PONTIFICA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

ANÁLISIS APLICATIVO DEL MÉTODO DE LOS ELEMENTOS FINITOS EN UN CAMPO ESTÁTICO-LINEAL E INTRODUCCIÓN A LA NO LINEALIDAD

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGÍSTER EN INGENIERÍA CIVIL CON MENCIÓN EN ESTRUCTURAS SISMORRESISTENTES

AUTOR

Jorge Enrique Alvarez Ruffrán

ASESOR

Mag. José Alberto Acero Martínez

Lima, Mayo del 2020

DEDICATORIA:

A mi familia quienes por ellos soy lo que soy. Para mis padres Jorge y Carmen, por su apoyo, consejos y comprensión. Me han dado todo lo que soy como persona, mis valores, mis principios, mi carácter, mi empeño, mi perseverancia, mi coraje para conseguir mis objetivos, a ellos todo mi amor y gratitud por apoyarme en todo momento.
A mi hermano Erick André, por estar siempre presente, quien me ayudo en todas las etapas de mi formación, por ser mi motivación, inspiración y felicidad.

Agradecimientos

A la Pontificia Universidad Católica del Perú, porque en sus aulas recibimos el conocimiento intelectual y humano de cada uno de los docentes que la integran, teniendo la seguridad de que seguirán formándose dignos profesionales que dejarán en alto el nombre de nuestra Alma Mater.

Especial agradecimiento a mi asesor José Acero Martínez por su esfuerzo, dedicación, conocimientos y motivación, ellos han sido fundamentales para la elaboración de la presente Tesis.

A mis maestros, por su amistad y apoyo, así como la sabiduría que me transmitieron en el desarrollo de mi formación.

Resumen

La presente investigación tuvo como tema establecer fundamentos para el análisis aplicativo del método de los elementos finitos, orientado al análisis estructural bajo rango estático lineal y dar una breve introducción de su aplicación en un campo no lineal. Teniendo en cuenta que para la aplicación eficiente del método es necesario el uso de ordenadores, se desarrollaron diversos algoritmos de programación (diagramas de flujo). Acompañado a dichos algoritmos se dejan un conjunto de códigos base y una guía para la optimización de los mismos; como objetivo general se tiene formular una guía teórica-aplicativa detallada y conjunta para la aplicación de diversos tipos de elementos finitos usados en el análisis de sistemas estructurales. A partir de la base dejada en la presente tesis se espera que investigadores y/o profesionales que tengan interés en el tema puedan aplicar la metodología en otros campos como es el dinámico, elaborando programas o códigos más complejos. Al culminar la investigación algunas de las conclusiones a destacar son: La optimización de un código basado en elementos finitos juega un rol de vital importancia debido al gran número de bucles que estos conllevan. Gran parte de las publicaciones sobre el método de los elementos finitos que orienta su uso aplicativo mediante códigos, se olvidan del ciclo de desarrollo común del mismo, lo cual conlleva a una falsa complejidad del método. El uso de diagramas de flujo para interrelacionar la parte aplicativa con la teórica mejora considerablemente la comprensión que existe en el mismo.

Palabras Clave: Algoritmos, análisis estructural, códigos, elementos finitos, optimización, programación.

Índice de Contenidos

Pág.

Índice de Tablas

Índice de Figuras

Índice de Anexos

CAPÍTULO 1

Introducción

CAPÍTULO I: INTRODUCCIÓN

1.1. MOTIVACIÓN

Durante el transcurso del tiempo, conjunto al avance tecnológico se han visto implementadas nuevas tecnologías en las diversas áreas de desarrollo humano, siendo la ingeniería estructural no ajena a estos avances se cuenta con nuevas formas de analizar una estructura. Una metodología de análisis que ha tomado fuerza desde la implementación de los ordenadores son los elementos finitos. Este método permite dar solución a un gran número de problemas complejos, razón por la cual es uno de los métodos más estudiados en ingeniería y desarrollado en diversos programas de ordenador, siendo estos últimos el medio único y fundamental para su uso a un nivel profesional.

A pesar de que el método es ampliamente usado en diversos campos de la ciencia e ingeniería y que existen un gran número de publicaciones del mismo, la información existente sobre el tema está enfocada principalmente desde dos puntos de vista. El primer enfoque es netamente teórico con poco énfasis en la parte aplicativa (uso de ordenador), abarcándola como una mera inclusión de código en un capítulo de sus publicaciones. El segundo enfoque es fundamentalmente aplicativo mediante el uso y aplicación de códigos, en algunos casos se da una breve introducción teórica con la inclusión de fórmulas. Desde un punto de vista educativo y de programación, estos enfoques carecen de un desarrollo teórico-aplicativo conjunto. Por tal motivo cuando un profesional recién introducido al tema desea aplicar el método tiende a presentar dificultades en la comprensión del mismo, creando una falsa complejidad en su desarrollo.

Por los motivos antes expuestos la presente investigación tiene como objetivo brindar al profesional una guía detallada para la aplicación del método de elementos finitos en el análisis de sistemas estructurales con un enfoque teórico-aplicativo conjunto. La guía consistirá en un marco de referencia con las bases fundamentales del método, códigos base para los diversos tipos de elementos finitos, herramientas de visualización de resultados, algoritmos de programación y pautas a tener en cuenta en la optimización del código con respecto al procesamiento de datos.

Teniendo fija la meta de esta investigación, se espera que profesionales e investigadores, que deseen estudiar el método de los elementos finitos, puedan

comprender de forma sencilla la aplicación del método. Se espera que esta tesis sirva como base para futuras investigaciones que elaboren códigos más complejos, abarcando incluso un campo dinámico para los distintos sistemas estructurales que son objeto de análisis en nuestra profesión.

1.2. OBJETIVOS

1.2.1. Objetivo General

Formular una guía detallada respecto a la aplicación del método de los elementos finitos para varios tipos de elementos usados en el análisis de sistemas estructurales en un campo estático.

1.2.2. Objetivos Específicos

- Realizar algoritmos de programación orientados a los tipos de elementos finitos tratados.
- Elaborar códigos base a manera de apoyo y guía, junto a ejemplos aplicativos basados en el análisis estructural en un campo estático.
- Establecer pautas que deban tenerse en cuenta para la elaboración de un código en elementos finitos a nivel profesional referidas a la optimización del mismo.
- Brindar conceptos básicos sobre el método de los elementos finitos e interrelacionarlos directamente con su aplicación en códigos y algoritmos.

1.3. METODOLOGÍA

Para elaborar una guía detallada en la aplicación del método de los elementos finitos, la presente investigación se dividió en 13 etapas de desarrollo. Dichas etapas combinan la integración de los conceptos básicos del método con su aplicación y optimización mediante códigos y algoritmos.

La primera etapa está referida a la recolección de información de diversas fuentes bibliográficas en donde se analizaron: Las bases del método de los elementos finitos y la metodología de enseñanza de diversos autores.

La segunda etapa corresponde al estudio de los conceptos fundamentales y aplicación mediante ordenador del método, en esta etapa se establecieron fundamentos de la parte teórica.

La tercera etapa consiste en el estudio y procesamiento de información sobre los diversos tipos de elementos finitos, en donde se establecieron las hipótesis base para cada tipo de elemento trabajado.

La cuarta etapa contempla la elaboración de algoritmos de programación los cuales se complementarán y servirán como base para la elaboración del código.

La quinta etapa presenta la elaboración del código base, interrelacionando el mismo con los algoritmos de programación.

La sexta etapa se centra en el pos-procesamiento de datos, contemplando las herramientas que se utilizaron para la visualización de resultados.

La séptima etapa contempla la verificación del código base realizando ejemplos aplicativos y comparando los resultados con los obtenidos por autores y programas orientados al método.

La octava etapa consiste en el estudio de técnicas de optimización del código, en donde se establecerán las pautas para la optimización de un código con respecto al tiempo que conlleva el procesamiento de datos.

La novena etapa presenta la aplicación de las técnicas de optimización, realizando varias iteraciones de optimización en el código, analizando la reducción en el tiempo de ejecución del mismo.

La décima y onceava etapa está referida al mejoramiento en la integración de los algoritmos de programación con el código base, mejora en la presentación de las herramientas de optimización e inclusión del código sobre los diversos ejemplos aplicativos elaborados.

La doceava y última etapa es la elaboración del informe final, en la cual se procede a culminar con la redacción para su respectiva emisión, revisión y levantamiento de observaciones del asesor y jurado correspondiente.

1.4. ORGANIZACIÓN DE LA TESIS

La tesis está organizada en diez capítulos de la siguiente forma:

Capítulo 1: Presenta la introducción del trabajo, con respecto a la motivación, objetivos, metodología, así como la organización de la tesis.

Capítulo 2: Muestra un breve marco teórico sobre autores que incursionaron en el tema, fundamentos básicos del método, definición de algoritmo y el tipo de software usado.

Capítulo 3: Expone los tipos de elementos finitos trabajados, explicando las características más relevantes en la formulación de los mismos. Se tocan temas como son las hipótesis fundamentales que los componen, campo de desplazamientos, deformaciones y tensiones.

Capítulo 4: Presenta de forma aplicativa cómo funciona el método, explica cada una de las matrices que lo componen y toma como ejemplo un caso sencillo, como es el axialmente deformable, para su fácil entendimiento.

Capítulo 5: Muestra características importantes para una fácil correlación entre lo aplicativo y lo teórico. Dichas características están comprendidas por: Métodos de representación de un código, el diagrama de flujo (método trabajado), composición de un diagrama de flujo y explicación de la aplicación del mismo a diversas partes del código trabajado (detalla el ciclo de desarrollo de un código).

Capítulo 6: Presenta y define los componentes más importantes de la programación en el software MatLab, como son funciones y subprogramas utilizados. También toca temas como son las técnicas de pos procesamiento de datos (herramientas de visualización de resultados).

Capítulo 7: Expone técnicas de optimización de un código, las cuales son de vital importancia en la aplicación de elementos finitos. Con la aplicación de dichas técnicas, se realiza una comparativa de un código sin optimizar contra uno optimizado (código final de la presente investigación).

Capítulo 8: Se realiza ejemplos para mejorar el entendimiento del método en forma aplicativa y verificar el nivel de precisión en los resultados.

Capítulo 9: Muestra una introducción al FEM en el campo no-lineal, dando a conocer: Sus fundamentos, algoritmo general de aplicación, recomendaciones para su programación y ejemplo de aplicación.

Capítulo 10: Presenta las conclusiones de la presenta tesis y se dan recomendaciones para posibles trabajos futuros a partir de esta investigación.

Anexos: Los anexos incluyen: El código de la presente investigación y los algoritmos para las diversas fases de desarrollo.

CAPÍTULO 2

Revisión bibliográfica: Estado del arte y Marco teórico

Resumen

Se presenta una revisión bibliográfica concisa y expresada en forma de comparativa, sobre autores relacionados directa e indirectamente con el método de los elementos finitos. Se detallan conceptos básicos del método, desde un punto de vista relacionado a la resistencia de materiales y mecánica de cuerpos deformables, para un fácil entendimiento a nivel ingenieril. También se definen herramientas básicas de solución de problemas como son los algoritmos y el tipo de software a usar conjunto al lenguaje de Programación (MatLab).

CAPÍTULO II: ESTADO DEL ARTE Y MARCO TEÓRICO

2.1. ANTECEDENTES

2.1.1. Documentos relacionados directamente al estudio del método

1. Oñate, E. (2013). Structural Analysis with the Finite Element Method. Linear Statics, VOL 1 y 2. Springer, Barcelona.

   Brinda información sobre el análisis de elementos finitos en un rango lineal, tratando diversos tipos de elementos (orientados al análisis estructural). Su publicación abarca fundamentos detallados de cada tipo de elemento, aplicación del método en ordenador mediante códigos y el uso del programa (MAT- fem), dando pautas para el uso del mismo.

2. Zienkiewicz & Taylor, Wisniewski, Zhu & Nithiarasu. (2004). El método de los Elementos Finitos, VOL 1. Las Bases. CIMNE, Barcelona.

   Libro base sobre el método de elementos finitos desde un punto de vista orientado en gran medida al campo matemático. Explica conceptos del método de forma detallada, desarrollando gran parte de las fórmulas que lo conceptualizan. Junto a su publicación se anexa un programa (FEAPpv) para la aplicación del método. Se resalta que el autor es altamente reconocido y uno de los primeros en desarrollar el método.

3. Manuel Vázquez. (2001). El método de los elementos finitos aplicado al análisis estructural. Noela, Madrid.

   La publicación que realiza Manuel Vázquez, está orientada directamente al entendimiento del método desde un punto de vista estructural, efectuando ejemplos de sistemas estructurales a lo largo de todo el texto. Detalla fundamentos base para cada tipo de elemento finito, y llega hasta un análisis dinámico para sistemas simples.

4. Alder Quispe Panca. (2015). Análisis Matricial de Estructuras – Introducción al método de los elementos Finitos. Macro, Perú.

   Su publicación inicia describiendo algunos otros métodos (como es el de rigidez), a partir de estos conceptos matriciales se enfoca en el método de los elementos finitos aplicado a elementos básicos como son el tipo barra y bidimensionales. Realiza ejemplos de estructuras básicas como son pórticos, armaduras, entre otros.

2.1.2. Otros documentos relacionados

5. Bang & Kwon. (2000). The finite Element using MATLAB. A.A. Balkema

   Su publicación está orientada a la aplicación del método incorporando códigos para su uso (sin llegar al pos-procesamiento de resultados). Al comienzo de cada capítulo otorga conceptos básicos del tipo de elemento trabajado, realizando el análisis de diversas estructuras mediante el código. De forma extra incorpora definiciones de diversas herramientas que comprende MatLab.

6. Peter Kattan. (2010). MATLAB Guide to Finite Elements. Springer New York

   Brinda conceptos básicos en el uso de las distintas fórmulas que comprende el método, realiza ejemplos aplicativos de diversos tipos de elementos finitos asignando el código de solución que lo acompañe. El código con el que trabaja el autor se encuentra disperso en varias funciones, las cuales el usuario debe ingresar de forma manual en la ventana de comandos de MatLab.

7. Khennane. A. (2013). Introduction to finite element analysis using MATLAB and Abaqus. CRC Press. New York

   En su publicación brinda conceptos básicos del método, dando a conocer el funcionamiento de los elementos finitos mediante MatLab. También realiza comparativas de resultados entre los obtenidos mediante su código y los que otorga Abaqus (programa de elementos finitos).

2.1.3. Integración Teórica-Aplicativa en las publicaciones

El método de los elementos finitos nace y se formula a lo largo del siglo XX, por lo que cuenta con un gran número de publicaciones sobre el mismo. Las publicaciones abarcan diversos campos de investigación y se orientan tanto en su desarrollo teórico como aplicativo.

En la tabla 1 se visualizan algunas publicaciones realizadas por autores con relevancia en el medio. Se debe resaltar que la tabla está orientada al desarrollo de temas específicos (tratados en la presente tesis), esto para tener una idea del tipo de enfoque que se está dando en la presente investigación mas no tratando de calificar dichas publicaciones.

Tabla 1. **Publicaciones relacionadas al estudio del método**

Fuente: Elaboración Propia

Autores / Publicación	Estudio Teórico		Aplicación del Método			Integración Teórica - Aplicativa
	Detallado	Básico / General	Programa / Ejecutable	Ejemplo aplicativo	Código Detallado	Diagrama de flujo (General)
E. Oñate / Calculo de estructuras por el método de los E. Finitos	✓		✓			✓
O.C. Zienkiewicz – R.L. Taylor / El método de los E. Finitos VOL 1.	✓		✓			
Manuel Vázquez / El método de los E. Finitos aplicado al A. Estructural	✓			✓		
Alder Panca / Análisis matricial de estructuras – Introducción al método de los elementos finitos		✓		✓		
Kwon, Young W. / The finite element method using MatLab		✓		✓	✓	
Khennane. A. / Introduction to FEM using MatLab and Abaqus		✓		✓	✓	
Peter Kattan / MatLab guide to finite elements: an interactive approach		✓		✓	✓	

Con respecto a la tabla anterior, el lector puede darse cuenta que la integración teórica/aplicativa es muy poco tratada. Los autores que se enfocan al estudio teórico (bases del método) tratan la parte aplicativa como la inclusión de un ejecutable en sus publicaciones, esto debido (en muchos casos) a lo amplio del método. Otros autores que tratan la parte aplicativa, se centran en la misma y no resuelven una interacción teórica/aplicativa conjunta, esto llega en gran medida a causar confusión al lector y una falsa complejidad en el estudio del método.

2.2. EL MÉTODO DE LOS ELEMENTOS FINITOS

El método de los elementos finitos (desde ahora llamado FEM / “Finite Element Method”), es un método numérico orientado principalmente a resolver problemas de matemática e ingeniería. Dicho método tiene como objetivo a nivel estructural la resolución de ecuaciones diferenciales parciales para la obtención de desplazamientos, y luego la obtención de esfuerzos y las deformaciones.

En el ámbito Ingenieril, el entendimiento del comportamiento de un sistema complejo presenta grandes limitaciones. Motivo por el cual, la forma común a proseguir es dividir el sistema en componentes individuales, llamados “elementos”, los cuales compartirán conexión con otros a sus alrededores. Dichos elementos podrán ser analizados individualmente para posteriormente proceder a reconstruir (ensamblar) el

sistema original aplicando las condiciones de contorno, logrando finalmente entender el comportamiento del sistema inicial (estructura).

Una gran desventaja del método, es el cálculo computacional que el mismo implica, ya que se debe analizar individualmente cada elemento discreto. Por dicho motivo a inicios del siglo XX cuando no se contaban con los ordenadores, se desarrollaron métodos de análisis aproximados, como el de Cross (1930). Dichos métodos están limitados a estructuras sencillas, lo cual conlleva a la representación de un modelo complejo mediante un sistema equivalente con solución aproximada a la real.

Estos métodos aproximados, parten en muchos casos con las mismas bases del FEM. Un claro ejemplo es el método de rigidez, el cual puede lograr demostrarse con las ecuaciones base con las que parte el FEM.

Se tiene que resaltar que en el presente capítulo se recopilo información de diversas publicaciones realizadas sobre el FEM, a partir de estas se dio un punto de vista fácil de entender a nivel ingenieril, orientando los conceptos a la resistencia de materiales, análisis de estructuras y mecánica de cuerpos deformables. Entre los autores de dichas publicaciones tenemos a E. Oñate (Oñate, 1995), M. Vazquez (Vázquez, 2001), Zienkiewicz (O C Zienkiewicz, Taylor, Wisniewski, Zhu, & Nithiarasu, 2004), K.Young (Bang & Kwon, 2000) y R. Romero (Rubio, G. C., & Romero, 2010).

2.2.1. Inicios del Método

Nace y se forma a lo largo del siglo XX, con respecto a este método se está de acuerdo con muchos otros autores con la premisa de que no tiene un personaje que haya desarrollado la metodología (un padre del método en sí). El método fue formulado gracias al aporte de muchos otros autores y la condensación de un gran número de investigaciones formuladas para obtener las bases de la propia metodología. Para entender el gran número de investigaciones que se desarrollaron con respecto al método se cita a Eugenio Oñate en su libro cálculo de estructuras por el Método de los elementos finitos. “De hecho las publicaciones científica sobre este tema se estiman sobre 10.000 solamente en 1990” (Oñate, 1995). En la figura 2.1 se aprecia un árbol genealógico con las más importantes publicaciones que dieron inicio al método de los elementos finitos.

Si tenemos en cuenta un orden cronológico de los principales hechos que atribuyen el FEM, surge a principios de los años 1940 ante la premisa de poder resolver

problemas de elasticidad bidimensional con técnicas matriciales, usándose en un principio la división del elemento continuo en elementos tipo barra. En 1943 Courant introdujo el concepto de “elemento continuo” al lograr resolver problemas de elasticidad plana mediante la división de su dominio en elementos triangulares. Ya en la década de los 60 la introducción de los ordenadores y su aporte en los cálculos dio como resultado un avance significativo en los métodos basados en técnicas matriciales y por ende el FEM, surgiendo muchas investigaciones del método en varias direcciones. En 1960 Clough sugiere por primera vez la denominación de “elemento finito” a la solución de problemas de elasticidad plana. A partir de estas fechas este método cobro gran relevancia en varias ramas de investigación siendo a la fecha el método más potente para la solución de problemas de ingeniería complejos.

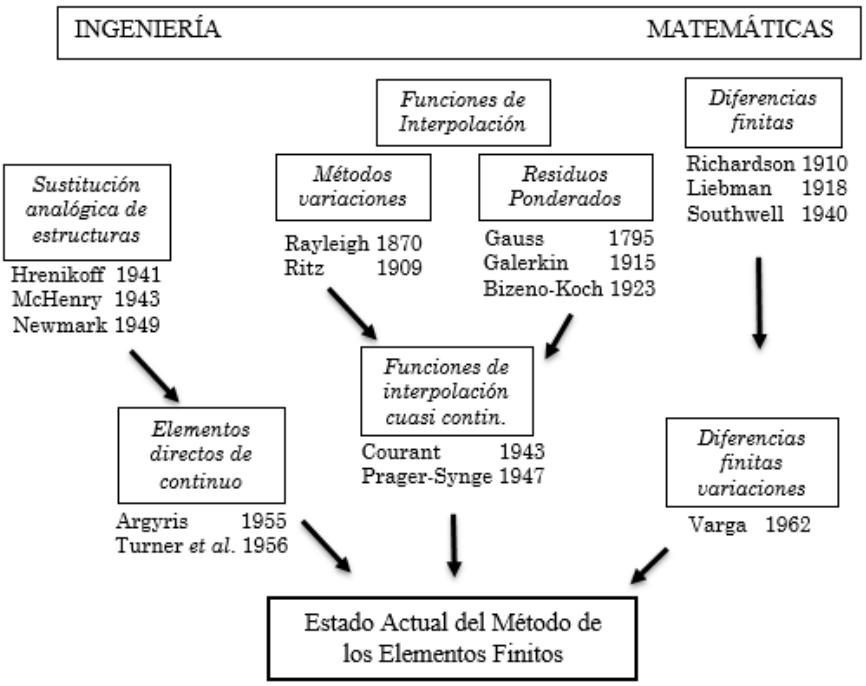


Figura 2.1 **Árbol genealógico del FEM**

Fuente:(Olgierd Cecil Zienkiewicz, Taylor, Zienkiewicz, & Taylor, 1977)

2.2.2. Enfoque del Método

El método de los elementos finitos abarca un extenso campo de aplicación e investigación, el cual se extiende a la matemática, física e ingeniería. Teniendo la metodología una interpretación y enfoque diferente para el campo que se esté abarcando.

La presente investigación está enfocada al campo del análisis estructural, orientando al FEM, al uso de matrices de gobierno, constitutivas, rigidez, entre otras. Una vez comprendidas las bases para su uso, se orientará dicha metodología a la programación, elaborándose algoritmos computacionales base, los cuales servirán como fundamento en la elaboración de códigos que se irán optimizando mediante pautas que se dejan en la presente investigación.

2.2.3. Conceptos Básicos del Método

2.2.3.1. Elemento Continuo – Elemento Discreto

Entender el comportamiento de todo un sistema complejo es algo que escapa de los límites de la mente humana, ya sea el ejemplo de un edificio con cada uno de los elementos que lo conforman. Por tal motivo, el procedimiento natural es dividir este sistema en “elementos”, los cuales se puedan entender individualmente para proceder a reconstruirlos (ensamblarlos) al sistema original y entender su comportamiento general. Un claro ejemplo de dicha división se visualiza en la figura 2.2 en el cual un sistema se divide en elementos individuales.

En caso de que la división progresiva de dicho sistema nos dé como resultado un modelo adecuado el cual posea componentes bien definidos se podrá realizar un análisis directo, en donde dichos componentes toman el nombre de “elementos discretos”. En otros casos esta subdivisión prosigue indefinidamente al no lograr obtenerse componentes exactos, por lo que llegamos a elementos basados en ecuaciones diferenciales o expresiones equivalentes, estos componentes se denominan “elementos continuos”, cuya solución se da mediante manipulaciones matemáticas las cuales no se pueden sistematizar. Dada esta problemática se plantearon varias medidas de discretización[1] aproximada, con la cual se espera que la solución continua sea lo más cercana posible a la discreta.

[1] Término usado ampliamente en el FEM, se le puede definir de una forma sencilla como la división de un sistema o elemento continuo en elementos que poseen componentes o bases bien definidas (elemento discreto), las cuales, vistas desde el análisis estructural, se puedan analizar matricialmente.

Se resalta que en otras publicaciones las interpretaciones se realizan desde un enfoque general, denominando al sistema continuo o discreta por las características de los elementos que lo conforman. Teniendo en cuenta que la presente investigación está orientada al análisis estructural, se empezaran a adaptar términos que sean fácil de comprender y correlacionar con nuestra línea de investigación.

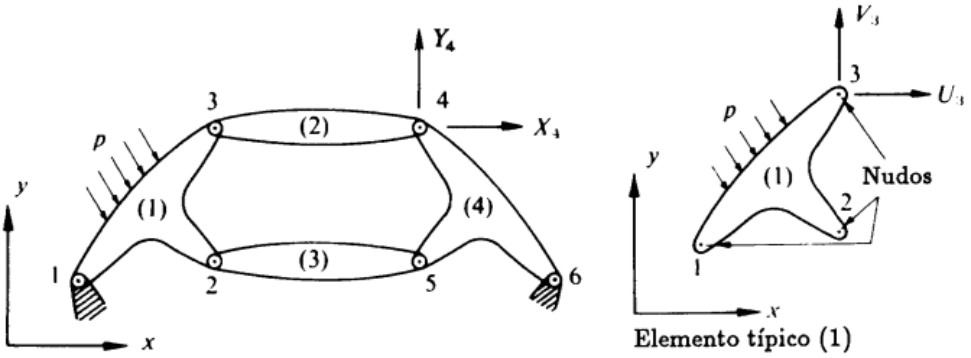


Figura 2.2 **Sistema Estructural / Discretización**

Fuente:(Olgierd Cecil Zienkiewicz et al., 1977)

2.2.3.2. Coordenadas Naturales

Para realizar un análisis aproximado de un elemento complejo en el FEM, es de vital importancia el uso de las coordenadas naturales, esto se debe a que son coordenadas adimensionales que están definidas entre los valores -1 / +1. Este hecho cobra importancia al hacer uso de la integración numérica por el método de Gauss (esta metodología se explicará a mayor detalle en el subcapítulo siguiente).

Las coordenadas naturales se determinan mediante la relación de longitudes, áreas o volúmenes. Siendo esto una investigación introductoria y aplicada al análisis estructural se tomarán las coordenadas naturales mediante la relación de longitudes. Se resalta que, si el lector tiene conocimientos de coordenadas naturales, este subcapítulo usa solo las definiciones básicas del tema y las orienta a su aplicación con el FEM.

A continuación, se procede a detallar el funcionamiento de las coordenadas naturales mediante la relación de longitudes, compatibles a un caso axialmente deformable.

Sea una línea recta entre los puntos 1 y 2, y siendo P un punto cualquiera definido por su coordenada cartesiana x (Fig. 2.3). La posición del Punto P se puede determinar en un sistema de coordenadas naturales ξ_1, ξ_2, mediante la relación de sus longitudes, como se muestra a continuación.

$$\xi_1 = \frac{L_1}{L} \quad , \quad \xi_2 = \frac{L_2}{L} \tag{2.1}$$

Es decir

$$\xi_1 = \frac{x_2 - x}{L} \quad , \quad \xi_2 = \frac{-x_1 + x}{L} \tag{2.2}$$

Ya que $L_1 + L_2 = L$, despejando la ecuación 2.1 se satisface la relación:

$$\xi_1 + \xi_2 = 1 \tag{2.3}$$

Observándose la figura 2.3 los valores de $\xi_1 \, y \, \xi_2$ varian entre 0 y 1, siendo o bien 0 o bien 1 en los extremos.

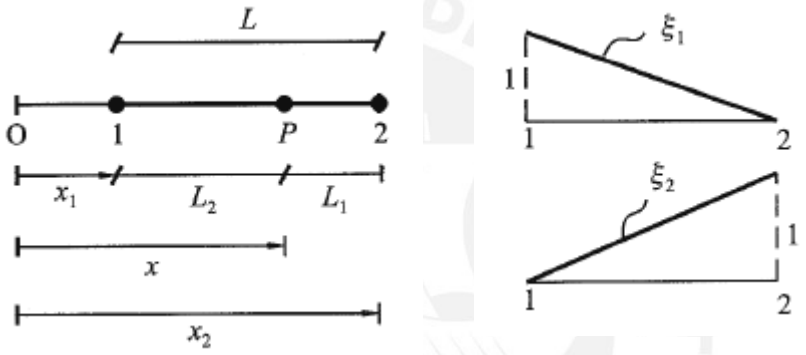


Figura 2.3 **Barra Típica (Form. Coord Natural)**

Fuente: (Olgierd Cecil Zienkiewicz et al., 1977)

Al no ser independientes $\xi_1 \, y \, \xi_2$, según la ecuación 2.3, la posición del punto P puede determinarse mediante una sola coordenada natural ξ. Usualmente esta coordenada natural ξ se define mediante la relación:

$$\xi = \frac{x - x_C}{L/2} = \frac{2(x - x_C)}{L} \tag{2.4}$$

Se resalta que la anterior ecuación es a grandes rasgos un artificio para obtener directamente el valor de la coordenada natural en un punto x . Siendo x_C la coordenada del punto medio (C) en la línea 1-2. Utilizando esta coordenada natural, las coordenadas de los puntos extremos 1 y 2 son -1 y 1, respectivamente, lo cual se aprecia en la figura 2.4. Esta ecuación brinda un sistema de elemento normalizado donde los extremos toman los valores de -1 y 1 con el punto central igual a 0, lo cual es lo que se desea para realizar una posterior integración numérica mediante Gauss Legendre.

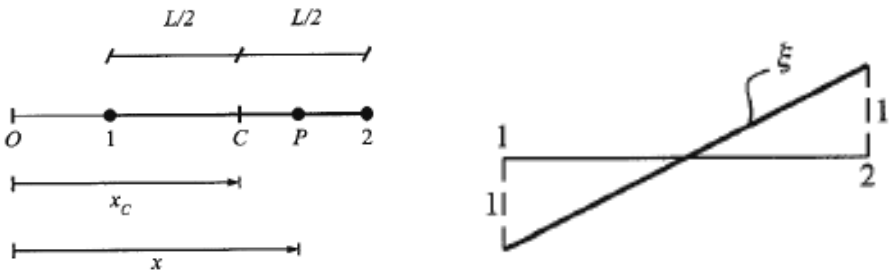


Figura 2.4 **Sistema Normalizado**

Fuente: (Manuel Vázquez., 2001)

2.2.3.3. Jacobiano

En la aplicación del FEM el uso de coordenadas naturales se hace indispensable para sistemas relativamente complejos, por lo cual el uso del jacobiano (herramienta que sirve para relacionar las derivadas de un sistema de ejes cartesianos a otro sistema de ejes) cobra importancia. Teniendo en cuenta la figura 2.5 y a partir de la regla de derivación en cadena tenemos:

$$\frac{\partial}{\partial \xi} = \frac{\partial}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial}{\partial y}\frac{\partial y}{\partial \xi} \qquad (2.5)$$

$$\frac{\partial}{\partial \eta} = \frac{\partial}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial}{\partial y}\frac{\partial y}{\partial \eta}$$

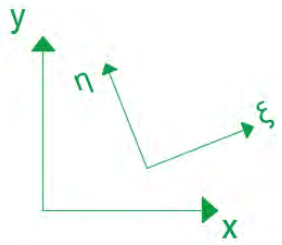


Figura 2.5 **Sistemas de ejes**

Fuente: (Manuel Vázquez., 2001) / Elaboración Propia

Expresando lo anterior en forma matricial nos queda la siguiente expresión.

$$\begin{vmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{vmatrix} = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix} \begin{vmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{vmatrix} \qquad (2.6)$$

Donde:

$$[J] = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix} \qquad (2.7)$$

A la anterior expresión se le otorga el nombre de jacobiano y es el medio de transformación de un sistema de coordenadas x, y a uno ξ, η, tal como se visualiza en la ec. 2.6 (para un caso bidimensional). A partir del jacobiano (Ec. 2.7) se puede deducir su determinante de forma sencilla, siendo:

$$\det[J] = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} \tag{2.8}$$

De igual forma se puede calcular la inversa del Jacobiano a partir del método del adjunto, bajo el cual se tiene que:

$$[J]^{-1} = \frac{Adj(J^T)}{\det[J]}$$

$$Adj(J^T) = \begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix}$$

$$[J]^{-1} = \frac{\begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix}}{\det[J]} \tag{2.9}$$

En este punto correlacionamos los sistemas de coordenadas a partir de sus vectores unitarios como se muestra en la figura 2.6.

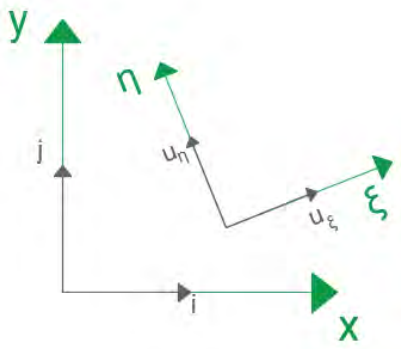


Figura 2.6 **Vectores unitarios de los sistemas de ejes**

Fuente: (Manuel Vázquez., 2001) / Elaboración Propia

Teniendo en cuenta que para el eje x, y los vectores unitarios son i, j y para el eje de coordenadas ξ, η los vectores unitarios son u_ξ, u_η, pudiendo obtener la siguiente relación:

$$dx\,\boldsymbol{i} = \frac{\partial x}{\partial \xi} d\xi\,\boldsymbol{u_\xi} + \frac{\partial x}{\partial \eta} d\eta\,\boldsymbol{u_\eta} \tag{2.10}$$

$$dy\,\boldsymbol{j} = \frac{\partial y}{\partial \xi} d\xi\,\boldsymbol{u_\xi} + \frac{\partial y}{\partial \eta} d\eta\,\boldsymbol{u_\eta}$$

Si multiplicamos vectorialmente las expresiones anteriores:

$$(dx\ \boldsymbol{i})\ x\ (dy\ \boldsymbol{j}) = \left(\frac{\partial x}{\vartheta \xi} d\xi\ \boldsymbol{u_\xi} + \frac{\partial x}{\partial \eta} d\eta\ \boldsymbol{u_\eta}\right)\ x\ \left(\frac{\partial y}{\partial \xi} d\xi\ \boldsymbol{u_\xi} + \frac{\partial y}{\partial \eta} d\eta\ \boldsymbol{u_\eta}\right)$$

$$dx\ dy\ \boldsymbol{k} = \left(\frac{\partial x}{\partial \xi}\ \frac{\partial y}{\partial \eta}\ \boldsymbol{u_\xi}\ \boldsymbol{x}\ \boldsymbol{u_\eta} + \frac{\partial x}{\partial \eta}\ \frac{\partial y}{\partial \xi}\ \boldsymbol{u_\eta}\ \boldsymbol{x}\ \boldsymbol{u_\xi}\right) d\xi\ d\eta$$

$$dx\ dy\ \boldsymbol{k} = \left(\frac{\partial x}{\partial \xi}\ \frac{\partial y}{\partial \eta}\ + \frac{\partial x}{\partial \eta}\ \frac{\partial y}{\partial \xi}\right) d\xi\ d\eta\ \boldsymbol{k}$$

$$dx\ dy\ = \left(\frac{\partial x}{\partial \xi}\ \frac{\partial y}{\partial \eta}\ + \frac{\partial x}{\partial \eta}\ \frac{\partial y}{\partial \xi}\right) d\xi\ d\eta \qquad (2.11)$$

A partir de la expresión anterior y la determinante del jacobiano (encontrada anteriormente – Ec. 2.8), se logra simplificar la expresión de la siguiente forma:

$$dA = dx\ dy\ = \left(\frac{\partial x}{\partial \xi}\ \frac{\partial y}{\partial \eta}\ + \frac{\partial x}{\partial \eta}\ \frac{\partial y}{\partial \xi}\right) d\xi\ d\eta$$

$$dA = dx\ dy\ = \det[J]\ d\xi\ d\eta \qquad (2.12)$$

Dicha expresión es de gran importancia en la aplicación de integración numérica para la obtención de la matriz de rigidez vista más adelante (subcapítulo 4.6), cobrando para un caso tridimensional la siguiente forma (la cual se obtiene bajo el mismo procedimiento, variando la dimensión de los vectores iniciales):

$$dV = dx\ dy\ dz\ = \det[J]\ d\xi\ d\eta\ d\zeta \qquad (2.13)$$

2.2.3.4. Integración Numérica – Gauss Legendre

El uso de las integrales para la obtención de la matriz de rigidez en el FEM (más adelante detallado), tiene un rol de vital importancia para encontrar la solución del sistema. Estas integrales tienden a complicarse a medida que el elemento se vuelve más complejo, por lo cual es común la aplicación de métodos de integración numérica. El método de integración más usado y el cual se detallará en esta investigación es la "Integración numérica de Gauss-Legendre", cabe mencionar que solo se detallaran las ecuaciones que lo conforman, ya que no es el fin de esta investigación conocer su formulación.

Suponiendo una función $f(\xi)$ para la cual se desea calcular su integral en un intervalo [-1,+1], es decir.

$$I = \int_{-1}^{+1} f(\xi)\, d\xi \qquad (2.14)$$

La integración numérica por Gauss-Legendre expresa: "el valor de dicha integral es igual a la suma de los productos de los valores del integrando en una serie de puntos conocidos en el interior del intervalo por unos coeficientes (pesos)

determinados" (Oñate, 1995). Es decir, para una formulación de orden p se tiene que:

$$Ip = \sum_{i=1}^{p} f(\xi) W_i \qquad (2.15)$$

En forma aplicativa se puede definir a la obtención de la respuesta ante una integral mediante Gauss como: La sumatoria de la función a integrar bajo un determinado número de puntos de integración (p), multiplicado por un peso, cuyos coeficientes se obtienen de la Cuadratura de Gauss (Tabla 2). Donde "W_i" es el peso correspondiente al punto de integración i, siendo "p" el total de dichos puntos, todo esto da como resultado la resolución de dicha integral. Se resalta que esta metodología sirve para la solución de integrales dobles y triples las cuales usaremos más adelante. A continuación, se muestra la tabla de valores de la cuadratura de Gauss-Legendre.

Tabla 2. **Coordenadas y pesos Gauss- Legendre**

Fuente: (Eugenio Oñate. 1995)

n	ξ_i	W_i
1	0.00000	2.00000
2	$\pm$0.57735	1.00000
3	$\pm$0.77459	0.55555
	0.00000	0.88888
4	$\pm$0.86113	0.34785
	$\pm$0.33998	0.65214
5	$\pm$0.90617	0.23692
	$\pm$0.53846	0.47862
	0.00000	0.56888
6	$\pm$0.93246	0.17132
	$\pm$0.86120	0.36076
	$\pm$0.23861	0.46791

El lector en este punto puede darse cuenta de la importancia en el uso de coordenadas naturales para el FEM. La tabla 2 muestra que los puntos de integración están especiados en el intervalo $-1 \leq \xi \leq 1$, lo cual abarca el rango de aplicación de las coordenadas naturales.

2.2.3.5. Principio de los Trabajos Virtuales (P.T.V)

Partiendo con la premisa de que el FEM discretiza un sistema en elementos, procedemos a analizar uno de ellos mediante el P.T.V. Suponiendo que dicho elemento está sometido a un conjunto de fuerzas externas $\{\boldsymbol{F_e}\}$, comprendidas por: Las fuerzas $\{\boldsymbol{P_e}\}$ (aplicadas en los nodos), las fuerzas $\{\boldsymbol{q_e}\}$ (distribuidas en su volumen

V_e) y las fuerzas $\{\boldsymbol{p_e}\}$ (distribuidas en su superficie S_e), lo cual se visualiza en la figura 2.7. Por principio de mecánica de materiales es fácil darse cuenta que dichas fuerzas causan un estado de esfuerzos el cual es definido mediante un vector de esfuerzos $\{\boldsymbol{\sigma}\}$.

Al aplicar tal conjunto de fuerzas externas se da el origen de un conjunto de desplazamientos nodales infinitesimales $\{\boldsymbol{\delta_e^*}\}$, los cuales tienen en cuenta las condiciones de borde del sistema. Estos desplazamientos nodales $\{\boldsymbol{\delta_e^*}\}$ dan lugar a un conjunto de desplazamientos virtuales $\{\boldsymbol{u_e^*}\}$, los cuales generan un estado de deformaciones virtuales definido por el vector de deformaciones $\{\boldsymbol{\varepsilon^*}\}$.

Siendo definido el trabajo externo por las fuerzas externas $\{\boldsymbol{F_e}\}$ durante los desplazamientos virtuales $\{\boldsymbol{\delta_e^*}\}$ y $\{\boldsymbol{u_e^*}\}$, de la siguiente forma:

$$W_e = \{\boldsymbol{\delta_e^*}\}\{\boldsymbol{P_e}\} + \int_{V_e} \{\boldsymbol{u_e^*}\}^{\boldsymbol{T}}\{\boldsymbol{q_e}\}\boldsymbol{dV_e} + \int_{S_e} \{\boldsymbol{u_e^*}\}^{\boldsymbol{T}}\{\boldsymbol{p_e}\}\boldsymbol{dS_e} \qquad (2.16)$$

El trabajo interno se basa en las fuerzas internas debidas a los esfuerzos $\{\boldsymbol{\sigma}\}$, durante las deformaciones virtuales $\{\boldsymbol{\varepsilon^*}\}$, el cual es igual a:

$$W_i = -\int_{V_e} \{\boldsymbol{\varepsilon^*}\}^{\boldsymbol{T}}\{\boldsymbol{\sigma}\}\boldsymbol{dV_e} \qquad (2.17)$$

Aplicando el P.T.V : “El trabajo que realizan las fuerzas externas aplicadas a la estructura y las fuerzas internas deben estar en equilibrio $W_e + W_i = 0$”, obtenemos la siguiente ecuación:

$$\int_{V_e} \{\boldsymbol{\varepsilon^*}\}^{\boldsymbol{T}}\{\boldsymbol{\sigma}\}\boldsymbol{dV_e} = \{\boldsymbol{\delta_e^*}\}\{\boldsymbol{P_e}\} + \int_{V_e} \{\boldsymbol{u_e^*}\}^{\boldsymbol{T}}\{\boldsymbol{q_e}\}\boldsymbol{dV_e} + \int_{S_e} \{\boldsymbol{u_e^*}\}^{\boldsymbol{T}}\{\boldsymbol{p_e}\}\boldsymbol{dS_e} \qquad (2.18)$$

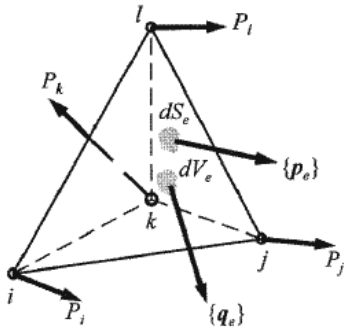


Figura 2.7 **Principio de trabajos virtuales, cargas en un elemento**

Fuente: (Manuel Vázquez., 2001)

2.2.3.6. Relación entre Deformaciones y Desplazamientos

El campo de deformaciones en un nodo del elemento puede ser representado mediante los desplazamientos en el mismo punto. Esta premisa nace del principio de compatibilidad en deformaciones descrita en cursos de mecánica estructural. Bajo esta relación y sin tener en cuenta ninguna restricción en su formulación se obtienen las siguientes ecuaciones:

$$\varepsilon_{xx} = \frac{\partial u}{\partial x} \quad \varepsilon_{yy} = \frac{\partial v}{\partial y} \quad \varepsilon_{zz} = \frac{\partial w}{\partial z} \tag{2.19}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \quad \gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \quad \gamma_{zx} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}$$

Las cuales se pueden representar matricialmente de la siguiente forma:

$$\begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.20}$$

Otorgando una formulación simplificada en función a las derivadas que sufren los desplazamientos, se llega a la siguiente expresión:

$$\{\varepsilon\} = [\partial]\{u\} \tag{2.21}$$

2.2.4. Matriz Constitutiva

La relación especifica entre las deformaciones y esfuerzos del sistema en un campo lineal se da a través de la “ley de Hooke generalizada” (algunos autores otorgan a esta relación el nombre de ecuaciones constitutivas). Se aclara que esta ley es de uso general para los diversos tipos de elementos, usándose para determinar las componentes del estado de deformación en función de las componentes del estado de esfuerzos. Dicha relación se ve simplificada en función de las restricciones del elemento que se esté analizando.

$$\varepsilon_{xx} = \frac{1}{E}\left[\sigma_{xx} - \upsilon\left(\sigma_{yy} + \sigma_{zz}\right)\right] \qquad \varepsilon_{yy} = \frac{1}{E}\left[\sigma_{yy} - \upsilon(\sigma_{xx} + \sigma_{zz})\right]$$

$$\varepsilon_{zz} = \frac{1}{E}\left[\sigma_{zz} - \upsilon\left(\sigma_{xx} + \sigma_{yy}\right)\right] \tag{2.22}$$

$$\gamma_{xy} = \frac{1}{G}\tau_{xy} \qquad \gamma_{yz} = \frac{1}{G}\tau_{yz} \qquad \gamma_{xz} = \frac{1}{G}\tau_{xz}$$

Donde:

$\sigma = Esfuerzo\ \ en\ el\ eje\ aplicado.$

$\varepsilon = Deformación\ \ en\ el\ eje\ aplicado.$

$E = Módulo\ de\ Elasticidad\ del\ sistema.$

$\tau = Cortante\ en\ el\ eje\ aplicado.$

$G = Módulo\ de\ Corte.$

$\gamma = Distorsión\ del\ sistema\ en\ el\ eje\ aplicado.$

$\upsilon = Coeficiente\ de\ Poisson.$

Las ecuaciones antes nombradas pueden expresarse matricialmente, bajo la siguiente forma:

$$\{\sigma\} = [D]\{\varepsilon\} \qquad (2.23)$$

Donde a [D] se le denomina matriz constitutiva y esta expresada en forma general como:

$$[D] = \frac{E}{(1+\upsilon)(1-2\upsilon)}\begin{bmatrix} 1-\upsilon & \upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & 1-\upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & \upsilon & 1-\upsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-2\upsilon)/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (1-2\upsilon)/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & (1-2\upsilon)/2 \end{bmatrix} \qquad (2.24)$$

2.2.5. Función de Desplazamientos

Para entender el comportamiento general de un sistema (estructura), se hace uso de expresiones matemáticas equivalentes las cuales expresen dicho comportamiento. Estas funciones matemáticas adquieren el nombre de funciones de desplazamientos, siendo capaces de otorgar en su formulación el desplazamiento en un punto cualquiera (nodo) del elemento.

Para entender de forma sencilla dichas funciones partimos de un caso típico como es el unidimensional con dos nodos a los extremos. La función de desplazamiento se puede expresar bajo la siguiente formulación.

$$u(x) = \alpha_1 + \alpha_2 x \qquad (2.25)$$

El término anterior es un polinomio que conlleva un número de parámetros "α" igual a 2. Estos parámetros representan el número de grados de libertad en todo el sistema, como se aprecia en la figura 2.8. Ahora si particularizamos la función anterior con respecto a cada nodo, obtenemos:

$$u_1 = u(x_1) = \alpha_1 + \alpha_2 x_1 \qquad (2.26)$$

$$u_2 = u(x_2) = \alpha_1 + \alpha_2 x_2$$

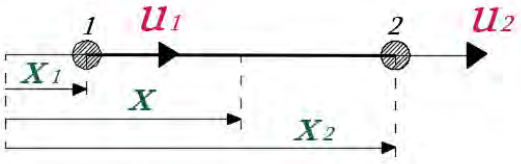


Figura 2.8 **Interpretación de la función de desp. (Unidimensional)**

Fuente: (Vázquez, 2001)

En donde los parámetros "α" serán iguales a:

$$\alpha_1 = \frac{x_2 u_1 - x_1 u_2}{L} \qquad \alpha_2 = \frac{-u_1 + u_2}{L} \tag{2.27}$$

Sustituyendo estos valores en la ecuación 2.25 y realizando una simplificación, obtendremos:

$$u(x) = N_1 u_1 + N_2 u_2 \tag{2.28}$$

La cual se puede expresar matricialmente de la siguiente manera:

$$u(x) = |N_1 \quad N_2| \begin{vmatrix} u_1 \\ u_2 \end{vmatrix} \tag{2.29}$$

En donde N_1 y N_2 son denominados “funciones de interpolación o forma”. Tomando para el caso ejemplificado (unidimensional, con deformación axial) los siguientes valores:

$$N_1 = \frac{x_2 - x}{L}, \qquad N_2 = \frac{-x_1 + x}{L} \tag{2.30}$$

Se debe resaltar que dichas funciones varían dependiendo del tipo de elemento que se está evaluando, por tal motivo es común generalizar la ecuación 2.29 para todos los tipos de la siguiente forma:

$$\{u_e\} = [N_e]\{\delta_e\} \tag{2.31}$$

Donde:

$\{u_e\} = Función\ de\ desplazamiento\ del\ elemento.$

$[N_e] = Matriz\ de\ forma\ o\ interpolación.$

$\{\delta_e\} = Vector\ de\ desplazamientos\ nodales\ del\ elemento.$

Finalmente podremos concluir que el comportamiento del sistema puede ser expresado mediante la ecuación 2.31. Dicha ecuación es fundamental para partir con una expresión matemática que exprese el comportamiento de los diversos tipos de elementos finitos.

2.2.6. Matriz de Deformaciones

También llamada matriz elemental, se obtiene en base a la matriz de interpolación (definida anteriormente) y tiene la siguiente forma:

$$[B_e] = [\partial][N_e] \tag{2.32}$$

Dicha matriz nace de la necesidad de tener una expresión matemática que relacione las deformaciones con los desplazamientos nodales del sistema. Para demostrar su formulación, partimos de la relación que existe entre los componentes de deformación y desplazamiento en un punto, generándose las siguientes relaciones de compatibilidad (Mayor información en el libro de la Elasticidad de Timoshenko (Timoshenko, 1946)):

$$\varepsilon_x = \frac{\partial u}{\partial x}, \quad \varepsilon_y = \frac{\partial v}{\partial y}, \quad \varepsilon_z = \frac{\partial w}{\partial z} \tag{2.33}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}, \quad \gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}, \quad \gamma_{zx} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}$$

Estas expresiones se pueden desarrollar matricialmente como se muestra a continuación:

$$\begin{vmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{vmatrix} = \begin{vmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{vmatrix} \begin{vmatrix} u \\ v \\ w \end{vmatrix} \tag{2.34}$$

Procediendo a expresarla en forma generalizada, obtenemos:

$$\{\varepsilon\} = [\partial]\{u\} \tag{2.35}$$

Relacionando dicha ecuación con la ecuación 2.31, tenemos que:

$$\{\varepsilon\} = [\partial][N_e]\{\delta_e\} \tag{2.36}$$

Simplificando la ecuación al otorgar una matriz equivalente, llegamos a la siguiente expresión:

$$\{\varepsilon\} = [B_e]\{\delta_e\} \tag{2.37}$$

Con la expresión anterior queda demostrada la formulación inicial de la matriz de deformación (ecuación 2.32), relacionando las deformaciones y desplazamientos nodales. En base al desarrollo de la matriz de deformación, comparto la idea de diversos autores, la cual es definir que esta matriz incorpora las propiedades geométricas del sistema en la formulación del método. El lector debe poder darse cuenta que para sistematizarla es necesario incorporar los nodos a través de coordenadas, las cuales parten de un sistema global (geometría del elemento).

2.2.7. Matriz de Rigidez

La matriz de rigidez nace del teorema de trabajos virtuales, el cual (detallado anteriormente) tiene como principio que tanto el trabajo interno y externo del sistema debe estar en equilibrio.

$$W_e + W_i = 0 \qquad (2.38)$$

En donde se interpreta que en el trabajo externo intervienen fuerzas aplicadas en los nudos, distribuidas en su volumen y en su superficie, como se pudo apreciar en el subcapítulo 2.2.3.5, por lo que retomamos la ecuación 2.16:

$$W_e = \{\delta_e^*\}^T\{P_e\} + \int_{V_e} \{u_e^*\}^T\{q_e\}dV_e + \int_{S_e} \{u_e^*\}^T\{p_e\}dS_e$$

Teniendo en mente que es el trabajo que realizan las fuerzas externas cuando se dan desplazamientos virtuales, se puede expresar de forma general como:

$$W_e = \sum F_j\delta_j^* = \{\delta_e^*\}^T\{F_e\} \qquad (2.39)$$

Sustituyendo la ecuación 2.31 en la 2.16, obtenemos la expresión de trabajo externo en función de la matriz de forma.

$$W_e = \{\delta_e^*\}^T\{P_e\} + \int_{V_e} \{\delta_e^*\}^T[N_e]^T\{q_e\}dV_e + \int_{S_e} \{\delta_e^*\}^T[N_e]^T\{p_e\}dS_e \qquad (2.40)$$

Para definir el trabajo interno del sistema, se debe tener en cuenta que la aplicación de cargas en el elemento genera deformaciones, las cuales conllevan a generar esfuerzos al sistema. Bajo esta premisa el trabajo interno queda expresado con la siguiente ecuación.

$$W_i = -\int_{V_e} \{\varepsilon^*\}^T\{\sigma\}dV_e \qquad (2.41)$$

Teniendo claro que la matriz constitutiva expresa la relación entre esfuerzos y deformaciones, obtenemos una expresión que los relacione en función a la matriz de deformación (ecuación 2.37):

$$\{\sigma\} = [D]\{\varepsilon\} \tag{2.42}$$
$$\{\sigma\} = [D][B_e]\{\delta_e\}$$

Por propiedad de la transpuesta en matrices la ecuación 2.37, se puede interpretar de la siguiente forma:

$$\{\varepsilon^*\}^T = \{\delta_e^*\}^T\,[B_e]^T \tag{2.43}$$

Reemplazando las ecuaciones 2.42 y 2.43 en la ecuación de trabajos internos (ecuación 2.41), obtenemos:

$$W_i = -\int_{V_e} \{\varepsilon^*\}^T\{\sigma\}dV_e = -\int_{V_e} \{\delta_e^*\}^T[B_e]^T[D][B_e]\{\delta_e\}dV_e$$

$$W_i = -\{\delta_e^*\}^T\left(\int_{V_e}[B_e]^T[D][B_e]dV_e\right)\{\delta_e\} \tag{2.44}$$

Entonces, igualando trabajos internos y externos tenemos como resultado.

$$\{\delta_e^*\}^T\left(\int_{V_e}[B_e]^T[D][B_e]dV_e\right)\{\delta_e\} = \{\delta_e^*\}^T\{P_e\} + \{\delta_e^*\}^T\int_{V_e}[N_e]^T\{q_e\}dV_e + \{\delta_e^*\}^T\int_{S_e}[N_e]^T\{p_e\}dS_e \tag{2.45}$$

Ahora suponiendo un desplazamiento virtual unitario en un nodo cualquiera con una dirección arbitraria (esto se relaciona con el principio de carga virtual y rigidez), la ecuación se reduce a:

$$\left(\int_{V_e}[B_e]^T[D][B_e]dV_e\right)\{\delta_e\} = \{P_e\} + \int_{V_e}[N_e]^T\{q_e\}dV_e + \int_{S_e}[N_e]^T\{p_e\}dS_e \tag{2.46}$$

Lo cual representa la ecuación matricial de equilibrio del elemento:

$$[k_e]\{\delta_e\} = \{F_e\} \tag{2.47}$$

Finalmente, la matriz de Rigidez será igual a:

$$[k_e] = \int_{V_e}[B_e]^T[D][B_e]dV_e \tag{2.48}$$

Dicha ecuación se formula para cada uno de los elementos discretos, relacionándolos mediante un posterior ensamblaje. En este punto se puede apreciar que es una integral compleja en función del elemento y se muestra muchas veces simplificada, siendo su expresión real una triple integral:

$$[k_e] = \iiint_{V_e}[B_e]^T[D][B_e]dV_e \tag{2.49}$$

Al ser una triple integral es difícil de sistematizarla por lo que se hace uso de métodos de integración numérica como el de Gauss-Legendre (Subcapítulo 2.2.3.4),

llevando a la integral en función de coordenadas naturales y aplicando el principio de integración para resolverla bajo sumatorias (ecuación 2.15), lo cual se verá a más detalle en el propio desarrollo del método (capitulo 4.6).

