Desarrollo de un software libre para el modelado de la propagación de ondas elásticas 2D por el método de elementos finitos

Peter Escamilla Mahecha, Sebastián Roa Prada

Maestría en Software Libre, Universidad Autónoma de Bucaramanga

Bucaramanga, Colombia

pescamilla@unab.edu.co

sroa@unab.edu.co

***Abstract* - This article explains the underlying reasons for the development of the open source software EWaveFem to model the propagation of two-dimensional elastic waves using the finite elements method. The software EWaveFem, which is presented to the public, employs constant strain triangles to subdivide the domain under study, as this is the simplest element shape of a mesh to model wave propagation in two dimensions.**

This program allows the user to specify the number of iterations, the number of elements along both the width and the height of the domain, the thickness of the domain, the material density, the area of the domain, the Poisson's module and the time step between each iteration. Once the parameters have been specified, the program executes the finite element algorithm and generates a file with the results for the displacement, speed, acceleration at each node, for each iteration.

This article also shows the details of the algorithm implemented of the finite element method with elastodynamics, including the generation of the stiffness matrix and the mass matrix for the elements.

After testing the developed code, it was verified the high feasibility of the finite element method to model elastic wave propagation, along with different boundary conditions and non-regular shapes. This first version of the developed software provides the basis for future improvements to allow for more general modeling conditions such as having quadrilateral element shapes, 3D modeling, and different materials in the same simulation.

Keywords: wave simulation, finite element method, wave modeling, seismography, free software, open source

***Resumen* – Este artículo explica el porqué de la necesidad del desarrollo del software libre EWaveFem para modelar la propagación de ondas elásticas bidimensionales usando el método de los elementos finitos. El software EWaveFem, que se presenta al público, emplea triángulos de deformación unitaria constante para subdividir el dominio bajo estudio, ya que es la forma más sencilla de los elementos de una malla para analizar la propagación de ondas en dos dimensiones.**

Este programa permite al usuario ingresar el número de iteraciones, el número de elementos que conformaran la grilla tanto a lo ancho como a lo alto, el espesor del dominio, la elasticidad del material, la densidad del material, el área del dominio, el módulo de Poisson y la diferencia de tiempo entre cada iteración, ejecuta el algoritmo de elementos finitos y genera un archivo con el desplazamiento, la velocidad y la aceleración de cada nodo en cada una de las diferentes iteraciones con la misma aplicación.

En este artículo también se muestra el algoritmo implementado del método de elementos finitos con elastodinámica, incluyendo la generación de la matriz de rigidez y la matriz de masa para los elementos.

Luego de probar el software desarrollado se pudo verificar la alta viabilidad del método de elementos finitos para el modelado de la propagación de ondas junto con diferentes condiciones de frontera y formas no regulares. Esta primera versión del software desarrollado provee las bases para luego desarrollar software que condiciones de modelado más generales tales como tener elementos cuadriláteros, modelado en 3D y diferentes materiales en un mismo modelado.

Palabras clave: simulación de ondas, elementos finitos, modelación de ondas, sismografía, software libre

I. Introducción

Las empresas de extracción de petróleo realizan grandes esfuerzos en el modelado de la propagación de ondas elásticas para hacer predicciones lo más acertadas posibles sobre la estructura del subsuelo y así reducir la incertidumbre en las actividades de exploración de petróleo en zonas de complejidad geológica [1].

Las predicciones de existencia de petróleo por medio de simulación de propagación de ondas generalmente se obtienen por modelos de diferencias finitas o de trazado de rayos. Pero hay casos donde la simulación requiere un mayor grado de precisión, como cuando se presentan diapiros de sal o medios con anomalías de velocidad. Cuando se está modelando con trazado de rayos o cuando se usa diferencias finitas, se dificulta la definición de geometrías complejas. En estos casos se hace necesario el uso del método de elementos finitos [2].

Un software que implementa este método combinado con el método espectral es SpecFem2D el cual utiliza el método de elementos espectrales que es a su vez una formulación del método de elementos finitos [3] [4] y tiene bastantes optimizaciones que permiten que trabaje gran cantidad de datos con menos recursos que la formulación general de elementos finitos [5] [6].

El uso del método de elementos finitos en el problema de propagación de ondas elásticas en aplicaciones de exploración petrolera no ha sido muy explorado a nivel industrial, por lo que es una gran oportunidad ingresar en este mercado dado el alto nivel de personalización al que se podría llegar y porque con este método es la mejor manera en que se puede hacer la definición de las formas complejas y de esa manera reducir la incertidumbre en la exploración.

Así que en este artículo se expone a un mayor detalle los motivos por los que se realiza este desarrollo, sus ventajas y desventajas. Se expone el por qué es conveniente manejarlo como un software libre, el modelado matemático donde se presenta el problema que se quiere solucionar de manera general pasando por todo el desarrollo. Y finalmente se exponen los resultados obtenidos luego del desarrollo del aplicativo.

II. Elección de Método

Existen diversos métodos que permiten simular la propagación de ondas elásticas, dentro de los cuales se encuentran los métodos directos tales como el método de diferencias finitas, el método pseudoespectral y el método de elementos finitos. Otros métodos que también se usan son los de ecuaciones integrales y los asintóticos.

Los métodos directos representan de manera más realista los problemas bajo estudio, lo que aporta claridad con respecto a las entradas y salidas del sistema que se está modelando. Como desventajas tienen que requieren más recursos que los otros métodos.

Como ventaja particular del método de elementos finitos se tiene que permite una mayor personalización en cuanto a la manera como realiza el modelado. Por ejemplo, permite gran flexibilidad en la aplicación de condiciones de frontera, que a su vez pueden ser funciones que representan fuerzas dinámicas en uno o varios puntos del dominio bajo estudio. Otra ventaja es que permite trabajar con mallas de esparcimiento variable, ubicando una mayor densidad de nodos donde más se requieren y ubicando menos nodos en las regiones donde se prevé que la respuesta del sistema va a cambiar poco o no se requiere conocer a detalle su comportamiento.

III. Software Libre

Dado lo específico del uso de métodos finitos, ya existe una serie de software reconocido, el cual aunque ofrece buenas prestaciones, es costoso y en caso de no cumplir las necesidades de cada cliente, llega a no ser útil para muchos. Aquí es donde el software libre plantea una gran ventaja, pues además de no tener costo asociado, en caso de quererse hacer alguna modificación se puede desarrollar sin problemas legales. Para la muestra, ya existen algunos casos de software libre para modelamiento por el método de elementos finitos [7].

Dado que las personalizaciones van haciendo crecer el software para ser más útil para muchos más usuarios, empresas y universidades, este proyecto desde su inicio busca principalmente entrar con una licencia de software libre para que pueda ser extendido por otras personas y el conocimiento sea para el bien de todos, permitiendo también a futuro la integración con software existente como gmsh [8].

El software que desarrollado se encuentra en el repositorio de proyectos libres llamado Github en la dirección https://github.com/pescamillam/EWaveFem a la espera de ser útil para más usuarios, y con una licencia que permite su uso comercial, pues en general se busca que sea ese el uso que se le dé a este software.

IV. Discretización del Dominio

El método de elementos finitos funciona subdividiendo el objeto de estudio que llamaremos dominio, por medio de una malla, conformada por una serie de elementos que están compuestos por nodos.

Para el desarrollo del método se inició asumiendo un dominio con forma rectangular, tal y como se muestra en la Fig. 1, suponiendo un perfil de terreno con distancia horizontal y profundidad vertical. La subdivisión del dominio se observa en la Fig. 2, en donde cada elemento obtenido tiene forma de cuadrado, y cada uno de estos cuadrados se subdivide en dos triángulos como se observa en la Fig. 3.

