1 CAPITULO I INTRODUCCION

1.1 El problema.

El método de elementos finitos es un método utilizado en amplios campos de la ingeniería, mediante este método es más accesible la realización de programas para la solución de distintos problemas que puedan tenerse en ingeniería.

Un problema frecuentemente visto en la carrera de ingeniería civil de la Universidad Autónoma Juan Misael Saracho es el no tener conocimiento necesario sobre el manejo de software computarizado estructural o el de tener conocimiento, pero no saber cómo funciona internamente dicho software debido a distintos factores, los cuales podemos enumerarlos a continuación:

- Falta de recursos para poder realizar cursos.
- Los programas de análisis estructural tienen un costo.
- Dificultad de manejar programas basados en lenguajes extranjeros.
- Falta de tiempo debido a horarios de clases.
- Las personas que saben manejar un software comercial como el Sap2000 no saben cómo éste calcula y determina resultados.

Asimismo otro problema es averiguar en qué medida se ajusta el método de elementos finitos al estudio del análisis estructural y por qué los software comerciales trabajan con dicho método

1.1.1 Antecedentes.

El método de elementos finitos fue propuesto primero en 1943, pero no fue hasta 1956 que se presentaron los primeros resultados obtenidos mediante este método y en 1960 se llamó al método como se le conoce ahora. A través de este lapso de tiempo se presentaron una serie de acontecimientos que aportaron significativamente al desarrollo y optimización del método y gracias a los adelantos tecnológicos el método se pudo implementar poco a poco al análisis computacional, llegando al punto de que la gran mayoría de paquetes computacionales para

solución de los diferentes problemas de ingeniería cuentan con este método implícito en su código fuente.

En la actualidad el método es empleado en una gran cantidad de programas y textos, permitiendo una fácil consecución y aprendizaje, con lo cual se convierte en una herramienta óptima para los ingenieros. El concepto básico de este método es el de dividir el continuo en un número finito de elementos (de allí su nombre), es decir discretizar el continuo y resolver sobre cada uno de los elementos las ecuaciones del sistema para después ensamblar la solución total.

1.1.2 Planteamiento.

El presente trabajo pretende incentivar la elaboración de software por medio del método de elementos finitos y su posterior aplicación en los diferentes campos de la ingeniería. La implementación del software de ingeniería ha tomado mucha fuerza en los últimos años y por esta razón los estudiantes y los mismos ingenieros deben estar a la vanguardia de la creación, el mercado y aplicación de estos programas computacionales para tener una buena competitividad en el mundo laboral y tecnológico.

El uso continuo de la programación en algunas actividades y campos hacen que sea una obligación de los ingenieros conocer acerca de esto y estar preparados para dar soluciones algorítmicas sencillas, pero efectivas a problemas que se pueden presentar a diario, con la creación de software se brinda oportunidad de expandir el conocimiento logrado no sólo en el campo de la programación de computadores, sino también en la aplicación del conocimiento adquirido durante el tiempo de preparación como ingeniero.

Se propone un proyecto sobre uno de los más importantes métodos computacionales en Ingeniería: El Método de los Elementos Finitos, pero aplicado en el campo de la Ingeniería Civil y más concretamente al Análisis Estructural de estructuras.

Existen distintos Software en el mercado que nos sirven para poder realizar el análisis de estructuras en ingeniería civil, lo que se pretende es ofrecer un material alternativo a estos paquetes realizado en un lenguaje sencillo de programación, en español, con facilidad de entendimiento y una fácil introducción de datos, con un código abierto que permita la

ampliación y mejoramiento de dicho programa a futuro por algún otro estudiante, docente interesado o por el mismo autor.

1.1.3 Formulación.

1er. Alternativa.- Realizar un programa que evalué el cálculo matricial de las siguientes estructuras:

- Vigas.
- Pórticos.
- Armaduras.

2da. Alternativa.- Realizar un programa que evalúe el cálculo matricial de placas planas.

1.1.4 Sistematización.

De las alternativas planteadas se opta por utilizar la primera, debido al mayor conocimiento adquirido y mayor bibliografía encontrada para dicho análisis, dejando abierta la segunda opción para un trabajo posterior que pueda ser realizado por estudiantes o docentes interesados, como por el propio autor, para así poder ir mejorando el programa de acuerdo a las distintas necesidades que se requieran en el análisis estructural.

1.2 Objetivos.

El presente trabajo de investigación tiene los siguientes objetivos:

1.2.1 General.

- Demostrar que mediante el Método de Elementos Finitos se puede realizar un software académico confiable y analizar en qué medida se ajustan los resultados arrojados por dicho método al utilizado por softwares comerciales.

1.2.2 Específicos.

- Diseñar un software computarizado mediante el método de elementos finitos para la resolución de vigas, pórticos y armaduras planas.

- Crear algoritmos de programación sencillos, de fácil entendimiento en la introducción de datos, así como también para la salida y visualización de resultados.
- Programar el software computarizado M.E.F. en idioma español utilizando el lenguaje de programación Matlab, más específicamente su entorno grafico denominado Guide.
- Presentar los resultados obtenidos en una hoja de cálculo de Excel en donde se muestre paso a paso la resolución de cada estructura ingresada.
- Aportar a la Universidad Autónoma Juan Misael Saracho el software computarizado M.E.F. de forma gratuita junto con su código de programación abierto y un tutorial de manejo de dicho software.
- Incentivar el desarrollo y ampliación del programa mencionado así como también incentivar la creación de nuevos programas mediante elementos finitos en distintos campos que puedan ser utilizados por alumnos de la Universidad Autónoma Juan Misael Saracho aprovechando cursos que se vayan dando para poder despertar la curiosidad de los estudiantes para poder estudiar mas a fondo este método universal de cálculo.
- Realizar la comprobación de los resultados del software computarizado M.E.F. con programas comerciales, en este caso el Sap2000 v14.

1.3 Justificación.

Las razones por las cuales se elabora el trabajo de investigación son las siguientes:

1.3.1 Teórica.

Los avances tecnológicos obligan al conocimiento cabal de los métodos más precisos para el cálculo en la ingeniería, donde el error cada día se vuelve mínimo, por lo tanto es necesario el conocimiento analítico y computacional de teorías de cálculo más precisas y exactas, es deber del ingeniero civil el conocer en su totalidad estos métodos modernos para poder realizar aportes que puedan ayudar a simplificar problemas que se dan en la vida laboral.

Con el nacimiento de los microcomputadores, el uso de métodos matriciales alcanzó un extraordinario desarrollo debido a la posibilidad de efectuar cálculos a grandes velocidades.
Entre 1945-1955 aparecen los primeros artículos referentes a un nuevo método de análisis que usaba matrices de flexibilidad o de rigidez de la estructura.
Los métodos matriciales surgen de necesidades en la industria aeronáutica.
En Ingeniería Estructural se necesitaban métodos que permitieran hacer diseños cada más complejos.
Los métodos matriciales tienen dos grandes variantes: el método de la flexibilidad, en el cual las incógnitas son las fuerzas y, el método de la rigidez, en el cual las incógnitas son los desplazamientos. Este enfoque se trabaja en todos los métodos del análisis estructural. Sin embargo, por ventajas computacionales el método de la rigidez ha ganado más aceptación.

1.3.2 Metodológica.

En base a conocimientos adquiridos en el campo de ingeniería y más exclusivamente en el campo de análisis matricial de estructuras y utilizando el método de la rigidez se pretende elaborar un software para el análisis matricial de estructuras mediante la herramienta de programación Matlab que tiene su propio código y lenguaje de programación, la metodología que se seguirá será primeramente trabajar en la introducción de datos, posteriormente en el cálculo de la estructura y finalmente en la salida de datos y presentación de resultados.

1.3.3 Práctica.

Con este trabajo de investigación se podrá ayudar en el mejor entendimiento del estudiante sobre los métodos matriciales, así también en la aplicación práctica que tienen dichos métodos al realizar programas informáticos con algoritmos sencillos para la realización de software que faciliten el cálculo de estructuras en ingeniería civil.

1.4 Marco de Referencia.

1.4.1 Espacial

El presente trabajo de aplicación tendrá lugar en la Ciudad de Tarija-Provincia Cercado del Departamento de Tarija.

1.4.2 Temporal

El siguiente trabajo de aplicación se realizará durante el primer periodo de gestión académica 2015 y puede servir de base a futuros proyectos interesados en el objeto de estudio.

1.5 Alcance del estudio.

1.5.1 Tipo de estudio.

Estudio investigativo, donde se consultará la diferente bibliografía con la que se pueda realizar un software práctico que permita la resolución de vigas, pórticos y armaduras en dos dimensiones utilizando el método de los elementos finitos.

1.5.2 Restricciones o limitaciones

El estudio se regirá en el siguiente alcance:

- Las vigas, pórticos y armaduras que deberán ser necesariamente resueltas por el programa a crear serán planas, es decir, el programa solamente se limitara al cálculo de estructuras en dos dimensiones.
- El programa trabajara utilizando secciones constantes en cada barra y asumiendo que los materiales serán homogéneos a lo largo de toda su longitud.
- En el caso de armaduras las cargas deberán ser puntuales y estar en los nudos, es decir que se debe idealizar la armadura.
- No se consideraran voladizos ni articulaciones en medio de las barras.
- En el caso de apoyo móvil, dicho apoyo deberá tener su desplazamiento en el eje X.
- No se permiten apoyos inclinados.
- Las barras de cada estructura deberan ser rectas a lo largo de su eje longitudinal

1.5.3 Premisas

Como parte del método de este trabajo, se plantearan las siguientes premisas que ayudaran en la simplificación del problema:

- El material se considera macizo (Continuo).
- El material de la pieza es homogéneo.

- El material es isótropo
- Las cargas son estáticas o cuasi estáticas.

1.5.4 Hipótesis

El método de elementos finitos puede programarse computacionalmente para diseñar un software académico que determine resultados confiables, comprobados realizando una comparación con un software comercial dando resultados que alcancen ±10% de presicion aproximadamente.

2 CAPITULO II METODO DE LOS ELEMENTOS FINITOS

2.1 Breve historia de los elementos finitos.

Los Conceptos de discretización numérica para resolver problemas de Ciencia e Ingeniería son la base para la formulación del método de elemento finito. La aproximación geométrica más antigua lleva a las pirámides egipcias de 5.000 años. Por otro lado la aproximación numérica podría registrarse históricamente en China, Egipto y Grecia.

Los registros muestran que los chinos calcularon el valor aproximado de π en el primer siglo de nuestra era, con un valor de 3.1547 siendo usado para calcular el volumen de un cilindro.

En el segundo siglo E.C. el astrónomo Chang Heng aproximó el valor de π como 3.146614245 en la Dinastía oriental de Jihn (265-317 E.C) en su comentario de matemáticas usó un polígono regular inscrito en una circunferencia para poder aproximar π la cual halló un valor de 3.1416 (3927/1250); es interesante notar que el uso un polígono de 3072 lados, es decir elementos finitos. De acuerdo con el manuscrito Ahmes, se muestra que para 1.500 A.C., los Egipcios usaban como valor de π = 3,1416. Un papiro de tiempos más tempranos, ahora en Moscú, indica que los egipcios usaron la fórmula para el volumen de una pirámide y el área de un círculo de manera aproximada; en 1.800 A. C. Arquímedes usó el concepto de elementos finitos para calcular volúmenes.

En el contexto estructural, las soluciones tanto en elasticidad como en análisis estructural tuvieron un inicio del Método del Elemento Finito con Timoshenko, pero si se considera que el análisis de marcos establece el inicio del método del Elemento Finito, entonces los pioneros fueron Castigliano, Mhor y Maxwell, entre otros, en el periodo 1850-1875.

En 1915, Maney de los Estados Unidos de Norteamérica, presentó el método pendiente-deformación, expresando los momentos en términos de desplazamientos lineales y angulares en los nodos de la estructura, lo cual es una de las formulaciones para plantear el método de las rigideces y un desarrollo similar, fue planteado por Ostenfeld en Dinamarca. En el año 1929, Hardy Cross hizo público un método para analizar marcos basado en distribuciones angulares, el cual se utilizó por los siguientes 35 años.

En forma paralela a los primeros trabajos sobre análisis de estructuras reticulares, se resolvieron problemas de mecánica del medio continuo usando una analogía con estructuras formadas por barras diagonales para generar mallas con elementos triangulares. A principios de los años cuarenta Courant propuso funciones de interpolación polinomiales por secciones para formular sub regiones triangulares como un caso especial del método variacional de Rayleigh-Ritz, que obtiene soluciones aproximadas.

Actualmente, el método del elemento finito es utilizado con la ayuda de las computadoras, lo cual ha contribuido a su desarrollo al mismo ritmo que las computadoras.

Las publicaciones clásicas por Argyris y Kelsey a mediados de los 50, hicieron surgir los conceptos de análisis de marcos, discretizando no sólo en nodos, sino además en puntos intermedios de las barras y análisis de un continuo, lo que marcó un crecimiento explosivo en el método del elemento finito.

Basándose en el planteamiento estático del elemento finito, se han ampliado las aplicaciones que incluyen diversos efectos físicos y vibraciones en el Análisis dinámico, pandeo y post-pandeo, no linealidades en la geometría y en el material, efectos térmicos, interacción entre fluidos y estructuras, Aero elasticidad, interacción acústica-estructura, teoría de la fractura, estructuras laminadas, propagación de oleaje, dinámica estructural, respuesta dinámica aleatoria y muchas más aplicaciones.

Como una consecuencia de tantos campos de estudio, el uso de los programas de computadora orientados a cada caso, se han convertido en una práctica en los sitios involucrados en el análisis estructural.

2.2 Aplicaciones del método.

Existen gran número de estructuras que su paso de revisión, tanto como post y pre proceso de diseño, son sometidos a rigurosos procesos de evaluación, tales como verificación de cortantes y flectores y saber si el diseño dado es correcto.

2.2.1 Aplicaciones al análisis estructural.

En programas como el Sap2000, el staad pro, tanto en el análisis de estructuras como en el análisis de suelo-estructura, así como el Safe o el programa Plaxis que son de uso para el cálculo por elementos finitos.

2.2.2 Aplicación en mecánica de fluidos.

El método de elementos finitos también puede ser usado en el análisis de mecánica de fluidos, librerías en python como ECOASTER o el Fenics o programas como Abaqus, Nastram nos pueden dar una idea de cómo este elemento físico se mueve e interacciona con el mundo aplicando soluciones particulares a la formulación de la ecuación general de los fluidos Navier y Stokes, idealizaciones de presas, así como simulaciones de comportamiento de las turbulencias de los fluidos en turbinas hidráulicas.

2.3 Análisis de vigas

El análisis de vigas para el estudio de elementos finitos, se hace para entender en forma elemental los diferentes tipos de discretizaciones a elementos de simples a complejos, ya que para poder entender los análisis y cálculo de placas y sólidos de revolución es necesario entender los diferentes ítems del estudio de vigas, tanto sometidas a fuerzas axiales como a flexiones, que pueden ser tratadas por teorías muy conocidas como Bernoulli y Timoshenko, sobre todo entendimiento de condiciones de contorno para pasar a estadios más complejos.

2.3.1 Análisis de flexión de vigas.

El clásico problema de vigas puede ser resuelto con el método tradicional de Resistencia de materiales, sin embargo resolver el problema con el método sofisticado del MEF es de gran interés didáctico, pues en este particular problema cada nodo puede ser trabajado con dos variables y pueden servir como conceptos primarios para estudio de placas y láminas.

Entre estos principios y conceptos básicos existen dos formas planteadas como estudio generalizado de las mismas, como son el estudio de vigas por el método Viga Bernoulli y el estudio de vigas por el método Viga Timoshenko.

2.3.2 Deformación en vigas

El análisis estructural de las vigas suele dividirse en vigas isostáticas e hiperestáticas.

Recordemos que esta división corresponde a las condiciones de apoyo que presente el elemento a analizar. Si la viga tiene un número igual o inferior a tres incógnitas en sus reacciones, bastará con aplicar las condiciones de equilibrio estático para resolverla.

$$\sum Fx = 0 \qquad \sum Fy = 0 \qquad \sum M = 0$$

Si en cambio, la viga presenta un mayor número de incógnitas, no bastará con las ecuaciones antes indicadas, sino que será necesario incorporar nuevas expresiones.

Para abordar el análisis de las vigas hiperestáticas o estáticamente indeterminadas resulta necesario analizar las deformaciones que experimentará la viga, luego de ser cargada. Las distintas cargas sobre la viga generan tensiones de corte y flexión en la barra y, a su vez, la hacen deformarse.

El análisis de las deformaciones tiene básicamente dos objetivos. Por una parte, el poder obtener nuevas condiciones, que traducidas en ecuaciones, nos permitan resolver las incógnitas en vigas hiperestáticas. Y por otra parte, las deformaciones en sí, deben ser limitadas. Los envigados de madera o acero, por ejemplo, pueden quedar correctamente diseñados por resistencia, vale decir, no se romperán bajo la carga, pero podrán deformarse más allá de lo deseable, lo que llevaría consigo el colapso de elementos de terminación como cielos falsos o ventanales. No resulta extraño entonces que muchos dimensionamientos queden determinados por la deformación y no por la resistencia.

2.3.3 Línea elástica

Denominaremos línea elástica a la curva que forma la fibra neutra una vez cargada la viga, considerando que ésta se encontraba inicialmente recta.

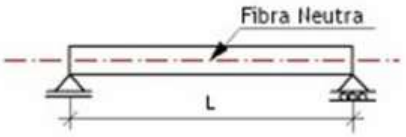


Figura 2.1 Fibra neutra

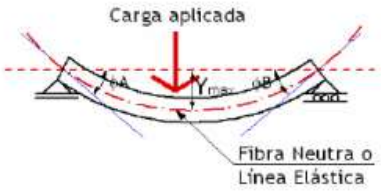


Figura 2.2 Fibra neutra con una carga aplicada

2.3.4 Supuesto base

Para establecer una serie de relaciones al interior de la sección, indicamos que se trata de una viga, cuyo material se encuentra solicitado dentro del rango de proporcionalidad entre tensiones y deformaciones, en donde se admite la conservación de las caras planas. Dicho en otra forma, donde se cumplen la ley de Hooke y la hipótesis de Bernoulli-Navier.

- **Ley de Hooke**

Establece que la relación entre la tensión y la deformación unitaria es una constante y se denomina módulo de elasticidad.

$$E = \frac{\tau}{e} \quad \rightarrow \quad \tau = E * e \tag{2.1}$$

Siendo:

E: Elasticidad

e: La deformación unitaria

τ: Tensión

2.3.5 Métodos de cálculo

Existen diferentes métodos para abordar el análisis de las deformaciones en las vigas:

- Método de Área de Momentos.
- Método de Doble Integración.
- Método de la Viga Conjugada.

Si bien, todos presentan su mecánica propia, a la vez tienen una partida común, que es justamente el análisis de la elástica, expuesto anteriormente.

En este caso, para la resolución de los problemas planteados, profundizaremos en el método de la doble integración.

- **Método de la doble integración**

Partiendo del análisis anterior, en el cual deducíamos la ecuación de la elástica, y con la misma premisa de que trabajaremos con ángulos bastante pequeños:

$$d\emptyset = \frac{M * dx}{EI} \qquad (2.2)$$

$$tan\,\emptyset = \emptyset$$

La derivada en cualquier punto de la una curva es igual a la pendiente de la tangente a la curva en ese punto.

$$tan\,\emptyset = \emptyset = \frac{dy}{dx} \qquad (2.3)$$

Reemplazando en la ecuación inicial obtenemos la Ecuación Diferencial de la Elástica de una viga:

$$\frac{d^2y}{dx} = \frac{M}{EI} \qquad (2.4)$$

Ecuación que al integrarla obtenemos la Ecuación general de la pendiente:

$$\frac{dy}{dx} = \frac{1}{EI}\int M\,dx + C_1 \qquad (2.5)$$

Si la ecuación anterior, la integramos de nuevo, hallamos la Ecuación general de la flecha:

$$y = \frac{1}{EI}\iint M\,dx + C_1 + C_2 \qquad (2.6)$$

Este método nos permite calcular las pendientes y deflexiones de la viga en cualquier punto. La dificultad radica en despejar las constantes de integración. Esto se logra analizando las condiciones de apoyo y la deformación de la viga.

2.4 Método de la rigidez

2.4.1 Método de la rigidez

Los métodos clásicos de análisis estructural desarrollados a fines del siglo XIX tienen las cualidades de la generalidad, simplicidad lógica y elegancia matemática.

Desgraciadamente, conducían a menudo a cálculos muy laboriosos cuando se los aplicaba en casos prácticos y, en aquella época, esto era un gran defecto.

Por esta razón sucesivas generaciones de ingenieros se dedicaron a tratar de reducir el conjunto de cálculos. Muchas técnicas ingeniosas de gran valor práctico fueron apareciendo (Método de Cross), pero la mayoría de las mismas eran aplicables sólo a determinados tipos de estructuras.

La principal objeción a los primeros métodos de análisis fue que los mismos conducían a sistemas con un gran número de ecuaciones lineales, difíciles de resolver manualmente.

Con los computadores, capaces de realizar el trabajo numérico, esta objeción no tiene ahora sentido, mientras que la generalidad de los métodos permanece. Esto explica por qué los métodos matriciales deben más, en su tratamiento básico de las estructuras, al siglo XIX que al XX.

El empleo de la notación matricial presenta dos ventajas en el cálculo de estructuras.

Desde el punto de vista teórico, permite utilizar métodos de cálculo en forma compacta, precisa y, al mismo tiempo, completamente general. Esto facilita el tratamiento de la teoría de estructuras como unidad, sin que los principios fundamentales se vean oscurecidos por operaciones de cálculo, por un lado, o diferencias físicas entre estructuras, por otro.

Desde el punto de vista práctico, proporciona un sistema apropiado de análisis de estructuras y determina una base muy conveniente para el desarrollo de programas de computación.

En contraste con estas ventajas, debe admitirse que los métodos matriciales se caracterizan por una gran cantidad de cálculo sistemático.

Las virtudes del cálculo con computadora radican en la eliminación de la preocupación por las operaciones rutinarias, el ingenio necesario para preparar el modelo con el que se pretende representar la realidad y el análisis crítico de los resultados.

Se debe ser consciente que sin un modelo adecuado o sin una interpretación final, el refinamiento en el análisis carece de sentido.

2.4.2 Método de la rigidez directa

Este método se basa en la hipótesis de que:

Partimos de una estructura lineal, en la que todos los movimientos y esfuerzos son funciones lineales de las cargas.

Las barras son rectas y de sección constante.

Como en cualquier problema estático, se deben cumplir las siguientes ecuaciones:

Ecuaciones de compatibilidad.

Ecuaciones constitutivas.

Ecuaciones de equilibrio.

Las Ecuaciones de compatibilidad relacionan las deformaciones de barras con los desplazamientos nodales. Si se introducen estas relaciones en las Ecuaciones constitutivas, se relacionan las fuerzas en los extremos de barras con los desplazamientos nodales.

Introduciéndose estas últimas relaciones en las Ecuaciones de equilibrio se obtiene un conjunto de ecuaciones de fuerzas nodales en función de desplazamientos nodales, que pueden ser consideradas como Ecuaciones de Equilibrio de la estructura en función de desplazamientos.

La resolución de este sistema de ecuaciones permite obtener el valor de las incógnitas (desplazamientos nodales), a partir de los cuales se obtienen las solicitaciones de las barras de la estructura, así como las reacciones.

Cuando se van a calcular las relaciones esfuerzos de extremos de barra-desplazamientos, es neutral escoger un sistema de coordenadas que haga estas ecuaciones lo más sencillas posible.

Se tomará por lo tanto como eje ‘x’ el que coincide con el eje geométrico de la pieza y los ejes ‘y’ y ‘z’ coincidentes con los ejes principales de la sección transversal.

Tal sistema pertenece a la barra, no depende de la orientación de la misma en la estructura, y se denominará sistema de ejes locales.

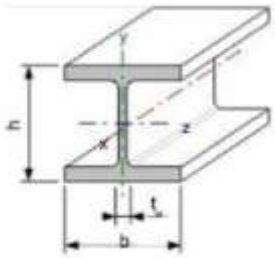


Figura 2.3 Representación de ejes locales

Por el contrario, cuando las piezas se unen entre sí para formar la estructura, es necesario tener un sistema de coordenadas común para todos los movimientos y esfuerzos de extremo de barras para poder aplicar las condiciones de equilibrio y compatibilidad.

A dicho sistema se lo denominará como sistema de ejes globales.

Por otro lado, dependiendo de la estructura que estemos tratando, tendremos unos desplazamientos u otros:

- Estructura articulada: Dos desplazamientos por nudo, el vertical y el horizontal.
- Estructura reticulada: Tres desplazamientos, los dos citados anteriormente más el giro que se produce en el eje ortogonal a los anteriores.

2.4.3 Procedimiento en el análisis matricial de estructuras

A la hora de realizar el análisis de una estructura por el método matricial, podemos distinguir las siguientes partes:

- Identificación estructural.
- Cálculo de la matriz de rigidez de barra.
- Cálculo de cargas nodales.
- Rotación de ejes en el plano.
- Cálculo de la matriz de rigidez global de la estructura.
- Cálculo de la matriz de cargas globales.
- Establecer las condiciones de contorno y su consiguiente cálculo de reacciones.

Explicaremos detalladamente cada una de las mismas.

2.4.4 Identificación estructural

En esta primera parte se definirá la estructura a base de datos y números.

- En primer lugar se definen unos ejes globales para toda la estructura.
- Conectividad de los elementos, se identifica para cada barra el nodo inicial y final. La misma queda definida automáticamente por el orden establecido para la numeración de los nodos de la barra.
- El eje ‘x’ local, coincide con el eje geométrico de la barra, siendo el sentido positivo el que va del nodo de menor numeración al de mayor numeración. Los otros ejes formarán un triedro directo.

2.4.5 Matriz de rigidez

- **Estructura articulada**: Consideramos la barra de una estructura articulada con ejes locales x’ e y’ orientados de la manera que comentábamos anteriormente.

  Supondremos que tratamos con una barra recta de sección transversal constante que responde a la Ley de Hooke.

Figura 2.4 Ejes locales y globales de una barra

En la barra i de la figura 3, el nodo inicial es el ‘j’ y el final es el ‘k’, quedando definida la orientación de la misma por los ejes locales ‘x’ e ‘y’.

Se considera que no existen deformaciones iniciales y que la deformación es elástica.

En este caso el alargamiento de la barra ’i’ estará dado por:

$$\Delta L_i = X^L_{X_k} - X^L_{X_j} \qquad (2.7)$$

Donde $X^L_{X_k}$ y $X^L_{X_j}$ son los desplazamientos producidos en la barra en dirección del eje x’.

