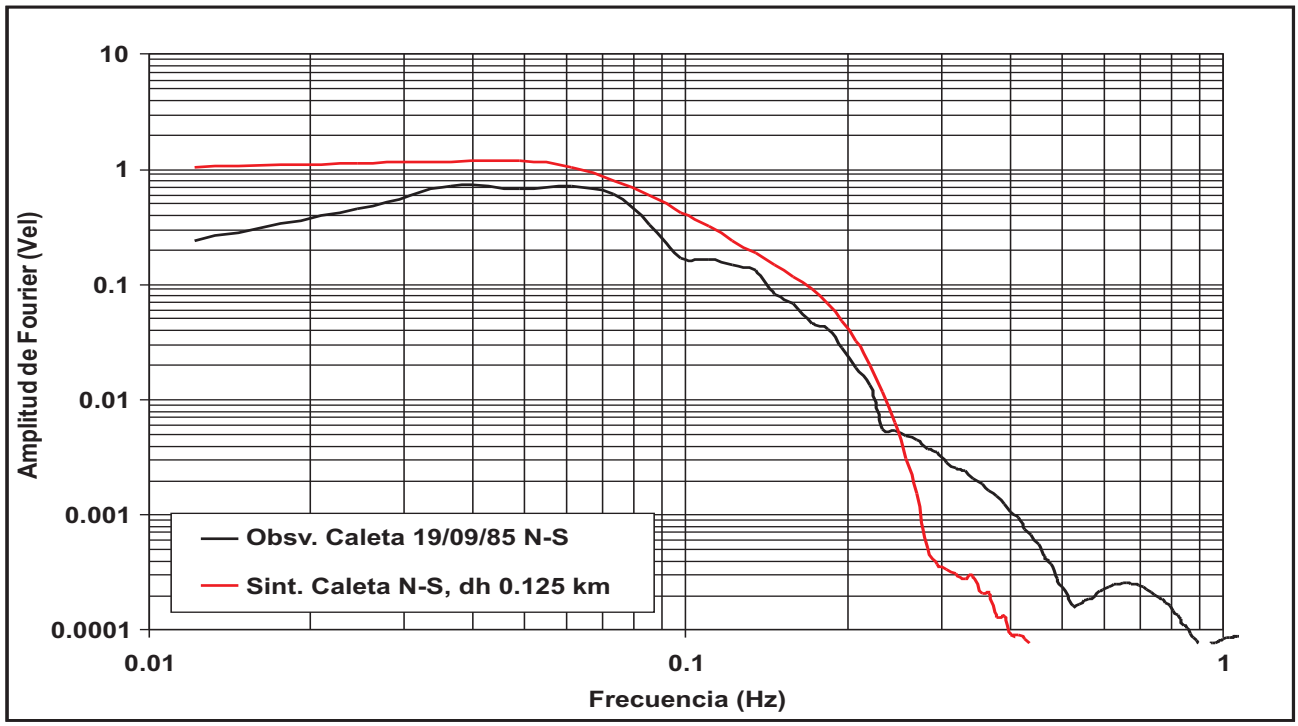


(b) Espectros de amplitudes de *Fourier* de los sismogramas observados y sintéticos en el sitio Caleta en la dirección paralela a la trinchera Mesoamérica (*X*) con discretizaciones espaciales $\Delta h = 0.125, 0.250, 0.5$ y 1 km.

Figura 4.10: Figuras que ilustran los sismogramas de las velocidades en el dominio del tiempo 4.10(a) y la frecuencia 4.10(b), respectivamente, para el macrosismo de 1985.

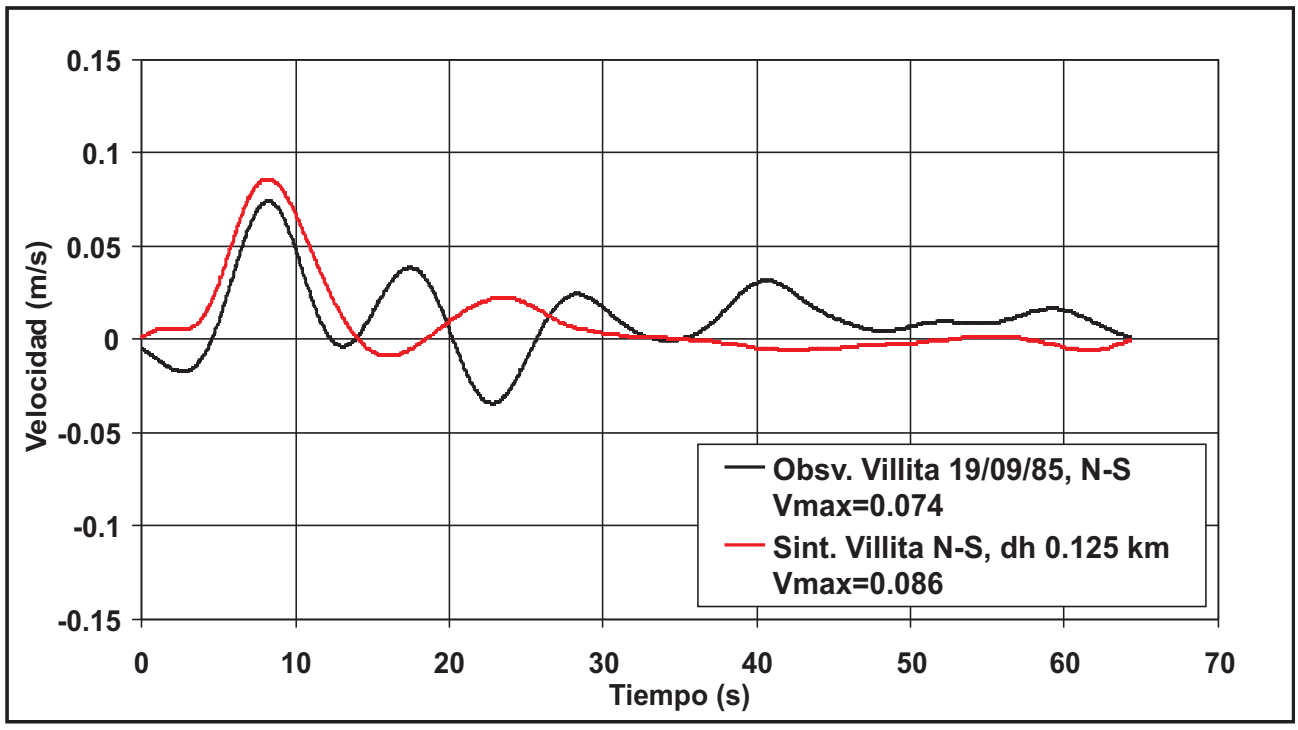


(a) Sismogramas (velocidades) observado y sintético en el sitio Caleta en la dirección N-S con una discretización espacial $\Delta h = 0.125$.

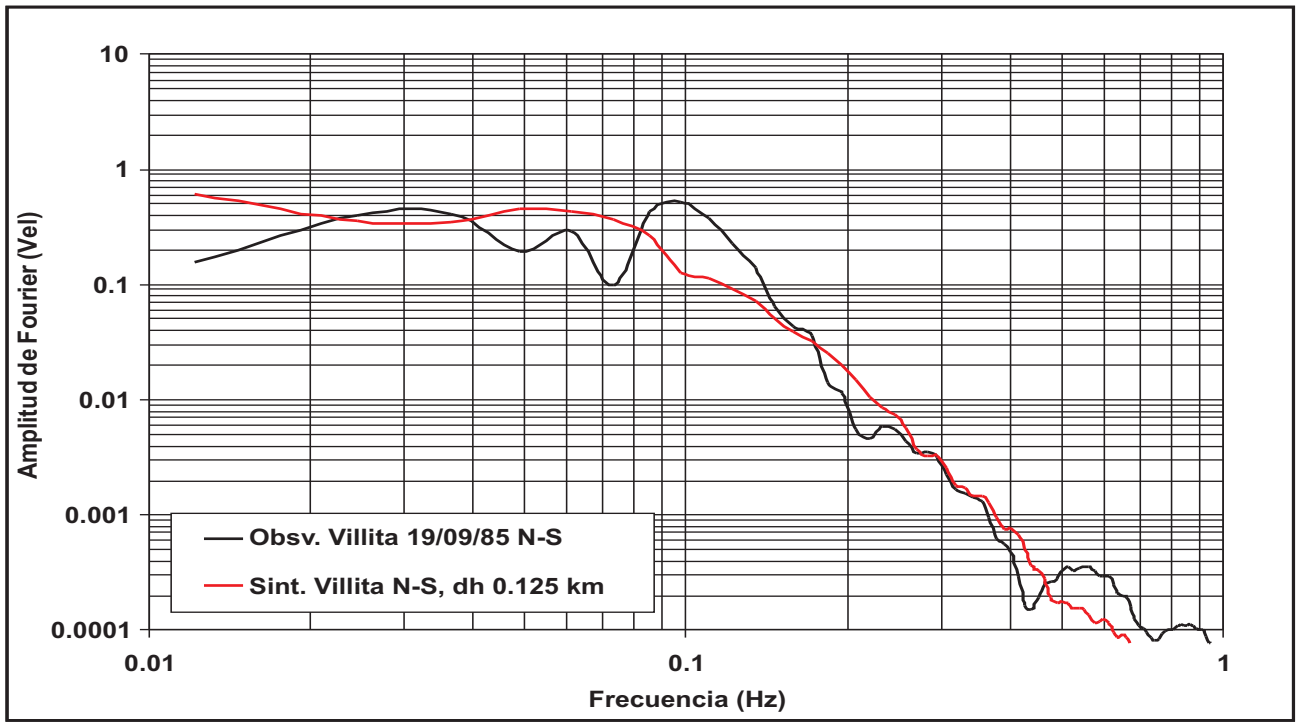


(b) Espectros de amplitudes de *Fourier* de los sismogramas (velocidades) observado y sintético en el sitio Caleta en la dirección N-S con una discretización espacial $\Delta h = 0.125$.

Figura 4.11: Figuras que ilustran los sismogramas de las velocidades en el dominio del tiempo 4.11(a) y la frecuencia 4.11(b), respectivamente, para el macrosismo de 1985.