2.3. ALGORITMOS DE PROGRAMACIÓN

Un algoritmo se define típicamente como “Un método para resolver un problema” (Joyanes Aguilar, 2003), dicho método surgió cuando KhoWarizmi un matemático de la época, planteo reglas paso a paso para aplicar suma, resta, multiplicación y división en números decimales. En 1945 el matemático G. Pólya publica una metodología general para la resolución de problemas matemáticos (Pólya, 1945). Dicha metodología fue adaptada como el método base en la programación y es usada como una guía de buenas prácticas para la elaboración de los mismos, encontrándose simplificada en tres pasos según el libro: “Fundamentos de informática y programación” (Martín, Toledo, & Cerverón, 1995). Su esquematización se muestra en la figura 2.9.

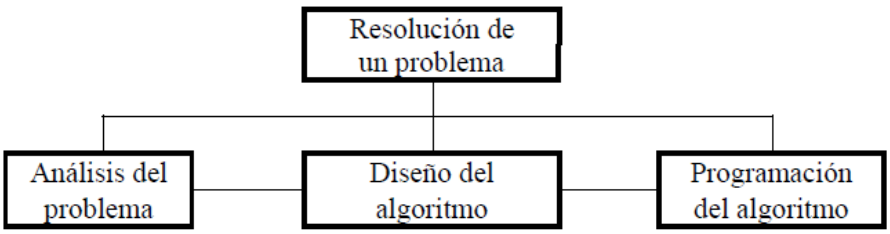


Figura 2.9 **La resolución de un problema de informática**

Fuente: (Martín et al., 1995)

El esquema anterior muestra que antes de efectuar la elaboración propia del código (programación del algoritmo) se debe analizar el problema a programar y diseñar el tipo de algoritmo con el que se trabajara. Los dos primeros pasos son independientes del tipo de lenguaje de programación a usar por lo que sirve como base para personas que no estén inmersas en el mundo de la programación.

En este punto el lector debe tener claro que los algoritmos de programación son herramientas indispensables que ahorran tiempo y mejoran la compresión de un código, esto debido a que no están relacionados directamente con el tipo de lenguaje a usar o incluir términos técnicos difíciles de comprender. En contrariedad a esta premisa la formulación de algoritmos es muy poco usada en el desarrollo de códigos, debido a que los programadores de gran experiencia y/o programadores que han trabajado un código desde los inicios, no requieren su uso para reconocer de manera

fluida el código, volviendo su implementación un sobrecargo en horas de trabajo y siendo en algunos casos más tediosa/demorosa la elaboración del algoritmo en comparación al propio código.

A continuación, se realizará una comparativa del costo en tiempo de comprensión de un código ejemplificativo para una rutina común como es la de cocinar un huevo (con y sin algoritmos de programación).

- ***Código Base sin Algoritmos de Programación:***

```
1   //  Programa para Cocinar un Huevo
2
3   disp('Si==1 No==0');
4
5   var1=input('Freir Huevo:');
6
7   if var1=1
8       disp("Freir Huevo")
9   else
10      disp("Hervir Huevo")
11
12
13  Var2=input('Agregar Sal:');
14
15  if var2=1
16      disp('Agregar Sal')
17  else
18      disp("Servirlo en el plato")
```

Figura 2.10 **Ejemplo de código, sin algoritmos de programación**

Fuente: Elaboración Propia

En la figura anterior se visualiza un código base sencillo, como es el de cocinar un huevo, en el cual solo se usan condicionales.

- ***Código Base con Algoritmos de Programación:***

Este ejemplo contiene el código mostrado anteriormente junto a una herramienta de representación del código (Algoritmo de Programación-Diagrama de Flujo).

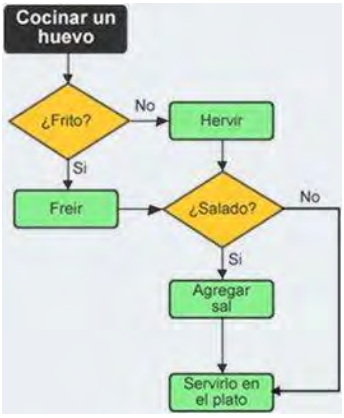


Figura 2.11 **Algoritmo de cocinar un huevo**

Fuente: ÁreaTecnología, 2016

En la figura 2.11 se visualiza el algoritmo del código mostrado en la figura 2.10, cuyo fin es dar a entender la rutina que se realiza al cocinar un huevo. A simple vista se puede apreciar que el código acompañado de un algoritmo es clave para el entendimiento del mismo, todo esto para una persona la cual no haya participado en su desarrollo o no tenga conocimientos en programación. Esta premisa aumenta su importancia en un código complejo en el cual se hace uso de un gran número de herramientas de programación (bucles, condicionales, etc) y variables de entrada/salida, como es el caso de los elementos finitos.

Diversos autores que han incursionado el estudio y la publicación de documentación respecto al FEM, cometen el error de no dar importancia al tema de aprendizaje conjunto de la teoría del FEM con su aplicación en ordenadores. Error que según el punto de vista del autor es la principal causa por la cual se crea una falsa complejidad de la aplicación del método en estudiantes e investigadores que desean incursionar en el tema.

2.4. SOFTWARE MATLAB

MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices") es un software con herramientas matemáticas y de carácter ingenieril, el cual ofrece un entorno de desarrollo integrado (IDE), incorporando consigo un lenguaje de programación propio (lenguaje MATLAB). Combina un entorno de carácter profesional con una interfaz de desarrollo sencilla, la cual es de mucha ayuda para un análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices/arrays directamente. (Mathworks, 2016)

Entre las funcionalidades de MatLab que destacan sobre otros programas (tipo IDE) resalta la sencillez que presta en la formulación e integración de matrices en la elaboración de códigos. La interfaz de programación en diseño gráfico se ve potenciada gracias a la herramienta GUIDE (editor de interfaces de usuario - GUI), la cual a través de axes y otras herramientas puede llegar a elaborar interfaces avanzadas para los programas que se estén formulando.

Este software es de uso común en el ámbito de la ingeniería, teniendo gran acogida por diversos investigadores, respaldado por una gran comunidad (Mathworks), siendo la versión usada en la presente investigación la R2019b.

CAPÍTULO 3

Tipos de Elementos Finitos

Resumen

En el presente capítulo se da una descripción de los principales tipos de elementos finitos utilizados para el análisis estructural y programados en los códigos anexados. Para su fácil entendimiento a nivel ingenieril se divide cada uno de ellos en cuatro partes: hipótesis fundamental (formulación base), campo de desplazamientos y campo de deformaciones/tensiones, todo esto debido a que cada elemento finito tiene una formulación distinta que lo caracteriza.

CAPÍTULO III: TIPOS DE ELEMENTOS FINITOS

3.1. ELEMENTOS ARTICULADOS (AXIALMENTE DEFORMABLE)

3.1.1. INTRODUCCIÓN

Para introducir los conceptos básicos que caracterizan al FEM lo habitual es comenzar con la descripción detallada de un elemento simple, como lo es una barra axialmente deformable (componente básico en las estructuras articuladas). Este componente es fundamental para comprender de forma sencilla como trabaja el FEM, lo cual se detalla en el capítulo 4 de la presenten investigación.

3.1.2. HIPÓTESIS FUNDAMENTAL (FORMULACIÓN BASE)

La formulación base para elementos axialmente deformables sigue los principios enseñados en cursos como Resistencia de Materiales.

1. Las deformaciones solo se dan en el eje longitudinal de la barra (solo cuenta con deformación axial).
2. Las secciones transversales en la barra permanecen planas antes y después de la deformación.
3. Todos los puntos contenidos en la sección transversal al plano medio (eje longitudinal) tienen el mismo desplazamiento horizontal.
4. La barra axial solo estará sometida a esfuerzos de tracción o compresión.

Las representaciones de algunas de estas hipótesis se pueden apreciar en la figura 3.1, la cual visualiza una barra empotrada en volado sometida a esfuerzos de tracción en toda su longitud. Se puede apreciar que las deformaciones solo ocurren en el eje longitudinal de la barra.

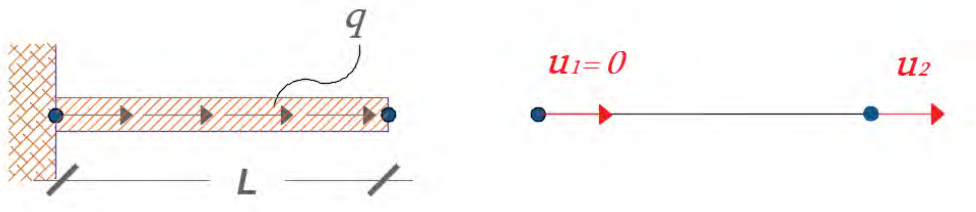


Figura 3.1 **Barra en volado bajo tracción en toda su longitud**

Fuente: Elaboración Propia

3.1.3. Campo de Desplazamientos

De las hipótesis 1, 2 y 3, acompañadas de la figura 3.1 se deduce que:

$$\begin{aligned} u(x,y,z) &= u(x) \\ v(x,y,z) &= 0 \\ w(x,y,z) &= 0 \end{aligned} \qquad (3.1)$$

Donde u representa el desplazamiento que la barra sufre en sentido de su eje. Se tiene que resaltar que esta definición está referida en función a los ejes locales de la misma, cambiando posiblemente si lo analizamos en función a los ejes globales del sistema. De forma matricial se puede representar el campo de desplazamientos de la siguiente forma:

$$u = [u(x),0\ ,0\ ]^T \qquad (3.2)$$

3.1.4. Campo de deformaciones y tensiones

Según la teoría clásica de la elasticidad, el vector de deformaciones tridimensional en un punto (general para un caso lineal elástico) está definido por seis componentes, los cuales se vieron en el subcapítulo 2.2.6 con la ecuación 2.33:

$$\varepsilon = \left[\varepsilon_x\ \ \varepsilon_y\ \ \varepsilon_z\ \ \gamma_{xy}\ \ \gamma_{xz}\ \ \gamma_{yz}\right]^T \qquad (3.3)$$

Donde:

$$\varepsilon_x = \frac{\partial u}{\partial x} \qquad \varepsilon_y = \frac{\partial v}{\partial y} \qquad \varepsilon_z = \frac{\partial w}{\partial z}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \qquad \gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \qquad \gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}$$

Al solo estar sometida solamente a deformación axial la barra según las hipótesis planteadas, el campo de deformaciones estaría compuesto de la siguiente forma:

$$\begin{aligned} \varepsilon_x = \frac{\partial u}{\partial x} \qquad \varepsilon_y = 0 \qquad \varepsilon_z = 0 \\ \gamma_{xy} = 0 \qquad \gamma_{xz} = 0 \qquad \gamma_{yz} = 0 \end{aligned} \qquad (3.4)$$

A partir de estas ecuaciones se deduce que la única deformación se da en el plano longitudinal de la misma (como se planteó en la hipótesis inicial). Con respecto a la obtención de esfuerzos es fácil darse cuenta que solamente se tendrá esfuerzos axiales a tracción o compresión con respecto a su eje longitudinal, estando el vector de esfuerzos conformado por un solo elemento:

$$\sigma = \{\sigma_x\} \qquad (3.5)$$

Se debe resaltar que, si bien la representación real de los esfuerzos en el elemento varia a lo largo de la sección transversal, por temas de simplificación en el análisis (hipótesis planteadas) se considera uniforme como se muestra en la figura 3.2.

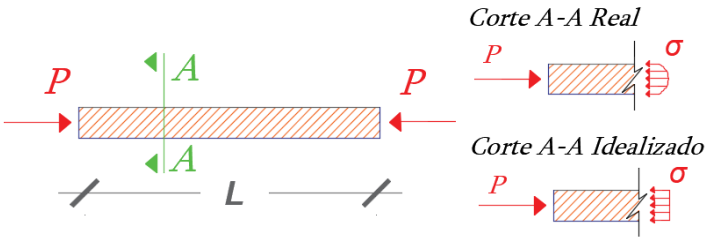


Figura 3.2 **Comportamiento real/idealizado de una Barra**

Fuente: Elaboración Propia

3.2. VIGAS: TEORÍA DE BERNOULLI

3.2.1. INTRODUCCIÓN

El análisis de una viga se puede realizar mediante dos formulaciones: La primera es la teoría clásica de flexión en vigas (Euler-Bernoulli), la segunda es la teoría de Timosheko, la cual considera efectos de corte en su deformación. En estos subcapítulos, se presentarán las hipótesis para las dos formulaciones, las cuales estarán directamente relacionadas al elemento tipo placa.

3.2.2. HIPÓTESIS FUNDAMENTAL (T. EULER-BERNOULLI)

Las hipótesis sobre las que se basa la teoría de Euler-Bernoulli para flexión en vigas son las siguientes:

1. El desplazamiento vertical en cualquier punto de la sección transversal de la viga se considera pequeño e igual al desplazamiento de su eje longitudinal.
2. El desplazamiento que se da en el eje Y se considera nulo (Figura 3.3).
3. Las secciones transversales normales al eje longitudinal, permanecerán planas y perpendiculares al eje después de ocurrida la deformación.

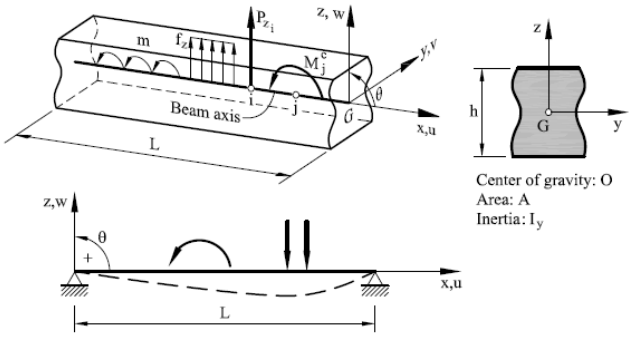


Figura 3.3 **Sec. Transversal. Cargas y modelo matemático de una viga**

Fuente: (Oñate, 2013)

3.2.3. Campo de Desplazamientos

De las hipótesis antes planteadas y apoyándose de la figura 3.4, formulamos:

$$u(x, y, z) = -z\,\theta(x)$$
$$v(x, y, z) = 0 \qquad (3.6)$$
$$w(x, y, z) = w(x)$$

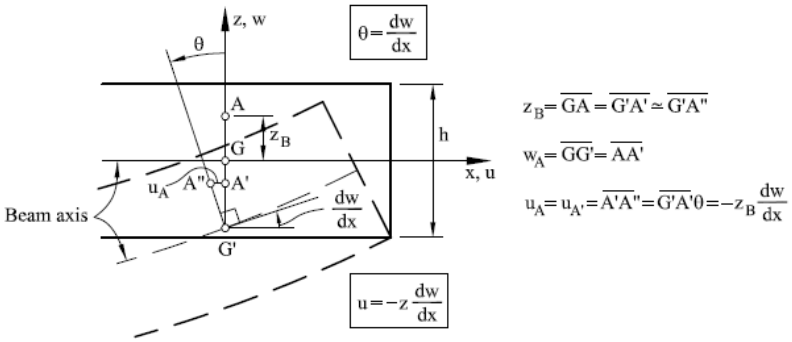


Figura 3.4 **Deformación de la viga bajo la teoría de Euler-Bernoulli**

Fuente: (Oñate, 2013)

Como se puede apreciar de la figura 3.4 y la hipótesis 3, el desplazamiento que se da en el eje x está en función del giro que sufre la viga. De tal forma el giro estará definido por:

$$\theta_x = \frac{dw}{dx} \qquad u = -z\,\frac{dw}{dx} \qquad (3.7)$$

Con las ecuaciones planteadas, se puede formular matricialmente el campo de desplazamientos de la siguiente forma:

$$u = \left[-z\,\frac{dw}{dx},\ 0\ ,\ w(x)\right]^T \qquad (3.8)$$

3.2.4. Campo de deformaciones y tensiones

Para obtener las deformaciones partimos de la teoría de deformaciones para un elemente tridimensional y aplicamos las hipótesis planteadas, con lo cual tendríamos las siguientes deformaciones:

$$\varepsilon_x = \frac{du}{dx} = -z\frac{d^2w}{dx^2} \qquad \varepsilon_y = 0 \qquad \varepsilon_z = 0 \qquad (3.9)$$
$$\gamma_{xy} = 0 \qquad \gamma_{xz} = 0 \qquad \gamma_{yz} = 0$$

A partir de estas ecuaciones se deduce que la teoría de Euler-Bernoulli no toma en cuenta efectos de corte en las deformaciones. Con respecto a la obtención de esfuerzos, al igual que la barra axialmente deformable, solo se tendrá esfuerzos en el eje x (eje de desarrollo), de la siguiente forma:

$$\sigma = \{\sigma_x\} \qquad (3.10)$$

3.3. VIGAS: TEORÍA DE TIMOSHENKO

3.3.1. HIPÓTESIS FUNDAMENTAL (T. TIMOSHENKO)

Las hipótesis sobre las que se basa la teoría de Timoshenko para flexión en vigas, son las siguientes:

1. El desplazamiento vertical en cualquier punto de la sección transversal de la viga, se considera pequeño e igual al desplazamiento en su eje longitudinal.
2. El desplazamiento que se da en el eje Y se considera nulo.
3. Las secciones transversales normales al eje longitudinal, permanecerán planas, mas no necesariamente perpendiculares al eje después de ocurrida la deformación.

3.3.2. Campo de Desplazamientos

De las hipótesis antes planteadas y apoyándose de la figura 3.5, formulamos:

$$u(x, y, z) = -z\,\theta(x)$$
$$v(x, y, z) = 0 \qquad (3.11)$$
$$w(x, y, z) = w(x)$$

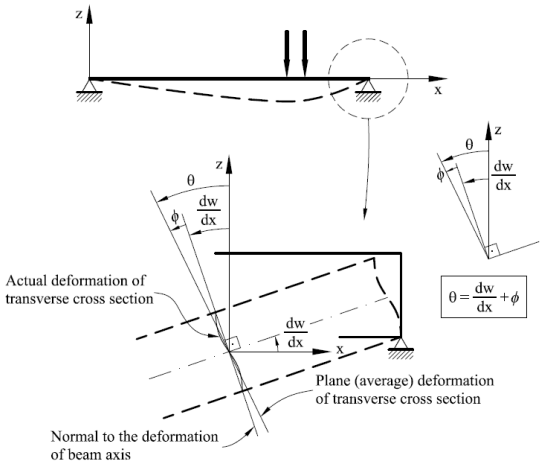


Figura 3.5 **Deformación dela viga bajo la teoría de Timoshenko**

Fuente: (Oñate, 2013)

Como se puede apreciar en la figura y la hipótesis 3, el desplazamiento que se da en el eje x está en función al giro que sufre la viga, el cual a su vez puede ser descompuesta en dos giros.

$$\theta_x = \frac{dw}{dx} + \phi \qquad u = -z\left(\frac{dw}{dx} + \phi\right) \tag{3.12}$$

A partir de esto, se pueden representar matricialmente los desplazamientos de la siguiente forma:

$$u = \left[-z\left(\frac{dw}{dx} + \phi\right),\ 0\ ,\ w(x)\right]^T \tag{3.13}$$

3.3.3. Campo de deformaciones y tensiones

Bajo las hipótesis planteadas, tendríamos las siguientes deformaciones para una viga bajo la teoría de Timoshenko:

$$\varepsilon_x = \frac{du}{dx} = -z\frac{d\theta}{dx} \qquad \varepsilon_y = 0 \qquad \varepsilon_z = 0 \tag{3.14}$$

$$\gamma_{xy} = \frac{dw}{dx} + \frac{du}{dz} = \frac{dw}{dx} - \theta = \phi \qquad \gamma_{xz} = 0 \qquad \gamma_{yz} = 0$$

A partir de estas ecuaciones se deduce que la teoría de Timoshenko toma en cuenta efectos de corte ante las deformaciones (como se planteó en la hipótesis iniciales), en el caso de la Teoría de Euler-Bernoulli estos efectos desaparecían al ser $\frac{dw}{dx} = \theta$.

Bajo lo planteado anteriormente el vector de esfuerzos del sistema cambiara, teniendo en cuenta esfuerzos de corte en el eje X.

$$\sigma = \begin{Bmatrix} \sigma_x \\ \tau_{xz} \end{Bmatrix} \tag{3.15}$$

3.4. PLACAS DELGADAS: TEORÍA DE KIRCHHOFF

3.4.1. INTRODUCCIÓN

Entre los elementos semi-planos se cuentan con diversas formulaciones de análisis como son los elementos Plate (Placas), Shell y Membrana, al estar limitada la presente investigación a un campo de aprendizaje se trabajará solamente con la primera formulación. Los elementos Plate o Placas en español, al igual que los elementos tipo viga, cuentan con dos formas de análisis: Placas gruesas bajo la teoría de Reissner-Mindlin y placas delgadas bajo la teoría de Kirchhoff.

De manera simplificada se puede considerar a la teoría de placas análoga a la teoría de vigas, siendo la viga de Euler-Bernoulli similar a la placa tipo Kirchhoff en su hipótesis de ortogonalidad de la normal y la teoría de Timoshenko similar a la placa de Reissner Mindlin, introduciendo efectos de corte en su deformación al no considerar ortogonalidad en su plano.

3.4.2. HIPÓTESIS FUNDAMENTAL (T. PLACAS / KIRCHHOFF)

Las hipótesis sobre las que se basa la teoría de placas tipo Kirchhoff son las siguientes:

1. Los puntos en el plano medio solo se desplazan verticalmente, es decir:
$$u = v = 0.$$
2. Todos los puntos contenidos en una normal al plano medio tienen el mismo desplazamiento vertical.
3. La tensión normal σ_z es despreciable.
4. Los puntos sobre rectas ortogonales al plano medio (figura 3.6), permanecen sobre rectas también ortogonales a la deformada del plano medio, después de ocurrida la deformación.

Se debe resaltar que las hipótesis 1, 2 y 4, permiten definir el campo de desplazamientos a través del espesor de la placa. La tercera hipótesis afecta a la relación tensión-deformación del elemento.

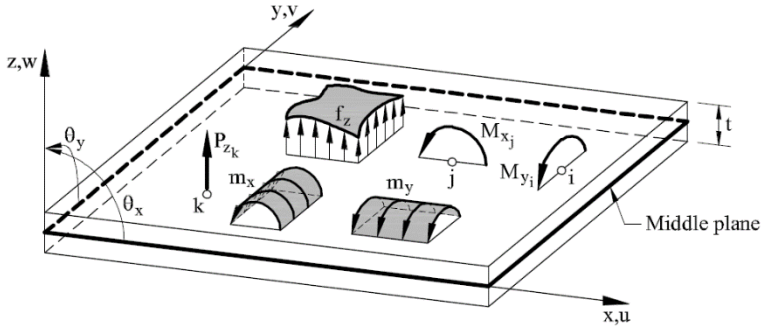


Figura 3.6 **Modelo Geométrico de una placa ante cargas**

Fuente: (Oñate, 2013)

3.4.3. Campo de Desplazamientos

De las hipótesis 1,2 y 4 antes planteadas y la figura 3.7 se deduce que:

$$\left.\begin{aligned} u(x,y,z) &= -z\theta_x(x,y) \\ v(x,y,z) &= -z\theta_y(x,y) \end{aligned}\right\} \ (1ra\ y\ 4ta\ hipótesis) \qquad (3.16)$$
$$\left.\begin{aligned} w(x,y,z) &= -w(x,y) \end{aligned}\right\} \ (2da\ hipótesis)$$

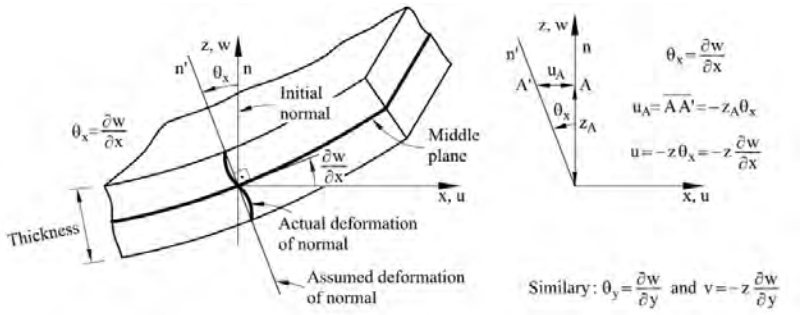


Figura 3.7 **Comportamiento del plano medio de una placa delgada**

Fuente: (Oñate, 2013)

Donde w es el desplazamiento vertical (flecha) de los puntos del plano medio y θ_x,θ_y son los ángulos que definen el giro de la normal contenido en los planos xz e yz, respectivamente (hipótesis 4). A partir de esto se puede definir matricialmente el vector de desplazamiento para un punto en el plano medio de la siguiente forma:

$$u = \left[w, \theta_x, \theta_y\right]^T \tag{3.17}$$

De la hipótesis 4 y la figura 3.7 se deduce que

$$\theta_x = \frac{\partial w}{\partial x} \qquad \theta_y = \frac{\partial w}{\partial y} \tag{3.18}$$

Si ingresamos lo obtenido en la ecuación 3.18 a la 3.17, obtenemos en detalle el vector de desplazamiento de un punto cualquiera del plano medio de la placa, como se muestra a continuación:

$$u = \left[w, \frac{\partial w}{\partial x}, \frac{\partial w}{\partial y}\right]^T \tag{3.19}$$

Pudiendo el campo de desplazamientos expresarse de la siguiente forma (a través de las ecuaciones 3.16 y 3.18).

$$u(x,y,z) = -z\frac{\partial w(x,y)}{\partial x} \qquad v(x,y,z) = -z\frac{\partial w(x,y)}{\partial y}$$

$$w(x,y,z) = w(x,y) \tag{3.20}$$

3.4.4. Campo de deformaciones y tensiones

Aplicando lo formulado en el campo de desplazamientos, obtenemos las siguientes ecuaciones:

$$\varepsilon_x = \frac{\partial u}{\partial x} = -z\frac{\partial^2 w}{\partial x^2} \qquad \varepsilon_y = \frac{\partial v}{\partial y} = -z\frac{\partial^2 w}{\partial y^2} \qquad \varepsilon_z = \frac{\partial w}{\partial z} \approx 0 \tag{3.21}$$

$$\gamma_{xz} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} = 0 \qquad \gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} = 0 \qquad \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = -2z\,\frac{\vartheta^2 w}{\vartheta x\,\vartheta y}$$

En este punto se puede deducir que para placas delgadas no se contemplan esfuerzos de corte con respecto a los ejes x e y proyectados con el eje z. Las deformaciones concluidas son resultados de las hipótesis fundamentales impuestas, siendo la principal diferencia entre una placa delgada y gruesa, y por lo cual el vector de esfuerzos estaría representado de la siguiente manera:

$$\sigma = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \qquad (3.22)$$

3.5. PLACAS GRUESAS: TEORÍA DE REISSNER - MINDLIN

3.5.1. HIPÓTESIS FUNDAMENTAL (T. PLACA / REISSNER-MINDLIN)

La teoría de placas de Reissner-Mindlin empieza con una formulación igual a la de Kirchhoff, con la única diferencia de la hipótesis de ortogonalidad de la normal durante la deformación.

De esta forma, se mantienen las tres primeras hipótesis de la teoría de Kirchhoff, modificándose la cuarta hipótesis sobre la ortogonalidad de la normal de la siguiente forma:

4. La recta es normal al plano medio antes de ocurrida la deformación, una vez ocurrida la deformación la recta permanece normal, mas no necesariamente ortogonal a dicho plano.

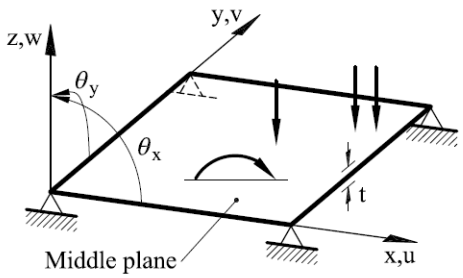


Figura 3.8 **Definición geométrica de una placa y convenio de signos**

Fuente: (Oñate, 2013)

3.5.2. Campo de Desplazamientos

De las hipótesis 1,2 (tratadas anteriormente) y 4, conjunto a la Fig. 3.9 se deduce:

$$u(x,y,z) = -z\theta_x(x,y) \qquad v(x,y,z) = -z\theta_y(x,y)$$

$$w(x,y,z) = w(x,y) \qquad (3.23)$$

Donde θ_x y θ_y son los ángulos que definen el giro de la normal. Así, el vector de desplazamientos para los puntos que se encuentren en el plano medio se define como:

$$u = \left[w, \theta_x, \theta_y\right]^T \qquad (3.24)$$

De la hipótesis 4 sobre el giro de la normal y la figura 3.9, se tiene que:

$$\theta_x = \frac{\partial w}{\partial x} + \phi_x \qquad \theta_y = \frac{\partial w}{\partial y} + \phi_y \qquad (3.25)$$

Es decir, los giros de la normal en un punto se componen de dos elementos: Los primeros, $\frac{\partial w}{\partial x}$ y $\frac{\vartheta w}{\vartheta y}$, son debidos al cambio de pendiente del plano medio. Los segundos, ϕ_x y ϕ_y, se deben al giro adicional de la normal al no permanecer necesariamente ortogonal a la deformada del plano medio.

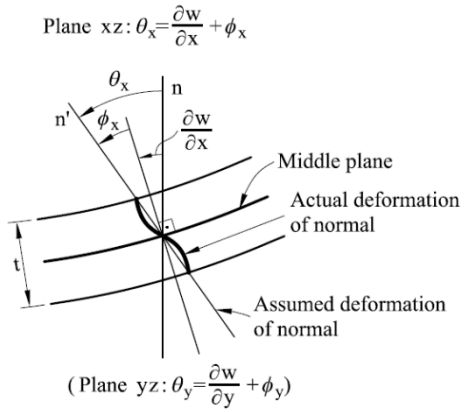


Figura 3.9 **Comportamiento del plano medio de una placa gruesa**

Fuente: (Oñate, 2013)

3.5.3. Campo de deformaciones y tensiones

Para obtener el campo de deformaciones partimos de la definición del campo de deformaciones para un elemento tridimensional. Implementando en dicha formulación el campo de desplazamientos obtenidos anteriormente, con lo cual tenemos:

$$\varepsilon_x = \frac{\partial u}{\partial x} = -z\frac{\partial \theta_x}{\partial x} \qquad \varepsilon_y = \frac{\partial v}{\partial y} = -z\frac{\partial \theta_y}{\partial y} \qquad \varepsilon_z = \frac{\partial w}{\partial z} \approx 0$$

$$\gamma_{xz} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} = -\theta_x + \frac{\partial w}{\partial x} = -\phi_x \qquad \gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} = -\theta_y + \frac{\partial w}{\partial y} = -\phi_y$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = -z\left(\frac{\partial \theta_x}{\partial y} + \frac{\partial \theta_y}{\partial x}\right) \qquad (3.26)$$

Agrupando las deformaciones como un vector, nos queda la siguiente formulación:

$$\varepsilon = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \\ \cdots \\ \gamma_{xz} \\ \gamma_{yz} \end{Bmatrix} = \begin{Bmatrix} -z\frac{\partial \theta_x}{\partial x} \\ -z\frac{\partial \theta_y}{\partial y} \\ -z\left(\frac{\partial \theta_x}{\partial y} + \frac{\partial \theta_y}{\partial x}\right) \\ \cdots \\ -\theta_x + \frac{\partial w}{\partial x} \\ -\theta_y + \frac{\partial w}{\partial y} \end{Bmatrix} = \begin{Bmatrix} \varepsilon_f \\ \cdots \\ \varepsilon_c \end{Bmatrix} \qquad (3.27)$$

En donde:

$$\varepsilon_f = \begin{Bmatrix} -z\frac{\partial \theta_x}{\partial x} \\ -z\frac{\partial \theta_y}{\partial y} \\ -z\left(\frac{\partial \theta_x}{\partial y} + \frac{\partial \theta_y}{\partial x}\right) \end{Bmatrix} \qquad \varepsilon_c = \begin{Bmatrix} -\theta_x + \frac{\partial w}{\partial x} \\ -\theta_y + \frac{\partial w}{\partial y} \end{Bmatrix} \qquad (3.28)$$

Llegados a este punto se puede deducir que los esfuerzos tangenciales en los ejes x,y respecto a z no son 0 a diferencia del análisis por Kirchhoff. Estando demostrándose que el análisis por el método de Reissner-Mindlin considera efectos de corte en el campo de deformaciones. Puesto que por la hipótesis 3 la tensión normal σ_z es nula, el vector de esfuerzos tendría la siguiente forma:

$$\sigma = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \\ \cdots \\ \tau_{xz} \\ \tau_{yz} \end{Bmatrix} = \begin{Bmatrix} \sigma_f \\ \cdots \\ \sigma_c \end{Bmatrix} \qquad (3.29)$$

En donde:

$$\sigma_f = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \qquad \sigma_c = \begin{Bmatrix} \tau_{xz} \\ \tau_{yz} \end{Bmatrix} \qquad (3.30)$$

3.6. ELEMENTO BIDIMENSIONAL

3.6.1. INTRODUCCIÓN

Los elementos bidimensionales se caracterizan por generar efectos de tensión y deformación plana. La diferencia entre los elementos tipo placa y los bidimensionales yace en la forma de aplicar las cargas, estando las cargas aplicadas en el mismo plano de desarrollo para los bidimensionales y ortogonales al plano para el tipo placa.

Un elemento bidimensional puede analizarse mediante tensión plana o deformación plana, la diferencia entre estos dos nace en poder definir si el elemento tiene un espesor definido o se le asigna uno representativo (unitario), lo cual afecta (simplifica) algunos pasos de análisis.

3.6.2. HIPÓTESIS FUNDAMENTAL

Las hipótesis sobre las que se basa un elemento bidimensional son las siguientes:

1. Los desplazamientos solo se darán en el plano x e y del elemento. En otras palabras, no hay desplazamientos perpendiculares al plano ($w = 0$).
2. A partir de la continuidad del sistema (espesor), se puede clasificar el sistema en tensión plana o deformación plana.
3. Los elementos con continuidad perpendicular a su plano (espesor no definido), se trabajarán con un espesor equivalente t=1 (en la mayoría de casos).

3.6.3. Campo de Desplazamientos

De la hipótesis 1 se deduce que:

$$u(x,y,z) = u(x,y) \qquad v(x,y,z) = v(x,y) \qquad w(x,y,z) = 0 \tag{3.31}$$

Donde w es el desplazamiento perpendicular al plano, el cual al ser un caso bidimensional será 0 (hipótesis 1). A partir de la ecuación 3.31 deducimos que el vector de desplazamientos tendría la siguiente forma:

$$u = [u(x,y), v(x,y)]^T \tag{3.32}$$

3.6.4. Campo de deformaciones y tensiones

Teniendo en cuenta las hipótesis planteadas y la figura 3.10. El campo de deformaciones para un elemento bidimensional estará definido de la siguiente forma:

$$\varepsilon_x = \frac{\partial u}{\partial x} \qquad \varepsilon_y = \frac{\partial v}{\partial y} \qquad \varepsilon_z = 0$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \approx \theta_1 + \theta_2 \qquad \gamma_{yz} = 0 \qquad \gamma_{xz} = 0 \tag{3.33}$$

Al no tener desplazamientos perpendiculares al plano se sobreentiende que las deformaciones respecto al mismo serán 0. Con la formulación realizada en la ecuación 3.34 el vector de esfuerzos para un elemento bidimensional estará definido con los siguientes componentes:

$$\{\sigma\} = \{\sigma_x, \sigma_y, \tau_{xy}\}^T \tag{3.34}$$

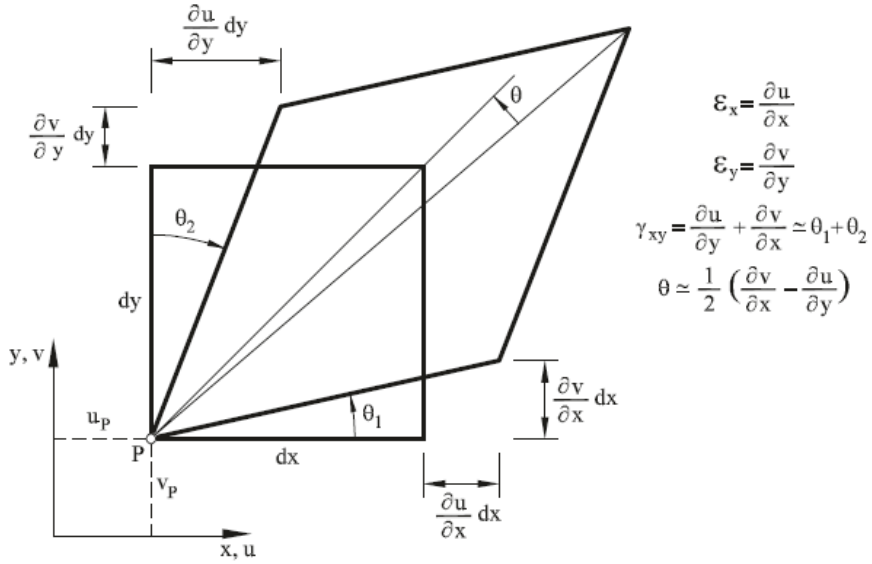


Figura 3.10 **Deformaciones en un elemento bidimensional**

Fuente: (Oñate, 2013)

3.7. ELEMENTO TRIDIMENSIONALES

3.7.1. INTRODUCCIÓN

Los elementos tridimensionales son la base para el análisis de estructuras complejas, su formulación parte de la idea de que el sistema cuenta con todos los grados de libertad “abiertos” (sin restricciones) para cada uno de los elementos. Se ha de resaltar que no se debe confundir los GDLs del análisis para el elemento con las condiciones de borde que se le aplica al sistema.

El análisis para elementos tipo tridimensionales ha sido muy estudiado en el transcurso de los años, por lo que se tienen muchas formulaciones respecto al mismo. Esto ya sea para elementos cúbicos irregulares, en forma de pirámide, entre otros. En esta investigación solamente se trabajará y elaborará el código para elementos cuadrangulares bajo un análisis de 8 nodos.

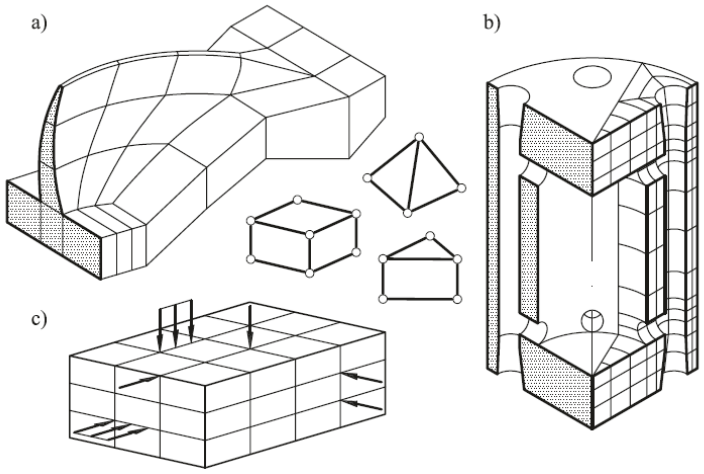


Figura 3.11 **Ejemplos de estructuras tridimensionales**

Fuente: (Oñate, 2013)

3.7.2. HIPÓTESIS FUNDAMENTAL

Tratándose del caso tridimensional no hay hipótesis de restricción, tomando completamente las bases de teoría de elasticidad para solidos tridimensionales (desplazamientos, deformaciones, tensiones).

En la figura 3.12 se aprecia como el caso tridimensional no cuenta con restricciones en su formulación, también se visualiza las diversas cargas a la cual puede estar sometido.

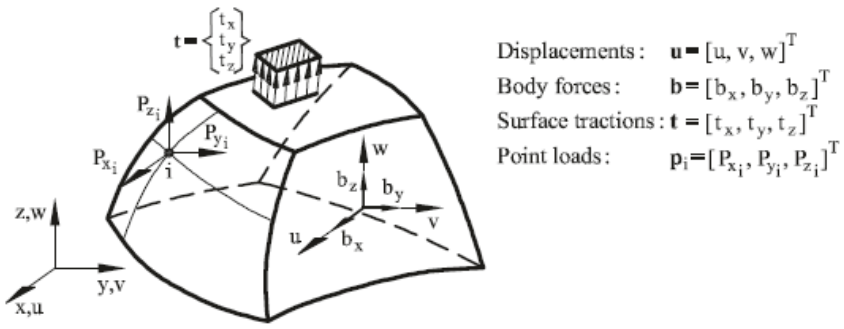


Figura 3.12 **Solido 3D. Desplazamientos y cargas**

Fuente: (Oñate, 2013)

3.7.3. Campo de Desplazamientos

Como se mencionó anteriormente el campo de desplazamientos no cuenta con restricciones en su formulación, por lo que el vector de desplazamientos abarcaría todos sus posibles componentes:

$$u = [u(x, y, z), v(x, y, z), w(x, y, z)]^T \qquad (3.35)$$

3.7.4. Campo de deformaciones y tensiones

El campo de deformaciones considera todos los posibles componentes para un elemento tridimensional, los cuales son los generales (Ec. 3.3 y 2.33):

$$\varepsilon = \begin{bmatrix} \varepsilon_x & \varepsilon_y & \varepsilon_z & \gamma_{xy} & \gamma_{xz} & \gamma_{yz} \end{bmatrix}^T$$

En donde:

$$\varepsilon_x = \frac{\partial u}{\partial x} \qquad \varepsilon_y = \frac{\partial v}{\partial y} \qquad \varepsilon_z = \frac{\partial w}{\partial z}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \qquad \gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \qquad \gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}$$

Estando el vector de esfuerzos, formulado de la siguiente forma (sin restricciones):

$$\{\sigma\} = \{\sigma_x, \sigma_y, \sigma_z, \tau_{xy}, \tau_{xz}, \tau_{yz}\}^T \qquad \textit{(3.36)}$$

CAPÍTULO 4

Desarrollo del Método de los Elementos Finitos

Resumen

En el presente capitulo se desarrolla el procedimiento de obtención de desplazamientos, deformaciones y esfuerzos. Se debe tener en cuenta que, si bien las hipótesis de formulación varían en función del tipo de elemento analizado, el desarrollo del método es similar. Se desarrolla a manera de ejemplo el elemento axialmente deformable para una comprensión sencilla del método.

CAPÍTULO IV: DESARROLLO DEL MÉTODO DE LOS ELEMENTOS FINITOS

4.1. INTRODUCCIÓN

El desarrollo del FEM es un procedimiento amplio, el cual orientado a la aplicación en ordenador mediante el uso de matrices se vuelve algo común y sistemático, variando su formulación dependiendo del tipo de elemento que se esté analizando. En este capítulo se desarrollará el elemento tipo barra con deformaciones axiales, ya que se desea dar a entender de la forma más sencilla posible el modo básico de trabajo del FEM. Una vez comprendida la forma de trabajar de un elemento sencillo como es dicha barra resultará fácil abarcar otros elementos, teniendo en cuenta que la principal variación para otros elementos nace de las hipótesis fundamentales (propiedades características) que tiene cada uno, las cuales se detallaron el capítulo anterior.

4.2. FUNCIÓN DE DESPLAZAMIENTOS

Retomando las bases enseñadas en el subcapítulo 2.2.5, la obtención del desplazamiento en un nodo cualquiera está representado mediante una “función de desplazamientos”. La formulación establecida en dicho capitulo es aplicable al comportamiento de una barra axialmente deformable, esto al estar en función de un eje y solo tener dos GDLs[2] por elemento, por lo cual retomamos la ecuación 2.25, conjunto a su formulación detallada para cada nodo (Ec. 2.26).

$$u(x) = \alpha_1 + \alpha_2 x$$

$$u(x_1) = u_1 = \alpha_1 + \alpha_2 x_1 \qquad u(x_2) = u_2 = \alpha_1 + \alpha_2 x_2$$

Se vuelve a recalcar que "α" son parámetros característicos de la función y representan el número de GDLs en el elemento (2 en este caso). Si el lector desea evaluar otro elemento como es el bidimensional, debe tener en cuenta que dicho elemento presenta un mayor número de GDLs y un campo de desplazamientos que no solo obedece a un eje x, como se vio en el subcapítulo 3.6.3, por lo cual su función de desplazamiento inicial cobraría mayor complejidad teniendo en cuenta la cantidad de nodos por elemento que se deseen evaluar.

[2] Nota del Autor: A lo largo de la presente investigación el termino Grados de Libertad, estará representado mediante las siglas GDLs, esto a motivo de evitar redundancia.

Siendo por ejemplo para un elemento bidimensional rectangular de 4 nodos la siguiente:

$$u(x,y) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy \qquad (4.1)$$
$$v(x,y) = \alpha_5 + \alpha_6 x + \alpha_7 y + \alpha_8 xy$$

Respecto a la formulación de estas funciones si el lector desea obtener una mayor información respecto al tema, se le recomienda el libro de Eugenio Oñate(Oñate, 2009), recalcando que dichas formulaciones ya están estipuladas y analizadas por lo cual el usuario tan solo debe elegir con cual desea trabajar.

4.3. FUNCIÓN DE FORMA

Teniendo clara la función de forma inicial asignada para la barra axialmente deformable y a partir de lo aprendido en el subcapítulo 2.2.5, retomamos las funciones de forma estipuladas en la ecuación 2.30:

$$N_1 = \frac{x_2 - x}{L}, \qquad N_2 = \frac{-x_1 + x}{L}$$

Para sistematizar la aplicación del FEM en ordenadores es vital interpretar cada elemento en matrices con uso sistemático por lo cual reformulamos lo anterior a la siguiente forma:

$$N_1 = 1 - \frac{x}{L}, \qquad N_2 = \frac{x}{L} \qquad (4.2)$$

Y la expresamos matricialmente en la denominada matriz de forma del elemento ("N_e"), a partir de la ecuación 2.29:

$$u(x) = |N_1 \quad N_2| \begin{vmatrix} u_1 \\ u_2 \end{vmatrix}$$
$$[N_e] = |N_1 \quad N_2|$$
$$[N_e] = \left[1 - \frac{x}{L} \quad \frac{x}{L}\right] \qquad (4.3)$$

4.4. MATRIZ CONSTITUTIVA

Como se vio anteriormente la relación esfuerzo/deformación (en un campo lineal) está representada mediante una matriz denominada “constitutiva”, la cual se formuló en las ecuaciones 2.23 y 2.24 a partir de la ley de Hooke generalizada (esta sin ninguna restricción):

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

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

Ya que se está evaluando la barra unidimensional, reformulamos las hipótesis iniciales, partiendo desde la relación esfuerzo/deformación inicial, con la inclusión de los coeficientes de Lamé (esto para una simplificación en los coeficientes):

$$\sigma_{xx} = \lambda\left(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}\right) + 2\,\mu\,\varepsilon_{xx} \qquad (4.4)$$

$$\sigma_{yy} = \lambda\left(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}\right) + 2\,\mu\,\varepsilon_{yy}$$

$$\sigma_{zz} = \lambda\left(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}\right) + 2\,\mu\,\varepsilon_{zz}$$

$$\tau_{xy} = G\,\gamma_{xy} \qquad \tau_{yz} = G\,\gamma_{yz} \qquad \tau_{zx} = G\,\gamma_{zx}$$

En donde λ y μ representan los coeficientes de Lamé y tienen la siguiente formulación:

$$\lambda = \frac{\upsilon\,E}{(1+\upsilon)(1-2\upsilon)} \qquad \mu = G = \frac{E}{2(1+\upsilon)} \qquad (4.5)$$

Ahora, en este punto debemos tener en cuenta las hipótesis fundamentales del elemento evaluado (Ec. 3.4):

$$\varepsilon_x = \frac{\partial u}{\partial x} \qquad \varepsilon_y = 0 \qquad \varepsilon_z = 0$$

$$\gamma_{xy} = 0 \qquad \gamma_{xz} = 0 \qquad \gamma_{yz} = 0$$

Aplicando dichas hipótesis a las ecuaciones en 4.4, notamos que solo nos quedamos con σ_{xx}, por lo que:

$$\sigma_{xx} = E\,\varepsilon_{xx} \qquad \sigma_{yy} = 0 \qquad \sigma_{zz} = 0 \qquad (4.6)$$

$$\tau_{xy} = 0 \qquad \tau_{yz} = 0 \qquad \tau_{zx} = 0$$

Por lo cual la matriz constitutiva para este caso estaría representada por:

$$[D] = E \qquad (4.7)$$

Esto era evidente en un principio por el tipo de elemento analizado, pero se siguió el procedimiento para entender cómo puede variar la matriz constitutiva en función al elemento que se está evaluando y al tipo de planteamiento de las hipótesis fundamentales.

4.5. MATRIZ DE RIGIDEZ

4.5.1. Matriz de deformación

Antes de empezar a desarrollar la matriz de rigidez, se debe tener en claro que existen diversos principios que se pueden optar para su formulación, como lo son el principio de trabajos virtuales (P.T.V) o el Principio de la energía Potencial. En el subcapítulo 2.2.3.5 se planteó el P.T.V, el cual se incorporará en el desarrollo que se está haciendo para la barra axialmente deformable.

Partimos de la formulación analizada para la matriz de deformación con las ecuaciones 2.32 y la 4.3.

$$[B_e] = [\partial][N_e]$$

$$[B_e] = \frac{d}{dx}\left[1 - \frac{x}{L} \quad \frac{x}{L}\right]$$

Aplicando la derivada obtenemos la matriz de deformación final para el elemento.

$$[B_e] = \left[-\frac{1}{L} \quad \frac{1}{L}\right] \tag{4.8}$$

4.5.2. Determinación de la matriz de Rigidez – Vector de Fuerzas

Sabiendo que la premisa base del P.T.V. formula que la suma del trabajo interno y externo de un sistema debe ser igual a 0, retomamos la ecuación simplificada obtenida en el subcapítulo 2.2.7 (Ec. 2.46).

$$\left(\int_{V_e} [B_e]^T [D][B_e] dV_e\right)\{\delta_e\} = \{P_e\} + \int_{V_e} [N_e]^T \{q_e\} dV_e + \int_{S_e} [N_e]^T \{p_e\} dS_e$$

En donde se puede visualizar que el conjunto de fuerzas externas se da mediante la siguiente formulación:

$$\{F_e\} = \{P_e\} + \int_{V_e} [N_e]^T \{q_e\} dV_e + \int_{S_e} [N_e]^T \{p_e\} dS_e \tag{4.9}$$

$$\{F_e\} = \{F_{Pe}\} + \{F_{qe}\} + \{F_{pe}\}$$

Donde:

$F_e = Vector\ de\ fuerzas\ aplicadas\ a\ todo\ el\ sistema.$

$F_{Pe} = Vector\ de\ fuerzas\ aplicadas\ directamente\ al\ nodo.$

$F_{qe} = Vector\ de\ fuerzas\ distribuidas\ en\ todo\ su\ volumen.$

$F_{pe} = Vector\ de\ fuerzas\ distribuidas\ en\ la\ superficie.$

Teniendo en cuenta que estamos analizando una barra axialmente deformable con 2 nodos por elemento, se pueden reducir las fuerzas externas a la siguiente formulación:

$$\{F_e\} = \{F_{Pe}\}$$

$$\{F_e\} = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \tag{4.10}$$

Para la obtención de la matriz de rigidez partimos de la ecuación de rigidez general (Ec. 2.48), e incorporamos la matriz de deformación formulada anteriormente (Ec. 4.8) y la matriz constitutiva (Ec. 4.7).

$$[k_e] = \int_{V_e} [B_e]^T [D][B_e] dV_e \tag{4.11}$$

$$[k_e] = \int_{V_e} \begin{bmatrix} -\frac{1}{L} \\ \frac{1}{L} \end{bmatrix}^T [E] \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix} dV_e$$

$$[k_e] = \int_{L} \begin{bmatrix} -\frac{1}{L} \\ \frac{1}{L} \end{bmatrix}^T [E] \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix} * A\, dx$$

Podemos simplificar la expresión anterior dejando la integral en función al cambio de área que sufre la barra en toda su longitud.

$$[k_e] = \frac{E \int_L A(x) A\, dx}{L^2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{4.12}$$

Finalmente, si la barra presente sección transversal constante la matriz de rigidez, estaría representada de la siguiente forma:

$$[k_e] = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{4.13}$$

Esta sería la matriz de rigidez del elemento si se trabajara de manera independiente cada barra (asignando un eje local a cada elemento), esta formulación es atípica en sistemas estructurales compuestos de varias barras ubicadas en distintas localizaciones. Para la obtención de una matriz de rigidez en función de ejes globales es común realizar una extrapolación del elemento con ayuda de la ecuación matricial de equilibrio de un elemento y una matriz de transformación o localización, variando la formulación a la siguiente forma.

$$[k_e]\{\delta_e\} = \{F_e\}$$

$$\left(\frac{EA}{L}\right)\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} = \begin{bmatrix}F_1\\ F_2\end{bmatrix}$$

$$\frac{EA}{L}\begin{bmatrix}1 & 0 & -1 & 0\\ 0 & 0 & 0 & 0\\ -1 & 0 & 1 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}u_1\\ v_1\\ u_2\\ v_2\end{bmatrix} = \begin{bmatrix}F_1\\ 0\\ F_2\\ 0\end{bmatrix} \qquad (4.14)$$

En este punto procedemos a convertir el análisis del elemento local a un elemento global. Para esto recordamos conceptos de análisis matricial, en donde tenemos una matriz de transformación igual a:

$$[R] = \begin{bmatrix}\cos\alpha & -\text{sen}\,\alpha\\ \text{sen}\,\alpha & \cos\alpha\end{bmatrix} \qquad (4.15)$$

Siendo la matriz de rigidez del elemento global obtenida de la siguiente forma:

$$\left[K_g\right] = [R][K_L][R]^T \qquad (4.16)$$

De la misma manera, las cargas aplicadas al sistema tendrían que formularse en función de los ejes globales, multiplicándose la matriz de transformación con el vector de fuerzas locales de la siguiente manera:

$$\left[F_g\right] = [R][F_L] \qquad (4.17)$$

Con esto obtendremos los desplazamientos en función de X/Y global, para obtener más información sobre la metodología de transformación de coordenadas, se recomienda leer el libro de Análisis Estructural de Ottazzi (Ottazzi Pasino, 2011) o ir al anexo C.4.C : “Matriz de rigidez de una barra en coordenadas globales” del libro de Elementos Finitos de Manuel Vázquez (Vázquez, 2001).