En el caso de estructuras articuladas, la única solicitación que podremos encontrar será el esfuerzo axil, por lo que, siendo el nudo ‘j’ el inicial y ‘k’ el final tendremos:

$$F_{X_k} = \frac{E*A}{L}\Delta L_i = \frac{E*A}{L}(X^L_{X_k} - X^L_{X_j})$$

$$F_{X_j} = \frac{E*A}{L}\Delta L_i = -\frac{E*A}{L}(X^L_{X_k} - X^L_{X_j})$$

Siendo:

E, El módulo de elasticidad, A el área transversal y L la longitud de la barra.

Sabiendo que en el eje y’ local de cada barra no podemos encontrar ningún esfuerzo, podremos expresar de forma matricial las ecuaciones anteriores:

$$\begin{bmatrix} F_{X_j} \\ F_{Y_j} \\ F_{X_k} \\ F_{Y_k} \end{bmatrix} = \begin{bmatrix} +\frac{E*A}{L} & 0 & -\frac{E*A}{L} & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{E*A}{L} & 0 & +\frac{E*A}{L} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} X^L_{X_j} \\ X^L_{Y_j} \\ X^L_{X_k} \\ X^L_{Y_k} \end{bmatrix} \qquad (2.8)$$

La expresión anterior se corresponde a la ecuación matricial de la barra 'i' en coordenadas locales y expresa las fuerzas de extremo de barra en función de los desplazamientos de los nodos.

A la matriz que relaciona y se la denomina matriz de rigidez de barra de la estructura articulada en coordenadas locales.

Expresado en forma compacta o simbólica:

$$FI^L = KI^L * XI^L \tag{2.9}$$

En esta ecuación vienen definidas las fuerzas extremo de ambos nodos, el inicial y el final de la barra para cualquier pareja de desplazamientos. Estas ecuaciones son simétricas, como se podía esperar a partir del teorema de reciprocidad. No es posible, sin embargo, resolverlas y obtener los desplazamientos (X) en términos de las fuerzas (F), puesto que la matriz K es singular. Esta información nos lleva a concluir que las barras pueden estar sometidas a cualquier esfuerzo arbitrario que no afectará a los movimientos del nodo inicial y final de la barra.

En el caso de que la estructura esté dispuesta de un muelle sometido a tracción.

$$F_{X_k} = K * \Delta L_i = K * (X^L_{X_k} - X^L_{X_j}) \tag{2.10}$$

$$F_{Yk} = K * \Delta L_i = K * (X^L_{X_k} - X^L_{X_j}) \tag{2.11}$$

Donde K es la constante de rigidez del muelle.

$$\begin{bmatrix} F_{X_j} \\ F_{Y_j} \\ F_{X_k} \\ F_{Y_k} \end{bmatrix} = \begin{bmatrix} K & 0 & K & 0 \\ 0 & 0 & 0 & 0 \\ K & 0 & K & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} X^L_{X_j} \\ X^L_{Y_j} \\ X^L_{X_k} \\ X^L_{Y_k} \end{bmatrix} \tag{2.12}$$

Mientras que si nos encontramos con un muelle que haga de efecto disipador tendremos para los nudos 'j' y 'k'.

$$F_{X_k} = C * \Delta L_i = C * (\dot{X}^L_{X_k} - \dot{X}^L_{X_j}) \tag{2.13}$$

$$F_{Y_k} = C * \Delta L_i = C * (\dot{X}^L_{X_k} - \dot{X}^L_{X_j}) \tag{2.14}$$

Siendo C la constante de amortiguamiento del amortiguador.

$$\begin{bmatrix} F_{X_j} \\ F_{Y_j} \\ F_{X_k} \\ F_{Y_k} \end{bmatrix} = \begin{bmatrix} C & 0 & C & 0 \\ 0 & 0 & 0 & 0 \\ C & 0 & C & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \dot{X}^L_{X_j} \\ \dot{X}^L_{Y_j} \\ \dot{X}^L_{X_k} \\ \dot{X}^L_{Y_k} \end{bmatrix} \qquad (2.15)$$

- **Estructura reticulada**: En este tipo de estructura nos encontramos con los movimientos traslacionales en el eje horizontal y vertical más el giro en el plano.

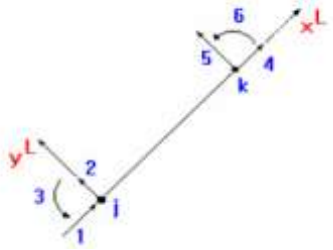


Figura 2.5 Barra articulada

Para la obtención de la matriz de rigidez, se aplican desplazamientos de valor unitario en uno de los movimientos disponibles y se restringen los demás.

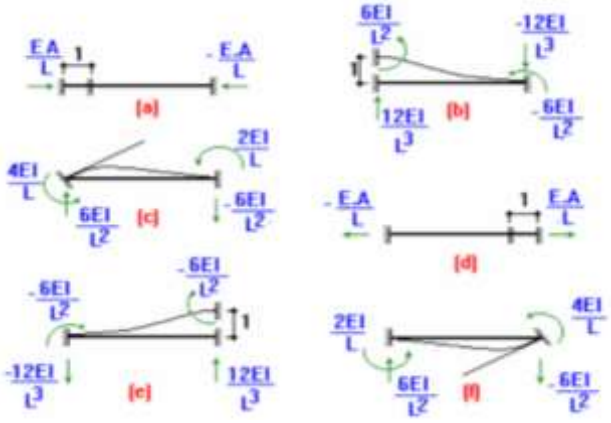


Figura 2.6 Reacciones unitarias

Las reacciones que se hallan con los desplazamientos unitarios serán los términos que formarán la matriz de rigidez de la estructura:

$$\begin{bmatrix} F_{Xj} \\ F_{Yj} \\ F_{Zj} \\ F_{Xk} \\ F_{Yk} \\ F_{Zk} \end{bmatrix} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \end{bmatrix} * \begin{bmatrix} X^L_{Xj} \\ X^L_{Yj} \\ X^L_{Zj} \\ X^L_{Xk} \\ X^L_{Yk} \\ X^L_{Zk} \end{bmatrix}$$

(2.16)

De la misma forma que ocurría en estructuras articuladas, expresada de forma compacta:

(2.17)

$$FI^L = KI^L * XI^L$$

2.4.6 Vector de cargas nodales

Hasta ahora se ha supuesto que las cargas estaban aplicadas en los nodos, y por lo tanto existe una correspondencia biunívoca entre los puntos de aplicación de las cargas y los desplazamientos que están siendo calculados. Si esto no ocurriera, por ejemplo se tuviesen cargas en el tramo de las barras, en forma distribuida o concentrada, se debe sustituir las cargas en las mismas por un sistema de cargas equivalentes aplicadas en los nodos que produzca en la estructura el mismo efecto que las cargas originales.

Aplicando el principio de superposición, que es válido por haber supuesto que el sistema es lineal, se puede descomponer las cargas tal como se indica en la figura:

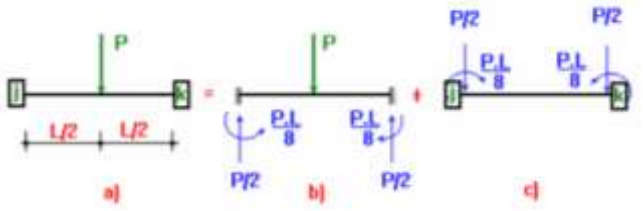


Figura 2.7 Cargas en los nodos

Como se puede observar las cargas, reacciones y deformaciones de la estructura a) serán equivalentes a la suma de los dos estados b) y c).

Como las deformaciones de nodos en b) son nulas, serán iguales las deformaciones de los casos c) y a). O sea que las cargas de c) producen la misma respuesta estructural en lo referente a desplazamientos de nudos que las cargas originales.

Estas serán entonces las cargas equivalentes en los nodos, que serán las reacciones que se producen en el empotramiento perfecto pero cambiadas de signo.

Los esfuerzos en los extremos de barra se obtienen por la suma de los casos (b) y (c).

$$F^a = F^b + F^c D^a = D^c \tag{2.18}$$

Por lo tanto, a la ecuación compacta habrá que añadirle las fuerzas del empotramiento perfecto, el caso (b).

$$FI^L = KI^L * XI^L + AI^L \tag{2.19}$$

Donde AIL representa el vector de fuerzas del empotramiento perfecto en ejes locales

2.4.7 Rotación de ejes en el plano

La rotacion de ejes en el plano se realiza para el caso de porticos y armaduras, se realizan utilizando las siguientes matrices de transformacion:

Armaduras

$$\begin{bmatrix} C & S & 0 & 0 \\ -S & C & 0 & 0 \\ 0 & 0 & C & S \\ 0 & 0 & -S & C \end{bmatrix}$$

donde:

C=Coseno director

S=Seno director

Porticos

$$\begin{bmatrix} Cos\theta & Sen\theta & 0 & 0 & 0 & 0 \\ -Sen\theta & -Cos\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & Cos\theta & Sen\theta & 0 \\ 0 & 0 & 0 & -Sen\theta & -Cos\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

2.4.8 Cálculo de la matriz de rigidez global de la estructura

Para el calculo de matriz de rigidez global de cada estructura se utiliza la siguiente matriz de rigidez para distintos casos

Armaduras

$$[K] = \frac{AE}{L} \begin{matrix} \\ C^2 \\ CS \\ -C^2 \\ -CS \end{matrix} \begin{matrix} \text{Uxi} \\ \begin{bmatrix} C^2 & CS & -C^2 & -CS \\ CS & S^2 & -CS & -S^2 \\ -C^2 & -CS & C^2 & CS \\ -CS & -S^2 & CS & S^2 \end{bmatrix} \end{matrix} \begin{matrix} \text{Uyi} \\ \text{Uxj} \\ \text{Uyj} \\ \\ \end{matrix} \begin{Bmatrix} Uxi \\ Uyi \\ Uxj \\ Uyj \end{Bmatrix}$$

donde:

C=Coseno director

S=Seno director

Portico

$$\left[\begin{array}{c|c|c|c|c|c} \frac{AE}{L}C^2 + \frac{12EI}{L^3}S^2 & \frac{AE}{L}SC - \frac{12EI}{L^3}SC & -\frac{6EI}{L^2}S & -\frac{AE}{L}C^2 - \frac{12EI}{L^3}S^2 & -\frac{AE}{L}SC + \frac{12EI}{L^3}SC & -\frac{6EI}{L^2}S \\ \frac{AE}{L}SC - \frac{12EI}{L^3}SC & \frac{AE}{L}S^2 + \frac{12EI}{L^3}C^2 & \frac{6EI}{L^2}C & -\frac{AE}{L}SC + \frac{12EI}{L^3}SC & -\frac{AE}{L}S^2 + \frac{12EI}{L^3}C^2 & \frac{6EI}{L^2}C \\ -\frac{6EI}{L^2}S & \frac{6EI}{L^2}C & \frac{4EI}{L} & \frac{6EI}{L^2}S & -\frac{6EI}{L^2}C & \frac{2EI}{L} \\ -\frac{AE}{L}C^2 - \frac{12EI}{L^3}S^2 & -\frac{AE}{L}SC + \frac{12EI}{L^3}SC & \frac{6EI}{L^2}S & \frac{AE}{L}C^2 + \frac{12EI}{L^3}S^2 & \frac{AE}{L}SC + \frac{12EI}{L^3}SC & \frac{6EI}{L^2}S \\ -\frac{AE}{L}SC - \frac{12EI}{L^3}SC & -\frac{AE}{L}S^2 + \frac{12EI}{L^3}C^2 & -\frac{6EI}{L^2}C & \frac{AE}{L}SC - \frac{12EI}{L^3}SC & \frac{AE}{L}S^2 + \frac{12EI}{L^3}C^2 & -\frac{6EI}{L^2}C \\ -\frac{6EI}{L^2}S & \frac{6EI}{L^2}C & \frac{2EI}{L} & \frac{6EI}{L^2}S & -\frac{6EI}{L^2}C & \frac{4EI}{L} \end{array}\right]$$

donde:

C=Coseno director

S=Seno director

2.4.9 Cálculo de la matriz de cargas globales

Cuando existen cargas en el vano, el vector de fuerzas de extremo fijo debe transformarse también.

{ F E } = [T] T {F E L }

{ F E } = Vector de fuerzas de empotramiento en coordenadas globales

{F E L } = Vector de fuerzas de empotramiento en coordenadas locales

[T] T =Matriz de transformacion

2.4.10 Establecer las condiciones de contorno y su consiguiente cálculo de reacciones.

Las condiciones de contorno se refieren al tipo de apoyo que presenta la estructura, resaltando que en una armadura los apoyos pueden ser solo fijos o moviles, para el caso de vigas o porticos los apoyos pueden ser fijos, moviles o empotrados, los grados de libertad de los contornos dependen del tipo de apoyo, a conntinuacion se vera una tabla donde se indica el tipo de apoyo y su restriccion respectiva, se utilizara el numero 1 para indicar si existe restriccion y el numero cero para indicar si el movimiento es libre:

Tipo de apoyo	Restriccion en X	Restriccion en Y	Restriccion al giro
Fijo	**1**	**1**	**0**
Movil	**0**	**1**	**0**
Empotrado	**1**	**1**	**1**

Una vez realicada la eliminacion de filas y columnas en funcion a las condiciones de contorno se procede al calculo de desplazamientos y consiguientemente al calculo de reacciones.

2.4.11 Fuerzas internas en un elemento

Primero se calculan los desplazamientos correspondientes a los grados de libertad libres, después se calculan las fuerzas en

coordenadas locales y se transforman a coordenadas globales con:

{ F i } = [T] { F } { F} = [T] T { F L }

Se recomienda usar coordenadas locales para calcular [K L] y después efectuar el triple producto

[T] T [K L] [T] = [K]

En este caso se calculan las fuerzas internas transformando primero los desplazamientos

{ u L } = [T] { u } { u L } = Desplazamientos en nudos locales

{ F L } = { F E L } + [K L] { u L } { u } = Desplazamientos en nudos globales

2.5 MatLab

2.5.1 Introducción

MATLAB es el nombre abreviado de “MATrix LABoratory”. MATLAB es un programa para realizar cálculos numéricos con vectores y matrices. Como caso particular puede también trabajar con números escalares -tanto reales como complejos- con cadenas de caracteres y con otras estructuras de información más complejas. Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones. MATLAB tiene también un lenguaje de programación propio.

2.5.2 Entorno gráfico de Matlab

Se trata de un entorno de trabajo muy gráfico e intuitivo, cuyas componentes más importantes son:

1. El escritorio de Matlab (Matlab Desktop), que es la ventana o contenedor de máximo nivel en la que se pueden situar los demás componentes.

2. Los componentes individuales, que podrán ser colocados o no, a gusto del usuario, en el escritorio de MatLab, orientados a tareas concretas dentro del programa, entre los que se puede citar:

a. Ventana de comandos (Command Window).

b. Ventana histórica de comandos (Command History).

c. Espacio de trabajo (Workspace).

d. Plataforma de lanzamiento (Launch Pad).

e. Directorio actual (Current Folder).

f. Ventana de ayuda (Help)

g. Editor de ficheros y depurador de errores (Editor).

h. Editor de vectores y matrices (Array Editor).

i. Ventana que permite estudiar cómo se emplea el tiempo de ejecución.

3 CAPITULO III PROGRAMACIÓN GENERAL DEL SOFTWARE COMPUTARIZADO M.E.F.

La aplicación con éxito en este caso del método de elementos finitos se reduce a un programa que fue elaborado con uso del lenguaje de programación MATLAB y con un sistema de funciones y scripts en Guide para la resolución de distintos tipos de problemas estructurales planos, estos conceptos se fueron desarrollando con teorías de elementos finitos de capítulos anteriores, el concepto que se elaboró para poder desarrollar el programa fue, en un sistema simple, gráfico y que tenga un uso netamente académico, ya que su fin es simplemente para uso docente-estudiante y para explicación en si del método paso a paso, ya que el lenguaje fue desarrollado de manera que pueda ser entendida para los estudiantes o docentes que tengan interés en el tema y puedan seguir investigando en el método. Se deja el código de

programacion abierto para posteriores estudiantes, docentes que tengan interés de investigación en este método y así poder ampliar su uso.

Toda la programación se realizó en el entorno grafico de MATLAB llamado Guide.

3.1 Características del software M.E.F.

En la figura (3.1) se muestra el diagrama de flujo del programa, en el cual se analizará para los, sistemas o bucles más importantes de ella la cual nos dará una idea total del sistema. Para centrar conceptos definiremos seguidamente las etapas fundamentales asociadas al análisis de una estructura por un programa de elementos finitos, así como la relación de cada etapa con las subrutinas puestas en el diagrama de flujo principal.

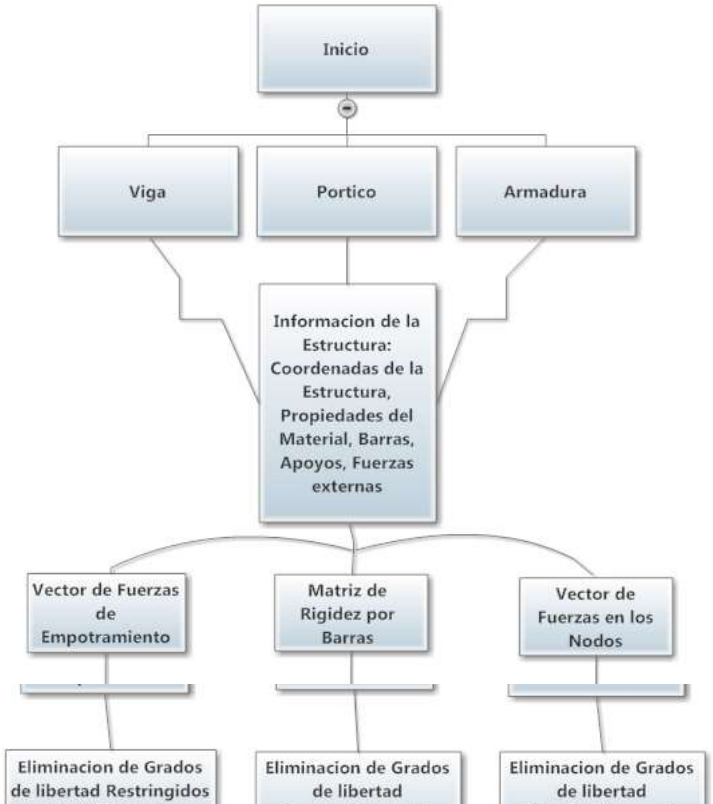


Figura 3.1 Diagrama de flujo principal del Programa M.E.F.

3.2 Inicio del programa computarizado M.E.F.

La primera ventana que se visualiza al iniciar el programa M.E.F. es la rutina de Inicio (Figura 3.2), en la cual se presenta el programa y se pone a elección el tipo de estructura a resolver, ya sea una Viga, un Pórtico o una Armadura, el código de programación de la ventana de Inicio se brinda a continuación.

Figura 3.2 Ventana de inicio

Código de programación de la ventana inicio:

```
function varargout = Inicio(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Inicio_OpeningFcn, ...
                   'gui_OutputFcn',  @Inicio_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function Inicio_OpeningFcn(hObject, eventdata, handles, varargin)
imshow(imread('UAJMS.png'));
set(handles.axes1,'xtick',[])
set(handles.axes1,'ytick',[])
handles.output = hObject;
guidata(hObject, handles);
function varargout = Inicio_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function viga_Callback(hObject, eventdata, handles)
VIGA; %
function portico_Callback(hObject, eventdata, handles)
PORTICO; %
function armadura_Callback(hObject, eventdata, handles)
```

```
ARM;
% --- Executes on button press in salir.
function salir_Callback(hObject, eventdata, handles)
close('Inicio')
```

3.3 Ingreso de datos

Una vez escogido el tipo de estructura se abrirán ventanas según la elección (Figura 3.3 Viga, Figura 3.4 Pórtico, Figura 3.5 Armadura), el ingreso de datos para los tres tipos de estructura es la misma en los casos de Coordenadas de la Estructura, Barras, Apoyos y Mostrar Resultados para poder simplificar el manejo del programa así como también poder reducir el código de programación para mayor efectividad y velocidad.

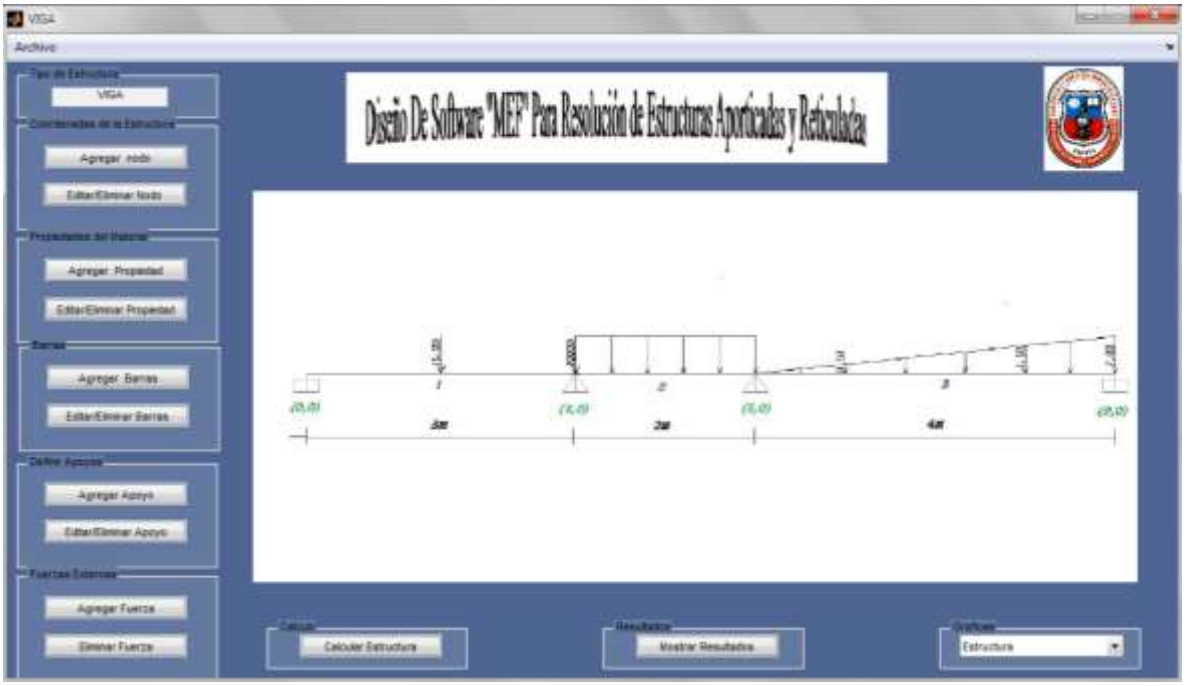


Figura 3.3 Ventana de viga

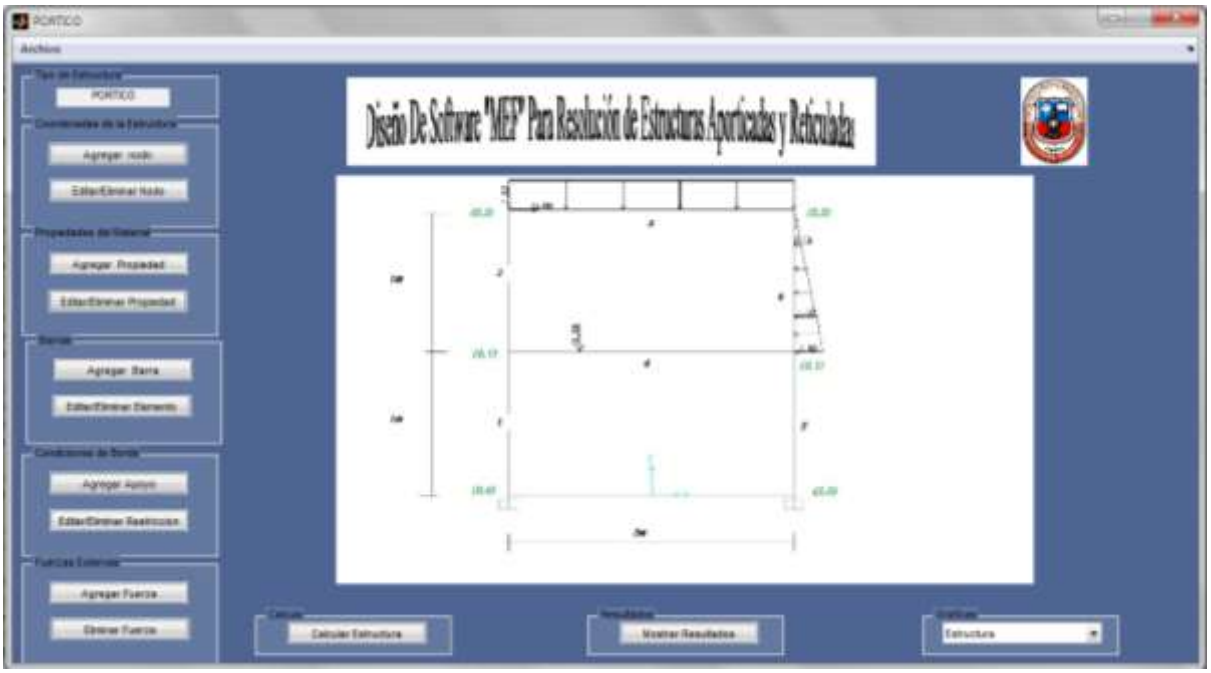


Figura 3.4 Ventana de pórtico.

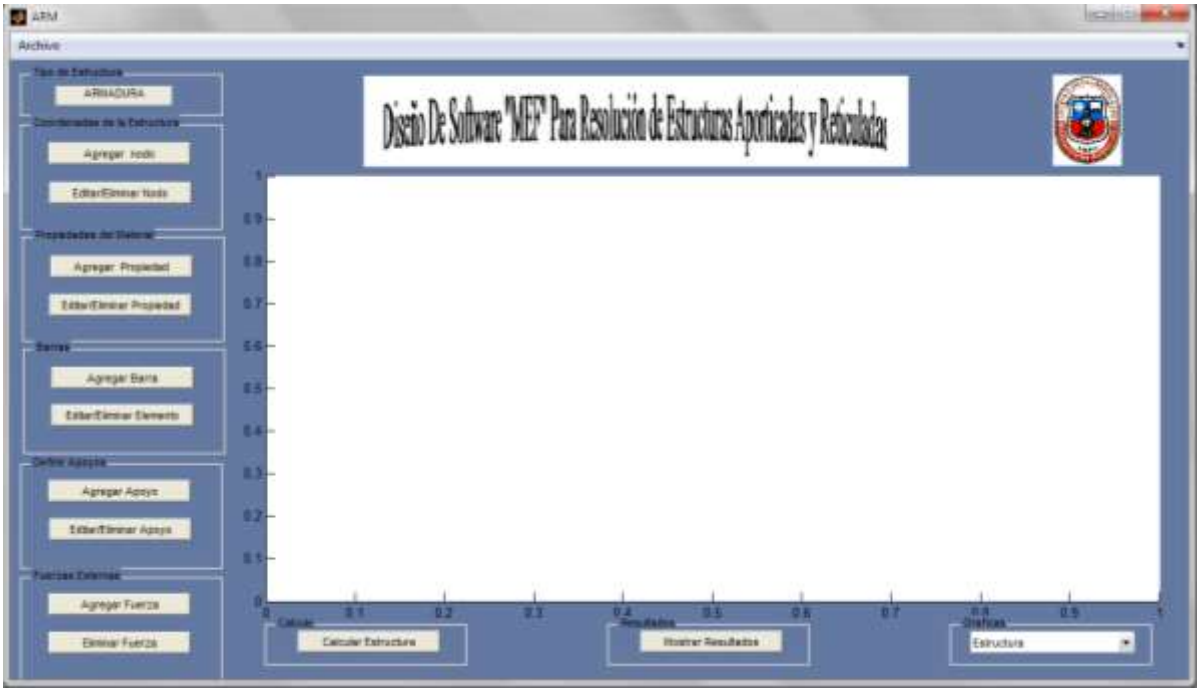


Figura 3.5 Ventana de armadura.