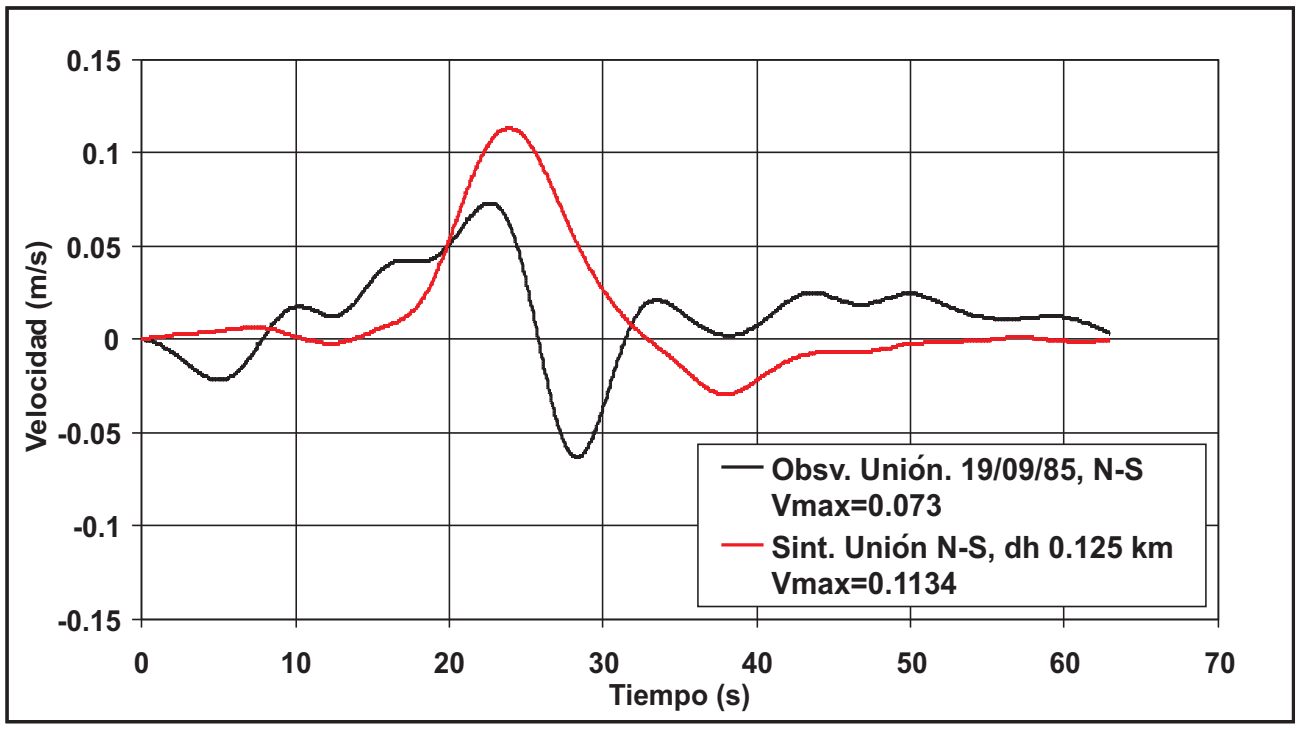


(a) Sismogramas (velocidades) observado y sintético en el sitio Villita en la dirección N-S con una discretización espacial $\Delta h = 0.125$.

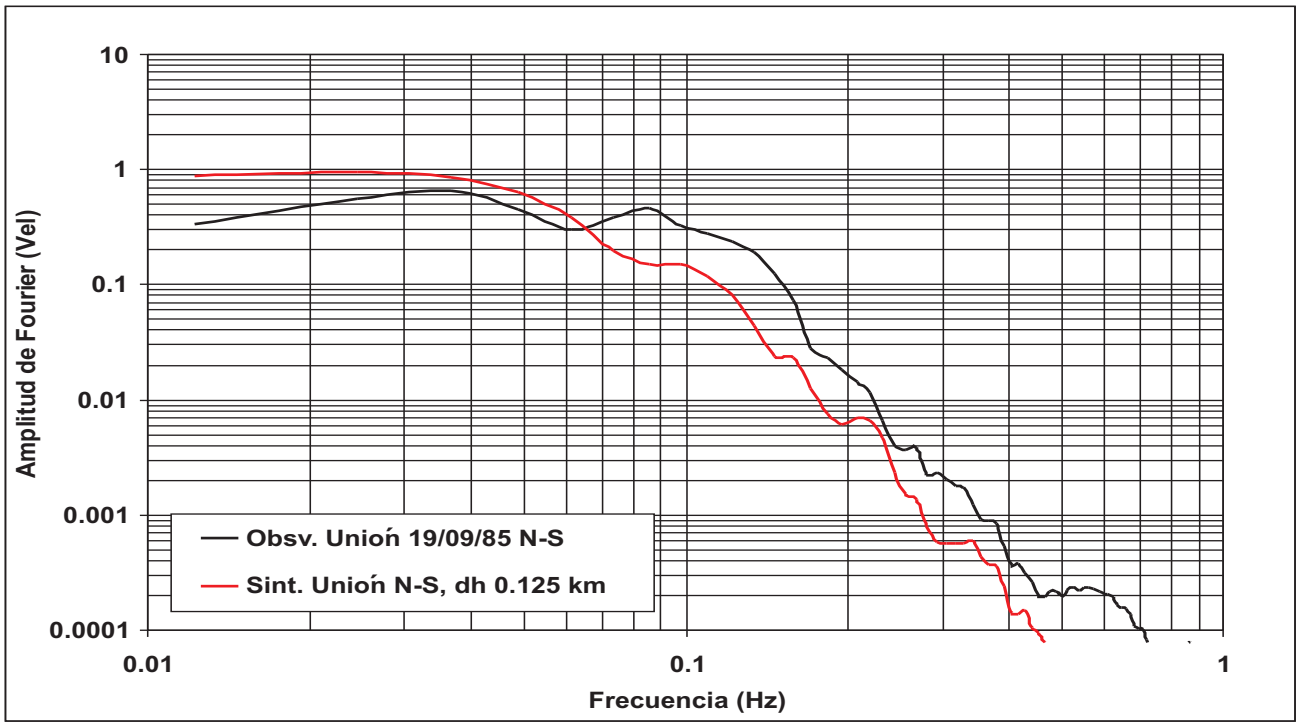


(b) Espectros de amplitudes de *Fourier* de los sismogramas (velocidades) observado y sintético en el sitio Villita en la dirección N-S con una discretización espacial $\Delta h = 0.125$.

Figura 4.12: Figuras que ilustran los sismogramas de las velocidades en el dominio del tiempo 4.12(a) y la frecuencia 4.12(b), respectivamente, para el macrosismo de 1985.

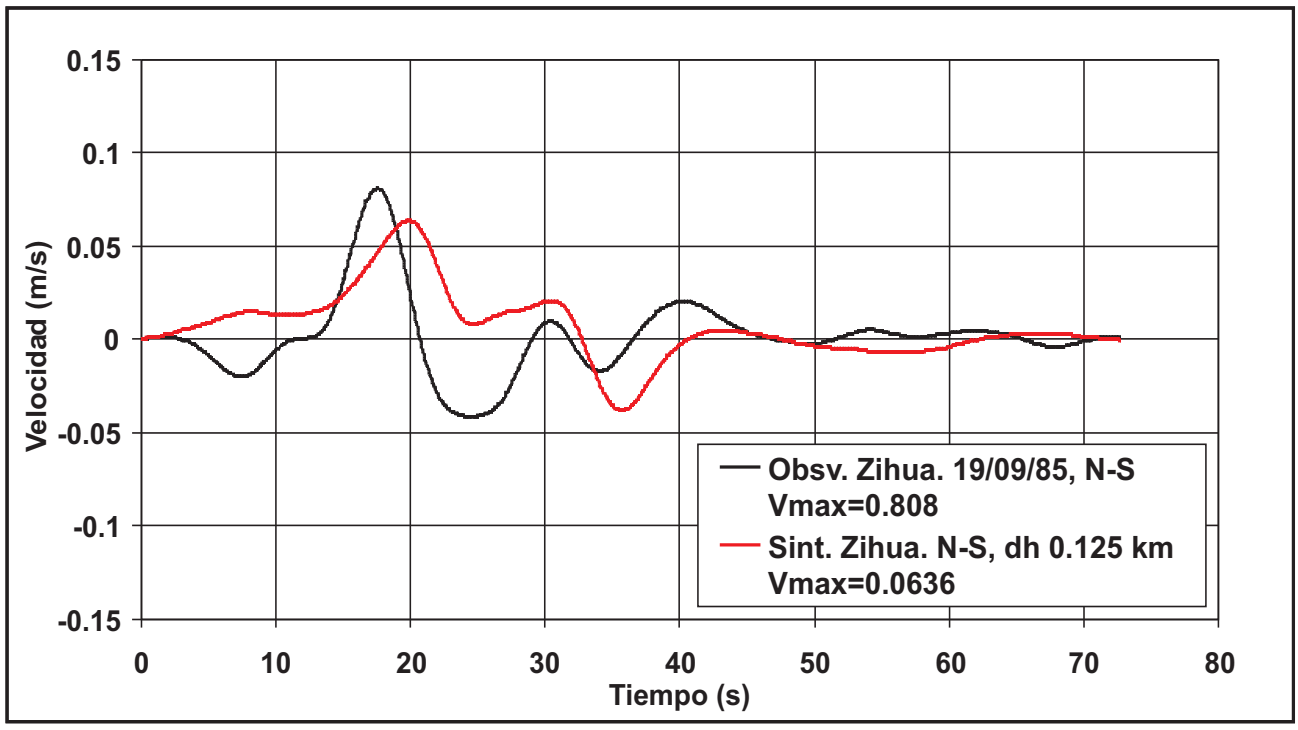


(a) Sismogramas (velocidades) observado y sintético en el sitio Unión en la dirección N-S con una discretización espacial $\Delta h = 0.125$.

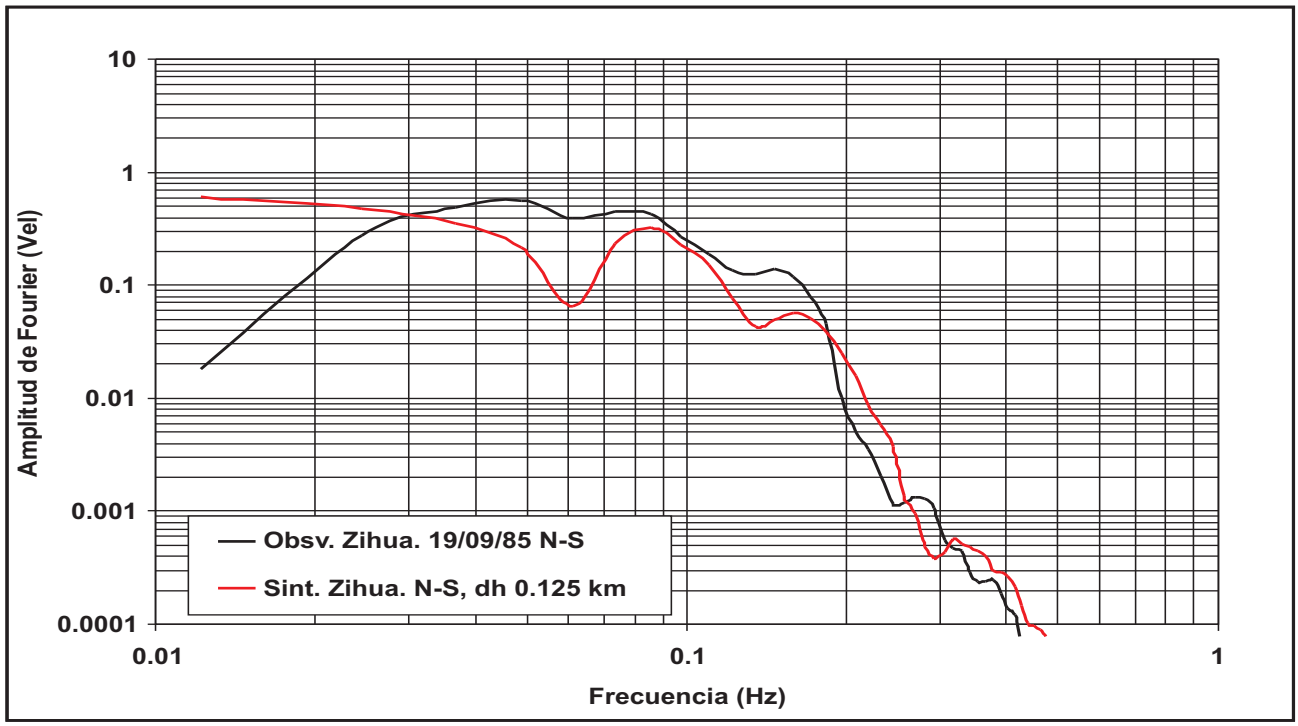


(b) Espectros de amplitudes de *Fourier* de los sismogramas (velocidades) observado y sintético en el sitio Unión en la dirección N-S con una discretización espacial $\Delta h = 0.125$.