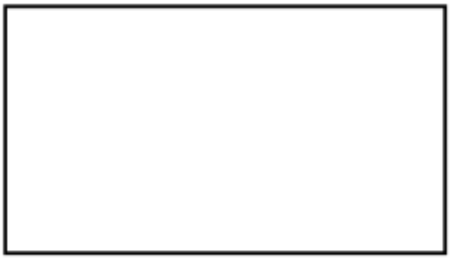

Fig. 1. Dominio.
Fuente: Autores

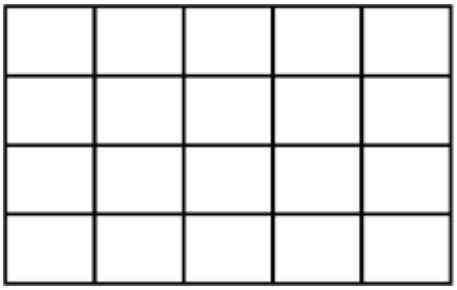

Fig. 2. Dominio subdividido en elementos.
Fuente: Autores

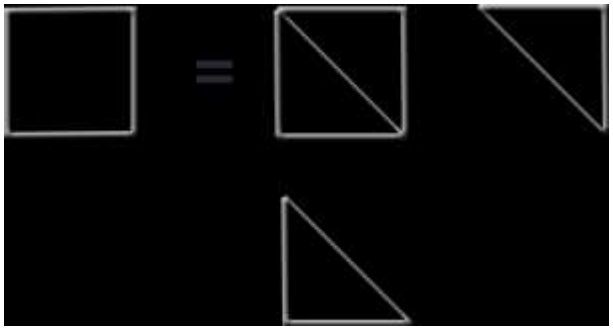

Fig. 3. Subdivisión del elemento tipo 1 y 2.
Fuente: Autores

Se seleccionaron los triángulos como elemento tipo para el proceso debido a su simplicidad y a su capacidad de formar figuras cuadradas o rectangulares fácilmente. Cada uno de los dos triángulos presenta posiciones diferentes, por lo tanto se nombran como elemento tipo 1 y elemento tipo 2.

Cada esquina de los elementos, ya sean cuadrados o triangulares, las llamaremos nodos, y vale la pena resaltar que los dos triángulos tipo se relacionan directamente por medio de los nodos de sus diagonales y a su vez los elementos cuadrados se relacionan mediante los nodos compartidos.

V. ALGORITMOS DESARROLLADOS

Con la discretización definida en el capítulo anterior es posible establecer el proceso necesario para la aplicación del método de elementos finitos, por lo tanto se establecen los subprocesos desarrollados como pasos necesarios para llegar a la correcta aplicación del método, pasos que serán definidos a lo largo de los subcapítulos siguientes.

A. Diagrama de flujo del proceso de solución por el método de elementos finitos

Una de las etapas más complejas del proceso es la creación de las matrices globales de rigidez y de masa, por lo que estas se dividen en matrices locales de cada tipo de elemento, de tal forma que puedan conjugarse en una sola matriz global para cada caso. Teniendo en cuenta esto se establece el diagrama con la ruta general del proceso resumido. Dicho proceso general se observa en la Fig. 4, partiendo del ingreso de datos por el usuario.

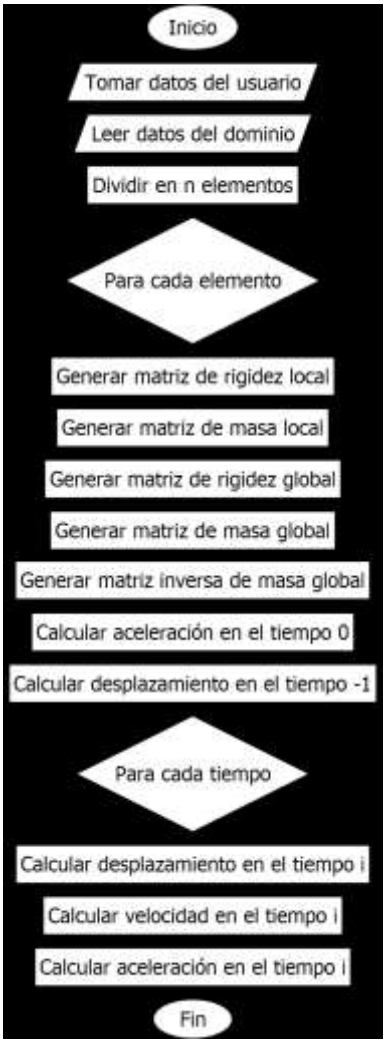

Fig. 4. Diagrama de flujo del algoritmo.
Fuente: Autores

El primer subproceso realizado es la división del dominio en *n* elementos teniendo en cuenta la cantidad de elementos a lo ancho y a lo alto según se ingresa por el usuario, luego se itera sobre todos los elementos como se detalla a continuación:

Se aplica el proceso de generar las matrices, el cual consiste en tomar cada elemento que se está iterando y generar una matriz de rigidez para los elementos tipo 1 y una para los tipo 2. El proceso de creación se define en los capítulos siguientes, de igual forma que la unión de estas matrices en la matriz global de rigidez.

Finalmente, el postproceso consiste en la obtención de los valores que se desean mostrar como resultado, por lo tanto, se calcula la aceleración en un tiempo inicial definido como 0, y un desplazamiento inicial como -1, suponiendo un estado inicial sin movimiento. Para los tiempos definidos, se establecen iteraciones que calculan los desplazamientos, velocidades y aceleraciones de cada nodo.

B. Cálculo del número de nodos

Para obtener la cantidad de nodos se utilizan como valores de entrada la cantidad de elementos a lo ancho y la cantidad de elementos a lo alto, según puede observarse en la Fig. 5, de tal forma que la cantidad de nodos en una dimensión corresponde a la cantidad de elementos aumentado en uno.

Fig. 5. Malla con elementos y nodos
Fuente: Autores

En consecuencia, el número de nodos puede calcularse por medio de la siguiente ecuación:

$$n = (w + 1)(h + 1) \tag{1}$$

Al desarrollar el producto de la ecuación (1) se obtiene:

$$n = wh + h + w + 1 \tag{2}$$

Donde:

n es el número total de nodos
w es la cantidad de elementos a lo ancho
h es la cantidad de elementos a lo alto

C. Definición de los elementos tipo

En la Fig. 6 se muestra el elemento tipo 1 y en la Fig. 7 el elemento tipo 2, compuestos de los nodos i, j y m, con la posición de estos nodos se generan tanto las matrices locales de masa como las locales de rigidez, este proceso se realiza ubicando en cada posición de la matriz global la suma de los valores de las matrices locales dependiendo del tipo de elemento, siguiendo paso a paso cada ecuación necesaria en el proceso como se muestra en las ecuaciones (3) y (4) [9].

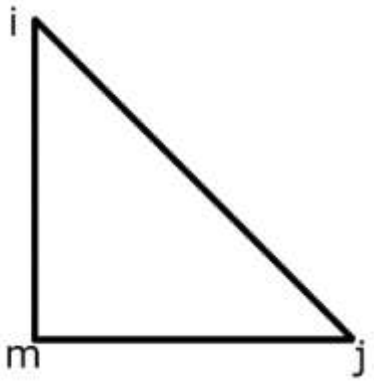

Fig. 6. Elemento tipo 1.
Fuente: Autores

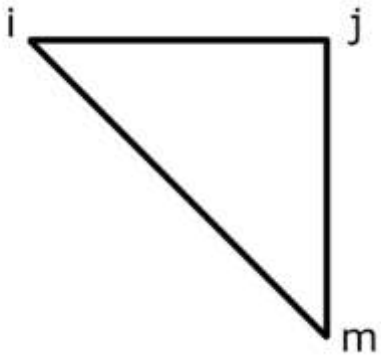

Fig. 7. Elemento tipo 2.
Fuente: Autores

Los detalles de la derivación de las ecuaciones (3) y (4) se pueden observar en el Apéndice I.

$$[k] = t \iint_A [B]^T [D][B] dxdy \tag{3}$$

$$[k] = tA[B]^T [D][B] \tag{4}$$

El elemento tipo 1 mostrado en la Fig. 6 es un triángulo que se ve complementado al juntarse con el de la Fig. 7, permitiendo formar una superposición como se muestra en la Fig. 8. Lo que con los múltiples grupos de elementos se puede generar una malla rectangular de cualquier tamaño.

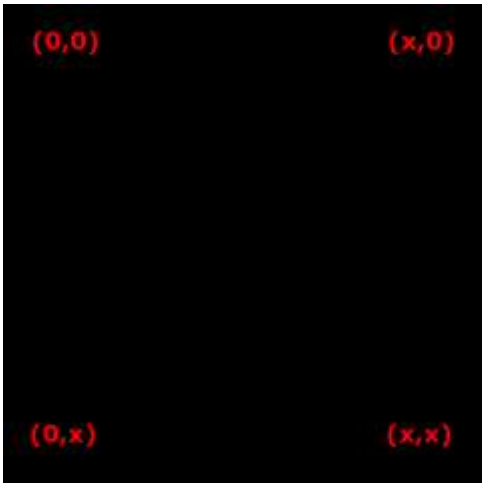

Fig. 8. Coordenadas locales de un grupo de elementos.
Fuente: Autores

Para proporcionar la localización correcta de cada elemento dentro del dominio, se define una distancia *x* entre los nodos de cada elemento, de acuerdo con las coordenadas mostradas en la Fig. 8. Esta coordenada hace referencia a la posición inicial de cada nodo, posición que irá cambiando con el avance de las iteraciones según el desarrollo del algoritmo de elementos finitos basado en el tiempo.