4.6. MÉTODOS DE INTEGRACIÓN

En el desarrollo de la matriz de rigidez del elemento (Ec. 2.48), se aprecia que su formulación contiene una integral de volumen. Dicha integral se puede desarrollar directamente siempre y cuando la sección sea uniforme (como se desarrolló anteriormente). En el caso de que el elemento no sea uniforme, el proceso típico es hacer uso de métodos de integración como el de Gauss-Legendre (visto anteriormente). Partamos desglosando la ecuación 2.48 en función de cada eje cartesiano, como se muestra a continuación:

$$[k_e] = \int_{V_e} [B_e]^T [D][B_e] dV_e$$

$$[k_e] = \iiint [B_e]^T [D][B_e]\, dxdydz \qquad (4.18)$$

Dicha integral presenta problemas en su desarrollo y su sistematización, por lo cual se procedemos a usar los principios del método de integración numérica de Gauss (subcapítulo 2.2.3.4). Ya que el método de integración hace uso de la Tabla 2 (Cuadratura de Gauss-Legendre), la cual comprende puntos de integración con un rango entre -1 a +1, es necesario transformar el sistema de coordenadas cartesianas a naturales, modificando la matriz de rigidez (Ec 4.18) de la siguiente forma:

$$[k_e] = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} [B_e]^T [D][B_e] |J| d\xi d\eta d\mu \qquad (4.19)$$

Se debe resaltar que dicha ecuación tiene en cuenta la formulación $|J|$, la cual es la determinante del Jacobiano y su formulación como medio de transformación de coordenadas se visualizó en el subcapítulo 2.2.3.3 con la ecuación 2.13, la cual se muestra a continuación:

$$dV_e = dx \cdot dy \cdot dz = \det[J]\, d\xi \cdot d\eta \cdot d\zeta$$

En este punto procedemos a aplicar el principio del método de Integración Numérica mediante la cuadratura de Gauss (Ec. 2.15), estando la triple integral de rigidez representada de la siguiente forma:

$$[k_e] = \sum_{p=1}^{n} \sum_{q=1}^{n} \sum_{r=1}^{n} [[B_e]^T [D][B_e]|J|]_{p,q,r}\, W_p W_q W_r \qquad (4.20)$$

Como se puede apreciar la triple integral queda expresada en forma de sumatorias, las cuales son fácil de sistematizar. Todas estas formulaciones se han expresadas de forma muy general, por lo que si el lector desea profundizar en el tema se recomienda el libro de E. Finitos de M. Vázquez (Vázquez, 2001).

CAPÍTULO 5

Implementación de Algoritmos de Programación

Resumen

Se presenta métodos para representar algoritmos de programación, tomando el diagrama de flujo como método usado en la presente investigación, definiéndose su funcionamiento y las partes que lo conforman. Respecto a la interacción de los algoritmos con la aplicación del FEM se explica su representación mediante dos ejemplos: el procedimiento general del método y el uso de la cuadratura de Gauss Legendre para la obtención de valores representativos.

CAPÍTULO V: IMPLEMENTACIÓN DE ALGORITMOS DE PROGRAMACIÓN

5.1. MÉTODOS DE REPRESENTACIÓN

La representación gráfica del algoritmo se debe realizar mediante el uso de herramientas, las cuales sean independientes del lenguaje de programación empleado. Con el pasar del tiempo han surgido diversos métodos para representar un algoritmo, siendo los más usuales:

1. Diagrama de flujo.
2. Diagrama N-S (Nassi-Schneiderman).
3. Lenguaje de especificación de algoritmos: pseudocodigo.
4. Lenguaje español, inglés.
5. Formulas.

Los dos primeros son frecuentes a encontrar en libros de programación, siendo el diagrama de flujo comúnmente enseñado en centros educativos. Por tal motivo se opta a usar dicha herramienta en la presente investigación.

5.2. DIAGRAMA DE FLUJO

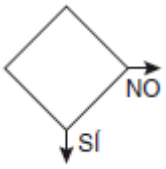
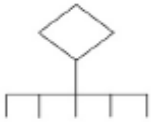
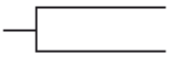
También conocido en inglés como flowchart, es una de las técnicas de representación gráfica más usada en la elaboración de programas. Una definición practica de dicha herramienta es la que brinda Jayanes Aguilar en su libro sobre Fundamentos de Programación: “Es un diagrama que utiliza símbolos (cajas), teniendo los pasos del algoritmo escritos en dichas cajas unidas por flechas, denominadas líneas de flujo, que indican la secuencia en que se debe ejecutar” (Joyanes Aguilar, 2003).

5.3. SÍMBOLOS EN UN DIAGRAMA DE FLUJO

Como se definió anteriormente un diagrama de flujo escribe los pasos a seguir mediante la aplicación de “cajas”. Dichos elementos se ven enlazados mediante flechas, representando la secuencia de cada uno de los pasos del algoritmo. Cada una de estas “cajas” tiene una propia representación gráfica y funcionalidad. A continuación se adjunta una tabla con los principales símbolos, la cual está basada en la que brinda J. Aguilar (Joyanes Aguilar, 2003).

Tabla 3. **Símbolos del diagrama de flujo**

Fuente: Elaboración Propia

Principales Símb.	Función
	Terminal, representa el comienzo / “inicio” y el final / “fin” de un programa. Puede representar también una parada o interrupción prevista que se necesite aplicar al programa.
	Entrada/Salida, representa cualquier tipo de introducción de datos (almacenamiento) o salida de información procesada en forma de data, gráficos, etc.
	Proceso, sirve para señalar la realización de una operación con respecto a la data. Su proceso abarca operaciones aritméticas, de transferencia, etc.
	Decisión, indica operaciones lógicas o comparativas entre datos. En función a los resultados se determinan distintos caminos a seguir, normalmente abarcando dos tipos de salidas en caso de cumplir o no la condición planteada.
	Decisión múltiple, símbolo que sirve para trazar distintos caminos que se pueden seguir en función al resultado y al condicional que se aplicó inicialmente
	Flecha, indicador de dirección/sentido en la ejecución de operaciones.
	Línea conectora, indica unión entre dos símbolos.
	Comentario, herramienta que sirve para añadir comentarios en el símbolo o flujo que se está realizando.

5.4. REPRESENTACIÓN PRINCIPAL DEL FEM CON D. FLUJO

Entendidos los conceptos básicos del método de elementos finitos y la composición de un diagrama de flujo respecto a los elementos que lo representan. Se procede a usar dicha herramientas para representar el procedimiento general del método de los elementos finitos en el rango lineal, el cual servirá como base para la generación del código para cada tipo de elemento trabajado.[3]

[3] Se resalta que dicho diagrama es general, y cada procedimiento tendrá su diagrama particular con sus respectivos subdiagramas si es necesario.

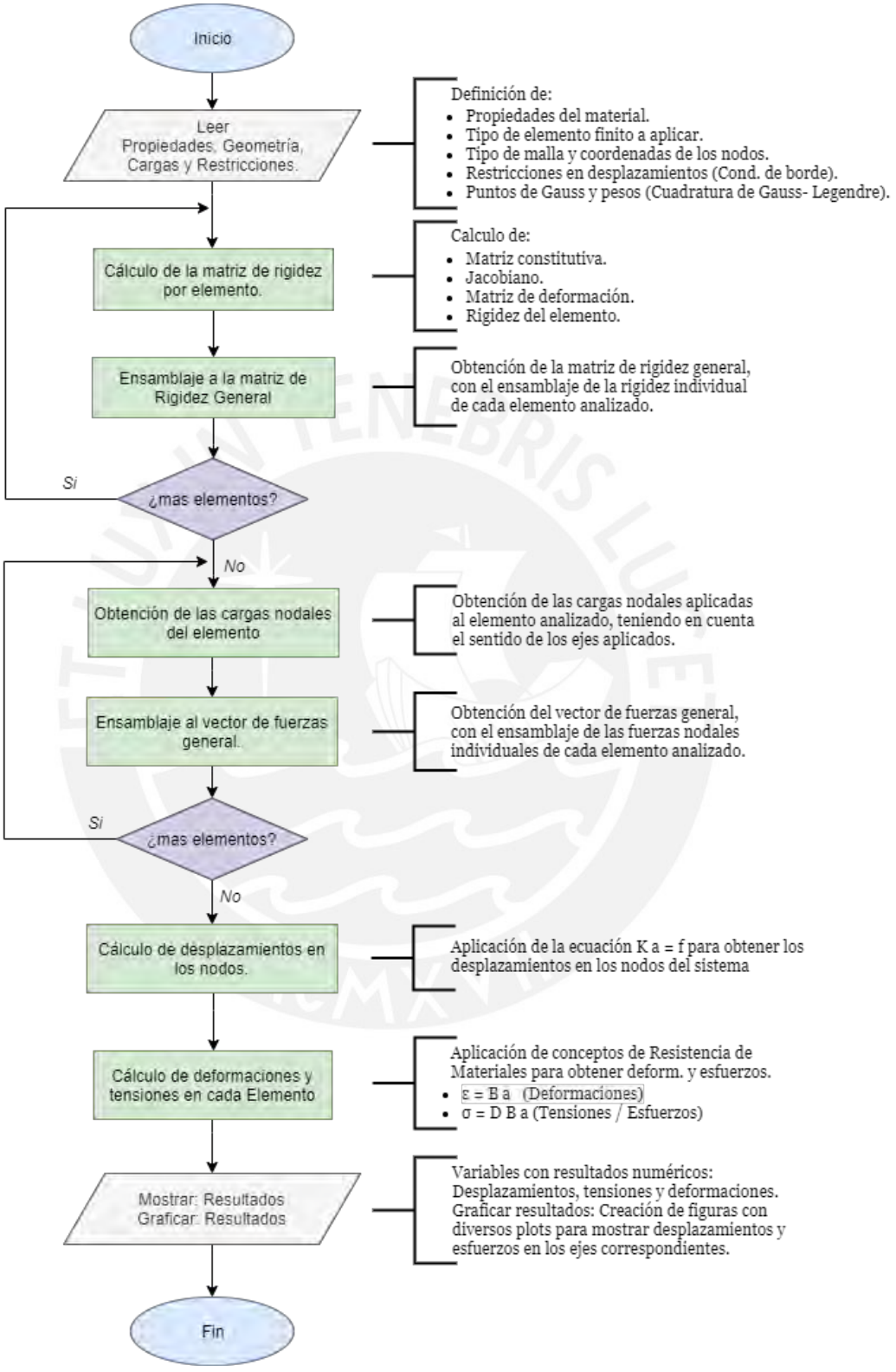


Figura 5.1 **Diagrama de flujo General en el FEM (Rango Lineal)**

Fuente: Draw.io / Elaboración Propia

El diagrama de flujo mostrado en la figura 5.1, representa los pasos generales a seguir para realizar un análisis mediante el FEM en el rango lineal. Se debe resaltar que algunas pautas como son las hipótesis de formulación o el propio mesh han sido obviadas, esto último por el hecho de ser general siendo lo anterior variable y definido por el tipo de elemento que se esté evaluando, por ejemplo, un elemento tipo barra tiene un mesh de tipo lineal a diferencia de un elemento tridimensional que puede optar por un mesh de tipo tridimensional.

A continuación, se mostrará el diagrama de flujo de un proceso en específico, como lo es la obtención de los Pesos y Coordenadas en la Cuadratura de Gauss-Legendre.

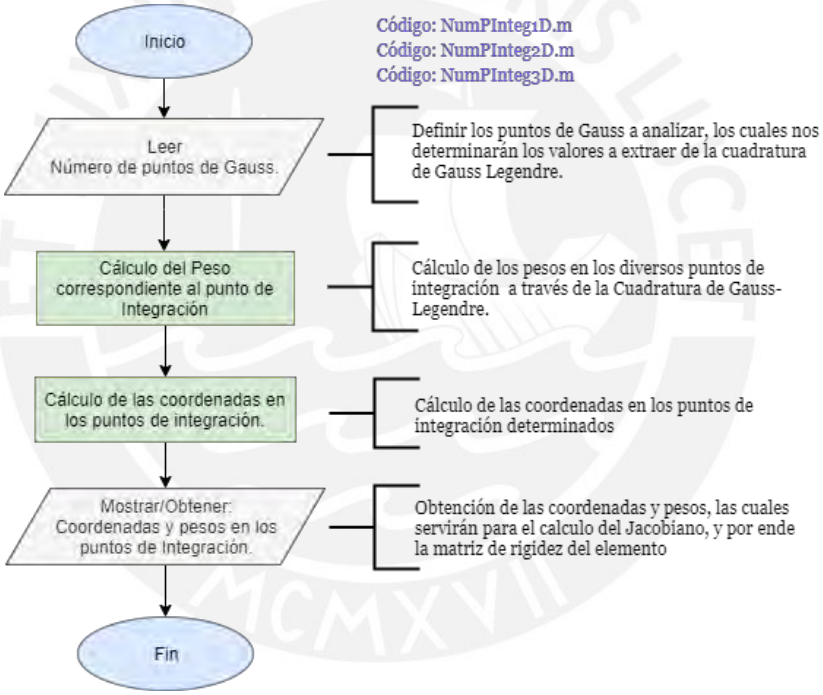


Figura 5.2 **Diagrama de flujo pesos y coordenadas (P. Integración)**

Fuente: Draw.io / Elaboración Propia

La figura 5.1 muestra un proceso general de todo el método en sí, a diferencia de la figura 5.2, la cual muestra tan solo una etapa específica de desarrollo. De igual forma se tienen diversos diagramas de flujo que detallan varias etapas y los cuales sirven para comprender de forma detallada el código (estas se pueden visualizar en el Anexo 01: Diagramas de Flujo). Finalmente, y a manera de ejemplo se muestra el diagrama de flujo para un código especifico como es el elemento tipo barra, se debe resaltar que los comentarios añaden la/las líneas de código exactas donde se realizan dicho procedimiento, esto para una mayor sencillez en la interacción con el código.

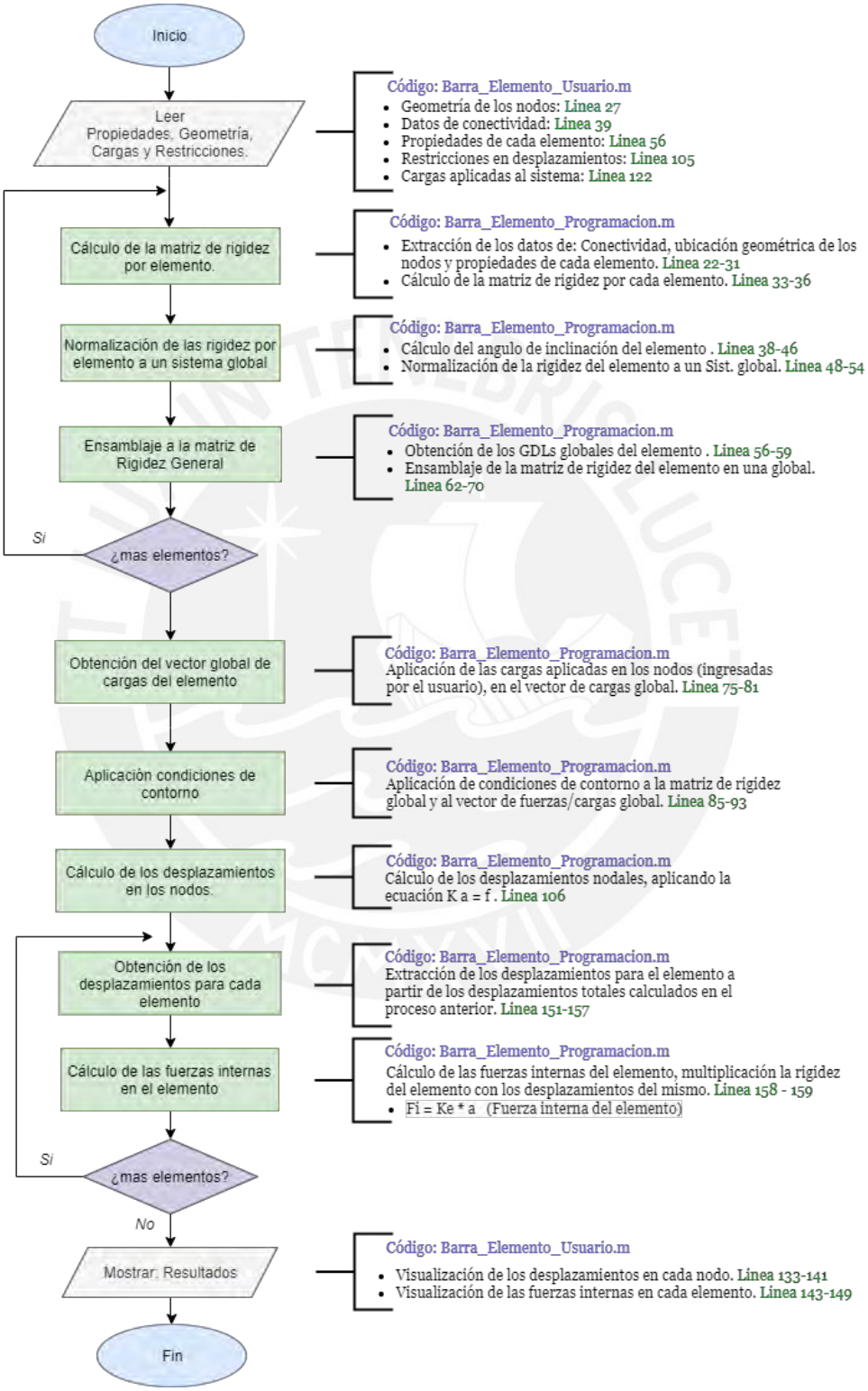


Figura 5.3 **Diagrama de flujo – Elemento tipo barra**

Fuente: Draw.io / Elaboración Propia

CAPÍTULO 6

Códigos base formulados en la investigación

Resumen

Se presentan las herramientas básicas para la elaboración de los códigos en los diversos tipos de elementos, junto a estas se detalla la composición (estructuración) de cada uno de los códigos incorporados en la presente tesis, describiéndose el ingreso y salida de datos.

CAPÍTULO VI: CÓDIGOS BASE FORMULADOS EN LA INVESTIGACIÓN

6.1. FUNCIONES Y SUBPROGRAMAS

Las funciones y subprogramas son parte fundamental en la estructuración de un programa complejo. Teniendo en cuenta la importancia de dichas herramientas en los subcapítulos siguientes se definirán el funcionamiento de las mismas, incorporando una lista con las principales funciones y subprogramas usadas en el código.

Se motiva al lector a revisar la lista de funciones y subprogramas conjunto a los algoritmos y sus códigos para entender la aplicación del FEM de forma ideal.

6.1.1. Funciones

La definición que brinda MathWorks (Equipo desarrollador de MatLab) es la siguiente: “Las funciones son archivos que pueden aceptar argumentos de entrada y devolver argumentos de salida. Los nombres del archivo y de la función deben ser iguales. Las funciones operan en variables dentro de su propia área de trabajo, en un espacio separado del área de trabajo a la que se accede desde la línea de comandos de MATLAB”.(Mathworks, 2016)

6.1.2. Subprogramas

Existen dos tipos de M-archivo, es decir, de archivos con extensión *.m (usada en el lenguaje de programación de MatLab). Un tipo de archivo son los ficheros de comandos (archivo script) y el otro son las funciones, las cuales se mencionaron anteriormente. “Un fichero de comandos contiene un conjunto de comandos que se ejecutan sucesivamente cuando se da la orden de ejecución del mismo en la ventana de comandos de Matlab o se incluye dicho nombre en otro fichero” (Fernández, 2009).

Estos archivos tipo M-files en el lenguaje de programación general son conocidos como programas especiales (subprogramas en la presente investigación), los cuales (con ayuda de las funciones) complementan todo el programa de forma eficaz. Una vez obtenidos los subprogramas se proceden a compatibilizar en uno general y compacto (programa master), el cual estaría listo a compilar y direccionaría a los demás subprogramas y subfunciones necesarias.

6.1.3. Lista de Funciones y Programas usados

A continuación, se detalla una lista de las principales subfunciones y subprogramas incorporados para los distintos elementos con una pequeña descripción de los mismos.

Se tiene que resaltar, que si bien el autor elaboro y optimizo cada uno de los códigos anexados, se usaron como referencia y punto de partida algunos publicaciones de diversos autores como son Kwon, Young W.(Bang & Kwon, 2000), Kattan, Petter Issa (Kattan, 2010) y Amar Khennane (Khennane, 2013).

- ***Barra_Elemento_Usuario.m:***

  Código para el ingreso de datos respecto a un sistema axialmente deformable / Articulado.

  Variables de Ingreso: Geometría del elemento (geom), conectividad (conect), propiedades del elemento (prop), Condiciones de frontera (nf), Fuerzas aplicadas al elemento (Carg).

  Variables de Salida: Graficas, Ordenamiento de resultados.

- ***Barra_Elemento_Programación.m:***

  Código para analizar un sistema axialmente deformable / Articulado mediante el FEM a partir de los datos ingresados por el usuario.

  Variables de Ingreso: Ninguno, se utilizan datos de la función: Barra_Elemento_Usuario.

  Variables de Salida: Vector de desplazamiento (desp), Vector de fuerzas (Fuerza).

  Nomenclatura del subprograma:

  [desp,Fuerza]=Barra_Elemento_Programacion(nnd,nel,conect,geom,prop,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL)

- ***Viga_TS_Usuario.m:***

  Código para el ingreso de datos respecto a un elemento tipo viga / Reticular bajo la formulación de Timoshenko.

  Variables de Ingreso: Geometría del elemento (geom), conectividad (conect), propiedades del elemento (prop), Condiciones de frontera (nf), Fuerzas aplicadas al elemento (Carg).

  Variables de Salida: Graficas, Ordenamiento de resultados.

- ***Viga_TS_Programación.m:***

  Código para analizar un elemento tipo viga / Reticular bajo la formulación de Timoshenko mediante el FEM a partir de los datos ingresados por el usuario.

  Variables de Ingreso: Ninguno, se utilizan datos de la función: Viga_TS_Usuario.

  Variables de Salida: Vector de desplazamiento (desp), Vector de fuerzas (Fuerza).

  Nomenclatura del subprograma:

  [desp,Fuerza]=Viga_TS_Programacion(nnd,nel,conect,geom,prop,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL)

- ***Viga_EB_Usuario.m:***

  Código para el ingreso de datos respecto a un elemento tipo viga / Reticular bajo la formulación de Euler Bernoulli.

  Variables de Ingreso: Geometría del elemento (geom), conectividad (conect), propiedades del elemento (prop), Condiciones de frontera (nf), Fuerzas aplicadas al elemento (Carg).

  Variables de Salida: Graficas, Ordenamiento de resultados.

- ***Viga_EB_Programación.m:***

  Código para analizar un elemento tipo viga / Reticular bajo la formulación de Euler mediante el FEM a partir de los datos ingresados por el usuario.

  Variables de Ingreso: Ninguno, se utilizan datos de la función: Viga_EB_Usuario.

  Variables de Salida: Vector de desplazamiento (desp), Vector de fuerzas (Fuerza).

  Nomenclatura del subprograma:

  [desp,Fuerza]=Viga_EB_Programacion(nnd,nel,conect,geom,prop,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL)

- ***Placa_Kirchhoff_Usuario.m:***

  Código para el ingreso de datos respecto a un elemento tipo Plate con la formulación de Kirchhoff.

  Variables de Ingreso: Geometría del elemento (geom), conectividad (conect), módulo de elasticidad (E), coeficiente de poisson (vu), espesor del elemento (Esp), condiciones de frontera (nf), Fuerzas aplicadas al elemento (Carg).

  Variables de Salida: Graficas, Ordenamiento de resultados.

- ***Placa_Kirchhoff_Programación.m:***

  Código para analizar un elemento tipo Plate bajo la formulación de Kirchhoff mediante el FEM a partir de los datos ingresados por el usuario.

  Variables de Ingreso: Ninguno, se utilizan datos de la función: Placa_Kirchhoff_Usuario.

  Variables de Salida: Vector de desplazamiento (desp), Vector de fuerzas (Element_Forces).

  Nomenclatura del subprograma:

  [desp,Element_Forces]= Placa_Kirchhoff_Programacion(nf,nne, nnd,nel,conect,geom,E,vu,Esp,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL)

- ***Placa_ ReissnerMindlin _Usuario.m:***

  Código para el ingreso de datos respecto a un elemento tipo Plate bajo la formulación de Reissner Mindlin.

  Variables de Ingreso: Geometría del elemento (geom), conectividad (conect), módulo de elasticidad (E), coeficiente de poisson (vu), espesor del elemento (Esp), condiciones de frontera (nf), Fuerzas aplicadas al elemento (Carg).

  Variables de Salida: Graficas, Ordenamiento de resultados.

- ***Placa_ReissnerMindlin_Programación.m:***

  Código para analizar un elemento tipo Plate bajo la formulación de Reissner Mindlin mediante el FEM a partir de los datos ingresados por el usuario.

  Variables de Ingreso: Ninguno, se utilizan datos de la función: Placa_ReissnerMindlin_Usuario.

  Variables de Salida: Vector de desplazamiento (desp), Vector de fuerzas (Element_Forces), Fuerzas a flexión para 2 puntos de integración (Element_Flex), Fuerzas a corte para 2 puntos de integración (Element:Corte).

  Nomenclatura del subprograma:

  [desp,Element_Forces,Element_Flex,Element_Corte]= Placa_ReissnerMindlin_Programacion(nf,nne,nnd,nel,conect,geom,E,vu,Esp,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL)

- ***Bi_TenPlana_Gauss_Usuario.m:***

  Código para el ingreso de datos respecto a un elemento bidimensional de tensión plana haciendo uso de la integración mediante Gauss.

Variables de Ingreso: Geometría del elemento (geom), conectividad (conect), módulo de elasticidad (E), coeficiente de poisson (vu), espesor del elemento (Esp), condiciones de frontera (nf), Fuerzas aplicadas al elemento (CargNod).

Variables de Salida: Graficas, Ordenamiento de resultados.

- ***Bi_TenPlana_Gauss_Programación.m:***

  Código para analizar un elemento bidimensional de tensión plana haciendo uso de la integración mediante Gauss a partir de los datos ingresados por el usuario.

  Variables de Ingreso: Ninguno, se utilizan datos de la función: Bi_TenPlana_Gauss_Usuario.

  Variables de Salida: Vector de desplazamiento (delta), Matriz de esfuerzos en el sistema (SIGMA).

  Nomenclatura del subprograma:

  [delta,SIGMA]= Bi_TenPlana_Gauss_Programacion(nf,ngp,nnd,nel, conect,geom,E,vu,Esp,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL)

- ***Tri_Elemento_Usuario.m:***

  Código para el ingreso de datos respecto a un elemento tridimensional haciendo uso de la integración mediante Gauss.

  Variables de Ingreso: Geometría del elemento (geom), conectividad (conect), módulo de elasticidad (E), coeficiente de poisson (vu), condiciones de frontera (nf), Fuerzas aplicadas al elemento (CargNod).

  Variables de Salida: Graficas, Ordenamiento de resultados.

- ***Tri_Elemento_Programación.m:***

  Código para analizar un elemento tridimensional haciendo uso de la integración mediante Gauss a partir de los datos ingresados por el usuario.

  Variables de Ingreso: Ninguno, se utilizan datos de la función: Tri_Elemento_Usuario.

  Variables de Salida: Vector de desplazamiento (delta), Matriz de esfuerzos en el sistema (SIGMA), esfuerzos a flexión para 2 puntos de integración (stress), Matriz de modificación de conectividad (conectesf).

  Nomenclatura del subprograma:

  [delta,SIGMA,stress,conectesf]=Tri_Elemento_Programacion(nf,nglx,ngly, nglz,nnd,nel,conect,geom,E,vu,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL)

6.2. POS PROCESAMIENTO - GRAF. DE RESULTADOS

La fase de pos procesamiento de datos es vital en el FEM, ya que es en la cual se exportan los resultados y realizan gráficas. Las graficas son importantes porque ayudan al diseñador a visualizar los desplazamientos, tensiones y deformaciones del sistema, comprendiendo rápidamente las zonas críticas del mismo. Para graficar los resultados se hace uso de: La geometría de los nodos para generar la malla y los datos de desplazamientos, deformaciones o tensiones para visualizar la variación de los mismos a lo largo de todo el sistema.

En la figura 6.1 se puede apreciar a manera de ejemplo una gráfica de desplazamientos para una escalera helicoidal, la cual se trabajó con elementos tridimensionales y se detalla en el capítulo 8. La grafica tiene en cuenta un colormap (“mapa de colores”) con degradación continua para una mejor presentación.

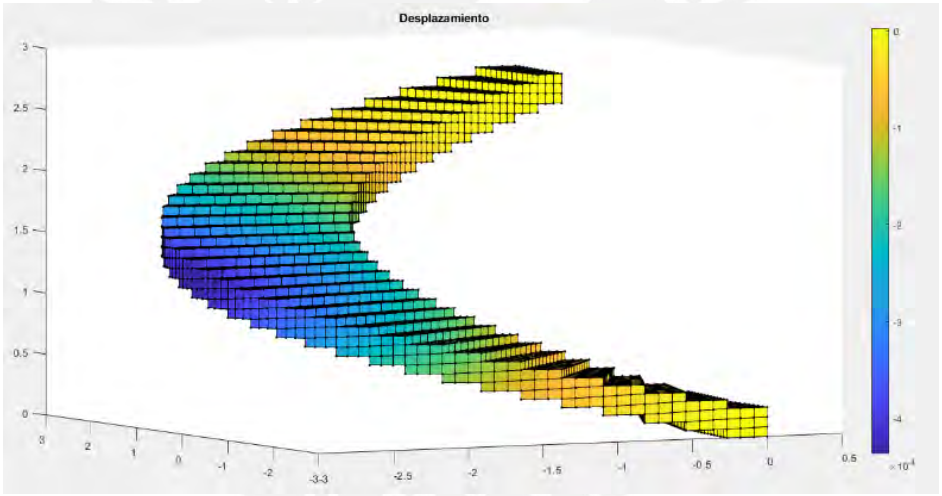


Figura 6.1 **Desplazamientos en Escalera Helicoidal (Forma Papelillo)**

Fuente: MatLab/ Código-Elaboración Propia

6.2.1. Herramientas para la Visualización de Resultados

La obtención de la figura 6.1 se realizó haciendo uso de un conjunto de herramientas que brinda MatLab, las cuales fueron orientadas para un trabajo conjunto entre los resultados y la geometría del sistema. Estas herramientas se detallan a continuación:

- ***Patch: patch('Faces',F,'Vertices',V)***

  Herramienta de MatLab que permite crear polígonos a partir de la geometría y conectividad que existe entre los mismos. Junto a esta herramienta se puede añadir un rango de valores en cada uno de los puntos, los cuales se promediarán y generarán capas con la variación/degradación de datos en forma gráfica entre los distintos nodos. En la figura 6.2 se aprecia un ejemplo bidimensional de esta herramienta.

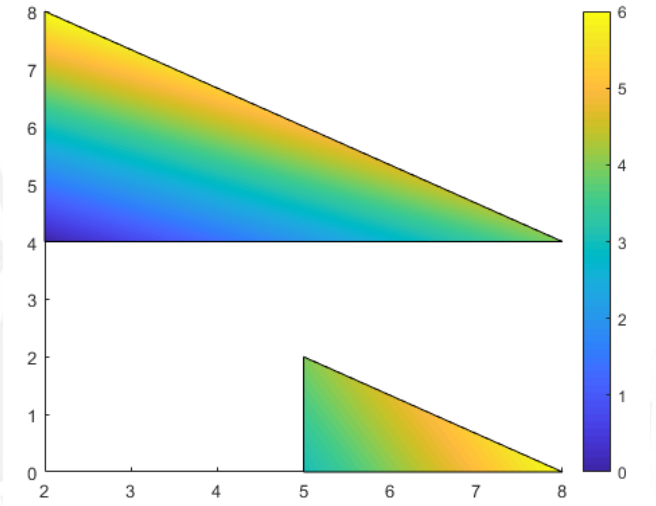


Figura 6.2 **Ejemplo de la herramienta patch**

Fuente: (Mathworks, 2011)

- ***Colorbar:***

  Herramienta que muestra la escala de colocar aplicada en gráficos, esta se puede personalizar en función del rango estipulado por el usuario, siendo definido no necesariamente por un rango numérico. Siendo el análisis del FEM un método orientado a obtener precisión en resultados no se modificarán propiedades de la herramienta, otorgando resultados que muestren los limites obtenidos. En la figura 6.2 se visualiza la aplicación de dicha herramienta a través de: La barra con la numeración que indica el valor para el color mostrado y el tipo de color asignado para la gráfica de azul a amarillo (por defecto).

6.3. APLICACIÓN DEL CÓDIGO

En este subcapítulo se especifica la forma de ingreso de datos a los diversos códigos presentados en la tesis. Se debe resaltar que todo ingreso de información está en el código con terminología Usuario, por ejemplo “Barra_Elemento_Usuario.m”. Los códigos con terminología Programación son los que guardan la metodología de análisis del FEM, siendo enlazados y usados automáticamente.

6.3.1. Elemento Barra

El código está especificado para elementos de 2 nodos por elemento.

- ***Geometría / Propiedades***

```
% Ingreso de Coordenadas de los Nodos. (Geométria)
% Coordenadas X/Y por Nodo
geom = [0. 0.; ...
        3. 0.; ...
        6. 0.];
% Ingreso de datos de conectividad entre nodos. (Conectividad)
% Conectividad Nodo Inicial/ Nodo Final por Elemento
conect = [1 2;...
          2 3];
% Propiedades Geométricas del Material por Barra (Propiedades)
% Propiedades: Modulo de Elasticidad/ Área por Elemento
prop = [200*10^9 0.001;...
        200*10^9 0.001];
```

- ***Cargas / Restricciones***

```
%  Aplicando Restricciones a los GDLs de los nodos
% nf=(Num. de nodo,Col):
% Col = 1 --- Restricción en X
%     = 2 --- Restricción en Y
nf(1,1) = 0;
nf(1,2) = 0;
nf(3,2) = 0;
% Aplicacion de la carga en los nodos del Sistema
% Carg(Num. de nodo,:)=[Carga en X   Carga en Y]
Carg(3,:)=[30000. 0.];      % Aplicando Cargas a los Nodos
```

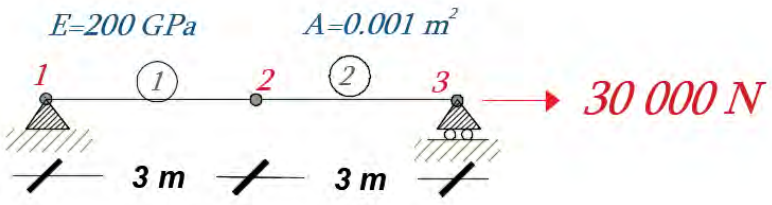


Figura 6.3 **Ejm. Introducir data - Barra**

Fuente: Elaboración Propia

6.3.2. Elemento Viga

El código está especificado para elementos de 2 nodos por elemento y se cuenta con dos códigos debido a las diferentes formas de análisis (Timoshenko, Bernoulli), se debe resaltar que la introducción de data para ambos es la misma por lo que se detallara el ingreso de data en forma general.

- ***Geometría / Propiedades***

```
%   Ingreso de coordenadas de los nodos. (Geometría)
geom = [0.; ...     % Coordenadas en X del elemento
        3.0];

% Ingreso de datos de conectividad entre nodos. (Conectividad)
% Conectividad Nodo Inicial/ Nodo Final por Elemento
conect = [1 2] ;

%   Propiedades del sistema (Elasticidad/Inercia) por elemento
prop = [200*10^9  0.0072];
```

- ***Cargas / Restricciones***

```
%  Aplicando Restricciones a los GDLs de los nodos
% nf=(Num. de nodo,Col):
% Col = 1 --- Restricción en Y
%     = 2 --- Restriccion en el giro
nf(1,1) = 0;    % Aplicando restricciones en los GDLs
nf(1,2) = 0;
%      Fuerzas aplicadas al sistema
% CargNod(Nodo,:)=[Carga en Y  Momento]
CargNod(2,:) = [30000 0];
```

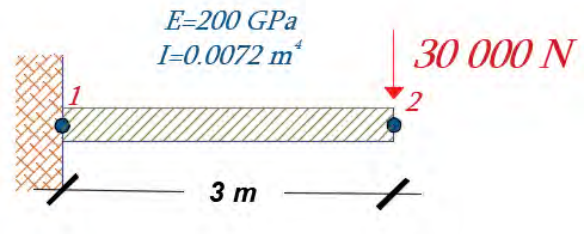


Figura 6.4 **Ejm. Introducir data - Viga**

Fuente: Elaboración Propia

6.3.3. Elemento tipo Placa

El código está especificado para elementos de 4 nodos por elemento y se cuenta con dos códigos debido a las diferentes formas de análisis (Kirchhoff, R. Mindlin), se debe resaltar que la introducción de data para ambos es la misma por lo que se detallara el ingreso de data en forma general.

- ***Geometría / Propiedades***

```
% Ingreso de Coordenadas de los Nodos. (Geométria)
% Coordenadas X/Y por Nodo
geom = [0.0  0.0;...    % Coordenadas en X,Y del Elem. Plate.
        2.0  0.0;...
        2.0  2.0;...
        0.0  2.0];
% Ingreso de datos de conectividad entre nodos. (Conectividad)
% Conectividad de 4 nodos por Elemento. (Análisis de elemento cuadrilatero de 4 nodos).
conect = [1 2 3 4];
%   Propiedades del material (Elasticidad/Poisson/Espesor)
E=200*10^9;     % Modulo de elasticidad del material
vu=0.3;         % Coeficiente de Poisson del material
Esp=0.10;       % Espesor del elemento plate
```

- ***Cargas / Restricciones***

```
%  Aplicando Restricciones a los GDLs de los nodos
% Fila = Num. de nodo
% Col = 1 --- Restricción en Z
%     = 2 --- Giro en X
%     = 3 --- Giro en Y
nf(1,1) = 0; nf(1,2)=0;  nf(1,3)=0;         nf(2,1) = 0; nf(2,2)=0;  nf(2,3)=0;
nf(3,2) = 0; nf(3,3)=0;                     nf(4,1) = 0; nf(4,2)=0;  nf(4,3)=0;
```

```
% Aplicacion de la carga en los nodos del Sistema
% Carg(Num. de nodo,:)=[Carga en Z]
Carg(3,1) = 50;              % Aplicando cargas a los nodos
```

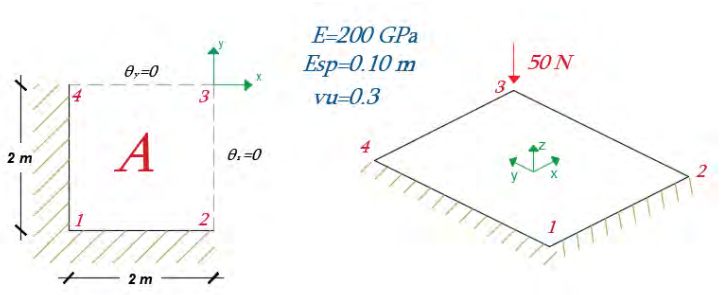


Figura 6.5 **Ejm. Introducir data - Placa**

Fuente: Elaboración Propia

6.3.4. Elemento Bidimensional

El código está especificado para elementos de 4 nodos por elemento y se cuenta con dos códigos debido a las diferentes formas de análisis (Gauss, Integración directa), se debe resaltar que la introducción de data para ambos es la misma por lo que se detallara el ingreso de data en forma general.

- ***Geometría / Propiedades***

Para el ingreso de data se recomienda usar archivos de texto, ya que este tipo de análisis conlleva en su mayoría un gran número de nodos como se muestra a continuación:

```
%   Ingreso de coordenadas de los nodos(Geométria) a partir de archivos txt.
dataSis='Int_Data_Coord.txt';
geom = importdata(dataSis);
```

```
%   Ingreso de datos de conectividad entre nodos(Conectividad) a partir de archivos txt.
dataSis='Int_Data_Conec.txt';
conect = importdata(dataSis);
```

Los archivos de texto contienen una data como se muestra a continuación (respetando el espaciado con tabulaciones):

	Int_Data_Coord.txt		
	N° de Nodo	***Coord X***	***Coord Y***
Data que se ingresa ->	***1***	***0***	***0***
	2	***2***	***0***
	3	***4***	***0***
	4	***0***	***1***
	5	***2***	***1***
	6	***4***	***1***

	Int_Data_Conec.txt				
	N° de Elemento	***Nodos (globales) según elemento analizado***			
Data que se Ingresa ->	***1***	***1***	***2***	***4***	***5***
	2	***2***	***3***	***5***	***6***

El ingreso de datos de las propiedades no varía y se muestra a continuación:

```
%   Propiedades del sistema
E = 2.535*10^10;        % Módulo de elasticidad del sistema
vu = 0.20;              % Coeficiente de Poisson
thick = 0.40;           % Espesor del sistema
```

- ***Cargas / Restricciones***

```
% Ingreso de restricciones Semiautomático
% i = Nodos restringidos
i=[1, 4];
[a]=size(i,1);
for ios=1:a
    % nf(Nodo,Direc)
    % Direc = Dirección de la restriccion
    % Direc = 1 --- Restricción en X
    % Direc = 2 --- Restricción en Y
    nf(i,1)=0;
    nf(i,2)=0;
end
```

```
% Ingreso de carga Semiautomático
% P = Magnitud de la carga
% i = Nodos donde se aplicara la carga
% Direc = Direccion de aplicación de la carga
       % Direc = 1 --- Carga en X
       % Direc = 2 --- Carga en Y
P=-15000;
Direc=2;
i=[3, 6];
[a]=size(i,1);
for ios=1:a
    % CargNod(Nodos,Col)=Carga
    % Col = 1 --- Restricción en X
    %     = 2 --- Restricción en Y
    CargNod(i,Direc)=P;
end
```

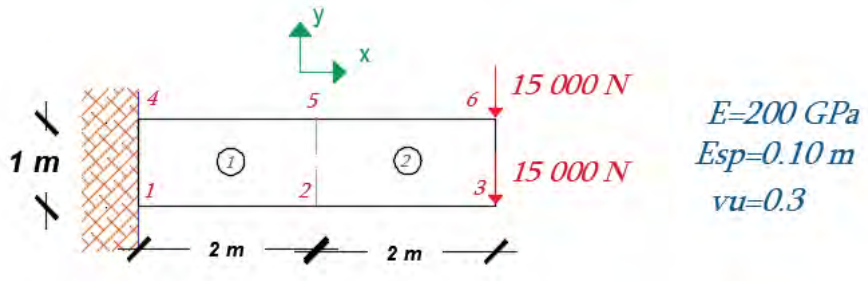


Figura 6.6 **Ejm. Introducir data – E. Bidimensional**

Fuente: Elaboración Propia

6.3.5. Elemento Tridimensional

El código está especificado para elementos de 8 nodos y 20 nodos, por lo que se cuenta con dos códigos, se debe resaltar que la introducción de data para ambos es la misma por lo que se detallara el ingreso de data en forma general.

- ***Geometría / Propiedades***

Para el ingreso de data se recomienda usar archivos de texto, ya que este tipo de análisis conlleva en su mayoría un gran número de nodos como se muestra a continuación:

```
%   Ingreso de coordenadas de los nodos(Geométria) a partir de archivos txt.
dataSis='Int_Data_Coord.txt';
geom = importdata(dataSis);
```

```
%   Ingreso de datos de conectividad entre nodos(Conectividad) a partir de archivos txt.
dataSis='Int_Data_Conec.txt';
conect = importdata(dataSis);
```

Los archivos de texto contienen una data como se muestra a continuación (respetando el espaciado con tabulaciones):

	Int_Data_Coord.txt			
	N° de Nodo	***Coord X***	***Coord Y***	***Coord Z***
Data que se ingresa ->	***1***	***0***	***0***	***0***
	2	***1***	***0***	***0***
	3	***1***	***1***	***0***
	4	***0***	***1***	***0***
	5	***0***	***0***	***1***
	6	***1***	***0***	***1***
	7	***1***	***1***	***1***
	8	***0***	***1***	***1***

	Int_Data_Conec.txt								
	N° de Elemento	***Nodos (globales) según elemento analizado***							
Data que se Ingresa ->	***1***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***

Si se está realizando un análisis de 20 nodos por elemento, el número de nodos (globales) que se ingresará a la conectividad será 20. El ingreso de datos de las propiedades no varía y se muestra a continuación:

```
%   Propiedades del sistema
E = 200*10^9;           % Módulo de elasticidad del sistema
vu = 0.30;              % Coeficiente de Poisson
```

- ***Cargas / Restricciones***

```
% Ingreso de restricciones Semiautomático
% i = Nodos restringidos
i=[1, 2, 3, 4]; % Vector con los nodos con GDLs restringidos
[a]=size(i,1);  % Asignación de los GDLs restringidos por nodo
for ios=1:a
    % nf(Nodo,Direc)
    % Direc = Dirección de la restriccion
    % Direc = 1 --- Restricción en X
    % Direc = 2 --- Restricción en Y
    % Direc = 3 --- Restricción en Z
    nf(i,1)=0;
    nf(i,2)=0;
    nf(i,3)=0;
end
```

```
% Ingreso de carga Semiautomático
% P = Magnitud de la carga
% i = Nodos donde se aplicara la carga
% Direc = Direccion de aplicación de la carga
       % Direc = 1 --- Carga en X
       % Direc = 2 --- Carga en Y
       % Direc = 3 --- Carga en Z
P=-250;
i=[5, 6, 7, 8]; % Nodos en las esquinas
Direc=3;
[a]=size(i,1);
for ios=1:a
    CargNod(i,Direc)=P;
end
```

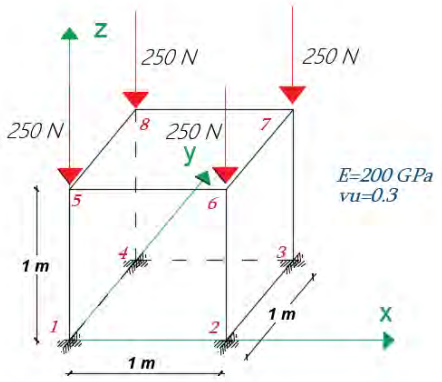


Figura 6.7 **Ejm. Introducir data – E. Tridimensional**

Fuente: Elaboración Propia

CAPÍTULO 7

Optimización del Software

Resumen

Se presentan las herramientas básicas para la optimización de un código, con respecto al tiempo de procesamiento de datos. Al aplicarse dichas herramientas en la elaboración del presente código, se muestra una comparativa del código sin optimizar (primeras fases de desarrollo) contra uno optimizado (código final).

CAPÍTULO VII: OPTIMIZACIÓN DEL CÓDIGO

7.1. BASES EN LA OPTIMIZACIÓN DEL CÓDIGO

En la programación el término optimización está referido al proceso de mejora del código para utilizar menos recursos del sistema, con lo cual se logra reducir el tiempo de procesamiento de datos. Cuando desarrollamos un código muchas veces incluimos la optimización en las diversas fases del mismo sin darnos cuenta. Por ejemplo, al principio los programadores suelen elaborar un código base para ver si logra cumplir los objetivos principales planteados (borrador), una vez cumplida la meta inicial se procede a replantear y reorganizar ciertas partes del código para hacerlo más comprensible y utilizar herramientas más directas, dicho proceso es la base de la optimización de un código.

Se debe tener en cuenta que una parte importante de la optimización es la generación de lazos para la entrada o salida de información a través subfunciones o subprogramas, teniendo que evaluarse en muchos casos si es necesaria tal subdivisión del código. Otro punto en específico es la exportación final de datos (resultados) y generación de gráficas, ya que su uso requiere una gran cantidad de recursos del ordenador para sistemas complejos, por lo que es necesario tener en cuenta la herramienta más óptima que se pueda aplicar para su generación.

A lo largo de este capítulo se mostrarán diversas formas de optimización del código en base al llamado de la información, uso correcto de herramientas (bucles), entre otros, todos estos en función al lenguaje de programación propio de MatLab.

7.1.1. Optimización en la formulación del código

La optimización en la formulación del código es la más básica y se da automáticamente cuando el programador cuenta con experiencia en la elaboración de códigos. En la presente investigación se quiere dar pautas básicas, las cuales estén orientadas al proceso de sistematizar un protocolo (uso de bucles), ya que es una herramienta de uso común e imprescindible en el FEM.

Se procede a mostrar tres pautas básicas que se deben tener en cuenta para la formulación de un código que incluya bucles:

- ***Valor constante:*** Cuando se cuenta con un valor en una línea de código que no varía en cada iteración se le denomina “valor constante”. Dicho valor causara la generación de una variable que lo acoja en cada ciclo, haciendo uso innecesario de recursos del ordenador. Por dicho motivo se debe orientar a inicializar dichas variables antes de dar inicio a los diversos bucles en el código.
- ***Valor redundante:*** Cuando se tiene una variable que varíe dentro de un sub-bucle, el cual este sometido a otro bucle con la misma variable de valor constate, se le denomina “valor redundante”. Esta mención se da ya que su calculó es redundante e injustificado al modificarse dicha variable en el bucle general y perdiendo toda data que almacene. Dicho uso de variable en el sub-bucle se puede evitar, al no llevarlo a cabo, solamente inicializando la variable en el bucle general.
- ***Flujo de datos:*** El flujo de datos (data-flow), es referido a la optimización con respecto a la eficiencia de los diversos cálculos realizados. Esto en palabras sencillas se da respecto al correcto planteamiento y uso óptimo de la variable, bucle, condicional, entre otras herramientas que se estén trabajando. El flujo de datos debe ser constante y se deberán eliminar líneas de código innecesarias, produciendo en muchos casos una reformulación completa del mismo.

Las pautas mencionadas anteriormente parecerán innecesarias si se analiza individualmente un ciclo del bucle, ya que solo se reducirán décimas o, centésimas de segundo su tiempo de análisis. Sin embargo, al repetirse el bucle un gran número de veces para sistemas complejos su optimización causará una gran reducción en el tiempo, como se apreciará posteriormente.

7.1.2. Optimización en el llamado de las funciones

El uso del tipo de función puede afectar drásticamente el tiempo de ejecución del código, ya que el tratamiento de información se da por medio de estas. El Staff de Mathworks (MathWorks Support Team) reporta en su foro de ayuda información sobre el mejor rendimiento entre las distintas funciones (MathWorks, 2018), las cuales se clasificaron como se muestra a continuación:

Tabla 4. **Rendimiento en el llamado de las funciones**

Fuente: Elaboración Propia

Orden de Rendimiento	***Función***	***Definición***
1	Inlined function	Cuando el autor reemplaza la llamada de la función con una copia del cuerpo de la misma.
2	File pass function	La función es definida en un archivo separado de MatLab. Los argumentos son transferidos llamando la función.
2	Nest pass function	Los argumentos son transferidos reemitiendo la función.
2	Sub pass function	Sub-función en donde los argumentos son transferidos llamando la función. (sub-transferencia)
3	Nest share function	Los argumentos son transferidos/buscados remitiendo la función, solamente los índices son provistos por la función transferida.
4	Sub global function	Los argumentos son provistos con variables globales, solamente los índices son provistos con el llamado de la función.
5	File global function	La función es definida en un archivo separado de MatLab. Los argumentos son provistos con el llamado de variables globales.

En la clasificación vista se visualiza que las funciones que trabajan directamente con el código, tienen un menor tiempo de procesamiento de datos a diferencia de las que trabajan con data en forma global. Por lo cual las primeras son una medida de optimizar el código usada en la presente investigación.

7.1.3. Redundancia en el Código

La redundancia en el código se da mayormente en las primeras fases de su formulación. Mediante el avance, recodificación y la optimización del mismo este código redundante se va optimizando, sin embargo, hay que tener en cuenta definiciones básicas de tipos de redundancia que existen en los codigos.

- ***Redundancia Total:*** También denominado “código muerto”. Se refiere a código que nunca es ejecutado (debido a que las condiciones que lo requieran nunca se cumplen) o que se ejecuta, pero sus resultados nunca se utilizan.
- ***Redundancia Parcial:*** La redundancia parcial está referida a partes de un código que se calculan más de una vez (sobrescribiéndolas) sin hacer uso de los datos que contenga. Por ejemplo, el bucle mostrado a continuación donde la función se calcula más de una vez (si se cumple la condicional). Posteriormente se visualiza el mismo ejemplo optimizando.

Código con Redundancia Parcial

```
if a>b
    [c] = Barra_Elemento_Programacion(nnd);
else
    ...
end
[d] = Barra_Elemento_Programacion(nnd);
```

Código Optimizado

```
[var] = Barra_Elemento_Programacion(nnd);
if a>b
    c=var;
else
    ...
end
d=var
```

7.1.4. La Optimización en el FEM

El proceso de optimizar un código basado en el FEM se oriente primordialmente a obtener un flujo continuo y eficaz de data en los bucles que analizan cíclicamente cada uno de los elementos. Teniendo en cuenta que en gran parte de los sistemas se harán uso de variables con una gran cantidad de información (matriz de rigidez), la cual si se sobrescribe innecesariamente causara un aumento radical en el tiempo de procesamiento de datos.

La optimización de tan solo una variable o un solo sub-bucle dentro de un bucle mayor, generará grandes resultados (como se mostrará posteriormente). Se debe resaltar que dicha optimización da como resultado un menor tiempo de procesamiento acompañado de un menor uso de recursos en el ordenador y una mejor compresión del código en forma general. Por tales motivos es vital aplicar por lo menos las técnicas básicas descritas anteriormente.

7.2. COMPARATIVA CÓDIGO SIN OPTIMIZAR VS OPTIMIZADO

Ya que el código de la presente investigación ha sido sometido a un gran número de cambios, aplicando dichas herramientas de optimización. Se presenta una comparativa del código sin optimizar (primeras fases de desarrollo) contra el código final optimizado (anexado en la presente tesis) obteniéndose los mismos resultados. El ejemplo a comparar se basa en la obtención de esfuerzos en una escalera helicoidal, trabajada como ejemplo más adelante (subcapítulo 8.4).

Tabla 5. **Comparativo código sin optimizar / código optimizado**

Fuente: Elaboración Propia

	Código sin Optimizar	Código Optimizado
Tiempo de obtención de resultados.	5946.726 seg 99.11 min	95.094 seg 1.585 min
Número de líneas de código de la función de procesamiento	415 líneas	103 líneas

En la tabla 5 se aprecia que la optimización genera una disminución considerable en el tiempo de procesamiento de datos, apreciable en este caso por ser un sistema con un gran número de elementos (3888 elementos). Tal nivel de optimización es debido a iterar un gran número de pruebas, en las cuales se modificó la forma de plantear el código, tipos de varíales, entre otros.

Si se desea visualizar cada uno de los cambios aplicados, se sugiriere visualizar el código encontrado en el Anexo 3: Código de la comparativa. A continuación, se muestra una parte de los resultados de la herramienta “Profile” incorporado a MatLab, para visualizar a más detalle los resultados.

Profile Summary

Generated 08-Sep-2019 22:00:38 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Tri_Elem_ProgBase_Helicoidal	1	5946.726 s	1007.761 s	
m_rigidglobal	3888	4931.434 s	4931.434 s	
restring_rigid	1	2.252 s	2.252 s	

Figura 7.1 **Parte del resumen herramienta Profile (C. sin optimizar)**

Fuente: MatLab/ Profile-Elaboración Propia

Profile Summary

Generated 08-Sep-2019 22:37:32 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Tri_Elem_Usua_Helic	1	95.094 s	20.762 s	
Tri_Elem_Prog_Helic	1	73.719 s	70.221 s	
Elem3D_8nodos	34992	1.266 s	1.266 s	

Figura 7.2 **Parte del resumen herramienta Profile (C. optimizado)**

Fuente: MatLab/ Profile -Elaboración Propia

CAPÍTULO 8

Aplicación del FEM en sistemas estructurales

Resumen

En este capítulo se resuelven ejemplos aplicativos referentes a diversos sistemas estructurales. La obtención de resultados se da a través del código incorporado en la presente investigación, comparando la precisión de los mismos con programas de ordenador y solución de diversos autores.