3.3.1 Coordenadas de la estructura

La ventana de ingreso de coordenadas se divide en 2 partes, la primera es la de Agregar Nodos y la segunda es la de Editar/Eliminar Nodos

- **Agregar nodos**

Esta es la primera ventana que se visualiza para a introducción de datos, en dicha ventana se debe ingresar las coordenadas de cada nodo de la estructura, comenzando por la numeración del nodo, seguido de su coordenada en X e Y.

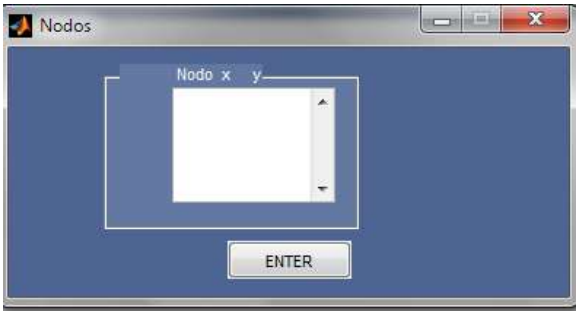


Figura 3.6 Ventana de Ingreso de Coordenadas

La rutina encargada del ingreso de nodos fue llamada Nodos.m, se muestra a continuación su programación.

Codigo de Programacion de la ventana Ingreso de Nodos:

```
function varargout = Nodos(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Nodos_OpeningFcn, ...
                   'gui_OutputFcn',  @Nodos_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Nodos_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
 a=dir('coordenada*.mat');
 for N=1:size(a,1)
    delete(['coordenada',num2str(N),'.mat']);
 end
handles.output = hObject;
guidata(hObject, handles);

function varargout = Nodos_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function datos_Callback(hObject, eventdata, handles)
function enter_Callback(hObject, eventdata, handles)
global M
M=str2num(get(handles.datos,'string'));
for N=1:size(M)
    P=M(N,:);
    Nombre=['coordenada',num2str(N),'.mat'];
    save(Nombre,'P')
end
close('Nodos')
```

Editar/Eliminar nodos

Esta rutina como su nombre lo indica se encarga de la edición y eliminación de nodos, al igual que la ventana coordenadas se debe ingresar primeramente la numeración del nodo a editar/eliminar, seguido de su coordenada en X e Y.

Figura 3.7 Ventana de Editar/Eliminar Nodo.

La rutina encargada de editar/eliminar nodos fue denominada ElimNodo.m, se muestra a continuación su programación:

Código de programación de la ventana ingreso de ElimNodo:

```
function varargout = elimNodo(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @elimNodo_OpeningFcn, ...
                   'gui_OutputFcn',  @elimNodo_OutputFcn, ...
```

```
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function elimNodo_OpeningFcn(hObject, eventdata, handles, varargin)
global M
set(handles.numero_nodo,'string',num2str(M))
handles.output = hObject;
guidata(hObject, handles);
function varargout = elimNodo_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function numero_nodo_Callback(hObject, eventdata, handles)
function enter_Callback(hObject, eventdata, handles)
global M
 a=dir('coordenada*.mat');
 for N=1:size(a,1)
    delete(['coordenada',num2str(N),'.mat']);
 end
M=str2num(get(handles.numero_nodo,'string'));
for N=1:size(M)
    P=M(N,:);
    Nombre=['coordenada',num2str(N),'.mat'];
    save(Nombre,'P')
end
close('elimNodo');
```

3.3.2. Propiedades del material

Esta rutina se encarga de recopilar la información de las propiedades del material como ser, Inercia y Modulo de Elasticidad en el caso de Vigas; Inercia, Modulo de Elasticidad y Área en el caso de Pórticos; Área y Modulo de Elasticidad en el caso de Armaduras, se divide en dos partes, Agregar Propiedad y Editar/ Eliminar Propiedad.

3.3.2.1. Agregar propiedad

La rutina de Agregar Propiedad se denominó Propiedades2.m para el caso de Armaduras, Propiedad1.m para el caso de Pórticos y Propiedades.m para el caso de Vigas.

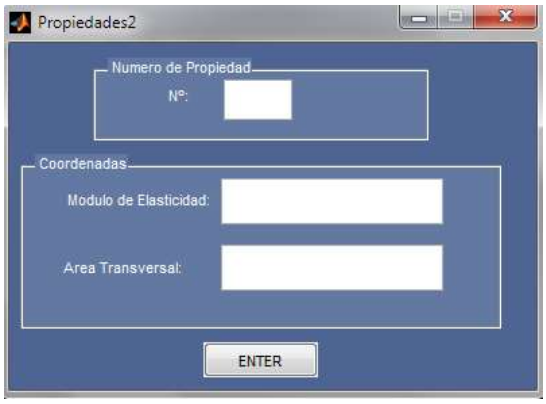


Figura 3.8 Agregar propiedad en armaduras.

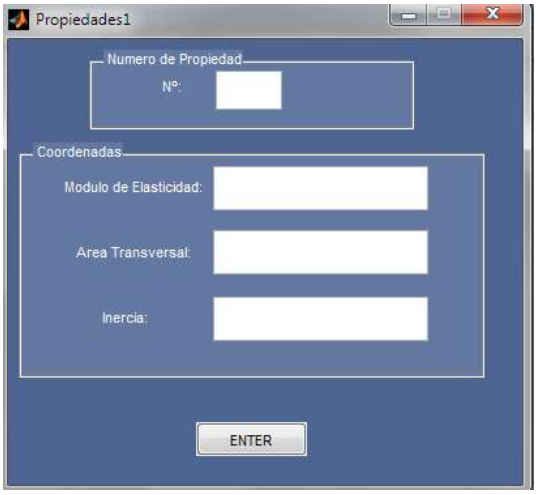


Figura 3.9 Agregar propiedad en pórticos.

Figura 3.10 Agregar propiedad en vigas.

Las rutinas encargadas de agregar propiedades del material contienen el siguiente código de programación.

Código de programación de la ventana agregar propiedades en vigas:

```
function varargout = Propiedades(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Propiedades_OpeningFcn, ...
                   'gui_OutputFcn',  @Propiedades_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function Propiedades_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = Propiedades_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function numero_propiedad_Callback(hObject, eventdata, handles)
handles.np=str2double(get(handles.numero_propiedad,'String'));
guidata(hObject, handles);
function modulo_de_elasticidad_Callback(hObject, eventdata, handles)
handles.me=str2double(get(handles.modulo_de_elasticidad,'String'));
guidata(hObject, handles);
function modulo_de_inercia_Callback(hObject, eventdata, handles)
handles.at=str2double(get(handles.modulo_de_inercia,'String'));
guidata(hObject, handles);
function enter_propiedad_Callback(hObject, eventdata, handles)
%global p
np=handles.np;
me=handles.me;
at=handles.at;
P=[np me at ];
Nombre=['propiedad',num2str(np),'.mat'];
save(Nombre,'P')
%p=p+1;
close('Propiedades')
function text7_CreateFcn(hObject, eventdata, handles)
function edit9_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Código de programación de la ventana agregar propiedades en pórticos:

```
function varargout = Propiedades1(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Propiedades1_OpeningFcn, ...
                   'gui_OutputFcn',  @Propiedades1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function Propiedades1_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = Propiedades1_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function numero_propiedad_Callback(hObject, eventdata, handles)
handles.np=str2double(get(handles.numero_propiedad,'String'));
guidata(hObject, handles);
function modulo_de_elasticidad_Callback(hObject, eventdata, handles)
handles.me=str2double(get(handles.modulo_de_elasticidad,'String'));
guidata(hObject, handles);
function Area_transversal_Callback(hObject, eventdata, handles)
handles.at=str2double(get(handles.Area_transversal,'String'));
guidata(hObject, handles);
function Modulo_de_inercia_Callback(hObject, eventdata, handles)
handles.in=str2double(get(handles.Modulo_de_inercia,'String'));
guidata(hObject, handles);
function enter_propiedad_Callback(hObject, eventdata, handles)
np=handles.np;
me=handles.me;
at=handles.at;
in=handles.in;
P=[np me at in];
Nombre=['propiedad',num2str(np),'.mat'];
save(Nombre,'P')
close('Propiedades1')
function Modulo_de_inercia_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Código de programación de la ventana agregar propiedades en armaduras:

```
function varargout = Propiedades2(varargin)
```

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Propiedades2_OpeningFcn, ...
                   'gui_OutputFcn',  @Propiedades2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function Propiedades2_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = Propiedades2_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function numero_propiedad_Callback(hObject, eventdata, handles)
handles.np=str2double(get(handles.numero_propiedad,'String'));
guidata(hObject, handles);
function modulo_de_elasticidad_Callback(hObject, eventdata, handles)
handles.me=str2double(get(handles.modulo_de_elasticidad,'String'));
guidata(hObject, handles);
function Area_transversal_Callback(hObject, eventdata, handles)
handles.at=str2double(get(handles.Area_transversal,'String'));
guidata(hObject, handles);
function enter_propiedad_Callback(hObject, eventdata, handles)
np=handles.np;
me=handles.me;
at=handles.at;
P=[np me at];
Nombre=['propiedad',num2str(np),'.mat'];
save(Nombre,'P')
close('Propiedades2')
```

3.3.3. Barras

El panel “Barras” se encarga de recopilar información de las conexiones de las barras, la columna uno enumera la barra, en la columna dos se coloca el número de propiedad y en la columna 3 y 4 el nodo inicial y final.

3.3.3.1. Agregar barra

La rutina de Barras en los tres casos, es decir, para vigas, pórticos y armaduras es la misma, el nombre con el q se la denomino es Elementos.m y tiene el siguiente código de programación y la ventana se presenta en la figura 3.11

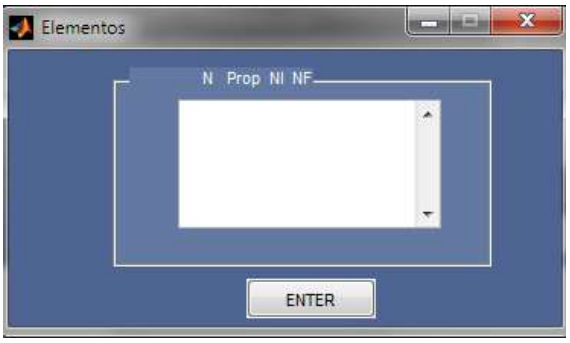


Figura 3.11 Ventana de ingreso de barras

Código de programación de la ventana agregar barras:

```
function varargout = Elementos(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Elementos_OpeningFcn, ...
                   'gui_OutputFcn',  @Elementos_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function Elementos_OpeningFcn(hObject, eventdata, handles, varargin)
 a=dir('elemento*.mat');
 for N=1:size(a,1)
    delete(['elemento',num2str(N),'.mat']);
 end
handles.output = hObject;
guidata(hObject, handles);
function varargout = Elementos_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function Numero_barras_Callback(hObject, eventdata, handles)
function enter_elementos_Callback(hObject, eventdata, handles)
global E
E=str2num(get(handles.Numero_barras,'string'));
for N=1:size(E)
    P=E(N,:);
    Nombre=['elemento',num2str(N),'.mat'];
    save(Nombre,'P')
end
close('Elementos')
```

3.3.3.2. Editar/Eliminar Barra

La rutina Editar/Eliminar Barra como su nombre lo dice se encarga de la edición o en su defecto la eliminación de barras, simplemente se debe colocar el número de barra que se desee eliminar en la figura 3.12 se presenta la ventana de Edición/Eliminación de Barras.

Figura 3.12 Ventana de Editar/Eliminar barra

Esta rutina de denomino con el nombre de ElimElemento.m y su código de programación es el siguiente:

```
function varargout = elimElemento(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @elimElemento_OpeningFcn, ...
                   'gui_OutputFcn',  @elimElemento_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function elimElemento_OpeningFcn(hObject, eventdata, handles, varargin)
global E
set(handles.eliminar_elemento,'string',num2str(E))
handles.output = hObject;
guidata(hObject, handles);
function varargout = elimElemento_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
```

```
function eliminar_elemento_Callback(hObject, eventdata, handles)
function enter_Callback(hObject, eventdata, handles)
global E
 a=dir('elemento*.mat');
 for N=1:size(a,1)
    delete(['elemento',num2str(N),'.mat']);
 end
E=str2num(get(handles.eliminar_elemento,'string'));
for N=1:size(E)
    P=E(N,:);
    Nombre=['elemento',num2str(N),'.mat'];
    save(Nombre,'P')
end
close('elimElemento');
```

3.3.4. Definir apoyos

La rutina de definir apoyos se encarga de recopilar la información correspondiente a los apoyos de la estructura, definiendo automáticamente los grados de libertad.

3.3.4.1. Agregar apoyo

Esta ventana se encarga de guardar los tipos de apoyos, cabe recalcar que para el caso de Armaduras al ser estructuras articuladas no presentan rotación entonces simplemente se utilizaran apoyos fijos y móviles.

Para el caso de armaduras la rutina se denomina BordesA.m a continuación se observa la figura 3.14 que representa la ventana de definir apoyos y posteriormente su código de programación.

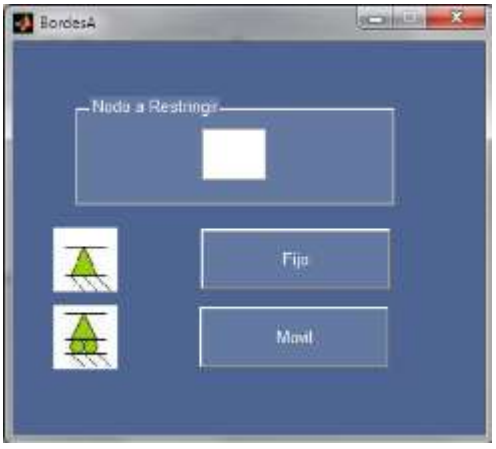


Figura 3.13 Ventana de apoyos

Código de programación de la ventana definir apoyos:

```
function varargout = BordesA(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @BordesA_OpeningFcn, ...
                   'gui_OutputFcn',  @BordesA_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function BordesA_OpeningFcn(hObject, eventdata, handles, varargin)
axes(handles.axes1)
imshow(imread('Fijo.jpg'));
set(handles.axes1,'xtick',[])
set(handles.axes1,'ytick',[])
handles.output = hObject;
axes(handles.axes2)
imshow(imread('Movil.jpg'));
set(handles.axes2,'xtick',[])
set(handles.axes2,'ytick',[])
handles.output = hObject;
guidata(hObject, handles);
function varargout = BordesA_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function Apoyo_fijo_Callback(hObject, eventdata, handles)
Nodo1=str2double(get(handles.N_nodo,'String'));
af=[0 0];
P=[Nodo1 af];
Nombre=['bordes',num2str(Nodo1),'.mat'];
save(Nombre,'P')
close('BordesA')
function apoyo_movil_Callback(hObject, eventdata, handles)
Nodo1=str2double(get(handles.N_nodo,'String'));
am=[ 1 0 ];
P=[Nodo1 am];
Nombre=['bordes',num2str(Nodo1),'.mat'];
save(Nombre,'P')
close('BordesA')
function N_nodo_Callback(hObject, eventdata, handles)
function N_nodo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function axes1_CreateFcn(hObject, eventdata, handles)
```

```
function axes2_CreateFcn(hObject, eventdata, handles)
```

3.3.4.2. Editar/Eliminar apoyos

La rutina de definir apoyos se encarga de editar y eliminar nodos que hayan sido introducidos por equivocación.

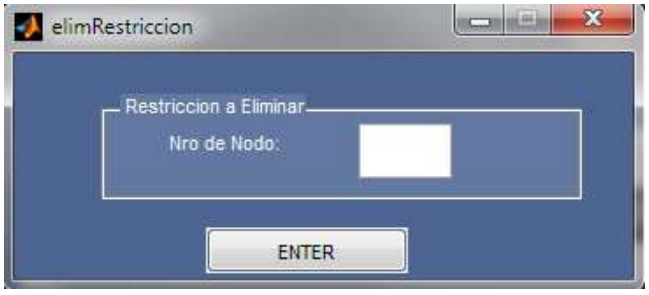


Figura 3.14 Ventana de apoyos a eliminar

Codigo de programación de la rutina elimRestriccion.m:

```
function varargout = elimRestriccion(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @elimRestriccion_OpeningFcn, ...
                   'gui_OutputFcn',  @elimRestriccion_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function elimRestriccion_OpeningFcn(hObject, eventdata, handles,
varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = elimRestriccion_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
function eliminar_Restriccion_Callback(hObject, eventdata, handles)
er=str2double(get(handles.eliminar_Restriccion,'String'));
delete(['bordes',num2str(er),'.mat']);
function enter_Callback(hObject, eventdata, handles)
close('elimRestriccion');
```

3.3.5. Fuerzas externas.

En la ventana de fuerzas externas (Figura 1.16) nos muestra la opción de insertar momentos de empotramiento y fuerzas ejercidas directamente en los nodos.

Figura 3.15 Ventana de fuerzas.

3.3.5.1. Momentos de empotramiento

La rutina Momentos de empotramiento en Vigas o Porticos muestra distintos estados de carga, que pudieran estar presentes en las barras. Simplemente se debe elegir el estado de carga que se tenga sobre la barra y ser abrirá una ventana donde se calcularan los momentos de manera sencilla e intuitiva. Figura 3.17

La rutina de esta ventana se denomina CasosEmpotramiento.m

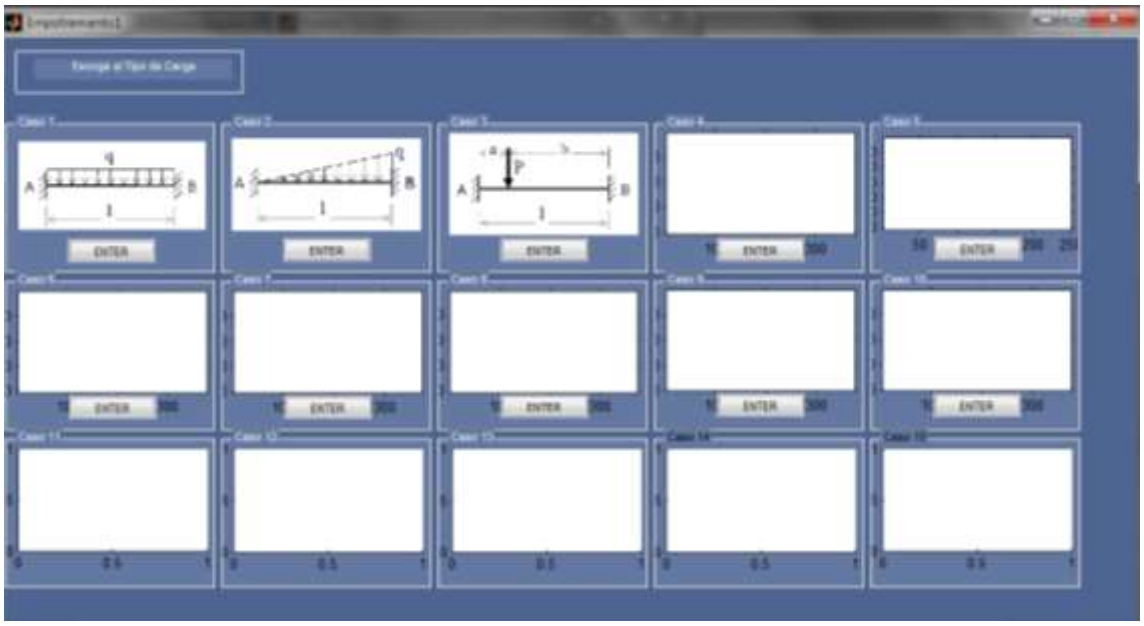


Figura 3.16 Ventana de casos de momentos de empotramiento.

Código de programación de la ventana CasosEmpotramiento:

```
function varargout = CasoEmpotramiento(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @CasoEmpotramiento_OpeningFcn, ...
                   'gui_OutputFcn',  @CasoEmpotramiento_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function CasoEmpotramiento_OpeningFcn(hObject, eventdata, handles,
varargin)
axes(handles.axes1); imshow(imread('M1.png'));
set(handles.axes1,'xtick',[]); set(handles.axes1,'ytick',[])
axes(handles.axes2); imshow(imread('M5.png'));
set(handles.axes2,'xtick',[]); set(handles.axes2,'ytick',[])
axes(handles.axes3); imshow(imread('M9.png'));
set(handles.axes3,'xtick',[]); set(handles.axes3,'ytick',[])
handles.output = hObject;
guidata(hObject, handles);
function varargout = CasoEmpotramiento_OutputFcn(hObject, eventdata,
handles)
```

```
varargout{1} = handles.output;
function enter1_Callback(hObject, eventdata, handles)
function enter2_Callback(hObject, eventdata, handles)
function enter3_Callback(hObject, eventdata, handles)
function enter4_Callback(hObject, eventdata, handles)
function enter5_Callback(hObject, eventdata, handles)
function enter6_Callback(hObject, eventdata, handles)
function enter7_Callback(hObject, eventdata, handles)
function enter8_Callback(hObject, eventdata, handles)
function enter9_Callback(hObject, eventdata, handles)
function enter10_Callback(hObject, eventdata, handles)
```

3.3.5.2. Fuerzas en los nodos

Esta rutina para vigas, pórticos y armaduras, se encarga de recopilar datos sobre fuerzas que estén actuando directamente en los nodos, dicha rutina lleva el nombre de Fexternas.m y en la misma se debe ingresar en la columna uno el número de nodo donde actuara la carga, en la columna 2, 3 y 4 se debe ingresar fuerzas y momentos (Solo fuerzas en el caso de Armaduras y colocar 0 en la columna de momento).

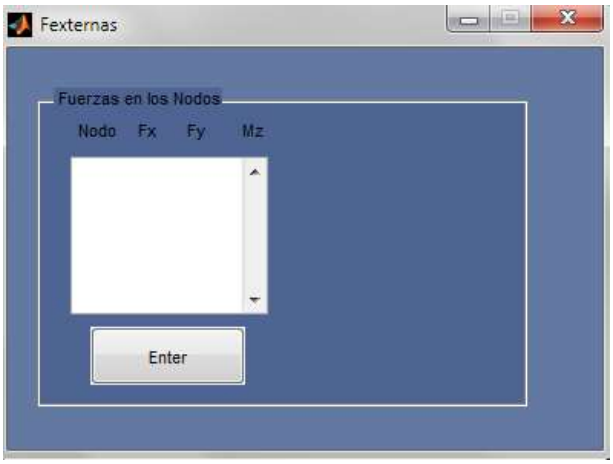


Figura 3.17 Ventana de Fuerzas en los Nodos.

3.3.6. Editar/Eliminar Fuerza

En esta ventana se debe introducir el caso de estado de carga en la primera ventana y en la segunda ventana colocar la barra a la que corresponde el caso de empotramiento.

La rutina se llama elimFuerza.m (Figura 3.19), tiene el siguiente código de programación:

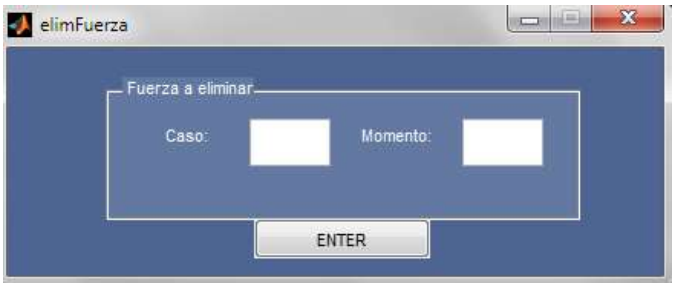


Figura 3.18 ventana de Editar/Eliminar fuerza

```
function varargout = elimFuerza(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @elimFuerza_OpeningFcn, ...
                   'gui_OutputFcn',  @elimFuerza_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function elimFuerza_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = elimFuerza_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function momento_Callback(hObject, eventdata, handles)
function caso_Callback(hObject, eventdata, handles)
function momento_CreateFcn(hObject, eventdata, handles)
function enter_eliminar_fuerza_Callback(hObject, eventdata, handles)
N=str2double(get(handles.momento,'String'));
c=str2double(get(handles.caso,'String'));
delete(['momentosMO',num2str(c),'_',num2str(N),'.mat']);
close('elimFuerza')
guidata(hObject, handles);
```

3.4. Calcular estructura

Esta rutina es una de las más importantes del programa puesto a que es en este apartado donde se realiza todo el cálculo de la estructura mediante el método de elementos finitos y mientras se realiza este cálculo, se va creando un archivo Excel donde se muestra el procedimiento de cálculo paso a paso, simplemente se debe presionar el botón “Calcular Estructura” y comienza el cálculo donde el progreso se puede visualizar en una barra(Figura 3.20) que tiene

el siguiente código de programación y va progresando a medida que se va realizando el cálculo.

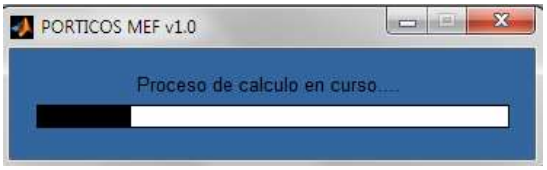


Figura 3.19 Barra de progreso de Calculo.

Codigo de programación de la barra de progreso:

```
clear all
wb=waitbar(0,'Proceso de calculo en curso....','color', [ 0.2 0.4 0.62
]);
set(wb,'Name','PORTICOS MEF v1.0');
wbObject=findobj(wb,'Type','Patch');
set(wbObject,'EdgeColor',[ 0 0 0],'FaceColor',[ 0 0 0 ]);
waitbar(10/100);
pause(1);
waitbar(20/100);
pause(1);
waitbar(30/100);
pause(1);
waitbar(40/100);
pause(1);
waitbar(50/100);
pause(1);
waitbar(60/100);
pause(1);
waitbar(70/100);
pause(1);
waitbar(80/100);
pause(1);
waitbar(90/100);
pause(1);
waitbar(100/100);
pause(1);
delete(wb)
```

La rutina recibe el nombre de function calcular_estructura_Callback (Figura3.21), a continuación se muestra el código de programación primeramente para el caso de vigas, luego para el caso de pórticos y finalmente para el caso de armaduras.

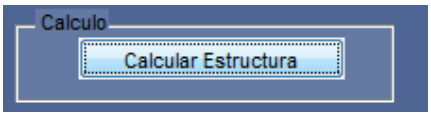


Figura 3.20 Botón para Calculo de estructuras.

Código de programación de la rutina function_calcular_estructura_Callback para el cálculo de vigas:

```
function calcular_estructura_Callback(hObject, eventdata, handles)
delete('calculos.xlsx');
wb=waitbar(0,'Proceso de calculo en curso....','color', [ 0.2 0.4 0.62
]);
set(wb,'Name','Color Change');
wbObject=findobj(wb,'Type','Patch');
set(wbObject,'EdgeColor',[ 0 0 0],'FaceColor',[ 0 0 0 ]);
ngl=0;
a=dir('coordenada*.mat');
for i=1:length(a)
 nombres=load(a(i).name);
 V(i,:)=nombres.P
end
V
x=V(:,2)
y=V(:,3)
NOD=size(V)
 nod=NOD(:,1)
 %Bordes
bordes=dir('bordes*.mat');
for i=1:length(bordes)
 nombres=load(bordes(i).name);
B(i,:)=nombres.P;
end
 B
 NNR=size(B)
 nnr=NNR(:,1)
 CG=ones(nod,2);
 for i=1:nnr
     m=B(i,1)
     CG(m,1)=B(i,3)
     CG(m,2)=B(i,4)
 end
 CG
 waitbar(10/100);
 pause(1);
 % grados de libertad
for i=1:nod
for j=1:2
if CG(i,j)~=0
ngl=ngl+1;
```

```
CG(i,j)=ngl;
else,end
end
end
CG
%Miembros
elem=dir('elemento*.mat');
for i=1:length(elem)
 nombres=load(elem(i).name);
 E(i,:)=nombres.P;
end
 E
 ini=E(:,3)
 fin=E(:,4)
 MBR=size(E)
 mbr=MBR(:,1)
 for i=1:mbr
for k=1:2
VC(i,k)= CG(ini(i), k);
VC(i,k+2) = CG(fin(i),k);
end
end
VC
waitbar(20/100);
 pause(1);
for i=1:mbr
Dx(i) = x(fin(i)) - x(ini(i));
Dy(i) = y(fin(i)) - y(ini(i));
L(i) = ((Dx(i))^2 + (Dy(i))^2)^0.5;
SENO(i) = Dy(i)/L(i);
COSENO(i) = Dx(i)/L(i);
end
L
SENO
COSENO
MEF= {'RESOLUCION POR EL PROGRAMA MEF'};
xlswrite('Calculos.xlsx', MEF, 2,'F1')
for i=1:mbr
MRIG1=['',char(65+i)];
MIRG2=num2str(2);
MRIG3=[ MRIG1,(MIRG2),];
xlswrite('Calculos.xlsx', {['Barra',num2str(i),]}, 2,[ '',MRIG3,]);
end
d = {'Longitud(m)'};
xlswrite('Calculos.xlsx', d, 2, 'A3')
xlswrite('Calculos.xlsx', L, 2, 'B3')
d = {'Seno'};
xlswrite('Calculos.xlsx', d, 2, 'A5')
xlswrite('Calculos.xlsx', SENO, 2, 'B5')
d = {'Coseno'};
xlswrite('Calculos.xlsx', d, 2, 'A7')
xlswrite('Calculos.xlsx', SENO, 2, 'B7')
%Propiedades
prop=dir('propiedad*.mat');
```

```
for i=1:length(prop)
 nombres=load(prop(i).name);
 P(i,:)=nombres.P;
end
 P
 PROP=zeros(mbr,2)
 h=E(:,2)
 for i=1:mbr
    PROP(i,1)=P(h(i),2)
    PROP(i,2)=P(h(i),3)
 end
PROP
Elas=PROP(:,1)
Inercia=PROP(:,2)
aux=zeros(ngl,ngl);
linea=11;
K_total=zeros(2*nod,2*nod);
VCNN=zeros(mbr,4);
waitbar(30/100);
 pause(1);
for i=1:mbr
VCNN(i,:)=[ 2*ini(i)-1 2*ini(i) 2*fin(i)-1 2*fin(i) ]
end
for i=1 : mbr
c=(4*Elas(i)*Inercia(i))/L(i)
a=(2*Elas(i)*Inercia(i))/L(i);
bb=(6*Elas(i)*Inercia(i))/(L(i)^2);
t=(12*Elas(i)*Inercia(i))/(L(i)^3);
%Matriz de rigidez de miembro K2 en coordenadas locales
K2(i,1,1)=t;K2(i,1,2)=bb;K2(i,1,3)=-t;K2(i,1,4)=bb;
K2(i,2,1)=bb;K2(i,2,2)=c;K2(i,2,3)=-bb;K2(i,2,4)=a;
K2(i,3,1)=-t;K2(i,3,2)=-bb;K2(i,3,3)=t;K2(i,3,4)=-bb;
K2(i,4,1)=bb;K2(i,4,2)=a;K2(i,4,3)=-bb;K2(i,4,4)=c;
%Matriz de paso T2-3
T2_3(i,1,1)=COSENO(i);T2_3(i,1,2)=SENO(i);T2_3(i,1,3)=0;T2_3(i,1,4)=0;
T2_3(i,2,1)=-SENO(i);T2_3(i,2,2)=COSENO(i);T2_3(i,2,3)=0;T2_3(i,2,4)=0;
T2_3(i,3,1)=0;T2_3(i,3,2)=0;T2_3(i,3,3)=COSENO(i);T2_3(i,3,4)=SENO(i);
T2_3(i,4,1)=0;T2_3(i,4,2)=0;T2_3(i,4,3)=-SENO(i);T2_3(i,4,4)=COSENO(i);
%Matriz de rigidez de miembro K3 en coordenadas globales
for m=1:4
for q=1:4
K2I(m,q)=K2(i,m,q);
end
end
for m=1:4
for q=1:4
T2_3I(m,q)=T2_3(i,m,q);
end
end
%fprintf ('\n Matriz de Rigidez en Coord. Globales:');
K3=T2_3I'*K2I*T2_3I
d = {'Matriz de Rigidez en Coord. Globales por miembro'};
xlswrite('Calculos.xlsx', d, 2, 'A9')
```

```
xlswrite('Calculos.xlsx', {['K',num2str(i),':']}, 2,
['A',num2str(linea)]);
 xlswrite('Calculos.xlsx', K3, 2,
['B',num2str(linea),':E',num2str(linea+3)]);
   linea=linea+size(K3,1)+2;
g_i=VCNN(i,:);
DeltaK_i=zeros(2*nod,2*nod);
DeltaK_i(g_i,g_i)=K3;
K_total=K_total+DeltaK_i;
%Ensamblaje
mataux=K3;
for j=1 :4;
if VC(i,j)==0;
mataux(j,:)=0;
mataux(:,j)=0;
else,end
end
k=zeros(ngl,ngl);
for cont1=1:4;
for cont2=1:4;
if mataux (cont1,cont2)~=0,
uno=VC(i,cont1);
dos=VC(i,cont2);
tres=mataux(cont1,cont2);
k(uno,dos)=k(uno,dos)+tres;
else,end
end
end
if i==1;
aux=k;
else if i~=1,
aux=aux+k;
else,end
end
end
%fprintf ('\n Matriz de Rigidez de la Estructura:');
Mat_rigidez=aux
waitbar(40/100);
 pause(1);
d = {'K total:'};
xlswrite('Calculos.xlsx', d, 2, ['A',num2str(13+6*mbr)])
xlswrite('Calculos.xlsx', {'Matriz de Rigidez Ensamblada de la Estructura
en Coordenadas Globales'}, 2, ['A',num2str(11+6*mbr)]);
MRIG1=[':',char(65+2*nod),];
MIRG2=num2str(13+2*nod+6*mbr);
MRIG3=[ MRIG1,(MIRG2),];
xlswrite('Calculos.xlsx', K_total, 2, ['B',num2str(14+6*mbr),'',MRIG3,]);
%Fuerzas
clear F Q1 VCJ
Q=zeros(1,ngl);
f=dir('fexternas*.mat');
for i=1:length(f)
 nombres=load(f(i).name);
 F(i,:)=nombres.P;
```

```
end
 F=F;
NJC=size(F);
njc=NJC(1);
for i=1 :njc
Q1(1)=F(i,3);
Q1(2)=F(i,4);
VCJ(i,:)=CG(F(i,1),:);
for m =1 :2
q=VCJ(i,m);
if q~=0
Q(q)=Q1(m);
else, end
end
end
Q_CJ=-Q'
waitbar(50/100);
pause(1);
% fprintf( '\n Vector de Cargas Totales:');
xlswrite('Calculos.xlsx', {'Matriz de Rigidez reducida'}, 2,
['A',num2str(15+2*nod+6*mbr)]);
xlswrite('Calculos.xlsx', {'K_reducida'}, 2,
['A',num2str(17+2*nod+6*mbr)]);
MRIG1=[':',char(65+ngl),];
MIRG2=num2str(17+2*nod+6*mbr+ngl);
MRIG3=[ MRIG1,(MIRG2),];
xlswrite('Calculos.xlsx',Mat_rigidez , 2,
['B',num2str(18+2*nod+6*mbr),'',MRIG3,]);
MIRG2=num2str(19+2*nod+6*mbr+ngl);
xlswrite('Calculos.xlsx', {'Vector de Cargas en los nodos(Ton)'}, 2,
['A',MIRG2,]);
MRIG1=num2str(20+2*nod+6*mbr+ngl);
Fext1Alt=zeros(2*nod,1);
xlswrite('Calculos.xlsx', Fext1Alt, 2, ['B',MRIG1,]);
MIRG2=num2str(21+4*nod+6*mbr+ngl);
xlswrite('Calculos.xlsx', {'Vector Reducido de Cargas en los nodos
(Ton)'}, 2, ['A',MIRG2,]);
MRIG1=num2str(22+4*nod+6*mbr+ngl);
Fext1Alt=zeros(2*nod,1);
xlswrite('Calculos.xlsx', Q_CJ, 2, ['B',MRIG1,]);
clear M
clear Q2_almac
%momentos de empotramiento
mom=dir('momento*.mat');
for i=1:length(mom)
 nombres=load(mom(i).name);
 M(i,:)=nombres.P;
end
 Q=zeros(1,ngl);
 Q2_almac1(mbr,6) = zeros;
 Q2_almac(mbr,4)=zeros;
 NMC=size(M);
nmc=NMC(1);
for ll=1:nmc
```

```
    MC=M(ll,1);
    Q2=[M(ll,3:4) M(ll,6:7)];
for mm =1:4
Q2_almac1(MC,mm)=Q2(mm)';
end
Q2_almac(:,1:2)=[Q2_almac1(:,1:2)]
Q2_almac(:,3:4)=[Q2_almac1(:,3:4)]
for m=1:4
for q=1:4
T2_3I(m,q)=T2_3(MC,m,q);
end
end
clear Q3
Q3 = (-1)*T2_3I'*Q2'
for g =1 :4
h = VC(MC,g);
if h ~=0
Q(h)= Q3(g)+Q(h);
else, end
end
end
Q_CM =Q'
MIRG2=num2str(23+4*nod+6*mbr+2*ngl);
xlswrite('Calculos.xlsx', {'Vector de Fuerzas de empotramiento(Ton)'}, 2,
['A',MIRG2,]);
MRIG1=num2str(24+4*nod+6*mbr+2*ngl);
Fext2Alt=zeros(2*nod,1);
xlswrite('Calculos.xlsx', Fext2Alt, 2, ['B',MRIG1,]);
fprintf( '\n Vector de Cargas Totales:');
MIRG2=num2str(25+6*nod+6*mbr+2*ngl);
xlswrite('Calculos.xlsx', {'Vector Reducido de Fuerzas de
empotramiento(Ton)'}, 2, ['A',MIRG2,]);
MRIG1=num2str(26+6*nod+6*mbr+2*ngl);
xlswrite('Calculos.xlsx', Q_CM, 2, ['B',MRIG1,]);
fprintf( '\n Vector de Cargas Totales:');
Q_TOTAL = Q_CJ + Q_CM
MIRG2=num2str(29+6*nod+6*mbr+2*ngl);
xlswrite('Calculos.xlsx', {'Vector de Fuerzas Totales(Ton)'}, 2,
['A',MIRG2,]);
MRIG1=num2str(30+6*nod+6*mbr+2*ngl);
Fext3Alt=zeros(2*nod,1);
xlswrite('Calculos.xlsx', Fext3Alt, 2, ['B',MRIG1,]);
MIRG2=num2str(31+8*nod+6*mbr+2*ngl);
xlswrite('Calculos.xlsx', {'Vector Reducido de Fuerzas Totales(Ton)'}, 2,
['A',MIRG2,]);
MRIG1=num2str(32+8*nod+6*mbr+2*ngl);
xlswrite('Calculos.xlsx', Q_TOTAL, 2, ['B',MRIG1,]);
fprintf( '\n Vector de Cargas Totales:');
waitbar(60/100);
 pause(1);
% Vector desplazamientos K*q = Q
fprintf( '\n Vector de Desplazamientos Generales:');
q_des = Mat_rigidez\Q_TOTAL
MRIG1=num2str(33+8*nod+6*mbr+3*ngl);
```

```
Des1=zeros(2*nod,1);
xlswrite('Calculos.xlsx', {'Vector de Desplazamientos Totales(x,y,z)'},
2, ['A',MRIG1,]);
MRIG1=num2str(34+8*nod+6*mbr+3*ngl);
xlswrite('Calculos.xlsx', Des1, 2, ['B',MRIG1,]);
MRIG1=num2str(35+10*nod+6*mbr+3*ngl);
xlswrite('Calculos.xlsx', {'Vector de Desplazamientos generales'}, 2,
['A',MRIG1,]);
MRIG1=num2str(36+10*nod+6*mbr+3*ngl);
xlswrite('Calculos.xlsx', q_des, 2, ['B',MRIG1,]);
MRIG1=num2str(37+10*nod+6*mbr+4*ngl);
Reac1=zeros(2*nod,1);
xlswrite('Calculos.xlsx', {'Reacciones de la Estructura'}, 2,
['A',MRIG1,]);
MRIG1=num2str(38+10*nod+6*mbr+4*ngl);
xlswrite('Calculos.xlsx', Reac1, 2, ['B',MRIG1,]);
D_total=zeros(1,2*nod)
waitbar(70/100);
 pause(1);
% Deformaciones y acciones finales de miembro
linea=39+12*nod+6*mbr+4*ngl;
linea1=45+12*nod+6*mbr+4*ngl;
linea2=51+12*nod+6*mbr+4*ngl;
for i=1:mbr
fprintf( '\n Miembro %d :',i);
fprintf( '\n Deformaciones de miembro:');
clear P1
clear P2
clear P
clear Q2_aux
for j=1:4
if VC (i,j)~=0
P1(j)=q_des(VC(i,j));
else
P1(j)=0;
end
end
P1=P1'
MRIG1=num2str(linea);
xlswrite('Calculos.xlsx',{['Vector de desplazamientos
barra:',num2str(i),':']}, 2, ['A',MRIG1,]);
MRIG1=num2str(linea+1);
xlswrite('Calculos.xlsx', P1, 2, ['B',MRIG1,]);
linea=linea+18;
for j=1:4
    D_total(VCNN(i,j))=P1(j);
end
for m=1:4
for q=1:4
K2I(m,q)=K2(i,m,q);
end
end
for m=1:4
for q=1:4
```

```
T2_3I(m,q)=T2_3(i,m,q);
end
end
%Matriz K3
K3=T2_3I'*K2I*T2_3I;
fprintf( '\n Acciones de miembro en coordenadas globales:');
P2=K3*P1
MRIG1=num2str(linea1);
xlswrite('Calculos.xlsx',{['Esfuerzos en coordenadas globales
barra:',num2str(i),':']}, 2, ['A',MRIG1,]);
MRIG1=num2str(linea1+1);
xlswrite('Calculos.xlsx', P2, 2, ['B',MRIG1,]);
linea1=linea1+18;
fprintf( '\n Acciones de miembro en coord. locales (P.
Complementario):');
P=T2_3I*P2
fprintf( '\n Acciones de miembro en coord. locales (P. Primario):');
for j=1:4
Q2_aux(j)=Q2_almac (i,j);
end
Q2_aux=Q2_aux'
fprintf( '\n Acciones finales (PP+PC) :');
P_FINAL=Q2_aux+P
Fg(i,:)=P_FINAL
F1(i,:)=P_FINAL
MRIG1=num2str(linea2);
xlswrite('Calculos.xlsx',{['Esfuerzos en coordenadas locales
barra:',num2str(i),':']}, 2, ['A',MRIG1,]);
MRIG1=num2str(linea2+1);
xlswrite('Calculos.xlsx', P_FINAL, 2, ['B',MRIG1,]);
linea2=linea2+18;
end
waitbar(80/100);
 pause(1);
D_total=D_total'
Reac=K_total*D_total
Fext=zeros(nod,2);
for i=1:njc
    Fext(F(i),:)=F(i,3:4);
end
Fext1=zeros(2*nod,1);
for i=1:nod
    Fext1(2*i-1)=Fext(i,1);
    Fext1(2*i)=Fext(i,2);
end
Fext1=-Fext1
M1=zeros(mbr,4);
for i=1:size(M,1)
    M1(M(i),1:2)=M(i,3:4)
    M1(M(i),3:4)=M(i,6:7)
end
M2=zeros(2*nod,1)
for i=1:mbr
    g_i=VCNN(i,:)
```

```
    M2_i=zeros(2*nod,1)
    M2_i(g_i)=M1(i,:)
    M2=M2+M2_i
end
Reac_finales=Reac-Fext1+M2
waitbar(90/100);
 pause(1);
 MRIG1=num2str(34+8*nod+6*mbr+3*ngl);
xlswrite('Calculos.xlsx', D_total, 2, ['B',MRIG1,]);
MRIG1=num2str(38+10*nod+6*mbr+4*ngl);
xlswrite('Calculos.xlsx', Reac_finales, 2, ['B',MRIG1,]);
MRIG1=num2str(20+2*nod+6*mbr+ngl);
xlswrite('Calculos.xlsx', Fext1, 2, ['B',MRIG1,]);
MRIG1=num2str(24+4*nod+6*mbr+2*ngl);
xlswrite('Calculos.xlsx', M2, 2, ['B',MRIG1,]);
F_total=Fext1-M2
MRIG1=num2str(30+6*nod+6*mbr+2*ngl);
xlswrite('Calculos.xlsx', F_total, 2, ['B',MRIG1,]);
 waitbar(100/100);
 pause(1);
 delete(wb);
save calcular_estructura
```

Código de programación de la rutina function_calcular_estructura_Callback para el cálculo de pórticos:

```
function calcular_estructura_Callback(hObject, eventdata, handles)
clear all
delete('calculos.xlsx');
wb=waitbar(0,'Proceso de calculo en curso....','color', [ 0.2 0.4 0.62
]);
set(wb,'Name','PORTICOS MEF v1.0');
wbObject=findobj(wb,'Type','Patch');
set(wbObject,'EdgeColor',[ 0 0 0],'FaceColor',[ 0 0 0 ]);
ngl=0;
a=dir('coordenada*.mat');
for i=1:length(a)
 nombres=load(a(i).name);
 V(i,:)=nombres.P
end
V
x=V(:,2)
y=V(:,3)
NOD=size(V)
 nod=NOD(:,1)
 %Bordes
bordes=dir('bordes*.mat');
for i=1:length(bordes)
 nombres=load(bordes(i).name);
B(i,:)=nombres.P;
end
 B
 NNR=size(B)
 nnr=NNR(:,1)
```

```
 CG=ones(nod,3);
 for i=1:nnr
     m=B(i,1)
     CG(m,1)=B(i,2)
     CG(m,2)=B(i,3)
     CG(m,3)=B(i,4)
 end
 CG
 % grados de libertad
for i=1:nod
for j=1:3
if CG(i,j)~=0
ngl=ngl+1;
CG(i,j)=ngl;
else,end
end
end
CG
waitbar(10/100);
 pause(1);
%Miembros
elem=dir('elemento*.mat');
for i=1:length(elem)
 nombres=load(elem(i).name);
 E(i,:)=nombres.P;
end
 E
 ini=E(:,3)
 fin=E(:,4)
 MBR=size(E)
 mbr=MBR(:,1)
 for i=1:mbr
for k=1:3
VC(i,k)= CG(ini(i), k);
VC(i,k+3) = CG(fin(i),k);
end
end
VC
for i=1:mbr
Dx(i) = x(fin(i)) - x(ini(i));
Dy(i) = y(fin(i)) - y(ini(i));
L(i) = ((Dx(i))^2 + (Dy(i))^2)^0.5;
SENO(i) = Dy(i)/L(i);
COSENO(i) = Dx(i)/L(i);
end
L
SENO
COSENO
waitbar(20/100);
 pause(1);
MEF= {'RESOLUCION POR EL PROGRAMA MEF'};
xlswrite('Calculos.xlsx', MEF, 2,'F1')
for i=1:mbr
MRIG1=['',char(65+i)];
```

```
MIRG2=num2str(2);
MRIG3=[ MRIG1,(MIRG2),];
xlswrite('Calculos.xlsx', {['Barra',num2str(i),]}, 2,[ '',MRIG3,]);
end
d = {'Longitud(m)'};
xlswrite('Calculos.xlsx', d, 2, 'A3')
xlswrite('Calculos.xlsx', L, 2, 'B3')
d = {'Seno'};
xlswrite('Calculos.xlsx', d, 2, 'A5')
xlswrite('Calculos.xlsx', SENO, 2, 'B5')
d = {'Coseno'};
xlswrite('Calculos.xlsx', d, 2, 'A7')
xlswrite('Calculos.xlsx', SENO, 2, 'B7')
%Propiedades
prop=dir('propiedad*.mat');
for i=1:length(prop)
 nombres=load(prop(i).name);
 P(i,:)=nombres.P;
end
 P
 PROP=zeros(mbr,3)
 h=E(:,2)
 for i=1:mbr
    PROP(i,1)=P(h(i),2)
    PROP(i,2)=P(h(i),3)
    PROP(i,3)=P(h(i),4)
 end
PROP
Elas=PROP(:,1)
Inercia=PROP(:,2)
Area=PROP(:,3)
aux=zeros(ngl,ngl);
linea=11;
K_total=zeros(3*nod,3*nod)
VCNN=zeros(mbr,6)
for i=1:mbr
VCNN(i,:)=[ 3*ini(i)-2 3*ini(i)-1 3*ini(i) 3*fin(i)-2 3*fin(i)-1
3*fin(i)]
end
waitbar(30/100);
 pause(1);
for i=1 : mbr
c=(4*Elas(i)*Inercia(i))/L(i);
a=(2*Elas(i)*Inercia(i))/L(i);
bb=(6*Elas(i)*Inercia(i))/(L(i)^2);
t=(12*Elas(i)*Inercia(i))/(L(i)^3);
r=((Elas(i)*Area(i))/L(i));
%Matriz de rigidez de miembro K2 en coordenadas locales
K2(i,1,1)=r;K2(i,1,2)=0;K2(i,1,3)=0;K2(i,1,4)=-r;K2(i,1,5)=0;K2(i,1,6)=0;
K2(i,2,1)=0;K2(i,2,2)=t;K2(i,2,3)=bb;K2(i,2,4)=0;K2(i,2,5)=-
t;K2(i,2,6)=bb;
K2(i,3,1)=0;K2(i,3,2)=bb;K2(i,3,3)=c;K2(i,3,4)=0;K2(i,3,5)=-
bb;K2(i,3,6)=a;
K2(i,4,1)=-r;K2(i,4,2)=0;K2(i,4,3)=0;K2(i,4,4)=r;K2(i,4,5)=0;K2(i,4,6)=0;
```

```
K2(i,5,1)=0;K2(i,5,2)=-t;K2(i,5,3)=-
bb;K2(i,5,4)=0;K2(i,5,5)=t;K2(i,5,6)=-bb;
K2(i,6,1)=0;K2(i,6,2)=bb;K2(i,6,3)=a;K2(i,6,4)=0;K2(i,6,5)=-
bb;K2(i,6,6)=c;
K2
%Matriz de paso T2-3
T2_3(i,1,1)=COSENO(i);T2_3(i,1,2)=SENO(i);T2_3(i,1,3)=0;T2_3(i,1,4)=0;T2_
3(i,1,5)=0;
T2_3(i,1,6)=0;
T2_3(i,2,1)=-
SENO(i);T2_3(i,2,2)=COSENO(i);T2_3(i,2,3)=0;T2_3(i,2,4)=0;T2_3(i,2,5)=0;
T2_3(i,2,6)=0;
T2_3(i,3,1)=0;T2_3(i,3,2)=0;T2_3(i,3,3)=1;T2_3(i,3,4)=0;T2_3(i,3,5)=0;T2_
3(i,3,6)=0;
T2_3(i,4,1)=0;T2_3(i,4,2)=0;T2_3(i,4,3)=0;T2_3(i,4,4)=COSENO(i);T2_3(i,4,
5)=SENO(i);
T2_3(i,4,6)=0;
T2_3(i,5,1)=0;T2_3(i,5,2)=0;T2_3(i,5,3)=0;T2_3(i,5,4)=-
SENO(i);T2_3(i,5,5)=COSENO(i);
T2_3(i,5,6)=0;
T2_3(i,6,1)=0;T2_3(i,6,2)=0;T2_3(i,6,3)=0;T2_3(i,6,4)=0;T2_3(i,6,5)=0;T2_
3(i,6,6)=1;
%Matriz de rigidez de miembro K3 en coordenadas globales
for m=1:6
for q=1:6
K2I(m,q)=K2(i,m,q);
end
end
for m=1:6
for q=1:6
T2_3I(m,q)=T2_3(i,m,q);
end
end
fprintf ('\n Matriz de Rigidez en Coord. Globales:');
K3=T2_3I'*K2I*T2_3I
d = {'Matriz de Rigidez en Coord. Globales por miembro'};
xlswrite('Calculos.xlsx', d, 2, 'A9')
xlswrite('Calculos.xlsx', {['K',num2str(i),':']}, 2,
['A',num2str(linea)]);
 xlswrite('Calculos.xlsx', K3, 2,
['B',num2str(linea),':G',num2str(linea+5)]);
   linea=linea+size(K3,1)+2;
%Ensamblaje
g_i=VCNN(i,:)
DeltaK_i=zeros(3*nod,3*nod)
DeltaK_i(g_i,g_i)=K3
K_total=K_total+DeltaK_i
mataux=K3;
mataux=K3;
for j=1 :6;
if VC(i,j)==0;
mataux(j,:)=0;
mataux(:,j)=0;
else,end
```