D. Matrices locales de rigidez

Antes de iniciar el proceso de creación de las matrices locales de rigidez, se asignan coordenadas de posición para cada nodo teniendo en cuenta sus grados de libertad, de tal forma que para un elemento triangular, compuesto por 3 nodos, en donde cada nodo tiene 2 grados de libertad, correspondientes al desplazamiento en *x* y *y*, se obtiene una matriz de dimensiones 6x6, en donde cada campo es nombrado como se observa en la Fig. 9.

	i		j		m	
i	[0,0]	[1,0]	[2,0]	[3,0]	[4,0]	[5,0]
	[0,1]	[1,1]	[2,1]	[3,1]	[4,1]	[5,1]
j	[0,2]	[1,2]	[2,2]	[3,2]	[4,2]	[5,2]
	[0,3]	[1,3]	[2,3]	[3,3]	[4,3]	[5,3]
m	[0,4]	[1,4]	[2,4]	[3,4]	[4,4]	[5,4]
	[0,5]	[1,5]	[2,5]	[3,5]	[4,5]	[5,5]

Fig. 9. Coordenadas de relación entre los grados de libertad en una matriz local.
Fuente: Autores

Luego de conformada la matriz se inicia el proceso matemático para calcular el valor de cada término de la matriz por medio de la ecuación (5), cuya deducción puede observarse en el Apéndice II, la cual es la matriz de rigidez para un elemento triangular de deformación unitaria constante. Su aplicación se realiza de forma mecánica dentro del proceso iterativo general, permitiendo observar la generación del valor de cada campo para el elemento tipo 1, y para el elemento tipo 2.

$$[k] = \frac{tE}{4A(1+v)(1-2v)} \times \begin{Bmatrix} \beta_i^2 w + \gamma_i^2 r & \beta_i\gamma_i v + \beta_i\gamma_i r & \beta_i\beta_j w + \gamma_i\gamma_j r \\ \beta_i\gamma_i v + \beta_i\gamma_i r & \gamma_i^2 w + \beta_i^2 r & \beta_j\gamma_i v + \beta_i\gamma_j r \\ \beta_i\beta_j w + \gamma_i\gamma_j r & \beta_j\gamma_i v + \beta_i\gamma_j r & \beta_j^2 w + \gamma_j^2 r \\ \beta_i\gamma_j v + \beta_j\gamma_i r & \gamma_i\gamma_j w + \beta_i\beta_j r & \beta_j\gamma_j v + \beta_j\gamma_j r \\ \beta_i\beta_m w + \gamma_i\gamma_m r & \beta_m\gamma_i v + \beta_i\gamma_m r & \beta_j\beta_m w + \gamma_j\gamma_m r \\ \beta_i\gamma_m v + \beta_m\gamma_i r & \gamma_i\gamma_m w + \beta_i\beta_m r & \beta_j\gamma_m v + \gamma_j\beta_m r \end{Bmatrix} \quad (5)$$

$$\begin{Bmatrix} \beta_i\gamma_j v + \beta_j\gamma_i r & \beta_i\beta_m w + \gamma_i\gamma_m r & \beta_i\gamma_m v + \beta_m\gamma_i r \\ \gamma_i\gamma_j w + \beta_i\beta_j r & \beta_m\gamma_i v + \beta_i\gamma_m r & \gamma_i\gamma_m w + \beta_i\beta_m r \\ \beta_j\gamma_j v + \beta_j\gamma_j r & \beta_j\beta_m w + \gamma_j\gamma_m r & \beta_j\gamma_m v + \gamma_j\beta_m r \\ \gamma_j^2 w + \beta_j^2 r & \beta_m\gamma_j v + \beta_j\gamma_m r & \gamma_j\gamma_m w + \beta_j\beta_m r \\ \beta_m\gamma_j v + \beta_j\gamma_m r & \beta_m^2 w + \gamma_m^2 r & \gamma_m\beta_m v + \beta_m\gamma_m r \\ \gamma_j\gamma_m w + \beta_j\beta_m r & \gamma_m\beta_m v + \beta_m\gamma_m r & \gamma_m^2 w + \beta_m^2 r \end{Bmatrix}$$

Donde:

$$\beta_i = y_j - y_m \quad \beta_j = y_m - y_i \quad \beta_m = y_i - y_j$$

$$\gamma_i = x_m - x_j \quad \gamma_j = x_i - x_m \quad \gamma_m = x_j - x_i \quad (6)$$

$$w = 1 - v \quad r = \frac{1-2v}{2}$$

A pesar de contar con la posibilidad de ingresar los datos iniciales manualmente en la interfaz de usuario, se resuelve la matriz a modo de ejemplo cambiando el valor de la distancia *x* por un valor de 30 m, en este caso los valores para el elemento tipo 1 serían:

$$\beta_i = 0 \quad \beta_j = 30 \quad \beta_m = -30$$

$$\gamma_i = -30 \quad \gamma_j = 0 \quad \gamma_m = 30 \quad (7)$$

Continuando el ejemplo, se asigna un espesor de 1 m, módulo de elasticidad $3x10^9$ N/m², área de 450 m², módulo de Poisson de 0.3, estas características hacen referencia a un suelo de limo arcilloso, de tal forma que la matriz de rigidez para un elemento de la malla sería:

$$[k] = \frac{3 \times 10^9}{4(450)(1+0{,}3)(1-2(0{,}3))} \times \begin{bmatrix} 180 & 0 & 0 & -180 & -180 & 180 \\ 0 & 630 & -270 & 0 & -270 & -630 \\ 0 & -270 & 630 & 0 & -630 & 270 \\ -180 & 0 & 0 & 180 & 180 & -180 \\ -180 & -270 & -630 & 180 & 810 & -450 \\ 180 & -630 & 270 & -180 & -450 & 810 \end{bmatrix} \quad (8)$$

E. Algoritmo para la conjugación de matrices de rigidez y de masa de cada elemento

Contando con las matrices locales de cada elemento tipo 1 y 2 es posible generar la matriz global de rigidez, para lo cual se necesita

realizar un proceso de conjugación que junte todos los elementos, de tal forma que se obtenga una matriz con el total de grados de libertad del dominio, y que a su vez tenga en cuenta el efecto que puede tener las contribuciones de las matrices locales de diferentes elementos sobre nodos compartidos.

La adición realizada se obtiene procesando elemento a elemento y sobre cada uno conjugando los valores de la matriz local en la matriz global dentro de las iteraciones necesarias para el ancho y alto de los elementos del dominio, basándose en la forma triangular del elemento tipo de acuerdo a la Fig. 6 y Fig. 7 y conjugándolos en la matriz global con las posiciones indicadas en la Fig. 10 y Fig. 11.

Donde:
c es la cantidad total de elementos a lo ancho del dominio.
i es la posición horizontal del elemento que cambia a medida que se itera sobre el dominio.
j es la posición vertical del elemento que cambia a medida que se itera sobre el dominio.

A modo de ejemplo se presenta la conjugación de la matriz de rigidez de un elemento tipo 1 con la matriz de rigidez de un elemento tipo 2, las cuales se pueden generar según las posiciones mostradas en la Fig. 10 y Fig. 11, la ecuación (9) para el elemento tipo 1 y la (10) para el elemento tipo 2, que luego de la conjugación se ven representadas en la ecuación (11).

