DESARROLLO DE UNA APLICACIÓN POR EL MÉTODO DE LOS MÍNIMOS CUADRADOS MÓVILES PARA RESOLVER EL PROBLEMA DE FLEXIÓN EN UNA VIGA SIMPLEMENTE APOYADA

MARÍA JULIANA SANABRIA MUÑOZ
ELKIN HUMBERTO VILLALOBOS GÓMEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍA MECÁNICA
BUCARAMANGA
2010

DESARROLLO DE UNA APLICACIÓN POR EL MÉTODO DE LOS MÍNIMOS CUADRADOS MÓVILES PARA RESOLVER EL PROBLEMA DE FLEXIÓN EN UNA VIGA SIMPLEMENTE APOYADA

MARÍA JULIANA SANABRIA MUÑOZ
ELKIN HUMBERTO VILLALOBOS GÓMEZ

Trabajo de grado para optar al título de
Ingeniero Mecánico

Director:
PEDRO JOSÉ DÍAZ GUERRERO
Ingeniero Mecánico

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍA MECÁNICA
BUCARAMANGA
2010

A mi madre quien con su incansable lucha, aprendí el gran valor de una mujer, quien a pesar de las adversidades puede salir adelante gracias a su fortaleza y su espíritu emprendedor.

A mi padre que a pesar de no estar presente en cuerpo, se que está a mi lado en espíritu cada vez que lo necesito, dándome su fortaleza y ganas de verme crecer.

A mis hermanos quienes de una u otra forma han sido mi referente para alcanzar mis propósitos.

A mi sobrina por ser uno de mis motivos para salir adelante.

Y a mi novio por su apoyo incondicional y voz de aliento cada vez que lo necesito.

Juliana Sanabria

A mi madre por su inmenso amor e infinito cariño, quien hizo que aun en la distancia su manto de amor siempre me cubriera, brindándome así el regalo más grande que me ha podido dar la vida y es el tenerla a ella como mi madre.

A mi padre que con sus palabras precisas y sabios consejos me hizo creer en mí mismo aun en los momentos más difíciles de mi vida.

A mis abuelos quienes me han hecho sentir siempre como un hijo más.

A mis hermanas y hermanos de quienes he recibidos palabras de amor y de respeto.

A mis tíos (as) y primos (as) de quienes siempre he recibido palabras de aliento.

Y a mi novia por haberme hecho sentir un hombre afortunado al tenerla a ella como compañera todos estos años.

Elkin Villalobos

AGRADECIMIENTOS

Al profesor y director de este proyecto de grado Ingeniero Pedro José Díaz Guerrero por su oportuno apoyo y dedicación durante el desarrollo de este proyecto.

Al profesor David Alfredo Fuentes Díaz por su desinteresada colaboración al enseñarnos el manejo de las liberarías MecLib de la escuela de Ingeniería Mecánica.

A todos los profesores de la escuela de ingeniería mecánica, quienes hicieron de nosotros personas con alta capacidad crítica, analítica e investigativa.

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

RESUMEN

TÍTULO: DESARROLLO DE UNA APLICACIÓN POR EL MÉTODO DE LOS MÍNIMOS CUADRADOS MÓVILES PARA RESOLVER EL PROBLEMA DE FLEXIÓN EN UNA VIGA SIMPLEMENTE APOYADA [*]

AUTORES:
María Juliana Sanabria Muñoz.
Elkin Humberto Villalobos Gomez.[**]

PALABRAS CLAVES:
Métodos numéricos, Diferencias finitas, Elementos finitos, Galerkin libre de malla, Mínimos cuadrados móviles.

DESCRIPCIÓN:
Los métodos libres de malla son en la actualidad una alternativa para superar las limitaciones que presentan los métodos numéricos tradicionales (Método de Elementos Finitos, Diferencias Finitas o Volúmenes Finitos) en el área de la mecánica de sólidos y mecánica de fluidos. El principal atractivo de los métodos libres de malla en proyectos donde se hace necesario un análisis numérico para aproximar la solución del fenómeno a estudiar, es que no se requiere la construcción de una malla, por el contrario el dominio está representado por un conjunto de nodos en los cuales se definen subdominios, lo cual permite una mayor flexibilidad a la hora de resolver problemas en los cuales las conexiones entre los nodos varían al avanzar el problema por ejemplo: crecimiento de grietas, fluidos en movimiento ó deformaciones elevadas. Con el ánimo de mostrar la validez y la exactitud del método de Galerkin Libre de Malla en primera instancia para resolver el problema de flexión en una viga simplemente apoyada se muestra la formulación mínima necesaria para dar solución a este fenómeno por medio de un análisis teórico, el método de diferencias finitas, el método de elementos finitos y el método de Galerkin libre de malla con el fin de establecer un primer parámetro de comparación entre las soluciones. Para tal fin esta implementación se hará en el lenguaje de programación C++, utilizando una herramienta para el desarrollo de software numérico llamada MecLib que consiste en una colección de librerías escritas en el lenguaje de programación C++.

[*] Trabajo de Grado
[**] Universidad Industrial de Santander, Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería Mecánica, Director: Ing. Mecánico, M.sc. Pedro José Díaz Guerrero

SUMMARY

TITLE: DEVELOPMENT OF AN APPLICATION BY THE MOVING LEAST SQUARES METHOD TO RESOLVE THE PROBLEM OF BENDING IN A SIMPLY SUPPORTED BEAM [*]

AUTHORS:
María Juliana Sanabria Muñoz.
Elkin Humberto Villalobos Gomez.[**]

KEY WORDS:
Numerical methods, finite differences, finite elements, mesh-free Galerkin, Moving Least squares.

DESCRIPTION:
The mesh-free methods are now an alternative to overcome the limitations of traditional numerical methods (Finite Element Method, Finite Difference and Finite Volume) in the area of solid mechanics and fluid mechanics. The main attraction of the mesh-free methods in projects where it is necessary numerical analysis to approximate the solution of the phenomenon to study, it is not requires construction of a mesh, on the contrary the domain is represented by a set of nodes which are defined in the subdomains, which allows greater flexibility to solve problems in which the connections between nodes vary in moving the problem, for example, crack growth, fluid flow or large deformation. In order to show the validity and accuracy of the mesh-free Galerkin method in first instance to resolve the problem of bending in a simply supported beam shows the minimum necessary formulation to solve this phenomenon by means of a theoretical analysis, finite difference method, finite element method and mesh-free Galerkin method in order to establish a first parameter of comparison between solutions. For this purpose this implementation will be in the C++ programming language using a development tool of numerical software called MecLib which is a collection of libraries written in C++ programming language.

[*] Thesis Project
[**] Universidad Industrial de Santander, Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería Mecánica, Director: Ing. Mecánico, M.sc. Pedro José Díaz Guerrero

INTRODUCCIÓN

En la actualidad la mayoría de los procesos de diseño e ingeniería de productos y procesos requieren del uso de herramientas computacionales. Con el pasar de los años estos problemas son cada vez de mayor envergadura, lo que conlleva a la investigación del cálculo de una solución óptima, apoyándose en herramientas CAD (diseño asistido por ordenador), que permiten modelar los sistemas a diseñar y analizar las complejas ecuaciones diferenciales, que gobiernan su funcionamiento.

Para tal fin se han desarrollado métodos analíticos para resolver dichas ecuaciones diferenciales tales como los métodos de diferencias finitas (FDM) y elementos finitos (FEM). Sin embargo para algunos problemas de alta complejidad, como el inicio de la fractura en un componente estructural sometido a carga dinámicas, la solución con dichos métodos se hace inalcanzable. Por lo tanto es en este punto, donde nace la necesidad de plantear nuevos métodos de solución que solventen dicha limitación.

Es por este motivo que se han desarrollado los métodos libres de malla, que como su nombre lo indica, la ausencia de una malla hacen que el dominio no sea discretizado como en los métodos tradicionales, si no representado por un conjunto de nodos en los cuales se definen subdominios, lo cual permite una mayor flexibilidad a la hora de resolver problemas en los que las conexiones entre los nodos varían al avanzar el problema por ejemplo: crecimiento de grietas, fluidos en movimiento, deformaciones elevadas.

Estos métodos libres de malla, pueden ser fácilmente desarrollados ya que no requieren involucrar el concepto de conectividad, por tal razón no hay necesidad

de dar ninguna información acerca de la relación entre nodos, proporcionando flexibilidad al añadir o eliminar nodos cuando y donde sea necesario.

La adaptación del enmallado para una gran variedad de problemas 1D, 2D o 3D, incluyendo lineales o no lineales así como el análisis de esfuerzos estáticos o dinámicos pueden ser efectivamente tratados por el método libre de malla de una forma relativamente simple.

1. SOLUCIÓN TEÓRICA DE UNA VIGA A FLEXIÓN SIMPLEMENTE APOYADA

La decisión de analizar una viga simplemente apoyada, surge de la necesidad que se presenta en la mayoría de problemas de la ingeniería mecánica relacionados con el área de modelamiento estructural, que se basan en el cálculo del desplazamiento generado al aplicar algún tipo de carga, es por esta razón que presentaremos a continuación la solución teórica de una viga simplemente apoyada utilizando los tipos de cargas más comunes (puntual, rectangular y triangular uniformemente distribuida) con el ánimo de establecer un parámetro de comparación entre la solución teórica y numérica.

1.1 ANÁLISIS TEÓRICO

Para el análisis de una viga simplemente apoyada utilizamos el método de tramos, ya que se necesita calcular el momento, en función de la posición, y así poder determinar el desplazamiento en diferentes puntos de la viga. Este método comienza con la determinación de las fuerzas que actúan sobre los apoyos, para posteriormente obtener las ecuaciones de momentos en función de la posición, por medio del análisis de tramos, de acuerdo con esto y teniendo en cuenta que la ecuación diferencial de la elástica para una viga simplemente apoyada es:

$$EI\frac{d^2u}{dx^2} - M(x) = 0 \qquad (1)$$

Donde u es el desplazamiento de la viga, y remplazando cada ecuación de momentos en la ecuación (1), se obtiene finalmente una ecuación para u en función de la posición deseada.

1.1.1 Viga Simplemente Apoyada Sometida a Carga Puntual:

Figura 1. Viga simplemente apoyada sometida a carga puntal.

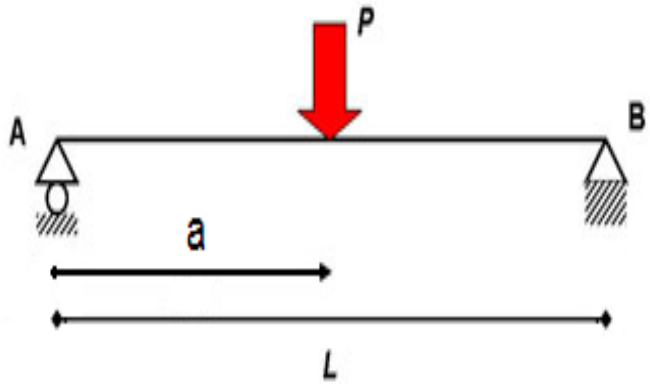
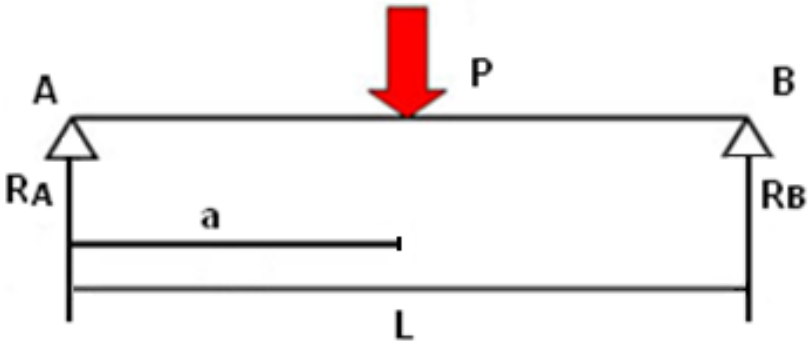


Figura 2. Diagrama de cuerpo libre de una viga simplemente apoyada sometida a flexión.



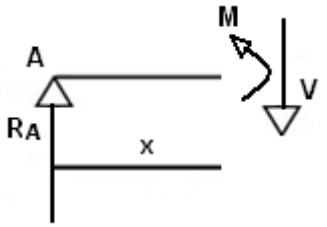
Para el cálculo de las reacciones en la viga se hace $\sum F_y$ y $\sum M_A$, obteniendo que:

$$R_B = \frac{Pa}{L} \quad y \quad R_A = \frac{P}{L}(L - a) \tag{2}$$

Ahora se procede a calcular los momentos sobre la viga mediante el análisis de cada tramo de la siguiente forma:

Para $x \leq a$

Figura 3. Diagrama de cuerpo libre para todo $\boldsymbol{x \leq a}$.



Siguiendo el análisis correspondiente se tiene que:

$$V = \frac{P}{L}(L - a) \tag{3}$$

$$M(x) = \frac{Px}{L}(L - a) \tag{4}$$

Remplazando la ecuación (4) en (1) se tiene:

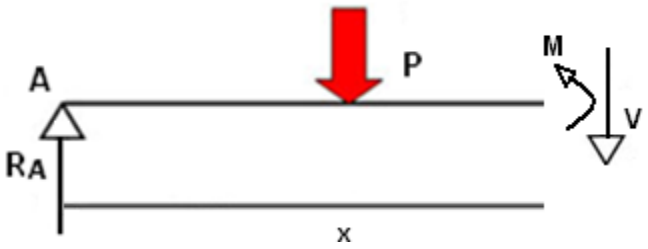
$$EI\frac{d^2u_1}{dx^2} = \frac{Px}{L}(L - a)$$

$$EI\frac{du_1}{dx} = \frac{Px^2}{2L}(L - a) + C_1 \tag{5}$$

$$EIu_1 = \frac{Px^3}{6L}(L - a) + C_1x + C_2$$

Para $x > a$

Figura 4. Diagrama de cuerpo libre para todo $\boldsymbol{x > a}$.



Se tiene que:

$$V = -\frac{Pa}{L} \tag{6}$$

$$M(x) = \frac{Pa}{L}(L - x) \tag{7}$$

Y remplazando la ecuación (7) en (1):

$$EI\frac{d^2u_2}{dx^2} = \frac{Pa}{L}(L - x)$$

$$EI\frac{du_2}{dx} = \frac{Pa}{L}\left(xL - \frac{x^2}{2}\right) + C_3 \tag{8}$$

$$EIu_2 = \frac{Pa}{L}\left(\frac{x^2L}{2} - \frac{x^3}{6}\right) + Cx_3 + C_4$$

Aplicando las siguientes condiciones de frontera:

$$\begin{gathered} u = 0\ para\ x = 0 \\ du_1 = du_2\ para\ x = a \\ u_1 = u_2 \quad para\ x = a \\ u = 0\ para\ x = L \end{gathered} \tag{9}$$

Finalmente se tienen las ecuaciones para el desplazamiento de la viga:

$$u_1 = \frac{P(L-a)}{6LEI}(-x^3 + (2aL - a^2)x) \tag{10}$$

$$u_2 = \frac{P}{6EI} * \left(-\frac{(L-a)x^3}{L} + (x-a)^3 + (L-a)\frac{2aL - a^2}{L}x\right) \tag{11}$$

1.1.2 Viga simplemente apoyada sometida a carga rectangular uniformemente distribuida:

Figura 5. Viga simplemente apoyada sometida a carga rectangular uniformemente distribuida.

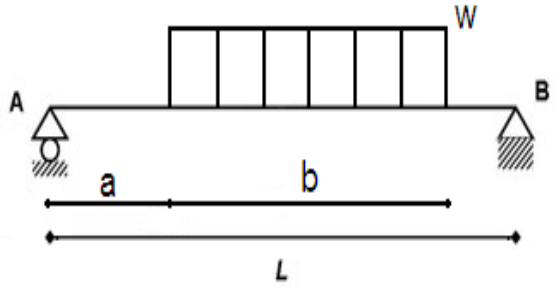
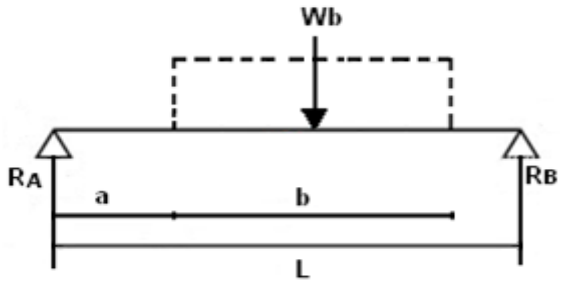


Figura 6. Diagrama de cuerpo libre de una viga simplemente apoyada sometida a flexión.



Siguiendo el análisis que se hizo para una viga simplemente apoyada sometida a carga puntual se tiene que.

Tabla 1. Análisis por tramos para la viga simplemente apoyada sometida a carga rectangular uniformemente distribuida.

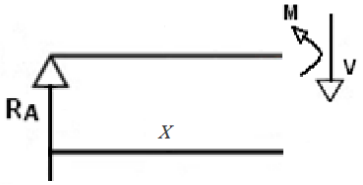
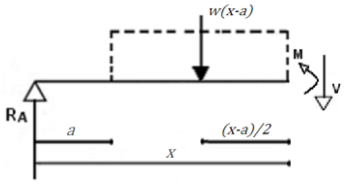
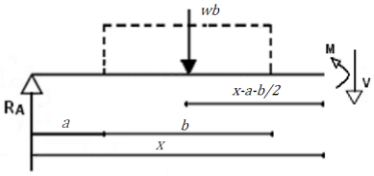
ANÁLISIS POR TRAMOS PARA LA VIGA SIMPLEMENTE APOYADA SOMETIDA A CARGA RECTANGULAR UNIFORMEMENTE DISTRIBUIDA	
Condiciones de frontera	$u = 0\ para\ x = 0$ $du_1 = du_2\ para\ x = a$ $u_1 = u_2\ \ para\ x = a$ $du_2 = du_3\ para\ x = a + b$ $u_2 = u_3\ \ para\ x = a + b$ $u = 0\ para\ x = L$ (12)
$x \leq a$	
	$M(x) = \frac{wb}{L}\left(L - a - \frac{b}{2}\right)x$ (13) $EI\frac{d^2u_1}{dx^2} = \frac{wb}{L}\left(L - a - \frac{b}{2}\right)x$ $EI\frac{du_1}{dx} = \frac{wb}{2L}\left(L - a - \frac{b}{2}\right)x^2 + C_1$ (14) $EIu_1 = \frac{wb}{6L}\left(L - a - \frac{b}{2}\right)x^3 + C_1x + C_2$ $u_1 = \frac{x}{EI}\left[\frac{wb}{6L}\left(L - a - \frac{b}{2.0}\right)x^2 + \frac{wab}{2}(a + b) + \frac{w}{24L}\left(a^4 - ((a + b)^4)\right) - \frac{wbL}{3}\left(a + \frac{b}{2}\right) + \frac{wb^3}{6}\right]$ (15)

Tabla 1. (Continúa)

<table>
<tr><td colspan="2">$a < x \leq a + b$</td></tr>
<tr><td> 
</td><td>$$M(x) = \frac{Wb}{L}\left(L - a - \frac{b}{2}\right)x - \frac{W(x-a)^2}{2} \quad (16)$$
$$EI\frac{d^2u_2}{dx^2} = \frac{wb}{L}\left(L - a - \frac{b}{2}\right)x - \frac{W(x-a)^2}{2}$$
$$EI\frac{du_2}{dx} = \frac{wb}{2L}\left(L - a - \frac{b}{2}\right)x^2 - \frac{W(x-a)^3}{6} + C_3 \quad (17)$$
$$EIu_2 = \frac{wb}{6L}\left(L - a - \frac{b}{2}\right)x^3 - \frac{W(x-a)^4}{24} + C_3x + C_4$$
$$u_2 = \frac{1}{EI}\left[\frac{wb}{6L}\left(L - a - \frac{b}{2}\right)x^3 - \frac{w}{24}(x-a)^4 + \frac{wab}{2}(a+b)x + \frac{w}{24L}(a^4 - (a+b)^4)x - \frac{wbL}{3}\left(a + \frac{b}{2}\right)x + \frac{wb^3}{6}x\right] \quad (18)$$</td></tr>
<tr><td colspan="2">$a + b < x \leq L$</td></tr>
<tr><td> 
</td><td>$$M(x) = \frac{wb}{L}\left(a + \frac{b}{2}\right)(L - x) \quad (19)$$
$$EI\frac{d^2u_3}{dx^2} = \frac{wb}{L}\left(a + \frac{b}{2}\right)(L - x)$$
$$EI\frac{du_3}{dx} = \frac{wb}{L}\left(a + \frac{b}{2}\right)\left(Lx - \frac{x^2}{2}\right) + C_5 \quad (20)$$
$$EIu_3 = \frac{wb}{2L}\left(a + \frac{b}{2}\right)\left(Lx^2 - \frac{x^3}{3}\right) + C_5x + C_6$$
$$u_3 = \frac{1}{EI}\left[\frac{wb}{L}\left(a + \frac{b}{2}\right)\left(\frac{L}{2} - \frac{x}{6}\right)x^2 + \frac{w}{24}(a^4 - (a+b)^4)\left(\frac{x}{L} - 1\right) - \frac{wbL}{3}\left(a + \frac{b}{2}\right)x\right] \quad (21)$$</td></tr>
</table>

1.1.3 Viga Simplemente Apoyada Sometida a Carga Triangular Uniformemente Distribuida:

Figura 7. Viga simplemente apoyada sometida a carga triangular uniformemente distribuida.