Figura 4.13: Figuras que ilustran los sismogramas de las velocidades en el dominio del tiempo 4.13(a) y la frecuencia 4.13(b), respectivamente, para el macrosismo de 1985.

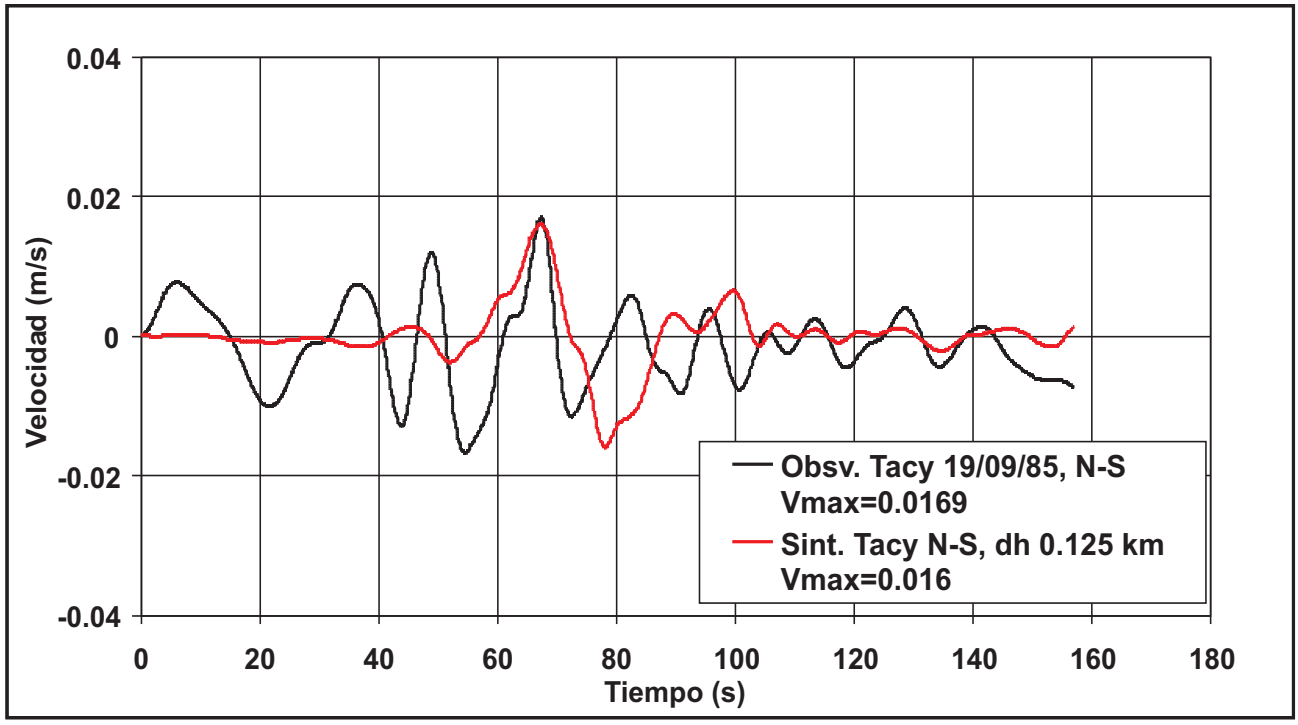


(a) Sismogramas (velocidades) observado y sintético en el sitio Zihuatanejo en la dirección N-S con una discretización espacial, $\Delta h = 0.125$.

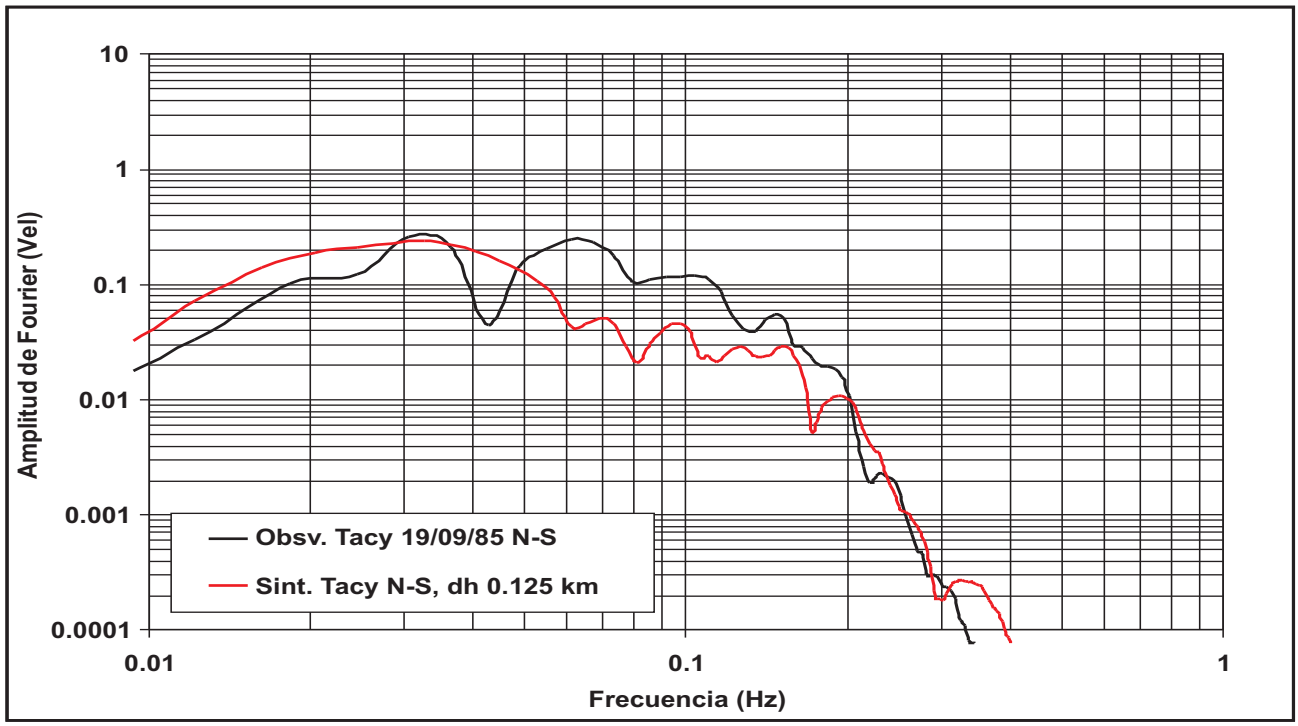


(b) Espectros de amplitudes de *Fourier* de los sismogramas (velocidades) observado y sintético en el sitio Zihuatanejo en la dirección N-S con una discretización espacial $\Delta h = 0.125$.

Figura 4.14: Figuras que ilustran los sismogramas de las velocidades en el dominio del tiempo 4.14(a) y la frecuencia 4.14(b), respectivamente, para el macrosismo de 1985.

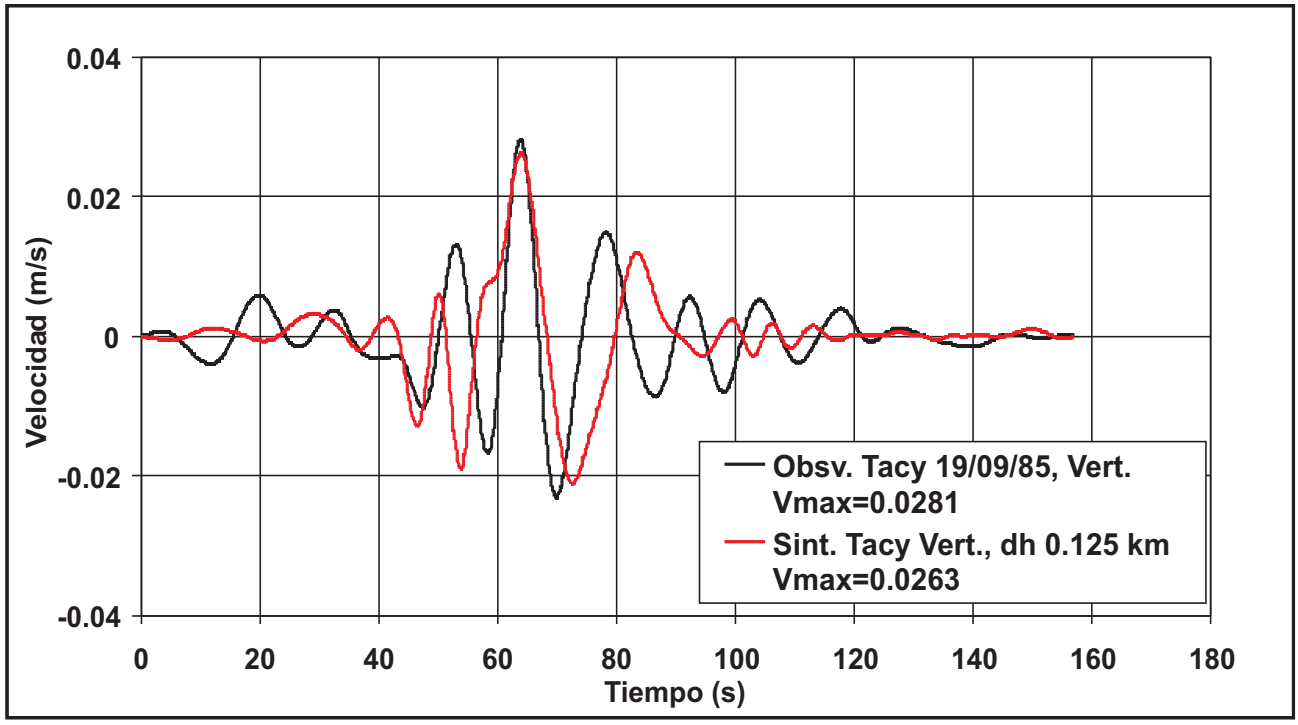


(a) Sismogramas (velocidades) observado y sintético en el sitio Tacubaya (México, D.F.) en la dirección N-S con una discretización espacial $\Delta h = 0.125$.