		((c+1)*j+i)*2	((c+1)*j+i)*2+1	((c+1)*(j+1)+i)*2	((c+1)*(j+1)+i)*2+1		((c+1)*(j+1)+(i+1))*2	((c+1)*(j+1)+(i+1))*2+1
((c+1)*j+i)*2		[r1][0][0]	[r1][0][1]	[r1][0][4]	[r1][1][4]		[r1][0][2]	[r1][1][2]
((c+1)*j+i)*2+1		[r1][0][1]	[r1][1][1]	[r1][0][5]	[r1][1][5]		[r1][0][3]	[r1][1][3]
		..						
((c+1)*(j+1)+i)*2		[r1][0][4]	[r1][0][5]	[r1][4][4]	[r1][4][5]		[r1][4][2]	[r1][5][2]
((c+1)*(j+1)+i)*2+1		[r1][1][4]	[r1][1][5]	[r1][4][5]	[r1][5][5]		[r1][4][3]	[r1][5][3]
		..						
((c+1)*(j+1)+(i+1))*2		[r1][0][2]	[r1][0][3]	[r1][4][2]	[r1][4][3]		[r1][2][2]	[r1][2][3]
((c+1)*(j+1)+(i+1))*2+1		[r1][1][2]	[r1][1][3]	[r1][5][2]	[r1][5][3]		[r1][2][3]	[r1][3][3]

Fig. 10. Conjugación de la matriz de rigidez o de masa local en la matriz rigidez global para el elemento tipo 1
Fuente: Autores

		((c+1)*j+i)*2	((c+1)*j+i)*2+1	((c+1)*j+(i+1))*2	((c+1)*j+(i+1))*2+1		((c+1)*(j+1)+(i+1))*2	((c+1)*(j+1)+(i+1))*2+1
((c+1)*j+i)*2		[r1][0][0]	[r1][0][1]	[r1][0][2]	[r1][1][2]		[r1][0][4]	[r1][1][4]
((c+1)*j+i)*2+1		[r1][0][1]	[r1][1][1]	[r1][0][3]	[r1][1][3]		[r1][0][5]	[r1][1][5]
		..						
((c+1)*j+(i+1))*2		[r1][0][2]	[r1][0][3]	[r1][2][2]	[r1][2][3]		[r1][4][2]	[r1][4][3]
((c+1)*j+(i+1))*2+1		[r1][1][2]	[r1][1][3]	[r1][2][3]	[r1][3][3]		[r1][5][2]	[r1][5][3]
		..						
((c+1)*(j+1)+i)*2		[r1][0][4]	[r1][0][5]	[r1][4][2]	[r1][5][2]		[r1][4][4]	[r1][4][5]
((c+1)*(j+1)+i)*2+1		[r1][1][4]	[r1][1][5]	[r1][4][3]	[r1][5][3]		[r1][4][5]	[r1][5][5]

Fig. 11. Conjugación de la matriz de rigidez o de masa local en la matriz rigidez global para el elemento tipo 2
Fuente: Autores

$$[k_1] = \begin{bmatrix} a & b & c & d & e & f \\ b & g & h & i & j & k \\ c & h & l & m & n & o \\ d & i & m & p & q & r \\ e & j & n & q & s & t \\ f & k & o & r & t & u \end{bmatrix} \tag{9}$$

$$[k_2] = \begin{bmatrix} a' & b' & c' & d' & e' & f' \\ b' & g' & h' & i' & j' & k' \\ c' & h' & l' & m' & n' & o' \\ d' & i' & m' & p' & q' & r' \\ e' & j' & n' & q' & s' & t' \\ f' & k' & o' & r' & t' & u' \end{bmatrix} \tag{10}$$

$$[K] = \begin{bmatrix} a & b & c & d & e & f & 0 & 0 \\ b & g & h & i & j & k & 0 & 0 \\ c & h & l+a' & m+b' & n+c' & o+d' & e' & f' \\ d & i & m+b' & p+g' & q+h' & r+i' & j' & k' \\ e & j & n+c' & q+h' & s+l' & t+m' & n' & o' \\ f & k & o+d' & r+i' & t+m' & u+p' & q' & r' \\ 0 & 0 & e' & j' & n' & q' & s' & t' \\ 0 & 0 & f' & k' & o' & r' & t' & u' \end{bmatrix} \tag{11}$$

Como se dijo antes, esta conjugación se hace por medio de la ubicación de los nodos locales dentro de la matriz global de rigidez y de masa.

F. Algoritmo para generar la matriz de masa

La creación de la matriz de masa se realiza de la misma forma que con la matriz de rigidez. El algoritmo se realiza mediante la ecuación (12) [9], cuya demostración se puede observar en el Apéndice III.

$$[m] = \frac{\rho t A}{12} \begin{bmatrix} 2 & 0 & 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 & 0 & 1 \\ 1 & 0 & 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 & 1 \\ 1 & 0 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 & 0 & 2 \end{bmatrix} \tag{12}$$

Los algoritmos necesarios para realizar este proceso son iguales a los de la adición de las matrices locales de rigidez, porque en ambos casos se debe tener en cuenta que se inicia desde los elementos tipo 1 y 2, para conformar la totalidad de los nodos del dominio y se usan las mismas matrices mostradas en las Fig. 10 y Fig. 11.

La aplicación del software no se limita a un material específico, sino que sus características deben ser ingresadas por el usuario según las condiciones iniciales y del terreno. Sin embargo a modo de ejemplo se desarrolla la ecuación con una densidad de un suelo limo arcilloso [10], con valor de a 1680,37 kg/m^3. Al aplicar la densidad de ejemplo a la ecuación (12), se obtiene la ecuación (13) de la matriz de masa para el elemento:

$$[m] = \frac{1680{,}37\ (450)}{12}\begin{bmatrix} 2 & 0 & 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 & 0 & 1 \\ 1 & 0 & 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 & 1 \\ 1 & 0 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 & 0 & 2 \end{bmatrix} \tag{13}$$

G. Algoritmo de elementos finitos en el tiempo

El algoritmo que se basa en el tiempo inicia apoyándose en la ecuación de elementos finitos cuyo desarrollo se muestra en el Apéndice I, y principalmente aplica el método propuesto por Morgan [5].

Se debe partir definiendo cada uno de los elementos individuales y por lo tanto la matriz de rigidez y de masa de cada uno.

$[k_i]$: Matriz de rigidez para el elemento i
$[m_i]$: Matriz de masa para el elemento i

Luego de tener las matrices de masa y de rigidez se toma el valor ingresado por el usuario como fuerza, y se define el vector de fuerza que va a afectar el sistema.

$\{F\}$: Vector de fuerza

Al tener estos valores se realiza la operación más costosa, que consiste en invertir la matriz de masa, para el caso del software desarrollado, se importa una librería externa, Apache Commons Math [11], con funciones matemáticas, la cual se encarga de calcular la matriz inversa de masa por medio de descomposición por LU [12].

$[M]^{-1}$: Inversa de la matriz de masa

Ahora se aplica un valor de 0 al para el tiempo inicial y un vector de desplazamiento inicial de 0.

$\{d_0\}$: Vector de desplazamiento en el tiempo 0

Y se podría determinar para cada tiempo la aceleración, el desplazamiento y la velocidad en cada nodo aplicando directamente la fórmula que opera las matrices, este proceso lo realiza la librería usada Apache Commons Math.

La primera ecuación para determinar estos valores es la de la aceleración en el tiempo $t = 0$.

$$\{\ddot{d}_0\} = [M]^{-1}(\{F_0\} - [K]\{d_0\}) \tag{14}$$

Luego de esto y determinando una velocidad inicial cualquiera se puede hallar el desplazamiento en el tiempo -1 (es necesario para las siguientes ecuaciones).

$$\{d_{-1}\} = \{d_0\} - \Delta t\{\dot{d}_0\} + \frac{(\Delta t)^2}{2}\{\ddot{d}_0\} \tag{15}$$

Ahora se itera sobre las siguientes ecuaciones para determinar en el tiempo *i* el desplazamiento, la aceleración y la velocidad.

$$\begin{gathered}\{d_i\} = [M]^{-1}\{(\Delta t)^2\{F_{i-1}\} \\ +\left[2[M] - (\Delta t)^2[K]\right]\{d_{i-1}\} - [M]\{d_{i-2}\}\}\end{gathered} \tag{16}$$

$$\{\ddot{d}_i\} = [M]^{-1}(\{F_i\} - [K]\{d_i\}) \tag{17}$$

$$\{\dot{d}_i\} = \frac{\{d_{i+1}\} - \{d_{i-1}\}}{2(\Delta t)} \tag{18}$$

Esta es la solución general para cualquier modelado de ondas por medio de elementos finitos en el tiempo.

VI. Funcionamiento de EWaveFem

Tomando en cuenta las definiciones y procedimientos planteados en las secciones anteriores, se desarrolló la aplicación EWaveFem. El lenguaje utilizado en su programación fue Java [13] e implementa directamente los algoritmos planteados en el presente documento, utilizando como único apoyo adicional el uso de una librería para procesamiento de matrices [11].

EWaveFem puede ser ejecutado en cualquier máquina, sin embargo existen algunos limitantes en la cantidad de nodos que se pretenden simular, ya que el máximo de nodos e instantes de tiempo dependen directamente del hardware existente en el equipo.

Considerando la cantidad de grados de libertad la cual está dada por la ecuación (2), que es dos veces la cantidad de nodos, que a su vez se obtiene por medio del procedimiento del cálculo de nodos expuesto en el capítulo anterior del presente documento, no es viable mostrar una de estas matrices globales pues son matrices de n x n, y la cantidad depende directamente de los datos ingresados por el usuario. Dicha ecuación (2) aplica para dominios rectangulares el cual es el alcance del software desarrollado en este proyecto.