CAPÍTULO VIII: APLICACIÓN DEL CÓDIGO EN SISTEMAS ESTRUCTURALES

8.1. ESTRUCTURA ARTICULADA

Se procederá a resolver el ejemplo 7.2 del libro de Análisis Estructural de Gianfranco Ottazzi (Ottazzi Pasino, 2011). El motivo es comparar resultados con respecto al análisis que realiza dicho autor mediante Trabajo Virtual y el efectuado por nuestro código mediante el FEM. El código del ejemplo se encuentra en los anexos del presente informe (Anexo 04 – Ej. Aplicativos).

Enunciado: Resolver la armadura articulada de sección transversal uniforme, sometida a una carga puntual de 3 toneladas como se muestra en la figura .8.1. Calcular los desplazamientos del sistema mediante el uso del código Barra_Elemento_Usuario.m y Barra_Elemento_Programacion.m.

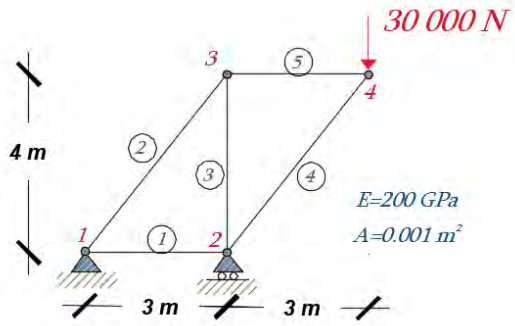


Figura 8.1 **Ejemplo - Estructura Articulada**

Fuente: Elaboración Propia

Ingresamos los datos sobre propiedades del material, ubicación geométrica de nodos y conexiones en el código Barra_Elemento_Usuario.m. Un fragmento sobre las condiciones de los elementos se procede a mostrar a continuación:

```
%   Ingreso de Coordenadas X/Y por Nodo. (Geometria)
geom = [0. 0.; ...
        3. 0.; ...
        3. 4.; ...
        6. 4.];

%   Ingreso de Datos de Conectividad Nodo Inicial/Nodo Final por Elemento
conect = [1 2;...
          1 3;...
          2 3;...
          2 4;...
          3 4];
```

```
% Propiedades Geométricas del Material por Barra (Propiedades)
% Propiedades: Modulo de Elasticidad/ Área por Elemento
prop = [200*10^9 0.001;...
        200*10^9 0.001;...
        200*10^9 0.001;...
        200*10^9 0.001;...
        200*10^9 0.001];
```

Las restricciones se encuentran en los nodos 1 y 2, la carga se encuentra en el nodo 4. Teniendo en cuenta el proceso general de aplicación del código (subcapítulo 6.3), se ingresarán las restricciones/cargas al código como se muestra a continuación:

```
%  Aplicando Restricciones a los GDLs de los nodos
% Fila = Num. de nodo
% Col = 1 --- Restricción en X
%     = 2 --- Restricción en Y
nf(1,1) = 0;
nf(1,2) = 0;
nf(2,2) = 0;
```

```
% Aplicacion de la carga en los nodos del Sistema
% Carg(Num. de nodo,:)=[Carga en X   Carga en Y]
Carg(4,:)=[0. 30000.];      % Aplicando Cargas a los Nodos
```

Nótese que cada uno de los códigos de ingreso de datos está enlazado automáticamente con los códigos de ejecución del FEM para cada tipo de elemento (Barra_Elemento_Usuario.m /Barra_Elemento_Programacion.m).

Una vez ingresados los datos del problema, procedemos a ejecutar el código para obtener resultados del análisis mediante el FEM. El código exporta automáticamente datos de esfuerzos, deformaciones y desplazamientos en los diversos nodos del sistema en forma numérica. Conociendo los resultados numéricos realizamos una comparativa de los desplazamientos obtenidos mediante el código con los que obtiene Ottazzi, pudiendo visualizar una gran similitud en los resultados.

Tabla 6. **Comparativa. Ejemplo / Estructura Articulada**

Fuente: Elaboración Propia

	Gianfranco Ottazzi		***Código FEM MatLab***	
Nodo	$u(x)(m)$	$u(y)(m)$	$u(x)(m)$	$u(y)(m)$
1	0	0	0	0
2	$-3.38 * 10^{-4}$	0	$-3.375 * 10^{-4}$	0
3	$2.363 * 10^{-3}$	$-6.0 * 10^{-4}$	$2.3625 * 10^{-3}$	$-6.0 * 10^{-4}$
4	$2.7 * 10^{-3}$	$-3.45 * 10^{-3}$	$2.7 * 10^{-3}$	$-3.45 * 10^{-3}$

8.2. LOSA EMPOTRADA CON CARGA CENTRAL

Se procederá a resolver el ejemplo de losa ante carga puntual del libro de Losas de Concreto Armado de Juan Bariola Bernales (Bernales, 1993). El motivo del ejemplo es comparar resultados con respecto al análisis que realiza dicho autor y el efectuado por nuestro código mediante el FEM. El código del ejemplo se encuentra en el Anexo 4 del presente informe.

Enunciado: Una losa cuadrada empotrada en sus extremos está sujeta a una carga concentrada en su centro (figura 8.2). Las dimensiones de la placa son 4m x 4m, con un espesor de 0.1 m. El módulo de Elasticidad es de 200 GPa, con coeficiente de Poisson's de 0.3. La carga concentrada es de 50 N. Calcular los desplazamientos y esfuerzos del sistema mediante el uso del Código Plate_Kirchhoff_Usuario.m y Plate_Kirchoff_Programacion.m.

Debido a la simetría en la losa, se puede dividir la misma en 4 elementos, teniendo en cuenta las condiciones de borde equivalentes mostradas en la figura 8.2; todo esto conlleva una optimización en el número de cálculos correspondientes.

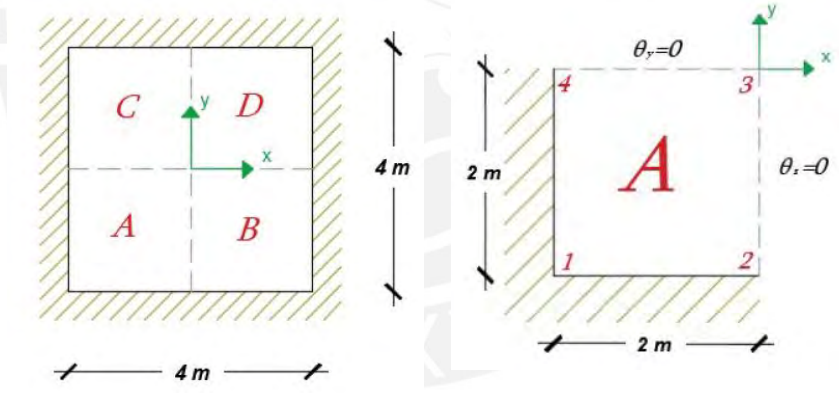


Figura 8.2 **Ejemplo de losa empotrada. Modelo general / simplificado**

Fuente: Elaboración Propia

Ingresamos los datos sobre propiedades del material, coordenadas geométricas y conectividad en el código Plate_Kirchhoff_Usuario.m. Un fragmento de dicho código se procede a mostrar a continuación:

```
% Ingreso de Coordenadas de los Nodos. (Geométria)
% Coordenadas X/Y por Nodo
geom = [0.0  0.0;...    % Coordenadas en X,Y del Elem. Plate.
        2.0  0.0;...
        2.0  2.0;...
        0.0  2.0];
% Ingreso de datos de conectividad entre nodos. (Conectividad)
% Conectividad de 4 nodos por Elemento. (Análisis de elemento cuadrilatero
```

```
% de 4 nodos).
conect = [1 2 3 4];

%   Propiedades del material (Elasticidad/Poisson/Espesor)
E=200*10^9;     % Modulo de elasticidad del material
vu=0.3;         % Coeficiente de Poisson del material
Esp=0.10;       % Espesor del elemento plate
```

Las restricciones se encuentran en los nodos 1, 2 y 4 (empotrados), la carga se encuentra en el nodo 3. A continuación se muestra el ingreso de esta data en el código:

```
%  Aplicando Restricciones a los GDLs de los nodos
% Fila = Num. de nodo
% Col = 1 --- Restricción en Z
%     = 2 --- Restricción en X
%     = 2 --- Restricción en Y
nf(1,1) = 0; nf(1,2)=0;  nf(1,3)=0;
nf(2,1) = 0; nf(2,2)=0;  nf(2,3)=0;
nf(3,2) = 0; nf(3,3)=0;
nf(4,1) = 0; nf(4,2)=0;  nf(4,3)=0;
```

```
% Aplicacion de la carga en los nodos del Sistema
% Carg(Num. de nodo,:)=[Carga en Z]
Carg(3,1) = 50;              % Aplicando cargas a los nodos
```

Una vez ingresados los datos del problema, procedemos a ejecutar el código para obtener resultados del análisis mediante el FEM. El código exporta automáticamente datos de esfuerzos, deformaciones y desplazamientos en los diversos nodos del sistema en forma numérica y gráfica. Una de las gráficas se muestra en al Fig.8.3, la cual muestra los desplazamientos en Z que sufre el cuarto de losa en toda su dimensión.

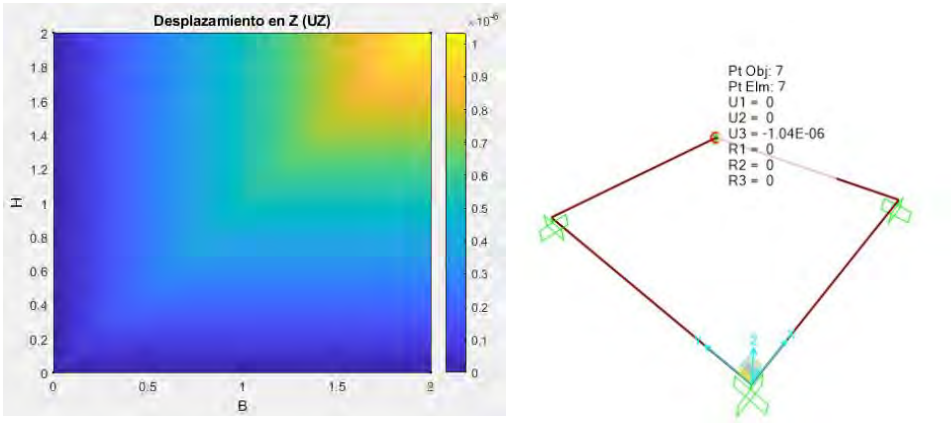


Figura 8.3 **Desplazamiento Uz / losa empotrada**

Fuente: MatLab / Código-Elaboración Propia

Conociendo los resultados numéricos realizamos una comparativa del desplazamiento en el nodo central obtenido mediante el código, con los que obtiene Juan Bariola mediante el FEM simplificado y la respuesta exacta.

Tabla 7. **Comparativa. Ejemplo / Elemento Plate - Kirchhoff**

Fuente: Elaboración Propia

	Nodo Central
Fuente	$u(z)(m)$
Código FEM	$1.034 * 10^{-6}$
Juan Bariola	$1.034 * 10^{-6}$
Timoshenko	$1.034 * 10^{-6}$
SAP 2000	$1.04 * 10^{-6}$

8.3. VIGA UNI, BI Y TRIDIMENSIONAL

En este ejemplo aplicativo se realizará una comparación de las diversas formas de análisis que se pueden aplicar a una viga simplemente apoyada (análisis uni, bi y tridimensional), realizando una verificación y comparación de resultados con el software SAP2000 y Abaqus. El código del ejemplo se encuentra en el Anexo 4 (Código – Viga Multidimensional) y Anexo 5 (Data).

Enunciado: Realizar una comparativa de las formas de análisis de una viga simplemente apoyada bajo una formulación uni, bi y tridimensional con una carga distribuida de $6000\ N/m$ para un elemento unidimensional y $15000\ N/m^2$ para la aplicación de la carga sobre área. El modelo de la viga se muestra en la figura 8.4, en donde se incluye la sección transversal de la misma. Siendo el módulo de elasticidad 25.35 GPa y el coeficiente de Poisson 0.20 obtener los esfuerzos, desplazamientos y momento máximo generado debido a la solicitación mediante el uso de los diversos códigos anexados, realizando un análisis comparativo de resultados.

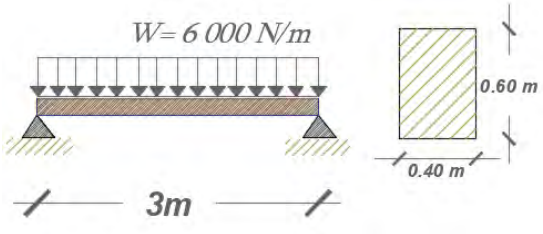


Figura 8.4 **Modelo Viga S.A. con Sección Transversal**

Fuente: AutoCAD/ Elaboración Propia

Para el caso unidimensional hacemos uso de los códigos Viga_EB_Usuario.m y Viga_EB_Programacion.m. Se dividió la viga en dos secciones para obtener el desplazamiento y momento en el medio de la misma, como se muestra en la figura 8.5.

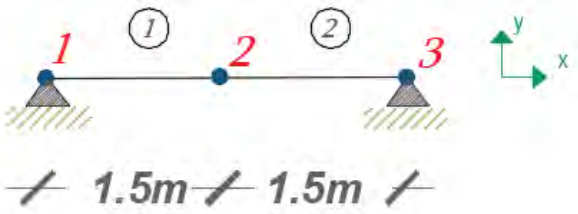


Figura 8.5 **Viga, análisis unidimensional**

Fuente: AutoCAD/ Elaboración Propia

A continuación, se muestran fragmentos del código correspondientes al ingreso de data respecto a la geométrica, conectividad, propiedades del material, carga y restricciones correspondientes.

```
% Ingreso de Coordenadas de los Nodos. (Geométria)
% Coordenada X por Nodo
geom = [0.; ...     % Coordenadas en X de la viga
        1.5;...
        3.0];
% Ingreso de datos de conectividad entre nodos. (Conectividad)
% Conectividad Nodo Inicial/ Nodo Final por Elemento
conect = [1 2;2 3];
%   Propiedades del sistema (Elasticidad/Inercia) por elemento.
prop = [2.535*10^10 0.0072;...
        2.535*10^10 0.0072];
```

```
%  Aplicando Restricciones a los GDLs de los nodos
% Fila = Num. de nodo
% Col = 1 --- Restricción en Y
%     = 2 --- Restriccion en el giro
nf(1,1) = 0;    % Aplicando restricciones en los GDLs
nf(3,1) = 0;
```

```
% Sistema fuerza nodal estatico equivalente para carga distribuida
% CargElem(Elemento,:)=[F.Cort-Inicial  Moment-Inicial  F.Cort-Final  Moment-Final]
% CargElem(Elemento,:)=[-q*L/2  -q*L^2/12  -q*L/2  q*L^2/12]
CargElem(1,:)=[-4500 -1125 -4500 1125];
CargElem(2,:)=[-4500 -1125 -4500 1125];
```

Ingresando los datos correspondientes obtenemos, obtenemos los resultados que se muestran en la Tabla 8, en la cual también se muestran los resultados provenientes de un análisis unidimensional en SAP2000 (Fig. 8.6).

Tabla 8. **Comparativa / Viga, A. Unidimensional**

Fuente: Elaboración Propia

	SAP2000		***Código FEM MatLab***		***Sol. Exacta***
Nodo	$u(y)(m)$	$M(z)$ $(N-m)$	$u(y)(m)$	$M(z)$ $(N-m)$	$u(y)(m)$
2	$3.4 * 10^{-5}$	6750	$3.4671 * 10^{-5}$	6750	$3.4671 * 10^{-5}$

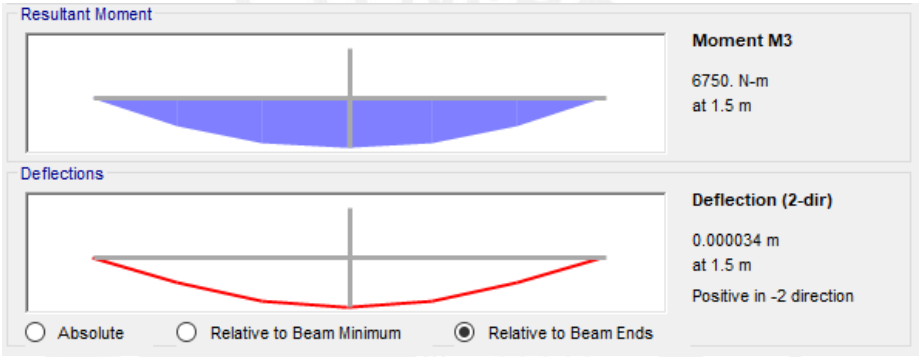


Figura 8.6 **Comprobación SAP2000, Viga unidimensional**

Fuente: SAP2000/ Elaboración Propia

Respecto al análisis bidimensional se hizo uso de los códigos Bi_TenPlana_Gauss_Usuario.m y Bi_TenPlana_Gauss_Programacion.m, se dividió la carga distribuida en 121 cargas puntuales de $148.76\ N$ (al tener 121 nodos en el borde superior), los apoyos solo se consideraron en el punto medio para simular el modelo simplemente apoyado. Todo lo anterior se aprecia en la figura 8.7.

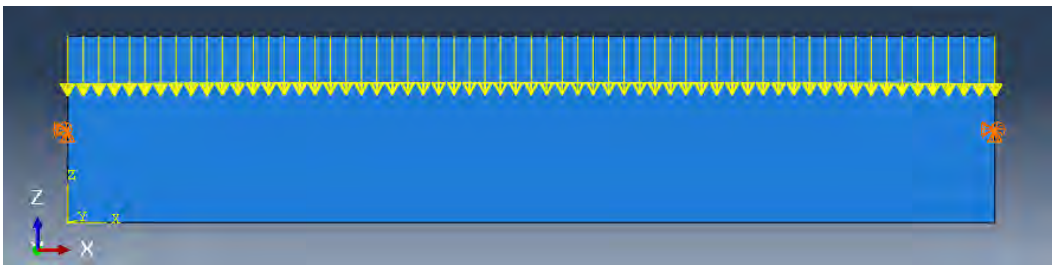


Figura 8.7 **Viga, análisis bidimensional**

Fuente: Abaqus-Modelo/ Elaboración Propia

A continuación, se muestra parte del código sobre el ingreso de data sobre la geometría, conectividad, cargas, entre otros.

```
%   Ingreso de coordenadas de los nodos(Geométria) a partir de archivos txt.
dataSis='VIGA2D_AltoMesh_Coord.txt';
geom = importdata(dataSis);
```

```
%   Ingreso de datos de conectividad entre nodos(Conectividad) a partir de archivos txt.
dataSis='VIGA2D_AltoMesh_Conec.txt';
conect = importdata(dataSis);
```

```
%   Propiedades del sistema
E = 2.535*10^10;        % Módulo de elasticidad del sistema
vu = 0.20;              % Coeficiente de Poisson
thick = 0.40;           % Espesor del sistema
```

```
% Ingreso de restricciones Semiautomático
% i = Nodos restringidos
i=[367, 427];
[a]=size(i,1);
for ios=1:a
    % nf(Nodo,Direc)
    % Direc = Dirección de la restriccion
    % Direc = 1 --- Restricción en X
    % Direc = 2 --- Restricción en Y
    nf(i,1)=0;
    nf(i,2)=0;
end
```

```
% Ingreso de carga Semiautomático
% P = Magnitud de la carga
% i = Nodos donde se aplicara la carga
% Direc = Direccion de aplicación de la carga
       % Direc = 1 --- Carga en X
    % Direc = 2 --- Carga en Y
P=-148.760;
Direc=2;
i=[734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749,...
    750, 751, 752,...,2293, 2295, 2297, 2299, 2301, 2303, 2305, 733, 793];
[a]=size(i,1);
for ios=1:a
    % CargNod(Nodos,Col)=Carga
    % Col = 1 --- Restricción en X
    %     = 2 --- Restricción en Y
    CargNod(i,Direc)=P;
end
```

Ingresada la data y ejecutado el código obtenemos resultados y los comparamos a los que se obtienen del software Abaqus, el código exporta automáticamente graficas de desplazamientos y esfuerzos. A continuación, se muestran las diferentes graficas en forma de comparativa.

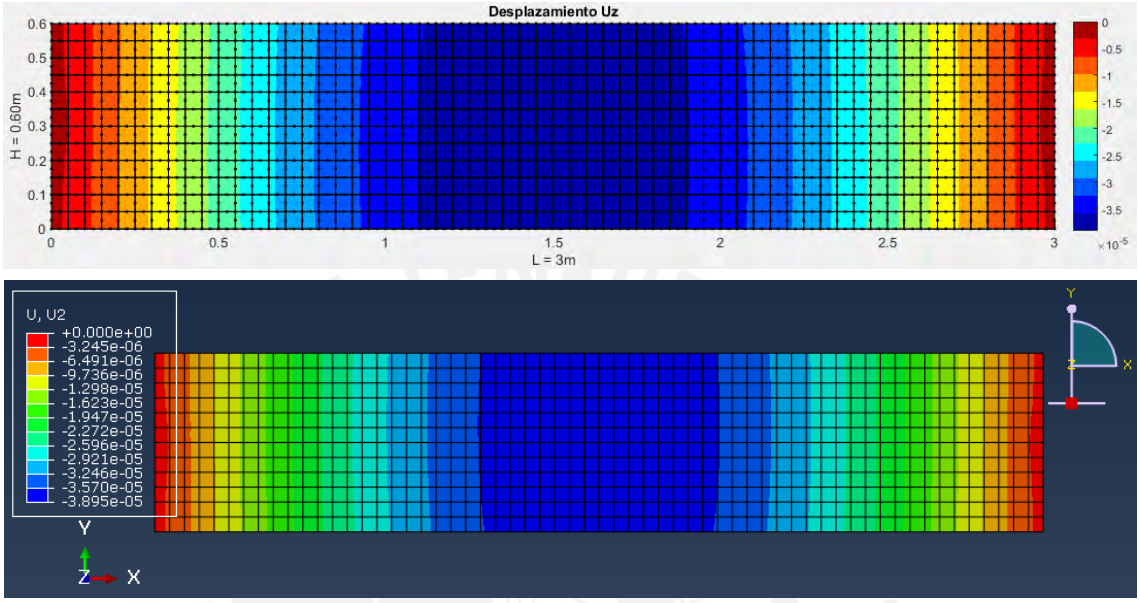


Figura 8.8 **Comparativa, Viga 2D / Abaqus (Desp. en Z)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

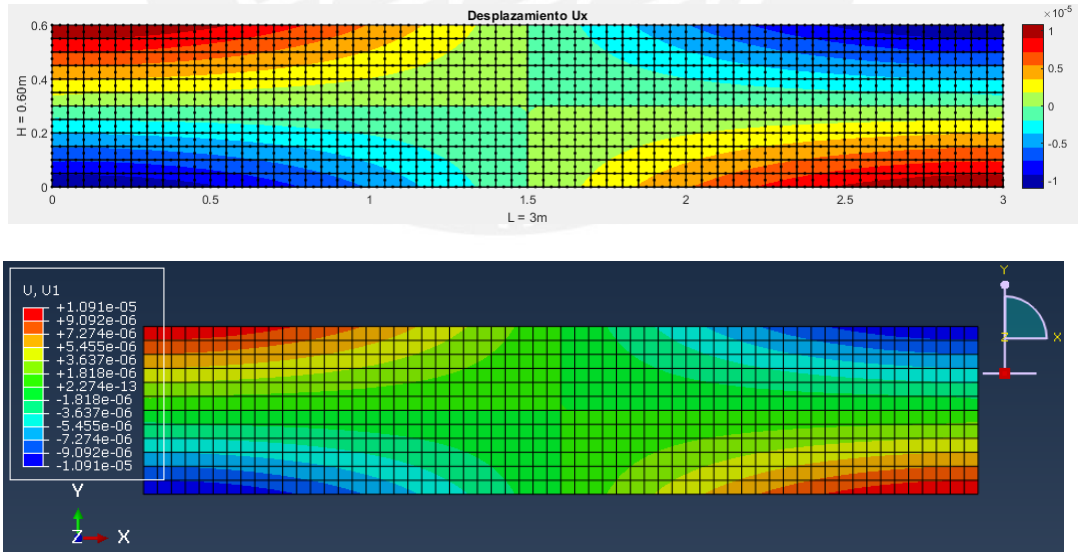


Figura 8.9 **Comparativa, Viga 2D / Abaqus (Desp. en X)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

Se procede a comparar los resultados numéricamente en la Tabla 9, se debe tener en cuenta que el desplazamiento en “y” es irrelevante al estar aplicándose la carga en un plano bidimensional.

Tabla 9. **Comparativa / Viga, A. Bidimensional (Desp.)**

Fuente: Elaboración Propia

	Abaqus		***Código FEM MatLab***	
	$Desp\ z\ (m)$	$Desp\ x\ (m)$	$Desp\ z\ (m)$	$Desp\ x\ (m)$
Max (+)	0	$1.091 * 10^{-5}$	0	$1.0911 * 10^{-5}$
Max (-)	$-3.895 * 10^{-5}$	$-1.091 * 10^{-5}$	$-3.895 * 10^{-5}$	$-1.0911 * 10^{-5}$

De igual manera realizamos una comparación en forma gráfica del esfuerzo en X (principal para obtener el momento) en la figura 8.10.

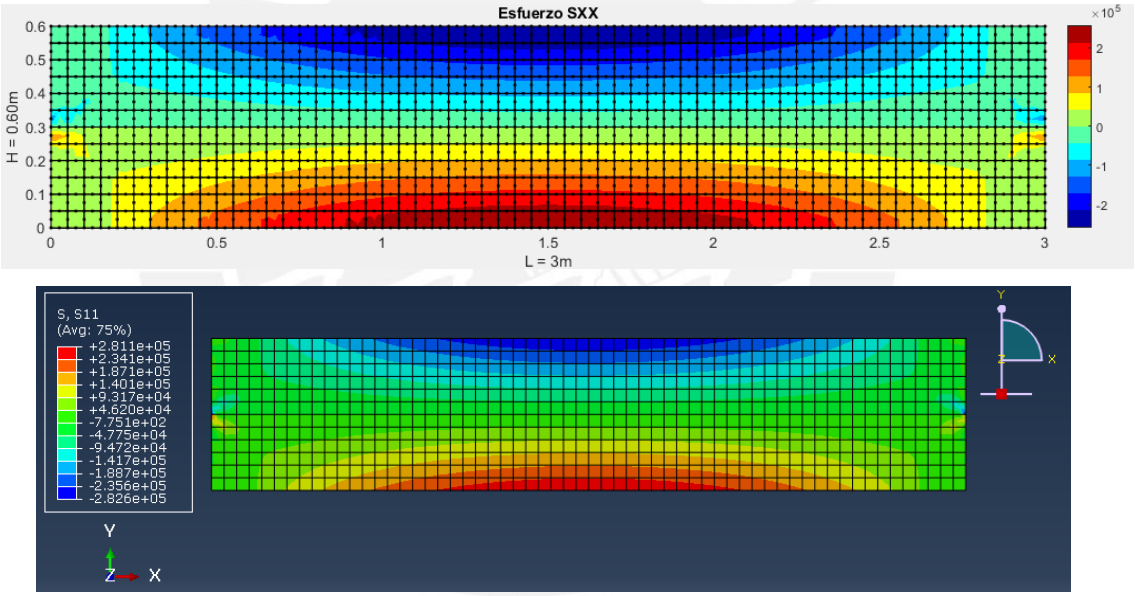


Figura 8.10 **Comparativa, Viga 2D / Abaqus (Esf. en X)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

A partir de estos resultados recordamos conceptos de Resistencia de Materiales para obtener de forma sencilla y aproximada (al ser el esfuerzo superior cercano al inferior) el momento máximo en el medio, aplicando la ecuación mostrada a continuación (mostrada gráficamente en la Figura 8.11):

$$M = \frac{\sigma_m}{c} \int y^2 dA \quad \rightarrow \quad M = \frac{I}{c}\ (\sigma_m) \qquad (8.1)$$

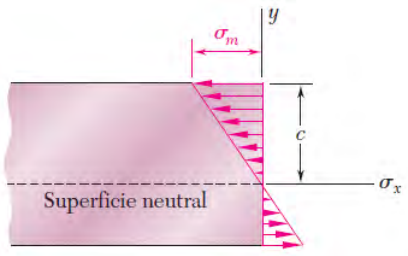


Figura 8.11 **Grafica esfuerzos por flexión en una viga**

Fuente: (Beer, Johnston, DeWolf, & Mazurek, 2007)

Se procede a comparar los resultados numéricamente en la Tabla 10, correspondientes a los desplazamientos obtenidos y momentos calculados a través de los esfuerzos.

Tabla 10. **Comparativa / Viga, A. Bidimensional (Momento)**

Fuente: Elaboración Propia

	Abaqus		***Código FEM MatLab***	
	$Esf\ x\ (N/m^2)$	$Mom.y\ (N-m)$	$Esf\ x\ (N/m^2)$	$Mom.y\ (N-m)$
Max (+)	$2.811 * 10^5$	6746	$2.5764 * 10^5$	6183.36
Max (-)	$-2.826 * 10^5$		$-2.5643 * 10^5$	

Visualizando la Tabla 10, nos damos cuenta de una variación en los resultados, la cual ocurre por varios factores. Primero, el cálculo que realiza Abaqus incorpora una integración reducida para reducir la deformación tangencial en casos de flexión (caso analizado), el código no incorpora esta funcionalidad, acercándose al valor real a través de un un mayor número de elementos en el mesh. Segundo, el cálculo de esfuerzos que realiza el código solo incorpora un punto de integración (nodo central del elemento), esto por ser un código orientado al aprendizaje y contar con un mayor número de puntos de integración necesitaría la integración de una herramienta (bucles) que promedien los 4 o más puntos de integración generados para cada nodo. Tercero, partir con un porcentaje de error en los desplazamientos genera un mayor error en los esfuerzos ya que su obtención depende de los primeros.

Conociendo porque se produce una variación en los resultados, se debe tener en cuenta que el error es mínimo, siendo los resultados prácticamente válidos para uso a nivel ingenieril. Si se desea una mayor cercanía al resultado real se tendrá que aumentar el número de elementos (mayor mesh), o integrar nuevas herramientas como una forma más exacta de análisis.

Para el análisis tridimensional se hizo una simple equivalencia de la carga distribuida a cargas virtuales (teniendo en cuenta el número de nodos - 205), por lo cual la carga distribuida es equivalente a una carga puntual de 28.125 N en las esquinas, 56.25 N en los bordes y 112.5 N en el centro.

Los apoyos se consideraron en la franja central de la sección inicial y final de la viga, esto para simular la condición de viga simplemente apoyada. Todo lo anterior se aprecia en la figura 8.12.

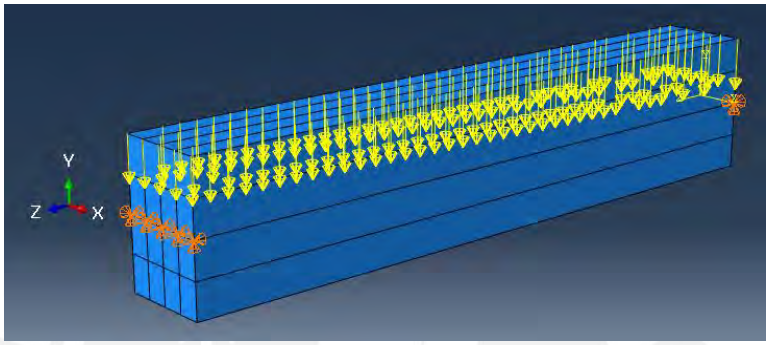


Figura 8.12 **Viga, análisis tridimensional**

Fuente: Abaqus-Modelo/ Elaboración Propia

A continuación, se muestra una parte de código con el ingreso de data.

```
%   Ingreso de coordenadas de los nodos(Geométria) a partir de archivos txt.
dataSis='VIGA3D_AltoMesh_Coord.txt';
geom = importdata(dataSis);
```

```
%   Ingreso de datos de conectividad entre nodos(Conectividad) a partir de archivos txt.
dataSis='VIGA3D_AltoMesh_Conec.txt';
conect = importdata(dataSis);
```

```
% Ingreso de restricciones Semiautomático
% i = Nodos restringidos
i=[5, 6, 7, 8, 11, 12, 15, 16, 26, 27]; % Vector con los nodos con GDLs restringidos
[a]=size(i,1);  % Asignación de los GDLs restringidos por nodo
for ios=1:a
    % nf(Nodo,Direc)
    % Direc = Dirección de la restriccion
    % Direc = 1 --- Restricción en X
    % Direc = 2 --- Restricción en Y
    % Direc = 3 --- Restricción en Z
    nf(i,1)=0;
    nf(i,2)=0;
    nf(i,3)=0;
end
```

```
% Ingreso de carga Semiautomático
% P = Magnitud de la carga
% i = Nodos donde se aplicara la carga
% Direc = Direccion de aplicación de la carga
       % Direc = 1 --- Carga en X
       % Direc = 2 --- Carga en Y
       % Direc = 3 --- Carga en Z
P=-28.125;
i=[39, 40, 47, 48]; % Nodos en las esquinas
Direc=3;
[a]=size(i,1);
for ios=1:a
    CargNod(i,Direc)=P;
end

P=-56.25;           % Nodos en los bordes
i=[37, 38, 41, 42, 43, 44, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827,...
   828, ... , 1022, 1023];
Direc=3;
[a]=size(i,1);
for ios=1:a
    CargNod(i,Direc)=P;
end

P=-112.5;           % Nodos en zonas centrales
i=[779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794,...
    795, ... , 938, 939, 940];
Direc=3;
[a]=size(i,1);
for ios=1:a
    CargNod(i,Direc)=P;
end
```

Para una mayor exactitud en los resultados (más adelante detallado) se realizará un enmallado con un gran número de elementos, dicho mesh se muestra en la Figura 8.13.

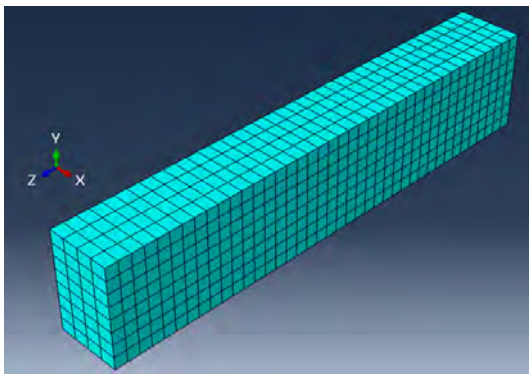


Figura 8.13 **Enmallado de Viga, análisis tridimensional**

Fuente: Abaqus-Modelo/ Elaboración Propia

Una vez ingresada la data y ejecutado el código obtenemos los siguientes resultados sobre desplazamientos, expresados en forma de comparativa.

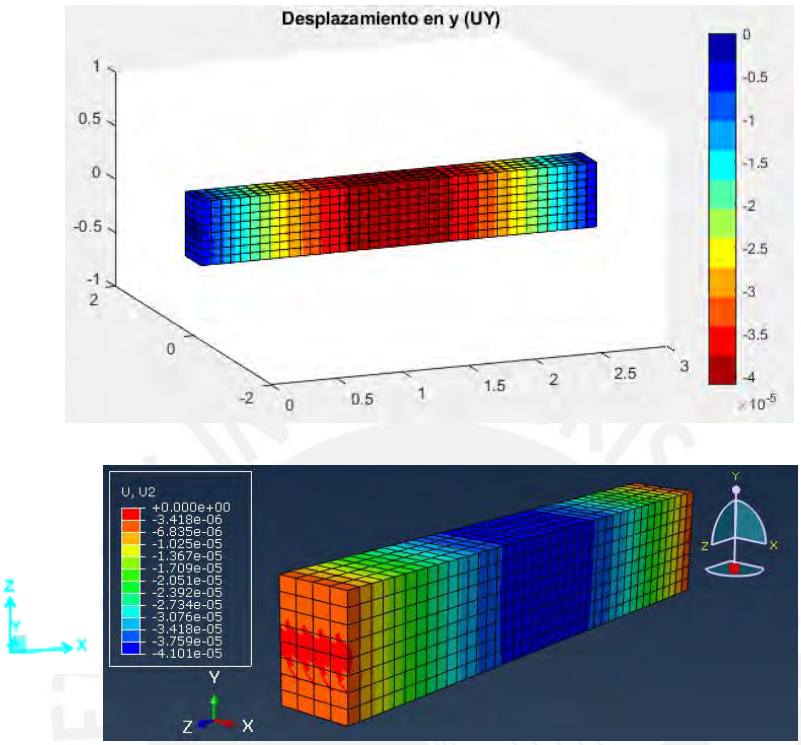


Figura 8.14 **Comparativa, Viga 3D / SAP2000 (Desp. en Y)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

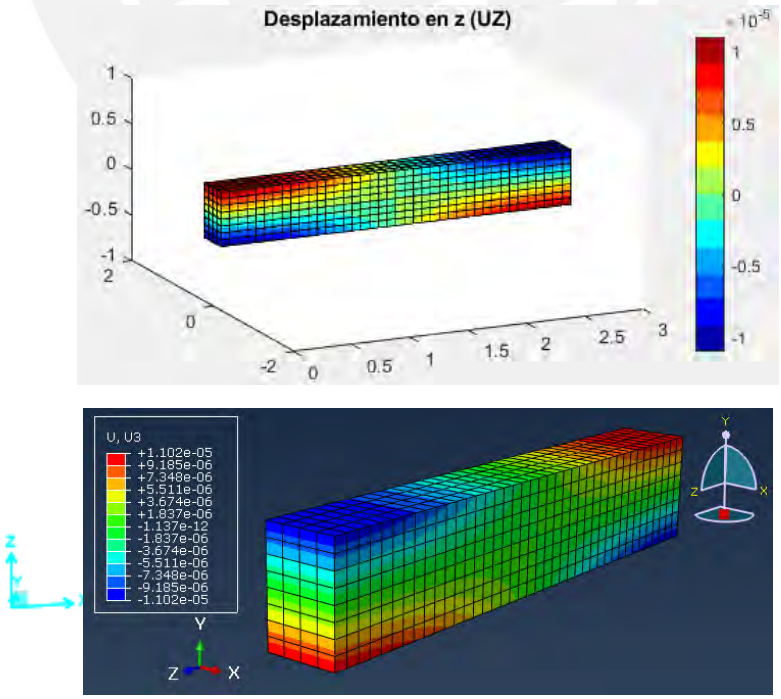


Figura 8.15 **Comparativa, Viga 3D / SAP2000 (Desp. en Z)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

A continuación, se realiza una comparativa en forma numérica de los desplazamientos.

Tabla 11. **Comparativa / Viga, A. Tridimensional (Desp)**

Fuente: Elaboración Propia

	Abaqus		***Código FEM MatLab***	
	$Desp\ y\ (m)$	$Desp\ z\ (m)$	$Desp\ y\ (m)$	$Desp\ z\ (m)$
Max (+)	0	$1.102 * 10^{-5}$	0	$1.1021 * 10^{-5}$
Max (-)	$-4.101 * 10^{-5}$	$-1.102 * 10^{-6}$	$-4.1013 * 10^{-5}$	$-1.1021 * 10^{-5}$

De igual manera realizamos una comparación en forma gráfica del esfuerzo en X (principal para obtener el momento) en la figura 8.16.

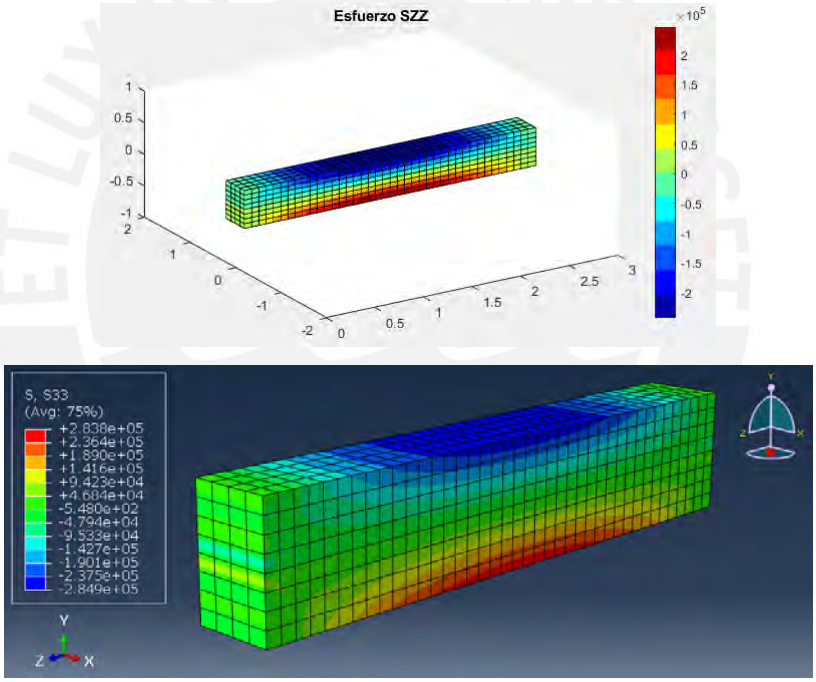


Figura 8.16 **Comparativa, Viga 3D / SAP2000 (Esf. en Z)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

Aplicando la ecuación 8.1 para obtener los momentos, realizamos una comparación de los mismos en forma numérica.

Tabla 12. **Comparativa / Viga, A. Tridimensional (Momento)**

Fuente: Elaboración Propia

	Abaqus		***Código FEM MatLab***	
	$Esf\ z\ (N/m^2)$	$Mom.x\ (N-m)$	$Esf\ z\ (N/m^2)$	$Mom.x\ (N-m)$
Max (+)	$2.838 * 10^5$	6811.2	$2.4650 * 10^5$	5916.0
Max (-)	$-2.849 * 10^5$		$-2.4217 * 10^5$	

Visualizando la Tabla 12, nos damos cuenta que igual al caso bidimensional existe una variación en los resultados correspondiente a los esfuerzos, la cual ocurre por varios factores (iguales al caso bidimensional). Primero, el software Abaqus incluye en su análisis la integración reducida. Segundo, el número de puntos de integración aplicados al análisis modifica la precisión en los resultados siendo mayor al aplicar un mayor número de puntos. Tercero, un porcentaje de error en los desplazamientos genera un error en los esfuerzos, al ser los esfuerzos dependientes (en su obtención) a los desplazamientos. El segundo punto se ve representado en la figura 8.17 en la cual se muestran los sitios/puntos donde se obtienen los resultados (como se aprecia no es exactamente en los nodos extremos), estos puntos se acercan cada vez más al resultado verdadero (nodos extremos) en función a la cantidad de puntos de integración aplicados para el análisis.

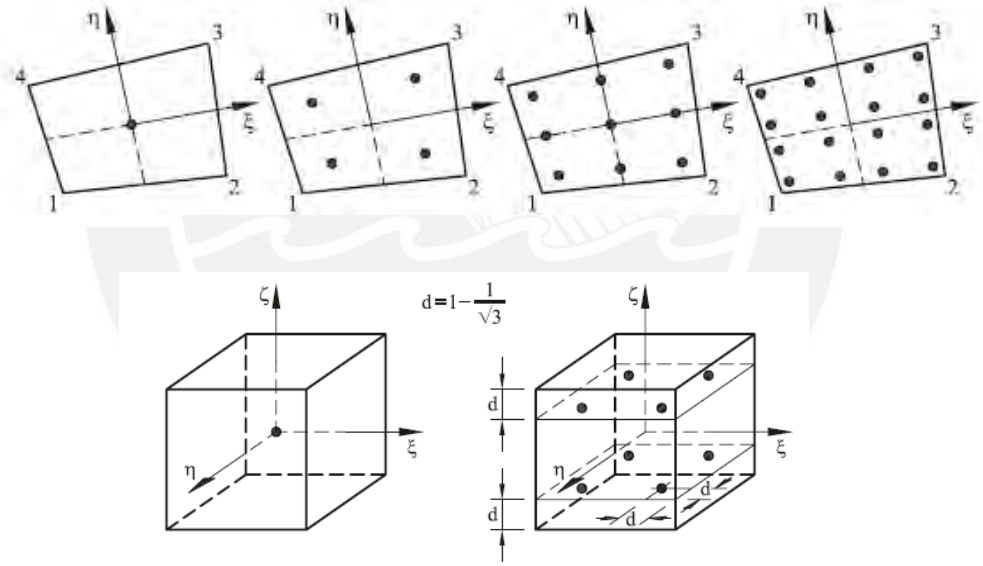


Figura 8.17 **Puntos de Int. Bid - Tridimensional**

Fuente: (Oñate, 2013)

Realizado el análisis de la viga de distintas formas procedemos a realizar un cuadro comparativo general con todos los resultados obtenidos:

Tabla 13. **Comparativa / Viga, resumen de resultados**

Fuente: Elaboración Propia

	Abaqus / SAP2000			***Código FEM MatLab***		
	$Desp\ z\ (m)$	$Esf\ x\ (N/m^2)$	$Mom.y\ (N-m)$	$Desp\ z\ (m)$	$Esf\ x\ (N/m^2)$	$Mom.y\ (N-m)$
1D Max (+)	0	-	6750	0	-	6750
1D Min (-)	$3.4*10^{-5}$	-		$3.4671*10^{-5}$	-	
2D Max (+)	0	$2.811*10^{5}$	6746	0	$2.5764*10^{5}$	6183.4
2D Min (-)	$-3.895*10^{-5}$	$-2.826*10^{5}$		$-3.895*10^{-5}$	$-2.5643*10^{5}$	
3D Max (+)	0	$2.838*10^{5}$	6811.2	0	$2.4650*10^{5}$	5916.0
3D Min (-)	$-4.101*10^{-5}$	$-2.849*10^{5}$		$-4.101*10^{-5}$	$-2.4217*10^{5}$	

En la tabla se muestran una cercanía en resultados, si se desea menor error tan solo sería necesario aumentar el nivel de enmallando o asignar nuevas herramientas o formas de análisis. Con respecto a esto último, si aplicamos dos puntos de integración en lugar de uno para el cálculo de esfuerzos se nota una notable mejora como se muestra en la siguiente tabla (cabe resaltar que ambas formas de análisis se ven incluidas en el código anexado):

Tabla 14. **Comparativa / Viga, resumen de resultados**

Fuente: Elaboración Propia

	Abaqus / SAP2000		***Código FEM MatLab***	
	$Esf\ x\ (N/m^2)$	$Mom.y\ (N-m)$	$Esf\ x\ (N/m^2)$	$Mom.y\ (N-m)$
	1 Punto de Integración			
3D Max (+)	$2.838*10^{5}$	6811.2	$2.4650*10^{5}$	5916.0
3D Min (-)	$-2.849*10^{5}$		$-2.4217*10^{5}$	
	2 Puntos de Integración			
3D Max (+)	$2.838*10^{5}$	6811.2	$2.6797*10^{5}$	6431.28
3D Min (-)	$-2.849*10^{5}$		$-2.6894*10^{5}$	

Finalmente, el lector debe tener en claro que el código, es tan solo un medio de ejemplificar la forma de aplicar el FEM y no un programa que este orientado netamente al análisis de sistemas estructurales, ya que este último necesita tener en cuenta muchas otras consideraciones en su programación para la obtención de resultados óptimos.

8.4. ESCALERA HELICOIDAL

Para este ejemplo aplicativo se realizará el análisis de una escalera helicoidal en forma de papelillo, se aplicará el código y corroborará el mismo mediante el uso del Abaqus (software orientado al FEM). El código del ejemplo se encuentra en el Anexo 3 (Comparativa – Código Optimizado) y Anexo 5 (Data), esto ya que se usó el mismo ejemplo para la comparativa de código optimizado y sin optimizar, en el cual solo se coloca el código correspondiente al ingreso de datos del usuario usando el código aplicativo general de un elemento tridimensional. (Anexo 2: Código Base para el análisis del FEM).

Objetivo: Analizar una escalera Helicoidal en forma de papelillo como se muestra en la figura 8.18. Dicha escalera está sometida a cargas puntuales colocadas en la mitad de toda su longitud. Obtener los desplazamientos generados debido a la solicitación pedida, mediante el uso del Código Tri_Elemento_Usuario.m y Tri_Elemento_Programacion.m.

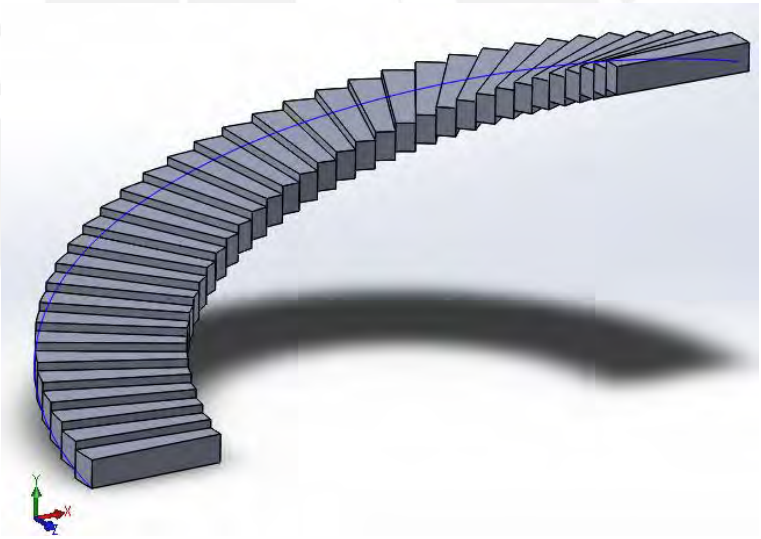


Figura 8.18 **Esc. helicoidal, modelo geométrico**

Fuente: SolidWorks / Elaboración Propia

En primer lugar, se realizó un modelo geométrico en SolidWorks tal como muestra en la figura 8.18. Una vez obtenido el modelo importamos la data a Abaqus para utilizar las herramientas de mesh y obtener datos de conectividad y coordenadas geométricas de los nodos. La importación de dicho modelo y el mesh se puede apreciar en la figura 8.19.

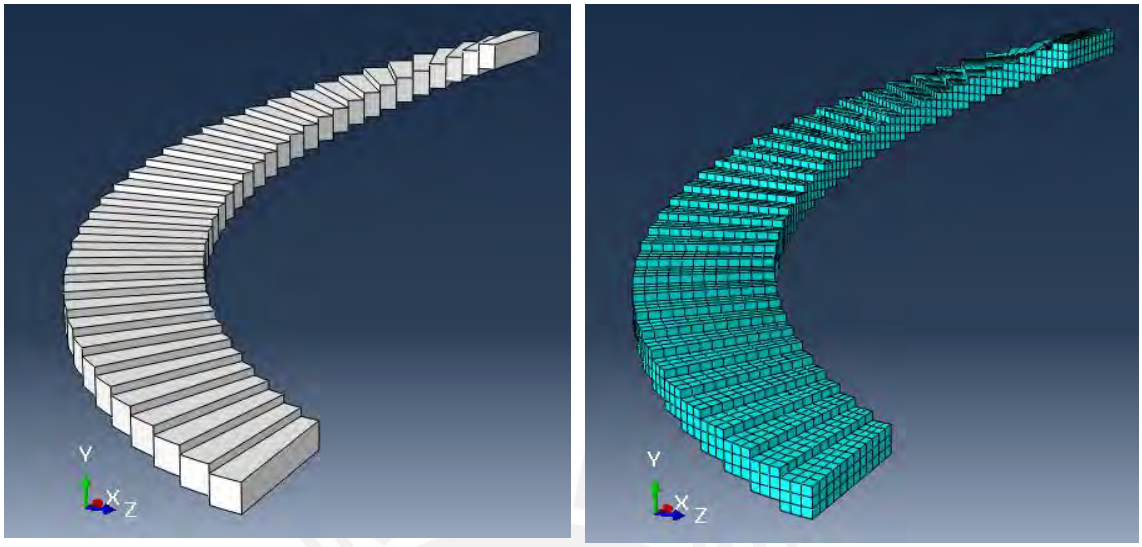


Figura 8.19 **Importación de data y mesh**

Fuente: Abaqus/ Modelo-Elaboración Propia

El ingreso de la data es a través del mismo código del ejemplo aplicativo de la viga tridimensional, por lo que el ingreso de información sigue el mismo procedimiento.

Una vez ingresados los datos del problema con la ayuda de la herramienta de mesh de Abaqus, procedemos a ejecutar el código para obtener resultados del análisis mediante el FEM. El código exporta automáticamente graficas de esfuerzos y desplazamientos correspondientes, obteniendo también los datos numéricos de los mismos en cada nodo. A continuación, se muestran las gráficas correspondientes los desplazamientos, obtenidos del código elaborado y los del modelo en Abaqus.