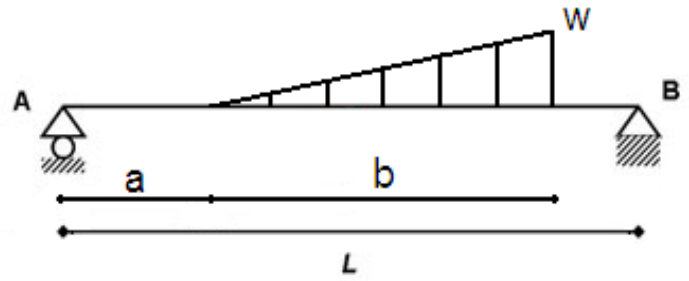
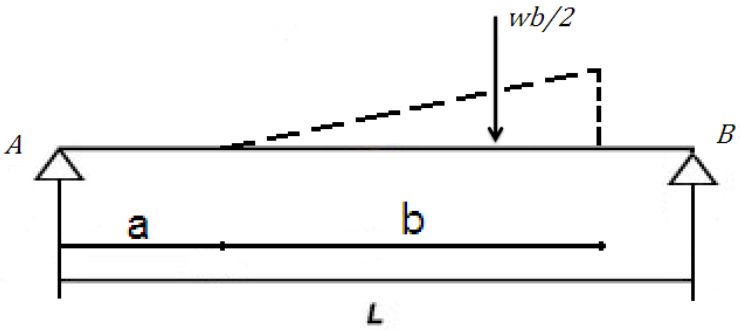


Figura 8. Diagrama de cuerpo libre de una viga simplemente apoyada sometida a flexión.



Siguiendo el análisis que se hizo para una viga simplemente apoyada sometida a carga puntual se tiene que.

Tabla 2. Análisis por tramos para la viga simplemente apoyada sometida a carga triangular uniformemente distribuida.

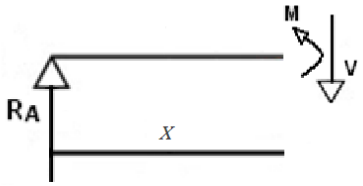
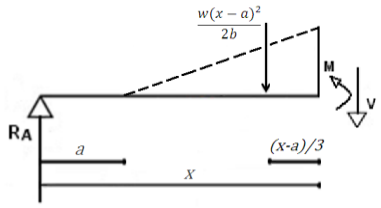
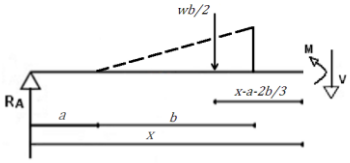
<table>
<tr><td colspan="2">ANÁLISIS POR TRAMOS PARA LA VIGA SIMPLEMENTE APOYADA SOMETIDA A CARGA TRIANGULAR UNIFORMEMENTE DISTRIBUIDA</td></tr>
<tr><td>Condiciones de frontera</td><td>$$u = 0\ para\ x = 0$$
$$du_1 = du_2\ para\ x = a$$
$$u_1 = u_2\ \ para\ x = a$$
$$du_2 = du_3\ para\ x = a + b$$
$$u_2 = u_3\ \ para\ x = a + b$$
$$u = 0\ para\ x = L$$ (22)</td></tr>
<tr><td colspan="2">$x \leq a$</td></tr>
<tr><td> 
</td><td>$$M(x) = \frac{wb}{2L}\left(L - a - \frac{2b}{3}\right)x$$ (23)
$$EI\frac{d^2u_1}{dx^2} = \frac{wb}{2L}\left(L - a - \frac{2b}{3}\right)x$$
$$EI\frac{du_1}{dx} = \frac{wb}{4L}\left(L - a - \frac{2b}{3}\right)x^2 + C_1$$ (24)
$$EIu_1 = \frac{wb}{12L}\left(L - a - \frac{2b}{3}\right)x^3 + C_1x + C_2$$
$$u_1 = \frac{x}{EI}\left[\frac{wb}{12L}\left(L - a - \frac{2b}{3}\right)x^2 + \frac{wb}{4}\left(a^2 + \frac{4ab}{3} + \frac{b^2}{9}\right) - \frac{wbL}{6}\left(a + \frac{2b}{3}\right) + \frac{wb^3}{24}\left(1 - \frac{1}{L}\left(a + \frac{4.0 * b}{5.0}\right)\right) - \frac{wba}{12L}(a + b)^2\right]$$ (25)</td></tr>
<tr><td colspan="2">$a < x \leq a + b$</td></tr>
<tr><td> 
</td><td>$$M(x) = \frac{Wb}{2L}\left(L - a - \frac{2b}{3}\right)x - \frac{w(x-a)^3}{6b}$$ (26)
$$EI\frac{d^2u_2}{dx^2} = \frac{Wb}{2L}\left(L - a - \frac{2b}{3}\right)x - \frac{w(x-a)^3}{6b}$$
$$EI\frac{du_2}{dx} = \frac{wb}{4L}\left(L - a - \frac{2b}{3}\right)x^2 - \frac{W(x-a)^4}{24b} + C_3$$ (27)
$$EIu_2 = \frac{wb}{12L}\left(L - a - \frac{2b}{3}\right)x^3 - \frac{W(x-a)^5}{120b} + C_3x + C_4$$
$$u_2 = \frac{1}{EI}\left[\frac{wb}{12L}\left(L - a - \frac{2b}{3}\right)x^3 + \frac{wb^3}{24}x + \frac{wb}{4}(a + b)\left(a + \frac{b}{3}\right)x - \frac{wbL}{6}\left(a + \frac{2b}{3}\right)x - \frac{wb^3}{24L}\left(a + \frac{4b}{5}\right)x - \frac{wab}{12L}(a + b)^2x - \frac{w}{120b}(x - a)^5\right]$$ (28)</td></tr>
</table>

Tabla 2. (Continúa)

<table>
<tr><td colspan="2">$a+b<x\leq L$</td></tr>
<tr><td> 
</td><td>$$M(x)=\frac{wb}{2L}\left(a+\frac{2b}{3}\right)(L-x) \quad (29)$$
$$EI\frac{d^2u_3}{dx^2}=\frac{wb}{2L}\left(a+\frac{2b}{3}\right)(L-x)$$
$$EI\frac{du_3}{dx}=\frac{wb}{2L}\left(a+\frac{2b}{3}\right)\left(Lx-\frac{x^2}{2}\right)+C_5 \quad (30)$$
$$EIu_3=\frac{wb}{4L}\left(a+\frac{2b}{3}\right)\left(Lx^2-\frac{x^3}{3}\right)+C_5x+C_6$$
$$u_3=\frac{1}{EI}\left[\frac{wb}{12L}\left(L-a-\frac{2b}{3}\right)x^3-\frac{wb}{2}\left(\frac{x}{6}-\frac{a}{2}-\frac{b}{3}\right)x^2-\frac{wbL}{6}\left(a+\frac{2b}{3}\right)x-\frac{wb^3}{24L}\left(a+\frac{4b}{5}\right)x-\frac{wab}{12L}(a+b)^2x+\frac{w}{24}b^3\left(a+\frac{4b}{5}\right)+\frac{wab}{12}(a+b)^2\right] \quad (31)$$</td></tr>
</table>

Cabe la pena notar que los procedimientos anteriormente descritos, se encuentran dentro del cálculo computacional desarrollado y que se pueden hacer diferentes configuraciones de carga dentro de este, solo que cuando se quiera aplicar más de una carga a la viga, no se devolverá una solución analítica, solamente numérica, ello debido a lo tedioso que se vuelve el cálculo de la solución teórica al tener diferentes configuraciones de cargas, ya que en el proceso de solución manual para obtener las ecuaciones de desplazamiento se requiere de un mayor número de condiciones de fronteras para hallar las constantes de integración haciendo que este proceso de solución se vuelva ineficiente por la gran inversión de tiempo que se debe hacer en los cálculos manuales.

1.2 IMPLEMENTACIÓN DEL MÉTODO TEÓRICO

La implementación del método teórico se hace al principio del código para cada método numérico.

A continuación se hace la declaración de variables para el cálculo teórico. Es notar que las variables definidas como int y double comienzan con la letra i y d respectivamente y los vectores y matrices comienzan con las letras V y M respectivamente:

Variables privadas

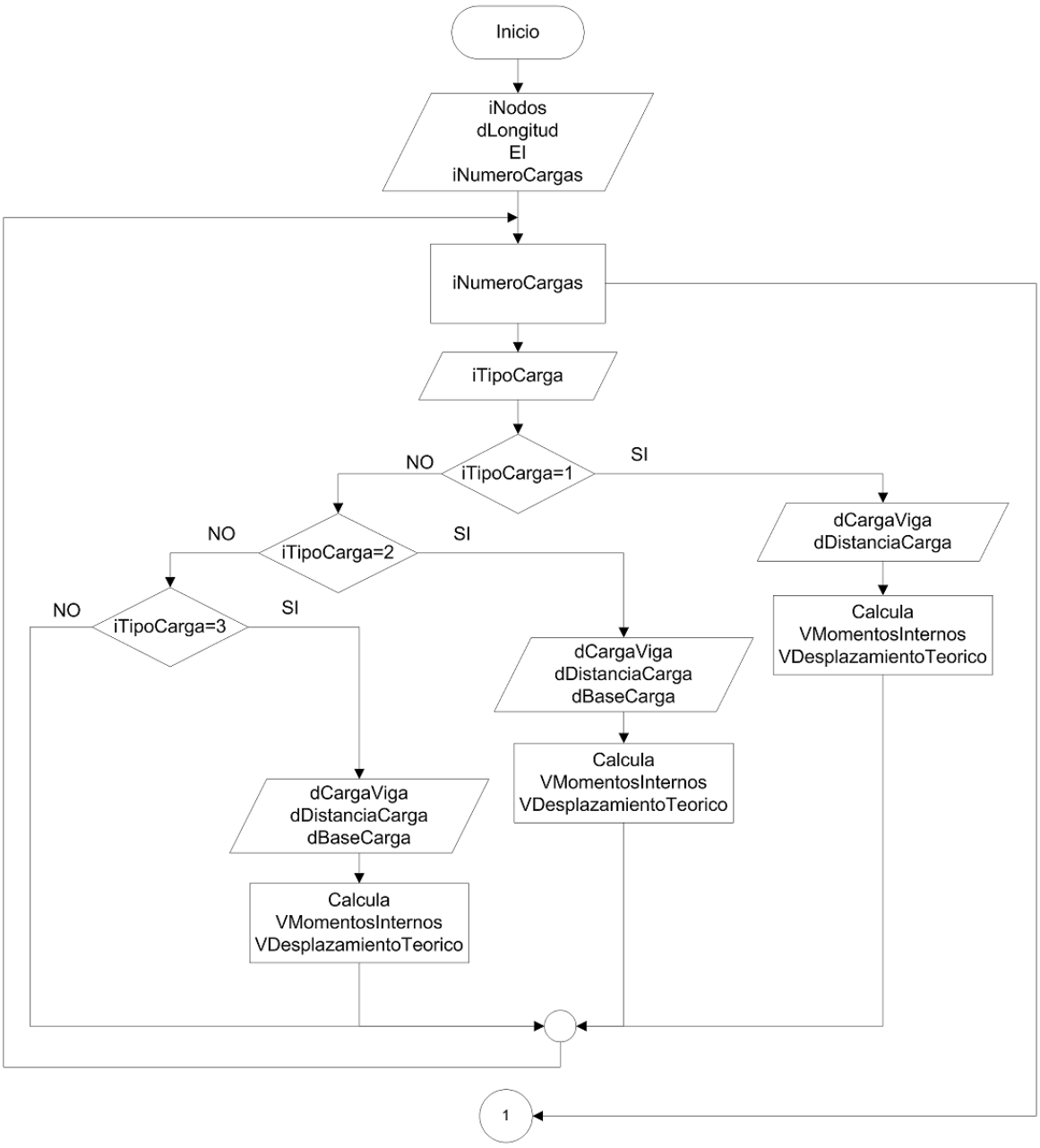
dNodosAntesCarga.	Numero de nodos antes de la carga.
dReaccionIzquierda.	Valor de la reacción izquierda.
dDistanciaDodoActual.	Distancia al nodo actual, a partir de apoyo izquierdo.
dLongitudElemento.	Longitud de cada elemento.
dDistanciaDespuesCarga.	Distancia después de la carga hasta el final de la viga.
VMomentosInternos.	Vector para calcular el momento interno en cada nodo.
VCargasPuntuales.	Vector calcula el valor de las cargas puntuales sobre la viga.
VDesplazamientoTeorico.	Almacena el desplazamiento, calculado analíticamente.

Variables públicas

iNumeroCargas.	Numero de cargas sobre la viga.
iNodos.	Numero de nodos en la malla.
iTipoCarga.	Tipo de carga sobre la viga (Puntual o Distribuida).
dLongitud.	Longitud de la viga.
dCargaViga.	Valor de la carga sobre la viga (p ò w).
dDistanciaCarga.	Distancia del apoyo izquierdo al inicio de la carga(a).
dBaseCarga.	Base de la carga distribuida sobre la viga (b).
EI.	Coeficiente de rigidez de la viga.

El desarrollo para el cálculo de los desplazamientos teóricos se muestra en el siguiente algoritmo:

Figura 9. Algoritmo para el cálculo de los desplazamientos por el análisis teórico.



La inicialización de las variables se hace en una clase que se genero para cada método numérico y su respectivo cálculo se hace en un código ejecutable.

Inicialmente se asigna el tamaño de los vectores y las matrices y se llenan de ceros, esto se hace por medio de las funciones chasize y llenar de la herramienta computacional MecLib.

```
VMomentosInternos.chaSize(iNodos);
VMomentosInternos.llenar(0.0);

VCargasPuntuales.chaSize(iNumeroCargas);
VCargasPuntuales.llenar(0.0);

VDesplazamientoTeorico.chaSize(iNodos);
VDesplazamientoTeorico.llenar(0.0);
```

Para el cálculo de los momentos generados en cada nodo de acuerdo a los tipos de cargas sobre la viga según las figuras (1), (5) y (7), se deben asignar valores para la carga aplicada sobre la viga, distancia a la que se encuentra la carga y base de la carga si es distribuida, el análisis de estos cálculos fue expuesto anteriormente y su orden de desarrollo va de acuerdo a la figura (9) donde se muestra el algoritmo para el cálculo del desplazamiento teórico, en el cual al terminar este proceso, comienza el desarrollo de cada método numérico.

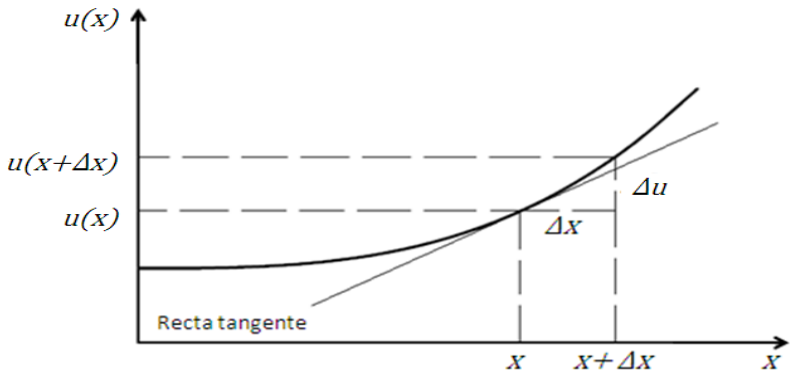
2. SOLUCIÓN EULER-BERNOULLI 1D POR EL MÉTODO DE DIFERENCIAS FINITAS

Con el ánimo de encontrar una solución a un modelo matemático que describe un fenómeno físico de ingeniería se hace necesario utilizar métodos de aproximación que puedan ser programados en una computadora, para este fin se requiere que las ecuaciones diferenciales ordinarias o parciales sean transformadas en ecuaciones algebraicas por medio del uso de métodos numéricos. Uno de estos métodos es el de diferencias finitas, el cual funciona mediante un proceso de discretización que se basa en la transformación de un dominio continuo representado por un conjunto de números infinitos que describen las funciones incógnitas en un conjunto finito de parámetros.

2.1 DESCRIPCIÓN DEL MÉTODO

Una forma de describir el funcionamiento del método de diferencias es por medio de la definición de la derivada ya que esta es el componente principal de una ecuación diferencial.

Figura 10. Derivada de una función.



Considerando una función $u(x)$ en la cual su primera derivada con respecto a un punto equivale a la pendiente de una recta tangente a la curva que describe la función en ese punto, la cual se define como:

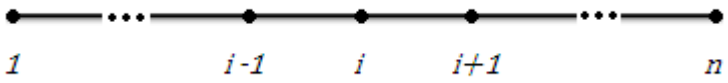
$$\frac{du(x)}{dx} = \lim_{\Delta x \to 0} \frac{\Delta u}{\Delta x} = \lim_{\Delta x \to 0} \frac{u(x + \Delta x) - u(x)}{\Delta x} \tag{32}$$

Donde

$$\frac{du(x)}{dx} \cong \frac{u(x + \Delta x) - u(x)}{\Delta x} \tag{33}$$

Como se pude ver en la Ecuación (33) esta expresión es una aproximación de la derivada en términos de una ecuación de diferencias la cual es la base del método de diferencias finitas.

Figura 11. Distribución de los nodos.