El proceso realizado por EWaveFem inicia cuando se completan los datos de entrada básicos como son: el número de iteraciones, el número de elementos que conformarán la grilla tanto a lo ancho como a lo alto, el espesor del dominio, la elasticidad del material, la densidad del material, el área de cada elemento, el módulo de Poisson y la diferencia de tiempo entre cada iteración. Luego de esto se ejecuta el algoritmo de elementos finitos y se genera un archivo con el desplazamiento, la velocidad y la aceleración de cada nodo en cada una de las diferentes iteraciones con la misma aplicación, y se genera la gráfica con la animación correspondiente a los desplazamientos.

La visualización de dicha animación se presenta en el capítulo VII subcapítulo F con los vectores que representan su velocidad, su aceleración y la fuerza que se está aplicando.

Como carga inicial en el sistema se genera un desplazamiento sobre la primera fila de nodos que consiste en una matriz de valores fijos en el aplicativo que aplican sobre la iteración en el tiempo 1, lo que luego genera el movimiento ondulatorio que va pasando entre los nodos de las filas inferiores.

VII. Resultados

A. Algoritmo de elementos finitos

Se desarrolló el algoritmo de elementos finitos que permite de manera general obtener el desplazamiento, velocidad y aceleración para cualquier nodo en cualquier tiempo de manera recursiva.
Se obtiene un software libre que implementa el algoritmo desarrollado, el cual simula una onda elástica en 2D con una malla de cualquier tamaño con cualquier cantidad de elementos y con las propiedades de los elementos definidos libremente por el usuario y con un vector de fuerza cualquiera incluyendo uno que cambia en función del tiempo.

B. Comunidad y repositorio

Se crea tanto el repositorio público https://github.com/pescamillam/EWaveFem para que cualquier persona pueda usar y modificar el código generado, como un grupo https://groups.google.com/forum/#!forum/ewavefem para tratar el futuro del proyecto y en caso de que alguien tenga dudas, tenga acceso a una comunidad de usuarios a la cual acudir.

C. Análisis a modo comparativo

Comparando el programa desarrollado con SpecFem2D, se encuentra que SpecFem2D es un software mucho más maduro, con gran cantidad de opciones, que sus archivos generados son archivos de texto e imágenes por cada tiempo que se desee capturar, pero no viene con por ejemplo una animación como actualmente lo tiene EWaveFem que permite una visualización rápida del resultado.

El software se crea en un lenguaje distinto que es ampliamente usado por las comunidades de desarrollo lo que puede lograr que sea fácilmente modificado en el futuro, además de tener una arquitectura extendible para otros tipos de elementos; con solo modificarle la matriz de masa y de rigidez ya sería posible usar otros tipos de malla en la modelación de las ondas.

D. Interfaz gráfica

En la Fig. 12 se observa la pantalla en donde se realiza la inserción de los distintos valores iniciales, estos valores corresponden a las los de todas las variables del material, tamaño de malla y otras necesarias para la ejecución del programa, y por lo tanto al resultado mostrado en las Fig. 14 y Fig. 15.

Fig. 12. Interfaz gráfica configuración parámetros iniciales.
Fuente: Autores

E. Diagrama de flujo con la implementación Algoritmo

La implementación del algoritmo está directamente escrita en código java, que se encuentra alojado en el repositorio GitHub, que puede encontrarse en https://github.com/pescamillam/EWaveFem/ y su funcionamiento general puede observarse en la Fig. 13:

Los parámetros que se le solicitan al usuario son los valores utilizados en el proceso, estos valores corresponden a:

- Ancho_dominio: indica en metros la distancia a lo ancho de la placa de suelo a simular.
- Alto_dominio: indica en metros la distancia a lo alto de la placa de suelo a simular.
- Cantidad_iteraciones: el número de pasos en el tiempo que se van a procesar
- Cantidad_ancho: el número de elementos en los que se divide el dominio horizontalmente

- Cantidad_alto: el número de elementos en los que se divide el dominio verticalmente
- Espesor: la profundidad de la placa del dominio, está dado en metros
- Elasticidad: el valor que indica la propiedad elástica del elemento, está dada en N/m^2.
- Densidad: el valor del peso sobre una unidad de volumen, está dada en Kg/m^3.
- Área: la superficie del dominio, está dada en m^2.
- Poisson: es la constante elástica que se obtiene cuando al ensanchar un elemento con respecto a lo que se encoje en la dirección perpendicular
- Delta_tiempo: la diferencia de tiempo entre cada instante, está dado en segundos
- Desplazamiento_inicial: valor en metros que se aplica sobre la matriz con el desplazamiento inicial de los nodos de la primera fila.

Fig. 13. Diagrama de flujo funcionamiento general del algoritmo
Fuente: Autores

F. Aplicación del software

En la Fig. 14 se puede ver una captura de la aplicación con todos sus nodos, sus correspondientes vectores, donde el vector azul es la velocidad y el vector rojo es la aceleración.

El software desarrollado permite ejecutar una simulación con 128 elementos y con 200 discretizaciones de tiempo en 369 segundos utilizando un máximo de 5Gb de memoria RAM.

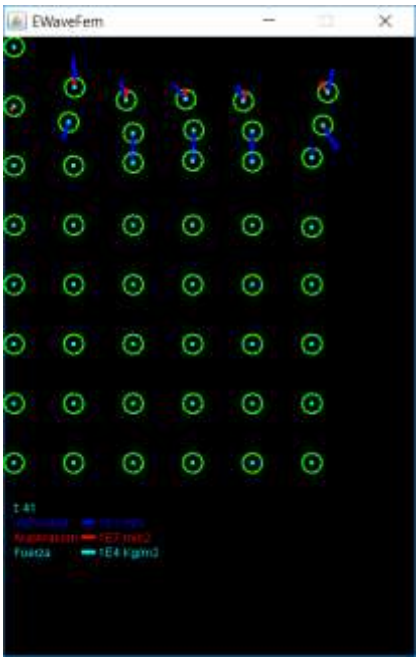

Fig. 14. EWaveFem mostrando en todos los nodos el vector de velocidad (Azul) y aceleración (rojo).
Fuente: Autores

En la Fig. 15 se muestra la visualización por nodo de los distintos vectores: color gris: desplazamiento, color verde: aceleración, color amarillo: velocidad, color cían: fuerza externa, esta imagen hace referencia al nodo 10, el cuál contando de arriba hacia abajo y de izquierda a derecha es el de la segunda fila cuarta columna y la gráfica muestra sus vectores verticales exclusivamente

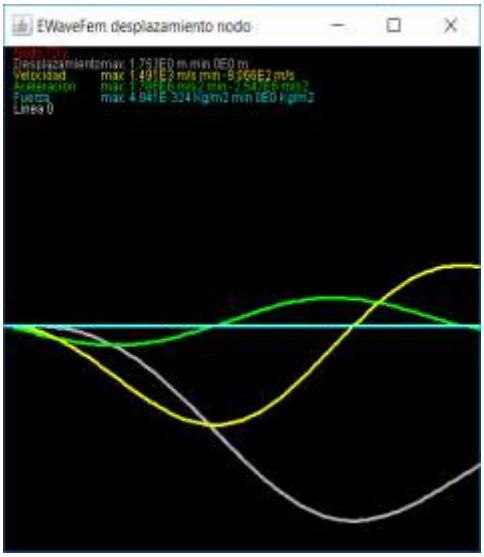

Fig. 15. Interfaz gráfica con datos de un solo nodo.
Fuente: Autores

VIII. Conclusiones

Una ventaja muy importante en un software para el modelado de propagación de ondas elásticas por el método de elementos finitos y más aún escrito en Java es la posibilidad de modificar su código para ser adaptado según las necesidades requeridas, dado que Java es de los lenguajes más ampliamente usados y el método de elementos finitos permite ajustar fácilmente las condiciones de frontera y los diversos comportamientos de cualquier grupo de nodos, ya sea en un problema específico o en un campo de acción diferente.

Se obtiene un software de modelado de ondas por elementos finitos libre y escalable susceptible de ampliación y mejoras en sus capacidades. Esta primera versión tiene la generación de las matrices de masa y rigidez, recibe todos los parámetros de la definición de los elementos incluyendo el material y sus dimensiones. Este software funciona inicialmente con mallas en 2D y en medios isotrópicos.

Mostrar una animación como parte del resultado obtenido permite apreciar rápidamente los datos para verificar el comportamiento del material simulado donde se pueden ver las ondas en movimiento causado por una fuerza aplicada sobre la fila superior.

Se identificaron los programas actuales de simulación de ondas que se presentan en el Apéndice IV donde se detectó que la mayoría son privativos y de alto costo, y que aunque hay algunos libres los lenguajes en los que estos están desarrollados no son muy comunes.