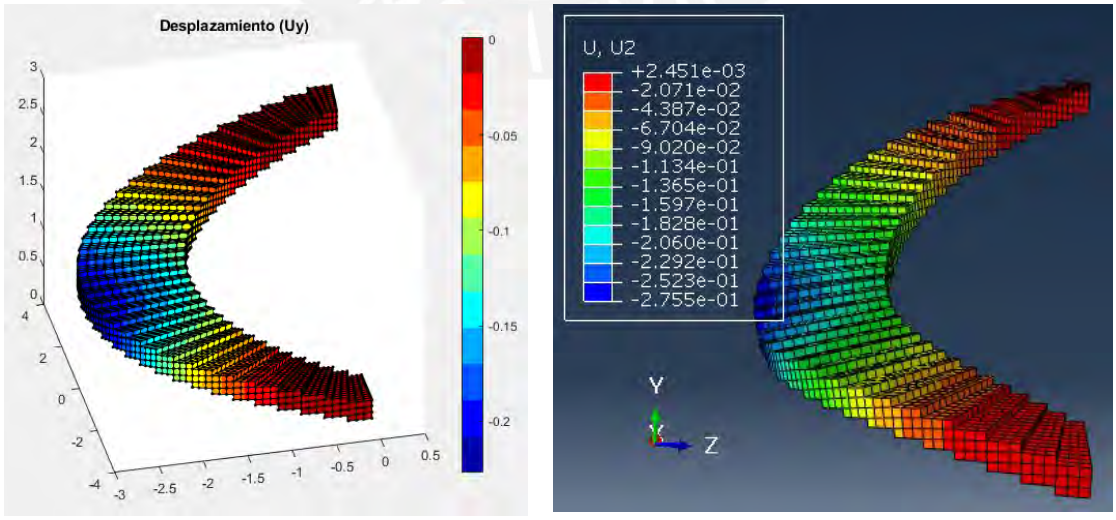


Figura 8.20 **Comparación Código / Abaqus (Desp. en Y)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

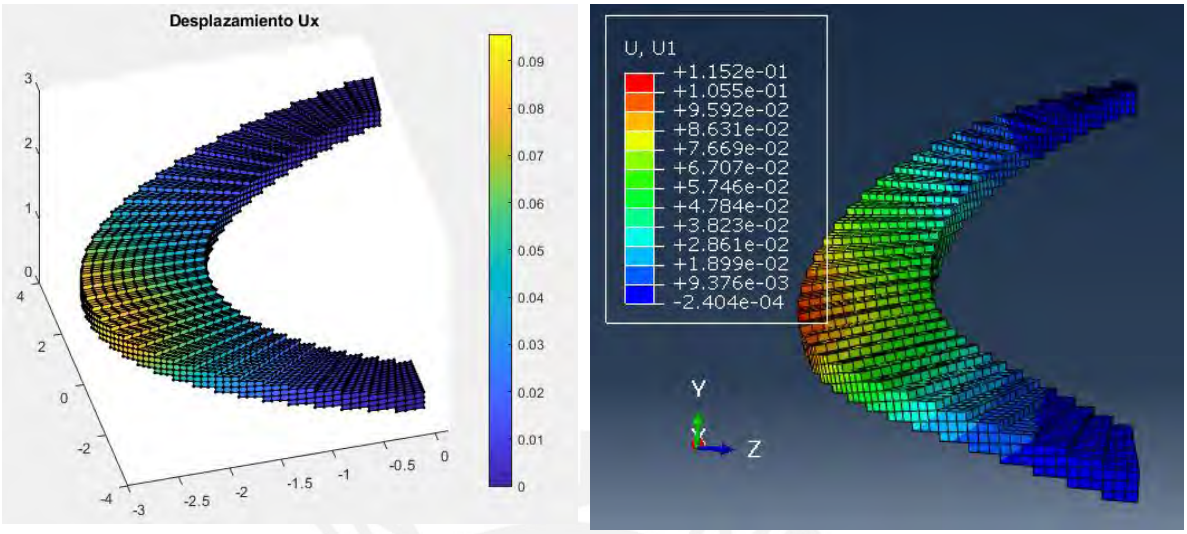


Figura 8.21 **Comparación Código / Abaqus (Desp. en X)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

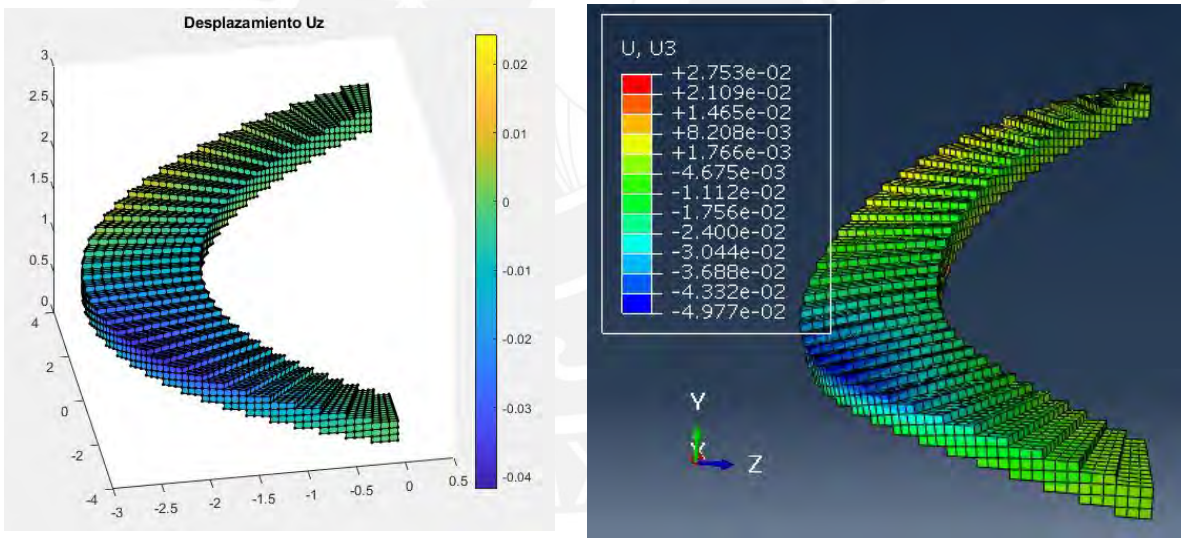


Figura 8.22 **Comparación Código / Abaqus (Desp. en Z)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

Las figuras 8.20, 8.21 y 8.22 muestran los desplazamientos que sufre la escalera ante las solicitaciones. A partir de dicha comparativa se aprecia directamente el correcto funcionamiento que brinda el código.

Se debe tener en cuenta que la variación en el límite de los resultados, se debe al hecho que se optó en obtener los esfuerzos mediante solo dos puntos de integración. Reducir el factor de error necesitaría un mayor enmallado, esto no podrá ser realizado

porque la herramienta MatLab tiene un límite de memoria incorporada para el procesamiento de datos y exceder dicho límite en el uso de operaciones con variables doublé generaría el error mostrado en la figura 8.23.

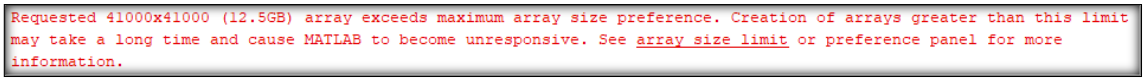


Figura 8.23 **Comparación Código / Abaqus (Desp. en Z)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

Este error se puede solucionar usando variables sparse en lugar de double, el uso de dichas variables se realiza en el código no-lineal (explicando más adelante). El uso de dichas variables hace necesaria una optimización radical en la asignación de variables, lo cual volvería al código más complejo, y no se opta por incluir en el campo lineal, ya que dicho código está orientado al aprendizaje sencillo.

Se debe tener en claro que el código, es tan solo un medio de ejemplificar la forma de aplicar el FEM y no un programa que este orientado netamente al análisis de sistemas estructurales.

CAPÍTULO 9

Introducción a la no linealidad

Resumen

El análisis de estructuras por elementos finitos en el rango no lineal es un campo extenso y complejo, por lo cual el presente capítulo se enfoca en introducir al lector en su aplicación. Los temas que se evaluaron son conceptos fundamentales del campo no lineal, algoritmo general del FEM en el campo no lineal, pautas a tener en cuenta en su aplicación haciendo uso de ordenadores y resultados de códigos que siguen el orden de procesos y las pautas planteadas.

CAPÍTULO IX: INTRODUCCIÓN A LA NO LINEALIDAD

9.1. CONCEPTOS BÁSICOS

La aplicación del FEM en un campo no lineal es un tema con una gran extensión, abarcando un gran número de ramas de especialización. Los campos de estudio más conocido son referentes a: El método de solución de la ecuación no lineal, modelo de comportamiento con el cual se desee trabajar, premisas de convergencia/precisión en los resultados, formulaciones para la aplicación/generación del mesh, aplicaciones de condiciones de contacto entre elementos, y muchos otros.

Dada la complejidad del método y las limitaciones que tiene la presente investigación (al no estar enfocada netamente al tema), se brindaran los fundamentos del método en forma introductoria desde un enfoque plenamente estructural.

En primer lugar, el lector debe comprender el significado de la no-linealidad en un sistema, si retomamos conceptos de resistencia de materiales para calcular el esfuerzo de una barra sometida a fuerzas en su eje hacemos uso de la formula $\sigma = \frac{F}{A}$. Dicha fórmula se aplica bajo la premisa de que el área es constante ante toda la carga aplicada, sin embargo, ante la aplicación de una carga de gran magnitud el área de la barra empezará a disminuir siendo ya no constante (la formula ya no se cumplirá). Dicho concepto nos da a entender una parte de la no linealidad de un sistema, y está reflejado en la figura 9.1 (a). En la misma figura, pero en la parte “c” se aplica otro ejemplo de no linealidad referente a la obtención de la deformación del elemento ($\varepsilon = \frac{\delta L}{L}$), dicha deformación está basada en el alargamiento que sufre la barra ante la aplicación de una fuerza de tracción, sin embargo, como se ejemplifico anteriormente la disminución del área generaría un alargamiento extra en la barra y por ende una mayor deformación en su eje longitudinal. Por tal motivo, dichas ecuaciones solo se cumplirán ante deformaciones pequeñas en las cuales el área se mantenga constante bajo un rango elástico.

Finalmente, un punto importante a tratar y por el cual se establecen modelos del comportamiento del material, es la relación esfuerzo/deformación de un elemento (figura 9.1-b). En la formulación de la matriz constitutiva (subcapítulo 2.2.4), se explicó que dicha matriz sirve para ingresar las propiedades del material y establecer una relación directa entre esfuerzos y deformaciones del elemento. Dicha formulación solo

se puede usar en el campo lineal, ya que como se aprecia en la figura, pasado el punto de fluencia se establece un comportamiento no lineal, por lo que su relación directa ($\{\sigma\} = [D]\{\varepsilon\}$) ya no se cumpliría al no ser lineal.

Todo lo anterior debe dejar en claro un hecho muy importante: Las ecuaciones básicas con las cuales se formulan muchos métodos simplificados (rigidez, carga unitaria, etc) ya no se aplicarían directamente en un campo no lineal.

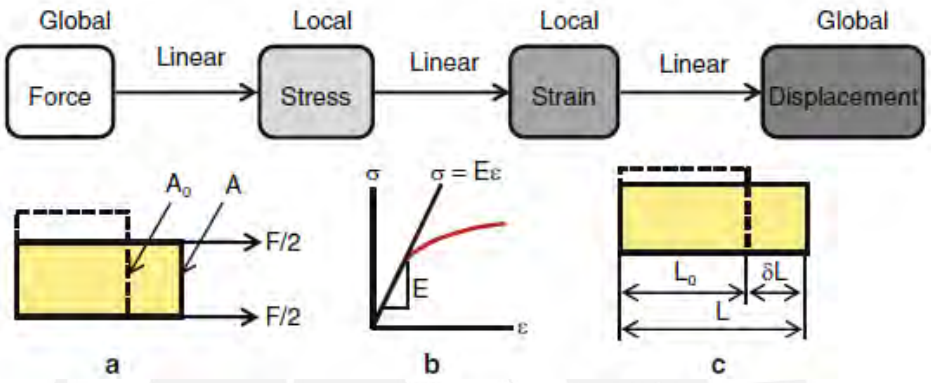


Figura 9.1 **Linealidad en sistemas estructurales**

Fuente: (Kim, 2014)

El desarrollo del FEM en un campo no lineal plantea un reajusta en las ecuaciones formuladas, dependientes del tipo de no-linealidad que se desee trabajar. Teniendo en claro que existen diversas no linealidades en un sistema: no linealidad en el comportamiento del material, en sus deformaciones, en sus condiciones de borde, etc.

A primera instancia un profesional que empieza a incursionar en este ámbito pensaría que lo mejor sería trabajar con todas las no-linealidades para obtener un comportamiento lo más cercano al real. Sin embargo, su análisis causaría una gran carga en el ordenador y en la mayoría de casos una no-convergencia en los resultados, por tal motivo es común centrarse en un tipo específico de no-linealidad como es la del comportamiento del material. Dicha no-linealidad se centra en obtener la curva de comportamiento mediante el cálculo de puntos, haciendo uso de ecuaciones algebraicas no-lineales en conjunto a herramientas iterativas para calcular resultados.

9.1.1. Sol. de ecuaciones algebraicas no-lineales

Para obtener la gráfica del comportamiento del material es común el uso de métodos iterativos, los cuales trazan la gráfica fuerza/desplazamiento en base a la obtención de diversos puntos de la misma. Estos métodos pueden estar planteados de muchas maneras, existiendo diversas metodologías de desarrollo en la actualidad, un método clásico y recomendable para entender inicialmente el funcionamiento general es el de Newton- Raphson.

Se debe tener en claro que dichos métodos hacen uso de iteraciones para la obtención de un punto en la curva fuerza/desplazamiento (esto cambia si es que el método está planteado en función a la obtención de desplazamientos), por tal motivo se hace uso de incrementos los cuales nos marcan una razón de obtención de dichos puntos. Si en el caso se aplicaran diversas cargas o restricciones al sistema, se pueden dividir estas solicitaciones en pasos, los cuales pueden tener diferentes razones de incrementos. A primera instancia todo este proceso puede resultar confuso en el lector al estar abarcándolo en forma muy general, por lo cual se le recomienda el subcapítulo 2.2 del libro de Nam Ho Kim (Kim, 2014).

9.2. ALGORITMO GENERAL DE APLICACIÓN

A continuación, se procede a mostrar un algoritmo general para la programación de un código orientado al FEM en un campo no lineal, basado en el código NLFEA (mas adelante tratado). El lector debe tener en cuenta que, si bien el algoritmo cumple los objetivos para métodos generales, se puede reformular si se incorporan nuevas metodologías (condiciones de borde, automatización del mesh, etc.), o si es que el variar el orden de algunos pasos mejore el rendimiento del código (optimización del código).

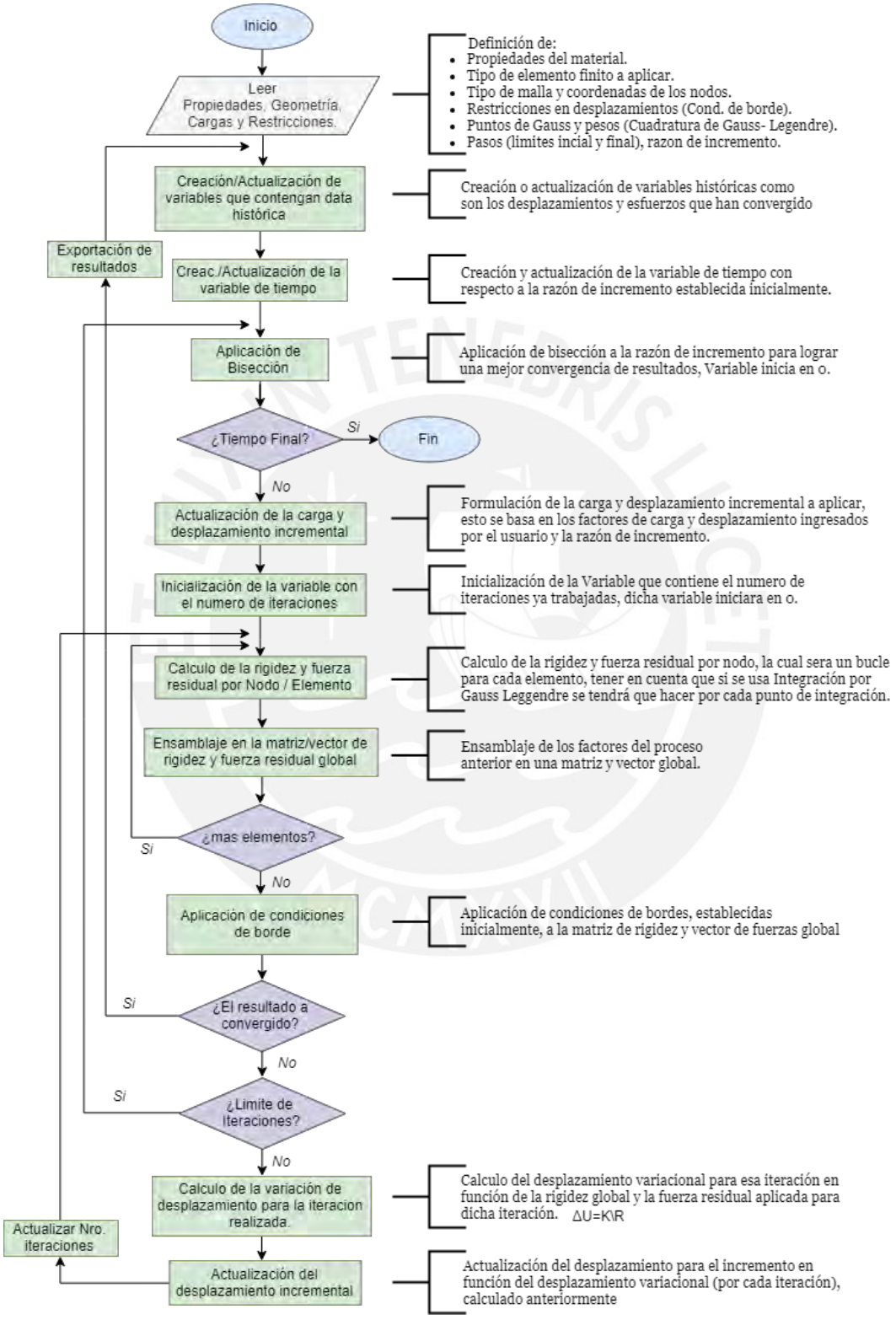


Figura 9.2 **Algoritmo General FEM. No-lineal**

Fuente: Draw.io / Elaboración Propia

9.3. CÓDIGO FEM EN EL CAMPO NO-LINEAL

Teniendo en cuenta que la aplicación del FEM en el campo no lineal es un añadido extra en la presente investigación, la elaboración de un código propio (como es en el campo lineal) es algo inabordable por las limitaciones y la complejidad del método. Por tal motivo se plantea partir del código NLFEA elaborado por Nam Ho Kim en su libro Introduction to Nonlinear Finite Element Analysis (Kim, 2014), el cual está escrito en MatLab. A partir de dicho código se elaboró el algoritmo de programación antes planteado y se procederá a realizar un proceso de optimización del mismo para verificar la importancia de la misma.

9.3.1. Optimización de un código FEM – No lineal

Si bien la optimización del FEM en un campo lineal es altamente recomendable por la reducción en el tiempo de procesamiento de datos, en un campo no-lineal dicha optimización es prácticamente imprescindible para el análisis de sistemas complejos. Su importancia se debe al hecho de que en un campo no-lineal se realizan bucles para analizar cada elemento, dentro de otros para cada iteración, dentro de otros para cada paso correspondiente y todo esto solamente es continuo si hay convergencia de resultados, por lo que básicamente se realiza un análisis complejo y extenso. Todo lo anterior es contemplado por los programas enfocados a esto que se usan profesionalmente en la actualidad, los cuales han sido refinados/optimizados mediante actualizaciones de versiones.

Siendo el tema de optimización en un campo no-lineal más complejo que el lineal, se dejan un conjunto de recomendaciones específicas adquiridas luego de un gran número de intentos y reformulaciones del código NLFEA propuesto por Nam Ho Kim. Dichas recomendaciones se deben usar en conjuntos a las estipuladas en el capítulo 7 y son válidas para cualquier otro código orientado al FEM.

- Cuando se desee almacenar una gran cantidad de información en algunas variables como es la matriz de rigidez general, se generará el problema del límite en la capacidad de asignación de memoria (para variables tipo double). Por lo cual es imprescindible el uso de variables “Sparse” o similares, dicha variable solo almacenara data distinta a cero en la matriz, pero representara la matriz en su totalidad. Se recomienda una vez declarada la variable sparse asignar los límites de la misma antes del ingreso de cualquier dato, esto para mejorar el tiempo de procesamiento de datos.

- Para la formulación de la matriz de rigidez global es común el ingreso iterativo de data proveniente de la matriz de rigidez individual de cada elemento, todo esto haciendo uso de variables tipo Sparse para sistemas complejos. Dicho proceso causa una gran carga computacional al estar transfiriendo información de forma iterativa en una variable compleja, por lo que se recomienda reemplazar dicho proceso por el ingreso independiente de todas las matrices de rigidez individual en una variable tridimensional. Una vez obtenida la rigidez de todos los elementos en dicha variable, se procedería a ingresar la información a la variable sparse en forma única y conjunta. El motivo de este artificio es debido a que el tiempo que tarda el ingreso de información en una variable sparse es mayor a otras variables típicas (por ejemplo, double).
- En la programación el traslado incensario de información es sinónimo de un mayor tiempo en la ejecución de un código, por ejemplo, en lugar de tener muchas sub-funciones o sub-programas esto se puede reemplazar por un código general y compacto. En el caso del FEM no-lineal, se recomienda hacer uso “si es necesario” de diversas funciones e incluso subprogramas ya sea para la exportación de gráficas, visualización, obtención de variables específicas, etc. El motivo principal de dicha recomendación es debido a que la propia metodología hace uso de un gran número de subprocesos, generando diversas variables y un gran número de líneas de código, las cuales en gran mayoría solo sirven para generar resultados específicos. Si no agrupamos dichas líneas en funciones o subprogramas externos, se generará incomprensión del propio código por parte de profesionales que no estén familiarizados con el mismo.

9.3.2. Resultados de la optimización de datos

Para tener en claro la importancia en la optimización de un código orientado al análisis mediante el FEM – no lineal, se aplicarán las pautas brindadas anteriormente a un código ya validado y publicado como es el NLFEA propuesto por Nam Ho Kim, esto para: Contemplar que dichas pautas son válidas en la optimización de cualquier código orientado al método en general y tener una idea de hasta cuanto puede variar el tiempo de procesamiento para sistemas complejos cuando se incorpora la optimización.

El ejemplo que se analizará será respecto a una conexión viga/columna de acero empernada, la cual consta de 12228 elementos. En la figura 9.3 se muestra el modelo

geométrico como el mesh asignado por Abaqus. Teniendo en claro que el único objetivo del ejemplo es ver el tiempo de procesamiento de datos del mismo, ya que la corroboración de resultados se realizara mediante ejemplos aplicativos desarrollados más adelante (Subcapítulo 9.4).

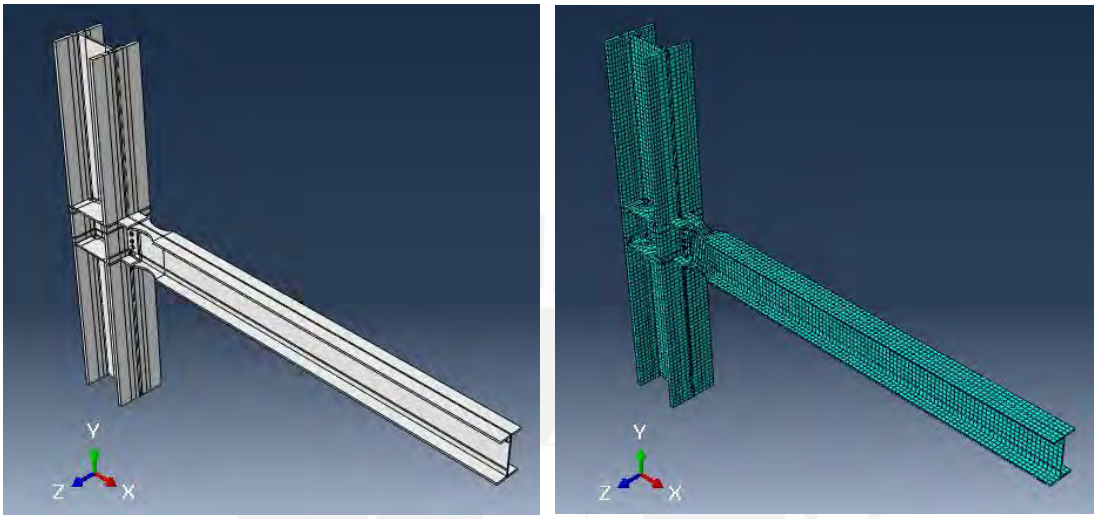


Figura 9.3 **Importación, generación del mesh conexión Viga/Columna**

Fuente: Abaqus / Elaboración Propia

El sistema se analizó bajo dos códigos: El NLFEA base publicado por Nam Ho Kim y el NLFEA optimizado (teniendo en cuenta las pautas establecidas). Las gráficas de desplazamientos se muestran a continuación.

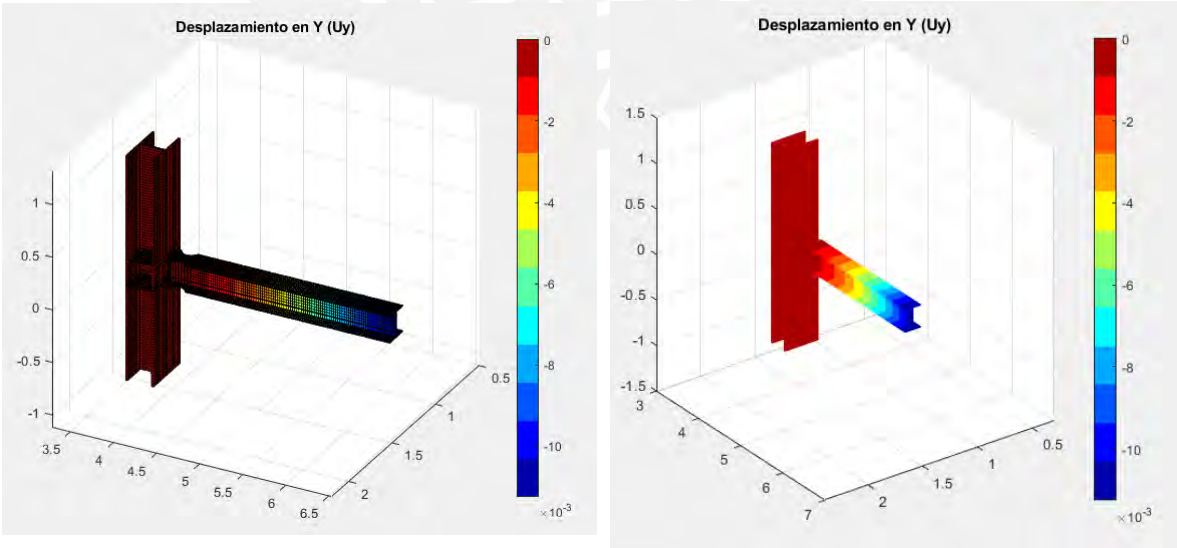


Figura 9.4 **Código (Desp. en Y)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

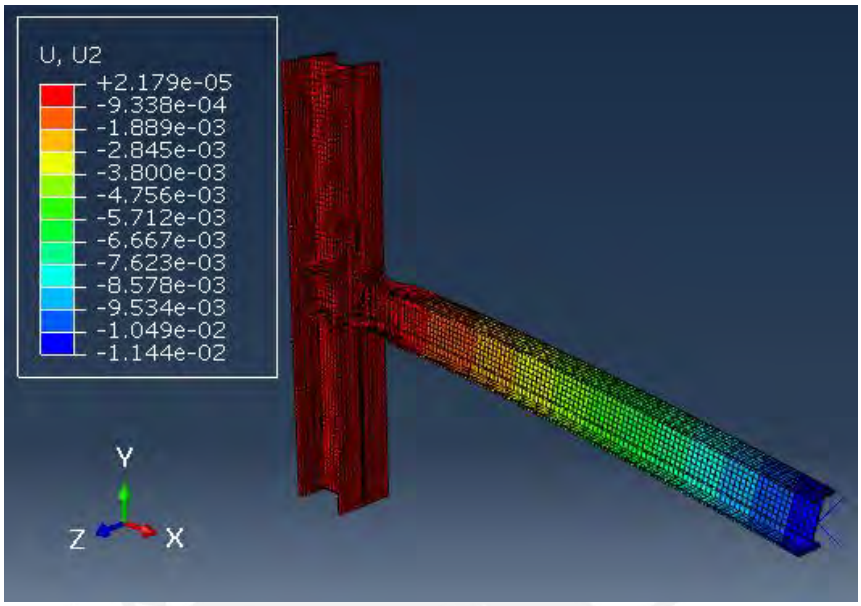


Figura 9.5 **Comprobación en Abaqus (Desp. en Y)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

Respecto al sistema analizado se muestra a continuación la variación que tiene el código base con el optimizado a través de la herramienta Profile.

Profile Summary

Generated 16-Oct-2019 18:06:08 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
NLFEA_base	1	3119.490 s	1.548 s	
NLFEAVER_ok	1	3117.819 s	12.850 s	
PLAST3D	15	3089.844 s	3014.399 s	

Figura 9.6 **Resumen herramienta Profile (NLFEA base)**

Fuente: MatLab/ Profile -Elaboración Propia

Profile Summary

Generated 16-Oct-2019 17:06:47 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
NLFEA_optimizado	1	2015.411 s	5.858 s	
NLFEAVERMODOK	1	2005.937 s	14.171 s	
PROUTVERMODOK	4	1375.832 s	1375.832 s	

Figura 9.7 **Resumen herramienta Profile (NLFEA optimizado)**

Fuente: MatLab/ Profile -Elaboración Propia

Tabla 15. **Comparativo código sin optimizar / optimizado (NLFEA)**

Fuente: Elaboración Propia

	Código sin Optimizar	Código Optimizado
Tiempo de obtención de resultados.	3119.490 seg 51.99 min	2015.411 seg 33.59 min
Optimización	35.39%	

Las figuras 9.6 y 9.7 (conjunto a la tabla 15), reflejan la variación en el tiempo de procesamiento de datos para un sistema medianamente complejo. Teniendo en cuenta los resultados obtenidos al realizar la comparativa se puede concluir que un código un tanto más “profesional” y trabajado (como es el publicado por el autor) también puede llegar a optimizarse, esto último es comúnmente pulido en programas profesionales a través de actualizaciones de versiones. Se debe tener en cuenta que todos los ejemplos aplicativos trabajados posteriormente se realizaran en el código optimizado.

9.4. APLICACIÓN DEL CÓDIGO NLFEA OPTIMIZADO

Para ejemplificar, lo que se puede lograr con el código optimizado y a forma de verificar el mismo, se procede a analizar dos sistemas y verificar los resultados con el Abaqus, la selección de dichos sistemas se realizó teniendo en cuenta las propias limitaciones del código NLFEA.

9.4.1. Elemento 3D sometido a tracción

Para ejemplificar, lo que puede lograr a analizar el software y corroborar resultados, se realizara el ejemplo trabajado por Nam Ho Kim (Kim, 2014), el cual es un sistema de cubos 3D. El sistema estará compuesto de dos elementos cúbicos conectados verticalmente, expuesto a 4 cargas puntuales de 11 000 Newton en cada una de las esquinas superiores del sistema. La figura 9.8 muestra el modelo geométrico tratado (modelado en SolidWorks) y su mesh (realizado con la herramienta que presenta Abaqus).

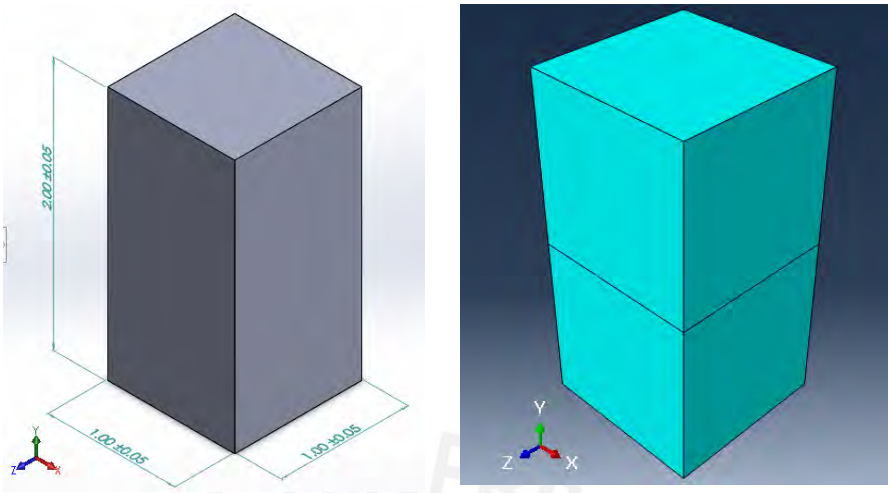


Figura 9.8 **Sistema 3D, modelo geométrico y mesh**

Fuente: SolidWorks - Abaqus / Elaboración Propia

Conociendo la geometría del sistema procedemos a mostrar las propiedades del material (Tabla 16) y el modelo bilineal que se trabajara simulando endurecimiento por deformación (figura 9.9).

Tabla 16. **Propiedades del sistema 3D – Elemento Cubico**

Fuente: Elaboración Propia

Propiedad	***Cantidad***	***Propiedad***	***Cantidad***
Mod. Elasticidad (E)	$2.069 * 10^{11}\ Pa$	Coef. Lame (λ)	$110.747\ x\ 10^{9}$
Mod. Plástico (H)	$1.0 * 10^{8}\ Pa$	Mod. de corte (μ)	$80.1938\ x\ 10^{9}$
Mod. de Poisson (υ)	0.29	Esf. de fluencia (σy)	$4.0\ x\ 10^{8}\ Pa$

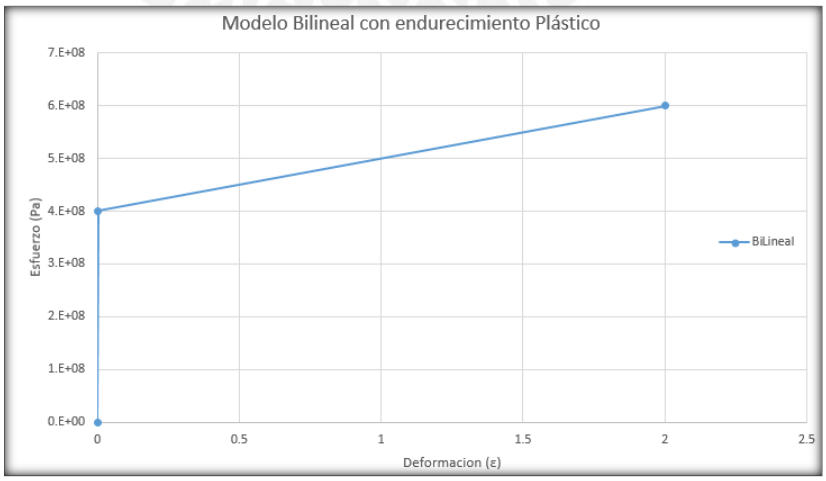


Figura 9.9 **Mod. bilineal con endurecimiento plástico (Elemento 3D)**

Fuente: Elaboración Propia

Una vez conocidas las propiedades del material, cargas aplicadas y la geometría del mismo, procedemos a realizar el análisis ingresando la información al NLFEA optimizado. Se obtienen los siguientes resultados:

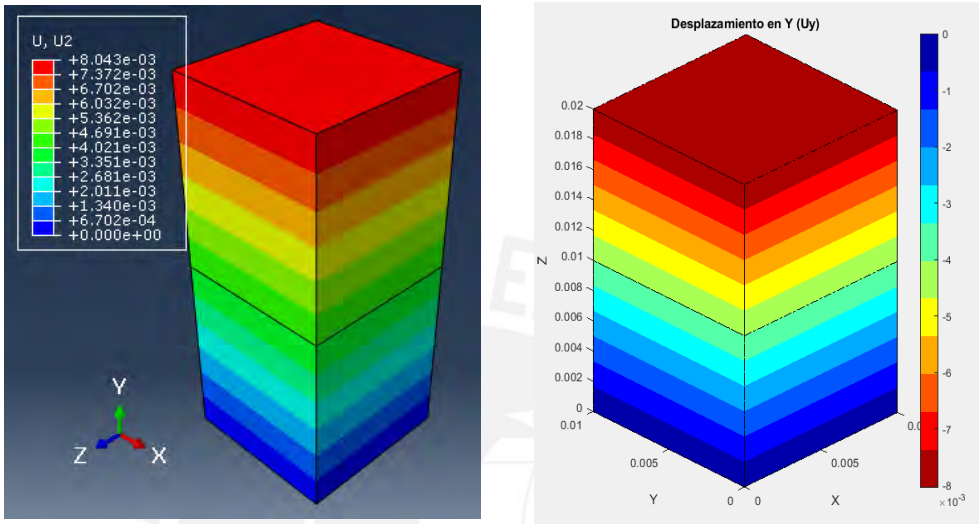


Figura 9.10 **Comparación Código / Abaqus (Desp. en Y)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

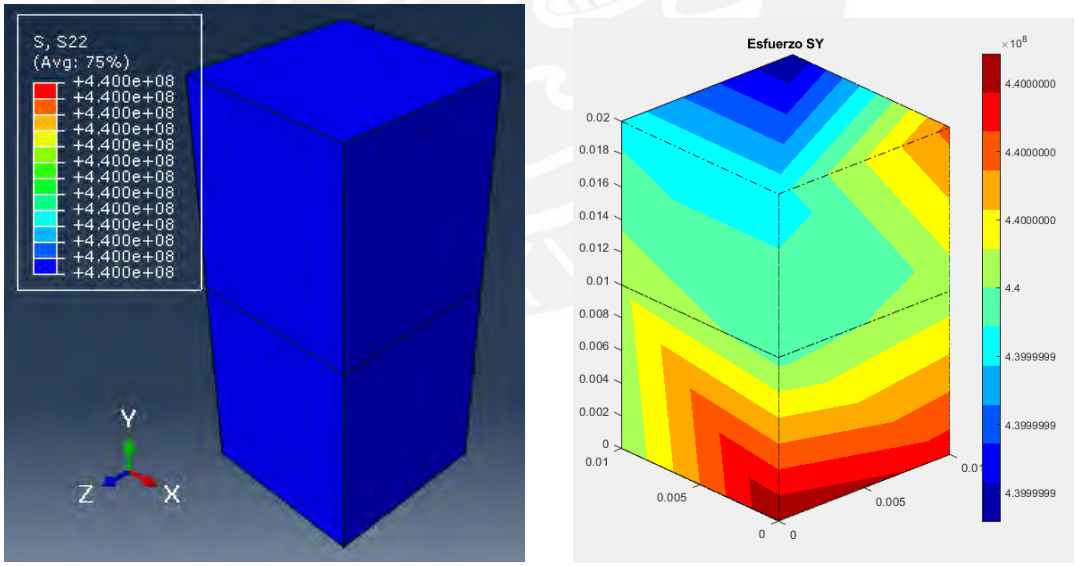


Figura 9.11 **Comparación Código / Abaqus (Esf. en Y – S22)**

Fuente: MatLab – Abaqus / Código – Modelo. Elaboración Propia

A primera instancia la gráfica anterior puede hacer parecer una gran diferencia en los resultados, pero si nos centramos en la data procedente del colorbar, la diferencia es prácticamente nula y es debida a diferencias decimales. Si realizamos una gráfica

de Fuerza / Desplazamiento podemos ver prácticamente igualdad en los resultados, quedando la única y pequeña diferencia en el punto de fluencia, esto debido al espaciamiento de los puntos de iteración.

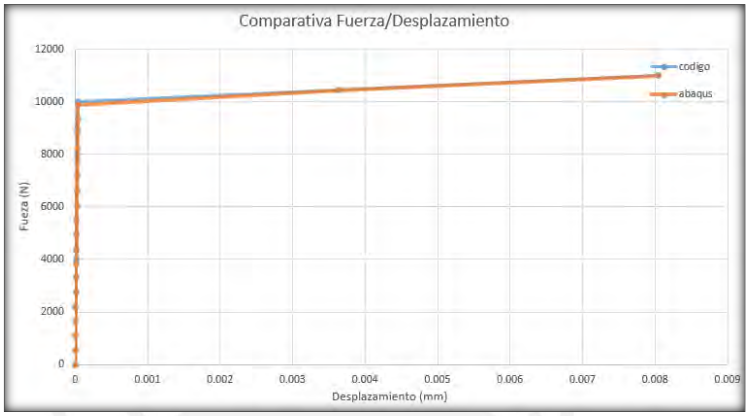
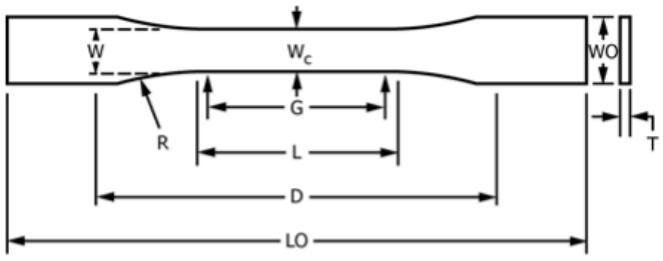


Figura 9.12 **Grafica Fuerza/ Desplazamiento (Elemento 3D)**

Fuente: Elaboración Propia

9.4.2. Ensayo a tracción ASTM D638

A manera de ver las capacidades del código, se realizará el ejemplo de un modelo más avanzado. Dicho modelo estará basado en un ensayo a tracción de una muestra de acero, siguiendo las especificaciones en dimensiones impuestas en la ASTM D638 Specimen Dimensions – Tipo I (AMERICAN SOCIETY FOR TESTING AND MATERIALS, 2014), mostradas en la figura 9.13.



TYPES I, II, III & V

Specimen Dimensions for Thickness, *T*, mm (in.)[A]

Dimensions (see drawings)	7 (0.28) or under		Over 7 to 14 (0.28 to 0.55), incl	4 (0.16) or under		Tolerances
	Type I	Type II	Type III	Type IV[B]	Type V[C,D]	
W—Width of narrow section[E,F]	13 (0.50)	6 (0.25)	19 (0.75)	6 (0.25)	3.18 (0.125)	±0.5 (±0.02)[B,C]
L—Length of narrow section	57 (2.25)	57 (2.25)	57 (2.25)	33 (1.30)	9.53 (0.375)	±0.5 (±0.02)[C]
WO—Width overall, min[G]	19 (0.75)	19 (0.75)	29 (1.13)	19 (0.75)	...	+ 6.4 (+ 0.25)
WO—Width overall, min[G]	...	...	...	...	9.53 (0.375)	+ 3.18 (+ 0.125)
LO—Length overall, min[H]	165 (6.5)	183 (7.2)	246 (9.7)	115 (4.5)	63.5 (2.5)	no max (no max)
G—Gage length[I]	50 (2.00)	50 (2.00)	50 (2.00)	...	7.62 (0.300)	±0.25 (±0.010)[C]
G—Gage length[I]	...	...	...	25 (1.00)	...	±0.13 (±0.005)
D—Distance between grips	115 (4.5)	135 (5.3)	115 (4.5)	65 (2.5)[J]	25.4 (1.0)	±5 (±0.2)
R—Radius of fillet	76 (3.00)	76 (3.00)	76 (3.00)	14 (0.56)	12.7 (0.5)	±1 (±0.04)[C]
RO—Outer radius (Type IV)	...	...	...	25 (1.00)	...	±1 (±0.04)

Figura 9.13 **Dimensiones del espécimen modelado**

Fuente: (AMERICAN SOCIETY FOR TESTING AND MATERIALS, 2014)

Conocidas las dimensiones del espécimen se realizó un modelo geométrico en SolidWorks, importándolo a Abaqus para aplicar el mesh correspondiente al elemento. Todo esto se muestra gráficamente en la figura 9.14.

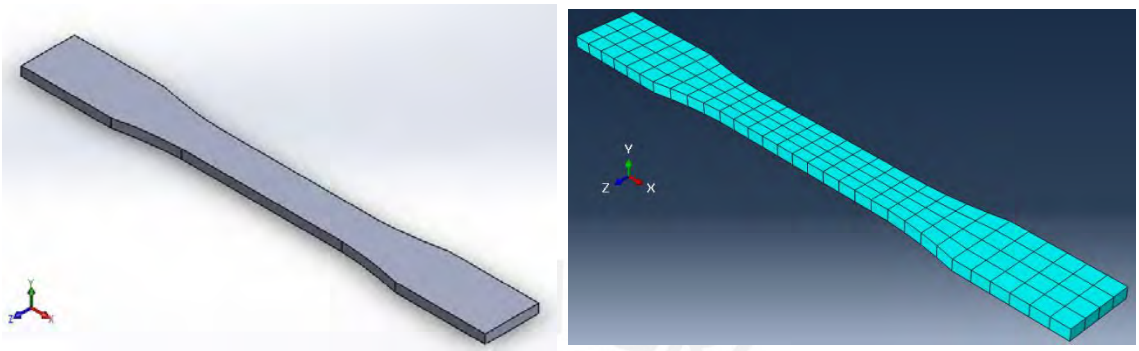


Figura 9.14 **Espécimen de acero, modelo geométrico y mesh**

Fuente: SolidWorks - Abaqus / Elaboración Propia

Para simular el ensayo a tracción (sin llegar a la rotura) se aplicó una carga longitudinal en un extremo del eje X, la cual corresponde a 30 000 Newton. De igual manera se restringió el movimiento en la parte inicial y final de la barra, esto para simular los puntos de anclaje de la misma (Figura 9.15).

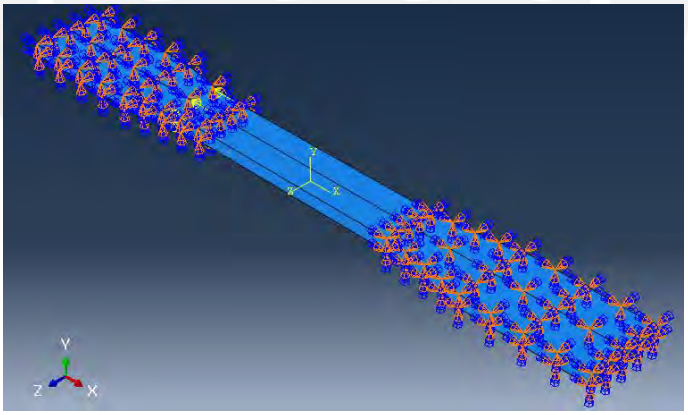


Figura 9.15 **Espécimen de acero, Carga y restricciones**

Fuente: SolidWorks - Abaqus / Elaboración Propia

En la tabla 17 se muestran las propiedades del material aplicadas y correspondientes a un espécimen de acero comúnmente ensayado.

Tabla 17. **Propiedades del sistema 3D – Ensayo Tracción**

Fuente: Elaboración Propia

Propiedad	***Cantidad***
Mod. Elasticidad (E)	$2.069 * 10^{11}\ Pa$
Mod. Plástico (H)	$1.0 * 10^{8}\ Pa$
Mod. de Poisson (υ)	0.29
Coef. Lame (λ)	$110.747\ x\ 10^{9}$
Mod. de corte (μ)	$80.1938\ x\ 10^{9}$
Esfuerzo de fluencia (σy)	$4.0\ x\ 10^{8}\ Pa$

En la gráfica 9.16 se muestra el modelo bilineal con endurecimiento por deformación aplicado.

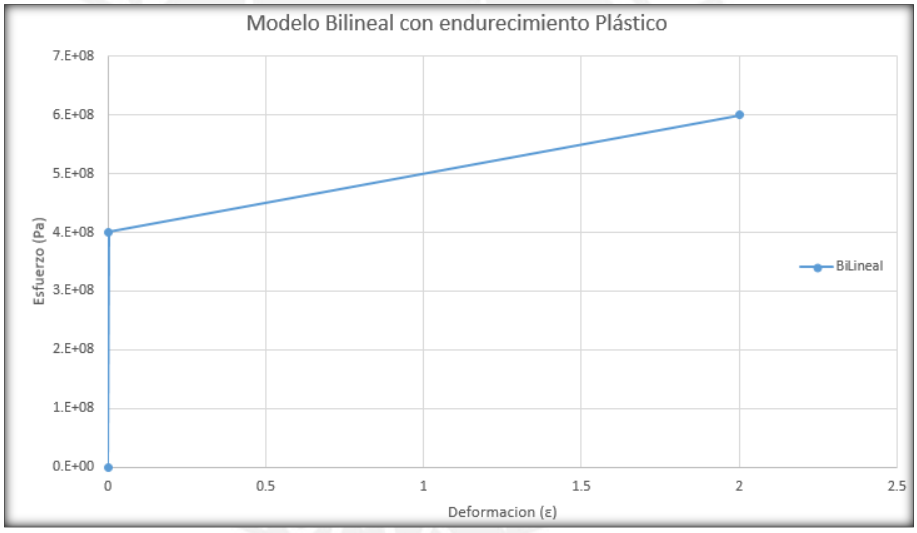


Figura 9.16 **Mod. bilineal con End. plástico (Ensayo a tracción)**

Fuente: Elaboración Propia

Importada la geometría del mesh y conocida las propiedades del material se realizó el análisis del mismo a través del código NLFEA optimizado y su respectiva comprobación mediante Abaqus. En la figura 9.17 se muestra los resultados en forma de una gráfica Fuerza vs Desplazamiento en un punto central de aplicación de la carga, de igual manera en la figura 9.18 se muestra el desplazamiento para la iteración final de aplicación de la carga.

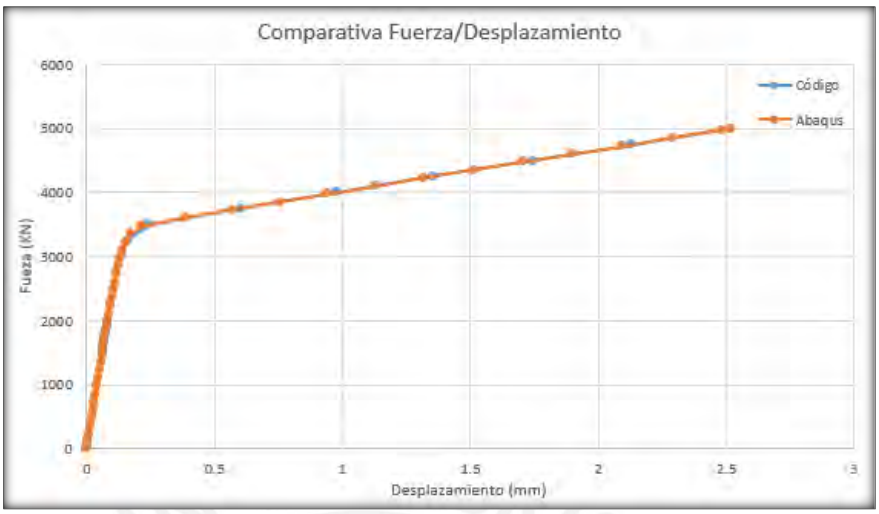


Figura 9.17 **Grafica Fuerza/ Desplazamiento (Ensayo a tracción)**

Fuente: Elaboración Propia

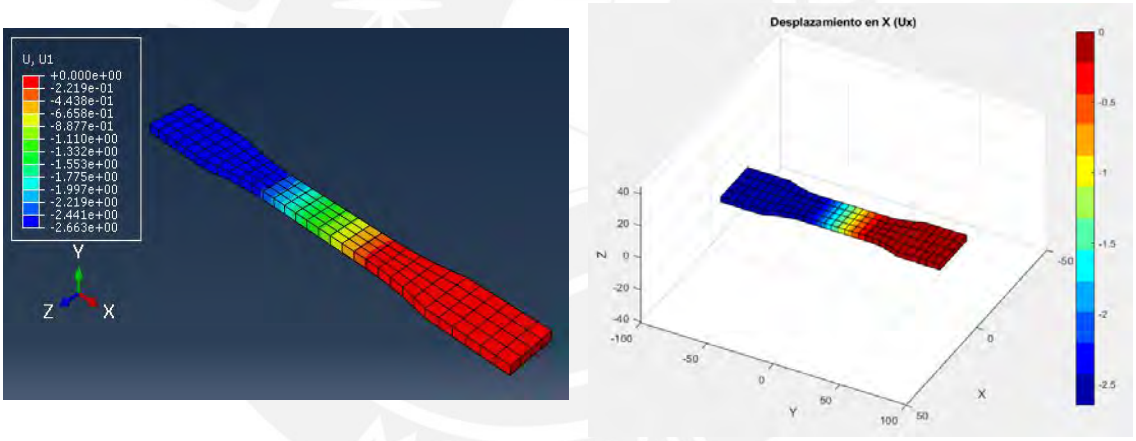


Figura 9.18 **Desp. en X, Comparativa Abaqus/Código (Ens. Tracción)**

Fuente: Abaqus, MatLab / Elaboración Propia

Como se muestran en las figuras 9.17 y 9.18 los resultados obtenidos mediante la aplicación del código y Abaqus son prácticamente iguales, teniendo en cuenta que el ejemplo aplicativo realizo es solo por fines académicos y a manera de corroborar el correcto funcionamiento del código optimizado.

CAPÍTULO 10

Conclusiones y Trabajos Futuros

CAPÍTULO X: CONCLUSIONES Y TRABAJOS FUTUROS

10.1 CONCLUSIONES

- La optimización de un código enfocado al FEM tiene un papel de vital importancia en su aplicación, esto debido al uso de extensas matrices en cada iteración de los bucles. Al aplicar los conceptos básicos de optimización discutidos en el capítulo 7 se puede lograr una mejora cercana al 100% en el tiempo de procesamiento de datos para sistemas complejos.

- El uso de diagramas de flujo para interrelacionar la parte aplicativa con la teórica mejora considerablemente la compresión que existe en el mismo, teniendo en cuenta que dicha herramienta es independiente del tipo de lenguaje de programación que se esté aplicando y está orientada a profesionales con conocimientos básicos de programación.

- Gran parte de las publicaciones enfocadas al uso aplicativo del FEM mediante códigos se olvidan de su ciclo de desarrollo, esto conlleva a que profesionales con baja o incluso mediana experiencia en la programación tengan dificultades en su entendimiento. Estando lo anterior reflejado en la falsa complejidad que se tiene sobre el método, la presente investigación toma en cuenta estos factores y brinda una metodóloga de aprendizaje teórica/aplicativa conjunta, concluyendo ser una forma de solución ante dicha problemática.

- El uso de la herramienta MatLab para el desarrollo del código fue de gran apoyo debido a que dicho software está orientado a un campo matemático, brindando al usuario facilidad en el uso de herramientas y visualización directa de variables (resultados). Sin embargo, dicho software presente dificultades si se desea elaborar un programa autónomo y avanzado, ya que la compilación/exportación del mismo (package) hace necesaria la instalación del propio software o una librería completa de herramientas para su ejecución, este limitante debe tenerse en cuenta si se desea elaborar algún programa complejo orientado al FEM.

- La verificación de resultados en los ejemplos aplicativos desarrollados concluye que se pueden evaluar estructuras medianamente complejas a partir de códigos

sencillos como fue el elaborado en la presente investigación, basándose en un análisis mediante el FEM.

- La aplicación del FEM en un campo lineal y no lineal es algo sencillo de programar, necesitando solamente el uso de funciones y bucles para su aplicación, este hecho se corrobora con el propio código anexado en la presente investigación y demuestra que para la aplicación del FEM mediante ordenadores no son necesarios conocimientos avanzados en la elaboración de códigos.

10.2 TRABAJOS FUTUROS

- La investigación realizada estuvo orientada en gran parte a un campo estático lineal, realizándose: códigos para diversos elementos, pautas para su optimización, bases teóricas, entre otros. Se espera que, a partir de la base dejada, se realicen investigaciones futuras, con la misma premisa, las cuales abarquen un campo dinámico.

- El uso del software MatLab cumple a totalidad los objetivos planteados en la presente investigación, siendo sencillo de entender al estar orientado a un campo matemático. Sin embargo, el mismo presenta problemas si se desea elaborar un código más avanzado, por lo que se espera que se elaboren/publiquen códigos usando otros lenguajes de programación, los cuales estén orientados a un uso profesional visto desde el campo de la codificación.

- La presente investigación tomo en cuenta en su código diversos tipos de elementos, realizando un análisis de los mismos de una forma sencilla con la cantidad mínima de nodos por elemento (4 nodos – bidimensional / 8 nodos – tridimensional), esto a motivo de dar facilidad en el entendimiento al lector. Teniendo en claro que el FEM es un método complejo que conlleva distintas formas posibles de analizar un elemento ya sea por el número de nodos, metodología en la formulación de ecuaciones, tipo de solución numérica aplicada, entre otros, se espera que futuras investigaciones orientadas a la aplicación del FEM tengan en cuenta esto y aporten nuevas codificaciones que incorporen otras formas de análisis.

REFERENCIAS BIBLIOGRÁFICAS

AMERICAN SOCIETY FOR TESTING AND MATERIALS. (2014). D638 − 14. *Standard Test Method for Tensile Properties of Plastics*, 17. https://doi.org/10.1520/D0638-14.1

Bang, H., & Kwon, Y. W. (2000). *The finite element method using MATLAB*. CRC press.

Beer, F. P., Johnston, E. R., DeWolf, J. T., & Mazurek, D. F. (2007). *Mecánica de materiales* (Vol. 2). McGraw-Hill.

Bernales, J. B. (1993). *Losas de Concreto Armado*. Peru: Instituto de Ingeniería Sísmica.

Fernández, M. C. C. (2009). Manual básico de Matlab. *Edit. Complutense, Madrid*.

Joyanes Aguilar, L. (2003). *Fundamentos de programación: algoritmos y estructura de datos y objetos*.

Kattan, P. I. (2010). *MATLAB guide to finite elements: an interactive approach*. Springer Science & Business Media.