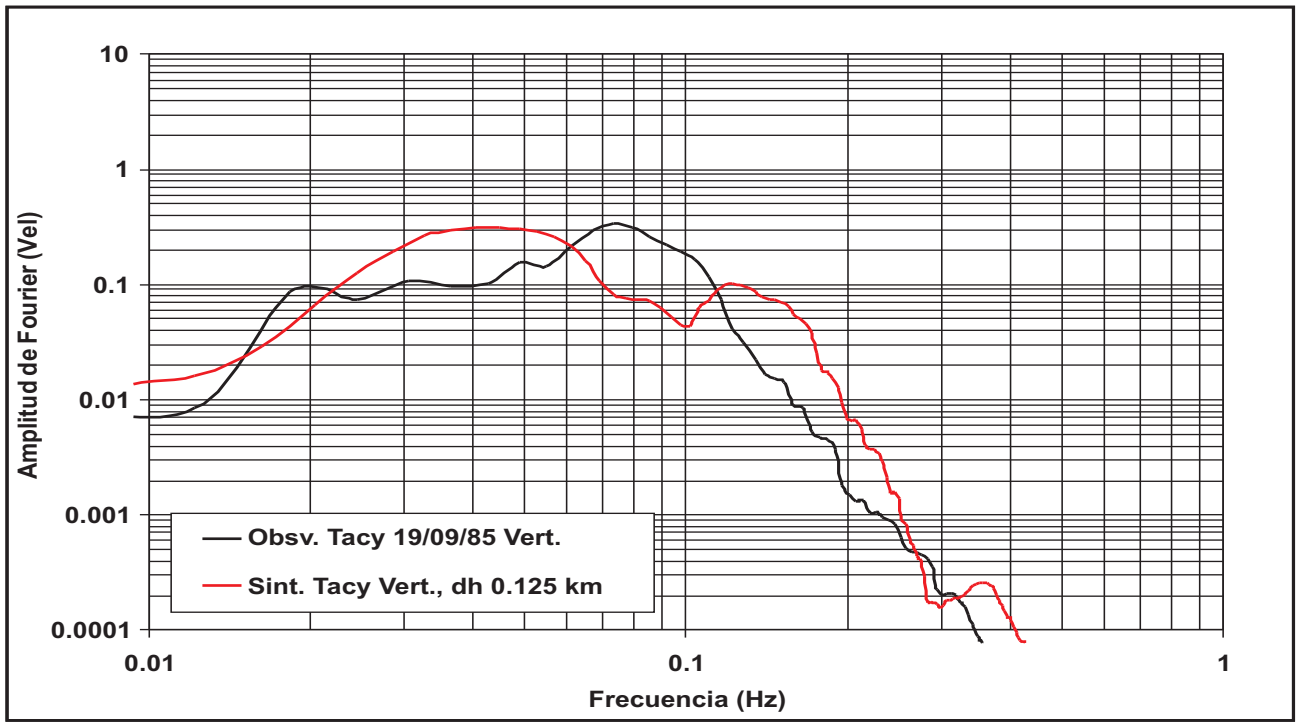


(b) Espectros de amplitudes de *Fourier* de los sismogramas (velocidades) observado y sintético en el sitio Tacubaya (México, D.F.) en la dirección N-S con una discretización espacial $\Delta h = 0.125$.

Figura 4.15: Figuras que ilustran los sismogramas de las velocidades en el dominio del tiempo 4.15(b) y la frecuencia 4.15(b), respectivamente, para el macrosismo de 1985.



(a) Sismogramas (velocidades) observado y sintético en el sitio Tacubaya (México, D.F.) en la dirección vertical con una discretización espacial $\Delta h = 0.125$.



(b) Espectros de amplitudes de *Fourier* de los sismogramas (velocidades) observado y sintético en el sitio Tacubaya (México, D.F.) en la dirección vertical con una discretización espacial $\Delta h = 0.125$.

Figura 4.16: Figuras que ilustran los sismogramas de las velocidades en el dominio del tiempo 4.16(a) y la frecuencia 4.16(b), respectivamente, para el macrosismo de 1985.

Las Figs. 4.17, 4.18 y 4.19 representan imágenes extraídas de una visualización en 2D de las velocidades en X, Y y Z, respectivamente en dos tiempos distintos, en $t_1 = 48$ s y $t_2 = 120$ s. Se muestran la zona de la fuente en el rectángulo interno y su dimensión 180×140 km, así como el sitio Caleta, el punto más cercano al hipocentro; el dominio completo de 500×600 km en planta y la ubicación de la Ciudad de México. Los tonos más intensos muestran las ondas de magnitud máxima del sismo, que van desde -0.2 a 0.2, y también nos permite apreciar ondas sísmicas con una longitud de onda grande.

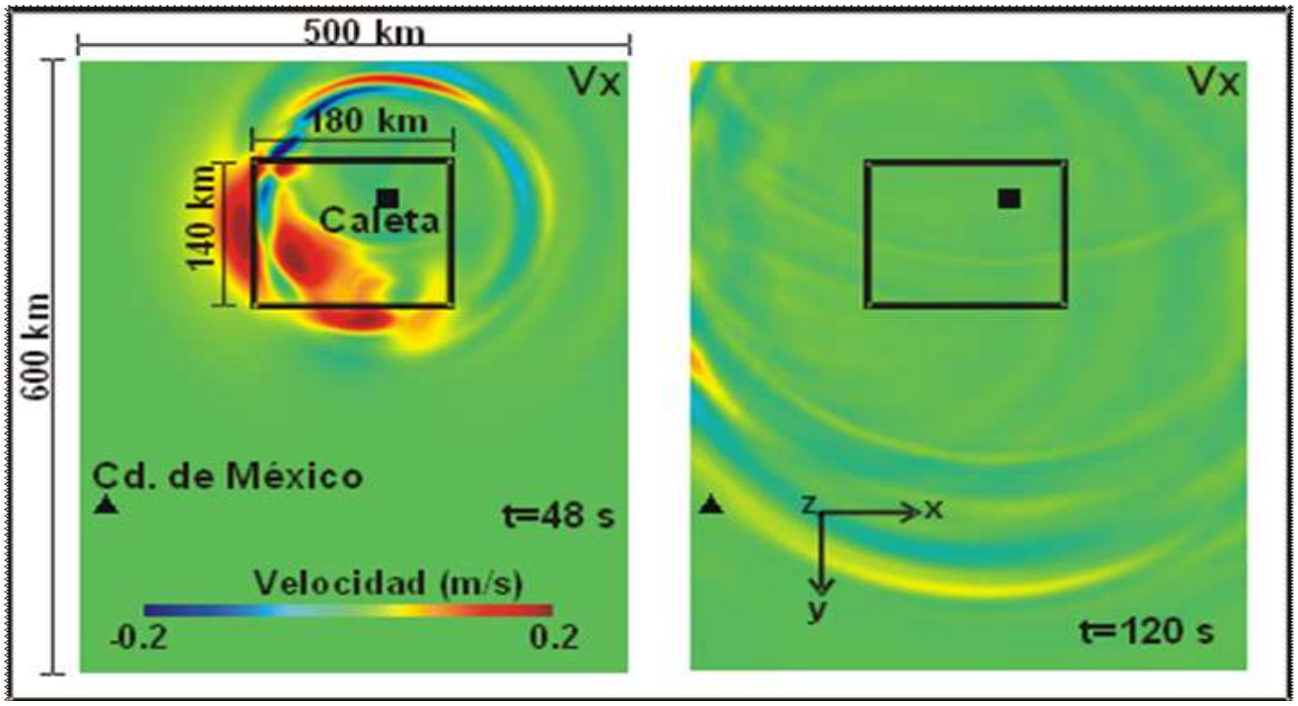


Figura 4.17: Simulación en 2D que muestra los campos de velocidades X sintéticos en la superficie del dominio físico para una discretización espacial $\Delta h = 1$ km, en los tiempos $t_1 = 48s$ y $t_2 = 120s$

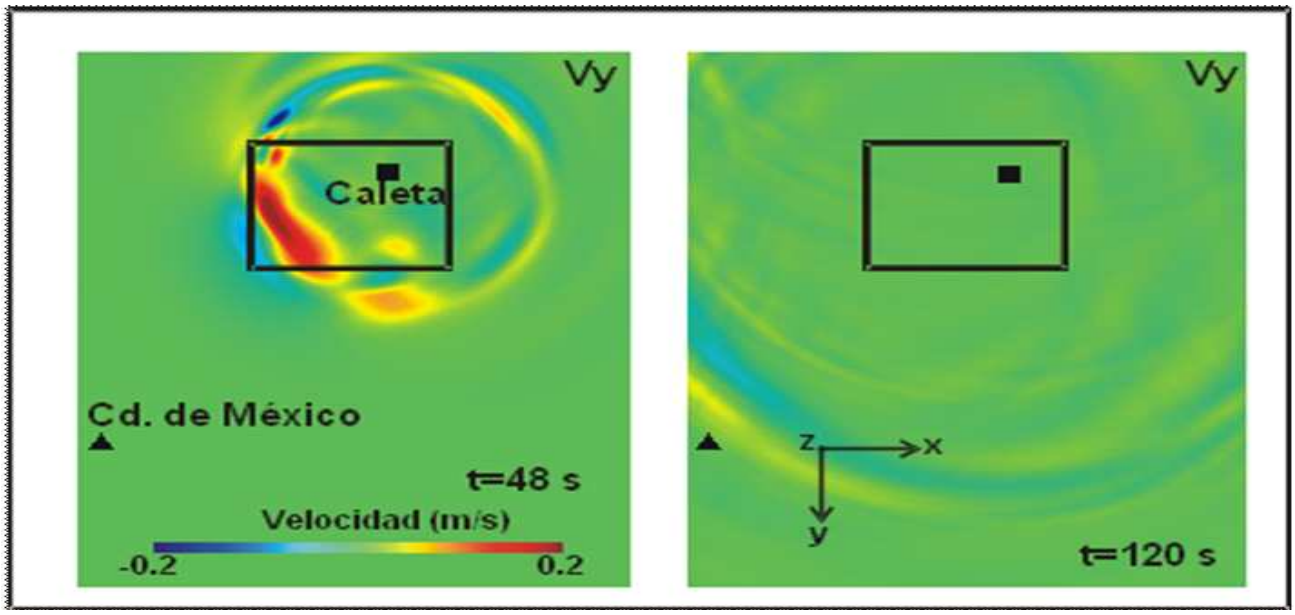


Figura 4.18: Simulación en 2D que ilustra los campos de velocidades Y sintéticos en la superficie del dominio físico para una discretización espacial $\Delta h = 1$ km, en los tiempos $t_1 = 48s$ y $t_2 = 120s$