```
end
k=zeros(ngl,ngl);
for cont1=1:6;
for cont2=1:6;
if mataux (cont1,cont2)~=0,
uno=VC(i,cont1);
dos=VC(i,cont2);
tres=mataux(cont1,cont2);
k(uno,dos)=k(uno,dos)+tres;
else,end
end
end
if i==1;
aux=k;
else if i~=1,
aux=aux+k;
else,end
end
end
fprintf ('\n Matriz de Rigidez de la Estructura:');
Mat_rigidez=aux
waitbar(40/100);
 pause(1);
d = {'K total:'};
xlswrite('Calculos.xlsx', d, 2, ['A',num2str(13+8*mbr)])
xlswrite('Calculos.xlsx', {'Matriz de Rigidez Ensamblada de la Estructura
en Coordenadas Globales'}, 2, ['A',num2str(11+8*mbr)]);
MRIG1=[':',char(65+3*nod),];
MIRG2=num2str(13+8*mbr+3*nod);
MRIG3=[ MRIG1,(MIRG2),];
xlswrite('Calculos.xlsx', K_total, 2, ['B',num2str(14+8*mbr),'',MRIG3,]);
%Fuerzas
clear F Q1 VCJ
Q=zeros(1,ngl);
f=dir('fexternas*.mat');
for i=1:length(f)
 nombres=load(f(i).name);
 F(i,:)=nombres.P;
end
 F;
NJC=size(F);
njc=NJC(1);
for i=1 :njc
Q1(1)=F(i,2);
Q1(2)=F(i,3);
Q1(3)=F(i,4);
VCJ(i,:)=CG(F(i,1),:);
for m =1 :3
q=VCJ(i,m);
if Q1(m)~=0
Q(q)=Q1(m);
else, end
end
end
```

```
Q_CJ=Q'
waitbar(50/100);
pause(1);

xlswrite('Calculos.xlsx', {'Matriz de Rigidez reducida'}, 2,
['A',num2str(14+8*mbr+3*nod)]);
xlswrite('Calculos.xlsx', {'K_reducida'}, 2,
['A',num2str(16+8*mbr+3*nod)]);
MRIG1=[':',char(65+ngl),];
MIRG2=num2str(16+8*mbr+3*nod+ngl);
MRIG3=[ MRIG1,(MIRG2),];
xlswrite('Calculos.xlsx',Mat_rigidez , 2,
['B',num2str(17+8*mbr+3*nod),'',MRIG3,]);
MIRG2=num2str(18+8*mbr+3*nod+ngl);
xlswrite('Calculos.xlsx', {'Vector de Cargas en los nodos(Ton)'}, 2,
['A',MIRG2,]);
MRIG1=num2str(19+8*mbr+3*nod+ngl);
Fext1Alt=zeros(3*nod,1);
xlswrite('Calculos.xlsx', Fext1Alt, 2, ['B',MRIG1,]);
MIRG2=num2str(20+8*mbr+6*nod+ngl);
xlswrite('Calculos.xlsx', {'Vector Reducido de Cargas en los nodos
(Ton)'}, 2, ['A',MIRG2,]);
MRIG1=num2str(21+8*mbr+6*nod+ngl);
xlswrite('Calculos.xlsx', Q_CJ, 2, ['B',MRIG1,]);
clear M
clear Q2_almac
%momentos de empotramiento
mom=dir('momento*.mat');
for i=1:length(mom)
 nombres=load(mom(i).name);
 M(i,:)=nombres.P;
end
 Q=zeros(1,ngl);
 Q2_almac(mbr,6) = zeros;
 NMC=size(M);
nmc=NMC(1);
for ll=1:nmc
    MC=M(ll,1);
    Q2=M(ll,2:7);
    Q2=Q2'
for mm =1:6
Q2_almac(MC,mm)=Q2(mm)';
end
for m=1:6
for q=1:6
T2_3I(m,q)=T2_3(MC,m,q);
end
end
T2_3I;
clear Q3
Q3 = (-1)*T2_3I'*Q2
for g =1 :6
h = VC(MC,g);
if h ~=0
```

```
Q(h)= Q3(g)+Q(h);
else, end
end
end
Q_CM =Q'
MIRG2=num2str(21+8*mbr+6*nod+2*ngl);
xlswrite('Calculos.xlsx', {'Vector de Fuerzas de empotramiento(Ton)'}, 2,
['A',MIRG2,]);
MRIG1=num2str(23+8*mbr+6*nod+2*ngl);
Fext2Alt=zeros(3*nod,1);
xlswrite('Calculos.xlsx', Fext2Alt, 2, ['B',MRIG1,]);
fprintf( '\n Vector de Cargas Totales:');
MIRG2=num2str(24+8*mbr+9*nod+2*ngl);
xlswrite('Calculos.xlsx', {'Vector Reducido de Fuerzas de
empotramiento(Ton)'}, 2, ['A',MIRG2,]);
MRIG1=num2str(25+8*mbr+9*nod+2*ngl);
xlswrite('Calculos.xlsx', Q_CM, 2, ['B',MRIG1,]);
fprintf( '\n Vector de Cargas Totales:');
Q_TOTAL = Q_CJ + Q_CM
MIRG2=num2str(25+8*mbr+9*nod+3*ngl);
xlswrite('Calculos.xlsx', {'Vector de Fuerzas Totales(Ton)'}, 2,
['A',MIRG2,]);
MRIG1=num2str(26+8*mbr+9*nod+3*ngl);
Fext3Alt=zeros(3*nod,1);
xlswrite('Calculos.xlsx', Fext3Alt, 2, ['B',MRIG1,]);
MIRG2=num2str(27+8*mbr+12*nod+3*ngl);
xlswrite('Calculos.xlsx', {'Vector Reducido de Fuerzas Totales(Ton)'}, 2,
['A',MIRG2,]);
MRIG1=num2str(28+8*mbr+12*nod+3*ngl);
xlswrite('Calculos.xlsx', Q_TOTAL, 2, ['B',MRIG1,]);
fprintf( '\n Vector de Cargas Totales:');
waitbar(60/100);
 pause(1);
% Vector desplazamientos K*q = Q
fprintf( '\n Vector de Desplazamientos Generales:');
q_des = Mat_rigidez\Q_TOTAL
MRIG1=num2str(29+8*mbr+12*nod+4*ngl);
Des1=zeros(3*nod,1);
xlswrite('Calculos.xlsx', {'Vector de Desplazamientos Totales(x,y,z)'},
2, ['A',MRIG1,]);
MRIG1=num2str(30+8*mbr+12*nod+4*ngl);
xlswrite('Calculos.xlsx', Des1, 2, ['B',MRIG1,]);
MRIG1=num2str(31+8*mbr+15*nod+4*ngl);
xlswrite('Calculos.xlsx', {'Vector de Desplazamientos generales'}, 2,
['A',MRIG1,]);
MRIG1=num2str(32+8*mbr+15*nod+4*ngl);
xlswrite('Calculos.xlsx', q_des, 2, ['B',MRIG1,]);
MRIG1=num2str(33+8*mbr+15*nod+5*ngl);
Reac1=zeros(3*nod,1);
xlswrite('Calculos.xlsx', {'Reacciones de la Estructura'}, 2,
['A',MRIG1,]);
MRIG1=num2str(34+8*mbr+15*nod+5*ngl);
xlswrite('Calculos.xlsx', Reac1, 2, ['B',MRIG1,]);
D_total=zeros(1,3*nod)
```

```
waitbar(70/100);
 pause(1);
% Deformaciones y acciones finales de miembro
linea=34+8*mbr+18*nod+5*ngl;
linea1=43+8*mbr+18*nod+5*ngl;
linea2=52+8*mbr+18*nod+5*ngl;
for i=1:mbr
fprintf( '\n Miembro %d :',i);
fprintf( '\n Deformaciones de miembro:');
clear P1
clear P2
clear P
clear Q2_aux
for j=1:6
if VC (i,j)~=0
P1(j)=q_des(VC(i,j));
else
P1(j)=0;
end
end
P1=P1'
MRIG1=num2str(linea);
xlswrite('Calculos.xlsx',{['Vector de desplazamientos
barra:',num2str(i),':']}, 2, ['A',MRIG1,]);
MRIG1=num2str(linea+1);
xlswrite('Calculos.xlsx', P1, 2, ['B',MRIG1,]);
linea=linea+27;
for j=1:6
    D_total(VCNN(i,j))=P1(j);
end
for m=1:6
for q=1:6
K2I(m,q)=K2(i,m,q);
end
end
for m=1:6
for q=1:6
T2_3I(m,q)=T2_3(i,m,q);
end
end
%Matriz K3
K3=T2_3I'*K2I*T2_3I;
% Acciones de miembro en coordenadas globales
P2=K3*P1
MRIG1=num2str(linea1);
xlswrite('Calculos.xlsx',{['Esfuerzos en coordenadas globales
barra:',num2str(i),':']}, 2, ['A',MRIG1,]);
MRIG1=num2str(linea1+1);
xlswrite('Calculos.xlsx', P2, 2, ['B',MRIG1,]);
linea1=linea1+27;
% Acciones de miembro en coord. locales (P. Complementario)
P=T2_3I*P2
% Acciones de miembro en coord. locales (P. Primario)
for j=1:6
```

```
Q2_aux(j)=Q2_almac (i,j);
end
Q2_aux=Q2_aux'
% Acciones finales (PP+PC)
P_FINAL=Q2_aux+P
FNODO(i,:)=P_FINAL
Fg(i,:)=P_FINAL
F1(i,:)=P_FINAL
MRIG1=num2str(linea2);
xlswrite('Calculos.xlsx',{['Esfuerzos en coordenadas locales
barra:',num2str(i),':']}, 2, ['A',MRIG1,]);
MRIG1=num2str(linea2+1);
xlswrite('Calculos.xlsx', P_FINAL, 2, ['B',MRIG1,]);
linea2=linea2+27;
end
waitbar(80/100);
 pause(1);
D_total=D_total'
Reac=K_total*D_total
Fext=zeros(nod,3);
for i=1:njc
    Fext(F(i),:)=F(i,2:4);
end
Fext1=zeros(3*nod,1);
for i=1:nod
    Fext1(3*i-2)=Fext(i,1);
    Fext1(3*i-1)=Fext(i,2);
    Fext1(3*i)=Fext(i,3);
end
 M2=zeros(3*nod,1)
for i=1:nod
  for j=1:1
    if CG(i,j)==0
        M2(3*i-2)=0
    else
        M2(3*i-2)=Q_CM(CG(i,j))
    end
    if CG(i,j+1)==0
        M2(3*i-1)=0
    else
        M2(3*i-1)=Q_CM(CG(i,j+1))
    end
    if CG(i,j+1)==0
        M2(3*i)=0
    else
        M2(3*i)=Q_CM(CG(i,j+2))
    end
  end
end
Q_total=Fext1+M2;
Reac_finales=Reac-Q_total;
waitbar(90/100);
 pause(1);
 MRIG1=num2str(30+8*mbr+12*nod+4*ngl);
```

```
xlswrite('Calculos.xlsx', D_total, 2, ['B',MRIG1,]);
MRIG1=num2str(34+8*mbr+15*nod+5*ngl);
xlswrite('Calculos.xlsx', Reac_finales, 2, ['B',MRIG1,]);
MRIG1=num2str(19+8*mbr+3*nod+ngl);
xlswrite('Calculos.xlsx', Fext1, 2, ['B',MRIG1,]);
MRIG1=num2str(23+8*mbr+6*nod+2*ngl);
xlswrite('Calculos.xlsx', M2, 2, ['B',MRIG1,]);
F_total=Fext1+M2;
MRIG1=num2str(26+8*mbr+9*nod+3*ngl);
xlswrite('Calculos.xlsx', F_total, 2, ['B',MRIG1,]);
 waitbar(100/100);
 pause(1);
 delete(wb);
save calcular_estructura
```

Código de programación de la rutina function_calcular_estructura_Callback para el cálculo de armaduras:

```
clear all
wb=waitbar(0,'Proceso de calculo en curso....','color', [ 0.2 0.4 0.62
]);
set(wb,'Name','Color Change');
wbObject=findobj(wb,'Type','Patch');
set(wbObject,'EdgeColor',[ 0 0 0],'FaceColor',[ 0 0 0 ]);
ngl=0;
a=dir('coordenada*.mat');
for i=1:length(a)
 nombres=load(a(i).name);
 V(i,:)=nombres.P;
end
V;
x=V(:,2);
y=V(:,3);
NOD=size(V);
 nod=NOD(:,1);

 %Bordes
bordes=dir('bordes*.mat');
for i=1:length(bordes)
 nombres=load(bordes(i).name);
B(i,:)=nombres.P;
end
 B;
 NNR=size(B);
 nnr=NNR(:,1);
 CG=ones(nod,2);
 for i=1:nnr
     m=B(i,1);
     CG(m,1)=B(i,2);
     CG(m,2)=B(i,3);
 end
 CG;
 waitbar(10/100);
```

```
 pause(1);
 % grados de libertad
for i=1:nod
for j=1:2
if CG(i,j)~=0
ngl=ngl+1;
CG(i,j)=ngl;
else,end
end
end
CG;
%Miembros
elem=dir('elemento*.mat');
for i=1:length(elem)
 nombres=load(elem(i).name);
 E(i,:)=nombres.P;
end
 E;
 ini=E(:,3);
 fin=E(:,4);
MBR=size(E);
 mbr=MBR(:,1);
 for i=1:mbr
for k=1:2
VC(i,k)= CG(ini(i), k);
VC(i,k+2) = CG(fin(i),k);
end
end
VC;
waitbar(20/100);
 pause(1);
for i=1:mbr
Dx(i) = x(fin(i)) - x(ini(i));
Dy(i) = y(fin(i)) - y(ini(i));
L(i) = ((Dx(i))^2 + (Dy(i))^2)^0.5;
SENO(i) = Dy(i)/L(i);
COSENO(i) = Dx(i)/L(i);
end
L;
SENO;
COSENO;
MEF= {'RESOLUCION POR EL PROGRAMA MEF'};
xlswrite('Calculos.xlsx', MEF, 2,'F1')
for i=1:mbr
MRIG1=['',char(65+i)];
MIRG2=num2str(2);
MRIG3=[ MRIG1,(MIRG2),];
xlswrite('Calculos.xlsx', {['Barra',num2str(i),]}, 2,[ '',MRIG3,]);
end
d = {'Longitud(m)'};
xlswrite('Calculos.xlsx', d, 2, 'A3')
xlswrite('Calculos.xlsx', L, 2, 'B3')
d = {'Seno'};
xlswrite('Calculos.xlsx', d, 2, 'A5')
```

```
xlswrite('Calculos.xlsx', SENO, 2, 'B5')
d = {'Coseno'};
xlswrite('Calculos.xlsx', d, 2, 'A7')
xlswrite('Calculos.xlsx', SENO, 2, 'B7')
%Propiedades
prop=dir('propiedad*.mat');
for i=1:length(prop)
 nombres=load(prop(i).name);
 P(i,:)=nombres.P;
end
 P;
 PROP=zeros(mbr,2);
 h=E(:,2);
 for i=1:mbr
    PROP(i,1)=P(h(i),2);
    PROP(i,2)=P(h(i),3);
 end
PROP;
Elas=PROP(:,1);
Area=PROP(:,2);
aux=zeros(ngl,ngl);
linea=11;
K_total=zeros(2*nod,2*nod);
VCNN=zeros(mbr,4);
waitbar(30/100);
 pause(1);
for i=1:mbr
VCNN(i,:)=[ 2*ini(i)-1 2*ini(i) 2*fin(i)-1 2*fin(i) ];
end
for i=1 : mbr
r=((Elas(i)*Area(i))/L(i));
%Matriz de rigidez de miembro K2 en coordenadas locales
K2(i,1,1)=r;K2(i,1,2)=0;K2(i,1,3)=-r;K2(i,1,4)=0;
K2(i,2,1)=0;K2(i,2,2)=0;K2(i,2,3)=0;K2(i,2,4)=0;
K2(i,3,1)=-r;K2(i,3,2)=0;K2(i,3,3)=r;K2(i,3,4)=0;
K2(i,4,1)=0;K2(i,4,2)=0;K2(i,4,3)=0;K2(i,4,4)=0;
%Matriz de paso T2-3
T2_3(i,1,1)=COSENO(i);T2_3(i,1,2)=SENO(i);T2_3(i,1,3)=0;T2_3(i,1,4)=0;
T2_3(i,2,1)=-SENO(i);T2_3(i,2,2)=COSENO(i);T2_3(i,2,3)=0;T2_3(i,2,4)=0;
T2_3(i,3,1)=0;T2_3(i,3,2)=0;T2_3(i,3,3)=COSENO(i);T2_3(i,3,4)=SENO(i);
T2_3(i,4,1)=0;T2_3(i,4,2)=0;T2_3(i,4,3)=-SENO(i);T2_3(i,4,4)=COSENO(i);
%Matriz de rigidez de miembro K3 en coordenadas globales
for m=1:4
for n=1:4
K2I(m,n)=K2(i,m,n);
end
end
for m=1:4
for n=1:4
T2_3I(m,n)=T2_3(i,m,n);
end
end
% fprintf ('\n Matriz de Rigidez en Coord. Globales:');
K3=T2_3I'*K2I*T2_3I;
```

```
d = {'Matriz de Rigidez en Coord. Globales por miembro'};
xlswrite('Calculos.xlsx', d, 2, 'A9')
xlswrite('Calculos.xlsx', {['K',num2str(i),':']}, 2,
['A',num2str(linea)]);
 xlswrite('Calculos.xlsx', K3, 2,
['B',num2str(linea),':E',num2str(linea+3)]);
   linea=linea+size(K3,1)+2;
g_i=VCNN(i,:);
DeltaK_i=zeros(2*nod,2*nod);
DeltaK_i(g_i,g_i)=K3;
K_total=K_total+DeltaK_i;
mataux=K3;
for j=1 :4;
if VC(i,j)==0;
mataux(j,:)=0;
mataux(:,j)=0;
else,end
end
k=zeros(ngl,ngl);
for cont1=1:4;
for cont2=1:4;
if mataux (cont1,cont2)~=0,
uno=VC(i,cont1);
dos=VC(i,cont2);
tres=mataux(cont1,cont2);
k(uno,dos)=k(uno,dos)+tres;
else,end
end
end
if i==1;
aux=k;
else if i~=1,
aux=aux+k;
else,end
end
end
% fprintf ('\n Matriz de Rigidez de la Estructura:');
Mat_rigidez=aux;
waitbar(40/100);
 pause(1);
d = {'K total:'};
xlswrite('Calculos.xlsx', d, 2, ['A',num2str(13+6*mbr)])
xlswrite('Calculos.xlsx', {'Matriz de Rigidez Ensamblada de la Estructura
en Coordenadas Globales'}, 2, ['A',num2str(11+6*mbr)]);
MRIG1=[':',char(65+2*nod),];
MIRG2=num2str(13+2*nod+6*mbr);
MRIG3=[ MRIG1,(MIRG2),];
xlswrite('Calculos.xlsx', K_total, 2, ['B',num2str(14+6*mbr),'',MRIG3,]);
%Fuerzas
clear F Q1 VCJ
Q=zeros(1,ngl);
f=dir('fexternas*.mat');
for i=1:length(f)
 nombres=load(f(i).name);
```

```
 F(i,:)=nombres.P;
end
 F;
NJC=size(F);
njc=NJC(1);
for i=1 :njc
Q1(1)=F(i,2);
Q1(2)=F(i,3);
VCJ(i,:)=CG(F(i,1),:);
for m =1 :2
n=VCJ(i,m);
if n~=0
Q(n)=Q1(m);
else, end
end
end
Q_CJ=Q';
waitbar(50/100);
pause(1);
% fprintf( '\n Vector de Cargas Totales:');
xlswrite('Calculos.xlsx', {'Matriz de Rigidez reducida'}, 2,
['A',num2str(13+(3*ngl+2)+5*mbr)]);
xlswrite('Calculos.xlsx', {'K_reducida'}, 2,
['A',num2str(17+3*ngl+5*mbr)]);
MRIG1=[':',char(65+ngl),];
MIRG2=num2str(15+(3*ngl+2+ngl)+5*mbr);
MRIG3=[ MRIG1,(MIRG2),];
xlswrite('Calculos.xlsx',Mat_rigidez , 2,
['B',num2str(18+3*ngl+5*mbr),'',MRIG3,]);
MIRG2=num2str(19+(4*ngl)+5*mbr);
xlswrite('Calculos.xlsx', {'Vector de Cargas Totales(Ton)'}, 2,
['A',MIRG2,]);
MRIG1=num2str(20+(4*ngl)+5*mbr);
Fext1Alt=zeros(2*nod,1);
xlswrite('Calculos.xlsx', Fext1Alt, 2, ['B',MRIG1,]);
Q_TOTAL = Q_CJ;
waitbar(60/100);
 pause(1);
MIRG2=num2str(21+4*ngl+5*mbr+2*nod);
xlswrite('Calculos.xlsx', {'Vector de Cargas(Ton)'}, 2, ['A',MIRG2,]);
MRIG1=num2str(22+4*ngl+5*mbr+2*nod);
xlswrite('Calculos.xlsx', Q_TOTAL, 2, ['B',MRIG1,]);
% Vector desplazamientos K*q = Q
% fprintf( '\n Vector de Desplazamientos Generales:');
q_des = Mat_rigidez\Q_TOTAL;
MRIG1=num2str(23+5*ngl+5*mbr+2*nod);
Des1=zeros(2*nod,1);
xlswrite('Calculos.xlsx', {'Vector de Desplazamientos Totales(x,y,z)'},
2, ['A',MRIG1,]);
MRIG1=num2str(24+5*ngl+5*mbr+2*nod);
xlswrite('Calculos.xlsx', Des1, 2, ['B',MRIG1,]);
MRIG1=num2str(25+5*ngl+5*mbr+4*nod);
xlswrite('Calculos.xlsx', {'Vector de Desplazamientos generales'}, 2,
['A',MRIG1,]);
```

```
MRIG1=num2str(26+5*ngl+5*mbr+4*nod);
xlswrite('Calculos.xlsx', q_des, 2, ['B',MRIG1,]);
MRIG1=num2str(27+6*ngl+5*mbr+4*nod);
Reac1=zeros(2*nod,1);
xlswrite('Calculos.xlsx', {'Reacciones de la Estructura'}, 2,
['A',MRIG1,]);
MRIG1=num2str(28+6*ngl+5*mbr+4*nod);
xlswrite('Calculos.xlsx', Reac1, 2, ['B',MRIG1,]);
%
% MRIG1=num2str(29+6*ngl+5*mbr+4*nod);
% xlswrite('Calculos.xlsx', {'Desplazamiento y Esfuerzos Internos por
miembro'}, 2, ['A',MRIG1,]);
D_total=zeros(1,2*nod);
waitbar(70/100);
 pause(1);
% Deformaciones y acciones finales de miembro
linea=29+6*ngl+5*mbr+6*nod;
linea1=35+6*ngl+5*mbr+6*nod;
linea2=41+6*ngl+5*mbr+6*nod;
for i=1:mbr
% fprintf( '\n Miembro %d :',i);
% fprintf( '\n Deformaciones de miembro:');
clear P1
clear P2
clear P
clear Q2_aux
for j=1:4
if VC (i,j)~=0
P1(j)=q_des(VC(i,j));
else
P1(j)=0;
end
end
P1=P1';
MRIG1=num2str(linea);
xlswrite('Calculos.xlsx',{['Vector de desplazamientos
barra:',num2str(i),':']}, 2, ['A',MRIG1,]);
MRIG1=num2str(linea+1);
xlswrite('Calculos.xlsx', P1, 2, ['B',MRIG1,]);
linea=linea+18;
for j=1:4
    D_total(VCNN(i,j))=P1(j);
end
for m=1:4
for n=1:4
K2I(m,n)=K2(i,m,n);
end
end
for m=1:4
for n=1:4
T2_3I(m,n)=T2_3(i,m,n);
end
end
%Matriz K3
```

```
K3=T2_3I'*K2I*T2_3I;
% fprintf( '\n Acciones de miembro en coordenadas globales:');
P2=K3*P1;
MRIG1=num2str(linea1);
xlswrite('Calculos.xlsx',{['Esfuerzos en coordenadas globales
barra:',num2str(i),':']}, 2, ['A',MRIG1,]);
MRIG1=num2str(linea1+1);
xlswrite('Calculos.xlsx', P2, 2, ['B',MRIG1,]);
linea1=linea1+18;
% fprintf( '\n Acciones de miembro en coord. locales (P.
Complementario):');
P=T2_3I*P2;
% fprintf( '\n Acciones de miembro en coord. locales (P. Primario):');
% fprintf( '\n Acciones finales (PP+PC) :');
P_FINAL=P;
MRIG1=num2str(linea2);
xlswrite('Calculos.xlsx',{['Esfuerzos en coordenadas locales
barra:',num2str(i),':']}, 2, ['A',MRIG1,]);
MRIG1=num2str(linea2+1);
xlswrite('Calculos.xlsx', P_FINAL, 2, ['B',MRIG1,]);
linea2=linea2+18;
end
waitbar(80/100);
 pause(1);
D_total=D_total';
Reac=K_total*D_total;
Fext=zeros(mbr,2);
for i=1:njc
    Fext(F(i),:)=F(i,2:3);
end
Fext1=zeros(2*nod,1);
for i=1:nod
    Fext1(2*i-1)=Fext(i,1);
    Fext1(2*i)=Fext(i,2);
end
Reac_finales=Reac-Fext1;
waitbar(90/100);
 pause(1);
MRIG1=num2str(24+5*ngl+5*mbr+2*nod);
xlswrite('Calculos.xlsx', D_total, 2, ['B',MRIG1,]);
MRIG1=num2str(28+6*ngl+5*mbr+4*nod);
xlswrite('Calculos.xlsx', Reac_finales, 2, ['B',MRIG1,]);
MRIG1=num2str(20+(4*ngl)+5*mbr);
xlswrite('Calculos.xlsx', Fext1, 2, ['B',MRIG1,]);
save calcular_estructura
waitbar(100/100);
 pause(1);
delete(wb);
```

3.5. Mostrar Resultados

Esta rutina se encarga de mostrar un archivo Excel donde se muestra el procedimiento de cálculo paso a paso para así poder lograr un entendimiento completo de la forma de resolución de estructuras mediante el método de elementos finitos.

La rutina (Figura 3.21) recibe como nombre function mostrar_resultados_Callback y tiene el siguiente código de programación:

Codigo de programación function mostrar_resultados_Callback:

```
function mostrar_resultados_Callback(hObject, eventdata, handles)
winopen( 'Calculos.xlsx' );
```

3.6. Graficas

La rutina "Graficas" se encarga de mostrarnos los diagramas de esfuerzos de la estructura, en el caso de vigas y pórticos, la ventana nos muestra la opción de visualizar 3 graficas (Figura 3.22), la primera se encarga de mostrar la estructura a analizar, la segunda nos muestra el diagrama de cortantes y la tercera opción nos muestra el diagrama de momentos flectores.

Figura 3.21 Ventana de graficas en pórticos y vigas

En el programa nos permite visualizar 2 ventanas (Figura 3.23), la primera es la estructura a analizar y la segunda es la deformación que sufrirá la estructura debido a las cargas que actúan sobre ella.

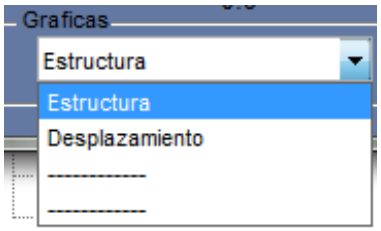


Figura 3.22 Ventana de Graficas en Armadura.

El código de programación para la ventana de Graficas en el caso de pórticos es el siguiente:

```
function estructura_Callback(hObject, eventdata, handles)
contenido=get(hObject,'String');
a=get(hObject,'Value');
texto=contenido(a);
switch cell2mat(texto)
    case 'Estructura'
Graficas
    case 'Cortante'
Graficas_Cortantes_Portico
    case 'Momentos'
Graficas_Momentos
end
```

El código de programación para la primera grafica denominada Estructura en el caso de pórticos fue denominado Graficas.m y su programación es la siguiente:

```
function varargout = Graficas(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Graficas_OpeningFcn, ...
                   'gui_OutputFcn',  @Graficas_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function Graficas_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

guidata(hObject, handles);
```

```
function varargout = Graficas_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;


function figure1_CreateFcn(hObject, eventdata, handles)
elem=dir('elemento*.mat');
for i=1:length(elem)
 nombres=load(elem(i).name);
 E(i,:)=nombres.P;
end
 ini=E(:,3);
 fin=E(:,4);
 a=dir('coordenada*.mat');
for i=1:length(a)
 nombres=load(a(i).name);
 V(i,:)=nombres.P;
end
x=V(:,2);
y=V(:,3);
for i=1:length(ini)
    co1=[x(ini(i)) x(fin(i))];
    co2=[y(ini(i)) y(fin(i))];
     hold on
     plot(co1,co2,'-k','linewidth',1.5)
end
for  i=1:length(ini)
    Ele=(x(ini(i))+x(fin(i)))/2;
    Ele1=(y(ini(i))+y(fin(i)))/2;
    hold on
    text(Ele,Ele1,{[num2str(E(i,1))]},'FontSize',12,'Color',[ 0 0 0 ])
end
for i=1:length(x)
    Co3=x(V(i,1))-0.04;
    Co4=y(V(i,1))+0.04;
    hold on
    text(Co3,Co4,{[num2str(V(i,1))]},'FontSize',12,'Color',[ 0 0 1])
end
axis([ min(x-0.2), max(x+0.2), min(y-0.2), max(y+0.2) ])
%Bordes
bordes=dir('bordes*.mat');
for i=1:length(bordes)
 nombres=load(bordes(i).name);
B(i,:)=nombres.P;
end
Tam=size(B);
for i=1:Tam(1,1)
if B(i,2:4)==[ 1 0 1];
hold on
fij1=[ x(B(i,1))-0.05  x(B(i,1))+0];
fij2=[ y(B(i,1))-0.05  y(B(i,1))+0];
plot(fij1,fij2)
hold on
fij3=[ x(B(i,1))+0 x(B(i,1))+0.05 ];
```

```
fij4=[ y(B(i,1))+0 y(B(i,1))-0.05 ];
plot(fij3,fij4)
hold on
fij5=[ x(B(i,1))-0.07 x(B(i,1))+0.07 ];
fij6=[ y(B(i,1))-0.05 y(B(i,1))-0.05 ];
plot(fij5,fij6)
hold on
fij7=[ x(B(i,1))-0.07 x(B(i,1))+0.07 ];
fij8=[ y(B(i,1))-0.08 y(B(i,1))-0.08 ];
plot(fij7,fij8)
elseif B(i,2:4)==[ 0 0 1]
 hold on
fij1=[ x(B(i,1))-0.05  x(B(i,1))+0];
fij2=[ y(B(i,1))-0.05  y(B(i,1))+0];
plot(fij1,fij2)
hold on
fij3=[ x(B(i,1))+0 x(B(i,1))+0.05 ];
fij4=[ y(B(i,1))+0 y(B(i,1))-0.05 ];
plot(fij3,fij4)
hold on
fij5=[ x(B(i,1))-0.07 x(B(i,1))+0.07 ];
fij6=[ y(B(i,1))-0.05 y(B(i,1))-0.05 ];
plot(fij5,fij6)
hold on
fij7=[ x(B(i,1))-0.07 x(B(i,1))-0.04 ];
fij8=[ y(B(i,1))-0.05 y(B(i,1))-0.08 ];
plot(fij7,fij8)
hold on
fij9=[ x(B(i,1))-0.046667 x(B(i,1))-0.016667];
plot(fij9,fij8)
hold on
fij10=[ x(B(i,1))-0.023334 x(B(i,1))+0.00666];
plot(fij10,fij8)
hold on
fij11=[ x(B(i,1))+0 x(B(i,1))+0.029999];
plot(fij11,fij8)
hold on
fij12=[x(B(i,1))+0.023333 x(B(i,1))+0.053332];
plot(fij12,fij8)
hold on
fij13=[x(B(i,1))+0.046666 x(B(i,1))+0.076662];
plot(fij13,fij8)
hold on
fij14=[x(B(i,1))+0.069999 x(B(i,1))+0.099995];
plot(fij14,fij8)
elseif B(i,2:4)==[ 0 0 0]
    hold on
fij5=[ x(B(i,1))-0.07 x(B(i,1))+0.07 ];
fij6=[ y(B(i,1))+0 y(B(i,1))+0 ];
plot(fij5,fij6)
hold on
fij7=[x(B(i,1))-0.07 x(B(i,1))-0.04 ];
fij8=[y(B(i,1))+0 y(B(i,1))-0.03 ];
plot(fij7,fij8)
```

```
hold on
fij9=[x(B(i,1))-0.046667 x(B(i,1))-0.016667];
plot(fij9,fij8)
hold on
fij10=[x(B(i,1))-0.023334 x(B(i,1))+0.00666];
plot(fij10,fij8)
hold on
fij11=[ x(B(i,1))+0 x(B(i,1))+0.029999];
plot(fij11,fij8)
hold on
fij12=[x(B(i,1))+0.023333 x(B(i,1))+0.053332];
plot(fij12,fij8)
hold on
fij13=[x(B(i,1))+0.046666 x(B(i,1))+0.076662];
plot(fij13,fij8)
hold on
fij14=[x(B(i,1))+0.069999 x(B(i,1))+0.099995];
plot(fij14,fij8)
end
end
xlabel('Longitud[m]');
ylabel('Altura[m]');
title('ESTRUCTURA')
```

El código de programación para el caso de diagrama de cortante en un pórtico es el siguiente y fue denominado como Graficas_Cortantes_Portico.m:

```
function varargout = Graficas_Cortantes_Portico(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Graficas_Cortantes_Portico_OpeningFcn, ...
                   'gui_OutputFcn',
@Graficas_Cortantes_Portico_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function Graficas_Cortantes_Portico_OpeningFcn(hObject, eventdata,
handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = Graficas_Cortantes_Portico_OutputFcn(hObject,
eventdata, handles)
varargout{1} = handles.output;
```

```
function figure1_CreateFcn(hObject, eventdata, handles)
clear
load calcular_estructura
% Matriz de nodos
n = V(:,2:3)
% Matriz de barras
b=E(:,3:4)
%Restricciones en los nodos
 %Bordes
bordes=dir('bordes*.mat');
for i=1:length(bordes)
 nombres=load(bordes(i).name);
B(i,:)=nombres.P;
end
 B
 NNR=size(B)
 nnr=NNR(:,1)
 RestriccionesNodo=ones(nod,3)
 for i=1:nnr
     m=B(i,1)
     RestriccionesNodo(m,1)=B(i,2)
     RestriccionesNodo(m,2)=B(i,3)
     RestriccionesNodo(m,3)=B(i,4)
 end
%Cargas aplicadas:
clear Mom
mom=dir('momento*.mat');
for i=1:length(mom)
 nombres=load(mom(i).name);
 Mom(i,:)=nombres.P;
end
nmc=size(Mom,1);
Pd=zeros(mbr,1)
for i=1:nmc
    if Mom(i,9)==1
    Pd(Mom(i))=Mom(i,8)
    end
end
Pt=zeros(mbr,1)
for i=1:nmc
    if Mom(i,9)==2
    Pt(Mom(i))=Mom(i,8)
    end
end
Pp=zeros(mbr,2)
for i=1:nmc
    if Mom(i,9)==3
    Pp(Mom(i),1)=Mom(i,8)
    Pp(Mom(i),2)=Mom(i,10)
    end
end
%Desplazamientos producidos en los movimientos no restringidos de los
nodos
```

```
Desp=q_des;
N=size(b);
numeronodos=size(n);
for i=1:N(1,1)
barra(i,:)=[n((b(i,1)),:) n((b(i,2)),:)];
end
% Datos de cada barra
for i=1:N(1,1)
L(i,1)= (((barra(i,3)-barra(i,1))^2) + ((barra(i,4)-barra(i,2))^2))^0.5;
angulo(i,1) = atan((barra(i,4)-barra(i,2))/(barra(i,3)-barra(i,1)));
end
% Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
if ((barra(i,1))<(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
alfa(i,:)=0;
elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=angulo(i,:);
elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=pi/2;
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=(pi/2)+angulo(i,:);
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
alfa(i,:)=pi;
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
alfa(i,:)=pi+angulo(i,:);
elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
    alfa(i,:)=(3*pi)/2;
elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
alfa(i,:)=(2*pi)+angulo(i,:);
end
end
% Particiones barra
for i =1 : N(1,1)
Li(i,:) = L(i,1) /100;
xL(i,:)=(0:Li(i,:):L(i));
end
% barras para el dibujo
for i=1:N(1,1)
for j=1 : 101
xb(i,j)= barra(i,1) + xL(i,j) * cos(alfa(i,1));
yb(i,j)= barra(i,2) + xL(i,j) * sin(alfa(i,1));
end
end
Desplazamientosentero=zeros(numeronodos(1,1),3);
contadordesp=1;
Restriccionesnododesp=RestriccionesNodo;
for i =1:numeronodos
for j=1:3;
if Restriccionesnododesp(i,j)==0
Desplazamientosentero(i,j)=0;
end
if Restriccionesnododesp(i)==1
Desplazamientosentero(i,j)=Desplazamientosentero(i,j)+Desp(contadordesp,1
);
```

```
contadordesp=1+contadordesp;
end
end
end
for i = 1:N(1,1)
for j=1:101
if Pd(i)==0
Md(i,j)=0;
else if Pd(i)~=0
Md(i,j)= -(Pd(i)*(xL(i,j)))+Fg(i,2);
end
end
end
end
% Calculo del momento producido por una carga triangular
for i=1:N(1,1)
for j=1:101
if Pt(i)==0
Mt(i,j)= 0;
else if Pt(i)~=0
Mt(i,j)=-((Pt(i)*xL(i,j)^2)/(2*L(i)))+Fg(i,2);
end
end
end
end
% Momento en barras con una carga puntual
for i =1:N(1,1)
for j=1:101
if Pp(i,1)==0
Mp(i,j)=0;
else if Pp(i,1)~=0
if (xL(i,j)/L(i))< Pp(i,2)
Mp(i,j)=Fg(i,2);
end
if (xL(i,j)/L(i))>= Pp(i,2)
Mp(i,j)= -Pp(i,1)+Fg(i,2);
end
    end
end
end
end
M=Md+Mt+Mp;
for i = 1:N(1,1)
for j=1:101
Mm1(i,j)=Fg(i,2);
end
end
for i=1:nmc
    Mm1(Mom(i),:)=M(Mom(i),:)
end
M=Mm1
% Factor de escala
Esc=(L(1)/10)/max(max(abs(M)));
% Escalado
```

```
for i=1:N(1,1)
ME(i,:)=M(i,:)*Esc;
end
% Giro momentos para su representación
for i=1:N(1,1)
for j=1:101
radio(i,j)=sqrt((xL(i,j)^2)+(ME(i,j)^2));
theta(i,j)= atan(ME(i,j)/xL(i,j)) + alfa(i);
xm(i,j)= barra(i,1)+ radio(i,j)*cos(theta(i,j));
ym(i,j)= barra(i,2) + radio(i,j)*sin(theta(i,j));
end
end
load calcular_estructura
for  i=1:length(ini)
    Ele=(x(ini(i))+x(fin(i)))/2;
    Ele1=(y(ini(i))+y(fin(i)))/2;
    hold on
    text(Ele,Ele1,{[num2str(E(i,1))]},'FontSize',12,'Color',[ 0 0 0 ])
end
for i=1:length(x)
    Co3=x(V(i,1))-0.04;
    Co4=y(V(i,1))+0.04;
    hold on
    text(Co3,Co4,{[num2str(V(i,1))]},'FontSize',12,'Color',[ 0 0 1])
end
%Representación figura
hold on
for i=1:N(1,1)
plot(xb(i,:),yb(i,:),'k','linewidth',1)
plot(xm(i,:),ym(i,:),'b','linewidth',2)
end
xlabel('Longitud[m]');
ylabel('Altura[m]');
title('ESTRUCTURA')
```

El código de programación para la visualización del diagrama de momentos flectores es el siguiente y fue denominado Graficas_Momentos.m:

```
function varargout = Graficas_Momentos(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Graficas_Momentos_OpeningFcn, ...
                   'gui_OutputFcn',  @Graficas_Momentos_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
```

```
    gui_mainfcn(gui_State, varargin{:});
end

function Graficas_Momentos_OpeningFcn(hObject, eventdata, handles,
varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = Graficas_Momentos_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
function figure1_CreateFcn(hObject, eventdata, handles)
clear
load calcular_estructura
% Matriz de nodos
n = V(:,2:3)
% Matriz de barras
b=E(:,3:4)
%Restricciones en los nodos
 %Bordes
bordes=dir('bordes*.mat');
for i=1:length(bordes)
 nombres=load(bordes(i).name);
B(i,:)=nombres.P;
end
 B
 NNR=size(B)
 nnr=NNR(:,1)
 RestriccionesNodo=ones(nod,3)
 for i=1:nnr
     m=B(i,1)
     RestriccionesNodo(m,1)=B(i,2)
     RestriccionesNodo(m,2)=B(i,3)
     RestriccionesNodo(m,3)=B(i,4)
 end
%Cargas aplicadas:
clear Mom
mom=dir('momento*.mat');
for i=1:length(mom)
 nombres=load(mom(i).name);
 Mom(i,:)=nombres.P;
end
nmc=size(Mom,1);
Pd=zeros(mbr,1)
for i=1:nmc
    if Mom(i,9)==1
    Pd(Mom(i))=Mom(i,8)
    end
end
Pt=zeros(mbr,1)
for i=1:nmc
    if Mom(i,9)==2
    Pt(Mom(i))=Mom(i,8)
    end
end
```

```
Pp=zeros(mbr,2)
for i=1:nmc
    if Mom(i,9)==3
    Pp(Mom(i),1)=Mom(i,8)
    Pp(Mom(i),2)=Mom(i,10)
    end
end
%Desplazamientos producidos en los movimientos no restringidos de los
nodos
Desp=q_des;
N=size(b);
numeronodos=size(n);
for i=1:N(1,1)
barra(i,:)=[n((b(i,1)),:) n((b(i,2)),:)];
end
% Datos de cada barra
for i=1:N(1,1)
L(i,1)= (((barra(i,3)-barra(i,1))^2) + ((barra(i,4)-barra(i,2))^2))^0.5;
angulo(i,1) = atan((barra(i,4)-barra(i,2))/(barra(i,3)-barra(i,1)));
end
% Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
if ((barra(i,1))<(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
alfa(i,:)=0;
elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=angulo(i,:);
elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=pi/2;
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=(pi/2)+angulo(i,:);
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
alfa(i,:)=pi;
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
alfa(i,:)=pi+angulo(i,:);
elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
    alfa(i,:)=(3*pi)/2;
elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
alfa(i,:)=(2*pi)+angulo(i,:);
end
end
% Particiones barra
for i =1 : N(1,1)
Li(i,:) = L(i,1) /100;
xL(i,:)=(0:Li(i,:):L(i));
end
% barras para el dibujo
for i=1:N(1,1)
for j=1 : 101
xb(i,j)= barra(i,1) + xL(i,j) * cos(alfa(i,1));
yb(i,j)= barra(i,2) + xL(i,j) * sin(alfa(i,1));
end
end
%Vector desplazamiento completo organizado en filas, cada fila tiene 3
```

```
%columnas, correspondientes a los 3 tipos de desplazamientos que tiene
cada
%nodo
Desplazamientosentero=zeros(numeronodos(1,1),3);
contadordesp=1;
RestriccionesnododesP=RestriccionesNodo;
for i =1:numeronodos
for j=1:3;
if Restriccionesnododesp(i,j)==0
Desplazamientosentero(i,j)=0;
end
if Restriccionesnododesp(i)==1
Desplazamientosentero(i,j)=Desplazamientosentero(i,j)+Desp(contadordesp,1
);
contadordesp=1+contadordesp;
end
end
end
% Calculo del momento producido por una carga triangular
for i = 1:N(1,1)
for j=1:101
if Pd(i)==0
Md(i,j)=0;
else if Pd(i)~=0
Md(i,j)= ((Pd(i)*(xL(i,j)^2))/2)-(Fg(i,2)*xL(i,j)) +Fg(i,3);
end
end
end
end
% Calculo del momento producido por una carga triangular
for i=1:N(1,1)
for j=1:101
if Pt(i)==0
Mt(i,j)= 0;
else if Pt(i)~=0
Mt(i,j)=((Pt(i)*xL(i,j)^3)/(6*L(i)))-(Fg(i,2)*xL(i,j))+ Fg(i,3);
end
end
end
end
% Momento en barras con una carga puntual
for i =1:N(1,1)
for j=1:101
if Pp(i,1)==0
Mp(i,j)=0;
else if Pp(i,1)~=0
if (xL(i,j)/L(i))< Pp(i,2)
Mp(i,j)= Fg(i,3) -(Fg(i,2)*xL(i,j));
end
if (xL(i,j)/L(i))>= Pp(i,2)
Mp(i,j)= (Pp(i,1)*(xL(i,j)-Pp(i,2)*L(i))) + Fg(i,3)- (Fg(i,2)*xL(i,j));
end
    end
end
```

```
end
end
M=Md+Mt+Mp;
for i = 1:N(1,1)
for j=1:101
Mm1(i,j)=Fg(i,3)-(Fg(i,2)*xL(i,j));
end
end
for i=1:nmc
    Mm1(Mom(i),:)=M(Mom(i),:)
end
M=Mm1
% Factor de escala
Esc=(L(1)/10)/max(max(abs(M)));
% Escalado
for i=1:N(1,1)
ME(i,:)=M(i,:)*Esc;
end
% Giro momentos para su representación
for i=1:N(1,1)
for j=1:101
radio(i,j)=sqrt((xL(i,j)^2)+(ME(i,j)^2));
theta(i,j)= atan(ME(i,j)/xL(i,j)) + alfa(i);
xm(i,j)= barra(i,1)+ radio(i,j)*cos(theta(i,j));
ym(i,j)= barra(i,2) + radio(i,j)*sin(theta(i,j));
end
end
load calcular_estructura
for  i=1:length(ini)
    Ele=(x(ini(i))+x(fin(i)))/2;
    Ele1=(y(ini(i))+y(fin(i)))/2;
    hold on
    text(Ele,Ele1,{[num2str(E(i,1))]},'FontSize',12,'Color',[ 0 0 0 ])
end
for i=1:length(x)
    Co3=x(V(i,1))-0.04;
    Co4=y(V(i,1))+0.04;
    hold on
    text(Co3,Co4,{[num2str(V(i,1))]},'FontSize',12,'Color',[ 0 0 1])
end
%Representación figura
hold on
for i=1:N(1,1)
plot(xb(i,:),yb(i,:),'k','linewidth',1)
plot(xm(i,:),ym(i,:),'b','linewidth',2)
end
xlabel('Longitud[m]');
ylabel('Altura[m]');
title('ESTRUCTURA')
```

En el caso de vigas se muestra a continuación su código de programación, primeramente para la Estructura a calcular, luego para Cortantes y Despues para Momentos.

Codigo de programación para visualizar la viga a analizar:

```
function varargout = GraficasV(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GraficasV_OpeningFcn, ...
                   'gui_OutputFcn',  @GraficasV_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function GraficasV_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = GraficasV_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function figure1_CreateFcn(hObject, eventdata, handles)
elem=dir('elemento*.mat');
for i=1:length(elem)
 nombres=load(elem(i).name);
 E(i,:)=nombres.P;
end
 ini=E(:,3);
 fin=E(:,4);
 a=dir('coordenada*.mat');
for i=1:length(a)
 nombres=load(a(i).name);
 V(i,:)=nombres.P;
end
x=V(:,2);
y=V(:,3);
for i=1:length(ini)
    co1=[x(ini(i)) x(fin(i))];
    co2=[y(ini(i)) y(fin(i))];
     hold on
     plot(co1,co2,'-k','linewidth',1.5)
end
for  i=1:length(ini)
    Ele=(x(ini(i))+x(fin(i)))/2;
    Ele1=((y(ini(i))+y(fin(i)))/2-0.05);
    hold on
    text(Ele,Ele1,{[num2str(E(i,1))]},'FontSize',12,'Color',[ 0 0 0 ])
end
for i=1:length(x)
    Co3=x(V(i,1))-0.04;
    Co4=y(V(i,1))+0.04;
```

```
    hold on
    text(Co3,Co4,{[num2str(V(i,1))]},'FontSize',12,'Color',[ 0 0 1])
end
axis([ min(x-0.2), max(x+0.2), min(y), max(y) ])
%Bordes
bordes=dir('bordes*.mat');
for i=1:length(bordes)
 nombres=load(bordes(i).name);
B(i,:)=nombres.P;
end
Tam=size(B);
for i=1:Tam(1,1)
if B(i,2:4)==[ 1 0 1];
hold on
fij1=[ x(B(i,1))-0.05  x(B(i,1))+0];
fij2=[ y(B(i,1))-0.05  y(B(i,1))+0];
plot(fij1,fij2)
hold on
fij3=[ x(B(i,1))+0 x(B(i,1))+0.05 ];
fij4=[ y(B(i,1))+0 y(B(i,1))-0.05 ];
plot(fij3,fij4)
hold on
fij5=[ x(B(i,1))-0.07 x(B(i,1))+0.07 ];
fij6=[ y(B(i,1))-0.05 y(B(i,1))-0.05 ];
plot(fij5,fij6)
hold on
fij7=[ x(B(i,1))-0.07 x(B(i,1))+0.07 ];
fij8=[ y(B(i,1))-0.08 y(B(i,1))-0.08 ];
plot(fij7,fij8)
elseif B(i,2:4)==[ 0 0 1]
 hold on
fij1=[ x(B(i,1))-0.05  x(B(i,1))+0];
fij2=[ y(B(i,1))-0.05  y(B(i,1))+0];
plot(fij1,fij2)
hold on
fij3=[ x(B(i,1))+0 x(B(i,1))+0.05 ];
fij4=[ y(B(i,1))+0 y(B(i,1))-0.05 ];
plot(fij3,fij4)
hold on
fij5=[ x(B(i,1))-0.07 x(B(i,1))+0.07 ];
fij6=[ y(B(i,1))-0.05 y(B(i,1))-0.05 ];
plot(fij5,fij6)
hold on
fij7=[ x(B(i,1))-0.07 x(B(i,1))-0.04 ];
fij8=[ y(B(i,1))-0.05 y(B(i,1))-0.08 ];
plot(fij7,fij8)
hold on
fij9=[ x(B(i,1))-0.046667 x(B(i,1))-0.016667];
plot(fij9,fij8)
hold on
fij10=[ x(B(i,1))-0.023334 x(B(i,1))+0.00666];
plot(fij10,fij8)
hold on
fij11=[ x(B(i,1))+0 x(B(i,1))+0.029999];
```

```
plot(fij11,fij8)
hold on
fij12=[x(B(i,1))+0.023333 x(B(i,1))+0.053332];
plot(fij12,fij8)
hold on
fij13=[x(B(i,1))+0.046666 x(B(i,1))+0.076662];
plot(fij13,fij8)
hold on
fij14=[x(B(i,1))+0.069999 x(B(i,1))+0.099995];
plot(fij14,fij8)
elseif B(i,2:4)==[ 0 0 0]
    hold on
fij5=[ x(B(i,1))-0.07 x(B(i,1))+0.07 ];
fij6=[ y(B(i,1))+0 y(B(i,1))+0 ];
plot(fij5,fij6)
hold on
fij7=[x(B(i,1))-0.07 x(B(i,1))-0.04 ];
fij8=[y(B(i,1))+0 y(B(i,1))-0.03 ];
plot(fij7,fij8)
hold on
fij9=[x(B(i,1))-0.046667 x(B(i,1))-0.016667];
plot(fij9,fij8)
hold on
fij10=[x(B(i,1))-0.023334 x(B(i,1))+0.00666];
plot(fij10,fij8)
hold on
fij11=[ x(B(i,1))+0 x(B(i,1))+0.029999];
plot(fij11,fij8)
hold on
fij12=[x(B(i,1))+0.023333 x(B(i,1))+0.053332];
plot(fij12,fij8)
hold on
fij13=[x(B(i,1))+0.046666 x(B(i,1))+0.076662];
plot(fij13,fij8)
hold on
fij14=[x(B(i,1))+0.069999 x(B(i,1))+0.099995];
plot(fij14,fij8)
end
end
xlabel('Longitud[m]');
ylabel('Altura[m]');
title('ESTRUCTURA')
```

El código de programación para cortantes es el siguiente:

```
function varargout = Grafico_Cortante_Vigas(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Grafico_Cortante_Vigas_OpeningFcn,
...
                   'gui_OutputFcn',  @Grafico_Cortante_Vigas_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
```

```
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function Grafico_Cortante_Vigas_OpeningFcn(hObject, eventdata, handles,
varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = Grafico_Cortante_Vigas_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
function figure1_CreateFcn(hObject, eventdata, handles)
clear
load calcular_estructura
% Matriz de nodos
n = V(:,2:3)
% Matriz de barras
b=E(:,3:4)
%Restricciones en los nodos
 %Bordes
 bordes=dir('bordes*.mat');
for i=1:length(bordes)
 nombres=load(bordes(i).name);
B(i,:)=nombres.P;
end
 B
 NNR=size(B)
 nnr=NNR(:,1)
 RestriccionesNodo=ones(nod,2)
 for i=1:nnr
     m=B(i,1)
     RestriccionesNodo(m,1)=B(i,3)
     RestriccionesNodo(m,2)=B(i,4)
 end
 %Cargas aplicadas:
clear Mom
mom=dir('momento*.mat');
for i=1:length(mom)
 nombres=load(mom(i).name);
 Mom(i,:)=nombres.P;
end
 nmc=size(Mom,1);
Pd=zeros(mbr,1)
for i=1:nmc
    if Mom(i,9)==1
    Pd(Mom(i))=Mom(i,8)
    end
end
Pt=zeros(mbr,1)
```