En el desarrollo del método de diferencias finitas se utilizara un arreglo de tres nodos, para esto evaluamos la primera derivada en los nodos externos de la siguiente forma:

$$\left.\frac{du}{dx}\right|_{x-1} \cong \frac{u_x - u_{x-1}}{\Delta x} \quad y \quad \left.\frac{du}{dx}\right|_{x+1} \cong \frac{u_{x+1} - u_x}{\Delta x} \tag{34}$$

Teniendo en cuenta que la segunda derivada es simplemente la derivada de la primera derivada se tiene para el nodo interno que:

$$\left.\frac{d^2u}{dx^2}\right|_x \cong \frac{\left.\frac{du}{dx}\right|_{x-\frac{1}{2}} - \left.\frac{du}{dx}\right|_{x+\frac{1}{2}}}{\Delta x} = \frac{\frac{u_{x+1}-u_x}{\Delta x} - \frac{u_x-u_{x-1}}{\Delta x}}{\Delta x} = \frac{u_{x-1}-2u_x+u_{x+1}}{\Delta x^2} \quad (35)$$

Sabiendo que la ecuación diferencial de una viga simplemente apoyada es:

$$\frac{d^2u}{dx^2} - \frac{M(x)}{EI} = 0 \quad (36)$$

Y reemplazando (35) en (36) se tiene:

$$\frac{u_{x-1}-2u_x+u_{x+1}}{\Delta x^2} = \frac{M(x)}{EI} \quad (37)$$

Donde u representa el desplazamiento de la viga, Δx la distancia entre dos puntos consecutivos, $M(x)$ el momento en la viga evaluada en x y EI es la rigidez del material.

De acuerdo con lo anterior para el desarrollo por diferencias finitas la ecuación (37) adopta la siguiente forma:

$$u_{i-1} - 2u_i + u_{i+1} = \frac{M(x)}{EI}\Delta x^2 \quad (38)$$

En la solución de las ecuaciones discretas obtenidas mediante el método de diferencias finitas se hace necesario el uso de un desarrollo matricial como se muestra a continuación.

$$[A][u] = [f] \tag{39}$$

Donde A es la matriz de rigidez de $n \times n$, donde n es el numero de nodos en el dominio. Estos coeficientes se toman de la ecuación (38) de la siguiente forma:

$$A_{i,i-1} = 1, \;\; A_{i,i} = -2 \;\; y \;\; A_{i,i+1} = 1, \quad i = 1, \dots, n \tag{40}$$

El resto de los $A_{i,j}$ son cero. Los vectores u y f son de tamaño n, en nuestro caso u representa el desplazamiento de la viga simplemente apoyada sometida a flexión y

$$f = \frac{M(x)}{EI}\Delta x^2 \tag{41}$$

El desarrollo computacional para el método de diferencias finitas, en el cual se hace el llenado de la matriz $\boldsymbol{A}$ (MRigidez) y el vector $\boldsymbol{f}$ (VTerminosIndependientes) de acuerdo a la ecuación (39), donde sus valores son tomados de las ecuaciones (40) y (41) respectivamente y posteriormente calcular el vector $\boldsymbol{u}$ (VDesplazamiento), se muestra a continuación:

```
void DiferenciasFinitas :: ResuelveProblema ()
{
int i,j;
for(i=1;i<=iNodos;i++)
{
        dDistanciaNodoActual=(i-1)*dLongitudElemento;
```

```
        CONDICIONES DE FRONTERA NODO 1
         if (i==1)
        {
                MRigidez(1,1)=1.0;
                VTerminosIndependientes(1)=0.0;
        }
        NODOS INTERNOS
        else if (i>1 && i<iNodos)
        {
                MRigidez(i,i-1)=1.0;
                MRigidez(i,i)=-2.0;
                MRigidez(i,i+1)=1.0;
                VTerminosIndependientes(i)=(dLongitudElemento
                *dLongitudElemento*VMomentosInternos(i))/EI;
        }
        CONDICIONES DE FRONTERA ÚLTIMO NODO
        else if (i==iNodos)
        {
                MRigidez(i,i)=1.0;
                VTerminosIndependientes(i)=0.0;
        }
}
MRigidez.factLU();
MRigidez.forwBackLU(VTerminosIndependientes,VDesplazamiento);
}
```

Este desarrollo matemático se hace con el fin de comprender la implementación del método de diferencias finitas para el desarrollo de una viga simplemente apoyada.

2.2 IMPLEMENTACIÓN DEL MÉTODO DE DIFERENCIAS FINITAS 1D

En la implementación del método de diferencias finitas se utilizo una sofisticada herramienta para el desarrollo de software numérico llamada MecLib, cuya principal área de aplicación es la solución numérica de ecuaciones diferenciales parciales. El MecLib básicamente es una colección de librerías escritas en el lenguaje de programación C++, las clases que contiene esta herramienta pueden ser utilizadas en un gran rango de aplicaciones numéricas, matemáticas, físicas y de ingeniería. En el desarrollo de la aplicación de una viga simplemente apoyada, esta herramienta computacional fue de gran ayuda para la solución de la ecuación diferencial que gobierna dicho fenómeno.

La solución computacional para una viga simplemente apoyada está conformada por una clase y un ejecutable.

A continuación se hace la declaración de variables para el desarrollo por diferencias finitas. Cabe señalar que las variables definidas como int y double comienzan con la letra i y d respectivamente y los vectores y matrices comienzan con las letras V y M respectivamente:

Variables privadas

MRigidez.	Matriz de rigidez.
VTerminosIndependientes.	Vector de términos independientes o fuerzas externas.
VDesplazamiento.	Almacena el desplazamiento, calculado numéricamente.
VDesplazamientoTeorico.	Almacena el desplazamiento, calculado analíticamente.
relojcpu reloj.	Calcula el tiempo de cálculo.

2.2.1 Descripción Paso a Paso. El código para la solución numérica, de la viga simplemente apoyada se divide en tres etapas, inicialmente se llaman las librerías y la clase creada para las diferencias finitas, de la siguiente forma:

```
#include "StdAfx.h"
#include "DiferenciasFinitas.h"
#include <iniciarLIB.h>

using namespace std;
int main(int argc, const char * argv[])
{
        iniciarLIB(argc,argv);
        DiferenciasFinitas problema;
        problema.Leer ();
        problema.ResuelveProblema ();
        problema.Reportar ();
        system("PAUSE");
        return (0);
}
```

La explicación de cada etapa se muestra a continuación:

- **problema.Leer ();**

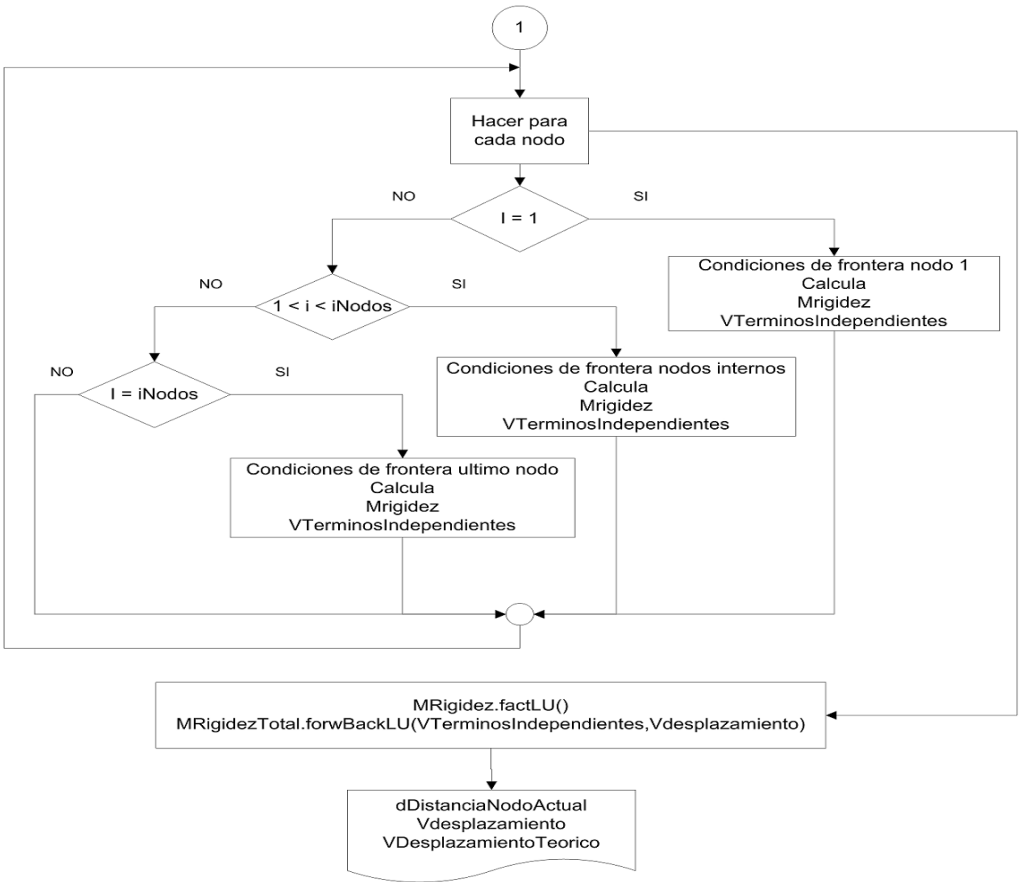
En esta etapa se leen las variables tales como geometría y cargas aplicadas sobre la viga y se calculan los momentos que generan las cargas sobre la viga para cada nodo, esto se hace en el archivo .h de la clase.

- **problema.ResuelveProblema ();**

Se hace el llenado de la matriz de rigidez y el vector de fuerza externas, de lo cual resulta el cálculo del vector u donde se almacena el desplazamiento de cada nodo.

En esta etapa se asignaran los valores para las condiciones de frontera para los nodos 1 y n según corresponda, después se llenan los valores de los nodos intermedios, finalmente se factoriza la matriz de rigidez y se calcula el sistema de ecuaciones lineales por medio de las funciones factLU y forwBackLU respectivamente, estas funciones hacen parte del paquete de desarrollo MecLib. A continuación en la figura (12) se muestra el algoritmo que se utiliza en el desarrollo de esta etapa.

Figura 12. Algoritmo para la implementación del método de diferencias finitas.



- **problema.Reportar ();**

Imprime los resultados en un archivo de texto “Reusltados.res” para comparar la respuesta obtenida por el método de diferencias finitas con el desplazamiento teórico de la viga simplemente apoyada, en cada nodo. A continuación se muestra el desarrollo de esta etapa.

```
void DiferenciasFinitas :: Reportar ()
{
        int i;
        Os ofile("Resultados.res",NEWFILE);
        ofile <<Reloj.reportar2("Tiempo de calculo");
```

```
        ofile <<"Posicion" <<"\t" "Desplazamiento Analitico" <<"\t"
        "Desplazamiento Numerico"<<endl;
        for(i=1;i<=iNodos;i++)
        {
                dDistanciaNodoActual=(i-1)*dLongitudElemento;
                ofile << oform("%4.3f   %17.5e   %25.5e\n",
                dDistanciaNodoActual, VDesplazamientoTeorico(i),
                VDesplazamiento(i));
        }
        ofile->cerrar();
}
```

Los resultados obtenidos son visualizados en un archivo de texto llamado "Resultados.res", de la siguiente forma:

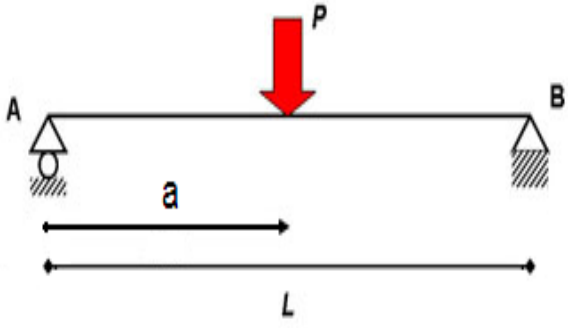
Tiempo de cálculo: 0.0000

Posición	Desplazamiento Analítico	Desplazamiento Numérico
0.000	0.00000e+000	0.00000e+000
0.100	-2.56944e-004	-2.60417e-004
0.200	-4.93056e-004	-5.00000e-004
0.300	-6.87500e-004	-6.97917e-004
0.400	-8.19444e-004	-8.33333e-004
0.500	-8.68056e-004	-8.85417e-004
0.600	-8.19444e-004	-8.33333e-004
0.700	-6.87500e-004	-6.97917e-004
0.800	-4.93056e-004	-5.00000e-004
0.900	-2.56944e-004	-2.60417e-004
1.0	0.00000e+000	0.00000e+000

2.3 ANÁLISIS DE RESULTADOS

Se realizaron diferentes configuraciones de carga en la implementación del método para una viga simplemente apoyada con el fin de estimar el porcentaje de error presentado. En todos los casos se utilizó la misma geometría, número de nodos y propiedades de material.

Tabla 3. Solución de una viga simplemente apoyada sometida a carga puntual por el Método de Diferencias Finitas.

Carga puntual		
	Numero de Nodos	100
	L	1 m.
	EI	2400000 N-m²
	Numero de cargas	1
	Tipo de carga	1
	P	10000 N.
	a	0,5 m.
	%Error	1,27%
	Tiempo de calculo	0.0160 s.

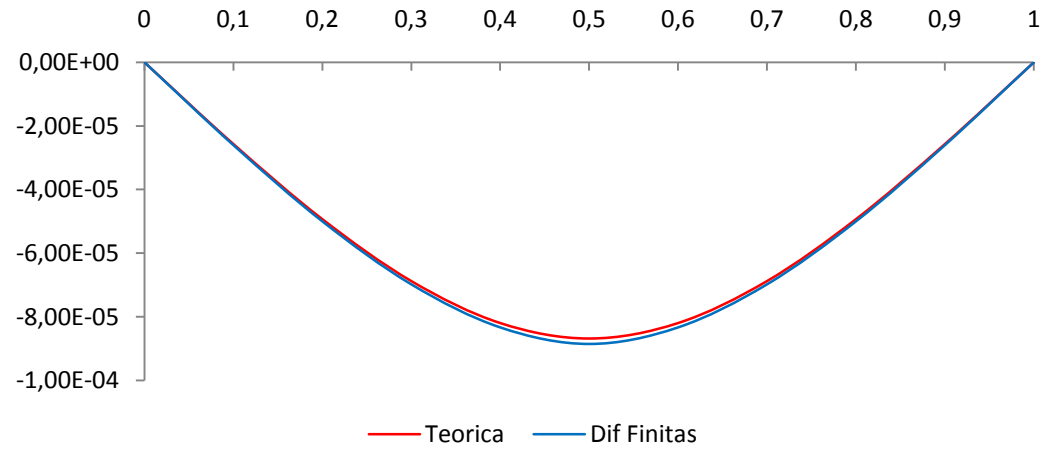
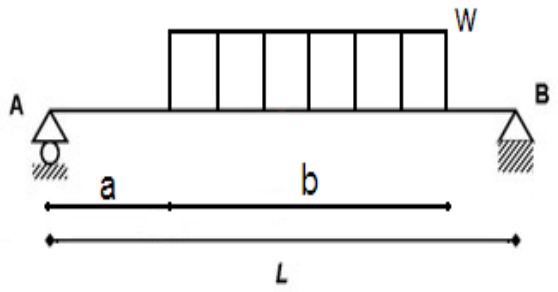


Tabla 4. Solución de una viga simplemente apoyada sometida a carga distribuida rectangular por el Método de Diferencias Finitas.

Carga Distribuida rectangular		
	Numero de Nodos	11
	L	1 m
	EI	2400000 N-m²
	Numero de cargas	1
	Tipo de carga	2
	W	5000 N/m
	a	0,3 m
	b	0,6 m
	%Error	0,67%
	Tiempo de calculo	0.0160 s.

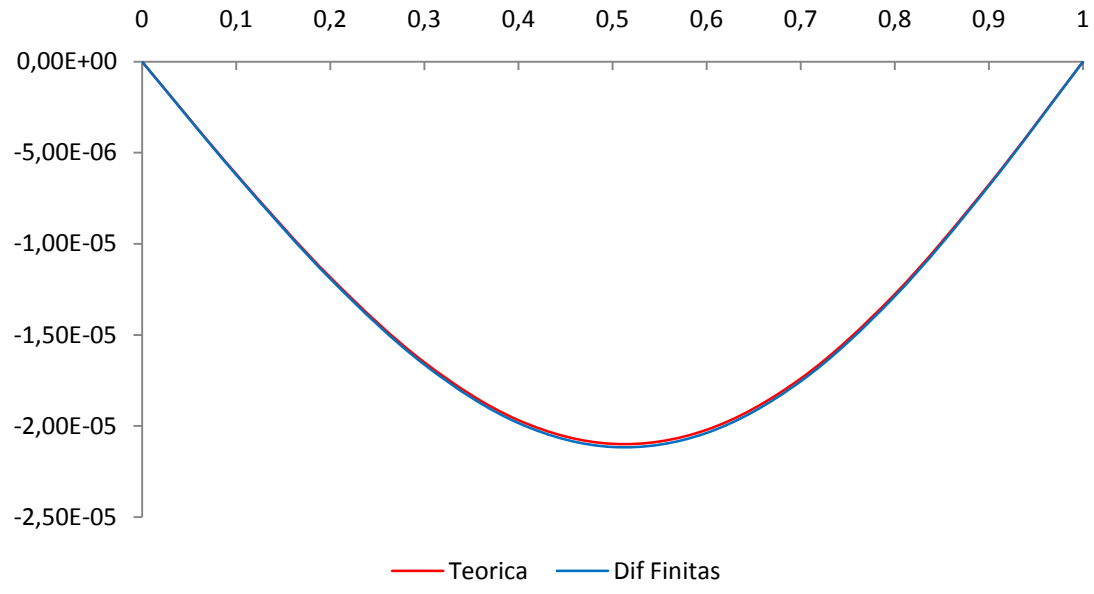
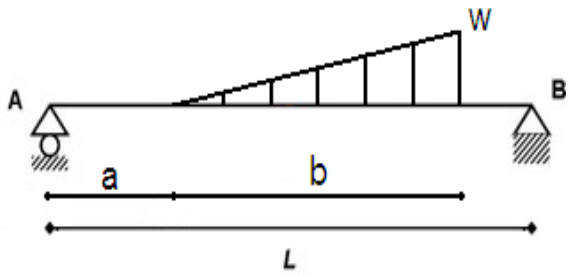
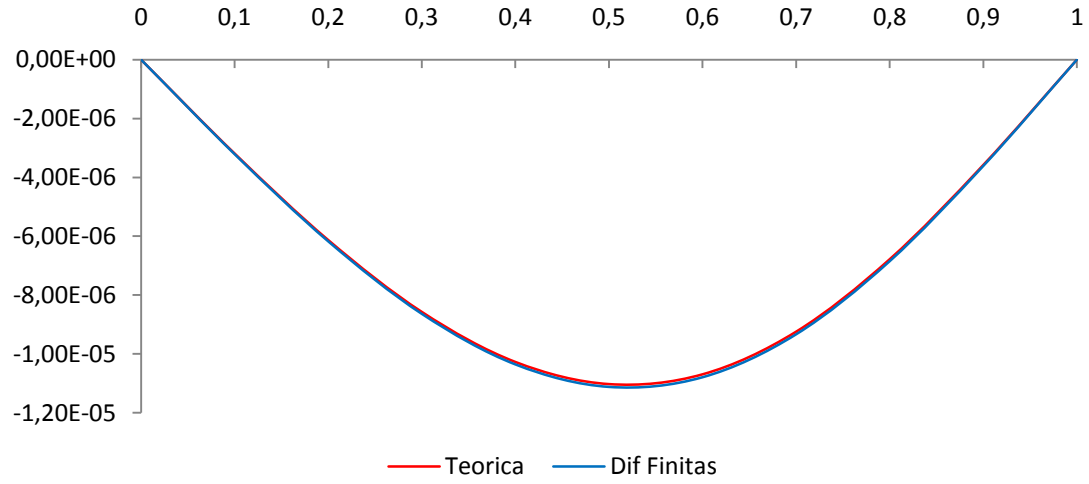


Tabla 5. Solución de una viga simplemente apoyada sometida a carga distribuida triangular por el Método de Diferencias Finitas.