Khennane, A. (2013). *Introduction to finite element analysis using MATLAB and Abaqus*. CRC Press.

Kim, N.-H. (2014). *Introduction to nonlinear finite element analysis*. Springer Science & Business Media.

Martín, G., Toledo, F., & Cerverón, V. (1995). Fundamentos de inform tica y programaci n. *Fundamentos de Informática y Programación*, 1–44. Retrieved from http://robotica.uv.es/pub/Libro/PDFs/CAPI1.pdf

Mathworks. (2011). Online Documentation. *Test*, 1–7. https://doi.org/10.1016/0376-7388(91)80092-K

Mathworks. (2016). MATLAB - Mathworks - MATLAB & Simulink. https://doi.org/2016-11-26

MathWorks, S. T. (2018). Which type of function call provides better performance in MATLAB? Retrieved September 20, 2003, from MATLAB Answers website: https://la.mathworks.com/matlabcentral/answers/99537-which-type-of-function-call-provides-better-performance-in-matlab#answer_108884

Oñate, E. (1995). Cálculo de estructuras por el método de elementos finitos. *CIMNE,*

Barcelona.

Oñate, E. (2009). *Structural Analysis with the Finite Element Method. Linear Statics. Vol. 1: Basis and Solids*. https://doi.org/10.1007/978-1-4020-8733-2

Oñate, E. (2013). *Structural analysis with the finite element method. Linear statics: volume 2: beams, plates and shells*. Springer Science & Business Media.

Ottazzi Pasino, G. A. (2011). *Material de Apoyo para la Enseñanza de los Cursos de Análisis Estructural*.

Pólya, G. (1945). How to solve it. Princeton. *New Jersey: Princeton University*.

Rubio, G. C., & Romero, M. V. (2010). *M todo del elemento finito: Fundamentos y aplicaciones con ANSYS.* México: Limusa.

Timoshenko, S. (1946). *Teoría de la elasticidad*. El Ateneo.

Vázquez, M. (2001). El método de los elementos finitos aplicado al análisis estructural. 1ª. *Edición: España: Noela*.

Zienkiewicz, O C, Taylor, R. L., Wisniewski, K., Zhu, J. Z., & Nithiarasu, P. (2004). *El método de los elementos finitos. Vol. 1, Las Bases*. CIMNE.

Zienkiewicz, Olgierd Cecil, Taylor, R. L., Zienkiewicz, O. C., & Taylor, R. L. (1977). *The finite element method* (Vol. 36). McGraw-hill London.

ANEXOS

ANEXO Nº01

Algoritmos de Programación (Diagramas de Flujo).

Diagrama de flujo: General en el FEM (Rango Lineal)

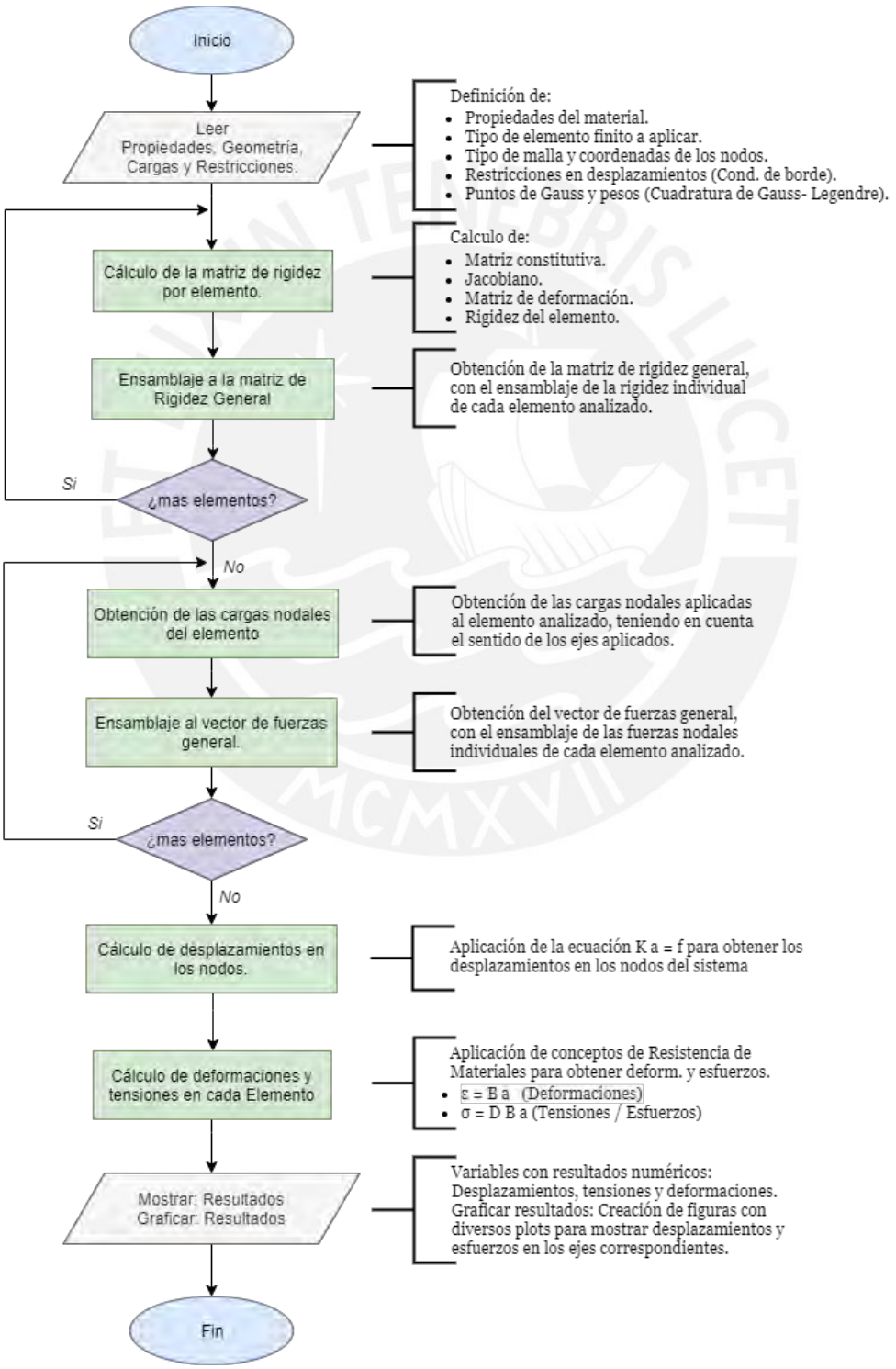


Diagrama de flujo. Elemento Barra

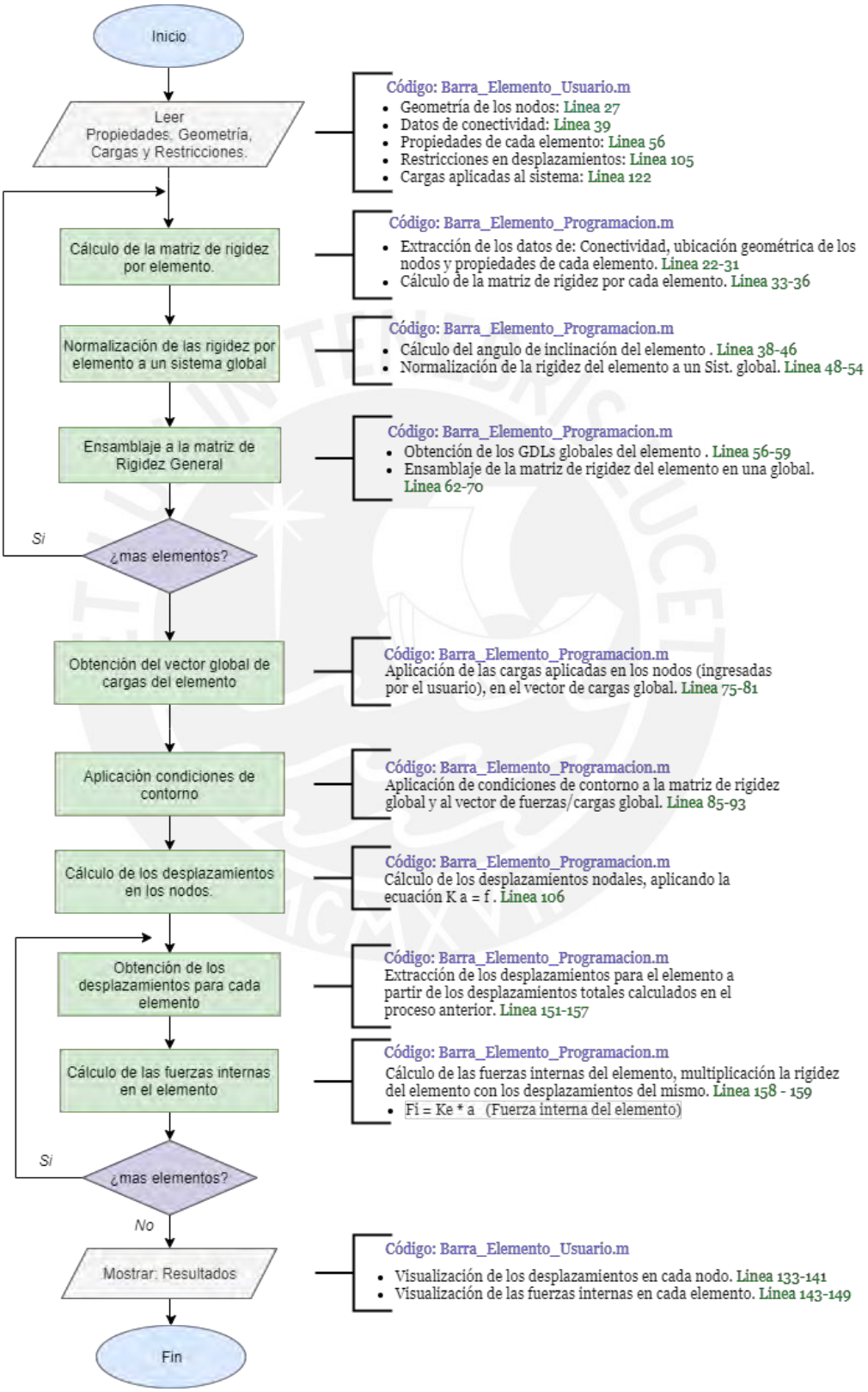


Diagrama de flujo. Elemento Viga – Timoshenko

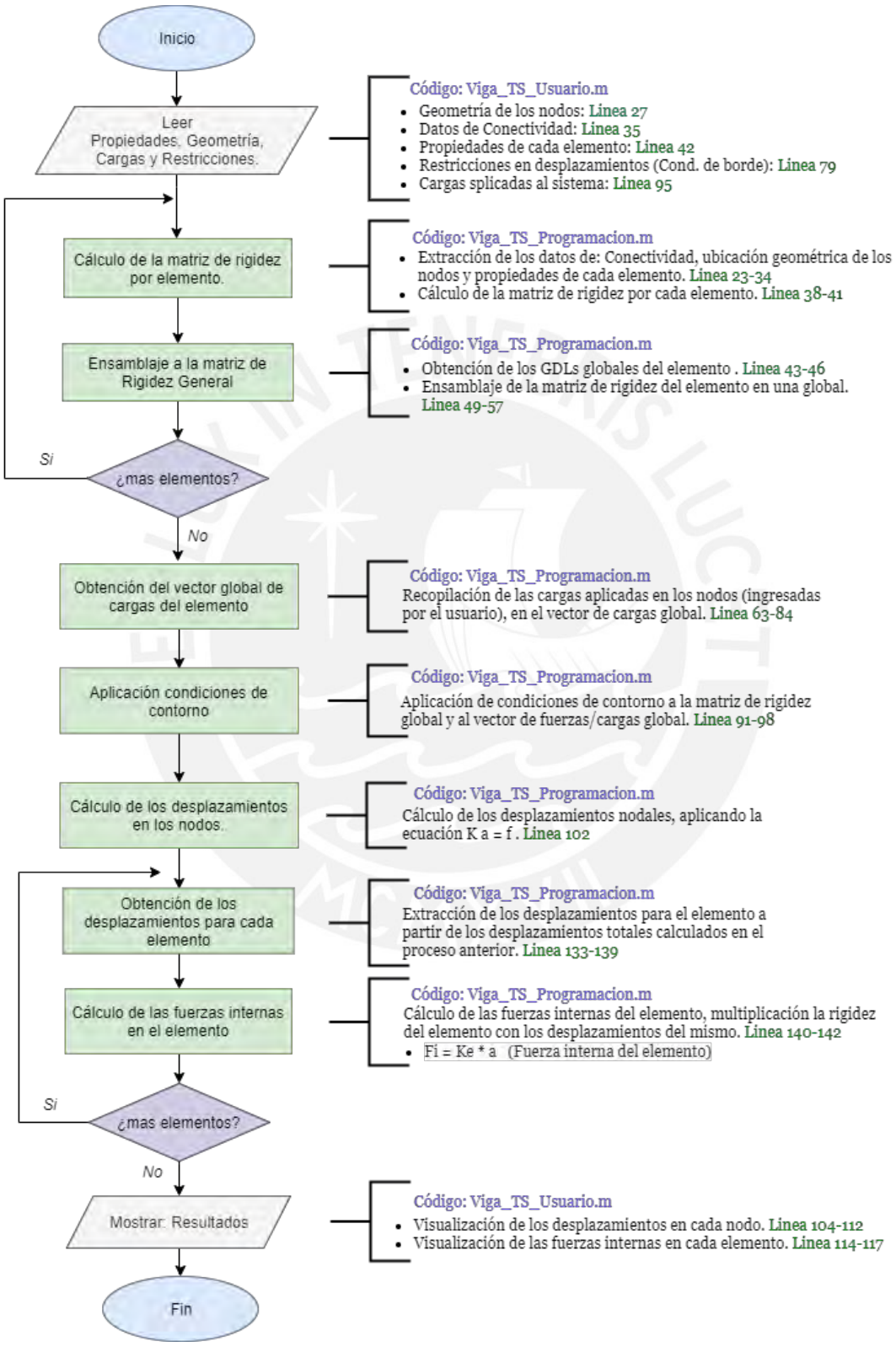
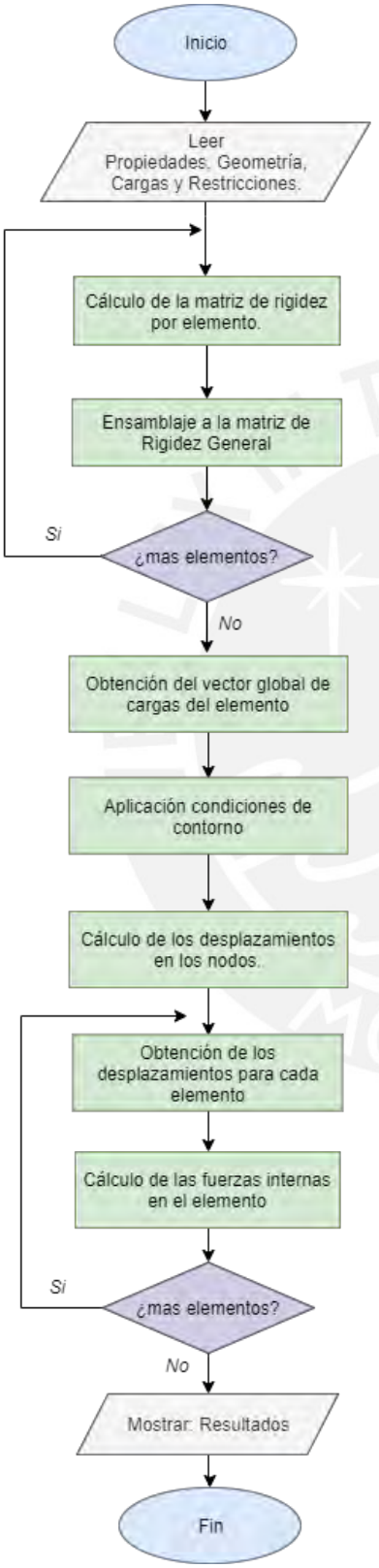


Diagrama de flujo. Elemento Viga - Bernoulli



Código: Viga_EB_Usuario.m
- Geometría de los Nodos: Linea 27
- Datos de Conectividad: Linea 33
- Propiedades de cada elemento: Linea 38
- Restricciones en desplazamientos: Linea 73
- Cargas Aplicadas al sistema: Linea 94

Código: Viga_EB_Programacion.m
- Extracción de los datos de: Conectividad, ubicación geométrica de los nodos y propiedades de cada elemento. Linea 23-31
- Cálculo de la matriz de rigidez por cada elemento. Linea 34-49

Código: Viga_EB_Programacion.m
- Obtención de los GDLs globales del elemento . Linea 51-54
- Ensamblaje de la matriz de rigidez del elemento en una global. Linea 57-65

Código: Viga_EB_Programacion.m
Recopilación de las cargas aplicadas en los nodos (ingresadas por el usuario), en el vector de cargas global. Linea 70-92

Código: Viga_EB_Programacion.m
Aplicación de condiciones de contorno a la matriz de rigidez global y al vector de fuerzas/cargas global. Linea 94-106

Código: Viga_EB_Programacion.m
Cálculo de los desplazamientos nodales, aplicando la ecuación K a = f . Linea 110

Código: Viga_EB_Programacion.m
Extracción de los desplazamientos para el elemento a partir de los desplazamientos totales calculados en el proceso anterior. Linea 149-155

Código: Viga_EB_Programacion.m
Cálculo de las fuerzas internas del elemento, multiplicación la rigidez del elemento con los desplazamientos del mismo. Linea 157-159
- Fi = Ke * a (Fuerza interna del elemento)

Código: Viga_EB_Usuario.m
- Visualización de los desplazamientos en cada nodo. Linea 111-119
- Visualización de las fuerzas internas en cada elemento. Linea 121-124

Diagrama de flujo. Elemento Bidimensional Gauss-Legendre

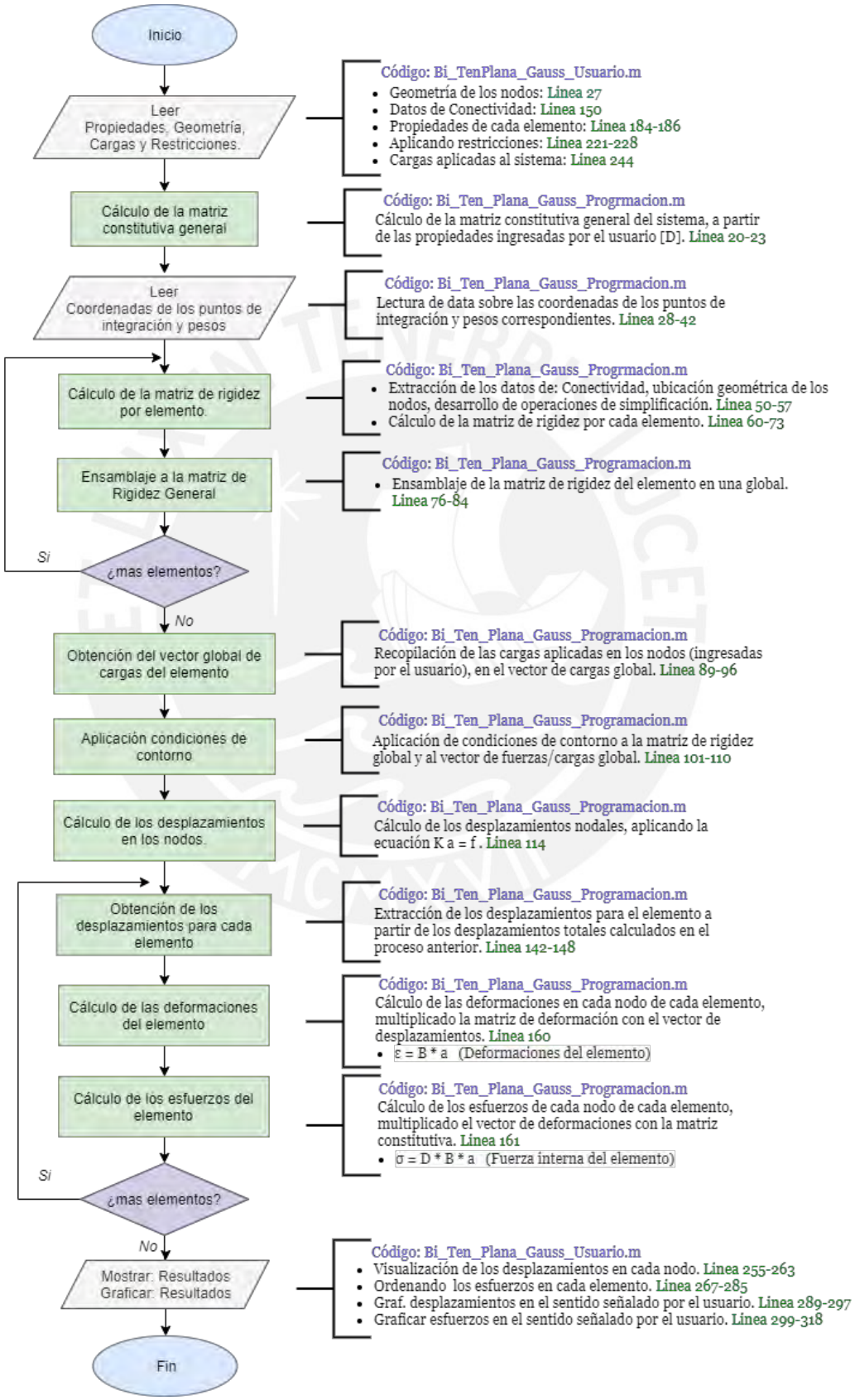


Diagrama de flujo. Elemento Plate-Kirchhoff

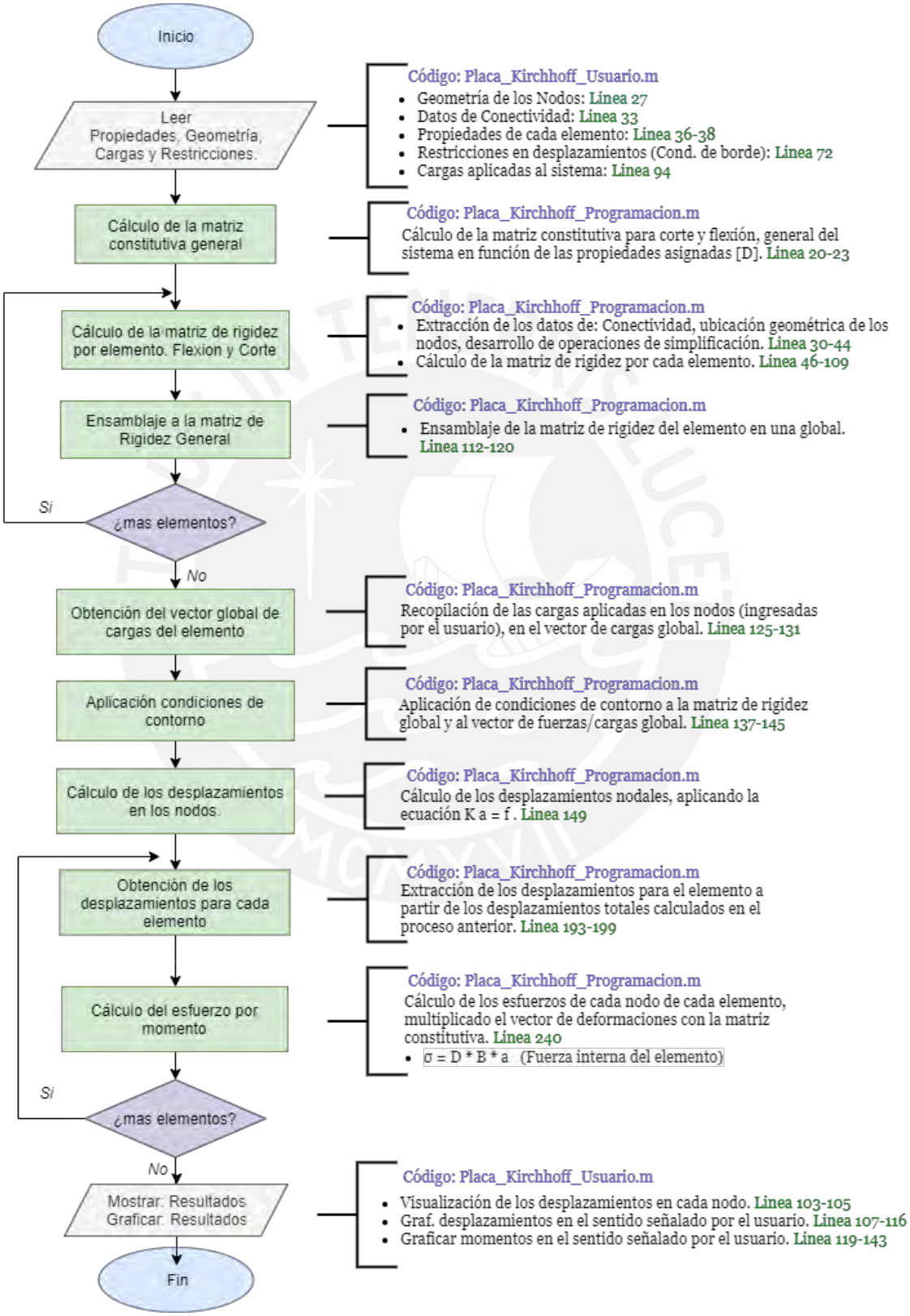


Diagrama de flujo. Elemento Plate-ReissnerMindlin

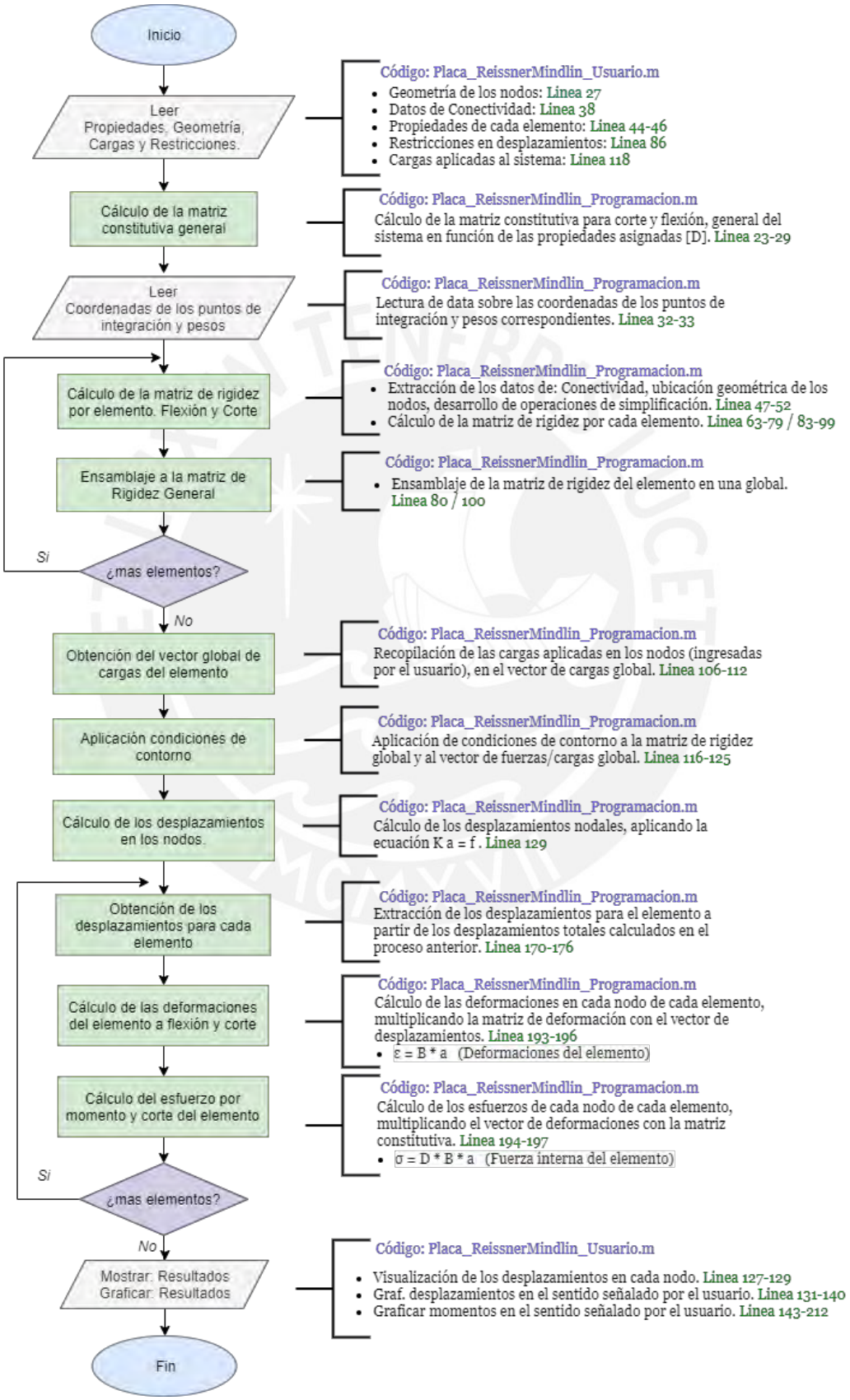


Diagrama de flujo. Elemento Tridimensional

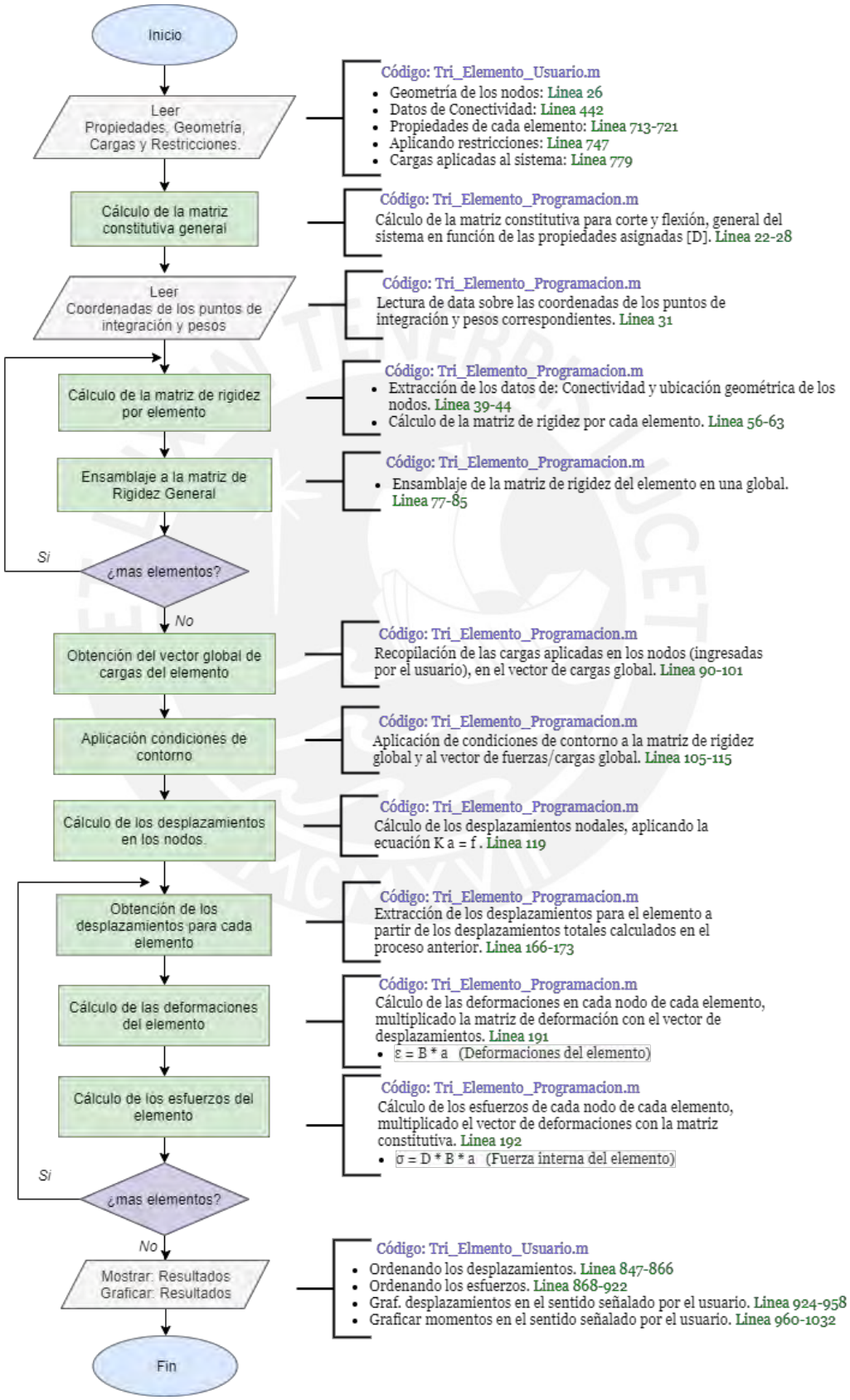


Diagrama de flujo: Pesos y Coordenadas en los Puntos de Integración

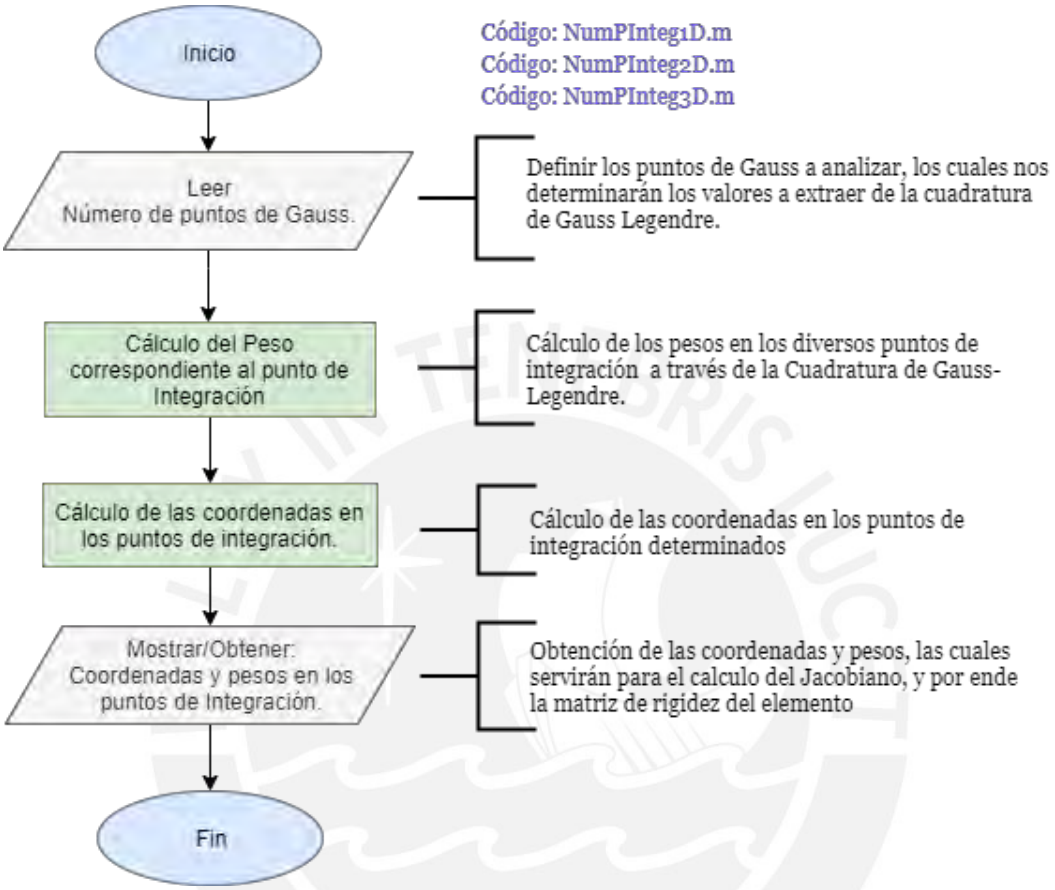


Diagrama de flujo: Matriz de Deformación del elemento

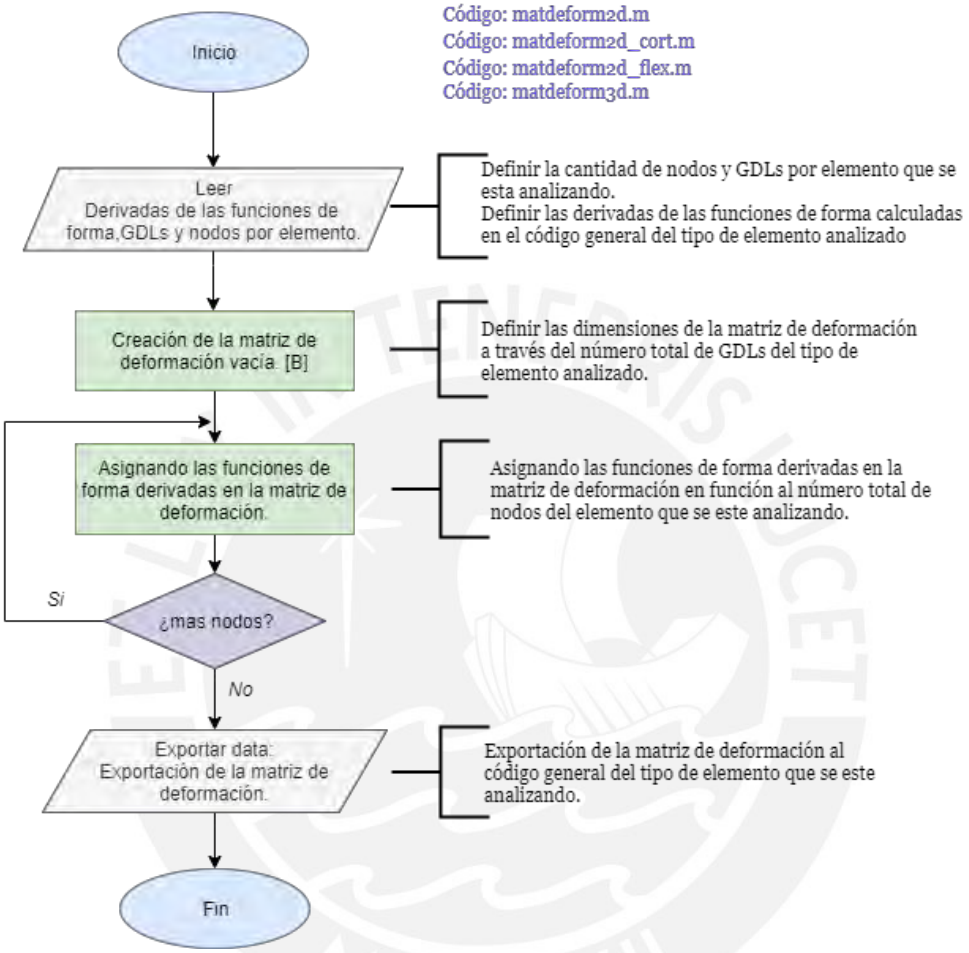


Diagrama de flujo: Funciones de forma del elemento (Coord. Globales)

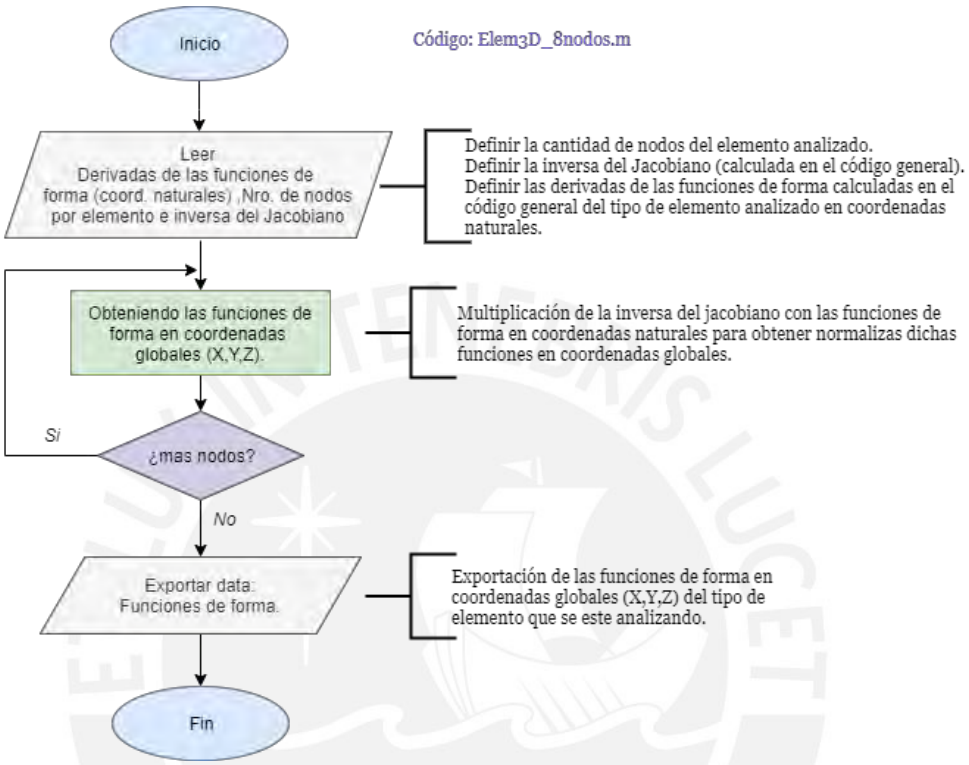
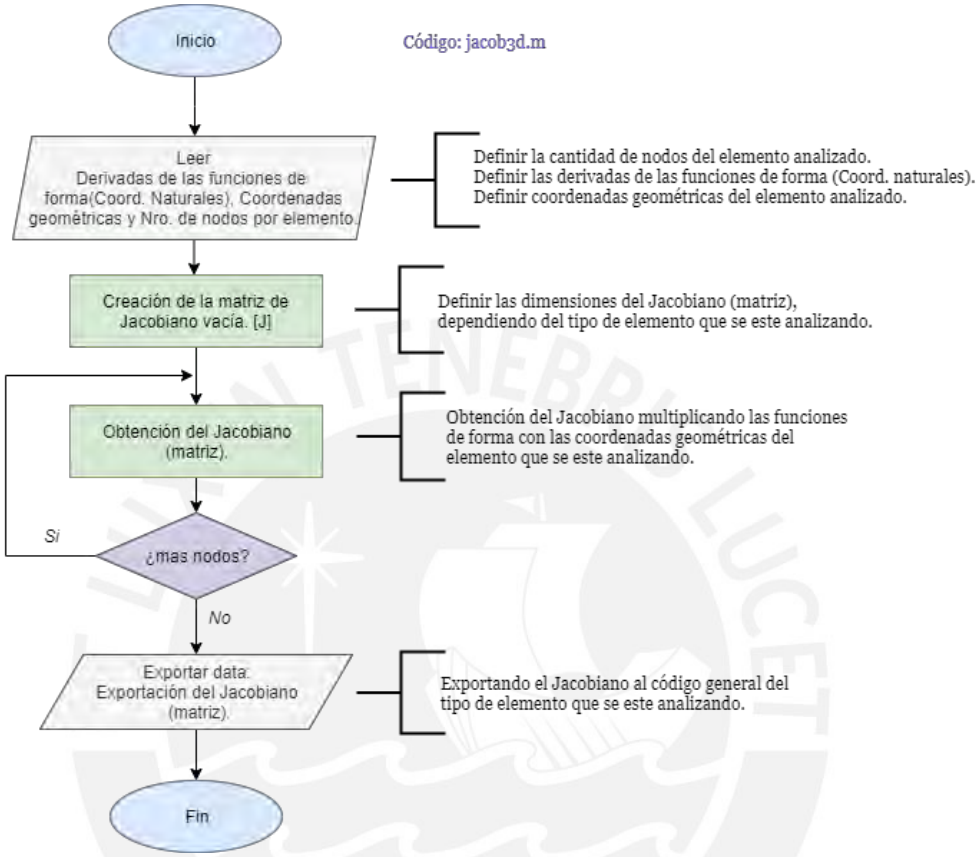


Diagrama de flujo: Jacobiano



ANEXO Nº02
Código base para el análisis con Elementos Finitos

CÓDIGO PARA BARRAS/EST. ARTICULADAS
(Parte Aplicativa del Código)

```
%{
Propósito:
    Analizar un sistema compuesto de Elementos Articuladas con el FEM.
    Código para el procesamiento de datos interno
Descripción de datos de ingreso:
    - Ninguno, se utilizarán datos del programa Barra_Elemento_Usuario.m
Diseño: Ing. Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Noviembre 2019
%}

function[desp,Fuerza] =
Barra_Elemento_Programacion(nnd,nel,conect,geom,prop,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL
)

%   Creación de la matrices y vectores vacíos
KG = zeros(sigdl);      % Rigidez del sistema  (Rigidez General)
F  = zeros(sigdl,1);    % Fuerzas en el sistema

%%%%%%%%%% ------- OBTENCIÓN DE RIGIDEZ EN EL SISTEMA ------- %%%%%%%%%%

for i=1:nel     % Bucle para cada uno de los elementos

    nodo_1=conect(i,1);     % Nodos del elemento analizado en la iteración
    nodo_2=conect(i,2);

    x1=geom(nodo_1,1); y1=geom(nodo_1,2);   % Coord.(x,y) de los nodos
    x2=geom(nodo_2,1); y2=geom(nodo_2,2);

    L = sqrt((x2-x1)^2 + (y2-y1)^2);    % Longitud del elemento

    E = prop(i,1);      % Modulo de elasticidad del elemento
    A = prop(i,2);      % Área Sec.Transversal del elemento

    kl=[E*A/L 0 -E*A/L 0 ; ...  % Matriz de rigidez por elemento
        0 0 0 0 ; ...
        -E*A/L 0 E*A/L 0 ; ...
        0 0 0 0 ];

    if(x2-x1)==0    % Calculo del Angulo de Inclinación del Elemento
        if(y2>y1)
            theta=2*atan(1);
        else
            theta=-2*atan(1);
        end
    else
        theta=atan((y2-y1)/(x2-x1));
```

```
    end

    ML = [cos(theta) -sin(theta) 0 0 ; ...   % Matriz de Localización
          sin(theta) cos(theta) 0 0 ; ...
          0 0 cos(theta) -sin(theta) ; ...
          0 0 sin(theta) cos(theta) ];

    %   Matriz de rigidez del elemento, en un sistema global
    kg=ML*kl*ML';

    g=[Numnd(nodo_1,1);    % Vector con los GDLs globales del elemento
       Numnd(nodo_1,2);
       Numnd(nodo_2,1);
       Numnd(nodo_2,2)];

    %   Ensamblaje de la rigidez del elemento en la Global
    for k=1:elgdl
        if g(k) ~= 0
            for j=1: elgdl
                if g(j) ~= 0
                    KG(g(k),g(j))= KG(g(k),g(j)) + kg(k,j);
                end
            end
        end
    end
end

%%%%%%%%% ------- OBTENCIÓN FUERZAS EN EL SISTEMA ------- %%%%%%%%%

for i=1:nnd     % Bucle para cada uno de los Nodos
    for j=1:ndgdl   % Bucle para cada uno de los GDLs por nodo
        if Numnd(i,j)~= 0
            F(Numnd(i,j)) = Carg(i,j); % Asignando la carga en un vector
        end
    end
end

%%%%%%% ------- APLICANDO RESTRICCIONES AL SISTEMA ------- %%%%%%%

n=length(ResGDL);
 for i=1:n
    c=ResGDL(i);
    for j=1:sigdl
       KG(c,j)=0;
    end
    KG(c,c)=1;
    F(c)=0;
 end

%%%%%%% ------- CORRECCIÓN DE ERRORES PARA BARRAS ------- %%%%%%%

 for j=1:sigdl
     if KG(j,j) == 0
         KG(j,j)=1;
```

```
     else
     end
 end

%%%%%%%%%% ------- CÁLCULO DE DESPLAZAMIENTOS ------- %%%%%%%%%%

delta = KG\F;
desp = delta;

%%%%%%%%%% ------- CÁLCULO DE FUERZAS INTERNAS ------- %%%%%%%%%%

for i=1:nel     % Bucle para cada uno de los elementos
    nodo_1=conect(i,1);     % Nodos del elemento analizado en la iteración
    nodo_2=conect(i,2);

    x1=geom(nodo_1,1); y1=geom(nodo_1,2);   % Coord.(x,y) de los nodos
    x2=geom(nodo_2,1); y2=geom(nodo_2,2);

    L = sqrt((x2-x1)^2 + (y2-y1)^2);    % Longitud del elemento

    E = prop(i,1);      % Módulo de elasticidad del elemento
    A = prop(i,2);      % Área Sec. Transversal del elemento

    kl=[E*A/L 0 -E*A/L 0 ; ...  % Matriz de rigidez por elemento
        0 0 0 0 ; ...
        -E*A/L 0 E*A/L 0 ; ...
        0 0 0 0 ];

    if(x2-x1)==0    % Calculo del Angulo de Inclinación del Elemento
        if(y2>y1)
            theta=2*atan(1);
        else
            theta=-2*atan(1);
        end
    else
        theta=atan((y2-y1)/(x2-x1));
    end

    ML = [cos(theta) -sin(theta) 0 0 ; ...   % Matriz de Localización
         sin(theta) cos(theta) 0 0 ; ...
         0 0 cos(theta) -sin(theta) ; ...
         0 0 sin(theta) cos(theta) ];

    %   Matriz de rigidez del elemento, en un sistema global
    kg=ML*kl*ML';

    g=[Numnd(nodo_1,1);   % Vector con los GDLs globales del elemento
       Numnd(nodo_1,2);
       Numnd(nodo_2,1);
       Numnd(nodo_2,2)];

    for j=1:elgdl   % Bucle para la obtención de los Desp. del elemento
        if g(j)== 0
            edg(j)=0.;   % Asignación de 0 en el desplazamiento (si esta restringido)
```

```
        else
            edg(j) = delta(g(j));
        end
    end
    fg = kg*edg';   % Obtención del vector en un sistema local
    fl=ML'*fg ;     % Modificación a un sistema global
    Fuerza(i) = fl(3);
end
end
```

Published with MATLAB® R2019b

CÓDIGO PARA BARRAS/EST. ARTICULADAS
(Ingreso de datos del Usuario)

```
%{
Propósito:
    Analizar un sistema compuesto de Elementos Articuladas con el FEM.
    Código para el ingreso de datos por el usuario.
Descripción de datos de ingreso:
    - Coordenadas de las barras.
    - Cargas impuestas en los diversos nodos.
    - Condiciones de borde en el sistema.
Variables de ingreso:
    Especificadas en el código.
Diseño: Ing. Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Noviembre 2019
%}

%%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%%

clear           % Limpieza del espacio de trabajo
clc             % Limpieza de la ventana de comandos
close all       % Cierre de todas las figuras existentes
format short e  % Formato numérico a trabajar

%%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%%

%   Ingreso de coordenadas de los nodos. (Geométria)
%   Coordenadas X/Y por nodo
geom = [0. 0.; ...
        1. 2.; ...
        2. 0.; ...
        3. 2.; ...
        4. 0.; ...
        5. 2.; ...
        6. 0.; ...
        7. 2.; ...
        8. 0.] ;

%   Ingreso de información sobre conectividad entre nodos. (Conectividad)
```

```
%   Conectividad nodo inicial/ nodo final por elemento
conect = [1 2 ; ...
        1 3 ; ...
        2 3 ; ...
        2 4 ; ...
        3 4 ; ...
        3 5 ; ...
        4 5 ; ...
        4 6 ; ...
        5 6 ; ...
        5 7 ; ...
        6 7 ; ...
        6 8 ; ...
        7 8 ; ...
        7 9 ; ...
        8 9 ] ;

%   Propiedades geométricas del material por barra (Elasticidad/Área)
prop = [30.e6 0.02 ; ...
        30.e6 0.045 ; ...
        30.e6 0.02 ; ...
        30.e6 0.045 ; ...
        30.e6 0.02 ; ...
        30.e6 0.045 ; ...
        30.e6 0.02 ; ...
        30.e6 0.045 ; ...
        30.e6 0.02 ; ...
        30.e6 0.045 ; ...
        30.e6 0.02 ; ...
        30.e6 0.045 ; ...
        30.e6 0.02 ; ...
        30.e6 0.045 ; ...
        30.e6 0.02 ];

%   Información sobre el sistema (Automático)
nne = 2;        % Número de nodos por elemento
ndgdl = 2;      % Número de GDLs por nodo
[nnd,~] = size(geom);       % Número de nodos en el sistema
[nel,~] = size(conect);     % Número de elementos
elgdl = nne*ndgdl;          % Número de GDLs por elemento
sigdl = nnd*ndgdl;          % Número de GDLs en el sistema

%   Aplicando numeración a todos los GDLs en el sistema
Numnd = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end
```

```
%   Condiciones de frontera por nodo
nf = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end

%   Ingresar las restricciones en los GDLs correspondientes [Nodo,GDL]
nf(1,1) = 0;                % Aplicando restricciones en los GDLs
nf(1,2) = 0;
nf(9,2) = 0;

%   Obteniendo los GDLs restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end

%      Cargas/Fuerzas aplicadas al sistema. [Nodo,:]
Carg = zeros(nnd, ndgdl);   % Creación de la matriz de cargas vacías (0)
Carg(2,:)=[15. 0.];         % Aplicando cargas a los nodos [Carga en X,Carga en Y]
Carg(3,:)=[0. -5.];
Carg(4,:)=[0. -7.];
Carg(7,:)=[0. -10.];

%   Ejecución de la función para el procesamiento de datos
disp('Procesando Datos');
[desp,Fuerza] =
Barra_Elemento_Programacion(nnd,nel,conect,geom,prop,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL
);
disp('Procesamiento de Datos Terminado');
%   Ordenando resultados
disp('Resultados');
for i=1:nnd
    for j=1:ndgdl
        nodo_desp(i,j) = 0;
        if nf(i,j)~= 0
            nodo_desp(i,j) = desp(nf(i,j));
        end
    end
end
display(nodo_desp);

for i=1:nel
```

```
    if Fuerza(i) > 0
        fprintf(' %g, %9.2f, %s\n',i, Fuerza(i), 'Tension');
    else
        fprintf(' %g, %9.2f, %s\n',i, Fuerza(i), 'Compression');
    end
end

%   Limpieza de variables innecesarias
clear Carg conect elgdl geom i j ResGDL prop n ndgdl nel nf nnd nne desp...
    Numnd sigdl
```