```
for i=1:nmc
    if Mom(i,9)==2
    Pt(Mom(i))=Mom(i,8)
    end
end
Pp=zeros(mbr,2)
for i=1:nmc
    if Mom(i,9)==3
    Pp(Mom(i),1)=Mom(i,8)
    Pp(Mom(i),2)=Mom(i,10)
    end
end
%Desplazamientos producidos en los movimientos no restringidos de los
nodos
Desp=q_des;
N=size(b);
numeronodos=size(n);
for i=1:N(1,1)
barra(i,:)=[n((b(i,1)),:) n((b(i,2)),:)];
end
% Datos de cada barra
for i=1:N(1,1)
L(i,1)= (((barra(i,3)-barra(i,1))^2) + ((barra(i,4)-barra(i,2))^2))^0.5;
angulo(i,1) = atan((barra(i,4)-barra(i,2))/(barra(i,3)-barra(i,1)));
end
% Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
if ((barra(i,1))<(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
alfa(i,:)=0;
elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=angulo(i,:);
elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=pi/2;
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=(pi/2)+angulo(i,:);
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
alfa(i,:)=pi;
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
alfa(i,:)=pi+angulo(i,:);
elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
    alfa(i,:)=(3*pi)/2;
elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
alfa(i,:)=(2*pi)+angulo(i,:);
end
end
% Particiones barra
for i =1 : N(1,1)
Li(i,:) = L(i,1) /100;
xL(i,:)=(0:Li(i,:):L(i));
end
% barras para el dibujo
for i=1:N(1,1)
for j=1 : 101
xb(i,j)= barra(i,1) + xL(i,j) * cos(alfa(i,1));
```

```
yb(i,j)= barra(i,2) + xL(i,j) * sin(alfa(i,1));
end
end
Desplazamientosentero=zeros(numeronodos(1,1),2);
for i=1:mbr
    Desplazamientosentero(i,1)=D_total(2*i-1)
    Desplazamientosentero(i,2)=D_total(2*i)
end
for i = 1:N(1,1)
for j=1:101
if Pd(i)==0
Md(i,j)=0;
else if Pd(i)~=0
Md(i,j)=-Fg(i,1)+(Pd(i)*(xL(i,j)));
end
end
end
end
% Calculo del momento producido por una carga triangular
for i=1:N(1,1)
for j=1:101
if Pt(i)==0
Mt(i,j)= 0;
else if Pt(i)~=0
Mt(i,j)=-Fg(i,1)+((Pt(i)*xL(i,j)^2)/(2*L(i)));
end
end
end
end
% Momento en barras con una carga puntual
for i =1:N(1,1)
for j=1:101
if Pp(i,1)==0
Mp(i,j)=0;
else if Pp(i,1)~=0
if (xL(i,j)/L(i))< Pp(i,2)
Mp(i,j)=-Fg(i,1);
end
if (xL(i,j)/L(i))>= Pp(i,2)
Mp(i,j)=Pp(i,1)-Fg(i,1);
end
    end
end
end
end
M=Md+Mt+Mp;
for i = 1:N(1,1)
for j=1:101
Mm1(i,j)=-Fg(i,1);
end
end
for i=1:nmc
    Mm1(Mom(i),:)=M(Mom(i),:)
end
```

```
M=Mm1
% Escalado
for i=1:N(1,1)
ME(i,:)=M(i,:);
end
% Giro momentos para su representación
for i=1:N(1,1)
for j=1:101
radio(i,j)=sqrt((xL(i,j)^2)+(ME(i,j)^2));
theta(i,j)= atan(ME(i,j)/xL(i,j)) + alfa(i);
xm(i,j)= barra(i,1)+ radio(i,j)*cos(theta(i,j));
ym(i,j)= barra(i,2) + radio(i,j)*sin(theta(i,j));
end
end
load calcular_estructura
for  i=1:length(ini)
    Ele=(x(ini(i))+x(fin(i)))/2;
    Ele1=(y(ini(i))+y(fin(i)))/2;
    hold on
    text(Ele,Ele1,{[num2str(E(i,1))]},'FontSize',12,'Color',[ 0 0 0 ])
end
%Representación figura
hold on
for i=1:N(1,1)
plot(xb(i,:),yb(i,:),'k','linewidth',1)
plot(xm(i,:),ym(i,:),'b','linewidth',2)
end
xlabel('Longitud[m]');
ylabel('Altura[m]');
title('ESTRUCTURA')
clc
```

El código de programación para diagramar momentos flectores es el siguiente:

```
function varargout = Graficas_Momentos_Vigas(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Graficas_Momentos_Vigas_OpeningFcn,
...
                   'gui_OutputFcn',  @Graficas_Momentos_Vigas_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
function Graficas_Momentos_Vigas_OpeningFcn(hObject, eventdata, handles,
varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = Graficas_Momentos_Vigas_OutputFcn(hObject,
eventdata, handles)
varargout{1} = handles.output;
function figure1_CreateFcn(hObject, eventdata, handles)
clear
load calcular_estructura
% Matriz de nodos
n = V(:,2:3)
% Matriz de barras
b=E(:,3:4)
%Restricciones en los nodos
 %Bordes
 bordes=dir('bordes*.mat');
for i=1:length(bordes)
 nombres=load(bordes(i).name);
B(i,:)=nombres.P;
end
 B
 NNR=size(B)
 nnr=NNR(:,1)
 RestriccionesNodo=ones(nod,2)
 for i=1:nnr
     m=B(i,1)
     RestriccionesNodo(m,1)=B(i,3)
     RestriccionesNodo(m,2)=B(i,4)
 end
 %Cargas aplicadas:
clear Mom
mom=dir('momento*.mat');
for i=1:length(mom)
 nombres=load(mom(i).name);
 Mom(i,:)=nombres.P;
end
 nmc=size(Mom,1);
Pd=zeros(mbr,1)
for i=1:nmc
    if Mom(i,9)==1
    Pd(Mom(i))=Mom(i,8)
    end
end
Pt=zeros(mbr,1)
for i=1:nmc
    if Mom(i,9)==2
    Pt(Mom(i))=Mom(i,8)
    end
end
Pp=zeros(mbr,2)
for i=1:nmc
    if Mom(i,9)==3
    Pp(Mom(i),1)=Mom(i,8)
```

```
    Pp(Mom(i),2)=Mom(i,10)
    end
end
%Desplazamientos producidos en los movimientos no restringidos de los
nodos
N=size(b);
numeronodos=size(n);
for i=1:N(1,1)
barra(i,:)=[n((b(i,1)),:) n((b(i,2)),:)];
end
% Datos de cada barra
for i=1:N(1,1)
L(i,1)= (((barra(i,3)-barra(i,1))^2) + ((barra(i,4)-barra(i,2))^2))^0.5;
angulo(i,1) = atan((barra(i,4)-barra(i,2))/(barra(i,3)-barra(i,1)));
end
% Ajuste ángulo alfa de cada barra
for i=1:N(1,1)
if ((barra(i,1))<(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
alfa(i,:)=0;
elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=angulo(i,:);
elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=pi/2;
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))<(barra(i,4)))
alfa(i,:)=(pi/2)+angulo(i,:);
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))==(barra(i,4)))
alfa(i,:)=pi;
elseif ((barra(i,1))>(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
alfa(i,:)=pi+angulo(i,:);
elseif ((barra(i,1))==(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
    alfa(i,:)=(3*pi)/2;
elseif ((barra(i,1))<(barra(i,3))) && ((barra(i,2))>(barra(i,4)))
alfa(i,:)=(2*pi)+angulo(i,:);
end
end
% Particiones barra
for i =1 : N(1,1)
Li(i,:) = L(i,1) /100;
xL(i,:)=(0:Li(i,:):L(i));
end
% barras para el dibujo
for i=1:N(1,1)
for j=1 : 101
xb(i,j)= barra(i,1) + xL(i,j) * cos(alfa(i,1));
yb(i,j)= barra(i,2) + xL(i,j) * sin(alfa(i,1));
end
end
Desplazamientosentero=zeros(numeronodos(1,1),2);
for i=1:mbr
    Desplazamientosentero(i,1)=D_total(2*i-1)
    Desplazamientosentero(i,2)=D_total(2*i)
end
for i = 1:N(1,1)
for j=1:101
```

```
if Pd(i)==0
Md(i,j)=0;
else if Pd(i)~=0
Md(i,j)= Fg(i,2)-(Fg(i,1)*xL(i,j))+((Pd(i)*(xL(i,j)^2))/2);
end
end
end
end
% Calculo del momento producido por una carga triangular
for i=1:N(1,1)
for j=1:101
if Pt(i)==0
Mt(i,j)= 0;
else if Pt(i)~=0
Mt(i,j)=Fg(i,2)-(Fg(i,1)*xL(i,j))+((Pt(i)*xL(i,j)^3)/(6*L(i)));
end
end
end
end
% Momento en barras con una carga puntual
for i =1:N(1,1)
for j=1:101
if Pp(i,1)==0
Mp(i,j)=0;
else if Pp(i,1)~=0
if (xL(i,j)/L(i))< Pp(i,2)
Mp(i,j)= Fg(i,2) -(Fg(i,1)*xL(i,j));
end
if (xL(i,j)/L(i))>= Pp(i,2)
Mp(i,j)= (Pp(i,1)*(xL(i,j)-Pp(i,2)*L(i))) + Fg(i,2)- (Fg(i,1)*xL(i,j));
end
    end
end
end
end
M=Md+Mt+Mp;
for i = 1:N(1,1)
for j=1:101
Mm1(i,j)=Fg(i,2)-(Fg(i,1)*xL(i,j));
end
end
for i=1:nmc
    Mm1(Mom(i),:)=M(Mom(i),:)
end
M=Mm1
% Escalado
for i=1:N(1,1)
ME(i,:)=M(i,:);
end
% Giro momentos para su representación
for i=1:N(1,1)
for j=1:101
radio(i,j)=sqrt((xL(i,j)^2)+(ME(i,j)^2));
theta(i,j)= atan(ME(i,j)/xL(i,j)) + alfa(i);
```

```
xm(i,j)= barra(i,1)+ radio(i,j)*cos(theta(i,j));
ym(i,j)= barra(i,2) + radio(i,j)*sin(theta(i,j));
end
end
load calcular_estructura
for  i=1:length(ini)
    Ele=(x(ini(i))+x(fin(i)))/2;
    Ele1=(y(ini(i))+y(fin(i)))/2;
    hold on
    text(Ele,Ele1,{[num2str(E(i,1))]},'FontSize',12,'Color',[ 0 0 0 ])
end
%Representación figura
hold on
for i=1:N(1,1)
plot(xb(i,:),yb(i,:),'k','linewidth',1)
plot(xm(i,:),ym(i,:),'b','linewidth',2)
end
xlabel('Longitud[m]');
ylabel('Altura[m]');
title('ESTRUCTURA')
```

Finalmente para el caso de armaduras se tiene el siguiente código de programación para visualizar su estructura deformada:

```
function varargout = GraficasAD(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GraficasAD_OpeningFcn, ...
                   'gui_OutputFcn',  @GraficasAD_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function GraficasAD_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = GraficasAD_OutputFcn(hObject, eventdata, handles)
function figure1_CreateFcn(hObject, eventdata, handles)
load calcular_estructura
for  i=1:length(ini)
    Ele=(x(ini(i))+x(fin(i)))/2;
    Ele1=(y(ini(i))+y(fin(i)))/2;
    hold on
    text(Ele,Ele1,{[num2str(E(i,1))]},'FontSize',12,'Color',[ 0 0 0 ])
end
```

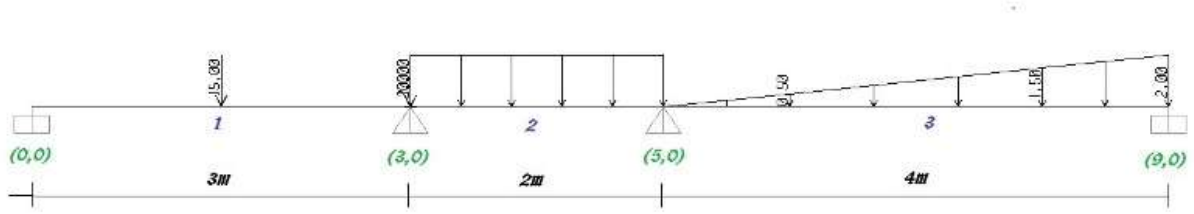
```
for i=1:length(x)
    Co3=x(V(i,1))-0.04;
    Co4=y(V(i,1))+0.04;
    hold on
    text(Co3,Co4,{[num2str(V(i,1))]},'FontSize',12,'Color',[ 0 0 1])
end
axis([ min(x-0.2), max(x+0.2), min(y-0.2), max(y+0.2) ])
%grafica
XY=V(:,2:3)
XYdef=zeros(size(XY));
fac=500;
c=0;
for i=1:nod
c=c+1;
XYdef(i,1)=XY(i,1)+fac*D_total(c)
c=c+1;
XYdef(i,2)=XY(i,2)+fac*D_total(c)
end
IJ=E(:,3:4)
for e=1:mbr
Q=[XY(IJ(e,1),1) XY(IJ(e,1),2);...
XY(IJ(e,2),1) XY(IJ(e,2),2)];
Qdef=[XYdef(IJ(e,1),1) XYdef(IJ(e,1),2);...
XYdef(IJ(e,2),1) XYdef(IJ(e,2),2)];
plot(Q(:,1),Q(:,2),'--k',Qdef(:,1),Qdef(:,2),'-b','linewidth',1.5)
hold on
end
Tam=size(B);
for i=1:Tam(1,1)
if B(i,2:3)==[ 1 0 ];
hold on
fij1=[ x(B(i,1))-0.05  x(B(i,1))+0];
fij2=[ y(B(i,1))-0.05  y(B(i,1))+0];
plot(fij1,fij2)
hold on
fij3=[ x(B(i,1))+0 x(B(i,1))+0.05 ];
fij4=[ y(B(i,1))+0 y(B(i,1))-0.05 ];
plot(fij3,fij4)
hold on
fij5=[ x(B(i,1))-0.07 x(B(i,1))+0.07 ];
fij6=[ y(B(i,1))-0.05 y(B(i,1))-0.05 ];
plot(fij5,fij6)
hold on
fij7=[ x(B(i,1))-0.07 x(B(i,1))+0.07 ];
fij8=[ y(B(i,1))-0.08 y(B(i,1))-0.08 ];
plot(fij7,fij8)
elseif B(i,2:3)==[ 0 0 ]
 hold on
fij1=[ x(B(i,1))-0.05  x(B(i,1))+0];
fij2=[ y(B(i,1))-0.05  y(B(i,1))+0];
plot(fij1,fij2)
hold on
fij3=[ x(B(i,1))+0 x(B(i,1))+0.05 ];
fij4=[ y(B(i,1))+0 y(B(i,1))-0.05 ];
```