Carga Distribuida triangular		
	Numero de Nodos	11
	L	1 m
	EI	2400000 N-m²
	Numero de cargas	1
	Tipo de carga	3
	W	5000 N/m
	a	0,2 m
	b	0,6 m
	%Error	0,66%
	Tiempo de calculo	0.0160 s.



2.4 CONCLUSIONES

1. El método de diferencias finitas tiene un desarrollo bastante simplificado, debido a que su formulación en el problema de la viga simplemente apoyada es sencilla de entender.

2. El porcentaje de error que se presenta en este método es de alrededor del 1% el cual es aceptable, además es de notar que la diferencia entre el análisis analítico y el numérico se acentúa alrededor del desplazamiento máximo de la viga, es allí donde se aprecia notablemente la inexactitud del método, punto en el cual la malla sufre la mayor deformación.

3. Al trabajar con una cantidad de nodos superior a la que se utiliza para el análisis de resultados mostrado anteriormente se puede observar que las curvas se tornan más suaves pero el porcentaje de error no varía.

3. SOLUCIÓN EULER-BERNOULLI 1D POR EL MÉTODO DE ELEMENTOS FINITOS

El método de elementos finitos es una herramienta de desarrollo numérico ampliamente utilizada en ingeniería mecánica, en lo concerniente al área de mecánica computacional para la solución de ecuaciones diferenciales de gran complejidad que gobiernen algún problema físico, este método se basa en la discretización de un dominio de naturaleza continua, por medio de un conjunto de elementos formados por nodos, que unidos entre si se les denomina malla.

3.1 DESCRIPCIÓN DEL MÉTODO

El desarrollo del método de elementos finitos según Ref (1) se basa en 5 pasos fundamentales, los cuales son:

1) Discretización del dominio: se basa en la localización y numeración de los puntos nodales especificando el valor de sus coordenadas.

2) Especificar la ecuación de aproximación: se debe asignar un valor para el orden de la ecuación de aproximación si es lineal o cuadrática y deben ser expresadas en términos de los valores nodales desconocidos.

3) Desarrollar los sistemas de ecuaciones: al utilizar el método de Galerkin para obtener la solución numérica de una ecuación diferencial, las funciones de peso en cada valor nodal desconocido es definido y se evalúa la integral residual del peso, este proceso genere una ecuación para cada valor nodal desconocido.

4) Solución del sistema de ecuaciones

5) Calculo de las cantidades de interés: estas cantidades están normalmente relacionadas con la derivada del parámetro a evaluar las cuales pueden ser: Componentes de los esfuerzos, flujo de calor y velocidades del fluido

En nuestro caso construiremos un modelo de elementos finitos para el problema de flexión de una viga simplemente apoyada basándonos en la formulación de Galerkin para obtener la solución numérica de la ecuación diferencial.

3.1.1 Formulación del Método de Galerkin. Para explicar el funcionamiento del método de Galerkin comenzamos por definir una ecuación diferencial que puede ser de la forma:

$$D\frac{d^2\emptyset}{dx^2} + Q = 0 \tag{42}$$

El sistema de ecuaciones lineales es generado por medio de la integral del peso residual.

$$-\int_0^H W(x)\left(D\frac{d^2\emptyset}{dx^2} + Q\right)dx = 0 \tag{43}$$

Para efectos de la formulación de Galerkin se requiere que la función de peso sea construida utilizando las funciones de forma $N_i \; y \; N_j$. Esta formulación establece que la función de peso en un nodo dado será igual a la función de forma asociada con dicho nodo.

MATRICES ELEMENTALES

Para la solución de las ecuaciones que se generan en el desarrollo del método de elementos finitos se hace indispensable un análisis matricial. Este análisis se hace por medio la construcción de matrices elementales, donde se define un vector columna {R} en la cual cada componente de este vector representa una ecuación residual.

$$\{R\} = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_{I-1} \\ R_I \end{bmatrix} \tag{44}$$

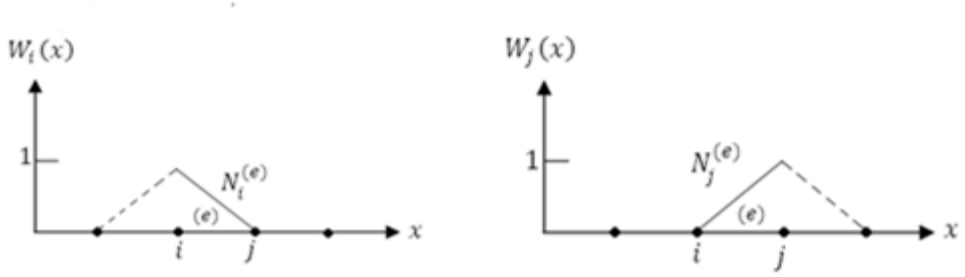
Donde R_I es la ecuación residual para el nodo I. Estas ecuaciones residuales también pueden ser expresadas como R_I^e donde $\boldsymbol{e}$ es el elemento que contribuye a la ecuación residual en el nodo I**.**

Para un elemento cualquiera que este conformado por los nodos $i\ y\ j$ la ecuación residual adopta la siguiente forma:

$$R_i^e = -\int_{X_i}^{X_j} N_i(x)\left(D\frac{d^2\emptyset}{dx^2} + Q \right) dx$$
$$R_j^e = -\int_{X_i}^{X_j} N_j(x)\left(D\frac{d^2\emptyset}{dx^2} + Q \right) dx \tag{45}$$

Donde estas son las funciones de forma para el elemento(e)

Figura 13. Funciones de peso para los elementos.



Al evaluar las integrales de la ecuación (45) queda la siguiente expresión:

$$R_i^e = D\left.\frac{d\emptyset}{dx}\right|_{x=X_i} + \frac{D}{L}\left(\Phi_i - \Phi_j\right) - \frac{QL}{2}$$
$$R_j^e = -D\left.\frac{d\emptyset}{dx}\right|_{x=X_j} + \frac{D}{L}\left(-\Phi_i + \Phi_j\right) - \frac{QL}{2} \quad (46)$$

Estas ecuaciones se pueden escribir de la siguiente manera:

$$\begin{Bmatrix} R_i^e \\ R_j^e \end{Bmatrix} = \begin{Bmatrix} I_i^e \\ I_j^e \end{Bmatrix} + \frac{D}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} \Phi_i \\ \Phi_j \end{Bmatrix} - \frac{QL}{2}\begin{Bmatrix} 1 \\ 1 \end{Bmatrix} \quad (47)$$

$$\left\{R^{(e)}\right\} = \left\{I^{(e)}\right\} + \left[k^{(e)}\right]\left\{\Phi^{(e)}\right\} - \left\{f^{(e)}\right\} \quad (48)$$

Donde $\left[k^{(e)}\right]$ es la matriz elemental de rigidez,

$$\left[k^{(e)}\right] = \frac{D}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad (49)$$

$\{f^{(e)}\}$ es el vector elemental de fuerza,

$$\{f^{(e)}\} = \frac{QL}{2}\begin{Bmatrix}1\\1\end{Bmatrix} \tag{50}$$

$\{\Phi^{(e)}\}$ es el vector columna de los valores nodales y el factor $\{I^{(e)}\}$ solo se tiene en cuenta en casos especiales, en nuestra aplicación de una viga simplemente apoyada, sometida a flexión no será utilizado.

El vector {R} representa un sistema de ecuaciones y se puede representar de la siguiente manera:

$$\{R\} = [K]\{\Phi\} - \{F\} = \{0\} \tag{51}$$

El desarrollo computacional para el método de elementos finitos, en el cual se hace el llenado de la matriz $\boldsymbol{K}$ (MRigidez) y el vector $\boldsymbol{F}$ (VTerminosIndependientes) según la ecuación (51), donde sus valores son tomados de acuerdo a la conectividad entre los elementos de la viga, los cuales tiene su respectiva matriz $\boldsymbol{k}^{(e)}$ (MRigidezLocal) y vector $\boldsymbol{f}^{(e)}$ (VFuerzasExternasLocal), conforme las ecuaciones (49) y (50) respectivamente para posteriormente calcular el vector $\boldsymbol{\Phi}$ (VDesplazamiento), se muestra a continuación:

```
void ElementosFinitos :: ResuelveProblema ()
{
int i,j;
dCteMRigidezLocal=EI/dLongitudElemento;
MRigidezLocal(1,2)=-1.0;
MRigidezLocal(2,1)=-1.0;
```

```
MRigidezLocal.mult(dCteMRigidezLocal);
for(j=1;j<=iNodos-1;j++)
{
        VFuerzasExternasLocal.llenar(1.0);
        if(j==1)
        {
                MRigidezGlobal(1,1)=MRigidezLocal(1,1);
                MRigidezGlobal(1,2)=MRigidezLocal(1,2);
                MRigidezGlobal(2,1)=MRigidezLocal(2,1);
                MRigidezGlobal(2,2)=MRigidezLocal(2,2);
                VFuerzasExternasGlobal(1)=VFuerzasExternasLocal(1);
                VFuerzasExternasGlobal(2)=VFuerzasExternasLocal(2);
        }
        else
        {
                MRigidezGlobal(j,j)=MRigidezGlobal(j,j)+MRigidezLocal(1,1);
                MRigidezGlobal(j,j+1)=MRigidezLocal(1,2);
                MRigidezGlobal(j+1,j)=MRigidezLocal(2,1);
                MRigidezGlobal(j+1,j+1)=MRigidezLocal(2,2);
                VFuerzasExternasGlobal(j)=VFuerzasExternasGlobal(j)+
                VFuerzasExternasLocal(1);
                VFuerzasExternasGlobal(j+1)=VFuerzasExternasLocal(2);
        }
}
for(i=1;i<=iNodos;i++)
{
        dCteFuerzasExternasLocal=VMomentosInternos(i)*
        LongitudElemento/2.0;
        VFuerzasExternasGlobal(i)=VFuerzasExternasGlobal(i)*
        dCteFuerzasExternasLocal;
```

```
}
for(i=2;i<=iNodos-1;i++)
{
        for(j=2;j<=iNodos-1;j++)
        {
                MRigidez(i-1,j-1)=MRigidezGlobal(i,j);
        }
}
for(i=2;i<=iNodos-1;i++)
{
        VFuerzasExternas(i-1)=VFuerzasExternasGlobal(i);
}
MRigidez.factLU();
MRigidez.forwBackLU(VFuerzasExternas,VDesplazamiento);
}
```

Valga la pena aclara que en el desarrollo del método de elementos finitos, solo se tiene un grado de libertad, ya que la solución del método sin malla solo tiene en cuenta un solo grado de libertad.

3.2 IMPLEMENTACIÓN DEL MÉTODO DE ELEMENTOS FINITOS 1D

En la implementación del método de elementos finitos al igual que en diferencias finitas se utilizo la herramienta MecLib. La solución computacional para una viga simplemente apoyada está conformada por una clase y un ejecutable, a continuación se presentara el código realizado:

Inicialmente se hace la declaración de variables para el desarrollo por el método de elementos finitos. Cabe señalar que las variables definidas como int y double

comienzan con la letra i y d respectivamente y los vectores y matrices comienzan con las letras V y M respectivamente:

Variables privadas

dCteMRigidezLocal.	Constante de la matriz de rigidez local
dCteFuerzasExternasLocal.	Constante del vector fuerzas externas locales
MRigidezLocal.	Matriz de rigidez correspondiente a cada elemento
MRigidezGlobal.	Matriz de rigidez que contiene las matrices de rigidez locales
MRigidez.	Matriz de rigidez en la que se aplican las condiciones de frontera
VFuerzasExternasLocal.	Vector de fuerzas externas correspondientes a cada elemento
VFuerzasExternasGlobal.	Vector de fuerzas externas contiene las fuerzas externas locales
VFuerzasExternas.	Vector de fuerzas externas con las condiciones de frontera
VDesplazamiento.	Vector calcula el desplazamiento en cada nodo de forma numérica
VDesplazamientoTeorico.	Vector calcula el desplazamiento en cada nodo de forma teórica
RelojCPU Reloj;	Calcula el tiempo de cálculo.

3.2.1 Descripción Paso a Paso. El código para la solución numérica, de la viga simplemente apoyada se divide en tres etapas, inicialmente se llaman las librerías y la clase creada para las diferencias finitas, de la siguiente forma:

```
#include "StdAfx.h"
#include "ElementosFinitos.h"
#include <iniciarLIB.h>

using namespace std;
int main(int argc, const char * argv[])
{
        iniciarLIB(argc,argv);
        ElementosFinitos problema;
        problema.Leer ();
        problema.ResuelveProblema ();
        problema.Reportar ();
        system("PAUSE");
        return (0);
}
```

La explicación de cada etapa se muestra a continuación:

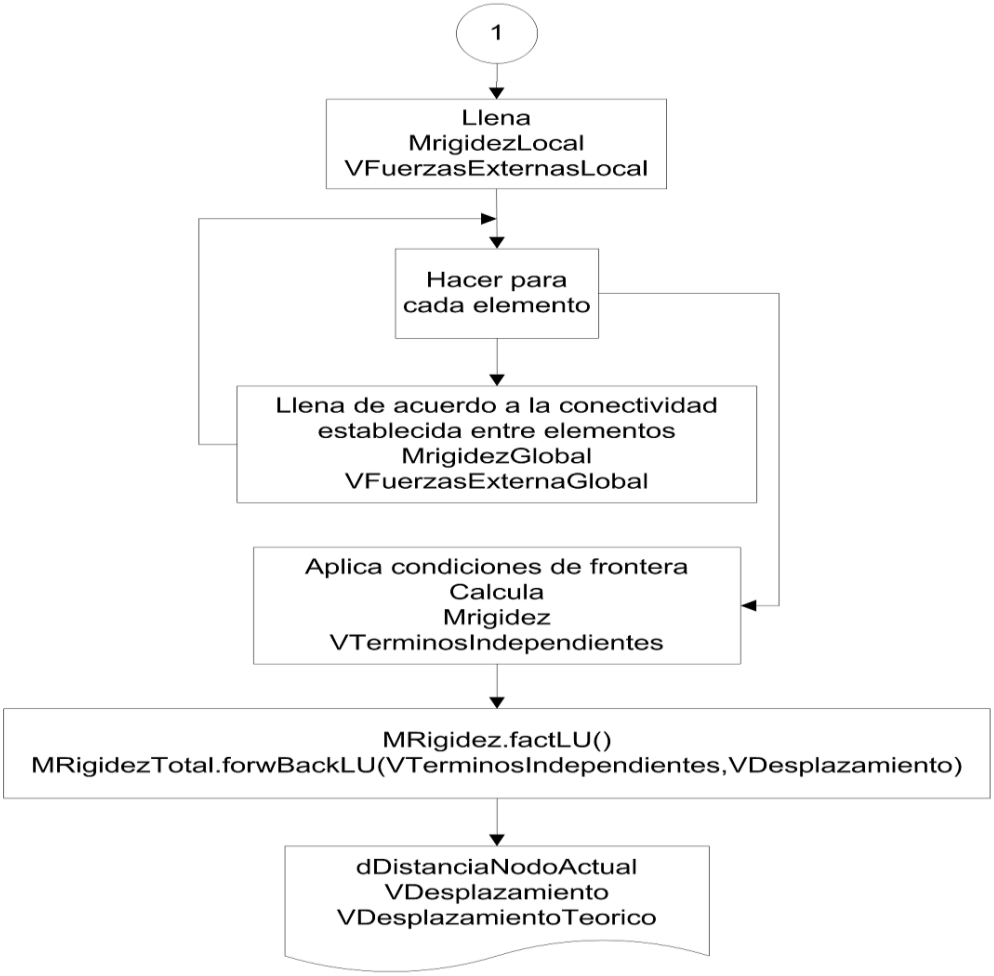
- **problema.Leer ();**

En esta etapa se leen las variables tales como geometría y cargas aplicadas sobre la viga y se calculan los momentos que generan las cargas sobre la viga para cada nodo.

- **problema.ResuelveProblema ();**

Se hace la conectividad entre la matriz de rigidez y el vector de fuerza externas de cada elemento y se les aplican las condiciones de frontera. Finalmente se factoriza la matriz de rigidez por medio de la función factLU y después se resuelva el sistema de ecuaciones lineales con la función forwBackLU que hacen parte del paquete de desarrollo MecLib. A continuación en la figura (14) se muestra el algoritmo que se utiliza en el desarrollo de esta etapa.

Figura 14. Algoritmo para la implementación del método de elementos finitos.



- **problema.Reportar ();**

Imprime los resultados en un archivo de texto “Reuslтados.res” para comparar la respuesta obtenida por el método de elementos finitos con el desplazamiento teórico de la viga simplemente apoyada, en cada nodo. A continuación se muestra el desarrollo de esta etapa.

```
void ElementosFinitos :: Reportar ()
{
        int i;
        Os ofile("Resultados.res",NEWFILE);
        ofile <<Reloj.reportar2("Tiempo de cálculo");
        ofile <<"Posición" <<"\t" "Desplazamiento Analítico" <<"\t"
        "Desplazamiento Numérico"<<endl;
        for(i=1;i<=iNodos;i++)
        {
                dDistanciaNodoActual=(i-1)*dLongitudElemento;
                if (i==1)
                {
                        ofile << oform("%4.3f   %10.3f   %25.3f\n", 0.0, 0.0, 0.0);
                }
                 else if (i>1 && i<iNodos)
                {
                        ofile << oform("%4.3f   %17.5e   %25.5e\n",
                        dDistanciaNodoActual, VDesplazamiento(i-1),
                        VDesplazamientoTeorico(i));
                }
         else if (i==iNodos)
         {
```

```
                    ofile << oform("%4.3f   %10.3f   %25.3f\n", dLongitud,
                    0.0, 0.0);
                }
            }
            ofile->cerrar();
        }
```

Los resultados obtenidos son visualizados en un archivo de texto llamado "Resultados.res", de la siguiente forma:

Tiempo de cálculo: 0.0000

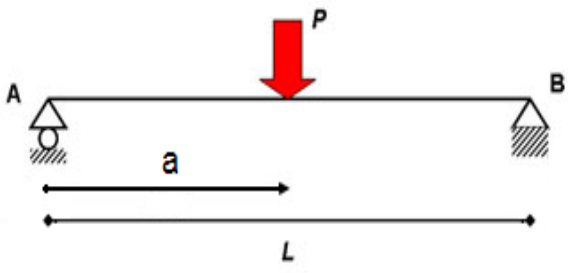
Posición	Desplazamiento Numérico	Desplazamiento Analítico
0.000	0.000	0.000
0.100	-6.20833e-006	-6.16667e-006
0.200	-1.19167e-005	-1.18333e-005
0.300	-1.66250e-005	-1.65000e-005
0.400	-1.98333e-005	-1.96753e-005
0.500	-2.11458e-005	-2.09722e-005
0.600	-2.03750e-005	-2.02031e-005
0.700	-1.75417e-005	-1.73889e-005
0.800	-1.28750e-005	-1.27587e-005
0.900	-6.81250e-006	-6.75000e-006
1.0	0.000	0.000

3.3 ANÁLISIS DE RESULTADOS

En el análisis de resultados para una viga simplemente apoyada se tomaron las mismas configuraciones de carga, geometría y propiedades del material, utilizadas

en el método de diferencias finitas con el fin de calcular el porcentaje de error que presenta el método con respecto a la solución analítica, también se mide el tiempo de cálculo con el fin de medir el coste computacional.