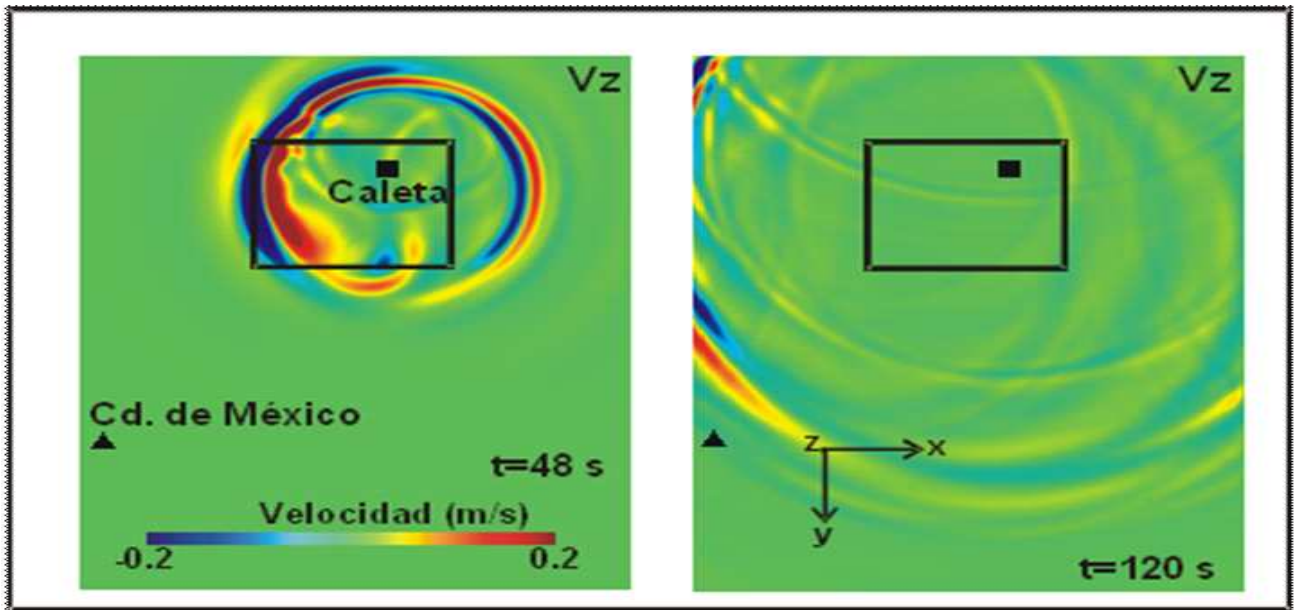


Figura 4.19: Simulación en 2D que representan los campos de velocidades Z sintéticos en la superficie del dominio físico para una discretización espacial $\Delta h = 1$ km, en los tiempos $t_1 = 48s$ y $t_2 = 120s$

4.4. CONCLUSIONES DEL CAPÍTULO.

Se listan, enseguida, las conclusiones más relevantes de este capítulo:

1. Actualmente es prácticamente imposible hacer ingeniería e investigación excluyendo las simulaciones numéricas, que involucran las siguientes fases: una formulación matemática del problema de interés, un método numérico y su implementación computacional que resuelva la descripción matemática propuesta.

2. El rendimiento alcanzado, tanto en el factor de aceleración *Sp* como en el de eficiencia *E* en el primer caso paralelizado *Sp* = 16.47 y E = 1.03, es superior al ideal 16 y 1, respectivamente, al emplear 16 procesadores y una partición *2d*, porque la cantidad de comunicaciones realizadas 48 y el tamaño de éstas 70 MB, aproximadamente, en cada iteración son pequeñas en comparación al ancho de banda (10 Gbits/s) de *KanBalam* y el tamaño de los arreglos es tal, en particular en la cantidad de datos en $Z = 62$ por proceso, que se hace un uso eficiente de la memoria *cache*.

3. Al realizar la paralelización más agresiva y escalable en *3d* y utilizar 128 procesadores se obtuvo un factor de aceleración de *Sp* = 120.8 y una eficiencia de E = 0.94. Este rendimiento, muy cercano al ideal, es debido a que la cantidad de comunicaciones 592 con un tamaño de total de datos, por cada iteración, de 390 MB, empieza a afectar la red de intercomunicación *Infiniband* de *KanBalam*, al igual que el tamaño de los datos en $Z = 124$, por proceso, teniendo que acceder a los datos de la *RAM* y no de la memoria *cache*.

4. Por último, al emplear 1,024 procesadores el desempeño es aceptable con un factor de aceleración *Sp* = 670 y una eficiencia $E = 0.65$, por la gran cantidad de comunicaciones realizadas 3,584 y el volumen de éstas 3 GB, por iteración, las cuales saturan la red *Infiniband* de comunicación de datos de *KanBalam*, además de la utilización de forma exhaustiva de la memoria *cache* por la cantidad de datos en la partición en $Z = 248$, por proceso, ocasionando un intercambio constante de los datos almacenados en ésta.

5. Con base en las comparaciones de los sismogramas sintéticos *vs* los observados en las gráficas podemos decir que son satisfactorias la aproximaciones tanto en la amplitud como en la forma de los sismogramas en el dominio del tiempo y en el de la frecuencia.

Capítulo 5

CONCLUSIONES.

A continuación se mencionan las conclusiones generales de esta tesis:

1. La teoría de la elastodinámica rige la física del problema modelado en esta tesis, que es la propagación de las ondas sísmicas considerando medios elásticos, homogéneos e isótropos. Este problema, como la mayoría de los problemas de la física se modelan mediante sistemas de ecuaciones diferenciales parciales, cuya solución requiere de la aplicación de métodos numéricos y de la generación de algoritmos computacionales eficientes. El método numérico de diferencias finitas alternadas en tres dimensiones de segundo orden en el tiempo y de cuarto orden en el espacio, es uno de ellos el cual permite, utilizando diferencias finitas centrales, sin duplicar la extensión espacial de los operadores proporcionar mayor precisión, y por lo tanto, utilizar en forma óptima los recursos de cómputo disponibles.

2. Una forma de optimizar un algoritmo serial es su paralelización, la cual permite aumentar la resolución del problema acorde con la disminución de las discretizaciones espacial y temporal, así como disminuir los tiempos de ejecución de los programas, permitiendo realizar más simulaciones en menos tiempo. El paralelismo de datos en *3d* y la biblioteca de envío de mensajes *MPI* es lo que se utilizó en paralelizar el código serial de propagación de ondas sísmicas en 3D, **DFA3D** en este trabajo, ya que este paradigma es el más escalable y ampliamente utilizado porque permite lograr cabalmente los objetivos básicos de la paralelización.

3. Se aplicó al programa de interés las precondiciones de *Pancake* y la metodología de *Foster* para efectuar su paralelización, así como la biblioteca de envío de mensajes *MPI*, la cual es escalable, portátil y eficiente, y el estándar *de facto* en la programación paralela.

4. El programa se implementó en la supercomputadora *KanBalam* de la *UNAM*, que es un cúmulo con 684 procesadores *Opteron* duales -1,368 núcleos- organizados en 342 nodos, cada uno de ellos con sistema operativo y memoria independiente 8 GB de *RAM* - aproximadamente 3 TB de memoria total; interconectados entre sí por una red de baja latencia, 13×10^{-6}s y un ancho de banda de 10 Gigabits/s –de la red *Infiniband*.

5. El algoritmo de **DFA3D** se utilizó para obtener sismogramas sintéticos del sismo del 19/09/1985 de magnitud $Ms = 8.1$, en un dominio físico en las direcciones X, Y y Z de

$500 \times 600 \times 124$ km^3. Este dominio fue discretizado espacialmente con un $\Delta x = \Delta y = \Delta z = \Delta h$ de 1, 0.5, 0.25 y 0.125 km. Por lo cual, se generaron tamaños discretizados del problema N_x, N_y, N_z de $500 \times 600 \times 124$, $1000 \times 1200 \times 248$, $2000 \times 2400 \times 496$, $4000 \times 4800 \times 992$, respectivamente. Los cuales fueron asignados a $n_x \times n_y \times n_z = M$ procesadores de *KanBalam*

6. Entre las métricas de rendimiento más utilizadas para la evaluación de los programas paralelos se pueden mencionar al factor de aceleración *Sp* (cociente del tiempo de ejecución serial entre el paralelo) y al de eficiencia *E* (igual a *Sp* por la cantidad de procesadores utilizados) empleados por la ley de *Amdahl*. Ésta proporciona una aproximación del rendimiento del programa, con base en el porcentaje del código paralelizable. Para el algoritmo de **DFA3D** dicha ley no es suficiente, ya que no involucra las comunicaciones, por lo cual se emplea una expresión modificada de la ley de *Amdahl* que incluye las comunicaciones intensivas requeridas por el programa. Además de las citadas métricas en este trabajo se utilizó la granularidad (cantidad de cómputo con relación a las comunicaciones) N_i/n_i con $i = x, y, z$ para calificar el código paralelizado.

7. Con el código paralelizado de **DFA3D** se efectuó un experimento computacional en el que, se utilizaron $n_x \times n_y \times n_z$ procesadores de: 1,4,4; 8,4,4 y 16,16,4 para obtener los sismogramas sintéticos correspondientes a $\Delta h = 0.5, 0.25$ y 0.125 km, respectivamente.

   - Para $M = 1 \times 4 \times 4 = 16$, $Sp = 16.47$ y $E = 1.03$ *vs* al ideal 16 y 1, respectivamente. La granularidad, en este caso, es gruesa ya que la cantidad de comunicaciones realizadas 48 y el tamaño de éstas 70 MB, en cada iteración, son pequeñas en comparación al ancho de banda de *KanBalam* y al uso eficiente de la memoria *cache*.
   - Para $M = 8 \times 4 \times 4 = 128$, $Sp = 120.8$, y $E = 0.94$ *vs* al ideal 128 y 1, respectivamente. Este rendimiento, muy cercano al ideal, es debido a que la granularidad es intermedia y la cantidad de comunicaciones 592 y tamaño total de datos 390 MB, por cada iteración, empieza a afectar la red de intercomunicación *Infiniband* de *KanBalam* y hay más accesos a la memoria *cache*.
   - Para $M = 16 \times 16 \times 4 = 1,024$, $Sp = 670$, y $E = 0.65$ *vs* al ideal 1,024 y 1, respectivamente. La granularidad es fina, por lo cual tenemos gran cantidad de comunicaciones realizadas 3,584 y el volumen de éstas 3 GB, por iteración, las cuales saturan la red *Infiniband* de *KanBalam* .