IX. Nomenclatura

n_{sd}: Número de dimensiones espaciales
$\mathbb{R}^{n_{sd}}$: n_{sd}-espacio euclideo
Ω: Dominio en$\mathbb{R}^{n_{sd}}$
u_i: Vectores de desplazamiento
$\dot{u_i}$: Vectores de velocidad
$\ddot{u_i}$: Vectores de aceleración
u_{0i}: Desplazamiento inicial
$\dot{u_{0i}}$: Velocidad inicial
Γ_{gi}: Límite de Ω_{gi}
ρ: Densidad
σ_{ij}: Tensor tensión de Cauchy
f_i: Vector de fuerza prescrito
g_i: Vector de desplazamiento limite prescrito
h_i: Vector de tracción limite prescrito
β: Diferencia vertical entre dos nodos.
x: Posición sobre el eje x del nodo.
y: Posición sobre el eje y del nodo.
w: Constante 1 menos módulo de Poisson
$[k]$: Matriz de rigidez local
$[K]$: Matriz de rigidez global
t: Grosor
E: Módulo de elasticidad
γ: Diferencia de unidades horizontales entre nodos de un elemento
v: Módulo de Poisson
r: Constante 1 menos dos veces el módulo de Poisson sobre dos
$[m]$: Matriz de masa local
$[M]$: Matriz de masa global
d: Desplazamiento
F: Vector de fuerzas externas aplicada sobre cada nodo
[N]: Matriz de función de forma
ρ: Densidad
$[B]$: Matriz de gradiente.
$[D]$: Matriz de Tensor Tensión.
$\{\varepsilon\}$: Vector tensión.
$\{f\}$: Carga total de un sistema.
$\{\sigma\}$: Relación Tensor Tensión.
Ω: Energía potencial.
U: Energía tensora.

X. Referencias

[1] Ecopetrol S.A., «Ecopetrol y Unired lanzan convocatoria de desafíos de innovación abierta,» 5 4 2015. [En línea]. Available: http://www.ecopetrol.com.co/wps/portal/es/ecopetrol-web/nuestra-empresa/sala-de-prensa/noticias/Noticias-2015/Noticias-2015/Ecopetrol-Unired-lanzan-convocatoria-de-desaflos-de-innovacion-abierta.

[2] Unired - Ecopetrol S.A., «Concurso InNovaTe: Desafíos de Innovación Abierta,» 18 9 2015. [En línea]. Available: http://unired.edu.co/images/instituciones/ICP/Innovate2015/docs/Anexo_1_Caracterizacion_geofisica.pdf.

[3] T. J. Hughes, The Finite Element Method, Linear Static and Dynamic Finite Element Analysis, New Jersey, Estados Unidos: Prentice-Hall Inc., 1987.

[4] J. M. Carcione, G. Herman y A. Ten Kroode, «Seismic Modeling,» *Geophysics,* vol. 67, nº 4, pp. 1304-1325, 2002.

[5] T. R. Morgan, Foundations of Wave Theory for Seismic Exploration, Heidelberg, Alemania: Springer, 1983.

[6] D. Komatitsch y J. P. Vilotte, «The spectral element method: an efficient tool to simulate the seismic response of 2D and 3D geological

structures,» *Bulletin of the seismological society of America,* vol. 88, nº 2, pp. 368-392, 1998.

[7] C. Galeano, J. Mantilla, C. Duque y M. Mejía, «Herramientas de software con licencia pública general para el modelado por elementos finitos,» *Revista DYNA, Universidad Nacional de Colombia, Volumen 74, Número 153,* pp. 313-324, 2007.

[8] C. Geuzaine y J. F. Remacle, «Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities.,» *International Journal for Numerical Methods in Engineering,* pp. 1-24, 2009.

[9] D. L. Logan, A first course in the finite element method, Stamford, Estados Unidos: Cengage Learning, 2011.

[10] A. Puppala, L. Mohammad y A. Allen, «Engineering behavior of lime-treated Louisiana subgrade soil,» *Transportation Research Record: Journal of the Transportation Research Board,* nº 1546, pp. 24-31, 1996.

[11] Apache, «Commons Math,» 2016. [En línea]. Available: http://commons.apache.org/proper/commons-math/. [Último acceso: 12 08 2018].

[12] R. H. Bartels y G. H. Golub, «The simplex method of linear programming using LU decomposition,» *Communications of the ACM, Volume 12, Issue 5,* pp. 266-268 , 1969.

[13] Oracle, «Java,» [En línea]. Available: https://www.java.com/es/. [Último acceso: 02 08 2018].

[14] Numérica LTDA., «Numérica desarrollando ideas,» 2018. [En línea]. Available: http://www.numerica.com.co/projects/EcoElast2D/main.htm.

[15] H. G. Castro, H. Burguener, R. R. Paz y M. E. De Bortoli, «Desarrollo de una interfaz gráfica para un código abierto de elementos finitos,» Universidad Tecnológica Nacional, Resistencia, Argentina, 2012.

[16] J. Tromp, D. Komattisch y Q. Liu, «Spectral-element and adjoint methods in seismology,» *Communications in Computational Physics,* vol. 3, nº 1, pp. 1-32, 2008.

[17] Computational Infrastructure for Geodynamics, *SPECFEM2D (Versión 7.0.0) {Software},* Obtenido de:, University of California: https://geodynamics.org/cig/software/specfem2d/, 2018.

APÉNDICE I. FORMULACIÓN DE LA ECUACIÓN DE ELEMENTOS FINITOS

El método de elementos finitos ha sido ampliamente usado desde que se crearon los primeros computadores. Se basa en que un problema de parámetros distribuidos sobre un dominio, el cual es difícil de resolver por métodos analíticos, puede ser simplificado por medio de un proceso de desratización, en donde se calculan los valores de la variable incógnita solo en ciertos puntos llamados nodos.

La sustentación de este método es la siguiente: (tomado de Hughes [3])

Aunque lo que se quiere solucionar es un problema transitorio en dos dimensiones para simplificar inicialmente se parte de un problema estacionario en una sola dimensión representado por la siguiente ecuación diferencial:

$$d^2 u/dx^2 + f = 0 \tag{19}$$

Donde f es una función definida en el intervalo [0, 1] y tiene las condiciones de frontera

$$u(1) = g \tag{20}$$

$$-du/dx\,(0) = h \tag{21}$$

Donde g y h son constantes
A partir de estas se busca la solución con la ecuación variacional

$$\int_0^1 w_{,x} u_{,x} dx = \int_0^1 wf dx + w(0)h \tag{22}$$

Integrando por partes se obtiene la ecuación

$$0 = \int_0^1 w\left(u_{,xx} + f\right) dx + w(0)\left[u_{,x}(0) + h\right] \tag{23}$$

Usando la ecuación de Galerkin se encuentra la siguiente ecuación

$$\sum_{B=1}^{n} a(N_A, N_B) d_B = (N_A, f) + N_A(0)h - a(N_A, N_{n+1})g \tag{24}$$

A partir de la cual se puede reescribir

$$K_{AB} = a(N_A, N_B) \tag{25}$$

$$F_A = (N_A, f) + N_A(0)h - a(N_A, N_{n+a})g \tag{26}$$

Obteniendo la ecuación

$$\sum_{B=1}^{n} K_{AB} d_B = F_A,\ A=1, 2, \dots, n \tag{27}$$

La cual se puede simplificar con notación matricial

$$K = [K_{AB}] = \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix} \tag{28}$$

$$F = \{F_A\} = \begin{Bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{Bmatrix} \tag{29}$$

$$d = \{d_B\} = \begin{Bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{Bmatrix} \tag{30}$$

Con esto se obtiene la formula general que se expresa de la forma:

$$F = kd \tag{31}$$

Donde *d* es el vector de desplazamientos en cada nodo, que es el dato de interés, *k* es la matriz de rigidez y F es la fuerza a la que está sometido cada nodo.

APÉNDICE II. FORMULACIÓN DE LA MATRIZ LOCAL DE RIGIDEZ

Esta matriz se halla a partir de la siguiente formulación según Logan [9]:

Al tomar uno de los elementos triangulares, sobre los cuales se busca definir el elemento general se obtienen sus grados de libertad.

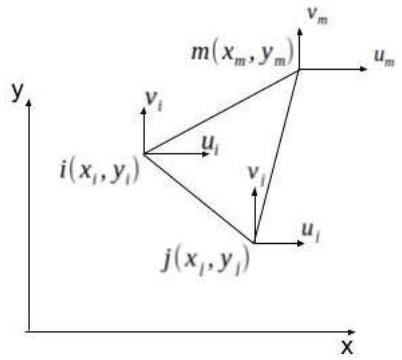

Fig. 16. Elemento triangular con grados de libertad.
Fuente: [9]

En la Fig. 16 se ven los diferentes nodos del elemento, i, j, m con sus respectivas posiciones identificadas por $i(x_i, y_i) j(x_j, y_j) m(x_m, y_m)$y con los grados de libertad $v_i u_i v_j u_j v_m u_m$ sobre cada nodo respectivamente.