```
plot(fij3,fij4)
hold on
fij5=[ x(B(i,1))-0.07 x(B(i,1))+0.07 ];
fij6=[ y(B(i,1))-0.05 y(B(i,1))-0.05 ];
plot(fij5,fij6)
hold on
fij7=[ x(B(i,1))-0.07 x(B(i,1))-0.04 ];
fij8=[ y(B(i,1))-0.05 y(B(i,1))-0.08 ];
plot(fij7,fij8)
hold on
fij9=[ x(B(i,1))-0.046667 x(B(i,1))-0.016667];
plot(fij9,fij8)
hold on
fij10=[ x(B(i,1))-0.023334 x(B(i,1))+0.00666];
plot(fij10,fij8)
hold on
fij11=[ x(B(i,1))+0 x(B(i,1))+0.029999];
plot(fij11,fij8)
hold on
fij12=[x(B(i,1))+0.023333 x(B(i,1))+0.053332];
plot(fij12,fij8)
hold on
fij13=[x(B(i,1))+0.046666 x(B(i,1))+0.076662];
plot(fij13,fij8)
hold on
fij14=[x(B(i,1))+0.069999 x(B(i,1))+0.099995];
plot(fij14,fij8)
end
end
xlabel('Longitud[m]');
ylabel('Altura[m]');
title('ESTRUCTURA')
axis equal
```

4. CAPITULO IV ANÁLISIS DE RESULTADOS

En este capítulo se muestran los resultados obtenidos una vez terminado el software se procede a su validación con un software de uso comercial, en este caso se utilizara el Sap2000 v14.0.0 para apreciar en qué medida los datos analizados concuerdan con dicho software.

4.1. Viga



Se analizara la siguiente viga que tiene los siguientes datos:

Modulo de elasticidad para todas las barras

E=2,1E10 Kg/m2

Seccion

a=0.3m

b=0.3m

L1=2m; L2=3m; L3=4m

I=0.000675m4

De la imagen se deducen las coordenadas y se enumeran las barras.

Barra 1: Carga puntual de 15kg ubicada a 0.5m

Barra 2: Carga distribuida de 2kg/m

Barra3: Carga triangular de 2kg/m

Una vez ingresados los datos y analizada la estructura en función al manual de Usuario(Anexo1) se procede a la comparación de Resultados.

K_reducida		
	3150	945
	945	2835

Figura 4.1 Matriz de Rigidez Reducida

Una ventaja del programa M.E.F. es que se pueden visualizar paso a paso todo el procedimiento de Calculo.

4.1.1. Reacciones Verticales

Nodo	Sap2000(Kg)	M.E.F.(Kg)	Comparación(%)
1	**8,63**	**8,63148**	**100,02**
2	**19,87**	**19,8657**	**99,98**
3	**1,44**	**1,44063**	**100,04**
4	**3,06**	**3,0621**	**100,07**

Tabla 4.1 Reaacciones Verticales

4.1.2. Momentos

Nodo	Sap2000(Kg*m)	M.E.F.(Kg*m)	Comparación (%)
1	**6,76**	**6,7565**	**99,95**
2	**0**	**0**	**100**
3	**0**	**0**	**100**
4	**-1,95**	**-1,9495**	**99,98**

Tabla 4.2 Momentos Flectores

4.1.3. Cortantes

Barra1	Sap2000(Kg)	M.E.F.(Kg)	Comparación(%)
1	8,6315	8,63148148	99,9997855
2	-6,3685	-6,36851852	100,000291
Barra2			
2	3,4972	3,49722222	100,000635
3	-0,5028	-0,50277778	99,9955803
Barra3			
3	0,9379	0,93784722	99,9943728
4	-3,0621	-3,06215278	100,001724

Tabla 4.3 Esfuerzos Cortantes

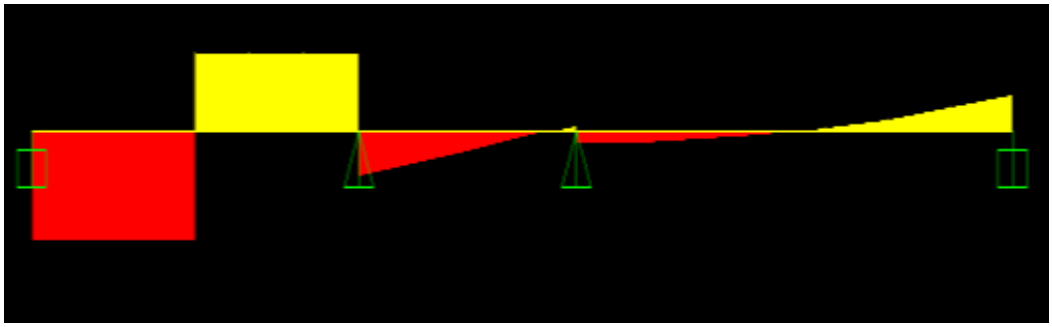


Figura 4.2 Diagrama de Esfuerzos Cortantes en Sap2000

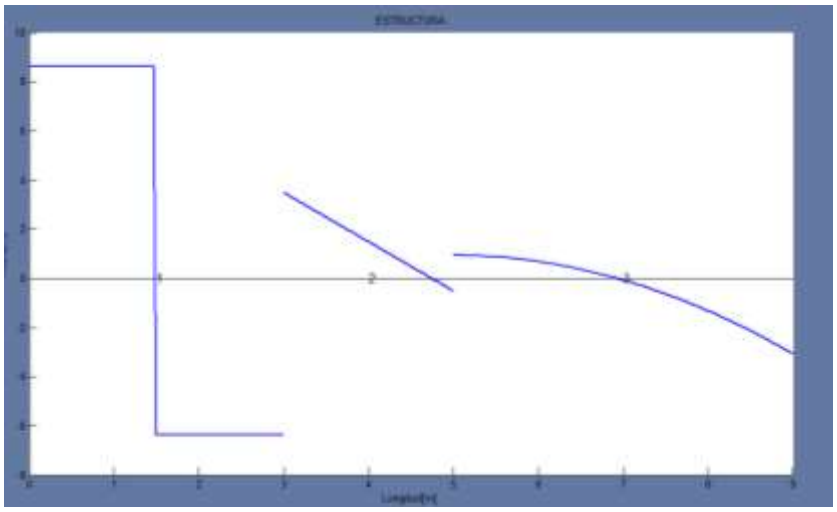


Figura 4.3 Diagrama de Esfuerzos Cortantes en M.E.F.

Se puede observar que el programa Sap2000 toma los cortantes con el signo cambiado es por esto que su grafica no corresponde a la que debería de ser tomada como real para la estructura analizada.

4.1.4. Momentos

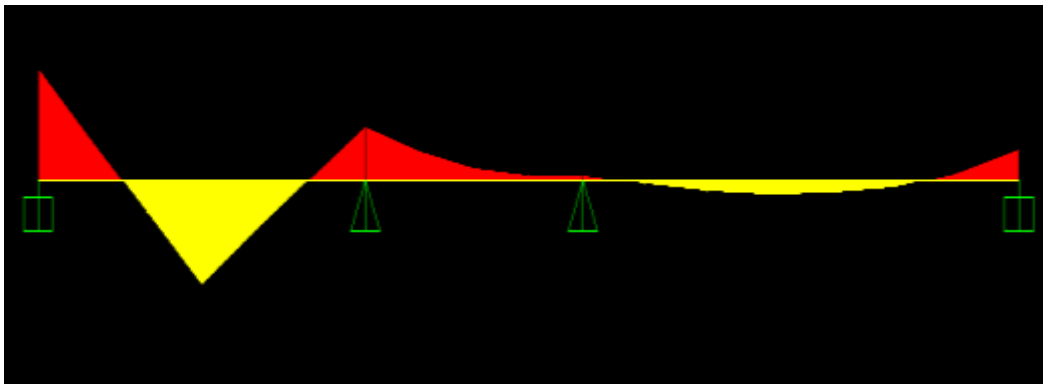


Figura 4.4 Diagrama de Momentos Sap2000

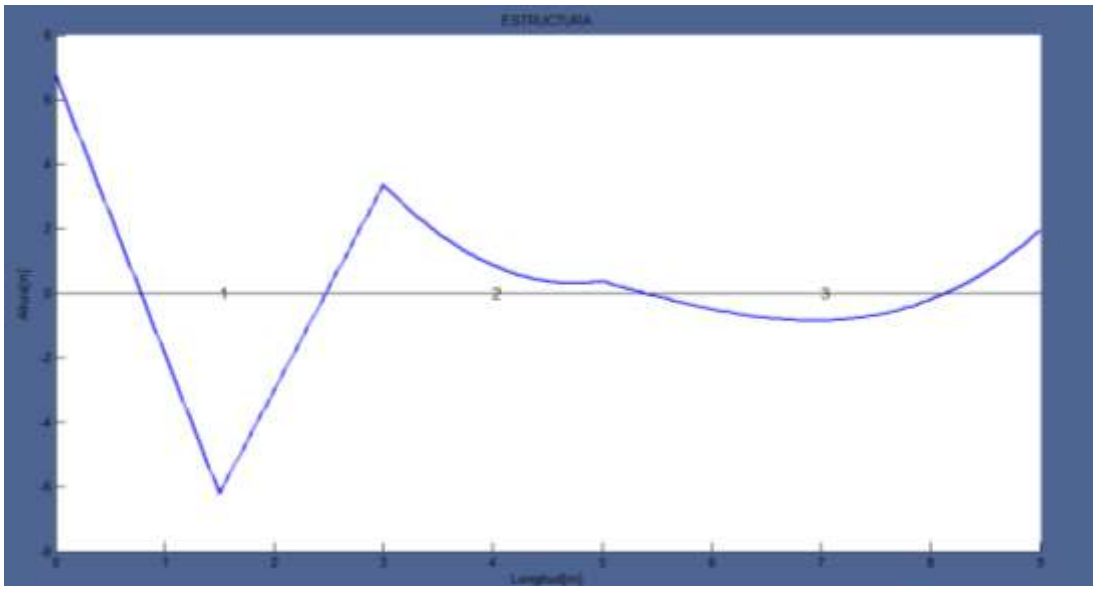


Figura 4.5 Diagrama de Momentos M.E.F.

Barra1	Sap2000(Kg)	M.E.F.(Kg)	Comparación (%)
1	6,76	6,7565	100,02
2	-3,36203	-3,36204	100,01
Barra2			
2	3,36203	3,36204	100,001
3	0,3676	0,36759259	99,998
Barra3			
3	0,36761	0,36759259	99,995
4	1,9495	1,94953704	100,002

Tabla 4.4 Momentos Flectores

4.2. Pórticos

Se analizara la siguiente viga que tiene los siguientes datos:

Modulo de elasticidad para todas las barras

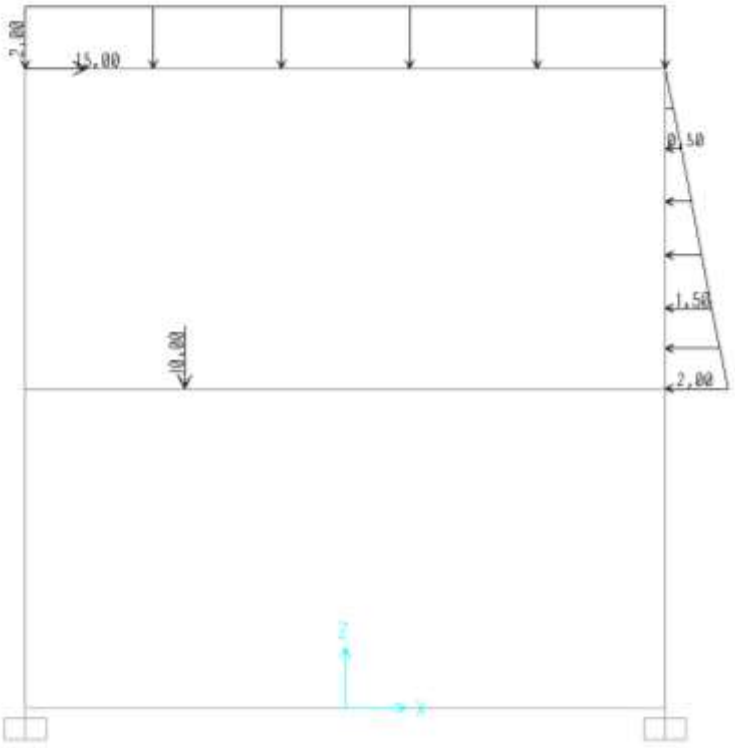
E=2,1E10 Kg/m2

Seccion

a=0.3m

b=0.3m

Cada piso es de 1m y el largo en el eje x es de 2m.



4.2.1. Reacciones Verticales

Nodo	Sap2000(Kg)	M.E.F.(Kg)	Comparación(%)
1	0,69	0,60203529	87,2514918
6	13,31	13,3979647	100,660892

Tabla 4.5 Reacciones Verticales

4.2.2. Reacciones Horizontales

Nodo	Sap2000(Kg)	M.E.F.(Kg)	Comparación(%)
1	-5,77	-5,63244325	97,6160008
6	-8,23	-8,36755675	101,671406

Tabla 4.6 Reacciones horizontales portico

4.2.3. Momentos

Nodo	Sap2000(Kg*m)	M.E.F.(Kg*m)	Comparación(%)
1	5,17	4,98583416	96,437798
6	-5,86	-5,8849031	100,424968

Tabla 4.7 Momentos

4.2.4. Cortantes

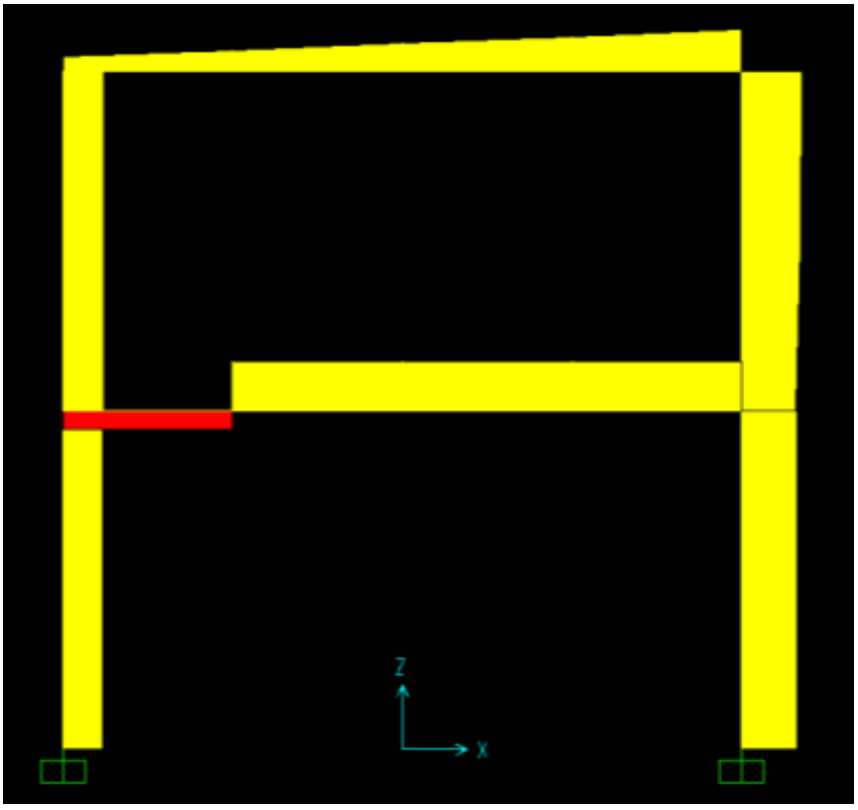


Figura4.6 Diagrama de Cortantes Sap2000

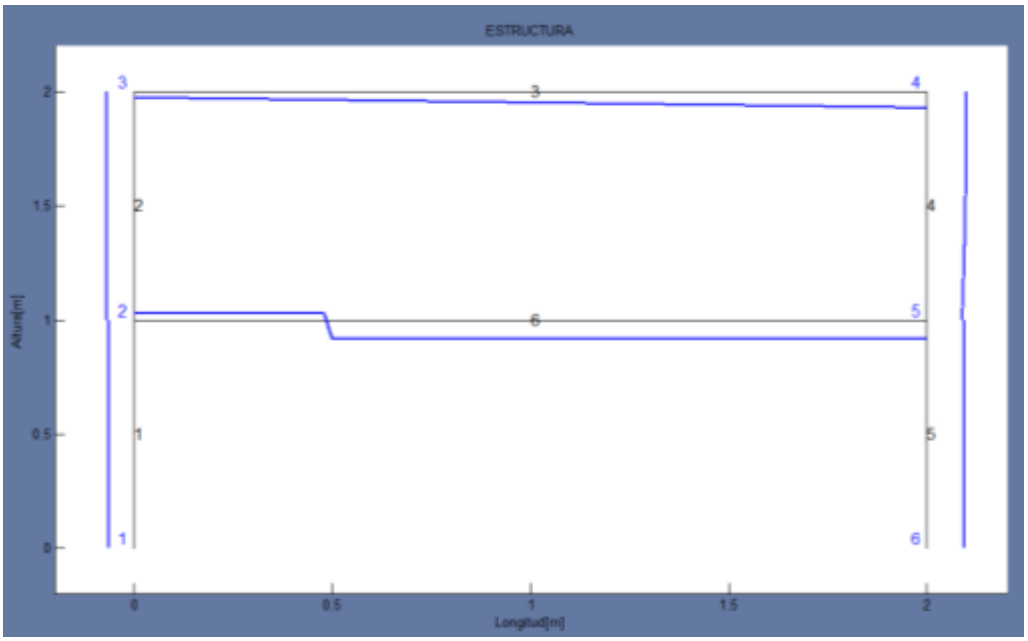


Figura 4.7 Diagrama de Cortantes M.E.F.

Barra1	Sap2000(Kg)	M.E.F.(Kg)	Comparación(%)
1	5,7665	5,63244325	97,6752492
2	5,7665	5,63244325	97,6752492
Barra2			
2	6,0766	6,05362863	99,6219701
3	6,0766	6,05362863	99,6219701
Barra3			
3	-2,1635	-2,17331303	100,453572
4	-6,1635	-6,17331303	100,159212
Barra4			
4	7,9234	8,94637137	112,910763
5	8,9234	7,94637137	89,0509376
Barra5			

5	**8,2335**	**8,36755675**	**101,628187**
6	**8,2335**	**8,36755675**	**101,628187**
Barra6			
2	**2,8491**	**2,77534833**	**97,4114045**
5	**-7,1509**	**-7,22465167**	**101,031362**

Tabla 4.8 Esfuerzos Cortantes portico

4.2.5. Momentos Flectores

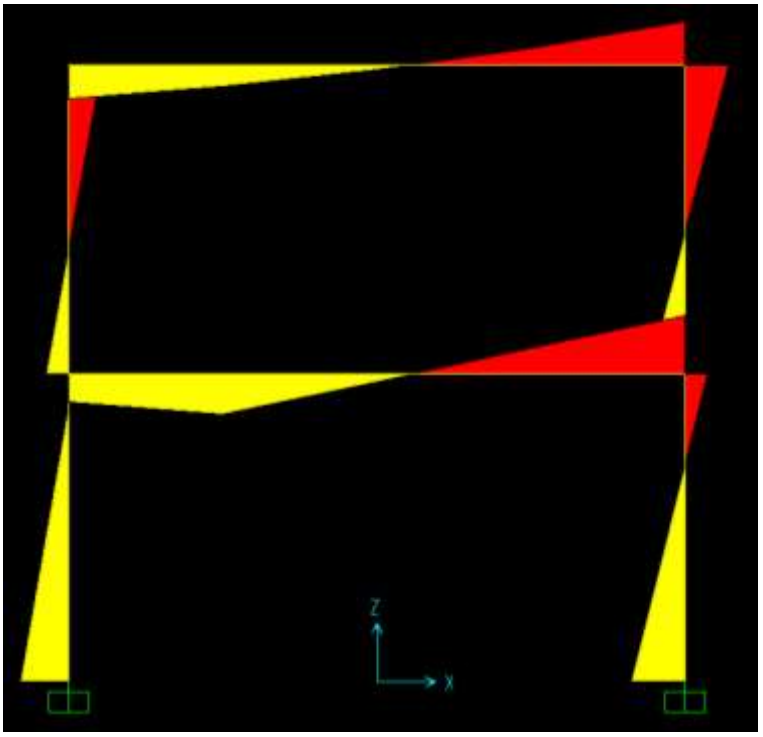


Figura 4.8 Diagrama de Momentos Flectores Sap 2000

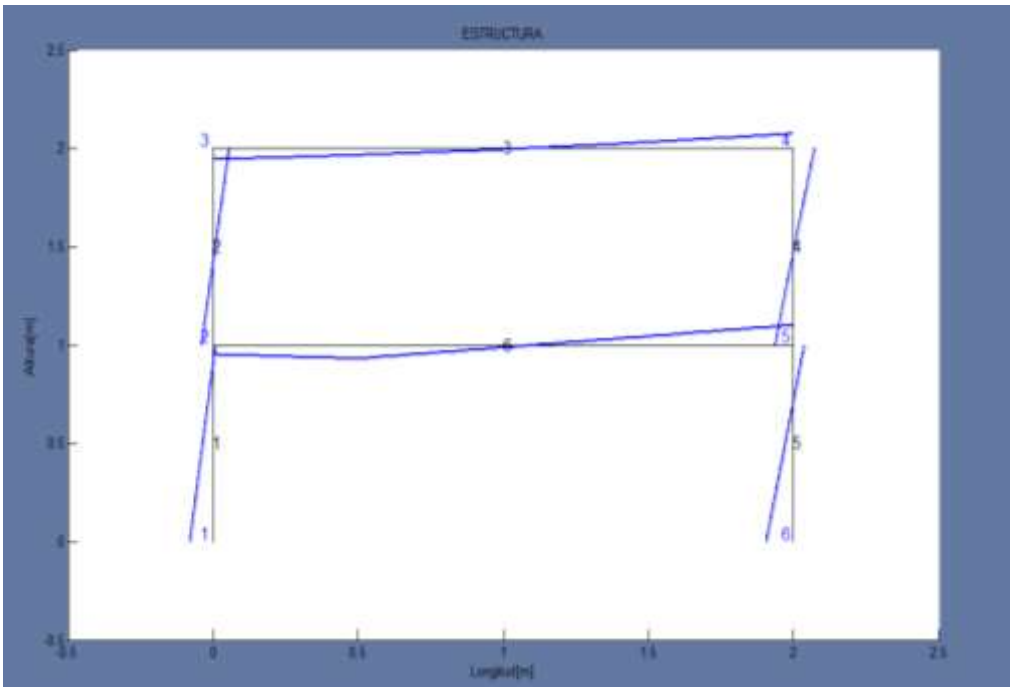


Figura 4.9 Diagrama de Momentos Flectores M.E.F

Barra1	Sap2000(Kg)	M.E.F.(Kg)	Comparación(%)
1	5,1618	4,98583416	96,5909984
2	-0,5963	-0,64660909	108,436876
Barra2			
2	2,4475	2,43728649	99,5826962
3	-3,6291	-3,61634214	99,6484568
Barra3			
3	3,6291	3,61634214	99,6484568
4	4,6979	4,73028392	100,689328
Barra4			
4	4,6978	4,73028392	100,691471
5	-3,8922	-3,88275411	99,7573124

Barra5			
5	**2,2657**	**2,48265366**	**109,575569**
6	**-5,8675**	**-5,8849031**	**100,296602**
Barra6			
2	**-3,044**	**-3,08389558**	**101,31063**
5	**6,2579**	**6,36540777**	**101,717953**

Tabla 4.9 Momentos flectores portico

4.2.6. Normales

Barra1	Sap2000(Kg)	M.E.F.(Kg)	Comparación(%)
1	**0,6856**	**0,60203529**	**87,8114488**
2	**-0,5963**	**-0,64660909**	**108,436876**
Barra2			
2	**2,4475**	**2,43728649**	**99,5826962**
3	**-3,6291**	**-3,61634214**	**99,6484568**
Barra3			
3	**8,9234**	**8,94637137**	**100,257428**
4	**8,9234**	**8,94637137**	**100,257428**
Barra4			
4	**6,1635**	**6,17331303**	**100,159212**
5	**6,1635**	**6,17331303**	**100,159212**
Barra5			
5	**13,3144**	**13,3979647**	**100,627627**
6	**13,3144**	**13,3979647**	**100,627627**

Barra6			
2	**0,3101**	**0,42118539**	**135,82244**
5	**0,3101**	**0,42118539**	**135,82244**

Tabla 4.10 Esfuerzos Normales portico

4.3. Armaduras

Se analizara la siguiente viga que tiene los siguientes datos:

Módulo de elasticidad para todas las barras:

E=2,1E10 Kg/m2

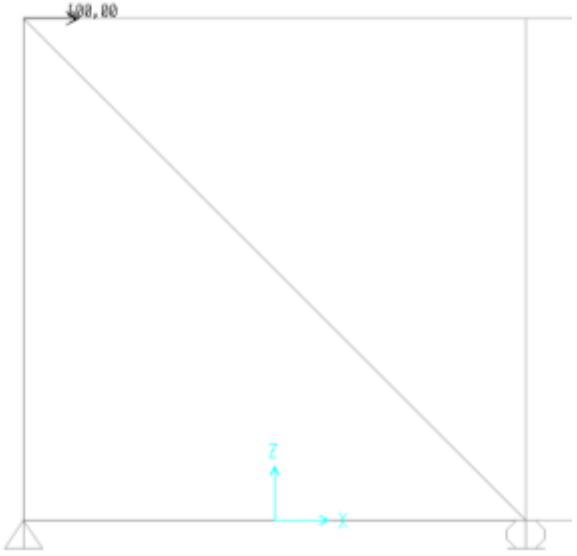
Sección

a=0.5m

b=0.25m

L=2m

A=0.09m2



4.3.1. Reacciones Verticales

Nodo	Sap2000(Kg)	M.E.F.(Kg)	Comparación (%)
1	-500	-500	100
3	500	500	100

Tabla 4.11 Reacciones verticales armadura

4.3.2. Reacciones Horizontales

Nodo	Sap2000(Kg)	M.E.F.(Kg)	Comparación (%)
1	-500	-500	100
6	0	0	100

Tabla 4.11 Reacciones horizontales armadura

4.3.3. Esfuerzos Axiales

Barra1	Sap2000(Kg)	M.E.F.(Kg)	Comparación(%)
1	-500	-500	100
2	-500	-500	100
Barra2			
2	707,106	707,106781	100
3	707,106	707,106781	100
Barra3			
3	-500	-500	100
4	-500	-500	100

Tabla 4.12 Esfuerzos Axiales

4.3.4. Esfuerzos Transversales

Barra1	Sap2000(Kg)	M.E.F.(Kg)	Comparación(%)
1	11,43	0	100
2	11,43	0	100
Barra2			
2	0	0	100
3	0	0	100
Barra3			
3	-11,4733	0	100
1	-11,4733	0	100

Tabla 4.13 Esfuerzos Tranversales

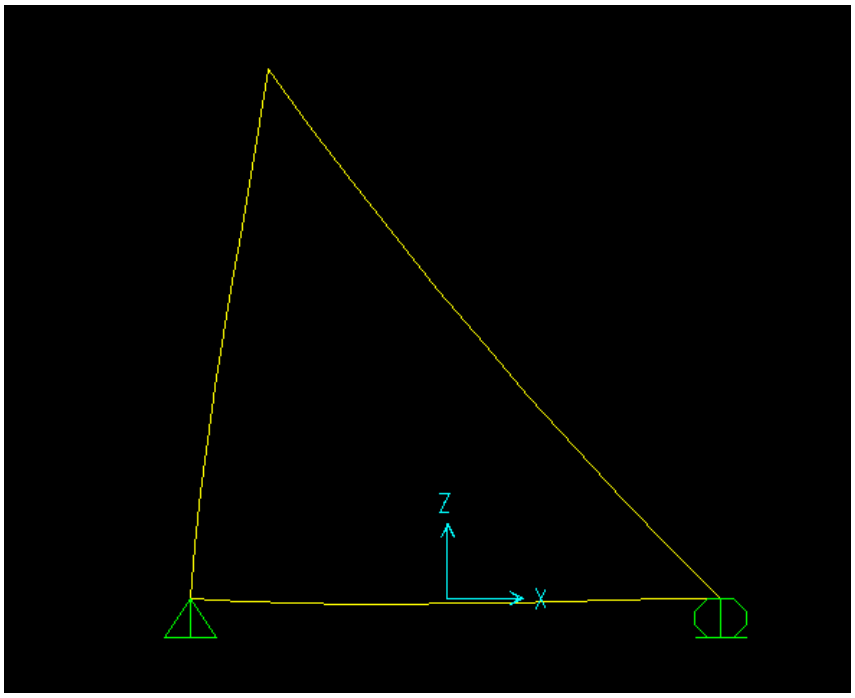


Figura 4.10 Deformada de la estructura en Sap2000

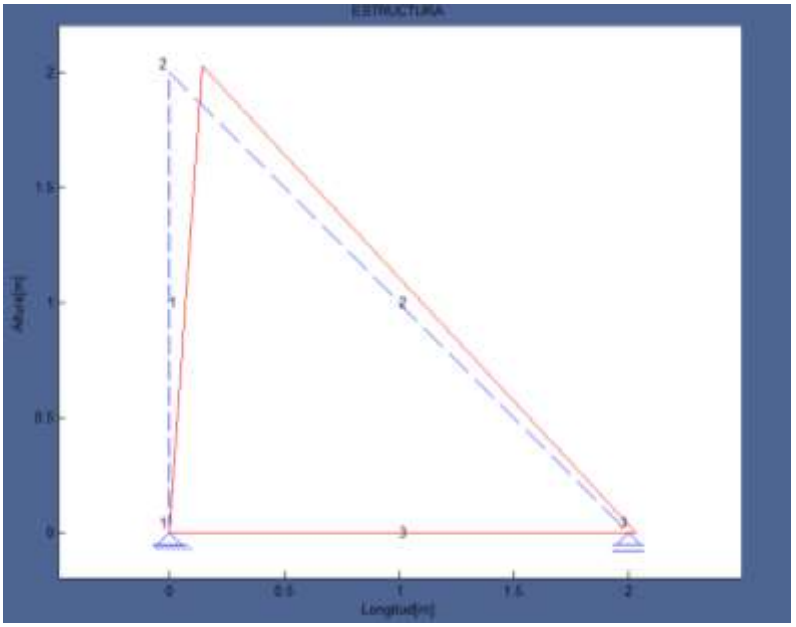


Figura 4.11 Deformada de la Estructura en M.E.F.

4.4. Contrastación de hipótesis.

La hipótesis fundamental, así como también las premisas manejadas para poder llegar a la simplificación del programa fueron concluidas en un software computarizado que nos devuelve resultados bastante similares a los del software comercial Sap2000, pudiendo comprobar que el método de los elementos finitos es un método confiable y con mayor efectividad en su programación.

CONCLUSIONES Y RECOMENDACIONES.

Conclusiones

- Se pudo realizar el software con resultados de entre 99% y 102% comparado con el Sap2000 por lo tanto se puede concluir que los programas comerciales trabajan con el método de elementos finitos y nuestra hipótesis ha sido resuelta como cierta.
- El programa Sap2000 tiene variaciones que viéndolas en porcentajes son relativamente pequeñas.
- Las mismas estructuras siendo analizadas en otros programas como el programa Larch y el programa VigaG que son programados por elementos finitos concuerdan con el programa M.E.F. en un 100% dejando un 0% de porcentaje de error.
- El algoritmo más difícil de programar fue el de ensamblaje de la matriz de rigidez, puesto que se pasaron varios días hasta lograr llegar a la solución óptima y utilizando el menor tamaño de algoritmo para agilizar el cálculo dentro de MATLAB.
- El código que se brindó en el capítulo 3 es el código más óptimo que se pudo programar para llegar a los resultados tan concordantes como se pudo ver en el capítulo 4.
- Se presentaron todos los cálculos realizados en una hoja de Excel en donde se muestra paso a paso el procedimiento mediante elementos finitos que se utilizó para llegar a resultados tan exactos y confiables.
- Se logró cumplir con el objetivo de brindar un software intuitivo con una interfaz gráfica amigable que permite poder aprender rápidamente la forma de ingreso de datos para así entender más rápidamente el funcionamiento del método de Elementos Finitos.
- Se deja en la presente tesis todo el código de programación utilizado para la realización del programa M.E.F. abiertamente para que futuros tesistas puedan

mejorar o implementar nuevos tipos de estructuras como en su caso podría ser el de Placas planas, Arcos, etc.

- A medida que se van realizando cálculos en el código de programación, se va escribiendo el archivo Excel donde puede ver toda la información anteriormente mencionada.
- La velocidad de cálculo de lenguaje Matlab en comparación con lenguajes más potentes como C+; C++, java o phyton en relativamente lenta, debido a que estos lenguajes realizan el cálculo considerablemente más rápido.
- Con el desarrollo de este proyecto se aporta al banco de software de análisis de elementos finitos que viene tomando forma en los últimos semestres mediante el desarrollo de otros problemas específicos de resistencia de materiales, para en un futuro proyecto reunir todos los trabajos y crear un software que involucre una solución numerosa de problemas y brindar así un buen uso educativo, e impulsar mediante el estudio de casos unidimensionales y bidimensionales el estudio amplio y completo para tres dimensiones del método de elementos finitos.
- Para poder recabar toda la información necesaria para proceder al proceso de cálculo de la estructura y posteriormente para sus graficas se realizaron una serie de alrededor de 20 sub rutinas o sub programas que facilitaron posteriormente el proceso de cálculo.
- Para el caso de realización de los cálculos dentro del programa para la presentación de graficas se recurrió al método de la doble integración pudiendo llegar a resultados similares al del Sap2000.
- Se logro aprovechar el curso de Manejo y Programacion en calculadoras HP50g para poder incentivar a los estudiantes a el estudio del método de los elementos finitos mediante la realización de programas que ayudan a facilitar el calculo de estructuras utilizando este método con la ayuda de la calculadora Hp50g, asimismo se pudo darles una breve descripción del lenguaje de programación

MatLab logrando que los alumnos muestren cierto interés en la aplicación que puede tener este poderoso lenguaje de programación.

- El software computacional M.E.F. dio resultados mas exactos con programas otros programas como el VigaG y Larch:

➢ **Vigas**

Reacciones Verticales

Nodo	VigaG (Kg)	M.E.F. (Kg)	Comparación (%)
1	**8,63148**	**8,63148**	**100**
2	**19,8657**	**19,8657**	**100**
3	**1,44063**	**1,44063**	**100**
4	**3,0621**	**3,0621**	**100**

Momentos

Nodo	VigaG (Kg*m)	M.E.F. (Kg*m)	Comparación (%)
1	**6,7665**	**6,7565**	**100**
2	**0**	**0**	**100**
3	**0**	**0**	**100**
4	**-1,9495**	**-1,9495**	**100**

➢ **Porticos**

Reacciones Verticales

Nodo	Larch (Kg)	M.E.F. (Kg)	Comparación (%)
1	**0,602**	**0,60203529**	**100**
6	**13,398**	**13,3979647**	**100**

Reacciones Horizontales

Nodo	Larch (Kg)	M.E.F.(Kg)	Comparación(%)
1	-5,632	-5,63244325	100
6	-8,368	-8,36755675	100

Momentos

Nodo	Larch (Kg*m)	M.E.F.(Kg*m)	Comparación(%)
1	4.986	4,98583416	100
6	5,885	5,8849031	100

- **Armadura**

Reacciones Verticales

Nodo	Larch(Kg)	M.E.F.(Kg)	Comparación (%)
1	-500	-500	100
3	500	500	100

Reacciones Horizontales

Nodo	Larch(Kg)	M.E.F.(Kg)	Comparación (%)
1	-500	-500	100
3	0	0	100

Recomendaciones

- Se debe incentivar a la carrera en la creación de softwares cada vez más complejos y no esperar a que se desarrollen softwares de terceros que bien podrían ser desarrollados por estudiantes con la ayuda de docentes de nuestra superior casa de estudios U.A.J.M.S.
- El estudiante tanto así como el profesional deben entender a cabalidad cómo funcionan los programas que se están utilizando como por ejemplo es Sap2000 que es también un software basado en Elementos Finitos.
- Se da pie a que las investigaciones con elementos finitos sigan adelante ya que es una gran herramienta para poder calcular problemas complejos en la ingeniería.
- Se recomienda el uso masivo del lenguaje de programación como Matlab para alumnos Universitarios porque nos da grandes alternativas de desarrollos para el cálculo de Ingeniería.
- Como actividades de incentivo se deberian realizar pequeños talleres, cursos y ponencias donde se muestre la importancia de este metodo en la actualidad, asimismo se deberia profundizar más en el desarrollo de sofware mediante este método para resolver problemas academicos, esto con ayuda de las autoridades para que puedan financiar dichos cursos y ponencias.
- Para el cálculo de pórticos, vigas y armaduras, se recomienda usar el sistema de acoplamiento de matriz mediante el método del vector de colocación, este método puede ser usado para todo tipo de estructura ya que es un algoritmo universal ya que se establece la opción de dirección y grado de libertad.

BIBLIOGRAFIA

MOORE, Holly “Matlab para Ingenieros” 1era edición

ALVAREZ MARIN, Diego Andrés “Repaso de análisis estructural matricial Universidad de Colombia sede Manizales”

MCCORMAC, Nelson “Análisis de estructuras, método clásico y método matricial” 3era edición

CELIGÜETA LIZARZA, Juan Tomas “Método de los Elementos Finitos para Análisis Estructural”3era edición

GARCIA, Javier de jalón 2004”Aprenda Matlab como si estuviera en primero”

H. PARTL: O.CZIENKIEWICHZ-R.LTAYLOR, El Método de Elementos Finitos, McGraw-Hill,(1998)

CÁZARES SERGIO GALLEGOS, Análisis de Sólidos y Estructural mediante el método de Elementos finitos,Tecnológico de Monterrey-LIMUSA,(2008)

O.CZIENKIEWICHZ-R.LTAYLOR, El Método de Elementos Finitos,McGraw-Hill,(1998)

SERGIO GALLEGOS CÁZARES, Análisis de Sólidos y Estructural mediante el método de Elementos finitos,Tecnológico de Monterrey-LIMUSA,(2008)

MANUEL VASQUEZ,ELOIZA LOPEZ, El Método de Elementos Finitos Aplicado al análisis Estructural.

ROBERTO AGUIAR FALCONÍ, Analisis Matricial de Estructuras.

NELSON MCCORMAC, Analisis de estructuras, Metodo Clasico y Matricial

ÍNDICE

CAPITULO I

INTRODUCCION **Página**

CONCLUSIONES Y RECOMENDACIONES

BIBLIOGRAFIA