8. Con base en las comparaciones de los sismogramas sintéticos *vs* los observados podemos afirmar que son satisfactorias tanto en el dominio del tiempo y en el de la frecuencia. Especialmente para las discretizaciones espaciales de 0.25 y 0.125km.

Finalmente, como trabajo futuro se sugieren:

- Refinar las características de la fuente sísmica, con el propósito que contenga frecuencias superiores a 1 Hz.

- Realizar *checkpoint*; es decir, que un instante dado se pueda guardar el estado de la ejecución del proceso para poder reanudarlo posteriormente a partir de dicho instante, sin necesidad de reiniciarlo ni perder todo el tiempo de ejecución hasta el momento en que se realizo el *checkpoint*.
- Emplear comunicaciones no bloqueables de *MPI* para traslapar éstas con cómputo y de esta manera incrementar el factor de aceleración y la eficiencia del programa.
- Mejorar la discretización; es decir, la malla tanto en la zona de ruptura como en la de la ciudad de México.

Apéndice A

TÓPICOS DE SISMOLOGÍA.

A.1. TÉCTONICA DE PLACAS.

La corteza terrestre está conformada por varias placas rígidas de diferentes tamaños Fig. A.1. Dichas placas están en constante movimiento y sus fronteras de contacto corresponden a las zonas sísmicas del orbe.

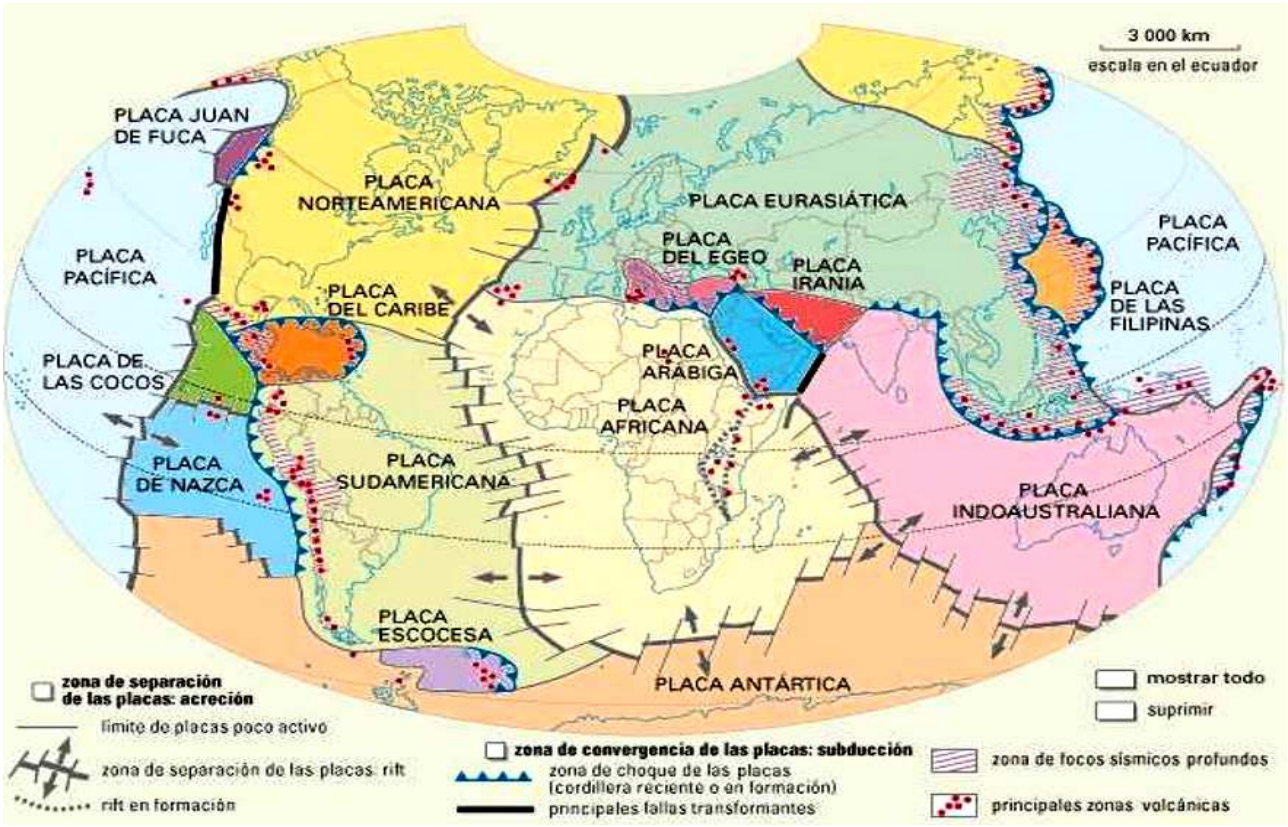


Figura A.1: Ubicación de las placas tectónicas mundiales

Existen tres tipos de contacto entre las placas Fig A.2:

1. *Transcurrente*, movimiento lateral, sin generación ni destrucción de fondo oceánico.
2. *Cordillera oceánica*, las placas son divergentes, generando fondo oceánico.
3. *Subducción*, una placa se introduce bajo la otra [58].

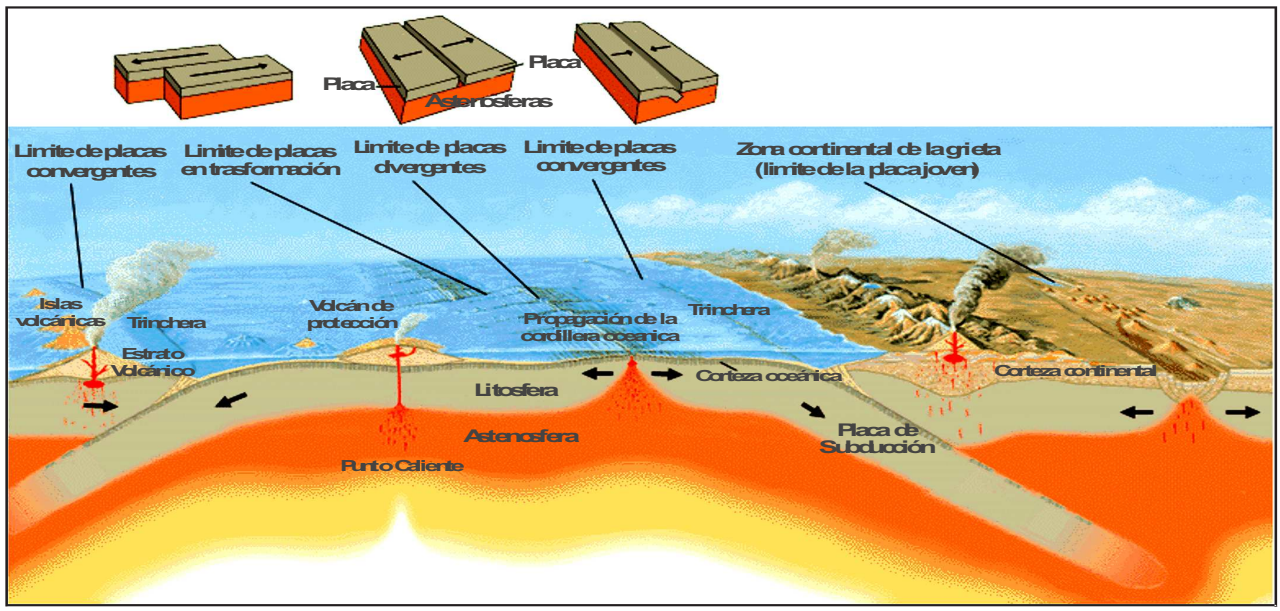


Figura A.2: En la parte superior de la figura se ilustran los tres tipos de contactos que se dan entre las placas. En la parte inferior se ilustra el fenómeno de subducción.

A esta última clasificación corresponde la mayor parte de la tectónica de la costa occidental de México, asociada al proceso de subducción de la placa de Cocos (oceánica) bajo la placa de Norte América (continental), Figs. A.2 y 1.1.

A.2. SISMOLOGÍA Y ONDAS ELÁSTICAS.

La sismología es la rama de la geofísica que se encarga de la medición y análisis de los movimientos ocurridos en el interior y la superficie de la Tierra debido a la propagación de ondas elásticas en su interior, registradas por los sismógrafos. Los sismogramas son gráficas que muestran la variación en el tiempo de las amplitudes del movimiento del terreno durante los sismos y pueden ser analógicos o digitales. Al ocurrir un sismo –que es una fractura súbita en el interior de la Tierra provocando vibraciones o movimientos del suelo, causados principalmente por la ruptura y fractura de las rocas– ondas de tipo elástico son emitidas y propagadas a través de un medio sólido-elástico, la Tierra. La fractura ocurre cuando la acumulación de energía excede la resistencia de las rocas. Los sismos ocurren, generalmente, en zonas de debilidad de

la corteza terrestre, que se conocen como fallas geológicas y son generados por la acumulación de esfuerzo en dicha corteza. El hipocentro es el punto inicial de la ruptura en el interior de la Tierra, en la cual se genera la liberación de energía o ruptura sísmica y está caracterizado por tres parámetros: latitud, longitud y profundidad. La fuente es el volumen de roca cuyo desplazamiento originó el sismo y dentro de ésta se ubica la falla; es decir, se le conoce como foco sísmico al hipocentro y a la zona de ruptura, en el cual ocurrió la liberación de energía del temblor. Epicentro es el punto proyectado sobre la superficie terrestre, ubicado justo arriba del hipocentro, y es la zona donde, generalmente, se siente con mayor intensidad el temblor; sólo requiere, para su ubicación, dos parámetros: latitud y longitud.

Las ondas elásticas se dividen en dos: ondas de cuerpo (volumétricas) y ondas superficiales. Las primeras, a su vez, son de dos tipos:

- longitudinales (onda ***P***, primaria, Fig. A.3(a)), las cuáles son ondas elásticas de compresión.
- ondas transversales (onda ***S***, secundaria, Fig. A.3(b)), que son ondas elásticas de corte.

Las segundas, las superficiales, reciben este nombre ya que el movimiento debido a estas ondas está limitado a la superficie del suelo, y conforme aumenta la profundidad, el desplazamiento provocado por éstas disminuye. Éstas, también, son de dos tipos:

- Ondas *Rayleigh* ***R***, Fig. A.3(c), el desplazamiento de partículas se encuentra en el plano vertical y describe una elipse en sentido contrario a la propagación de las ondas.
- Ondas *Love* ***L***, Fig. A.3(d), cuyo desplazamiento es similar al de las ondas ***S***, sin desplazamiento vertical y sacude el suelo en un plano horizontal y paralelo a la superficie terrestre, pero en ángulo recto a la dirección de propagación.

A.3. TÉCTONICA DE MÉXICO.