Con esta definición se pueden generar las funciones lineales de movimiento

$$\begin{aligned} u(x,y) &= a_1 + a_2 x + a_3 y \\ v(x,y) &= a_4 + a_5 x + a_6 y \end{aligned} \tag{32}$$

Y con estas dos funciones se puede obtener la función de desplazamiento general Ψ

$$\begin{aligned} \Psi &= \begin{Bmatrix} u(x,y) = a_1 + a_2 x + a_3 y \\ v(x,y) = a_4 + a_5 x + a_6 y \end{Bmatrix} \\ &= \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{Bmatrix} \end{aligned} \tag{33}$$

Con esta función se puede obtener el valor de $a_x$tomando las diferentes funciones de desplazamiento para cada nodo

$$\begin{aligned} u_i = u(x_i, y_i) &= a_1 + a_2 x_i + a_3 y_i \\ u_j = u(x_j, y_j) &= a_1 + a_2 x_j + a_3 y_j \\ u_m = u(x_m, y_m) &= a_1 + a_2 x_m + a_3 y_m \\ v_i = v(x_i, y_i) &= a_4 + a_5 x_i + a_6 y_i \\ v_j = v(x_j, y_j) &= a_4 + a_5 x_j + a_6 y_j \\ v_m = v(x_m, y_m) &= a_4 + a_5 x_m + a_6 y_m \end{aligned} \tag{34}$$

Tomando las primeras 3 ecuaciones se tendría:

$$\begin{Bmatrix} u_i \\ u_j \\ u_m \end{Bmatrix} = \begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_m & y_m \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} \tag{35}$$

A partir de la cual se obtiene que

$$[x]^{-1} = \frac{1}{2A} \begin{bmatrix} \alpha_i & \alpha_j & \alpha_m \\ \beta_i & \beta_j & \beta_m \\ \gamma_i & \gamma_j & \gamma_m \end{bmatrix} \tag{36}$$

Donde

$$2A = \begin{vmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_m & y_m \end{vmatrix} \tag{37}$$

A es el área del triángulo y

$$\begin{aligned} \alpha_i &= x_j y_m - x_m y_j & \alpha_j &= x_m y_i - x_i y_m & \alpha_m &= x_i y_j - y_i x_j \\ \beta_i &= y_j - y_m & \beta_j &= y_m - y_i & \beta_m &= y_i - y_j \\ \gamma_i &= x_m - x_j & \gamma_j &= x_i - x_m & \gamma_m &= x_j - x_i \end{aligned} \tag{38}$$

Ahora se puede derivar la función de desplazamiento$u(x, y)$de $\{\Psi\}$ Se obtendría la función:

$$\{u\} = \frac{1}{2A} [1 \quad x \quad y] \begin{bmatrix} \alpha_i & \alpha_j & \alpha_m \\ \beta_i & \beta_j & \beta_m \\ \gamma_i & \gamma_j & \gamma_m \end{bmatrix} \begin{Bmatrix} u_i \\ u_j \\ u_m \end{Bmatrix} \tag{39}$$

La cual daría como resultado:

$$\begin{aligned} u(x,y) = \frac{1}{2A} \{ &(\alpha_i + \beta_i x + \gamma_i y) u_i \\ &+ (\alpha_j + \beta_j x + \gamma_j y) u_j + (\alpha_m + \beta_m x + \gamma_m y) u_m \} \end{aligned} \tag{40}$$

De la misma manera se resuelve para v(x, y) así:

$$\begin{aligned} v(x,y) = \frac{1}{2A} \{ &(\alpha_i + \beta_i x + \gamma_i y) v_i \\ &+ (\alpha_j + \beta_j x + \gamma_j y) v_j + (\alpha_m + \beta_m x + \gamma_m y) v_m \} \end{aligned} \tag{41}$$

Para simplificar estas dos ecuaciones se puede realizar la siguiente definición

$$\begin{aligned} N_i &= \frac{1}{2A}(\alpha_i + \beta_i x + \gamma_i y) \\ N_j &= \frac{1}{2A}(\alpha_j + \beta_j x + \gamma_j y) \\ N_m &= \frac{1}{2A}(\alpha_m + \beta_m x + \gamma_m y) \end{aligned} \tag{42}$$

Con lo que se podrían reescribir las ecuaciones (22) y (23)

$$\begin{aligned} u(x,y) &= N_i u_i + N_j u_j + N_m u_m \\ v(x,y) &= N_i v_i + N_j v_j + N_m v_m \end{aligned} \tag{43}$$

Lo cual podría escribirse de manera abreviada en forma matricial como

$$\{\Psi\} = \begin{bmatrix} N_i & 0 & N_j & 0 & N_m & 0 \\ 0 & N_i & 0 & N_j & 0 & N_m \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \\ u_m \\ v_m \end{Bmatrix} \tag{44}$$

La cual a su vez abreviada puede ser vista como

$$\{\Psi\} = [N]\{d\} \tag{45}$$

Ahora definiendo el vector de deformaciones unitarias

$$\{\varepsilon\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{Bmatrix} \dfrac{\delta u}{\delta x} \\ \dfrac{\delta v}{\delta y} \\ \dfrac{\delta u}{\delta y} + \dfrac{\delta v}{\delta x} \end{Bmatrix} \tag{46}$$

Donde usando la ecuación (14) se tendría que

$$\varepsilon_y = a_6 \gamma_{xy} = a_3 + a_5 \tag{47}$$

Ahora se puede derivar la ecuación de movimiento así:

$$u_{,x} = N_{i,x} u_i + N_{j,x} u_j + N_{m,x} u_m \tag{48}$$

Y derivando las funciones de forma se tendría:

$$N_{i,x} = \frac{1}{2A}\frac{\delta}{\delta x}(\alpha_i + \beta_i x + \gamma_i y) = \frac{\beta_i}{2A} \tag{49}$$

De la misma forma se obtiene:

$$N_{j,x} = \frac{\beta_j}{2A} N_{m,x} = \frac{\beta_m}{2A} \tag{50}$$

Con lo que se tendría la formulas:

$$\begin{aligned} \frac{\delta u}{\delta x} &= \frac{1}{2A}(\beta_i u_i + \beta_j u_j + \beta_m u_m) \\ \frac{\delta v}{\delta y} &= \frac{1}{2A}(\gamma_i v_i + \gamma_j v_j + \gamma_m v_m) \\ \frac{\delta u}{\delta y} + \frac{\delta v}{\delta x} &= \frac{1}{2A}(\gamma_i u_i + \beta_i v_i + \gamma_j u_j + \beta_j v_j + \gamma_m u_m + \beta_m v_m) \end{aligned} \tag{51}$$

Estas fórmulas pueden pasarse a modo matricial indicando el vector tensor así:

$$\{\varepsilon\} = \frac{1}{2A}\begin{bmatrix} \beta_i & 0 & \beta_j & 0 & \beta_m & 0 \\ 0 & \gamma_i & 0 & \gamma_j & 0 & \gamma_m \\ \gamma_i & \beta_i & \gamma_j & \beta_j & \gamma_m & \beta_m \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \\ u_m \\ v_m \end{Bmatrix} \tag{52}$$

Esto simplificado nos daría la formula

$$\{\varepsilon\} = [B]\{d\} \tag{53}$$

Ahora la relación de Tensor Tensión se formularia como

$$\{\sigma\} = [D]\{\varepsilon\} \tag{54}$$

Lo que sería igual a

$$\{\sigma\} = [D][B]\{d\} \tag{55}$$

Ahora usando el principio de la mínima energía potencial y teniendo en cuenta que la energía tensora es

$$U = \frac{1}{2}\iiint_V \{\varepsilon\}^T \{\sigma\} dV \tag{56}$$

La energía potencial está dada por

$$\Omega_b = -\iiint_V \{\Psi\}^T \{X\} dV \tag{57}$$

Con esto se tiene la carga total del sistema {f} que es

$$\{f\} = \iiint_V [N]^T X dV + P + \iint_S [N_S]^T \{T_S\} dS \tag{58}$$

Con esto se obtiene

$$[k] = \iiint_V [B]^T [D][B] dV \tag{59}$$

Para un elemento con grosor constante se sacaría la constante quedando

$$[k] = t \iint_A [B]^T [D][B] dxdy \tag{60}$$

Así que la matriz de rigidez es igual a:

$$[k] = tA[B]^T [D][B] \tag{61}$$

Donde para un elemento triangular la matriz de Tensor Tensión [D] es igual a:

$$[D] = \frac{E}{(1+v)(1-2v)} \begin{bmatrix} 1-v & v & 0 \\ v & 1-v & 0 \\ 0 & 0 & \frac{1-2v}{2} \end{bmatrix} \tag{62}$$