Tabla 6. Solución de una viga simplemente apoyada sometida a carga puntual por el Método de Elementos Finitos.

Carga puntual		
	Numero de Nodos	100
	L	1 m.
	EI	2400000 N-m²
	Numero de cargas	1
	Tipo de carga	1
	P	10000 N.
	a	0,5 m.
	%Error	1,27%
	Tiempo de calculo	0,0160 s.

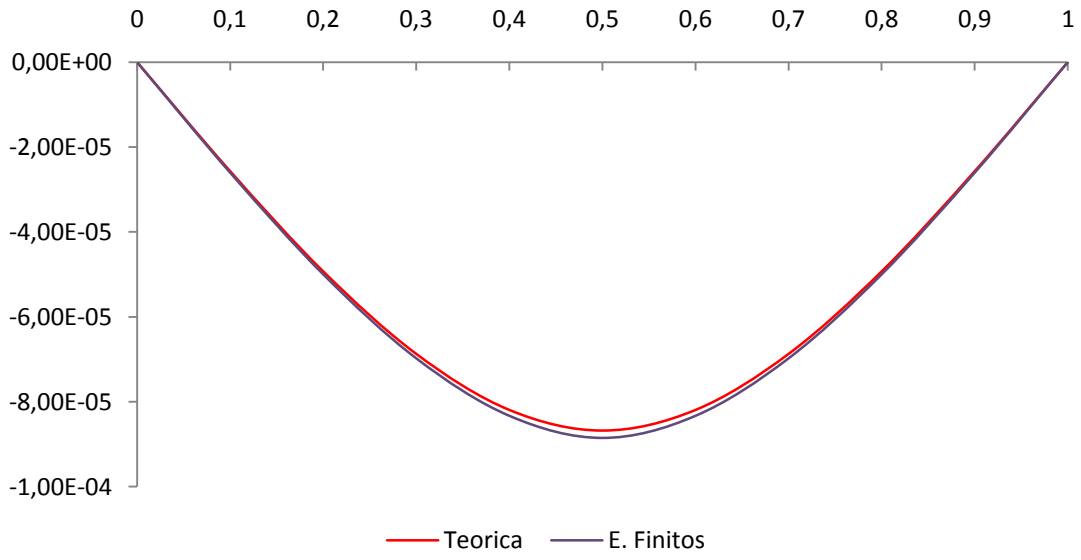
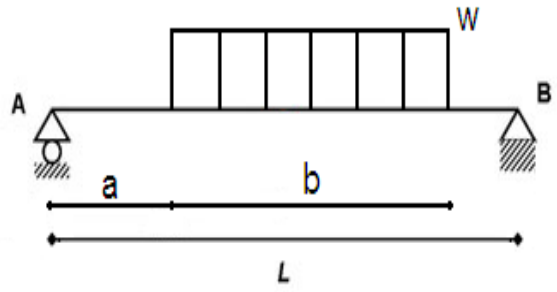


Tabla 7. Solución de una viga simplemente apoyada sometida a carga distribuida rectangular por el Método de Elementos Finitos.

Carga Distribuida rectangular



Numero de Nodos	11
L	1 m
EI	2400000 N-m²
Numero de cargas	1
Tipo de carga	2
W	5000 N/m
a	0,3 m
b	0,6 m
%Error	0,17%
Tiempo de calculo	0.4530 s.

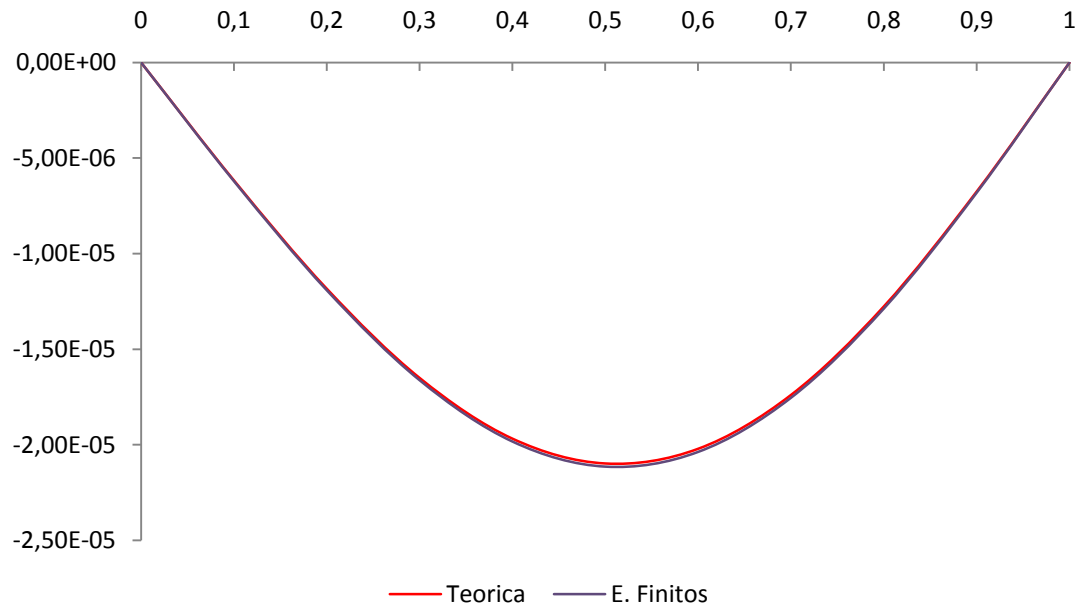
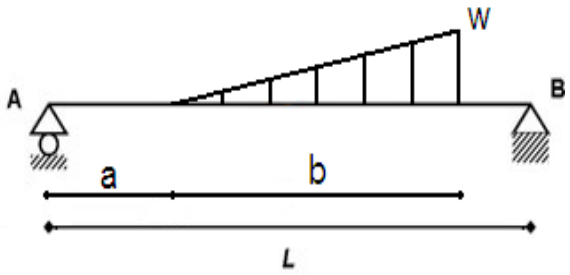
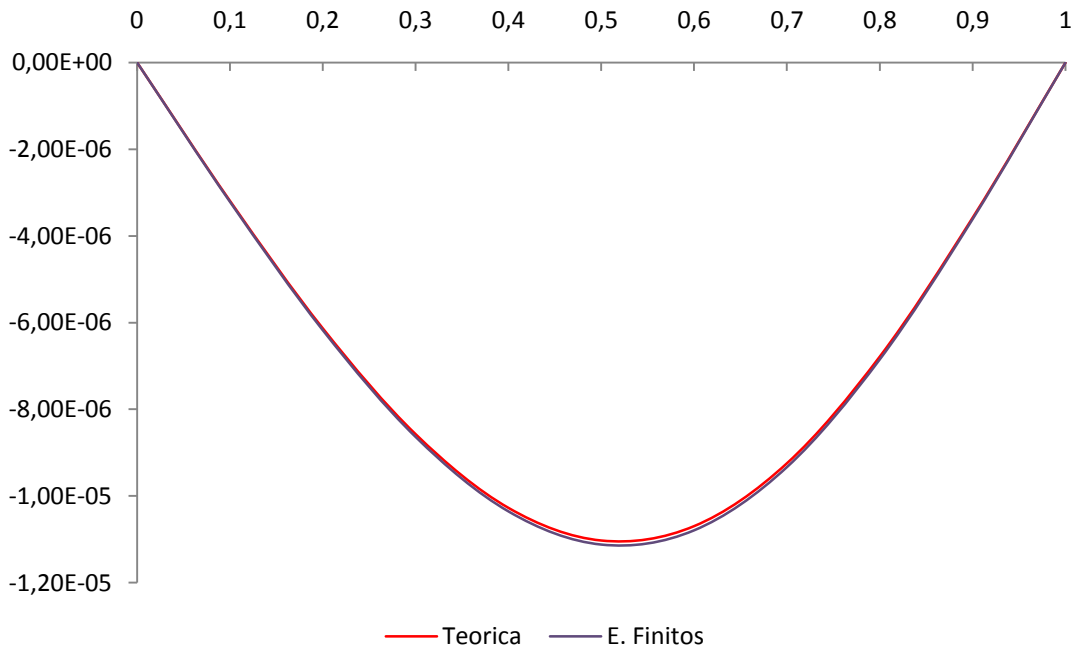


Tabla 8. Solución de una viga simplemente apoyada sometida a carga distribuida triangular por el Método de Elementos Finitos.

Carga Distribuida triangular 1



Numero de Nodos	11
L	1 m
EI	2400000 N-m²
Numero de cargas	1
Tipo de carga	3
W	5000 N/m
a	0,2 m
b	0,6 m
%Error	0,17%
Tiempo de calculo	0.4370s.



3.4 CONCLUSIONES

1. Las funciones de aproximación que se usaron en el desarrollo del método de elementos finitos, fueron lineales, esto hizo que la solución de la viga simplemente apoyada, arrojara resultados idénticos al método de diferencias finitas, a pesar de que tienen formulaciones diferentes.

2. El porcentaje de error que se presenta en este método es de alrededor del 1% el cual es aceptable.

3. Se pudo observar que en la medida que la malla se desplazaba más, el porcentaje de error aumentaba, esto se vio claramente, cuando el desplazamiento de la viga era máximo, lo cual se debe a que cuando los elemento se distorsionan de un modo considerable, la precisión del método se pierde en gran medida.

4. Es importante tener en cuenta que en la medida que el grado del polinomio de aproximación se aumente, el porcentaje de error disminuye, ya que el error se distribuye a lo lago de la viga de una forma más suave.

5. Al trabajar con una cantidad de nodos superior a la que se utiliza para el análisis de resultados mostrado anteriormente se puede observar que las curvas se tornan más suaves y el porcentaje de error disminuye.

4. SOLUCIÓN EULER-BERNOULLI 1D POR EL MÉTODO DE GALERKIN LIBRE DE MALLA

El método de Galerkin libre de malla (Ref. 2 y 3) es un método numérico que surge a raíz de las limitaciones que se presentan con los métodos numéricos tradicionales (FEM, FDM y FVM), para el modelado de problemas complejos en el área de la mecánica aplicada, en los cuales dichos métodos presentan una serie de dificultades a la hora de ser implementados. Los métodos libres de mallas se utilizan principalmente para la solución de ecuaciones diferenciales complejas que describen algunos fenómenos físicos de ingeniería y tienen como principal ventaja que no se requiere de una malla para tal fin, lo cual hace que el dominio no sea discretizado si no representado por un conjunto de nodos en los cuales se definen subdominios.

4.1 DESCRIPCIÓN DEL MÉTODO

4.1.1 Generación de Nodos. En el método libre de malla el dominio es representado por un conjunto de nodos, estos pueden ser distribuidos de forma arbitraria o uniforme, entre los cuales no hay necesidad de establecer ningún tipo de relación, por lo tanto no hay necesidad de construir mallas o elementos para interpolar las diferentes variables de generación de nodos. Esto nos permite liberarnos de la conectividad entre nodos que debe ser establecida previamente, en los métodos convencionales como las diferencias finitas y los elementos finitos.

Su implementación computacional se desarrolla así:

```
for(i=1;i<=iNodos;i++)
{
        VPosicionNodos(i)= VPosicionNodos(i)+((i*dLongitudElemento)-
        (dLongitudElemento));
}
```

4.1.2 Generación de los Puntos de Integración. Los puntos de integración son los lugares donde se resuelve numéricamente la integral para el cálculo de la función de forma. Este parámetro, representa físicamente el número de puntos de control en la integración, por lo tanto es claro que a mayor número de puntos de integración, mayor será el tiempo de cálculo. Para el desarrollo de una viga simplemente apoyada se tomara un punto de integración, entre dos nodos consecutivos.

Computacionalmente la posición de los puntos de integración es asignada de la siguiente forma:

```
for(i=1;i<=iNodos;i++)
{
        VPuntosIntegracion(i)=VPuntosIntegracion(i)+((((i*    dLongitudElemento)-
        dLongitudElemento))-(dLongitudElemento/2));
        if (i==1)
        {
                VPuntosIntegracion(i)=0.0;
        }
}
```

Note que el primer punto de integración se encuentra en la posición 0.0, este es el punto donde la condición del límite esencial es forzada.

4.1.3 Generación de las Funciones de Forma. En los métodos libres de malla la aproximación de los mínimos cuadrados móviles (MLS) es el método más usado para el cálculo de las funciones de forma, a pesar que su principal inconveniente radica en que estas no poseen la propiedad delta de Kronecker, lo que conlleva a un aumento del error en la solución de problemas de dominio.

A continuación se va a presentar el fundamento teórico del MLS.

4.1.3.1 Mínimos Cuadrados Móviles. Uno de los métodos sin mallas más utilizados en la actualidad es el método de Galerkin libre de malla, este método se considera un método libre de malla porque solo se requiere de un conjunto de nodos y de las condiciones de frontera para generar las ecuaciones discretas. La conectividad entre los nodos y las funciones de aproximación son construidas directamente por el método.

El método de Galerkin libre de malla está basado en los mínimos cuadrados móviles (MLS) de aproximación según (Ref. 4 y 5) y es el encargado de construir las funciones de forma, las cuales no satisfacen el criterio delta Kronecker $\left(\Phi_I(x_j) \neq \delta_k\right)$, lo que con lleva a que $u^h(x_I) \neq u_I$, hecho por el cual las funciones de forma no son interpoladores, como si sucede con en el método de elementos finitos.

Este método de aproximación está constituido por una serie de componentes tales como: una función de peso de soporte compacto asociada a cada nodo, una base que generalmente consiste en un polinomio y un conjunto de coeficientes que dependen de la posición del nodo. La función de peso que se utiliza en este

método toma un valor diferente de cero solo en un pequeño subdominio de influencia del nodo evaluado, haciendo que la superposición de estos dominios defina la conectividad nodal.

A continuación se dará detalle de la construcción de las funciones de forma.

Sea $u(x)$ la función a determinar en el dominio Ω y $u^h(x)$ la aproximación de dicha función en x, se tiene que:

$$u^h(x) = \sum_{j=0}^{m} p_j(x)a_j(x) = \boldsymbol{p}^T(x)\boldsymbol{a}(x) \tag{52}$$

Donde $\boldsymbol{p}(x)$ es un vector de funciones de bajo orden (m), ya que se obtiene la precisión necesaria sin aumentar la complejidad del problema y el tiempo de cálculo. Para desarrollos 1D se tiene que:

$$P^T(x) = \{1 \quad x \quad x^2 \quad \dots \; x^m\} \tag{53}$$

En el desarrollo de la aplicación de una viga simplemente apoyada se tomó

$$P^T(x) = \{1 \quad x\} \tag{54}$$

Su cálculo computacional se muestra a continuación, donde $P(x)$ está representado como MPolinomioAproximacion:

```
for(j=1;j<=2;j++)
{
        for(k=1;k<=iNodos;k++)
```

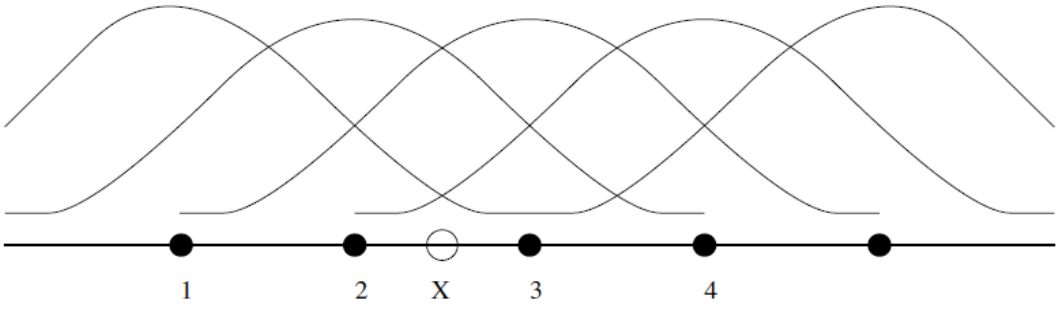
```
        {
            if (j==1)
            {
                MPolinomioAproximacion(j,k)=VUnos(k);
            }
            else
            {
                MPolinomioAproximacion(j,k)=VPosicionNodos(k);
            }
        }
}
```

Siguiendo con la ecuación (52), $\boldsymbol{a}(x)$ es el vector de coeficientes dispuesto de la siguiente forma:

$$a^T(x) = [a_0(x) \quad a_1(x) \quad a_2(x), \dots, a_m(x)] \tag{55}$$

Estos coeficientes, se determinan usando los valores de la función sobre el dominio de influencia.

Figura 15. Traslape de los dominios de influencias.



Al determinar $\boldsymbol{a}(x)$ es necesario construir una función con el fin de minimizar la ponderación así:

$$J = \sum_{I=1}^{n} w\,(x - x_I)[u_L^h(x_I, x) - u_I]^2 = \sum_{I=1}^{n} w\,(x - x_I)[\boldsymbol{p}^T(x_I)\boldsymbol{a}(x) - u_I]^2 \qquad (56)$$

Donde $w(x - x_I)$ es la función de peso, u_I es el parámetro nodal en el nodo I y n es el número de nodos.

En un punto arbitrario x estos coeficientes se escogen de forma tal que J sea mínimo, condición que se cumple cuando:

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{a}} = 0 \qquad (57)$$

Aplicando esta condición Ecuación (56), se tiene un sistema de ecuaciones lineales así:

$$\boldsymbol{A}(x)\boldsymbol{a}(x) = \boldsymbol{B}(x)\boldsymbol{u} \qquad (58)$$

De la cual se tiene que:

$$\boldsymbol{a}(x) = \boldsymbol{A}^{-1}(x)\boldsymbol{B}(x)\boldsymbol{u} \qquad (59)$$

Donde:

$$\begin{aligned} \boldsymbol{A} &= \sum_{I=1}^{n} w(x - x_I)\boldsymbol{p}(x_I)\boldsymbol{p}^T(x_I) \\ &= w(x - x_1)\begin{bmatrix} 1 & x_1 \\ x_1 & x_1^2 \end{bmatrix} + w(x - x_2)\begin{bmatrix} 1 & x_2 \\ x_2 & x_2^2 \end{bmatrix} + \cdots w(x - x_n)\begin{bmatrix} 1 & x_n \\ x_n & x_n^2 \end{bmatrix} \end{aligned} \qquad (60)$$

El Cálculo computacional de $\boldsymbol{A}$ (MFactorA) se obtiene de la siguiente forma, donde inicialmente se calcula $\boldsymbol{p}^T(x)$ (MPPT), dentro de este ciclo se calcula $(\boldsymbol{A})_{,x}$ (MDerivadaFactorA).

```
Matriz<double> MPPT(2,2);
MPPT.llenar(0.0);
Matriz<double> MDerivadaFactorA(2,2);
MDerivadaFactorA.llenar(0.0);
Vector<double> VPoliAproxNodoActual(2);
VPoliAproxNodoActual.llenar(0.0);
MFactorA.llenar(0.0);
for(j=1;j<=iNodos;j++)
{
        VPoliAproxNodoActual.llenar(0.0);
        for(k=1;k<=2;k++)
        {
                VPoliAproxNodoActual(k)=MPolinomioAproximacion(k,j);
        }
        MPPT.llenar(0.0);
        MPPT(1,1)=VPoliAproxNodoActual(1);
        MPPT(1,2)=VPoliAproxNodoActual(2);
        MPPT(2,1)=VPoliAproxNodoActual(2);
        MPPT(2,2)=VPoliAproxNodoActual(2)*
        VPoliAproxNodoActual(2);
        MFactorA.agregar(MFactorA,VFuncionPeso(j),MPPT);
        MDerivadaFactorA.agregar(MDerivadaFactorA,
        VDerivadaFuncionPeso(j),MPPT);
}
```

Siguiendo con la ecuación (56) se tiene que:

$$\boldsymbol{B}(x) = [w(x-x_1)\boldsymbol{p}(x_1), w(x-x_2)\boldsymbol{p}(x_2), \dots w(x-x_n)\boldsymbol{p}(x_n)]$$
$$= \left[w(x-x_1)\begin{bmatrix}1\\x_1\end{bmatrix}, w(x-x_2)\begin{bmatrix}1\\x_2\end{bmatrix}, \dots w(x-x_n)\begin{bmatrix}1\\x_n\end{bmatrix}\right] \quad (61)$$

El Cálculo computacional de $\boldsymbol{B}$ (MFactorB) se obtiene de la siguiente forma:

```
Matriz<double> MFuncionPeso(2,iNodos);
MFuncionPeso.llenar(0.0);
for(j=1;j<=2;j++)
{
        for(k=1;k<=iNodos;k++)
        {
                MFuncionPeso(j,k)=MFuncionPeso(j,k)+
                VFuncionPeso(k);
        }
}
for (j=1;j<=2;j++)
{
        for(k=1;k<=iNodos;k++)
        {
                MFactorB(j,k)=MPolinomioAproximacion(j,k)*
                MFuncionPeso(j,k);
        }
}
```

y

$$\boldsymbol{u}^T = [u_1, u_2, \dots u_n] \quad (62)$$

Finalmente remplazando la ecuación (56) en la (59) se tiene que:

$$\boldsymbol{u}^h(x) = \sum_{I=1}^{n} \Phi_I(x) u_I = \Phi(\mathrm{x})\mathbf{u} \tag{63}$$

En la cual la función de forma es definida por:

$$\Phi_I(x) = \sum_{j=0}^{m} p_j(x) \left(\boldsymbol{A}^{-1}(x)\boldsymbol{B}(x)\right)_{jI} = \boldsymbol{p}^T \boldsymbol{A}^{-1} \boldsymbol{B}_I \tag{64}$$

Donde j es el punto de integración evaluado.