México está situado en una de las zonas sísmicas con mayor actividad del orbe, la cual se ubica, como se mencionó, en las costas del Océano Pacífico, y está rodeado por cinco placas tectónicas: Caribe, Cocos, Norteamericana, Pacífico y Rivera mostradas en la Fig. 1.1. La identificación del área tectónica se define por el ángulo de inclinación de la placa oceánica de la zona. Las variaciones en los parámetros geofísicos y geológicos son capaces de originar modificaciones en el proceso de subducción a diferentes profundidades. La geometría de subducción de la costa occidental de México muestra, de forma general, la placa de Cocos adentrándose (subducción) a la placa de Norte América con un ángulo de poca inclinación (14°, para las zonas de Guerrero, Michoacán y Oaxaca), y está delimitada al oeste por la placa de Rivera y al este por la placa del Caribe [59, 60, 61].

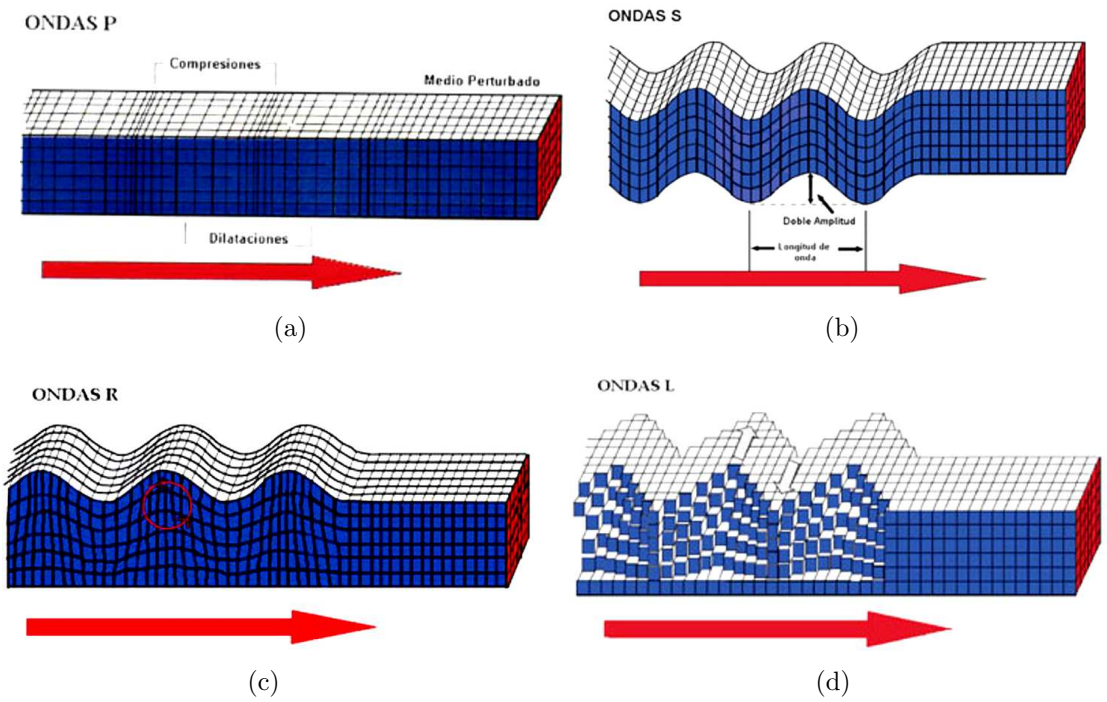


Figura A.3: Imágenes con los distintos tipos de ondas elásticas: ***P***, ***S***, ***R*** y ***L***.

A.4. INTENSIDAD Y MAGNITUD.

Existen dos métricas principales para determinar el tamaño de un sismo, las cuáles son: intensidad –*grado de destrucción causada en el área afectada*– y magnitud –*cantidad de energía liberada* de un sismo.

A.4.1. INTENSIDAD.

Es una medida subjetiva del daño y la violencia de los efectos provocados por el movimiento del suelo sobre los objetos, las personas, la vulnerabilidad de las construcciones y depende de las condiciones del terreno.

En 1883, se realizó la primera escala de intensidad para catalogar y cuantificar los terremotos, elaborada por el sismólogo italiano *M. de Rossi* y el suizo *F. Forel*, quienes propusieron una escala de diez grados conocida como *Rossi-Forel*, y está basada en el poder destructivo, realizando estudios de los daños ocasionados por los sismos. *G. Mercalli* crea una nueva escala de, también, 10 grados de intensidad en 1902, posteriormente incrementada a 12 por *A. Cancani. Sieberg*, en 1923, publica una escala más detallada basada en el trabajo de *Mercalli-Cancani*, modificada posteriormente por los sismólogos norteamericanos *Wood* y *Newman* en 1931, condensando la escala de *Mercalli-Cancani-Sieberg*, surgiendo, de esta manera, la escala *Mercalli* Modificada (*MM*). Esta escala de 12 grados está expresada en números romanos.

Una intensidad *I* define un suceso percibido por algunas personas, mientras que la intensidad *XII* es asignada a eventos extremos que provocan destrucción total. Sismos de intensidad entre

II y *III* son, casi, equivalentes a los de magnitud entre 3 y 4 grados en la escala de *Richter*, mientras que los niveles *XI* y *XII* pueden asociarse a las magnitudes de 8 y 9 grados en la escala de *Richter*.

El empleo de esta escala de intensidad ha permitido crear mapas de isosistas -curvas formadas por la unión de puntos de la superficie con igual intensidad- y representan los efectos del sismo en distintos lugares en los cuales se sintió éste. La forma de estas líneas depende de la orientación de la falla, la longitud de la zona de ruptura, los tipos de terrenos, formaciones geológicas, entre otros factores.

A.4.2. MAGNITUD

Esta métrica es cuantitativa e instrumental del tamaño o importancia de un evento sísmico asociada a la energía sísmica liberada por un sismo durante la ruptura de la falla. La magnitud no está acotada por arriba; carece de límite superior, teóricamente, pero está limitada por la resistencia de las rocas de la corteza terrestre y la longitud de ruptura de la zona de falla. Es, además, independiente del sitio de observación.

Charles Francis Richter, sismólogo del Instituto Tecnológico de California (*CalTech*), utilizó el término magnitud para clasificar los sismos, por primera vez, y de esta forma poder estimar la energía liberada y compararla con otros eventos sísmicos.

La magnitud está ligada a una función logarítmica calculada a partir de la amplitud de la señal registrada por el sismógrafo (*ML, Ms, mb*) o por la duración (*MD*) del sismograma. Ésta puede ser determinada por medio de distintas escalas existentes que son funciones del tipo de onda sísmica utilizada para medir el tamaño del sismo, siendo, entre otras, las más importantes: magnitud local (*ML*); magnitud de las ondas superficiales (*Ms*); magnitud de las ondas de volumen (*mb*), momento sísmico (*Mo*) y magnitud momento (*Mw*).

A.4.3. MAGNITUD LOCAL *ML*.

Se definió en California, para sismos locales, con una distancia epicentral de 600 km aproximadamente, y está determinada a partir de la amplitud máxima registrada por un sismógrafo tipo *Wood Anderson (WA)* con constantes específicas (factor de amortiguamiento = 0.8; amplificación estática = 2,800 y período = 0.8) localizado a 100 km de la fuente sísmica, expresada mediante la siguiente ecuación:

$$ML = LogA\,(\Delta) - LogA_0\,(\Delta) \tag{A.1}$$

;donde A es la amplitud máxima de la traza registrada a una distancia Δ del epicentro y A_0 la amplitud máxima producida por un sismo patrón, que produciría una deflexión de 0.001 mm en un sismógrafo ubicado a 100 km del epicentro. Para una región particular y un sismógrafo distinto del tipo *WA* se debe realizar una adecuación en la distancia del término A_0 .

A.4.4. MAGNITUD DE LAS ONDAS SUPERFICIALES *Ms*.

Esta magnitud es válida para sismos con foco no someros, donde la amplitud máxima producida debe ser medida en el modo fundamental por las ondas superficiales $\boldsymbol{R}$, con un período

$T \approx 18$ a 20 segundos. La expresión empleada queda como:

$$Ms = Log\left(\frac{A}{T}\right) + 1.66\log\Delta + 3.3 \tag{A.2}$$

;donde A es la máxima amplitud del desplazamiento del suelo, medido en micras; Δ es la distancia epicentral, medida en grados y T es el período de la onda, expresado en segundos.

La expresión (A.2) es válida para distancias dentro del intervalo $20° \leq \Delta \leq 90°$ y para terremotos con focos ubicados a profundidades menores a 70 km.

A.4.5. MAGNITUD DE LAS ONDAS DE VOLUMEN *mb*.

Determinar la magnitud Ms para los sismos cuya profundidad focal sea mayor a 50 km es complicado ya que no se generan ondas superficiales con amplitud suficiente. Para compensar esta insuficiencia se utiliza un factor de corrección para poder utilizar las ondas de volumen. La magnitud mb se basa en la amplitud de ondas de volumen con períodos cercanos a 1 segundo y considera la relación que existe entre la amplitud de las ondas ***P*** y ***S*** (ondas de volumen). Ésta es válida para sismos ocurridos a distintas profundidades y registrados a distancias comprendidas entre $5°y$ $90°$. La fórmula de *Gutenberg* nos permite calcular esta magnitud, la cual se define como:

$$mb = \log\left(\frac{A}{T}\right) + Q\,(\Delta \cdot h) \tag{A.3}$$

;donde A es la amplitud de la señal sísmica medida sobre la componente vertical de un registro corto, especificado en micras; T es el período, especificado en segundos; Q es el factor de atenuación de la onda expresado en función de la distancia epicentral Δ y h la profundidad del foco, expresado en km.

Las escalas de magnitud Ms y mb no reflejan de forma adecuada el tamaño de sismos muy grandes, subestiman su valor y se saturan; es decir, proporcionan una estimación poco precisa de la energía liberada. La saturación de la magnitud mb se da entre $6.5° - 6.8°$ y de la magnitud Ms entre $8.3° - 8.7°$.

A.4.6. MOMENTO SÍSMICO *Mo*.

El momento sísmico es una medida con mayor consistencia para medir el tamaño de un sismo basada en el área de ruptura y el desplazamiento medio de ésta que generó el sismo. Se puede calcular a partir de la amplitud del espectro de las ondas sísmicas.

La orientación, dirección de la falla y el tamaño del sismo son descritos mediante la geometría de la falla y el momento sísmico. Estos parámetros pueden determinarse a partir del análisis de las formas de onda del sismo. Las distintas formas y direcciones del movimiento de ondas registradas en diferentes distancias y azimuts desde el foco del sismo son utilizadas para determinar la geometría de la falla y la amplitud de la onda para conocer el momento sísmico. La cantidad de energía liberada por un sismo, empleando el momento sísmico, puede relacionarse con los parámetros de la falla mediante la relación de *Aki*:

$$Mo = \mu A < d > \tag{A.4}$$

;donde Mo momento sísmico, expresado en dinas-cm.; μ es la rigidez o cizallamiento producido por el sismo en la roca, expresado en $dinas/cm^2$; A es el área de la ruptura o falla, especificada en cm; y <d> es el promedio del desplazamiento de la falla expresado en cm.