Published with MATLAB® R2019b

CÓDIGO PARA VIGAS/EST. RETICULARES – Método. Timoshenko (Parte Aplicativa del Código)

```
{
Propósito:
    Analizar un sistema compuesto de una viga con la teoría de
    Timoshenko mediante el FEM.
    Código para el procesamiento de datos internos
Descripción de datos de ingreso:
    - Ninguno, se utilizaran datos del programa Viga_TS_Usuario
Diseñador: Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Noviembre 2019
%}

function[desp,Fuerza] =
Viga_TS_Programacion(nnd,nel,conect,geom,prop,elgdl,CargNod,CargElem,sigdl,Numnd,ndgdl,R
esGDL)

%   Creación de la matrices y vectores vacíos
KG = zeros(sigdl);      % Rigidez del sistema  (Rigidez General)
F  = zeros(sigdl,1);    % Fuerzas en el sistema

%%%%%%%%%% ------- OBTENCIÓN DE RIGIDEZ EN EL SISTEMA ------- %%%%%%%%%%

for i=1:nel     % Bucle para cada uno de los elementos

    % Matriz de rigidez por elemento (Rigidez)
    nodo_1=conect(i,1);     % Nodos del elemento analizado en la iteración
    nodo_2=conect(i,2);

    x1=geom(nodo_1);        % Coord.(x) de los nodos
    x2=geom(nodo_2);

    L = abs(x2-x1);         % Longitud del elemento

    EI = prop(i,1)*prop(i,2);   % Modulo de elast. e inercia del elemento
    MG = prop(i,3);             % Modulo de corte del elemento
    Ar = prop(i,4);             % Área del elemento
```

```
    c=EI/L;
    d=(5/6)*MG*Ar/(4*L);
    kl= [ 4*d         2*d*L     -4*d        2*d*L;...   % Rigidez por elemento
         2*d*L       c+d*L^2   -2*d*L     -c+d*L^2;...
        -4*d        -2*d*L      4*d       -2*d*L;...
         2*d*L      -c+d*L^2   -2*d*L      c+d*L^2];

    g=[Numnd(nodo_1,1);    % Vector condiciones de contorno
       Numnd(nodo_1,2);
       Numnd(nodo_2,1);
       Numnd(nodo_2,2)];

    %   Ensamblaje de la rigidez del elemento en la global
    for k=1:elgdl
        if g(k) ~= 0
            for j=1: elgdl
                if g(j) ~= 0
                    KG(g(k),g(j))= KG(g(k),g(j)) + kl(k,j);
                end
            end
        end
    end

end

%%%%%%%%% ------- OBTENCIÓN DE FUERZAS EN EL SISTEMA ------- %%%%%%%%%

for i=1:nnd     % Bucle para obtener la carga directamente aplicada al nodo
    for j=1:ndgdl
        if Numnd(i,j)~= 0
            F(Numnd(i,j)) = CargNod(i,j);
        end
    end
end

for i=1:nel
    nodo_1=conect(i,1);
    nodo_2=conect(i,2);
    g=[Numnd(nodo_1,1);    % Vector condiciones de contorno
       Numnd(nodo_1,2);
       Numnd(nodo_2,1);
       Numnd(nodo_2,2)];

    for j=1:elgdl   % Asignando la carga o momento al nodo por elemento
        if g(j)~= 0
            F(g(j))= F(g(j)) + CargElem(i,j);
        end
    end
end

%%%%%%% ------- APLICANDO CONDICIONES DE BORDE AL SISTEMA ------- %%%%%%%

n=length(ResGDL);
```

```
sdof=size(KG);

 for i=1:n
    c=ResGDL(i);
    for j=1:sdof
       KG(c,j)=0;
    end
    KG(c,c)=1;
    F(c)=0;
 end

%%%%%%%%%% ------- CALCULO DE DESPLAZAMIENTOS ------- %%%%%%%%%%

delta = KG\F;
desp = delta;

%%%%%%%%%% ------- CALCULO DE FUERZAS INTERNAS ------- %%%%%%%%%%

for i=1:nel     % Bucle para cada uno de los elementos
    % Matriz de rigidez por elemento (Rigidez)
    nodo_1=conect(i,1);     % Nodos del elemento analizado en la iteración
    nodo_2=conect(i,2);

    x1=geom(nodo_1);        % Coord.(x) de los nodos
    x2=geom(nodo_2);

    L = abs(x2-x1);         % Longitud del elemento

    EI = prop(i,1)*prop(i,2);   % Modulo de Elast. e Inercia del elemento
    MG = prop(i,3);             % Modulo de corte del elemento
    Ar = prop(i,4);             % Área del elemento

    c=EI/L;
    d=(5/6)*MG*Ar/(4*L);
    kl= [ 4*d         2*d*L     -4*d        2*d*L;...   % Rigidez por elemento
         2*d*L       c+d*L^2   -2*d*L     -c+d*L^2;...
        -4*d        -2*d*L      4*d       -2*d*L;...
         2*d*L      -c+d*L^2   -2*d*L      c+d*L^2];

    g=[Numnd(nodo_1,1);    % Vector condiciones de contorno
       Numnd(nodo_1,2);
       Numnd(nodo_2,1);
       Numnd(nodo_2,2)];

    for j=1:elgdl   % Bucle para la obtención de los Desp. del elemento
        if g(j)== 0
            edg(j)=0.;   % Asignación de 0 en el desplazamiento (si esta restringido)
        else
            edg(j) = delta(g(j));
        end
    end
    fl = kl*edg';           % Obtención del vector de fuerzas
    f0 = CargElem(i,:);     % Modificación a un sistema global
    Fuerza(i,:) = fl-f0';
```

```
end
end
```

Published with MATLAB® R2019b

CÓDIGO PARA VIGAS/EST. RETICULARES – Método. Timoshenko (Ingreso de Datos del Usuario)

```
%{
Propósito:
    Analizar un sistema compuesto de una viga con la teoría de
    Timoshenko mediante el FEM.
    Código para el ingreso de datos por el usuario.
Descripción de datos de ingreso:
    - Coordenadas de las barras.
    - Cargas impuestas en los diversos nodos.
    - Condiciones de borde en el sistema.
Variables de ingreso:
    Especificadas en el código.
Diseño:  Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Noviembre 2019
%}

%%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%%

clear           % Limpieza del espacio de trabajo
clc             % Limpieza de la ventana de comandos
close all       % Cierre de todas las figuras existentes
format short e  % Formato numérico a trabajar

%%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%%

%   Ingreso de coordenadas de los nodos. (Geometría)
geom = [0.; ...     % Coordenadas en X de la viga
        2.; ...
        4.; ...
        6.; ...
        8.; ...
        10];

%   Ingreso de datos de conectividad entre nodos. (Conectividad)
conect = [1 2 ; ... % Conectividad de nodos en el extremo del elemento
          2 3 ; ...
          3 4 ; ...
          4 5 ; ...
          5 6];

%   Propiedades geométricas del material por barra (Elasticidad/Inercia/Mod. Corte)
prop = [10^7 1/12 3.8*10^6 1; ... % Propiedades del material para cada elemento
        10^7 1/12 3.8*10^6 1; ...
        10^7 1/12 3.8*10^6 1; ...
```

```
        10^7 1/12 3.8*10^6 1; ...
        10^7 1/12 3.8*10^6 1];

%   Información sobre el sistema (Automático)
nne = 2;        % Número de nodos por elemento
ndgdl = 2;      % Número de GDLs por nodo
[nnd,~] = size(geom);       % Número de nodos en el sistema
[nel,~] = size(conect);     % Número de elementos
elgdl = nne*ndgdl;          % Número de GDLs por elemento
sigdl = nnd*ndgdl;          % Número de GDLs en el sistema

%   Dando numeración a todos los GDLs en el sistema
Numnd = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

%   Condiciones de frontera por nodo
nf = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
nf(1,1) = 0;                % Aplicando restricciones en los GDLs
nf(6,2) = 0;

%   Obteniendo los GDLs restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end

%      Cargas/Fuerzas aplicadas al sistema
CargNod = zeros(nnd, 2);    % Carga en y/o momento aplicado directamente
CargNod(6,1) = 50;

CargElem = zeros(nnd, elgdl);   % Creación de la matriz de cargas vacías (0)
```

```
%   Ejecutando la función para el procesamiento de datos
disp('Procesando Datos');
[desp,Fuerza] =
Viga_TS_Programacion(nnd,nel,conect,geom,prop,elgdl,CargNod,CargElem,sigdl,Numnd,ndgdl,R
esGDL);

%   Ordenando los desplazamientos
for i=1:nnd
    for j=1:ndgdl
        nodo_desp(i,j) = 0;
        if nf(i,j)~= 0
            nodo_desp(i,j) = desp(nf(i,j));
        end
    end
end
display(nodo_desp);

for i=1:nel
    fprintf(' %g, %9.2f, %9.2f, %9.2f, %9.2f\n',i, ...
    Fuerza(i,1),Fuerza(i,2),Fuerza(i,3),Fuerza(i,4));
end

disp('Procesamiento de Datos Terminado');

%   Limpieza de variables innecesarias
clear Carg conect elgdl geom i j ResGDL prop n ndgdl nel nf nnd nne desp...
    Numnd sigdl
```

Published with MATLAB® R2019b

CÓDIGO PARA VIGAS/EST. RETICULARES – Método. E.Bernoulli
(Parte Aplicativa del Código)

```
%{
Propósito:
    Analizar un sistema compuesto de una viga con la teoría de
    Euler-Bernoulli mediante el FEM.
    Código para el procesamiento de datos. Interno
Descripción de datos de ingreso:
    - Ninguno, se utilizaran datos del código Viga_EB_Usuario
Diseño:  Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Junio 2019
%}

function[desp,Fuerza] =
Viga_EB_Programacion(nnd,nel,conect,geom,prop,elgdl,CargNod,CargElem,sigdl,Numnd,ndgdl,R
esGDL,Rot)

%   Creacion de la matrices y vectores vacíos
KG = zeros(sigdl);      % Rigidez del sistema  (Rigidez General)
F  = zeros(sigdl,1);    % Fuerzas en el sistema
```

```
%%%%%%%%% ------- OBTENCIÓN DE RIGIDEZ EN EL SISTEMA ------- %%%%%%%%%

for i=1:nel     % Bucle para cada uno de los elementos

    nodo_1=conect(i,1);     % Nodos del elemento analizado en la iteración
    nodo_2=conect(i,2);

    x1=geom(nodo_1);        % Coord.(x) de los nodos
    x2=geom(nodo_2);

    L = abs(x2-x1);         % Longitud del elemento

    EI = prop(i,1)*prop(i,2);   % Modulo de Elast. e Inercia del elemento

    % Obtención  de la rigidez por elemento
    if Rot(i, 1) == 0
        kl=[ 3*EI/L^3 0 -3*EI/L^3 3*EI/L^2 ; ...
             0 0 0 0 ; ...
            -3*EI/L^3 0 3*EI/L^3 -3*EI/L^2 ; ...
             3*EI/L^2 0 -3*EI/L^2 3*EI/L ];
    elseif Rot(i, 2) == 0
        kl=[ 3*EI/L^3 3*EI/L^2 -3*EI/L^3 0 ; ...
             3*EI/L^2 3*EI/L -3*EI/L^2 0 ; ...
            -3*EI/L^3 -3*EI/L^2 3*EI/L^3 0 ; ...
             0 0 0 0 ] ;
    else
        kl=[ 12*EI/L^3 6*EI/L^2 -12*EI/L^3 6*EI/L^2 ; ...
              6*EI/L^2 4*EI/L -6*EI/L^2 2*EI/L ; ...
            -12*EI/L^3 -6*EI/L^2 12*EI/L^3 -6*EI/L^2 ; ...
              6*EI/L^2 2*EI/L -6*EI/L^2 4*EI/L ];
    end

    g=[Numnd(nodo_1,1);    % Vector condiciones de contorno
       Numnd(nodo_1,2);
       Numnd(nodo_2,1);
       Numnd(nodo_2,2)];

    %   Ensamblaje de la rigidez del elemento en la Global
    for k=1:elgdl
        if g(k) ~= 0
            for j=1: elgdl
                if g(j) ~= 0
                    KG(g(k),g(j))= KG(g(k),g(j)) + kl(k,j);
                end
            end
        end
    end
end

%%%%%%%%% ------- OBTENCIÓN FUERZAS EN EL SISTEMA ------- %%%%%%%%%

for i=1:nnd     % Bucle para obtener la carga directamente aplicada al nodo
    for j=1:ndgdl
```

```
        if Numnd(i,j)~= 0
            F(Numnd(i,j)) = CargNod(i,j);
        end
    end
end

for i=1:nel
    nodo_1=conect(i,1); % Nodos del elemento analizado en la iteración
    nodo_2=conect(i,2);

    g=[Numnd(nodo_1,1);    % Vector condiciones de contorno
       Numnd(nodo_1,2);
       Numnd(nodo_2,1);
       Numnd(nodo_2,2)];

    for j=1:elgdl   % Asignando la carga o momento al nodo por elemento
        if g(j)~= 0
            F(g(j))= F(g(j)) + CargElem(i,j);
        end
    end
end

%%%%%%% ------- APLICANDO CONDICIONES DE BORDE AL SISTEMA ------- %%%%%%%

n=length(ResGDL);
sdof=size(KG);

 for i=1:n
    c=ResGDL(i);
    for j=1:sdof
       KG(c,j)=0;
    end
    KG(c,c)=1;
    F(c)=0;
 end

%%%%%%%%%% ------- CALCULO DE DESPLAZAMIENTOS ------- %%%%%%%%%%

delta = KG\F;
desp = delta;

%%%%%%%%%% ------- CALCULO DE FUERZAS INTERNAS ------- %%%%%%%%%%

for i=1:nel     % Bucle para cada uno de los elementos
    nodo_1=conect(i,1);     % Nodos del elemento analizado en la iteración
    nodo_2=conect(i,2);

    x1=geom(nodo_1);        % Coord.(x) de los Nodos
    x2=geom(nodo_2);

    L = abs(x2-x1);         % Longitud del elemento

    EI = prop(i,1)*prop(i,2);   % Modulo de Elast. e Inercia del elemento
```

```
    % Obtención de la rigidez por elemento
    if Rot(i, 1) == 0
        kl=[ 3*EI/L^3 0 -3*EI/L^3 3*EI/L^2 ; ...
             0 0 0 0 ; ...
            -3*EI/L^3 0 3*EI/L^3 -3*EI/L^2 ; ...
             3*EI/L^2 0 -3*EI/L^2 3*EI/L ];
    elseif Rot(i, 2) == 0
        kl=[ 3*EI/L^3 3*EI/L^2 -3*EI/L^3 0 ; ...
             3*EI/L^2 3*EI/L -3*EI/L^2 0 ; ...
            -3*EI/L^3 -3*EI/L^2 3*EI/L^3 0 ; ...
             0 0 0 0 ] ;
    else
        kl=[ 12*EI/L^3 6*EI/L^2 -12*EI/L^3 6*EI/L^2 ; ...
              6*EI/L^2 4*EI/L -6*EI/L^2 2*EI/L ; ...
            -12*EI/L^3 -6*EI/L^2 12*EI/L^3 -6*EI/L^2 ; ...
              6*EI/L^2 2*EI/L -6*EI/L^2 4*EI/L ];
    end

    g=[Numnd(nodo_1,1);    % Vector condiciones de contorno
       Numnd(nodo_1,2);
       Numnd(nodo_2,1);
       Numnd(nodo_2,2)];

    for j=1:elgdl   % Bucle para la obtención de los Desp. del elemento
        if g(j)== 0
            edg(j)=0.;   % Asignación de 0 en el desplazamiento (si esta restringido)
        else
            edg(j) = delta(g(j));
        end
    end

    fl = kl*edg';           % Obtención del vector en un sistema local
    f0 = CargElem(i,:);     % Modificación a un sistema global
    Fuerza(i,:) = fl-f0';
end
end
```

Published with MATLAB® R2019b

CÓDIGO PARA VIGAS/EST. RETICULARES – Método. E.Bernoulli (Ingreso de Datos del Usuario)

```
%{
Propósito:
    Analizar un sistema compuesto de una viga con la teoría de
    Euler-Bernoulli mediante el FEM.
    Código para el ingreso de datos por el usuario.
Descripción de datos de ingreso:
    - Coordenadas de las barras.
    - Cargas impuestas en los diversos nodos.
    - Condiciones de borde en el sistema.
Variables de ingreso:
```

```
    Especificadas en el código.
Diseño: Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Setiembre 2019
%}

%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%

clear           % Limpieza del espacio de trabajo
clc             % Limpieza de la ventana de comandos
close all       % Cierre de todas las figuras existentes
format short e  % Formato numérico a trabajar

%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%

%   Ingreso de coordenadas de los nodos. (Geometría)
geom = [0.; ...     % Coordenadas en X del elemento
        4000.; ...
        9000.; ...
        16000];

%   Ingreso de información sobre la conectividad entre nodos extremos. (Conectividad)
conect = [1 2 ; ...
        2 3 ; ...
        3 4 ] ;

%   Propiedades del material para cada elemento  (Elasticidad/Inercia)
prop = [200000 200.e+6; ...
        200000 200.e+6; ...
        200000 200.e+6 ];

%   Información sobre el sistema (Automático)
nne = 2;        % Número de nodos por elemento
ndgdl = 2;      % Número de GDLs por nodo
[nnd] = size(geom,1);       % Número de nodos en el sistema
[nel] = size(conect,1);     % Número de elementos
elgdl = nne*ndgdl;          % Número de GDLs por elemento
sigdl = nnd*ndgdl;          % Número de GDLs en el sistema

%   Dando numeración a todos los GDLs en el sistema
Numnd = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para asignar la numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

%   Condiciones de frontera por nodo
nf = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
```

```
for i=1:nnd     % Bucle para asignar la numeración
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
nf(1,1) = 0;    % Aplicando restricciones en los GDLs
nf(1,2) = 0;
nf(2,1) = 0;
nf(3,1) = 0;
nf(4,1) = 0;
nf(4,2) = 0;

%   Obteniendo los GDLs restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle para asignar la numeración
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end

%   Aplicación de rotulas al sistema
Rot = ones(nel, 2);

%      Cargas/Fuerzas aplicadas al sistema
CargNod = zeros(nnd, 2);    % Carga en y/o momento aplicado directamente

CargElem = zeros(nnd, elgdl);  % Carga en y/o momento aplicado distribuidamente en el
elemento
CargElem(1,:)=[ -1.e4 -1.e7 -1.e4 1.e7];
CargElem(2,:)=[ -1.e4 -8.333e6 -1.e4 8.333e6];

%   Limpieza de variables innecesarias
clear c n i j

%   Ejecutando la función para el procesamiento de datos
disp('Procesando Datos');
[desp,Fuerza] =
Viga_EB_Programacion(nnd,nel,conect,geom,prop,elgdl,CargNod,CargElem,sigdl,Numnd,ndgdl,R
esGDL,Rot);

%   Ordenando resultados
disp('Resultados');

for i=1:nnd
    for j=1:ndgdl
        nodo_desp(i,j) = 0;
        if nf(i,j)~= 0
            nodo_desp(i,j) = desp(nf(i,j));
```

```
        end
    end
end
display(nodo_desp);

for i=1:nel
    fprintf(' %g, %9.2f, %9.2f, %9.2f, %9.2f\n',i, ...
    Fuerza(i,1),Fuerza(i,2),Fuerza(i,3),Fuerza(i,4));
end

disp('Procesamiento de Datos Terminado');

%   Limpieza de variables innecesarias
clear Carg conect elgdl geom i j ResGDL prop n ndgdl nel nf nnd nne desp...
    Numnd sigdl Rot
```

Published with MATLAB® R2019b

CÓDIGO PARA ELEMENTOS PLATE – Método. Kirchhoff (Parte Aplicativa del Código)

```
%{
Propósito:
    Analizar un sistema compuesto de elementos cuadriláteros uniforme tipo
    Plate mediante el método de Kirchhoff y haciendo uso de puntos de integración.
    Código para el procesamiento de datos interno
Descripción de datos de ingreso:
    - Ninguno, se utilizaran datos del programa Placa_Kirchhoff_Usuario
Diseño: Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Noviembre 2019
%}

function[desp,Element_Forces] =
Placa_Kirchhoff_Programacion(nf,nne,nnd,nel,conect,geom,E,vu,Esp,elgdl,Carg,sigdl,Numnd,
ndgdl,ResGDL)

%   Creación de la matrices y vectores vacíos
KG = zeros(sigdl);      % Rigidez del sistema  (Rigidez General)
F  = zeros(sigdl,1);    % Fuerzas en el sistema

%%%%%%%%%% ------- OBTENCIÓN DE LA MATRIZ CONSTITUTIVA ------- %%%%%%%%%%
DR= E*(Esp^3)/(12*(1.-vu*vu));
MCF=DR*[1 vu 0. ;...
     vu 1 0. ;...
     0. 0. (1.-vu)/2];

%%%%%%%%%% ------- OBTENCIÓN DE RIGIDEZ EN EL SISTEMA ------- %%%%%%%%%%

for i=1:nel     % Bucle para cada uno de los elementos

    %   Obtención de los datos de conectividad del elemento
```

```
l=0;
for k=1: nne
    for j=1:ndgdl
        l=l+1;
        g(l)=nf(conect(i,k),j);
    end
end

%   Obtención de los datos geométricos del elemento
coord=zeros(nne,2);
for k=1: nne
    for j=1:2
        coord(k,j)=geom(conect(i,k),j);
    end
end

base=abs(coord(1,1)-coord(2,1));    % Longitud de la base del elemento
altura=abs(coord(4,2)-coord(1,2));  % Longitud de la altura del elemento

%   Matriz de rigidez por elemento (Rigidez Local-Coord Locales)
a=base/2;
b=altura/2;

k1=[6 6*a 0 -6 6*a 0 -3 3*a 0 3 3*a 0;...
    6*a 8*a^2 0 -6*a 4*a^2 0 -3*a 2*a^2 0 3*a 4*a^2 0;...
    0 0 0 0 0 0 0 0 0 0 0 0;...
    -6 -6*a 0 6 -6*a 0 3 -3*a 0 -3 -3*a 0;...
    6*a 4*a^2 0 -6*a 8*a^2 0 -3*a 4*a^2 0 3*a 2*a^2 0;...
    0 0 0 0 0 0 0 0 0 0 0 0;...
    -3 -3*a 0 3 -3*a 0 6 -6*a 0 -6 -6*a 0;...
    3*a 2*a^2 0 -3*a 4*a^2 0 -6*a 8*a^2 0 6*a 4*a^2 0;...
    0 0 0 0 0 0 0 0 0 0 0 0;...
    3 3*a 0 -3 3*a 0 -6 6*a 0 6 6*a 0;...
    3*a 4*a^2 0 -3*a 2*a^2 0 -6*a 4*a^2 0 6*a 8*a^2 0;...
    0 0 0 0 0 0 0 0 0 0 0 0];
k1=(b/6*a^3)*k1;

k2=[6 0 6*b 3 0 3*b -3 0 3*b -6 0 6*b;...
    0 0 0 0 0 0 0 0 0 0 0 0;...
    6*b 0 8*b^2 3*b 0 4*b^2 -3*b 0 2*b^2 -6*b 0 4*b^2;...
    3 0 3*b 6 0 6*b -6 0 6*b -3 0 3*b;...
    0 0 0 0 0 0 0 0 0 0 0 0;...
    3*b 0 4*b^2 6*b 0 8*b^2 -6*b 0 4*b^2 -3*b 0 2*b^2;...
    -3 0 -3*b -6 0 -6*b 6 0 -6*b 3 0 -3*b;...
    0 0 0 0 0 0 0 0 0 0 0 0;...
    3*b 0 2*b^2 6*b 0 4*b^2 -6*b 0 8*b^2 -3*b 0 4*b^2;...
    -6 0 -6*b -3 0 -3*b 3 0 -3*b 6 0 -6*b;...
    0 0 0 0 0 0 0 0 0 0 0 0;...
    6*b 0 4*b^2 3*b 0 2*b^2 -3*b 0 4*b^2 -6 0 8*b^2];
k2=(a/(6*b^3))*k2;

k3=[1 a b -1 0 -b 1 0 0 -1 -a 0;...
    a 0 2*a*b 0 0 0 0 0 0 -a 0 0;...
    b 2*a*b 0 -b 0 0 0 0 0 0 0 0;...
```

```
        -1 0 -b 1 -a b -1 a 0 1 0 0;...
        0 0 0 -a 0 -2*a*b a 0 0 0 0 0;...
        -b 0 0 b -2*a*b 0 0 0 0 0 0 0;...
        1 0 0 -1 a 0 1 -a -b -1 0 0;...
        0 0 0 a 0 0 -a  0 2*a*b 0 0 0;...
        0 0 0 0 0 0 -b 2*a*b 0 b 0 0;...
        -1 -a 0 1 0 0 -1 0 b 1 a -b;...
        -a 0 0 0 0 0 0 0 0 a 0 -2*a*b;...
        0 0 0 0 0 0 0 0 0 -b -2*a*b 0];
    k3=(vu/(2*a*b))*k3;

    k4=[21 3*a 3*b -21 3*a -3*b 21 -3*a -3*b -21 -3*a 3*b;...
        3*a 8*a^2 0 -3 -2*a^2 0 3*a 2*a^2 0 -3*a -2*a^2 0;...
        3*b 0 8*b^2 -3*b 0 -8*b^2 3*b 0 2*b^2 -3*b 0 -8*b^2;...
        -21 -3*a -3*b 21 -3*a 3*b -21 3*a 3*b 21 3*a -3*b;...
        3*a -2*a^2 0 -3*a 8*a^2 0 3*a -8*a^2 0 -3*a 2*a^2 0;...
        -3*b 0 -8*b^2 3*b 0 8*b^2 -3*b 0 -2*b^2 3*b 0 2*b^2;...
        21 3*a 3*b -21 3*a -3*b 21 -3*a -3*b -21 -3*a 3*b;...
        -3*a 2*a^2 0 3*a -8*a^2 0 -3*a 8*a^2 0 3*a -2*a^2 0;...
        -3*b 0 2*b^2 3*b 0 -2*b^2 -3*b 0 8*b^2 3*b 0 -8*b^2;...
        -21 -3*a -3*b 21 -3*a 3*b -51 3*a 3*b 21 3*a -3*b;...
        -3*a -2*a^2 0 3*a 2*a^2 0 -3*a -2*a^2 0 3*a 8*a^2 0;...
        3*b 0 -8*b^2 -3*b 0 2*b^2 3*b 0 -8*b^2 -3*b 0 8*b^2];
    k4=((1-vu)/(30*a*b))*k4;

    kg=DR*(k1+k2+k3+k4);

    %   Ensamblaje de la rigidez del elemento en la Global
    for k=1:elgdl
        if g(k) ~= 0
            for j=1: elgdl
                if g(j) ~= 0
                    KG(g(k),g(j))= KG(g(k),g(j)) + kg(k,j);
                end
            end
        end
    end
end

%%%%%%%%% ------- OBTENCIÓN FUERZAS EN EL SISTEMA ------- %%%%%%%%%

for i=1:nnd     % Bucle para cada uno de los nodos
    for j=1:ndgdl   % Bucle para cada uno de los GDLs
        if Numnd(i,j)~= 0
            F(Numnd(i,j)) = Carg(i,j);
        end
    end
end

%%%%%%% ------- APLICANDO CONDICIONES DE BORDE AL SISTEMA ------- %%%%%%%

n=length(ResGDL);

 for i=1:n
```

```
    c=ResGDL(i);
    for j=1:sigdl
       KG(c,j)=0;
    end

    KG(c,c)=1;
    F(c)=0;
 end

%%%%%%%%% ------- CALCULO DE DESPLAZAMIENTOS ------- %%%%%%%%%

delta = KG\F;

%%%%%%%%% ------- DESPLAZAMIENTOS ORDENADOS ------- %%%%%%%%%

for i=1:nnd
    for j=1:ndgdl
        node_disp(i,j) = 0;
        if nf(i,j)~= 0
            node_disp(i,j) = delta(nf(i,j)) ;
        end
    end
end
desp=node_disp;

%%%%%%%%%% -- CALCULO DE LOS MOMENTOS Y FUERZAS CORTANTES -- %%%%%%%%%%

%   Calculo de los momentos y fuerzas cortantes en el centro del elemento
i=1;
for iel=1:nel
    %   Obtención de los datos de conectividad del elemento
    l=0;
    for k=1: nne
        for j=1:ndgdl
            l=l+1;
            g(l)=nf(conect(iel,k),j);
        end
    end

    %   Obtención de los datos geométricos del elemento
    coord=zeros(nne,2);
    for k=1: nne
        for j=1:2
            coord(k,j)=geom(conect(iel,k),j);
        end
    end

    base=abs(coord(1,1)-coord(2,1));    % Longitud de la base del elemento
    altura=abs(coord(4,2)-coord(1,2));  % Longitud de la altura del elemento

    a=base/2;
    b=altura/2;

    eld=zeros(elgdl,1); % Inicialización del vector de desplazamientos en 0 para cada
```

```
elemento

    for m=1:elgdl       % Bucle para asignar los desplazamientos y giros en un vector en
General
        if g(m)==0
            eld(m)=0.;
        else
            eld(m)=delta(g(m)); % Recuperación de los desplazamientos en el vector
general
        end
    end

    %   Obtención de la matriz C del sistema
    C=[1 -a -b a^2 a*b b^2 -a^3 -a^2*b -a*b^2 -b^3 a^3*b a*b^3;...
       0 1 0 -2*a -b 0 3*a^2 2*a*b b^2 0 -3*a^2*b -b^3;...
       0 0 1 0 -a -2*b 0 a^2 2*a*b 3*b^2 -a^3 -3*a*b^2;...
       1 a -b a^2 -a*b b^2 a^3 -a^2*b a*b^2 -b^3 -a^3*b -a*b^3;...
       0 1 0 2*a -b 0 3*a^2 -2*a*b b^2 0 -3*a^2*b -b^3;...
       0 0 1 0 a -2*b 0 a^2 -2*a*b 3*b^2 a^3 3*a*b^2;...
       1 a b a^2 a*b b^2 a^3 a^2*b a*b^2 b^3 a^3*b a*b^3;...
       0 1 0 2*a b 0 3*a^2 2*a*b b^2 0 3*a^2*b b^3;...
       0 0 1 0 a 2*b 0 a^2 2*a*b 3*b^2 a^3 3*a*b^2;...
       1 -a b a^2 -a*b b^2 -a^3 a^2*b -a*b^2 b^3 -a^3*b -a*b^3;...
       0 1 0 -2*a b 0 3*a^2 -2*a*b b^2 0 3*a^2*b b^3;...
       0 0 1 0 -a 2*b 0 a^2 -2*a*b 3*b^2 -a^3 -3*a*b^2];

    %   Obtención de la matriz de deformación para los cuatro nodos del elemento
    for intx=1: 4
        %for inty=1: 2
        if intx==1
            x=geom(1,1)-a;
            y=geom(1,2)-b;
        elseif intx==2
            x=geom(2,1)-a;
            y=geom(2,2)-b;
        elseif intx==3
            x=geom(3,1)-a;
            y=geom(3,2)-b;
        else
            x=geom(4,1)-a;
            y=geom(4,2)-b;
        end

        Q=[0 0 0 2 0 0 6*x 2*y 0 0 6*x*y 0;...
           0 0 0 0 0 2 0 0 2*x 6*y 0 6*x*y;...
           0 0 0 0 2 0 0 4*x 4*y 0 6*x^2 6*y^2];
        % Matriz de deformación del elemento
        MD=Q*C^-1;

        % Momentos obtenidos en cada punto del elemento
        Moment=-MCF*MD*eld;
        Element_Forces(i,:)=Moment';
        i=i+1;
    end
```

```
end

display(Element_Forces)
```

Published with MATLAB® R2019b

CÓDIGO PARA ELEMENTOS PLATE – Método. Kirchhoff (Ingreso de Datos del Usuario)

```
%{
Propósito:
    Analizar un sistema compuesto de elementos cuadriláteros uniforme tipo
    Plate mediante el método de Kirchhoff y en base a 4 nodos por elemento.
    Código para el ingreso de datos por el usuario.
Descripción de datos de ingreso:
    - Coordenadas de los nodos en el elemento Plate.
    - Cargas impuestas en los diversos nodos.
    - Restricciones en el sistema.
Variables de ingreso:
    Especificadas en el código.
Diseño: Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Setiembre 2019
%}

%%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%%

clear           % Limpieza del espacio de trabajo
clc             % Limpieza de la ventana de comandos
close all       % Cierre de todas las figuras existentes
format short e  % Formato numérico a trabajar

%%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%%

%   Ingreso de coordenadas de los nodos. (Geométria)
geom = [0.0  0.0;...    % Coordenadas en X,Y del Elem. Plate.
        2.0  0.0;...
        2.0  2.0;...
        0.0  2.0];

%   Ingreso de información sobre la conectividad entre nodos extremos. (Conectividad)
conect = [1 2 3 4];  % Conectividad de nodos en el extremo del elemento

%   Propiedades del material (Elasticidad/Poisson/Espesor)
E=2*10^10;      % Modulo de elasticidad del material
vu=0.3;         % Coeficiente de Poisson del material
Esp=0.10;       % Espesor del elemento plate

%   Información sobre el sistema (Automático)
nne = 4;        % Numero de nodos por elemento
ndgdl = 3;      % Numero de GDLs por nodo
[nnd,~] = size(geom);       % Número de nodos en el sistema
```

```
[nel,~] = size(conect);     % Número de elementos
elgdl = nne*ndgdl;          % Número de GDLs por elemento
sigdl = nnd*ndgdl;          % Número de GDLs en el sistema
npgf = 3;                   % Número de puntos de Gauss a flexión

%   Dando numeración a todos los GDLs en el sistema
Numnd = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para asignar la condición de libre
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

%   Condiciones de frontera por nodo
nf = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para asignar la numeración
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
nf(1,1) = 0; nf(1,3)=0;
nf(1,2)=0;
nf(2,1) = 0; nf(2,3)=0;
nf(2,2)=0;
nf(3,3) = 0;
nf(3,2)=0;
nf(4,1) = 0; nf(4,3)=0;
nf(4,2)=0;

%   Obteniendo los GDLs restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle para extraer los GDLs restringidos
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end

%      Cargas/Fuerzas aplicadas al sistema
Carg = zeros(nnd, ndgdl);   % Creación de la matriz de cargas vacías (0)
Carg(3,1) = 5;              % Aplicando cargas a los nodos

%   Ejecutando la función para el procesamiento de datos
disp('Procesando Datos');
```

```
[desp,Element_Forces] =
Placa_Kirchhoff_Programacion(nf,nne,nnd,nel,conect,geom,E,vu,Esp,elgdl,Carg,sigdl,Numnd,
ndgdl,ResGDL);

%%%%%%%%%%%%%%%%% ------- MOSTRANDO GRÁFICOS ------- %%%%%%%%%%%%%%%%%

% Desplazamientos
UZ=desp(:,1);
UX=desp(:,2);
UY=desp(:,3);

figure
cmin = min(UZ);
cmax = max(UZ);
caxis([cmin cmax]);
patch('Faces',conect, 'Vertices', geom, 'FaceVertexCData',UZ,...
'Facecolor','interp','Marker','.');
colorbar;
title('Desplazamiento en Z (UZ)')
xlabel('B')
ylabel('H')

% Momentos
MX=Element_Forces(:,1);
MY=Element_Forces(:,2);
MXY=Element_Forces(:,3);

figure
cmin = min(MX);
cmax = max(MX);
caxis([cmin cmax]);
patch('Faces',conect, 'Vertices', geom, 'FaceVertexCData',MX,...
'Facecolor','interp','Marker','.');
colorbar;
title('Momento en X (MX)')
xlabel('B')
ylabel('H')

figure
cmin = min(MY);
cmax = max(MY);
caxis([cmin cmax]);
patch('Faces',conect, 'Vertices', geom, 'FaceVertexCData',MY,...
'Facecolor','interp','Marker','.');
colorbar;
title('Momento en Y (MX)')
xlabel('B')
ylabel('H')

disp('Procesamiento de Datos Terminado');

%   Limpieza de variables innecesarias
clear c n i j a b cmax cmin DR x y xcoord ycoord zcoord vu C coord E eld delta...
```

```
    F g k k1 k2 k3 k4 kg KG MCF MD m l intx iel nel nf nnd nne ndgdl node_disp...
    npgf Q ResGDL sigdl Numnd Moment geom elgdl Element_Forces Esp
```

Published with MATLAB® R2019b

CÓDIGO PARA ELEMENTOS PLATE – Método. Reissner Mindlin (Parte Programada del Código)

```
%{
Propósito:
    Analizar un sistema compuesto de elementos cuadriláteros uniforme tipo
    Plate mediante el método de Reisssner Mindlin y en base a 4 nodos por elemento.
    Código para el ingreso de datos por el usuario.
Descripción de datos de ingreso:
    - Ninguno, se utilizaran datos del programa Placa_ReissnerMindlin_Usuario
Diseño: Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Noviembre 2019
%}

function[desp,Element_Forces,Element_Flex,Element_Corte] =
Placa_ReissnerMindlin_Usuario(nglxb,nglyb,nglxs,nglys,nf,nne,nnd,nel,conect,geom,E,vu,Es
p,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL)

%   Creación de la matrices y vectores vacíos
KG = zeros(sigdl);      % Rigidez del sistema  (Rigidez General)
F  = zeros(sigdl,1);    % Fuerzas en el sistema
xcoord=zeros(1,nne);    % Matriz de coordenadas x por elemento(Sistemática)
ycoord=zeros(1,nne);    % Matriz de coordenadas y por elemento(Sistemática)
zcoord=zeros(1,nne);    % Matriz de coordenadas z por elemento(Sistemática)

%%%%%%%%%% ------- OBTENCIÓN DE LA MATRIZ CONSTITUTIVA ------- %%%%%%%%%%
MCF= E*(Esp^3)/(12*(1.-vu*vu))* ...
    [1 vu 0. ;...
     vu 1 0. ;...
     0. 0. (1.-vu)/2] ;
MCS= E/(2*(1.+vu))* ...
    [Esp 0 ;...
     0 Esp];

% %%%%%%%%%% ------- OBTENCIÓN DE PUNTOS DE GAUSS Y PESOS ------- %%%%%%%%%%
[pointb,weightb]=NumPInteg2D(nglxb,nglyb);  % Obteniendo puntos de int. y pesos
(flexión)
[points,weights]=NumPInteg2D(nglxs,nglys);  % Obteniendo puntos de int. y pesos (corte)

%%%%%%%%%% ------- OBTENCIÓN DE RIGIDEZ EN EL SISTEMA ------- %%%%%%%%%%

%   Creación de las matrices bases
beeb=zeros(3,elgdl);            % Matriz de deformación a flexión
bees=zeros(2,elgdl);            % Matriz de deformación a corte

for iel=1:nel               % Bucle para el número total de elementos
```

```
keb=zeros(elgdl,elgdl) ; % Inicialización de la matriz de rigidez (flexión)
kes=zeros(elgdl,elgdl) ; % Inicialización de la matriz de rigidez (corte)

%   Obtención de los datos de conectividad del elemento
l=0;
for k=1: nne
    for j=1:ndgdl
        l=l+1;
        g(l)=nf(conect(iel,k),j);
    end
end

%   Obtención de los datos geométricos del elemento
coord=zeros(nne,2);
for k=1: nne
    for j=1:2
        coord(k,j)=geom(conect(iel,k),j);
    end
end

%%%--- MATRIZ DE RIGIDEZ ELEMENTOS A FLEXIÓN ---%%%
for intx=1:nglxb
    x=pointb(intx,1);       % Punto de Int. Eje X
    wtx=weightb(intx,1);    % Peso Eje X
    for inty=1:nglyb
        y=pointb(inty,2);       % Punto de Int. Eje Y
        wty=weightb(inty,2);     % Peso Eje Y
        [shape,dhdr,dhds]=Elem2D_4nodos(x,y); % Cálculo de funciones de forma
                                        % y sus derivadas
        der=[dhdr;dhds];
        jacob=der*coord;                    % Cálculo del Jacobiano
        detjacob=det(jacob);                % Determinante del Jacobiano
        invjacob=inv(jacob);                % Inversa del Jacobiano
        deriv=invjacob*der;                 % Funciones de forma en coor. globales
        beeb=matdeform2d_flex(deriv,nne,elgdl);    % Matriz de deformación
        keb=keb + detjacob*wtx*wty*beeb'*MCF*beeb; % Matriz de rigidez (Flexión)
    end
end

% Ensamblaje en la matriz de rigidez general
for i=1:elgdl
    if g(i) ~= 0
        for j=1: elgdl
            if g(j) ~= 0
                KG(g(i),g(j))= KG(g(i),g(j)) + keb(i,j);
            end
        end
    end
end

%%%--- MATRIZ DE RIGIDEZ ELEMENTOS A CORTE ---%%%
for intx=1:nglxs
    x=points(intx,1);       % Punto de Int. Eje X
    wtx=weights(intx,1);    % Peso Eje X
```

```
        for inty=1:nglys
            y=points(inty,2);       % Punto de Int. Eje Y
            wty=weights(inty,2);    % Peso Eje Y
            [shape,dhdr,dhds]=Elem2D_4nodos(x,y); % Cálculo de Funciones de Forma
                                                  % y sus derivadas
            der=[dhdr;dhds];
            jacob=der*coord;                    % Cálculo del Jacobiano
            detjacob=det(jacob);                % Determinante del Jacobiano
            invjacob=inv(jacob);                % Inversa del Jacobiano
            deriv=invjacob*der;                 % Funciones de forma en coor. globales
            bees=matdeform2d_cort(deriv,shape,nne,elgdl);  % Matriz de deformación
            kes=kes + (5/6)*detjacob*wtx*wty*bees'*MCS*bees; % Matriz de rigidez (Corte)
        end
    end

    % Ensamblaje en la matriz de rigidez general
    for i=1:elgdl
        if g(i) ~= 0
            for j=1: elgdl
                if g(j) ~= 0
                    KG(g(i),g(j))= KG(g(i),g(j)) + kes(i,j);
                end
            end
        end
    end

end

%%%%%%%%% ------- OBTENCIÓN DE FUERZAS EN EL SISTEMA ------- %%%%%%%%%

for i=1:nnd     % Bucle para cada uno de los nodos
    for j=1:ndgdl   % Bucle para cada uno de los GDLs
        if Numnd(i,j)~= 0
            F(Numnd(i,j)) = Carg(i,j);
        end
    end
end

%%%%%%% ------- APLICANDO CONDICIONES DE BORDE AL SISTEMA ------- %%%%%%%

n=length(ResGDL);
 for i=1:n
    c=ResGDL(i);
    for j=1:sigdl
       KG(c,j)=0;
    end

    KG(c,c)=1;
    F(c)=0;
 end

%%%%%%%%% ------- CÁLCULO DE DESPLAZAMIENTOS ------- %%%%%%%%%

delta = KG\F;       % Obtención de los desplazamientos y giros correspondientes
```

```
%%%%%%%%%% ------- DESPLAZAMIENTOS ORDENADOS ------- %%%%%%%%%%

for i=1:nnd
    for j=1:ndgdl
        node_disp(i,j) = 0;
        if nf(i,j)~= 0
            node_disp(i,j) = delta(nf(i,j)) ;
        end
    end
end
desp=node_disp;

%%%%%%%%%% -- CÁLCULO DE LOS MOMENTOS Y FUERZAS CORTANTES -- %%%%%%%%%%

%   Calculo de los momentos y fuerzas cortantes en el centro del elemento
nglx=1; ngly=1;     % Número de Ptos. de int. asignados
[point,weight]=NumPInteg2D(nglx,ngly);  % Obteniendo puntos de int. y pesos


for iel=1:nel
    %   Obtención de los datos de conectividad del elemento
    l=0;
    for k=1: nne
        for j=1:ndgdl
            l=l+1;
            g(l)=nf(conect(iel,k),j);
        end
    end

    %   Obtención de los datos geométricos del elemento
    coord=zeros(nne,2);
    for k=1: nne
        for j=1:2
            coord(k,j)=geom(conect(iel,k),j);
        end
    end

    eld=zeros(elgdl,1); % Inicialización del vector de desplazamientos en 0

    for m=1:elgdl       % Bucle para el vector de desplazamientos General
        if g(m)==0
            eld(m)=0.;
        else
            eld(m)=delta(g(m)); % Recuperación de los desplazamientos en el vector
general
        end
    end
    %   Bucle para la obtención de momento y cortante por elemento
    for intx=1: nglx
        x=point(intx,1);        % Punto de Int. Eje X
        wtx=weight(intx,1);     % Peso Eje X
        for inty=1: ngly
            y=point(inty,2);        % Punto de Int. Eje Y
```

```
            wty=weight(inty,2);     % Peso Eje Y
            [shape,dhdr,dhds]=Elem2D_4nodos(x,y); % Calculo de funciones de Forma
                                                  % y sus derivadas
            der=[dhdr;dhds];        % Separación de las deriv. Func Forma
            jacob=der*coord;        % Calculo del Jacobiano
            detjacob=det(jacob);    % Determinante del Jacobiano
            invjacob=inv(jacob);    % Inversa del Jacobiano

            deriv=invjacob*der;     % Deriv. Func forma en Coord Globales
            beeb=matdeform2d_flex(deriv,nne,elgdl);        % Matriz de deform. (flexión)
            chi_b = beeb*eld ;      % Deformación a flexión
            Moment = MCF*chi_b ;    % Momentos del elemento
            bees=matdeform2d_cort(deriv,shape,nne,elgdl);  % Matriz de deform (corte)
            chi_s = bees*eld ;      % Deformación a corte
            Shear = (1)*MCS*chi_s ; % Esfuerzo cortante del elemento
        end
    end
    Element_Forces(iel,:)=[Moment' Shear'];
end

%%%%%%%%%% -- CALCULO DE LOS MOMENTOS Y FUERZAS CORTANTES 2 -- %%%%%%%%%%

%   Cálculo de los momentos y fuerzas cortantes en el centro del elemento
nglx=2; ngly=2;         % Número de Ptos. de int. asignados
[point,weight]=NumPInteg2D(nglx,ngly);  % Obteniendo puntos de int. y pesos

p=1;
for iel=1:nel
    %   Obtención de los datos de conectividad del elemento
    l=0;
    for k=1: nne
        for j=1:ndgdl
            l=l+1;
            g(l)=nf(conect(iel,k),j);
        end
    end

    %   Obtención de los datos geométricos del elemento
    coord=zeros(nne,2);
    for k=1: nne
        for j=1:2
            coord(k,j)=geom(conect(iel,k),j);
        end
    end

    eld=zeros(elgdl,1);     % Inicialización del vector de desplazamientos en 0

    for m=1:elgdl       % Bucle para el vector de desplazamientos general
        if g(m)==0
            eld(m)=0.;
        else
            eld(m)=delta(g(m)); % Recuperación de los desplazamientos en el vector
general
        end
```

```
    end

    %   Bucle para la obtención de momento y cortante por elemento
    k=1;
    for intx=1: nglx
        wtx=weight(intx,1);          % Peso en el Pto. de Int Eje X
        for inty=1: ngly        % Bucle simplificado para cada Pto de Int.
            if intx==1 && inty==1       % Primero
                x=-0.5774;
                y=-0.5774;
            elseif intx==1 && inty==2   % Segundo
                x=0.5774;
                y=-0.5774;
            elseif intx==2 && inty==1   % Tercero
                x=0.5774;
                y=0.5774;
            elseif intx==2 && inty==2   % Cuarto
                x=-0.5774;
                y=0.5774;
            end

            wty=weight(inty,2);     % Peso en el Pto. de Int Eje Y
            [shape,dhdr,dhds]=Elem2D_4nodos(x,y); % Cálculo de funciones de forma
                                            % y sus derivadas

            der=[dhdr;dhds];        % Separación de las deriv. Func Forma
            jacob=der*coord;        % Cálculo del Jacobiano
            detjacob=det(jacob);    % Determinante del Jacobiano
            invjacob=inv(jacob);    % Inversa del Jacobiano

            deriv=invjacob*der;     % Deriv. Func forma en coord globales

            beeb=matdeform2d_flex(deriv,nne,elgdl);    % Matriz de deform. (flexión)
            chi_b = beeb*eld ;      % Deformación a flexión
            Moment = MCF*chi_b ;    % Momentos del elemento
            bees=matdeform2d_cort(deriv,shape,nne,elgdl);  % Matriz de deform. (corte)
            chi_s = bees*eld ;      % Deformación a corte
            Shear = (1)*MCS*chi_s ; % Esfuerzos cortantes del elemento

            Moment2(k,:)=Moment;
            Shear2(k,:)=Shear;
            k=k+1;
        end
    end
    Element_Corte(p:p+3,1:2)=[Shear2];   % Traslado de Cort. a una Matriz general
    Element_Flex(p:p+3,1:3)=[Moment2];   % Traslado de Moment. a una Matriz General
    p=4*iel+1;
end
```

Published with MATLAB® R2019b

CÓDIGO PARA ELEMENTOS PLATE – Método. Reissner Mindlin (Ingreso de Datos del Usuario)

```
%{
Propósito:
    Analizar un sistema compuesto de elementos cuadriláteros uniforme tipo
    Plate mediante el metodo de Reisssner Mindlin y en base a 4 nodos por elemento.
    Código para el ingreso de datos por el usuario.
Descripción de Datos de Ingreso:
    - Coordenadas de los nodos en el elemento Plate.
    - Cargas Impuestas en los diversos nodos.
    - Condiciones de borde en el sistema.
Variables de Ingreso:
    Especificadas en el código.
Diseño: Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Junio 2019
%}

%%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%%

clear           % Limpieza del espacio de trabajo
clc             % Limpieza de la ventana de comandos
close all       % Cierre de todas las figuras existentes
format short e  % Formato numérico a trabajar

%%%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%%%

%   Ingreso de coordenadas de los nodos. (Geométria)
geom = [0.0  0.0;...    % Coordenadas en X,Y del Elem. Plate.
        100  0.0;...
        200  0.0;...
        0.0  100;...
        100  100;...
        200  100;...
        0.0  200;...
        100  200;...
        200  200];

%   Ingreso de datos de conectividad entre nodos. (Conectividad)
conect = [1 2 5 4;   % Conectividad de nodos en el extremo del elemento
          2 3 6 5;
          4 5 8 7;
          5 6 9 8];

%   Propiedades del material (Elasticidad/Poisson/Espesor)
E=200000;       % Módulo de elasticidad del material
vu=0.3;         % Coeficiente de Poisson del material
Esp=10;         % Espesor del elemento plate

%   Información sobre el sistema (Automático)
nne = 4;        % Numero de nodos por elemento
ndgdl = 3;      % Numero de GDLs por nodo
```

```
[nnd,~] = size(geom);       % Número de nodos en el sistema
[nel,~] = size(conect);     % Número de elementos
elgdl = nne*ndgdl;          % Número de GDLs por elemento
sigdl = nnd*ndgdl;          % Número de GDLs en el sistema
nglxb=2; nglyb=2;           % Número de puntos de Gauss a flexión
nglxs=1; nglys=1;           % Número de puntos de Gauss a corte

%   Dando numeración a todos los GDLs en el sistema
Numnd = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para asignar numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

%   Condiciones de frontera por nodo
nf = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para asignar numeración
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end

% Bucle para asignar automáticamente las restricciones
Ancho = max(max(geom(:,1)));           %   Ancho en dirección X
Largo = max(max(geom(:,2)));           %   Largo en dirección Y

for i=1:nnd
    if geom(i,1) == 0
        nf(i,1) = 0. ;
        nf(i,2) = 0. ;
        nf(i,3) = 0. ;
    elseif geom(i,2) == 0
        nf(i,1) = 0. ;
        nf(i,2) = 0. ;
        nf(i,3) = 0. ;
    elseif geom(i,2) == Largo && geom(i,1) == Ancho
        nf(i,1) = 0. ;
        nf(i,2) = 0. ;
    elseif geom(i,2) == Largo
        nf(i,2) = 0. ;
    elseif geom(i,1) == Ancho
        nf(i,1) = 0. ;
    end
end
```

```
%   Obteniendo los GDLs restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle para extraer los GDLs restringidos
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end

%      Cargas/Fuerzas aplicadas al sistema
Carg = zeros(nnd, ndgdl);       % Creación de la matriz de cargas vacías (0)
Carg(9,3) = -20/4;              % Aplicando cargas a los nodos

%   Ejecutando la función para el procesamiento de datos
disp('Procesando Datos');
[desp,Element_Forces,Element_Flex,Element_Corte] =
Placa_ReissnerMindlin_Programacion(nglxb,nglyb,nglxs,nglys,nf,nne,nnd,nel,conect,geom,E,
vu,Esp,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL);

%%%%%%%%%%%%%%%%% ------- MOSTRANDO GRÁFICOS ------- %%%%%%%%%%%%%%%%

% Desplazamientos
UX=desp(:,1);
UY=desp(:,2);
UZ=desp(:,3);

figure
cmin = min(UZ);
cmax = max(UZ);
caxis([cmin cmax]);
patch('Faces',conect, 'Vertices', geom, 'FaceVertexCData',UZ,...
'Facecolor','interp','Marker','.');
colorbar;
title('Desplazamiento en Z (UZ)')
xlabel('B')
ylabel('H')

%%%%%%%%%% ------- ORDENANDO ESFUERZOS ------- %%%%%%%%%%
% 1 Punto de integración
for k = 1:nnd
    mx = 0. ; my = 0.; mxy = 0.; qx = 0.; qy = 0.;
    ne = 0;
    for iel = 1:nel
        for jel=1:nne
            if conect(iel,jel) == k
                ne=ne+1;
                mx = mx + Element_Forces(iel,1);
                my = my + Element_Forces(iel,2);
                mxy = mxy + Element_Forces(iel,3);
                qx = qx + Element_Forces(iel,4);
                qy = qy + Element_Forces(iel,5);
            end
```

```
        end
    end
    MX(k,1) = mx/ne;
    MY(k,1) = my/ne;
    MXY(k,1) = mxy/ne;
    QX(k,1) = qx/ne;
    QY(k,1) = qy/ne;
end

% 4 Puntos de integración
Element_Bending=Element_Flex;
Element_Shear=Element_Corte;
for k = 1:nnd
    mx = 0. ; my = 0.; mxy = 0.; qx = 0.; qy = 0.;
    ne = 0; p=0;
    for iel = 1:nel
        for jel=1:nne
            if conect(iel,jel) == k
                ne=ne+1;
                p=iel*4-4+jel;
                mx = mx + Element_Bending(p,1);
                my = my + Element_Bending(p,2);
                mxy = mxy + Element_Bending(p,3);
                qx = qx + Element_Shear(p,1);
                qy = qy + Element_Shear(p,2);
            end
        end
    end
    MX2(k,1) = mx/ne;
    MY2(k,1) = my/ne;
    MXY2(k,1) = mxy/ne;
    QX2(k,1) = qx/ne;
    QY2(k,1) = qy/ne;
end

figure
cmin = min(MX);
cmax = max(MX);
caxis([cmin cmax]);
patch('Faces',conect, 'Vertices', geom, 'FaceVertexCData',MX,...
'Facecolor','interp','Marker','.');
colorbar;
title('Momento en X (MX) - 1 Pto de Integracion')
xlabel('B')
ylabel('H')

figure
cmin = min(MY);
cmax = max(MY);
caxis([cmin cmax]);
patch('Faces',conect, 'Vertices', geom, 'FaceVertexCData',MY,...
'Facecolor','interp','Marker','.');
colorbar;
title('Momento en Y (MY) - 1 Pto de Integracion')
```

```
xlabel('B')
ylabel('H')

% Momentos 4 puntos de integración
Element_Forces=zeros(16,5);
Element_Forces(:,1:3)=Element_Flex;
Element_Forces(:,4:5)=Element_Corte;


%   Obtención de la gráfica
figure
hold on
cmin = min(MX2);
cmax = max(MX2);
caxis([cmin cmax]);
patch('Faces',conect, 'Vertices', geom, 'FaceVertexCData',MX2,...
'Facecolor','interp','Marker','.');
colorbar;
title('Momento en X (MX) - 4 Ptos de Integracion')

disp('Procesamiento de Datos Terminado');

%   Limpieza de variables innecesarias
clear c n i j a b cmax cmin DR x y xcoord ycoord zcoord vu C coord E eld delta...
    F g k k1 k2 k3 k4 kg KG MCF MD m l intx iel nel nf nnd nne ndgdl node_disp...
    npgf Q ResGDL sigdl Numnd Moment geom elgdl Element_Forces Esp
```