El cálculo computacional de la función de forma $\Phi_I(x)$ (VFuncionForma) y de $\boldsymbol{A}^{-1}$ (MFactorAinv) se muestra a continuación, previamente se calculo $\boldsymbol{p}^T$, $\boldsymbol{A}$ y $\boldsymbol{B}$:

```
Matriz<double> MFactorAinv(2,2);
MFactorAinv.llenar(0.0);
MFactorA.inversa(MFactorAinv);
Vector<double> VPuntoIntegracionActual(2);
VPuntoIntegracionActual.llenar(0.0);
VPuntoIntegracionActual(1)=1.0;
VPuntoIntegracionActual(2)=dPuntoIntegracion;
Vector<double> VTerminoFuncionForma(2);
VTerminoFuncionForma.llenar(0.0);
MFactorAinv.prod(VPuntoIntegracionActual,VTerminoFuncionForma
VFuncionForma.llenar(0.0);
for(j=1;j<=iNodos;j++)
{
        for(k=1;k<=2;k++)
```

```
        {
            VFuncionForma(j)=VFuncionForma(j)+(VTerminoFuncionForma(k)*
            MFactorB(k,j));
        }
}
```

Es necesario también obtener la derivada de la función de forma, la cual se obtiene así:

$$\Phi_{I,x} = (\boldsymbol{p}^T \boldsymbol{A}^{-1} \boldsymbol{B}_I)_{,x} = \boldsymbol{p}_{,x}^T \boldsymbol{A}^{-1} \boldsymbol{B}_I + \boldsymbol{p}^T (\boldsymbol{A}^{-1})_{,x} \boldsymbol{B}_I + \boldsymbol{p}^T \boldsymbol{A}^{-1} \boldsymbol{B}_{I,x} \qquad (65)$$

Su cálculo computacional se desarrolla de la siguiente forma:

- Calculo de $\boldsymbol{p}_{,x}^T$ (MDerivadaFuncionPeso)

```
Matriz<double> MDerivadaFuncionPeso(2,iNodos);
MDerivadaFuncionPeso.llenar(0.0);
for(j=1;j<=2;j++)
{
        for(k=1;k<=iNodos;k++)
        {
            MDerivadaFuncionPeso(j,k)=MDerivadaFuncionPeso(j,k)+
            VDerivadaFuncionPeso(k);
        }
}
```

- Calculo de $\boldsymbol{B}_{I,x}$ (MDerivadaFactorB)

```
Matriz <double> MDerivadaFactorB(2,iNodos);
```

```
MDerivadaFactorB.llenar(0.0);
for (j=1;j<=2;j++)
{
        for(k=1;k<=iNodos;k++)
        {
                MDerivadaFactorB(j,k)=MPolinomioAproximacion(j,k)*
                MDerivadaFuncionPeso(j,k);
        }
}
```

- Calculo de $(\boldsymbol{A}^{-1})_{,x}$ (MFactorDerivadaAinv)

```
Matriz<double> MFactorDerivadaAinv(2,2);
MFactorDerivadaAinv.llenar(0.0);
MFactorDerivadaAinv.prod(MDerivadaFactorA,MFactorAinv);
Matriz <double> MDerivadaFactorAinv(2,2);
MDerivadaFactorAinv.llenar(0.0);
MDerivadaFactorAinv.prod(MFactorAinv,MFactorDerivadaAinv);
MDerivadaFactorAinv.mult(-1.0);
```

- Calculo del primer término de la derivada de la función de forma $\boldsymbol{p}_{,x}^{T}\boldsymbol{A}^{-1}\boldsymbol{B}_{I}$

```
Matriz<double> MAinvB(2,iNodos);
MAinvB.llenar(0.0);
for (j=1;j<=2;j++)
{
        for(k=1;k<=iNodos;k++)
        {
                MAinvB(j,k)=(MFactorAinv(1,j)*MFactorB(1,k))+
                (MFactorAinv(2,j)*MFactorB(2,k));
```

```
        }
}
Vector<double> VPtdevAinvB(iNodos);
VPtdevAinvB.llenar(0.0);
for(j=1;j<=iNodos;j++)
{
        VPtdevAinvB(j)=MAinvB(2,j);
}
```

- Calculo del segundo término de la derivada de la función de forma $\boldsymbol{p}^T(\boldsymbol{A}^{-1})_{,x}\boldsymbol{B}_I$

```
Matriz<double> MAinvdevB(2,iNodos);
MAinvdevB.llenar(0.0);
for (j=1;j<=2;j++)
{
        for(k=1;k<=iNodos;k++)
        {
                MAinvdevB(j,k)=(MDerivadaFactorAinv(1,j)*MFactorB(1,k))+
                (MDerivadaFactorAinv(2,j)*MFactorB(2,k));
        }
}
```

- Calculo del tercer término de la derivada de la función de forma $\boldsymbol{p}^T\boldsymbol{A}^{-1}\boldsymbol{B}_{I,x}$

```
Matriz<double> MAinvBdev(2,iNodos);
MAinvBdev.llenar(0.0);
for (j=1;j<=2;j++)
{
        for(k=1;k<=iNodos;k++)
        {
```

```
                MAinvBdev(j,k)=(MFactorAinv(1,j)*MDerivadaFactorB(1,k))+
                (MFactorAinv(2,j)*MDerivadaFactorB(2,k));
        }
}
```

- Suma de los términos de la derivada de la función de forma $\boldsymbol{\Phi}_{I,x}$:

```
Matriz<double> MSumaMatrices(2,iNodos);
MSumaMatrices.llenar(0.0);
MSumaMatrices.agregar(MAinvdevB,MAinvBdev);
Vector<double> V2y3TerminoDFF(iNodos);
V2y3TerminoDFF.llenar(0.0);
for (j=1;j<=2;j++)
{
        for(k=1;k<=iNodos;k++)
        {
                V2y3TerminoDFF(k)=V2y3TerminoDFF(k)+
                (VPuntoIntegracionActual(j)*MSumaMatrices(j,k));
        }
}
VDerivadaFuncionForma.agregar(VPtdevAinvB,V2y3TerminoDFF);
```

4.1.3.2 Funciones de Peso. La función de peso, tiene dos cometidos dentro del cálculo de la función de forma los cuales son:

- Provee de pesos a los residuos en diferentes nodos del dominio de influencia.
- Asegura que los nodos entran o salen del dominio de influencia de forma gradual en la medida que x cambia, lo cual garantiza la condición de continuidad de la aproximación entre los subdominios.

La condición más clara que debe cumplir la función de peso w, es que sea una función monótona decreciente, esto indica que los nodos más próximos al actualmente evaluado, tienen más peso que los más alejados. Algunos ejemplos típicos de esta condición se muestran a continuación ref. (2, 6 y 7):

$$Spline\ 3^{er}\ orden\ w(x-x_I) \equiv w(r) = \begin{cases} 4r^3 - 4r^2 + \frac{2}{3} & r \le 0.5 \\ -\frac{4}{3}r^3 + 4r^2 - 4r + \frac{4}{3} & 0.5 < r \le 1 \\ 0 & r > 1 \end{cases} \quad (66)$$

$$Spline\ 4^o\ orden\ w(x-x_I) \equiv w(r) = \begin{cases} -3r^4 + 8r^3 - 6r^2 + 1 & r \le 1 \\ 0 & r > 1 \end{cases} \quad (67)$$

$$Spline\ 2k-esimo\ orden\ w(x-x_I) \equiv w(r) = \begin{cases} (1-r^2)^4 & r \le 1 \\ 0 & r > 1 \end{cases} \quad (68)$$

$$Singular\ w(x-x_I) \equiv w(r) = \begin{cases} r^{-k} - 1 & r \le 1 \\ 0 & r > 1 \end{cases} \quad (69)$$

$$Exponencial\ 1\ w(x-x_I) \equiv w(r) = \begin{cases} e^{-(r/c)^{2k}} & r \le 1 \\ 0 & r > 1 \end{cases} \quad (70)$$

$$Exponencial\ 2\ w(x-x_I) \equiv w(r) = \begin{cases} e^{1/(r^2-1)} & r \le 1 \\ 0 & r > 1 \end{cases} \quad (71)$$

$$Exponencial\ 3\ w(x-x_I) \equiv w(r) = \begin{cases} e^{\frac{r^2}{k(r^2-1)}} & r \le 1 \\ 0 & r > 1 \end{cases} \quad (72)$$

Para la solución de una viga simplemente apoyada se decidió tomar la función de peso de la ecuación (69), luego de calcular los porcentajes de error que se

obtenían utilizando cada una de las ecuaciones anteriormente mostradas, además para dicha función de peso se tomo $k = 5$, debido a que fue el presento mayor estabilidad en la solución.

Para el cálculo de la relación r se tiene que:

$$r = \frac{d_I}{d_{mI}} = \frac{\|x - x_I\|}{d_{max} C_I} \tag{73}$$

Donde d_{mI} es el tamaño del dominio de influencia del $I^{\text{ésimo}}$ nodo, de tal forma que para C_I igual a la distancia entre dos nodos consecutivos, ya que estos están uniformemente distribuidos, se tiene que d_{max} es un factor de escala de la función de forma, que indica el numero de nodos que se encuentran dentro de cada subdominio el cual toma un valor normalmente entre 2.0-4.0. Para nuestro desarrollo se tomo $d_{max} = 2.0$ ya que este valor es el que presenta mayor estabilidad en la solución de la viga simplemente apoyada.

Otro factor importante de calcular es la derivada de la función de peso, que para la función tomada de la ecuación (69) resulta así:

$$\frac{dw_I}{dx} = \frac{dw_I}{dr}\frac{dr}{dx} = \begin{cases} -kr^{-(k+1)} sign(x - x_I) & para\ r \leq 1 \\ 0 & para\ r > 1 \end{cases} \tag{74}$$

Donde la función $sign$ devuelve el signo del argumento $(x - x_I)$, es decir, 1 si es positivo, -1 si es negativo.

En la implantación computacional la función de peso esta previamente definida como se muestra a continuación:

```
VFuncionPeso.llenar(0.0);
VDerivadaFuncionPeso.llenar(0.0);
for(j=1;j<=iNodos;j++)
{
        ddrdx=sign(VDistaniaPuntoIntegracion(j))/VDominioInfluencia(j);
        dRelacion=abs(VDistaniaPuntoIntegracion(j))/VDominioInfluencia(j);
        if (dRelacion<=1.0)
        {
                VFuncionPeso(j)=pow(dRelacion,-5)-1;
                VDerivadaFuncionPeso(j)=(-5.0*pow(dRelacion,-6)*ddrdx;
        }
        else if (dRelacion>1.0)
        {
                VFuncionPeso(j)=0.0;
                VDerivadaFuncionPeso(j)=0.0;
        }
}
```

4.1.4 Formulación del Método De Galerkin Libre De Malla Con Multiplicadores de Lagrange. Considerando el problema de la viga simplemente apoyada, para el cual la ecuación diferencial de la elástica es:

$$\frac{d^2u}{dx^2} - \frac{M(x)}{EI} = 0 \qquad (75)$$

Para obtener as ecuaciones discretas es necesario el uso de la forma débil de la ecuación de equilibrio y las condiciones de frontera. Estas ecuaciones pueden ser obtenidas de forma matricial de la siguiente forma:

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{G} \\ \boldsymbol{G^T} & 0 \end{bmatrix} \begin{Bmatrix} \boldsymbol{u} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f} \\ \boldsymbol{q} \end{Bmatrix} \tag{76}$$

Donde

$$\boldsymbol{K}_{Ij} = \int_0^1 \boldsymbol{\Phi}_{I\,,x}^{T} EI \boldsymbol{\Phi}_{j\,,x} dx \tag{77}$$

$$\boldsymbol{G}_{Ij} = -\boldsymbol{\Phi}_j \big| \Gamma_{uI} \tag{78}$$

$$\boldsymbol{f}_I = \boldsymbol{\Phi}_I t_x \Gamma_t + \int_0^1 \boldsymbol{\Phi}_I b dx \tag{79}$$

$$\boldsymbol{q}_j = -u_j \tag{80}$$

Donde Γ_u y Γ_t son el limite esencial y natural respectivamente, b son las fuerza interiores aplicadas en el dominio que actúan sobre la viga y t las fuerza exteriores

A continuación se muestra el cálculo computacional de la matriz de rigidez y el vector de fuerzas externas, además se le imponen las condiciones de frontera:

- Calculo de $\boldsymbol{K}_{Ij}$ y $\boldsymbol{f}_I$

```
Matriz<double> MDerivadaFuncionForma(iNodos,iNodos);
MDerivadaFuncionForma.llenar(0.0);
```

```
Vector <double> VTerminoFuerzaExternas(iNodos);
VTerminoFuerzaExternas.llenar(0.0);
if (i>1)
{
        for (j=1;j<=iNodos;j++)
        {
                for(k=1;k<=iNodos;k++)
                {
                        MDerivadaFuncionForma(k,j)=(VDerivadaFuncionForma(k)*
                        VDerivadaFuncionForma(j));
                }
        }
        MDerivadaFuncionForma.mult(EI);
        MRigidez.agregar(MRigidez,MDerivadaFuncionForma);
        for(j=1;j<=iNodos;j++)
        {
                VTerminoFuerzaExternas(j)=VFuncionForma(j)*
                VMomentosInternos(j);
        }
        VFuerzasExternas.agregar(VFuerzasExternas,VTerminoFuerzaExternas);
}
```

- Imposición de las condiciones de frontera

```
for(j=2;j<=iNodos-1;j++)
{
        for(k=2;k<=iNodos-1;k++)
        {
                MRigidezTotal(j-1,k-1)=MRigidez(j,k);
        }
```

```
}
for(j=2;j<=iNodos-1;j++)
{
        VFuerzasExternasTotal(j-1)=VFuerzasExternas(j);
}
```

- Calculo de $\boldsymbol{u}$ (VDesplazamiento)

```
MRigidezTotal.factLU();
MRigidezTotal.forwBackLU(VFuerzasExternasTotal,VDesplazamiento);
```

En general los métodos libres de malla llevan 15 años de desarrollo, a pesar de su corta edad, promete ser más eficiente que otros métodos numéricos tradicionales, ya que el desplazamiento de los nodos, no es un obstáculo en su implementación. Es de notar que la exactitud del método se presenta en la medida que se tome de forma adecuada los parámetros que varían dentro del método para el desarrollo de diferentes fenómenos en la ingeniería y en las ciencias puras.

4.2 IMPLEMENTACIÓN DEL MÉTODO DE GALERKIN LIBRE DE MALLA 1D

Al igual que en el desarrollo del método de diferencias finitas se utilizo la herramienta de desarrollo computacional MecLib. La solución computacional para una viga simplemente apoyada está conformada por una clase y un ejecutable, a continuación se presentara el código realizado:

Inicialmente se hace la declaración de variables para el desarrollo por el método de elementos finitos. Cabe señalar que las variables definidas como int y double comienzan con la letra i y d respectivamente y los vectores y matrices comienzan con las letras V y M respectivamente:

Variables privadas

dFactorDominioInfluencia	Factor del dominio de influencia del nodo (2>=dfactordominioinfluencia<=3)
dPuntoIntegracion	Punto de integracion actual
dRelacion	Relación (r=||x-xI||/DominioInfluencia) de la distancia entre nodos y el dominio del nodo
ddrdx	Derivada de r respecto a x
VDominioInfluencia	Vector que contiene el dominio de influencia de cada nodo. (el dominio de influencias es igual para todos los nodos si estos están uniformemente espaciados)
VPosicionNodos	Vector de posición de cada nodo a lo largo de la viga.
VPuntosIntegracion	Vector de posición de los puntos de integración a lo largo de la viga
VDistaniaPuntoIntegracion	Vector que almacena la distancia de cada nodo al punto de integración
VFuncionPeso	Almancena la función del peso nodo actual
VDerivadaFuncionPeso	Almacena la derivada de la función de peso del nodo actual
VFuncionForma	Calcula la función de forma respecto al punto de integración actual
VDerivadaFuncionForma	Calcula la derivada de la función de forma
VFuerzasExternas	Calcula el vector de fuerzas externas
VDesplazamiento	Calcula el desplazamiento de cada nodo
VFuerzasExternasTotal	Contiene el vector fuerzas externas a excepción de las fronteras
MPolinomioAproximacion	Polinomio de aproximación

MFactorB	Matriz para el cálculo del factor B (factor para el cálculo de los coeficientes ***a*** de la función de aproximación, A(x)a(x)=B(x)u
MFactorA	Matriz para el cálculo del factor A (factor para el cálculo de los coeficientes ***a*** de la función de aproximación, A(x)a(x)=B(x)u)
MRigidez	Matriz de rigidez
MRigidezTotal	Contiene la matriz de rigidez a excepción de las fronteras

4.2.1 Descripción Paso a Paso. El código para la solución numérica, de la viga simplemente apoyada se divide en tres etapas, inicialmente se llaman las librerías y la clase creada para las diferencias finitas, de la siguiente forma:

```
#include "StdAfx.h"
#include "SinMalla.h"
#include <iniciarLIB.h>

using namespace std;
int main(int argc, const char * argv[])
{
        iniciarLIB(argc,argv);
        SinMalla problema;
        problema.Leer ();
        problema.ResuelveProblema ();
        problema.Reportar ();
        system("PAUSE");
        return (0);
```

La explicación de cada etapa se muestra a continuación:

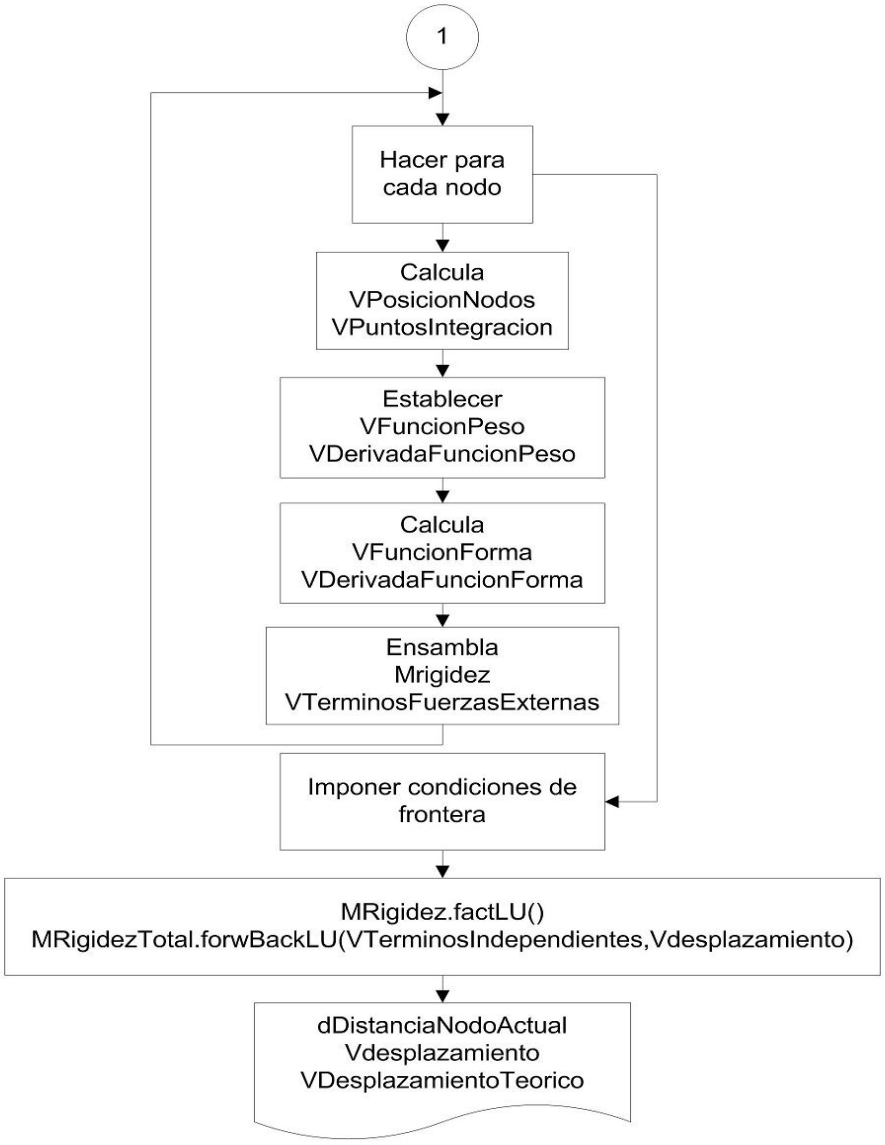
- **problema.Leer ();**

  Se leen las variables tales como geometría y cargas aplicadas sobre la viga y se calculan los momentos que generan las cargas sobre la viga para cada nodo.

- **problema.ResuelveProblema ();**

  Se hace principalmente la distribución de los nodos y los puntos de integración, seguidamente se calcula la función de forma y su derivada, la función de peso y el dominio de influencia ya se encuentran predeterminados. Posteriormente se calcula la matriz de rigidez y el vector fuerzas externas, a los cuales, se les aplican las condiciones de frontera. Finalmente es calculado el vector donde se encuentra el desplazamiento de cada nodo por medio de las funciones factLU y forwBackLu del MecLib. A continuación en la figura (16) se muestra el algoritmo que se utiliza en el desarrollo de esta etapa

Figura 16. Algoritmo para la implementación del método sin malla.



- **problema.Reportar ();**

  Imprime los resultados en un archivo de texto “Reusltados.res” para comparar la respuesta obtenida por el metodo de galerkin libre de malla con el desplazamiento teórico de la viga simplemente apoyada, en cada nodo. A continuación se muestra el desarrollo de esta etapa.

```
void SinMalla :: Reportar ()
{
int i;
Os ofile("Resultados.res",NEWFILE);
ofile << reloj.reportar2("Tiempo de cálculo");
ofile <<"Posición" <<"\t" "Desplazamiento Numérico" <<"\t" "Desplazamiento
Analítico"<<endl;
for(i=1;i<=iNodos;i++)
{
        dDistanciaNodoActual=(i-1)*dLongitudElemento;
        if (i==1)
        {
                ofile << oform("%4.3f   %10.3f   %25.3f\n", 0.0, 0.0, 0.0);
        }
        else if (i>1 && i<iNodos)
        {
                ofile << oform("%4.3f   %17.5e   %25.5e\n",
                dDistanciaNodoActual, VDesplazamiento(i-1),
                VDesplazamientoTeorico(i));
        }
        else if (i==iNodos)
        {
                ofile << oform("%4.3f   %10.3f   %25.3f\n", dLongitud, 0.0, 0.0);
        }
}
ofile->cerrar();
}
```

Los resultados obtenidos son visualizados en un archivo de texto llamado "Resultados.res", de la siguiente forma:

Tiempo de cálculo: 0.0160

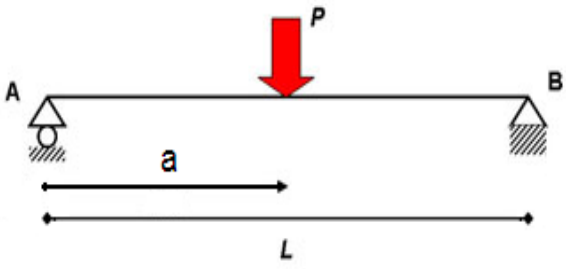
Posición	Desplazamiento Numérico	Desplazamiento Analítico
0.000	0.000	0.000
0.100	-3.19523e-006	-3.19167e-006
0.200	-6.14381e-006	-6.13333e-006
0.300	-8.59168e-006	-8.57529e-006
0.400	-1.02974e-005	-1.02759e-005
0.500	-1.10535e-005	-1.10286e-005
0.600	-1.07221e-005	-1.06963e-005
0.700	-9.26942e-006	-9.24589e-006
0.800	-6.80012e-006	-6.78333e-006
0.900	-3.58542e-006	-3.57917e-006
1.0	0.000	0.000

4.3 ANÁLISIS DE RESULTADOS

En el análisis de resultados para una viga simplemente apoyada se tomaron las mismas configuraciones de carga, geometría y propiedades del material, utilizadas en el método de diferencias finitas con el fin de calcular el porcentaje de error que presenta el método con respecto a la solución analítica, también se mide el tiempo de cálculo con el fin de medir el coste computacional.

Tabla 9. Solución de una viga simplemente apoyada sometida a carga puntual por el Método de Galerkin Libre de Malla.

Carga puntual



Numero de Nodos	100
L	1 m.
EI	2400000 N-m²
Numero de cargas	1
Tipo de carga	1
P	10000 N.
a	0,5 m.
%Error	0,18%
Tiempo de calculo	0.4220 s.

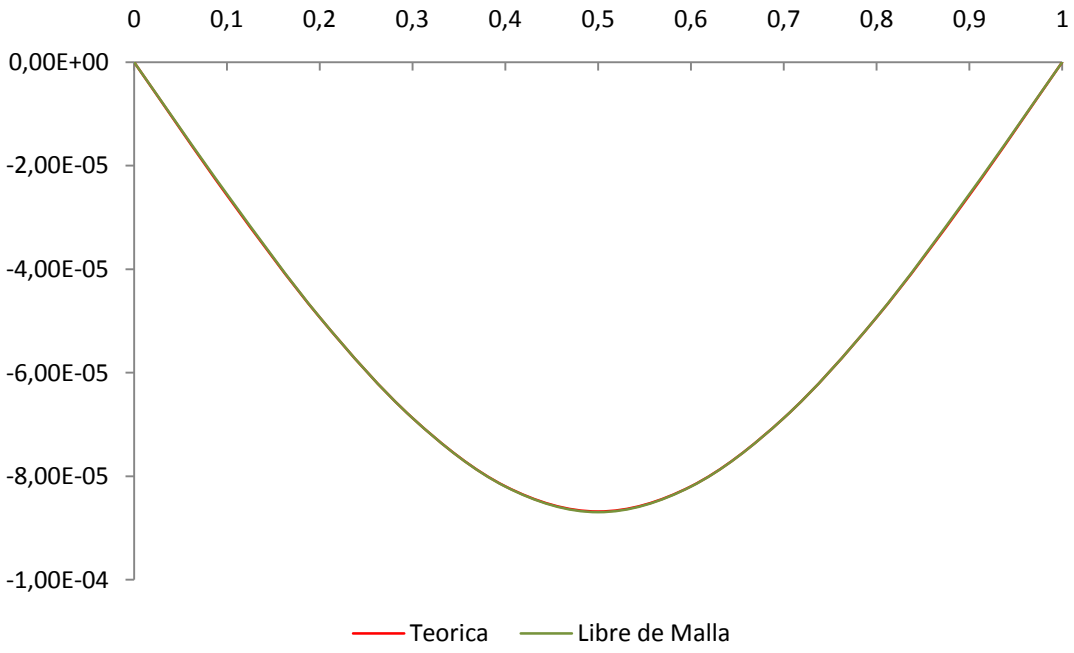
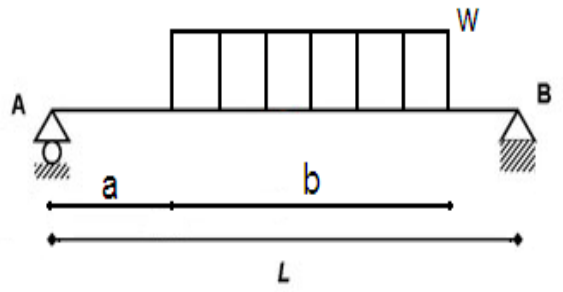


Tabla 10. Solución de una viga simplemente apoyada sometida a carga distribuida rectangular por el Método de Galerkin Libre de Malla.

Carga Distribuida rectangular



Numero de Nodos	11
L	1 m
EI	2400000 N-m²
Numero de cargas	1
Tipo de carga	2
W	5000 N/m
a	0,3 m
b	0,6 m
%Error	0,17%
Tiempo de calculo	0.4530 s.

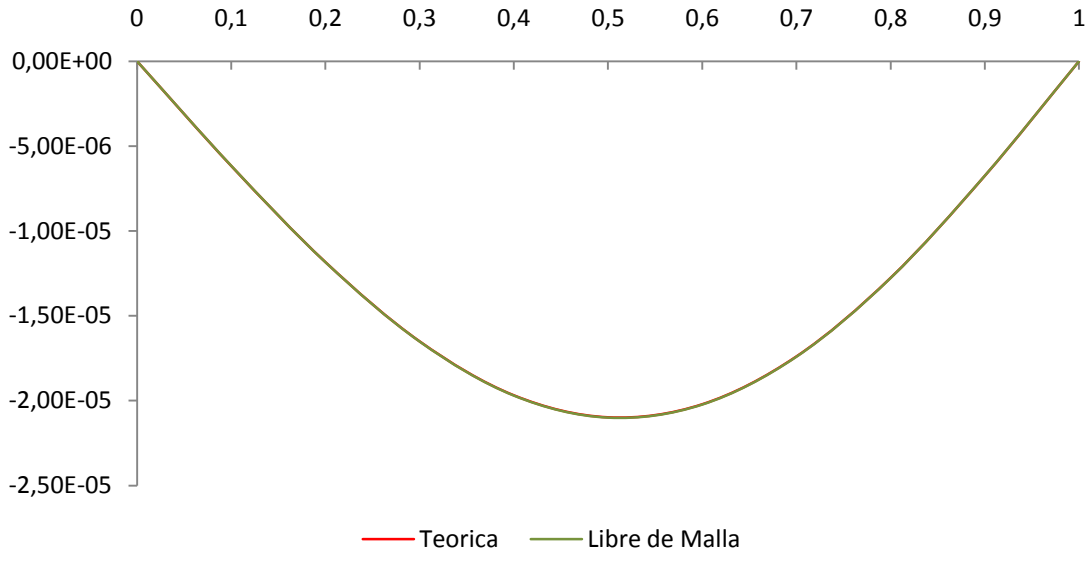
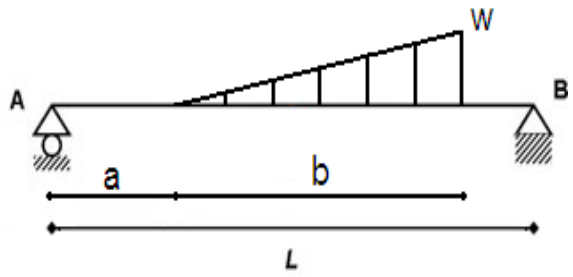
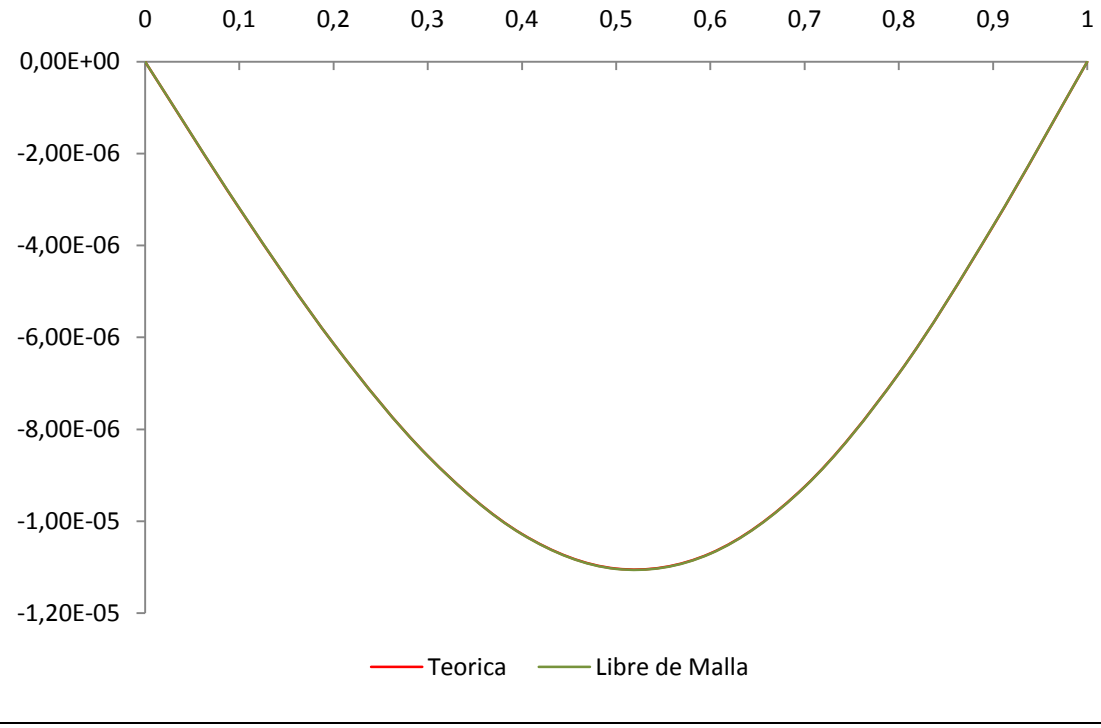


Tabla 11. Solución de una viga simplemente apoyada sometida a carga distribuida triangular por el Método de Galerkin Libre de Malla.

Carga Distribuida triangular 1		
	Numero de Nodos	11
	L	1 m
	EI	2400000 N-m²
	Numero de cargas	1
	Tipo de carga	3
	W	5000 N/m
	a	0,2 m
	b	0,6 m
	%Error	0,17%
	Tiempo de calculo	0.4370s.



4.4 CONCLUSIONES

1. Se observa con gran satisfacción, que el porcentaje de error presentado por el método de Galerkin libre de malla se encuentra por debajo de 0,2%, por tal razón se puede asumir que este un valor lo suficientemente confiable. Este porcentaje de error puede disminuir en la media que se aumenten los puntos de integración o se seleccione un polinomio de aproximación de base superior.

2. Sin lugar a dudas la ausencia de una malla es una gran ventaja, ya que la conectividad entre nodos se realiza en tiempo real y reduce el coste de generar una malla al inicio del proceso del análisis. Es de notar que el cálculo de tiempo para cada configuración es relativamente alto, esto se debe a que la herramienta computacional utilizada MecLib, presenta algunas limitaciones en el momento de realizar operaciones con matrices y vectores.

3. La selección de la función de peso es muy importante en el desarrollo del método ya que el error de la solución numérica depende de la aproximación que realice la función de forma. Por lo tanto es importante tener presente cual es la función de forma que más se acopla al fenómeno estudiado, con el fin de obtener el mínimo porcentaje de error en la solución de cada problema.

4. El método de Galerkin libre de malla, tiene una formulación matemática relativamente sencilla de realizar, tanto analíticamente como numéricamente.

5. Por último se puede decir que la implantación de método de Galerkin libre de malla, requiere una concisa selección de los siguientes parámetros:

- Se debe considerar, en primer lugar, un número finito de nodos que se encuentran en el dominio, teniendo en cuenta que a mayor numero de nodos el error tiende a reducirse, pero el coste computacional aumenta notablemente.

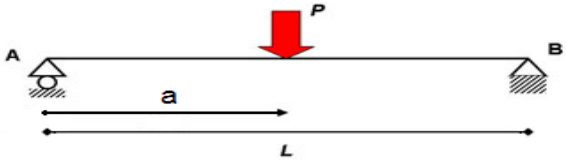
- En segundo lugar, el tamaño del dominio de soportes d_{max}, el cual no es más que un factor de escala de la función de forma, que indica el número de nodos que se encuentran dentro de cada subdominio. Se ha encontrado, para la solución de una viga simplemente apoyada que el valor ideal es de 2.0, ya que un valor superior o inferior aumenta drásticamente el porcentaje de error con respecto a la solución analítica.
- El siguiente parámetro a tener en cuenta es la influencia de la base del polinomio de aproximación que da lugar a la función de forma, de tal forma que error será menor en la medida que este polinomio se parezca más al polinomio de la aplicación numérica.
- Posteriormente el cuarto parámetro es la definición del numero de los puntos de Gauss, el cual no tiene mayor influencia en el porcentaje de error con respecto a la solución analítica, pero si aumenta el tiempo de cálculo.

5. ANÁLISIS DE RESULTADOS FINALES.

A continuación se hará el análisis de los resultados comparando los tres métodos numéricos expuestos anteriormente, con la solución analítica, los datos de geometría, propiedades del material y valores de cargas y distancias se hará de forma similar a los mostrados anteriormente.

Tabla 12. Solución teórica y numérica de una viga simplemente apoyada sometida a carga puntual.

Carga puntual



Numero de Nodos	11
L	1 m.
EI	2400000 N-m²
Numero de cargas	1
Tipo de carga	1
P	10000 N.
a	0,5 m.