A.4.7. MAGNITUD MOMENTO *Mw.*

La magnitud momento desarrollada por *Hiroo Kanamori* en el Instituto Tecnológico de California en 1977 está basada en el momento sísmico. Esta magnitud es empleada en la medición de períodos más largos y está expresada como sigue:

$$Mw = \frac{2}{3} \log_{10} M_O - 10.7 \tag{A.5}$$

;donde Mo está expresado en *Newtons* $\times$ metro.

A.5. RUPTURA SÍSMICA.

Una proporción de la energía elástica almacenada en forma de esfuerzo en la roca deformada es gastada en crear la falla, romper la roca y vencer la fricción entre ambas caras de la fractura, que trata de frenar el movimiento. Otra proporción puede permanecer en las rocas (esfuerzo residual) y el resto se libera en forma de ondas sísmicas. Esta energía liberada o energía sísmica es la que se propaga y es conocida como sismo [62].

Las concentraciones de esfuerzo pueden ocurrir donde una aspereza (una zona con resistencia a la ruptura mucho mayor que la del material en derredor) haya resistido mientras se rompía el material a su alrededor. Los lugares vecinos a zonas de baja resistencia a la ruptura o a microfallas (áreas pequeñas donde la fricción es considerada nula), donde el corrimiento de las caras produce concentraciones de deformación y, probablemente, debilitamiento de la roca, son sitios donde también pueden ocurrir concentraciones de esfuerzo. Cuando se rompe el lugar donde existe una gran concentración de esfuerzo, con su correspondiente corrimiento, se producen concentraciones de esfuerzo en los bordes de la ruptura que, si son mayores de lo que puede soportar la roca, hacen que la falla se propague; es decir, que aumente y continúe hasta que las concentraciones de esfuerzo producidas ya no sean lo suficientemente grandes para romper la roca, generando así una nueva superficie de ruptura.

Si el medio en torno al hipocentro tiene un nivel alto de esfuerzo es muy fácil que la ruptura se propague; en cambio si el nivel es bajo (por ejemplo, por la ocurrencia reciente de un sismo) es posible que la ruptura se detenga ya sea porque encuentra una aspereza que no pueda romper con las concentraciones de esfuerzo producidas o a cambios en la orientación del plano de la ruptura.

Bibliografía.

[1] Chavez M., Olsen K. B., and Cabrera E. Broadband modeling of strong ground motions for prediction purposes for subduction earthquakes occurring in the Colima-Jalisco region of Mexico. En *13WCEE*, page 1653, Vancouver, B.C., Canada, August 1-6, 2004.

[2] Caicedo M. I. and Mora P. J. *Tópicos en Propagación de Ondas*. Universidad Simón Bolívar, 2004.

[3] Aki K. and Richards P. *Quantitative Seismology*. University Science Books, second edition, 2002.

[4] Minkoff S.E. Spatial Parallelism of a 3D Finite Difference Velocity-Stress Elastic Wave Propagation code. *SIAM J. Sci. Comput.*, 14:1–19, 2002.

[5] Cabrera Flores Eduardo César. Introducción al Supercómputo. En *Semana de Supercómputo 2001*. Universidad Nacional Autónoma de México, 2001. DGSCA/Supercómputo.

[6] Madariaga R. Dynamics of an Expanding Circular Fault. *Bull. Seismol. Soc. Amer.*, 79:655–699, 1989.

[7] Departamento de Supercómputo, DGSCA, UNAM, Supercomputadora KanBalam. `http://www.super.unam.mx/index.php?option=com_content&task=view&id=35&Itemid=66`.

[8] Hennesy L. John and Patterson David. *Computer Architecture: A Quantitative Approach*. Morgan Kaufman Publishers, Inc., San Francisco, CA, second edition, 1996.

[9] Lloyd D. Fosdick, Elizabeth R. Jessup, Carolyn J. C. Schauble, and Gitta Domik. *Introduction to High-Performance Scientific Computing*. MIT, Press, 1996.

[10] Flynn M. Some Computer Organizations and Their Effectiveness. *IEEE Trans. Comput*, C-21:948, 1972.

[11] Wilson G.K. Grand challenges to computacional science, Future Generation Computer. *Systems*, 5(2-3):171–189, 1989.

[12] Moore E. Gordon. Cramming more components onto integrated circuits. *Electronics*, 38(8), April 1965.

[13] Moore E. Gordon. Excerpts from a Conversation with Gordon Moore: Moores Law. Technical report, Intel, 2005.

[14] Brown Stephen D. and Zvonko G. Vranesic. *Fundamentals of digital logic with VHDL design*. McGraw-Hill, second edition, 2005.

[15] Opengl, wikipedia. http://es.wikipedia.org/wiki/OpenGL.

[16] OpenGL, sitio oficial . http://www.opengl.org/.

[17] CUDA. http://www.nvidia.com/object/cuda_home.html.

[18] CUBLAS. http://developer.download.nvidia.com/compute/cuda/1_1/CUBLAS_Library_1.1.pdf.

[19] Cathy May et al. *The PowerPC architecture : a specification for a new family of RISC processors*. Morgan Kaufman Publishers, second edition, 1994.

[20] PowerPC. http://www-03.ibm.com/systems/p/hardware/whitepapers/power/ppc_arch.html.

[21] Lapsley Phil. *DSP processor fundamentals : architectures and features*. IEEE, 1997.

[22] Sitio de las 500 supercomputadoras más poderosas del orbe. http://www.top500.org, 2008.

[23] CELL IBM. http://www.research.ibm.com/cell.

[24] Gschwind M. and Hofstee H.P. and Flachs B. and Hopkins M. and Watanabe Y. and Yamazaki T. Synergistic Processing in Cell's Multicore Architecture. *Micro, IEEE*, 26(2):10–24, 2006.

[25] Linux on Cell BE-based Systems. http://www.bsc.es/projects/deepcomputing/linuxoncell/.

[26] The Potential of the Cell Processor for Scientific Computing. http://repositories.cdlib.org/cgi/viewcontent.cgi?article=4262&context=lbnl.

[27] Buttari Alfredo, Luszczek Piotr, Kurzak Jakub, Dongarra Jack, and Bosilca George. SCOP3: A Rough Guide to Scientific Computing On the PlayStation 3. Technical report, University of Tennessee Knoxville, 2007.

[28] Departamento de Supercómputo, DGSCA, UNAM, Supercomputadora Origin2000. http://www.super.unam.mx/index.php?option=com_content&task=view&id=50&Itemid=86.

[29] Parallel Virtual Machine. http://www.csm.ornl.gov/pvm.

[30] Message Passing Interface. http://www-unix.mcs.anl.gov/mpi.

[31] Message Passing Interface, Forum. http://www.mpi-forum.org.

[32] Pthreads definición de la IEEE. *Std* 1003.1.

[33] OpenMP. http://www.openmp.org/blog.

[34] Corba. http://www.corba.org.

[35] Corba FAQ. http://www.omg.org/gettingstarted/corbafaq.htm.

[36] Distributed Component Object Model.
http://en.wikipedia.org/wiki/Distributed_Component_Object_Model.

[37] Buyya R. *High performance cluster computing, vol. 2 : programming and applications*. Prentice Hall, 1999.

[38] Amdahl G. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. En *AFIPS*, pages 483–485, 1967.

[39] Pancake M.C. Is Parallelism for You? *IEEE Computational Science & Engineering*, 3(2), June 1996.

[40] Foster I. *Designing and Building Parallel Programs*. Addison-Wesley, 1995.

[41] Cormen T., Leiserson C., Rivest R., and Stein C. *Designing and Building Parallel Programs*. Cambridge, Massachusetts : MIT, second edition, 2001.

[42] Pacheco P. *Parallel Programming with MPI*. Morgan-Kaufman, San Francisco, CA, 1997.

[43] McBryan Oliver A. An overview of message passing environments. *Parallel Computing*, 20(4):417–444, April 1994.

[44] Gordillo Ruiz José Luis. Algoritmo paralelo de entrenamiento de redes neuronales artificiales para sistemas de control. Tesis de Licenciatura, Universidad Nacional Autónoma de México, 1998.

[45] The Message Passing Interface (MPI) Standar. http://www-unix.mcs.anl.gov/mpi.

[46] MPI: A Message-Passing Interface Standar. Message Passing Interface Forum. http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html, 1995. Ver. 1.1.

[47] MPI-2: Extensions to the Message-Passing Interface. Message Passing Interface Forum. http://www-unix.mcs.anl.gov/mpi, 1997.

[48] MPICH1. http://www-unix.mcs.anl.gov/mpi/mpich1.

[49] MPICH2. http://www.mcs.anl.gov/research/projects/mpich2.

[50] LAM. http://www.lam-mpi.org.

[51] OpenMPI. http://www.open-mpi.org.

[52] Decyk Viktor K., Norton, Charles D. Gardner, and Henry J. Why Fortran? *Computing in Science & Engineering*, 9(4):68–71, July-Aug 2007.

[53] InfiniBand. http://en.wikipedia.org/wiki/InfiniBand.

[54] Lustre. http://wiki.lustre.org.

[55] Allen M., Herrera I., and Pinder G. *Numerical Modelling in Science and Engineering*. John Wiley & Sons, New York, York, 1988.

[56] Mendoza C. and Hartzell S. Slip Distribution of the 9/19/1985 September 1985 Michoacan, Mexico, Earthquake: near Source and Telesismic constrains. *Bull. Seismol. Soc. Amer.*, 79:655–699, 1989.

[57] Cabrera E., Chavez M., Madariaga R., Perea N., and Frisenda M. 3D Parallel Elastodynamic Modeling of Large Subduction Earthquakes. En *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, volume 4757/2007 of *Lecture Notes in Computer Science*, pages 373–380, París, Francia, October 2007. Springer Berlin / Heidelberg.

[58] Morgan W. Jason. Rises,Trenches, Great Faults, and Crustal Blocks. *Journal of Geophysical Research*, 73(6):1959–1982, 1968.

[59] Dewey J and Suárez G. Seismotectonics of Middles America. *The Geology of North America Decade Map*, 1:309–321, 1991.

[60] Pardo Pedemonte Mario Hernan. *Características sismotectónicas de la subducción de las placas de Rivera y Cocos de México*. Tesis de Doctorado, Universidad Nacional Autónoma de México, 1993.

[61] Kostoglodov V. and Ponce L. Relationship between subduction and seismicity in the Mexican part of the Middle America Trench. *Journal of Geophysical Research*, 99(B1):729–742, 1994.

[62] McCann W. R., Nishemko S. P., Sykes L. R., and Krause J. H. Seismic gaps and Plate tectonics: seismic potential for major boundaries. *Pure Appl. Geophysics*, 117:1082–1147, 1979.