Y la matriz de gradiente [B] es igual a:

$$[B] = \left[[B_i] [B_j] [B_m] \right] \tag{63}$$

Donde:

$$[B_i] = \frac{1}{2A} \begin{bmatrix} \beta_i & 0 \\ 0 & \gamma_i \\ \gamma_i & \beta_i \end{bmatrix} \tag{64}$$

$$[B_j] = \frac{1}{2A} \begin{bmatrix} \beta_j & 0 \\ 0 & \gamma_j \\ \gamma_j & \beta_j \end{bmatrix} \tag{65}$$

$$[B_m] = \frac{1}{2A} \begin{bmatrix} \beta_m & 0 \\ 0 & \gamma_m \\ \gamma_m & \beta_m \end{bmatrix} \tag{66}$$

Se define la matriz de rigidez para un elemento triangular:

$$[k] = \frac{tE}{4A(1+v)(1-2v)} \begin{bmatrix} \beta_i & 0 & \gamma_i \\ 0 & \gamma_i & \beta_i \\ \beta_j & 0 & \gamma_j \\ 0 & \gamma_j & \beta_j \\ \beta_m & 0 & \gamma_m \\ 0 & \gamma_m & \beta_m \end{bmatrix} \times \begin{bmatrix} 1-v & v & 0 \\ v & 1-v & 0 \\ 0 & 0 & \frac{1-2v}{2} \end{bmatrix} \begin{bmatrix} \beta_i & 0 & \beta_j & 0 & \beta_m & 0 \\ 0 & \gamma_i & 0 & \gamma_j & 0 & \gamma_m \\ \gamma_i & \beta_i & \gamma_j & \beta_j & \gamma_m & \beta_m \end{bmatrix} \tag{67}$$

APÉNDICE III. FORMULACIÓN DE LA MATRIZ DE MASA

Para la formulación de la matriz de masa primero se toma la matriz de función de forma [9]:

$$[N] = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 \end{bmatrix} \tag{68}$$

Y la fórmula de la matriz de masa del elemento

$$[m] = \iiint_V \rho [N]^T [N] dV \tag{69}$$

Para obtener la matriz de masa del elemento triangular simple, teniendo en cuenta que:

$$dV = tdA \int_A N_1^2 \, dA = \frac{1}{6} A \int_A N_1 N_2 dA = \frac{1}{12} A \tag{70}$$

Se tiene que la matriz de masa sería:

$$[m] = \frac{\rho t A}{12} \begin{bmatrix} 2 & 0 & 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 & 0 & 1 \\ 1 & 0 & 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 & 1 \\ 1 & 0 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 & 0 & 2 \end{bmatrix} \tag{71}$$

APÉNDICE IV. SOFTWARE IDENTIFICADO PARA SIMULACIÓN POR ELEMENTOS FINITOS

Elmer es un software libre iniciado por una universidad finlandesa y actualmente desarrollado en conjunto por varias universidades, empresas, investigadores independientes y personas interesadas en el tema de desarrollo de software para modelos numéricos. Este software se encuentra desarrollado principalmente en Fortran, inicialmente por varias universidades finlandesas y luego se presentó como código abierto por más universidades, corporaciones y personas individuales.

EcoElast2D. Es un software propiedad de Numerica, este software realiza la simulación de la propagación de ondas elásticas a partir del método de diferencias finitas. Actualmente se puede conseguir comercialmente por medio de la empresa http://www.numerica.com.co/. En la Fig. 17 se aprecia una muestra de los resultados que arroja el software Ecoelast 2D. Su código es cerrado pero es un buen ejemplo de software desarrollado localmente con el mismo propósito del proyecto de esta tesis.

3Dhp90. La universidad de Texas ha desarrollado software para la simulación de propagación de ondas por medio del método de elementos finitos. Este es un software privativo que ha sido desarrollado con el patrocinio de grandes marcas como Shell, Petrobras, ExxonMobil.

Este software fue desarrollado en Fortran y ha sido objeto de estudio en diversos proyectos a nivel de doctorado en la universidad mencionada.

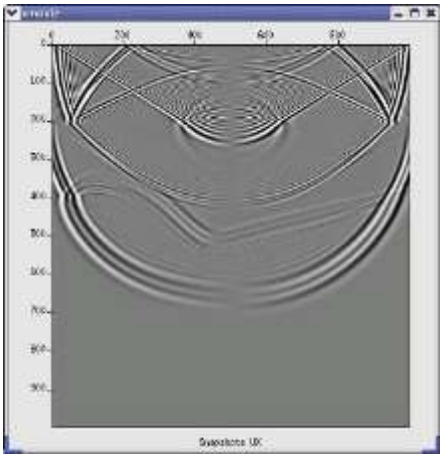

Fig. 17. ecoElast2D
Fuente: [14]

PETSc-FEM un software de modelado por elementos finitos realizado en Argentina y su posterior interfaz gráfica [15] siendo esta última desarrollada en Java, aunque esta separación de interfaz y núcleo puede hacer que con futuras versiones del núcleo se vuelva incompatible la interfaz.

SpecFem2D. Software abierto de simulación de ondas sísmicas escrito en Fortran [16], este software incluye la implementación de fronteras absorbentes, se mantiene un desarrollo activo aunque no es tan fuerte como el que tiene el software Elmer, sino que este SpecFem2D tiene compatibilidad para simulaciones de petróleo y gas, además tiene un software similar llamado SpecFem3D el cual está siendo más activamente desarrollado, imágenes del programa en funcionamiento pueden ser vistas en la Fig. 18 y la Fig. 19.

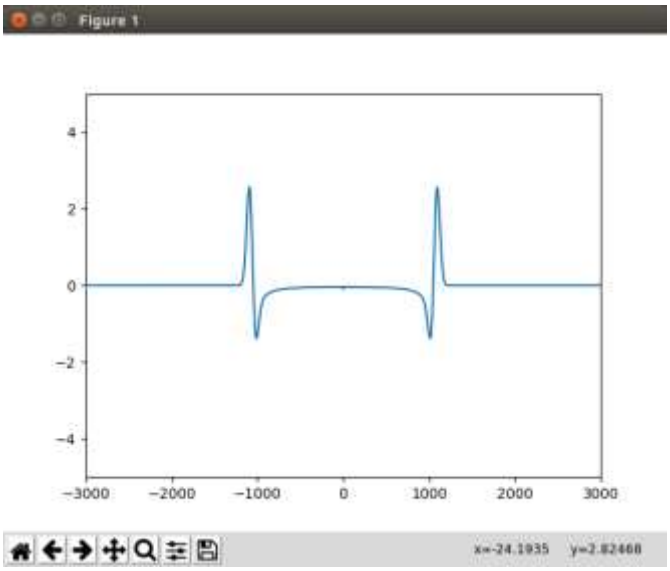

Fig. 18. specfem1D
Fuente: Autor

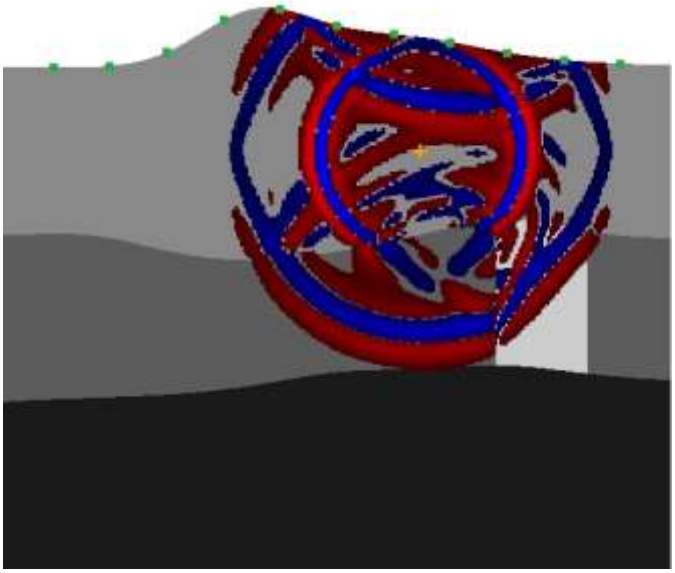

Fig. 19. specfem2D
Fuente: [17]