Posición	Desplazamiento				%Error D. Finitas	%Error Sin malla	%Error E. Finitos
	Teórica	Dif Finitas	Sin malla	E. Finitos			
0	0	0	0	0	0,00	0,00	0,00
0,1	-2,57E-05	-2,60E-05	-2,55E-05	-2,60E-05	1,35	0,79	1,35
0,2	-4,93E-05	-5,00E-05	-4,93E-05	-5,00E-05	1,41	0,07	1,41
0,3	-6,88E-05	-6,98E-05	-6,88E-05	-6,98E-05	1,52	0,01	1,52
0,4	-8,19E-05	-8,33E-05	-8,20E-05	-8,33E-05	1,69	0,05	1,69
0,5	-8,68E-05	-8,85E-05	-8,69E-05	-8,85E-05	2,00	0,14	2,00
0,6	-8,19E-05	-8,33E-05	-8,20E-05	-8,33E-05	1,69	0,05	1,69
0,7	-6,88E-05	-6,98E-05	-6,88E-05	-6,98E-05	1,52	0,01	1,52
0,8	-4,93E-05	-5,00E-05	-4,93E-05	-5,00E-05	1,41	0,07	1,41
0,9	-2,57E-05	-2,60E-05	-2,55E-05	-2,60E-05	1,35	0,79	1,35
1	0	0	0	0	0,00	0,00	0,00
Tiempo de cálculo(s)		0,016	0,422	0,0160	**1,27**	**0,18**	**1,27**

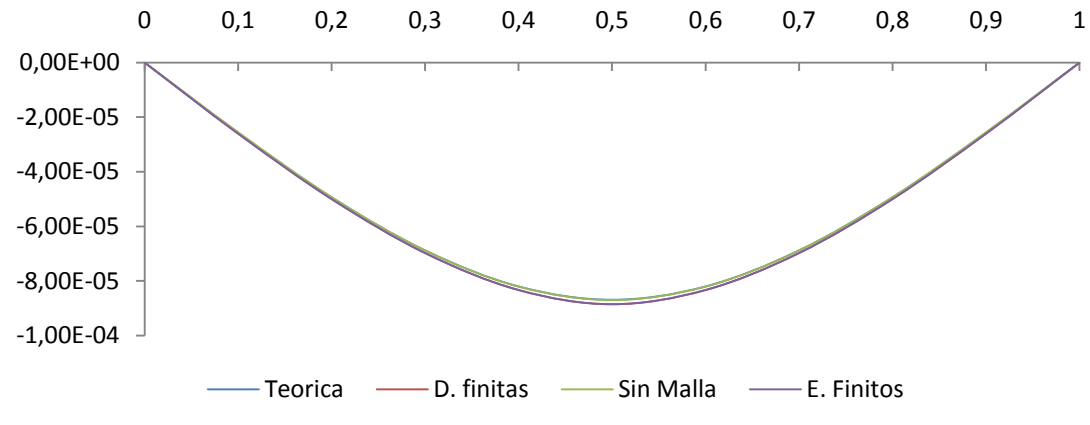
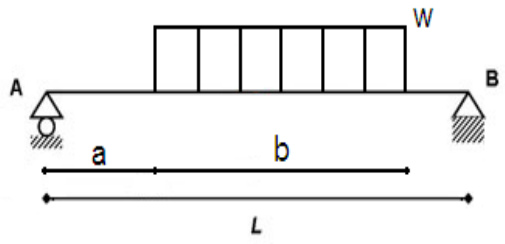


Tabla 13. Solución teórica y numérica de una viga simplemente apoyada sometida a carga distribuida rectangular.

Carga distribuida rectangular



Numero de Nodos	11
L	1 m.
EI	2400000 N-m²
Numero de cargas	1
Tipo de carga	2
w	5000 N-m.
a	0,3 m.
b	0,6 m

Posición	Desplazamiento				%Error D. Finitas	%Error Sin malla	%Error E. Finitos
	Teórica	Dif Finitas	Sin malla	E. Finitos			
0	0	0	0	0	0,00	0,00	0,00
0,1	-6,17E-06	-6,21E-06	-6,17E-06	-6,21E-06	0,68	0,12	0,68
0,2	-1,18E-05	-1,19E-05	-1,19E-05	-1,19E-05	0,70	0,18	0,70
0,3	-1,65E-05	-1,66E-05	-1,65E-05	-1,66E-05	0,76	0,20	0,76
0,4	-1,97E-05	-1,98E-05	-1,97E-05	-1,98E-05	0,80	0,22	0,80
0,5	-2,10E-05	-2,11E-05	-2,10E-05	-2,11E-05	0,83	0,23	0,83
0,6	-2,02E-05	-2,04E-05	-2,02E-05	-2,04E-05	0,85	0,23	0,85
0,7	-1,74E-05	-1,75E-05	-1,74E-05	-1,75E-05	0,88	0,24	0,88
0,8	-1,28E-05	-1,29E-05	-1,28E-05	-1,29E-05	0,91	0,24	0,91
0,9	-6,75E-06	-6,81E-06	-6,76E-06	-6,81E-06	0,93	0,19	0,93
1	0	0	0	0	0,00	0,00	0,00
Tiempo de cálculo(s)		0.0160	0.4530	0,0320	**0,67**	**0,17**	**0,67**

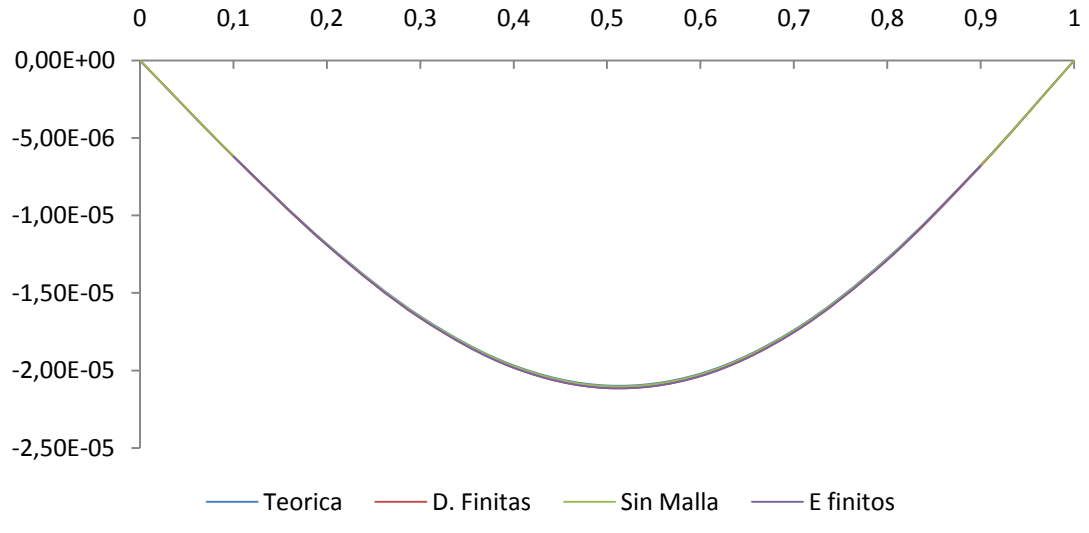
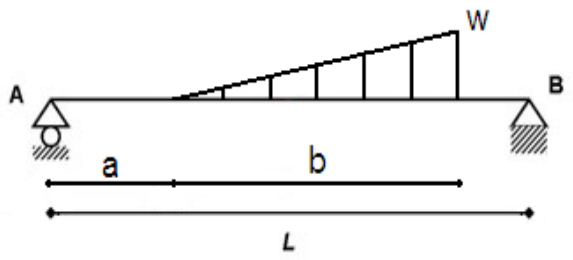


Tabla 14. Solución teórica y numérica de una viga simplemente apoyada sometida a carga distribuida triangular.

Carga distribuida triangular



Numero de Nodos	11
L	1 m.
EI	2400000 N-m²
Numero de cargas	1
Tipo de carga	3
w	5000 N-m.
a	0,2 m.
b	0,6 m

Posición	Desplazamiento				%Error D. Finitas	%Error Sin malla	%Error E. Finitos
	Teórica	Dif Finitas	Sin malla	E. Finitos			
0	0	0	0	0,00	0,00	0,00	0,00
0,1	-3,19E-06	-3,21E-06	-3,20E-06	-3,21E-06	0,65	0,11	0,65
0,2	-6,13E-06	-6,17E-06	-6,14E-06	-6,17E-06	0,68	0,17	0,68
0,3	-8,58E-06	-8,64E-06	-8,59E-06	-8,64E-06	0,72	0,19	0,72
0,4	-1,03E-05	-1,04E-05	-1,03E-05	-1,04E-05	0,77	0,21	0,77
0,5	-1,10E-05	-1,11E-05	-1,11E-05	-1,11E-05	0,83	0,23	0,83
0,6	-1,07E-05	-1,08E-05	-1,07E-05	-1,08E-05	0,88	0,24	0,88
0,7	-9,25E-06	-9,33E-06	-9,27E-06	-9,33E-06	0,93	0,25	0,93
0,8	-6,78E-06	-6,85E-06	-6,80E-06	-6,85E-06	0,92	0,25	0,92
0,9	-3,58E-06	-3,61E-06	-3,59E-06	-3,61E-06	0,88	0,17	0,88
1	0	0	0	0,00	0,00	0,00	0,00
Tiempo de cálculo(s)		0.0160	0.4370	0.0160	**0,67**	**0,66**	**0,17**

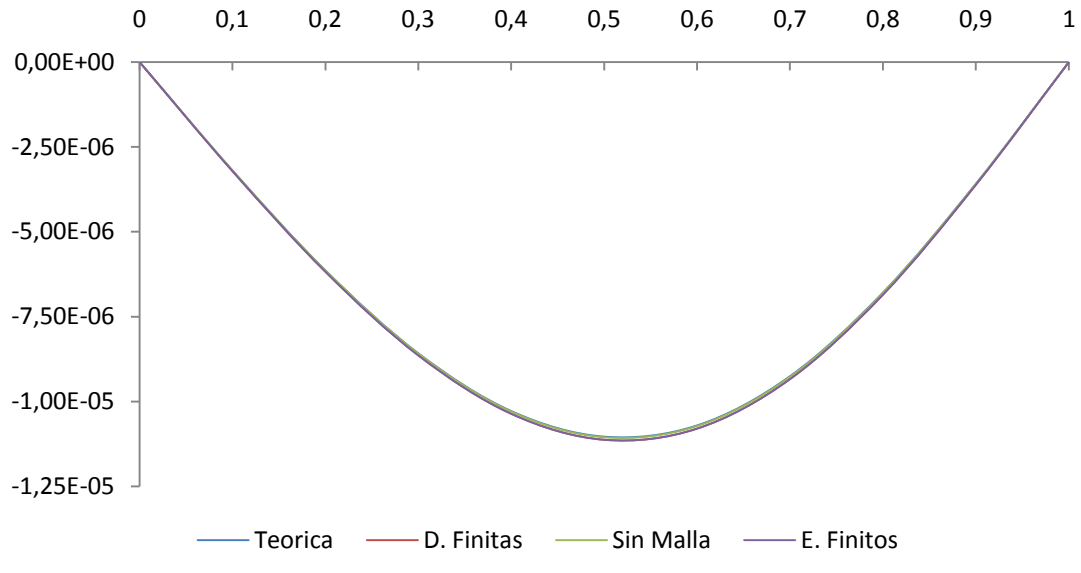
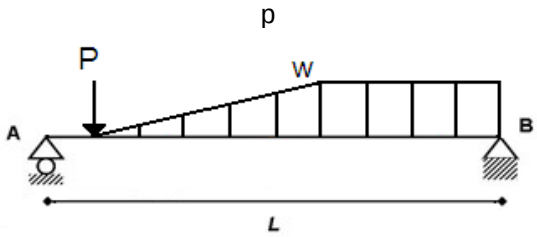


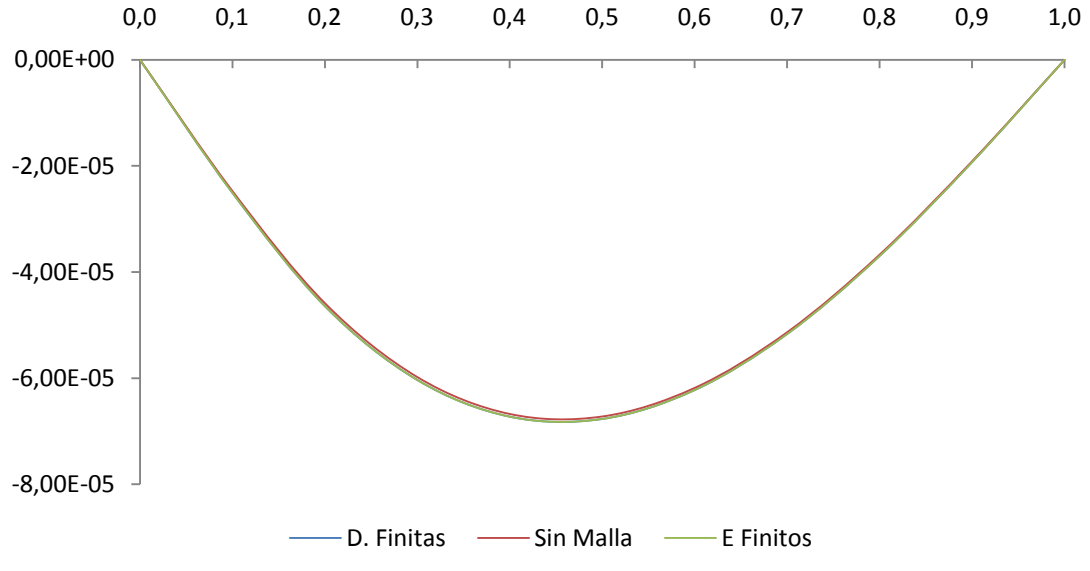
Tabla 15. Solución numérica de una viga simplemente apoyada sometida a diferentes tipos de carga.

Configuración de diferentes tipos de cargas



Numero de Nodos	11
L	1m
EI	2400000 N-m²

Distribución de las cargas		Posición	Dif Finitas	Sin malla	E. Finitos
Numero de cargas	3				
Tipo de carga	1	0,0	0,0	0,0	0,00E+00
P	10000 N.	0,1	-2,51E-05	-2,47E-05	-2,51E-05
a	0,2 m.	0,2	-4,64E-05	-4,59E-05	-4,64E-05
Tipo de carga	3	0,3	-6,03E-05	-5,98E-05	-6,03E-05
w	5000 N-m.	0,4	-6,72E-05	-6,67E-05	-6,72E-05
a	0,2 m.	0,5	-6,77E-05	-6,72E-05	-6,77E-05
b	0,4 m	0,6	-6,22E-05	-6,18E-05	-6,22E-05
Tipo de carga	2	0,7	-5,17E-05	-5,13E-05	-5,17E-05
w	5000 N-m.	0,8	-3,69E-05	-3,67E-05	-3,69E-05
a	0,6 m.	0,9	-1,93E-05	-1,91E-05	-1,93E-05
b	0,4 m	1,0	0,0	0,0	0,00E+00



6. CONCLUSIONES GENERALES

- Con el desarrollo de esta tesis se ha podido demostrar que los métodos numéricos convencionales utilizados para análisis estructural en el área de mecánica computacional aunque presenten un porcentaje de error aceptable dentro de la solución a este tipo de problemas están siendo superados notablemente por los métodos de elementos sin malla debido a su bajo porcentaje de error en la solución.

- Una de las implicaciones de no tener que definir una malla es que la conectividad de los nodos se puede hacer en tiempo real haciendo que la adaptabilidad de los nodos sea relativamente simple y automatizable, mientras que con el método de elementos finitos de debe predefinir una malla al comenzar el proceso de desarrollo aumentando con ello el costo computacional.

- A pesar de las ventajas que tiene el método sin malla sobre los métodos convencionales (FEM y FDM) una de las pocas desventajas que tiene este método sobre los otros es que presenta una mayor complejidad en la resolución de algoritmos ya que se debe calcular dentro de él, las funciones de forma, cosa que no sucede con los métodos de elementos finitos y diferencias finitas.

- El método de los mínimos cuadrados móviles posee una ventaja especial sobrelos otros métodos de construcción de funciones de forma, ya que se generan funciones linealmente dependientes, haciendo que cualquier combinación de las funciones generen las mismas funciones de forma, esto implica que al utilizar cualquier base siempre se obtendrá la misma función de forma.

- Una desventaja que tiene el método sin mallas con relación al método de elementos finitos es que al utilizar los mínimos cuadrados móviles (MLS) para la construcción de las funciones de forma, estas no poseen la propiedad del delta de Kronecker, lo que conlleva a un aumento del error en la solución de problemas de contorno.

- El método de Galerkin libre de malla, por su misma complejidad, resulta tener un costo computacional más elevado, pero este gasto se puede ver representado en la exactitud de las soluciones. Por lo tanto aunque el beneficio no se va a ver altamente representado en la solución de fenómenos sencillos, si van a ser de gran utilidad en la solución de análisis más complejos.

- A través del desarrollo de este proyecto de grado se puede concluir satisfactoriamente el cumplimiento de los objetivos planteados inicialmente, dentro de los cuales está el de realizar un documento que contenga la sustentación de los conceptos teóricos necesarios para la solución de un problema de flexión por medio del estudio de los elementos sin malla.

- Para el desarrollo de la aplicación de el problema de una viga simplemente apoyada sometida a flexión se utilizo el lenguaje de programación C++ el cual nos permitió utilizar las librerías **MecLib** adaptadas y puestas a punto por el profesor *David Alfredo fuentes Díaz* .

REFERENCIAS BIBLIOGRÁFICAS

1. **Larry J. Segerlind** (1984), "*Applied finite element analysis* (Secon ed), New York, U.S.A., Copyright.

2. **G. R. Liu** (2003), *Mesh free methods: moving beyond the finite element method* (First ed), Texas, U.S.A., CRC Press LLC.

3. **Youping Chen, James D. Lee and Azim Eskandarian** (2006), *"Meshless Methods in Solid Mechanics"*, Washington, DC, U.S.A., Springer.

4. **J. P. Ponthot and T. Belytschko** (1997), "*Arbitrary Lagrangian-Eulerian formulation for element-free Galerkin method*", Northwestern University, Chicago, U.S.A.

5. **J. Dolbow and T. Belytschko** (1998), "*An introduction to programming the meshless element free Galerkin method*", Northwestern University, Chicago, U.S.A.

6. **T. Fries and H. Matthies** (2004) "*Classification and Overview of Meshfree Methods"*, Brunswick, Alemania, Copyright

7. **Q. Zeng and D. Lu** (2005) *"Galerkin Meshless Methods Based on Partition of Unity Quadrature"*, University of Science and Technology of China, Hefei, China

8. **T. Belytschko, Y.Y. Lu and L. Gu** (1994). *"Element Free Galerkin Methods"*, Northwestern University, Chicago, U.S.A.

9. **T. Belytschko, Y. Krongauz, M. Fleming, D. Organ and W.K. Liu** (1996). *"Smoothing and accelerated Computations in the Element Free Galerkin Method"*, Northwestern University, Chicago, U.S.A.

10. **Liu, W. K., S. Jun and L. Gu** (1995), *"Reproducing Kernel Particle Methods"*, Northwestern University, Chicago, U.S.A.

11. **Duarte, C.A. and J.T. Oden** (1996), *"H-p Clouds – An h-p Meshless Method"*, Texas Institute for Computational an Applied Mathematics, Texas, U.S.A.

12. **Monaghan, J.J.** (1992) "Smooth Particle Hydrodynamics", Monash University, Clayton, Australia.

13. **Belytschko, T., Y. Krongauz, D. Organ, M. Fleming and P. Krysl** (1996), *"Meshless Methods: An Overview and Recent Developments"*, Northwestern University, Chicago, U.S.A.

14. **Belytschko, T., D. Organ and Y. Krongauz** (1995), *"A Coupled Finite Element – Elment Free Galerkin Method",* Northwestern University, Chicago, U.S.A.