Published with MATLAB® R2019b

CÓDIGO PARA ELEMENTOS BIDIMENSIONALES (Gauss) – Tensión Plana (Parte Programada del Código)

```
%{
Propósito:
    Analizar un sistema compuesto de elementos bidimensionales bajo tensión
    Plana. Elementos con 8 nodos, bajo integración numérica por Gauss-Legendre.
    Código para el procesamiento de datos internos.
Descripción de datos de ingreso:
    - Ninguno, se utilizaran datos del programa Bi_TenPlana_Gauss_Usuario
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Noviembre 2019
%}

function[delta,SIGMA] =
Bi_TenPlana_Programacion(nf,ngp,nnd,nel,conect,geom,E,vu,thick,elgdl,CargNod,sigdl,Numnd
,ndgdl,ResGDL,nne)

%   Creación de matrices y vectores vacíos
KG = zeros(sigdl);      % Rigidez del sistema
F  = zeros(sigdl,1);    % Fuerzas en el sistema
```

```
%%%%%%%%%% ------- OBTENCIÓN DE LA MATRIZ CONSTITUTIVA ------- %%%%%%%%%%
c=E/(1.-vu*vu);
MC=c*[1 vu 0. ;...
      vu 1 0. ;...
      0. 0. .5*(1.-vu)];

%%%%%%%%%% ------- OBTENCIÓN DE PUNTOS DE GAUSS Y PESOS ------- %%%%%%%%%%
samp=zeros(ngp,2);
%
if ngp==1
    samp=[0. 2];
    elseif ngp==2
        samp=[-1./sqrt(3) 1.;...
               1./sqrt(3) 1.];
    elseif ngp==3
        samp= [-.2*sqrt(15.) 5./9; ...
                0. 8./9.;...
                .2*sqrt(15.) 5./9];
    elseif ngp==4
        samp= [-0.861136311594053 0.347854845137454; ...
                -0.339981043584856 0.652145154862546; ...
                0.339981043584856 0.652145154862546; ...
                0.861136311594053 0.347854845137454];
end

%%%%%%%%%% ------- OBTENCIÓN DE RIGIDEZ EN EL SISTEMA ------- %%%%%%%%%%

for i=1:nel     % Bucle para cada uno de los elementos
    l=0;

    % Obtención de las coordenadas y conectividad por elemento
    coord=zeros(nne,ndgdl);
    for k=1: nne
        for j=1:ndgdl
            coord(k,j)=geom(conect(i,k),j);
            l=l+1;
            g(l)=Numnd(conect(i,k),j);
        end
    end

    % Cálculo de la matriz de rigidez por elemento
    ke=zeros(elgdl,elgdl);   % Inicialización de la matriz de rigidez por elemento
    for ig=1: ngp
        wi = samp(ig,2);
        for jg=1: ngp
            wj=samp(jg,2);
            [der,fun] = Elem2D_8nodos(samp, ig,jg); % Funciones de forma y sus derivadas
            jac=der*coord;      % Cálculo del Jacobiano
            d=det(jac);         % Determinante del Jacobiano
            jac1=inv(jac);      % Inversa del Jacobiano
            deriv=jac1*der;     % Deriv. Func forma en Coord. Globales
            bee=matdeform2d(deriv,nne,elgdl);   % Matriz de deformación
            ke=ke + d*thick*wi*wj*bee'*MC*bee; % Rigidez por elemento
        end
```

```
    end

    % Ensamblaje de la matriz de rigidez
    for q=1:elgdl
        if g(q) ~= 0
            for j=1: elgdl
                if g(j) ~= 0
                    KG(g(q),g(j))= KG(g(q),g(j)) + ke(q,j);
                end
            end
        end
    end
end

%%%%%%%%% ------- OBTENCIÓN FUERZAS EN EL SISTEMA ------- %%%%%%%%%

for i=1: nnd    % Bucle para obtener la carga directamente aplicada al nodo
    if Numnd(i,1) ~= 0
        F(Numnd(i,1))= CargNod(i,1);
    end
    if Numnd(i,2) ~= 0
        F(Numnd(i,2))= CargNod(i,2);
    end
end


%%%%%% ------- APLICANDO CONDICIONES DE BORDE AL SISTEMA ------- %%%%%%

n=length(ResGDL);
sdof=size(KG);
 for i=1:n
    c=ResGDL(i);
    for j=1:sdof
       KG(c,j)=0;
    end
    KG(c,c)=1;
    F(c)=0;
 end

%%%%%%%%% ------- CALCULO DE DESPLAZAMIENTOS ------- %%%%%%%%%

delta = KG\F;   % Desplazamientos de todo el sistema

% %%%%%%%%% ------- CALCULO DE ESFUERZOS INTERNAS ------- %%%%%%%%%

ngp=1;      % Numero de Puntos de int. asignados
samp=[0. 2];

for i=1:nel

    % Creando la matriz de coordenadas por elemento
    coord=zeros(nne,ndgdl);
    l=0;    % Bucle para la obtención de conectividad en GDLs
    for k=1:nne
```

```
        for j=1:ndgdl
            l=l+1;
            g2(l)=nf(conect(i,k),j);
        end
    end
    l=0;
    for k=1: nne
        for j=1:ndgdl
            coord(k,j)=geom(conect(i,k),j);
            l=l+1;
            g(l)=Numnd(conect(i,k),j);
        end
    end

    eld=zeros(elgdl,1);         % Iniciando el vector de desplazamientos por elemento
    for m=1:elgdl
        if g2(m)==0
            eld(m)=0.;
        else
            eld(m)=delta(g(m)); % Obtención de los desplazamientos por Elemento
        end
    end

    % Bucle para la obtención de los esfuerzos por elemento
    for ig=1: ngp
        wi = samp(ig,2);
        for jg=1: ngp
            wj=samp(jg,2);
            [der,fun] = Elem2D_8nodos(samp, ig,jg); % Funciones de forma y sus derivadas
            jac=der*coord;      % Cálculo del Jacobiano
            jac1=inv(jac);      % Inversa del Jacobiano
            deriv=jac1*der;     % Deriv. Func forma en coord globales
            bee=matdeform2d(deriv,nne,elgdl);   % Matriz de deformación
            eps=bee*eld;        % Deformaciones del elemento
            sigma=MC*eps;       % Esfuerzos del elemento
        end
    end
SIGMA(i,:)=sigma;   % Matriz de esfuerzos generales del sistema
end
```

Published with MATLAB® R2019b

CÓDIGO PARA ELEMENTOS BIDIMENSIONALES (Gauss) – Tensión Plana (Ingreso de Datos del Usuario)

```
%{
Propósito:
    Analizar un sistema compuesto de elementos bidimensionales bajo tensión
    plana. Elementos con 8 nodos, bajo integración numérica por Gauss-Legendre.
    Código para el ingreso de datos por el usuario.
Descripción de datos de ingreso:
    - Coordenadas.
```

```
    - Cargas impuestas en los diversos nodos.
    - Restricciones en el sistema.
Variables de ingreso:
    Especificadas en el código.
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Noviembre 2019
%}

%%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%%

clear           % Limpieza del espacio de trabajo
clc             % Limpieza de la ventana de comandos
close all       % Cierre de todas las figuras existentes
format short e  % Formato numérico a trabajar

%%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%%

%   Ingreso de coordenadas de los nodos. (Geométria)
geom = [ 0 0; ...    % Coordenadas en X,Y del elem. bidimensional
        0 50 ; ...
        0 100 ; ...
        0 150 ; ...
        0 200 ; ...
        0 250 ; ...
        0 300 ; ...
        0 350 ; ...
        0 400 ; ...
        50 0 ; ...
        50 100 ; ...
        50 200 ; ...
        50 300 ; ...
        50 400 ; ...
        100 0 ; ...
        100 50 ; ...
        100 100 ; ...
        100 150 ; ...
        100 200 ; ...
        100 250 ; ...
        100 300 ; ...
        100 350 ; ...
        100 400 ; ...
        150 0 ; ...
        150 100 ; ...
        150 200 ; ...
        150 300 ; ...
        150 400 ; ...
        200 0 ; ...
        200 50 ; ...
        200 100 ; ...
        200 150 ; ...
        200 200 ; ...
        200 250 ; ...
        200 300 ; ...
```

```
200 350 ; ...
200 400 ; ...
250 0 ; ...
250 100 ; ...
250 200 ; ...
250 300 ; ...
250 400 ; ...
300 0 ; ...
300 50 ; ...
300 100 ; ...
300 150 ; ...
300 200 ; ...
300 250 ; ...
300 300 ; ...
300 350 ; ...
300 400 ; ...
350 0 ; ...
350 100 ; ...
350 200 ; ...
350 300 ; ...
350 400 ; ...
400 0 ; ...
400 50 ; ...
400 100 ; ...
400 150 ; ...
400 200 ; ...
400 250 ; ...
400 300 ; ...
400 350 ; ...
400 400 ; ...
450 0 ; ...
450 100 ; ...
450 200 ; ...
450 300 ; ...
450 400 ; ...
500 0 ; ...
500 50 ; ...
500 100 ; ...
500 150 ; ...
500 200 ; ...
500 250 ; ...
500 300 ; ...
500 350 ; ...
500 400 ; ...
550 0 ; ...
550 100 ; ...
550 200 ; ...
550 300 ; ...
550 400 ; ...
600 0 ; ...
600 50 ; ...
600 100 ; ...
600 150 ; ...
600 200 ; ...
```

```
        600 250 ; ...
        600 300 ; ...
        600 350 ; ...
        600 400 ; ...
        650 0 ; ...
        650 100 ; ...
        650 200 ; ...
        650 300 ; ...
        650 400 ; ...
        700 0 ; ...
        700 50 ; ...
        700 100 ; ...
        700 150 ; ...
        700 200 ; ...
        700 250 ; ...
        700 300 ; ...
        700 350 ; ...
        700 400 ; ...
        750 0 ; ...
        750 100 ; ...
        750 200 ; ...
        750 300 ; ...
        750 400 ; ...
        800 0 ; ...
        800 50 ; ...
        800 100 ; ...
        800 150 ; ...
        800 200 ; ...
        800 250 ; ...
        800 300 ; ...
        800 350 ; ...
        800 400] ;

%   Ingreso de datos de conectividad entre nodos. (Conectividad)
conect = [ 1 10 15 16 17 11 3 2 ; ... % Conectividad de nodos en el extremo del elemento
           3 11 17 18 19 12 5 4 ; ...
           5 12 19 20 21 13 7 6 ; ...
           7 13 21 22 23 14 9 8 ; ...
           15 24 29 30 31 25 17 16 ; ...
           17 25 31 32 33 26 19 18 ; ...
           19 26 33 34 35 27 21 20 ; ...
           21 27 35 36 37 28 23 22 ; ...
           29 38 43 44 45 39 31 30 ; ...
           31 39 45 46 47 40 33 32 ; ...
           33 40 47 48 49 41 35 34 ; ...
           35 41 49 50 51 42 37 36 ; ...
           43 52 57 58 59 53 45 44 ; ...
           45 53 59 60 61 54 47 46 ; ...
           47 54 61 62 63 55 49 48 ; ...
           49 55 63 64 65 56 51 50 ; ...
           57 66 71 72 73 67 59 58 ; ...
           59 67 73 74 75 68 61 60 ; ...
           61 68 75 76 77 69 63 62 ; ...
           63 69 77 78 79 70 65 64 ; ...
```

```
            71 80 85 86 87 81 73 72 ; ...
            73 81 87 88 89 82 75 74 ; ...
            75 82 89 90 91 83 77 76 ; ...
            77 83 91 92 93 84 79 78 ; ...
            85 94 99 100 101 95 87 86 ; ...
            87 95 101 102 103 96 89 88 ; ...
            89 96 103 104 105 97 91 90 ; ...
            91 97 105 106 107 98 93 92 ; ...
            99 108 113 114 115 109 101 100 ; ...
            101 109 115 116 117 110 103 102 ; ...
            103 110 117 118 119 111 105 104 ; ...
            105 111 119 120 121 112 107 106 ];

%   Propiedades del material.
E = 40000.;        % Módulo de elasticidad del sistema
vu = 0.17;           % Coeficiente de Poisson
thick = 100;         % Espesor del sistema

%   Información sobre el sistema (Automático)
nne = 8;                    % Número de nodos por elemento
ndgdl = 2;                  % Número de GDLs por nodo
ngp = 2;                    % Número de puntos de Gauss
[nnd] = size(geom,1);       % Número de nodos en el sistema
[nel] = size(conect,1);     % Número de elementos
elgdl = nne*ndgdl;          % Número de GDLs por elemento
sigdl = nnd*ndgdl;          % Número de GDLs en el sistema

%   Dando numeración a todos los GDLs en el sistema
Numnd = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

%   Condiciones de frontera por nodo
nf = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end

for i=1:nnd     % Aplicando restricciones
    if geom(i,1) == 800.
        nf(i,1) = 0;
```

```
    end
    if geom(i,1) == 100. && geom(i,2) == 0.
        nf(i,2) = 0;
    end
end


%   Obteniendo los GDLs restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end

%      Fuerzas aplicadas al sistema
CargNod = zeros(nnd, 2);    % Carga aplicada directamente
CargNod(79,2)=-30000.;

%   Limpieza de variables innecesarias
clear c n i j

%   Ejecutando la función para el procesamiento de datos
disp('Procesando Datos');
[delta,SIGMA] =
Bi_TenPlana_Gauss_Programacion(nf,ngp,nnd,nel,conect,geom,E,vu,thick,elgdl,CargNod,sigdl
,Numnd,ndgdl,ResGDL,nne);

%%%%%%%%% ------- DESPLAZAMIENTOS ORDENADOS ------- %%%%%%%%%

for i=1:nnd
    for j=1:ndgdl
        node_disp(i,j) = 0;
        if nf(i,j)~= 0
            node_disp(i,j) = delta(Numnd(i,j)) ;
        end
    end
end
desp=node_disp;

%%%%%%%%% ------- ESFUERZOS ORDENADOS ------- %%%%%%%%%

for k = 1:nnd
    sigx = 0. ;sigy = 0.; tau = 0.;
    ne = 0;
    for iel = 1:nel
        for jel=1:nne
            if conect(iel,jel) == k
                ne=ne+1;
                sigx = sigx+SIGMA(iel,1);
                sigy = sigy + SIGMA(iel,2);
```

```
                tau = tau + SIGMA(iel,3);
            end
        end
    end
    ZX(k,1) = sigx/ne;
    ZY(k,1) = sigy/ne;
    ZT(k,1)=tau/ne;
    Z1(k,1)= ((sigx+sigy)/2 + sqrt(((sigx+sigy)/2)^2 +tau^2))/ne;
    Z2(k,1)= ((sigx+sigy)/2 - sqrt(((sigx+sigy)/2)^2 +tau^2))/ne;
end

%%%%%%%%%%%%%%%%%% ------- MOSTRANDO GRÁFICOS ------- %%%%%%%%%%%%%%%%%%

% Desplazamientos
U2 = desp(:,2);
figure
cmin = min(U2);
cmax = max(U2);
caxis([cmin cmax])
patch('Faces', conect, 'Vertices', geom, 'FaceVertexCData',U2,...
'Facecolor','interp','Marker','.');
colorbar

% Esfuerzos
figure
cmin = min(Z1);
cmax = max(Z1);
caxis([cmin cmax]);
patch('Faces', conect, 'Vertices', geom, 'FaceVertexCData',Z1,...
'Facecolor','interp','Marker','.');
colorbar;
title('Shear stress Z1 (MPa)')

figure
cmin = min(ZX);
cmax = max(ZX);
caxis([cmin cmax]);
patch('Faces', conect, 'Vertices', geom, 'FaceVertexCData',ZX,...
'Facecolor','interp','Marker','.');
colorbar;
title('Shear stress ZX (MPa)')

disp('Procesamiento de datos terminado');

%   Limpieza de variables innecesarias
clear c n i j a b cmax cmin DR x y xcoord ycoord zcoord vu C coord E eld delta...
    F g k k1 k2 k3 k4 kg KG MCF MD m l intx iel nel nf nnd nne ndgdl node_disp...
    npgf Q ResGDL sigdl Numnd Moment geom elgdl Element_Forces Esp
```

Published with MATLAB® R2019b

CÓDIGO PARA ELEMENTOS TRIDIMENSIONALES
(Parte Programada del Código)

```
%{
Propósito:
    Analizar un sistema compuesto de elementos tridimensional con el FEM.
    CÓdigo para el procesamiento de datos interno
Descripción de datos de ingreso:
    - Ninguno, se utilizaran datos del código Tri_Elemento_Usuario
Diseño: Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Noviembre 2019
%}

function[delta,SIGMA,stress,conectesf] =
Tri_Elemento_Programacion(nf,nglx,ngly,nglz,nnd,nel,conect,geom,E,vu,elgdl,CargNod,sigdl
,Numnd,ndgdl,ResGDL,nne)

%   Creación de matrices y vectores vacíos
KG = zeros(sigdl);      % Rigidez del sistema  (Rigidez General)
F  = zeros(sigdl,1);    % Fuerzas en el sistema
xcoord=zeros(1,nne);    % Vector de coordenadas x por elemento(Sistemática)
ycoord=zeros(1,nne);    % Vector de coordenadas y por elemento(Sistemática)
zcoord=zeros(1,nne);    % Vector de coordenadas z por elemento(Sistemática)

%%%%%%%%%% ------- OBTENCIÓN DE LA MATRIZ CONSTITUTIVA ------- %%%%%%%%%%
MC= E/((1+vu)*(1-2*vu))* ...
    [(1-vu)  vu     vu      0         0       0;
    vu     (1-vu)   vu      0         0       0;
    vu       vu   (1-vu)    0         0       0;
    0        0      0    (1-2*vu)/2   0       0;
    0        0      0       0    (1-2*vu)/2   0;
    0        0      0       0         0   (1-2*vu)/2];

%%%%%%%%%% ------- OBTENCIÓN DE PUNTOS DE GAUSS Y PESOS ------- %%%%%%%%%%
[point3,weight3]=NumPInteg3D(nglx,ngly,nglz);

%%%%%%%%%% ------- OBTENCIÓN DE RIGIDEZ EN EL SISTEMA ------- %%%%%%%%%%

for iel=1:nel               % Bucle para el número total de elementos
    iel                     % Para poder ver el número de elemento analizado en pantalla
    k=zeros(elgdl,elgdl);   % Inicialización de la matriz de rigidez por elemento

    for i=1:nne    % Extracción de las coordenadas en función de X Y Z
        nd(i)=conect(iel,i);        % Conectividad entre nodos
        xcoord(i)=geom(nd(i),1);    % Coordenadas en X del elemento
        ycoord(i)=geom(nd(i),2);    % Coordenadas en Y del elemento
        zcoord(i)=geom(nd(i),3);    % Coordenadas en Z del elemento
    end

    for intx=1:nglx
        x=point3(intx,1);               % Punto de Int. Eje X
        wtx=weight3(intx,1);            % Peso Eje X
```

```
        for inty=1:ngly
            y=point3(inty,2);           % Punto de Int. Eje Y
            wty=weight3(inty,2);        % Peso Eje Y
            for intz=1:nglz
                z=point3(intz,3);       % Punto de Int. Eje Z
                wtz=weight3(intz,3);    % Peso Eje Z

                [shape,dhdr,dhds,dhdt]=Elem3D_8nodos(x,y,z);      % Func. forma y
derivadas
                jacob3=jacob3d(nne,dhdr,dhds,dhdt,xcoord,ycoord,zcoord);  % Jacobiano
                detjacob=det(jacob3);                % Determinante del Jacobiano
                invjacob=inv(jacob3);                % Inversa del Jacobiano

                [dhdx,dhdy,dhdz]=Deriv3DGlobal(nne,dhdr,dhds,dhdt,invjacob);    %
Derivadas en coord globales
                kinmtx=matdeform3d(nne,dhdx,dhdy,dhdz);            % Matriz de
deformación
                k=k+kinmtx'*MC*kinmtx*wtx*wty*wtz*detjacob;     % Matriz de rigidez
            end
        end
    end

    l=0;    % Bucle para la obtención de conectividad en GDLs
    for ka=1:nne
        for j=1:ndgdl
            l=l+1;
            index(l)=Numnd(conect(iel,ka),j);
        end
    end

    % Ensamblaje de la rigidez por elemento a una global
     for q=1:elgdl
        if index(q) ~= 0
            for j=1: elgdl
                if index(j) ~= 0
                    KG(index(q),index(j))= KG(index(q),index(j)) + k(q,j);
                end
            end
        end
     end
end

%%%%%%%%% ------- OBTENCIÓN FUERZAS EN EL SISTEMA ------- %%%%%%%%%

for i=1: nnd    % Bucle para obtener la carga directamente aplicada al nodo
    if Numnd(i,1) ~= 0
        F(Numnd(i,1))= CargNod(i,1);
    end
    if Numnd(i,2) ~= 0
        F(Numnd(i,2))= CargNod(i,2);
    end
    if Numnd(i,3) ~= 0
        F(Numnd(i,3))= CargNod(i,3);
    end
```

```
end

%%%%%%% ------- APLICANDO CONDICIONES DE BORDE AL SISTEMA ------- %%%%%%%

n=length(ResGDL);
sdof=size(KG);

 for i=1:n
    c=ResGDL(i);
    for j=1:sdof
       KG(c,j)=0;
    end
    KG(c,c)=1;
    F(c)=0;
 end

%%%%%%%%% ------- CÁLCULO DE DESPLAZAMIENTOS ------- %%%%%%%%%

delta = KG\F;

%%%%%%%%% ------- OBTENCIÓN FUERZAS EN EL SISTEMA CENTRO DEL ELEMENTO -------
%%%%%%%%%

nglx=1; ngly=1; nglz=1;     % Número de puntos de integración asignados
[point3,weight3]=NumPInteg3D(nglx,ngly,nglz);  % Pesos y Coord. de los puntos de
integración

% Modificando el orden en la conectividad
[a]=size(conect,1);
conectesf=zeros(a,8);
conectesf(:,1)=conect(:,5);
conectesf(:,2)=conect(:,1);
conectesf(:,3)=conect(:,4);
conectesf(:,4)=conect(:,8);
conectesf(:,5)=conect(:,6);
conectesf(:,6)=conect(:,2);
conectesf(:,7)=conect(:,3);
conectesf(:,8)=conect(:,7);

%   Bucle para la obtención de los esfuerzos para cada elemento
for iel=1:nel
    iel                     % Para poder ver el número de elemento analizado en pantalla
    k=zeros(elgdl,elgdl);   % Inicialización de la matriz de rigidez por elemento

    for i=1:nne    % Extracción de las coordenadas en función de X Y Z
        nd(i)=conect(iel,i);        % Conectividad entre nodos
        xcoord(i)=geom(nd(i),1);    % Coordenadas en X del elemento
        ycoord(i)=geom(nd(i),2);    % Coordenadas en Y del elemento
        zcoord(i)=geom(nd(i),3);    % Coordenadas en Z del elemento
    end

    l=0;    % Bucle para la obtención de conectividad en GDLs
    for ka=1:nne
```

```
        for j=1:ndgdl
            l=l+1;
            index(l)=Numnd(conect(iel,ka),j);
        end
    end

    l=0;    % Bucle para la obtención de conectividad en GDLs
    for ka=1:nne
        for j=1:ndgdl
            l=l+1;
            index2(l)=nf(conect(iel,ka),j);
        end
    end

    eld=zeros(elgdl,1); % Bucle para la obtención de los desplazamientos del elemento
    for m=1:elgdl
        if index2(m)==0
            eld(m)=0.;
        else
            eld(m)=delta(index(m));
        end
    end

    %  Bucle para obtener los esfuerzos para cada elemento
    for intx=1: nglx
        x=point3(intx,1);               % Punto de Integración, Eje X
        wtx=weight3(intx,1);            % Peso Eje X
        for inty=1: ngly
            y=point3(inty,2);           % Punto de Integración, Eje Y
            wty=weight3(inty,2) ;       % Peso Eje Y
            for intz=1: nglz
                z=point3(intz,3);       % Punto de Integración, Eje Z
                wtz=weight3(intz,3) ;   % Peso Eje Z
                [shape,dhdr,dhds,dhdt]=Elem3D_8nodos(x,y,z);  % Func. forma y derivadas
                jacob3=jacob3d(nne,dhdr,dhds,dhdt,xcoord,ycoord,zcoord);  % Jacobiano
                detjacob=det(jacob3);                       % Determinante del Jacobiano
                invjacob=inv(jacob3);                       % Inversa del Jacobiano
                [dhdx,dhdy,dhdz]=Deriv3DGlobal(nne,dhdr,dhds,dhdt,invjacob); % Derivada
en coord globales
                kinmtx=matdeform3d(nne,dhdx,dhdy,dhdz);     % Matriz de deformación del
elemento
                eps=kinmtx*eld;     % Deformación del elemento
                sigma=MC*eps;       % Esfuerzo del elemento
            end
        end
    end
SIGMA(iel,:)=sigma;     % Traslado de esfuerzos a una Matriz General
end

%%%%%%%%% ------- OBTENCIÓN FUERZAS EN EL SISTEMA EXTREMOS (2 PUNTOS) DEL ELEMENTO ----
--- %%%%%%%%%

nglx=2; ngly=2; nglz=2;     % Número de puntos de integración asignados
[point3,weight3]=NumPInteg3D(nglx,ngly,nglz);   % Pesos y Coord de los Ptos. de
```

```
integración
intp=0;
for iel=1:nel
    iel                     % Para poder ver el bucle de elementos en pantalla al correr
    k=zeros(elgdl,elgdl);   % Inicialización de la matriz de rigidez por elemento
    for i=1:nne    % Extracción de las coordenadas en función de X Y Z
        nd(i)=conect(iel,i);        % Conectividad entre NODOS
        xcoord(i)=geom(nd(i),1);    % Coordenadas en X del elemento
        ycoord(i)=geom(nd(i),2);    % Coordenadas en Y del elemento
        zcoord(i)=geom(nd(i),3);    % Coordenadas en Z del elemento
    end

    l=0;    % Bucle para la obtención de conectividad en GDLs
    for ka=1:nne
        for j=1:ndgdl
            l=l+1;
            index(l)=Numnd(conect(iel,ka),j);
        end
    end

    l=0;    % Bucle para la obtención de conectividad en GDLs
    for ka=1:nne
        for j=1:ndgdl
            l=l+1;
            index2(l)=nf(conect(iel,ka),j);
        end
    end

    eld=zeros(elgdl,1); % Bucle para la obtención de los desplazamientos del elemento
    for m=1:elgdl
        if index2(m)==0
            eld(m)=0.;
        else
            eld(m)=delta(index(m));
        end
    end

    for intx=1: nglx
        x=point3(intx,1);                   % Punto de Integración, Eje X
        wtx=weight3(intx,1);                % Peso Eje X
        for inty=1: ngly
            y=point3(inty,2);               % Punto de Integración, Eje Y
            wty=weight3(inty,2) ;           % Peso Eje Y
            for intz=1: nglz
                z=point3(intz,3);           % Punto de Integración, Eje Z
                wtz=weight3(intz,3) ;       % Peso Eje Z
                intp=intp+1;
                [shape,dhdr,dhds,dhdt]=Elem3D_8nodos(x,y,z);  % Func. forma y derivadas
                jacob3=jacob3d(nne,dhdr,dhds,dhdt,xcoord,ycoord,zcoord);  % Jacobiano
                detjacob=det(jacob3);                   % Determinante del Jacobiano
                invjacob=inv(jacob3);                   % Inversa del Jacobiano
                [dhdx,dhdy,dhdz]=Deriv3DGlobal(nne,dhdr,dhds,dhdt,invjacob); % Derivada
en coord globales
                kinmtx=matdeform3d(nne,dhdx,dhdy,dhdz);    % Matriz de deformación
```

```
                eps=kinmtx*eld;         % Deformación del elemento
                sigma=MC*eps;           % Esfuerzos del elemento
                for i=1:6
                    stress(intp,i)=sigma(i);    % Traslado de esfuerzos a una matriz
general
                end
            end
        end
    end
end
```

Published with MATLAB® R2019b

CÓDIGO PARA ELEMENTOS TRIDIMENSIONALES
(Ingreso de Datos del Usuario)

```
%{
Propósito:
    Analizar un sistema compuesto de elementos tridimensional con el FEM.
    Código para el ingreso de datos por el usuario.
Descripción de datos de ingreso:
    - Coordenadas.
    - Cargas impuestas en los diversos nodos.
    - Restricciones en el sistema.
Variables de ingreso:
    Especificadas en el código.
Diseño: Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Junio 2019
%}

%%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%%

clear           % Limpieza del espacio de trabajo
clc             % Limpieza de la ventana de comandos
close all       % Cierre de todas las figuras existentes
format short e  % Formato numérico a trabajar

%%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%%

%   Ingreso de coordenadas de los nodos. (Geométria)
geom = [-0.300 -0.200  3.000   ;   % Coordenadas en X,Y,Z del Elem. Trid.
        -0.225 -0.200  3.000   ;
        -0.150 -0.200  3.000   ;
        -0.075 -0.200  3.000   ;
        0.000  -0.200  3.000   ;
        0.075  -0.200  3.000   ;
        0.150  -0.200  3.000   ;
        0.225  -0.200  3.000   ;
        0.300  -0.200  3.000   ;
        -0.300 -0.200  2.625   ;
        -0.225 -0.200  2.625   ;
```

```
-0.150 -0.200  2.625  ;
-0.075 -0.200  2.625  ;
0.000  -0.200  2.625  ;
0.075  -0.200  2.625  ;
0.150  -0.200  2.625  ;
0.225  -0.200  2.625  ;
0.300  -0.200  2.625  ;
-0.300 -0.200  2.250  ;
-0.225 -0.200  2.250  ;
-0.150 -0.200  2.250  ;
-0.075 -0.200  2.250  ;
0.000  -0.200  2.250  ;
0.075  -0.200  2.250  ;
0.150  -0.200  2.250  ;
0.225  -0.200  2.250  ;
0.300  -0.200  2.250  ;
-0.300 -0.200  1.875  ;
-0.225 -0.200  1.875  ;
-0.150 -0.200  1.875  ;
-0.075 -0.200  1.875  ;
0.000  -0.200  1.875  ;
0.075  -0.200  1.875  ;
0.150  -0.200  1.875  ;
0.225  -0.200  1.875  ;
0.300  -0.200  1.875  ;
-0.300 -0.200  1.500  ;
-0.225 -0.200  1.500  ;
-0.150 -0.200  1.500  ;
-0.075 -0.200  1.500  ;
0.000  -0.200  1.500  ;
0.075  -0.200  1.500  ;
0.150  -0.200  1.500  ;
0.225  -0.200  1.500  ;
0.300  -0.200  1.500  ;
-0.300 -0.200  1.125  ;
-0.225 -0.200  1.125  ;
-0.150 -0.200  1.125  ;
-0.075 -0.200  1.125  ;
0.000  -0.200  1.125  ;
0.075  -0.200  1.125  ;
0.150  -0.200  1.125  ;
0.225  -0.200  1.125  ;
0.300  -0.200  1.125  ;
-0.300 -0.200  0.750  ;
-0.225 -0.200  0.750  ;
-0.150 -0.200  0.750  ;
-0.075 -0.200  0.750  ;
0.000  -0.200  0.750  ;
0.075  -0.200  0.750  ;
0.150  -0.200  0.750  ;
0.225  -0.200  0.750  ;
0.300  -0.200  0.750  ;
-0.300 -0.200  0.375  ;
-0.225 -0.200  0.375  ;
```

```
-0.150 -0.200  0.375  ;
-0.075 -0.200  0.375  ;
0.000  -0.200  0.375  ;
0.075  -0.200  0.375  ;
0.150  -0.200  0.375  ;
0.225  -0.200  0.375  ;
0.300  -0.200  0.375  ;
-0.300 -0.200  0.000  ;
-0.225 -0.200  0.000  ;
-0.150 -0.200  0.000  ;
-0.075 -0.200  0.000  ;
0.000  -0.200  0.000  ;
0.075  -0.200  0.000  ;
0.150  -0.200  0.000  ;
0.225  -0.200  0.000  ;
0.300  -0.200  0.000  ;
-0.300 -0.100  3.000  ;
-0.225 -0.100  3.000  ;
-0.150 -0.100  3.000  ;
-0.075 -0.100  3.000  ;
0.000  -0.100  3.000  ;
0.075  -0.100  3.000  ;
0.150  -0.100  3.000  ;
0.225  -0.100  3.000  ;
0.300  -0.100  3.000  ;
-0.300 -0.100  2.625  ;
-0.225 -0.100  2.625  ;
-0.150 -0.100  2.625  ;
-0.075 -0.100  2.625  ;
0.000  -0.100  2.625  ;
0.075  -0.100  2.625  ;
0.150  -0.100  2.625  ;
0.225  -0.100  2.625  ;
0.300  -0.100  2.625  ;
-0.300 -0.100  2.250  ;
-0.225 -0.100  2.250  ;
-0.150 -0.100  2.250  ;
-0.075 -0.100  2.250  ;
0.000  -0.100  2.250  ;
0.075  -0.100  2.250  ;
0.150  -0.100  2.250  ;
0.225  -0.100  2.250  ;
0.300  -0.100  2.250  ;
-0.300 -0.100  1.875  ;
-0.225 -0.100  1.875  ;
-0.150 -0.100  1.875  ;
-0.075 -0.100  1.875  ;
0.000  -0.100  1.875  ;
0.075  -0.100  1.875  ;
0.150  -0.100  1.875  ;
0.225  -0.100  1.875  ;
0.300  -0.100  1.875  ;
-0.300 -0.100  1.500  ;
-0.225 -0.100  1.500  ;
```

```
-0.150 -0.100  1.500  ;
-0.075 -0.100  1.500  ;
0.000  -0.100  1.500  ;
0.075  -0.100  1.500  ;
0.150  -0.100  1.500  ;
0.225  -0.100  1.500  ;
0.300  -0.100  1.500  ;
-0.300 -0.100  1.125  ;
-0.225 -0.100  1.125  ;
-0.150 -0.100  1.125  ;
-0.075 -0.100  1.125  ;
0.000  -0.100  1.125  ;
0.075  -0.100  1.125  ;
0.150  -0.100  1.125  ;
0.225  -0.100  1.125  ;
0.300  -0.100  1.125  ;
-0.300 -0.100  0.750  ;
-0.225 -0.100  0.750  ;
-0.150 -0.100  0.750  ;
-0.075 -0.100  0.750  ;
0.000  -0.100  0.750  ;
0.075  -0.100  0.750  ;
0.150  -0.100  0.750  ;
0.225  -0.100  0.750  ;
0.300  -0.100  0.750  ;
-0.300 -0.100  0.375  ;
-0.225 -0.100  0.375  ;
-0.150 -0.100  0.375  ;
-0.075 -0.100  0.375  ;
0.000  -0.100  0.375  ;
0.075  -0.100  0.375  ;
0.150  -0.100  0.375  ;
0.225  -0.100  0.375  ;
0.300  -0.100  0.375  ;
-0.300 -0.100  0.000  ;
-0.225 -0.100  0.000  ;
-0.150 -0.100  0.000  ;
-0.075 -0.100  0.000  ;
0.000  -0.100  0.000  ;
0.075  -0.100  0.000  ;
0.150  -0.100  0.000  ;
0.225  -0.100  0.000  ;
0.300  -0.100  0.000  ;
-0.300 0.000   3.000  ;
-0.225 0.000   3.000  ;
-0.150 0.000   3.000  ;
-0.075 0.000   3.000  ;
0.000  0.000   3.000  ;
0.075  0.000   3.000  ;
0.150  0.000   3.000  ;
0.225  0.000   3.000  ;
0.300  0.000   3.000  ;
-0.300 0.000   2.625  ;
-0.225 0.000   2.625  ;
```

```
-0.150 0.000   2.625  ;
-0.075 0.000   2.625  ;
0.000  0.000   2.625  ;
0.075  0.000   2.625  ;
0.150  0.000   2.625  ;
0.225  0.000   2.625  ;
0.300  0.000   2.625  ;
-0.300 0.000   2.250  ;
-0.225 0.000   2.250  ;
-0.150 0.000   2.250  ;
-0.075 0.000   2.250  ;
0.000  0.000   2.250  ;
0.075  0.000   2.250  ;
0.150  0.000   2.250  ;
0.225  0.000   2.250  ;
0.300  0.000   2.250  ;
-0.300 0.000   1.875  ;
-0.225 0.000   1.875  ;
-0.150 0.000   1.875  ;
-0.075 0.000   1.875  ;
0.000  0.000   1.875  ;
0.075  0.000   1.875  ;
0.150  0.000   1.875  ;
0.225  0.000   1.875  ;
0.300  0.000   1.875  ;
-0.300 0.000   1.500  ;
-0.225 0.000   1.500  ;
-0.150 0.000   1.500  ;
-0.075 0.000   1.500  ;
0.000  0.000   1.500  ;
0.075  0.000   1.500  ;
0.150  0.000   1.500  ;
0.225  0.000   1.500  ;
0.300  0.000   1.500  ;
-0.300 0.000   1.125  ;
-0.225 0.000   1.125  ;
-0.150 0.000   1.125  ;
-0.075 0.000   1.125  ;
0.000  0.000   1.125  ;
0.075  0.000   1.125  ;
0.150  0.000   1.125  ;
0.225  0.000   1.125  ;
0.300  0.000   1.125  ;
-0.300 0.000   0.750  ;
-0.225 0.000   0.750  ;
-0.150 0.000   0.750  ;
-0.075 0.000   0.750  ;
0.000  0.000   0.750  ;
0.075  0.000   0.750  ;
0.150  0.000   0.750  ;
0.225  0.000   0.750  ;
0.300  0.000   0.750  ;
-0.300 0.000   0.375  ;
-0.225 0.000   0.375  ;
```

```
-0.150 0.000   0.375   ;
-0.075 0.000   0.375   ;
0.000  0.000   0.375   ;
0.075  0.000   0.375   ;
0.150  0.000   0.375   ;
0.225  0.000   0.375   ;
0.300  0.000   0.375   ;
-0.300 0.000   0.000   ;
-0.225 0.000   0.000   ;
-0.150 0.000   0.000   ;
-0.075 0.000   0.000   ;
0.000  0.000   0.000   ;
0.075  0.000   0.000   ;
0.150  0.000   0.000   ;
0.225  0.000   0.000   ;
0.300  0.000   0.000   ;
-0.300 0.100   3.000   ;
-0.225 0.100   3.000   ;
-0.150 0.100   3.000   ;
-0.075 0.100   3.000   ;
0.000  0.100   3.000   ;
0.075  0.100   3.000   ;
0.150  0.100   3.000   ;
0.225  0.100   3.000   ;
0.300  0.100   3.000   ;
-0.300 0.100   2.625   ;
-0.225 0.100   2.625   ;
-0.150 0.100   2.625   ;
-0.075 0.100   2.625   ;
0.000  0.100   2.625   ;
0.075  0.100   2.625   ;
0.150  0.100   2.625   ;
0.225  0.100   2.625   ;
0.300  0.100   2.625   ;
-0.300 0.100   2.250   ;
-0.225 0.100   2.250   ;
-0.150 0.100   2.250   ;
-0.075 0.100   2.250   ;
0.000  0.100   2.250   ;
0.075  0.100   2.250   ;
0.150  0.100   2.250   ;
0.225  0.100   2.250   ;
0.300  0.100   2.250   ;
-0.300 0.100   1.875   ;
-0.225 0.100   1.875   ;
-0.150 0.100   1.875   ;
-0.075 0.100   1.875   ;
0.000  0.100   1.875   ;
0.075  0.100   1.875   ;
0.150  0.100   1.875   ;
0.225  0.100   1.875   ;
0.300  0.100   1.875   ;
-0.300 0.100   1.500   ;
-0.225 0.100   1.500   ;
```

```
-0.150 0.100  1.500  ;
-0.075 0.100  1.500  ;
0.000  0.100  1.500  ;
0.075  0.100  1.500  ;
0.150  0.100  1.500  ;
0.225  0.100  1.500  ;
0.300  0.100  1.500  ;
-0.300 0.100  1.125  ;
-0.225 0.100  1.125  ;
-0.150 0.100  1.125  ;
-0.075 0.100  1.125  ;
0.000  0.100  1.125  ;
0.075  0.100  1.125  ;
0.150  0.100  1.125  ;
0.225  0.100  1.125  ;
0.300  0.100  1.125  ;
-0.300 0.100  0.750  ;
-0.225 0.100  0.750  ;
-0.150 0.100  0.750  ;
-0.075 0.100  0.750  ;
0.000  0.100  0.750  ;
0.075  0.100  0.750  ;
0.150  0.100  0.750  ;
0.225  0.100  0.750  ;
0.300  0.100  0.750  ;
-0.300 0.100  0.375  ;
-0.225 0.100  0.375  ;
-0.150 0.100  0.375  ;
-0.075 0.100  0.375  ;
0.000  0.100  0.375  ;
0.075  0.100  0.375  ;
0.150  0.100  0.375  ;
0.225  0.100  0.375  ;
0.300  0.100  0.375  ;
-0.300 0.100  0.000  ;
-0.225 0.100  0.000  ;
-0.150 0.100  0.000  ;
-0.075 0.100  0.000  ;
0.000  0.100  0.000  ;
0.075  0.100  0.000  ;
0.150  0.100  0.000  ;
0.225  0.100  0.000  ;
0.300  0.100  0.000  ;
-0.300 0.200  3.000  ;
-0.225 0.200  3.000  ;
-0.150 0.200  3.000  ;
-0.075 0.200  3.000  ;
0.000  0.200  3.000  ;
0.075  0.200  3.000  ;
0.150  0.200  3.000  ;
0.225  0.200  3.000  ;
0.300  0.200  3.000  ;
-0.300 0.200  2.625  ;
-0.225 0.200  2.625  ;
```

```
-0.150 0.200  2.625  ;
-0.075 0.200  2.625  ;
0.000  0.200  2.625  ;
0.075  0.200  2.625  ;
0.150  0.200  2.625  ;
0.225  0.200  2.625  ;
0.300  0.200  2.625  ;
-0.300 0.200  2.250  ;
-0.225 0.200  2.250  ;
-0.150 0.200  2.250  ;
-0.075 0.200  2.250  ;
0.000  0.200  2.250  ;
0.075  0.200  2.250  ;
0.150  0.200  2.250  ;
0.225  0.200  2.250  ;
0.300  0.200  2.250  ;
-0.300 0.200  1.875  ;
-0.225 0.200  1.875  ;
-0.150 0.200  1.875  ;
-0.075 0.200  1.875  ;
0.000  0.200  1.875  ;
0.075  0.200  1.875  ;
0.150  0.200  1.875  ;
0.225  0.200  1.875  ;
0.300  0.200  1.875  ;
-0.300 0.200  1.500  ;
-0.225 0.200  1.500  ;
-0.150 0.200  1.500  ;
-0.075 0.200  1.500  ;
0.000  0.200  1.500  ;
0.075  0.200  1.500  ;
0.150  0.200  1.500  ;
0.225  0.200  1.500  ;
0.300  0.200  1.500  ;
-0.300 0.200  1.125  ;
-0.225 0.200  1.125  ;
-0.150 0.200  1.125  ;
-0.075 0.200  1.125  ;
0.000  0.200  1.125  ;
0.075  0.200  1.125  ;
0.150  0.200  1.125  ;
0.225  0.200  1.125  ;
0.300  0.200  1.125  ;
-0.300 0.200  0.750  ;
-0.225 0.200  0.750  ;
-0.150 0.200  0.750  ;
-0.075 0.200  0.750  ;
0.000  0.200  0.750  ;
0.075  0.200  0.750  ;
0.150  0.200  0.750  ;
0.225  0.200  0.750  ;
0.300  0.200  0.750  ;
-0.300 0.200  0.375  ;
-0.225 0.200  0.375  ;
```

```
        -0.150 0.200   0.375   ;
        -0.075 0.200   0.375   ;
        0.000  0.200   0.375   ;
        0.075  0.200   0.375   ;
        0.150  0.200   0.375   ;
        0.225  0.200   0.375   ;
        0.300  0.200   0.375   ;
        -0.300 0.200   0.000   ;
        -0.225 0.200   0.000   ;
        -0.150 0.200   0.000   ;
        -0.075 0.200   0.000   ;
        0.000  0.200   0.000   ;
        0.075  0.200   0.000   ;
        0.150  0.200   0.000   ;
        0.225  0.200   0.000   ;
        0.300  0.200   0.000];

%  Cambio de eje de las coordenadas
[nnd]=size(geom,1);     % Número total de nodos en el sistema

geom2=zeros(nnd,3);       % Modificación de los ejes en función de la data inicial.
geom2(:,1)=geom(:,3);
geom2(:,2)=geom(:,2);
geom2(:,3)=geom(:,1);
geom=geom2;

%   Ingreso de información sobre conectividad entre nodos. (Conectividad)
conect = [  82 83     92     91     1      2      11     10     ;
            83 84     93     92     2      3      12     11     ;
            84 85     94     93     3      4      13     12     ;
            85 86     95     94     4      5      14     13     ;
            86 87     96     95     5      6      15     14     ;
            87 88     97     96     6      7      16     15     ;
            88 89     98     97     7      8      17     16     ;
            89 90     99     98     8      9      18     17     ;
            91 92     101    100    10     11     20     19     ;
            92 93     102    101    11     12     21     20     ;
            93 94     103    102    12     13     22     21     ;
            94 95     104    103    13     14     23     22     ;
            95 96     105    104    14     15     24     23     ;
            96 97     106    105    15     16     25     24     ;
            97 98     107    106    16     17     26     25     ;
            98 99     108    107    17     18     27     26     ;
            100       101    110    109    19     20     29     28     ;
            101       102    111    110    20     21     30     29     ;
            102       103    112    111    21     22     31     30     ;
            103       104    113    112    22     23     32     31     ;
            104       105    114    113    23     24     33     32     ;
            105       106    115    114    24     25     34     33     ;
            106       107    116    115    25     26     35     34     ;
            107       108    117    116    26     27     36     35     ;
            109       110    119    118    28     29     38     37     ;
            110       111    120    119    29     30     39     38     ;
            111       112    121    120    30     31     40     39     ;
```

```
112     113     122     121     31      32      41      40      ;
113     114     123     122     32      33      42      41      ;
114     115     124     123     33      34      43      42      ;
115     116     125     124     34      35      44      43      ;
116     117     126     125     35      36      45      44      ;
118     119     128     127     37      38      47      46      ;
119     120     129     128     38      39      48      47      ;
120     121     130     129     39      40      49      48      ;
121     122     131     130     40      41      50      49      ;
122     123     132     131     41      42      51      50      ;
123     124     133     132     42      43      52      51      ;
124     125     134     133     43      44      53      52      ;
125     126     135     134     44      45      54      53      ;
127     128     137     136     46      47      56      55      ;
128     129     138     137     47      48      57      56      ;
129     130     139     138     48      49      58      57      ;
130     131     140     139     49      50      59      58      ;
131     132     141     140     50      51      60      59      ;
132     133     142     141     51      52      61      60      ;
133     134     143     142     52      53      62      61      ;
134     135     144     143     53      54      63      62      ;
136     137     146     145     55      56      65      64      ;
137     138     147     146     56      57      66      65      ;
138     139     148     147     57      58      67      66      ;
139     140     149     148     58      59      68      67      ;
140     141     150     149     59      60      69      68      ;
141     142     151     150     60      61      70      69      ;
142     143     152     151     61      62      71      70      ;
143     144     153     152     62      63      72      71      ;
145     146     155     154     64      65      74      73      ;
146     147     156     155     65      66      75      74      ;
147     148     157     156     66      67      76      75      ;
148     149     158     157     67      68      77      76      ;
149     150     159     158     68      69      78      77      ;
150     151     160     159     69      70      79      78      ;
151     152     161     160     70      71      80      79      ;
152     153     162     161     71      72      81      80      ;
163     164     173     172     82      83      92      91      ;
164     165     174     173     83      84      93      92      ;
165     166     175     174     84      85      94      93      ;
166     167     176     175     85      86      95      94      ;
167     168     177     176     86      87      96      95      ;
168     169     178     177     87      88      97      96      ;
169     170     179     178     88      89      98      97      ;
170     171     180     179     89      90      99      98      ;
172     173     182     181     91      92      101     100     ;
173     174     183     182     92      93      102     101     ;
174     175     184     183     93      94      103     102     ;
175     176     185     184     94      95      104     103     ;
176     177     186     185     95      96      105     104     ;
177     178     187     186     96      97      106     105     ;
178     179     188     187     97      98      107     106     ;
179     180     189     188     98      99      108     107     ;
181     182     191     190     100     101     110     109     ;
```

```
182     183     192     191     101     102     111     110     ;
183     184     193     192     102     103     112     111     ;
184     185     194     193     103     104     113     112     ;
185     186     195     194     104     105     114     113     ;
186     187     196     195     105     106     115     114     ;
187     188     197     196     106     107     116     115     ;
188     189     198     197     107     108     117     116     ;
190     191     200     199     109     110     119     118     ;
191     192     201     200     110     111     120     119     ;
192     193     202     201     111     112     121     120     ;
193     194     203     202     112     113     122     121     ;
194     195     204     203     113     114     123     122     ;
195     196     205     204     114     115     124     123     ;
196     197     206     205     115     116     125     124     ;
197     198     207     206     116     117     126     125     ;
199     200     209     208     118     119     128     127     ;
200     201     210     209     119     120     129     128     ;
201     202     211     210     120     121     130     129     ;
202     203     212     211     121     122     131     130     ;
203     204     213     212     122     123     132     131     ;
204     205     214     213     123     124     133     132     ;
205     206     215     214     124     125     134     133     ;
206     207     216     215     125     126     135     134     ;
208     209     218     217     127     128     137     136     ;
209     210     219     218     128     129     138     137     ;
210     211     220     219     129     130     139     138     ;
211     212     221     220     130     131     140     139     ;
212     213     222     221     131     132     141     140     ;
213     214     223     222     132     133     142     141     ;
214     215     224     223     133     134     143     142     ;
215     216     225     224     134     135     144     143     ;
217     218     227     226     136     137     146     145     ;
218     219     228     227     137     138     147     146     ;
219     220     229     228     138     139     148     147     ;
220     221     230     229     139     140     149     148     ;
221     222     231     230     140     141     150     149     ;
222     223     232     231     141     142     151     150     ;
223     224     233     232     142     143     152     151     ;
224     225     234     233     143     144     153     152     ;
226     227     236     235     145     146     155     154     ;
227     228     237     236     146     147     156     155     ;
228     229     238     237     147     148     157     156     ;
229     230     239     238     148     149     158     157     ;
230     231     240     239     149     150     159     158     ;
231     232     241     240     150     151     160     159     ;
232     233     242     241     151     152     161     160     ;
233     234     243     242     152     153     162     161     ;
244     245     254     253     163     164     173     172     ;
245     246     255     254     164     165     174     173     ;
246     247     256     255     165     166     175     174     ;
247     248     257     256     166     167     176     175     ;
248     249     258     257     167     168     177     176     ;
249     250     259     258     168     169     178     177     ;
250     251     260     259     169     170     179     178     ;
```

```
251     252     261     260     170     171     180     179     ;
253     254     263     262     172     173     182     181     ;
254     255     264     263     173     174     183     182     ;
255     256     265     264     174     175     184     183     ;
256     257     266     265     175     176     185     184     ;
257     258     267     266     176     177     186     185     ;
258     259     268     267     177     178     187     186     ;
259     260     269     268     178     179     188     187     ;
260     261     270     269     179     180     189     188     ;
262     263     272     271     181     182     191     190     ;
263     264     273     272     182     183     192     191     ;
264     265     274     273     183     184     193     192     ;
265     266     275     274     184     185     194     193     ;
266     267     276     275     185     186     195     194     ;
267     268     277     276     186     187     196     195     ;
268     269     278     277     187     188     197     196     ;
269     270     279     278     188     189     198     197     ;
271     272     281     280     190     191     200     199     ;
272     273     282     281     191     192     201     200     ;
273     274     283     282     192     193     202     201     ;
274     275     284     283     193     194     203     202     ;
275     276     285     284     194     195     204     203     ;
276     277     286     285     195     196     205     204     ;
277     278     287     286     196     197     206     205     ;
278     279     288     287     197     198     207     206     ;
280     281     290     289     199     200     209     208     ;
281     282     291     290     200     201     210     209     ;
282     283     292     291     201     202     211     210     ;
283     284     293     292     202     203     212     211     ;
284     285     294     293     203     204     213     212     ;
285     286     295     294     204     205     214     213     ;
286     287     296     295     205     206     215     214     ;
287     288     297     296     206     207     216     215     ;
289     290     299     298     208     209     218     217     ;
290     291     300     299     209     210     219     218     ;
291     292     301     300     210     211     220     219     ;
292     293     302     301     211     212     221     220     ;
293     294     303     302     212     213     222     221     ;
294     295     304     303     213     214     223     222     ;
295     296     305     304     214     215     224     223     ;
296     297     306     305     215     216     225     224     ;
298     299     308     307     217     218     227     226     ;
299     300     309     308     218     219     228     227     ;
300     301     310     309     219     220     229     228     ;
301     302     311     310     220     221     230     229     ;
302     303     312     311     221     222     231     230     ;
303     304     313     312     222     223     232     231     ;
304     305     314     313     223     224     233     232     ;
305     306     315     314     224     225     234     233     ;
307     308     317     316     226     227     236     235     ;
308     309     318     317     227     228     237     236     ;
309     310     319     318     228     229     238     237     ;
310     311     320     319     229     230     239     238     ;
311     312     321     320     230     231     240     239     ;
```

```
312     313     322     321     231     232     241     240     ;
313     314     323     322     232     233     242     241     ;
314     315     324     323     233     234     243     242     ;
325     326     335     334     244     245     254     253     ;
326     327     336     335     245     246     255     254     ;
327     328     337     336     246     247     256     255     ;
328     329     338     337     247     248     257     256     ;
329     330     339     338     248     249     258     257     ;
330     331     340     339     249     250     259     258     ;
331     332     341     340     250     251     260     259     ;
332     333     342     341     251     252     261     260     ;
334     335     344     343     253     254     263     262     ;
335     336     345     344     254     255     264     263     ;
336     337     346     345     255     256     265     264     ;
337     338     347     346     256     257     266     265     ;
338     339     348     347     257     258     267     266     ;
339     340     349     348     258     259     268     267     ;
340     341     350     349     259     260     269     268     ;
341     342     351     350     260     261     270     269     ;
343     344     353     352     262     263     272     271     ;
344     345     354     353     263     264     273     272     ;
345     346     355     354     264     265     274     273     ;
346     347     356     355     265     266     275     274     ;
347     348     357     356     266     267     276     275     ;
348     349     358     357     267     268     277     276     ;
349     350     359     358     268     269     278     277     ;
350     351     360     359     269     270     279     278     ;
352     353     362     361     271     272     281     280     ;
353     354     363     362     272     273     282     281     ;
354     355     364     363     273     274     283     282     ;
355     356     365     364     274     275     284     283     ;
356     357     366     365     275     276     285     284     ;
357     358     367     366     276     277     286     285     ;
358     359     368     367     277     278     287     286     ;
359     360     369     368     278     279     288     287     ;
361     362     371     370     280     281     290     289     ;
362     363     372     371     281     282     291     290     ;
363     364     373     372     282     283     292     291     ;
364     365     374     373     283     284     293     292     ;
365     366     375     374     284     285     294     293     ;
366     367     376     375     285     286     295     294     ;
367     368     377     376     286     287     296     295     ;
368     369     378     377     287     288     297     296     ;
370     371     380     379     289     290     299     298     ;
371     372     381     380     290     291     300     299     ;
372     373     382     381     291     292     301     300     ;
373     374     383     382     292     293     302     301     ;
374     375     384     383     293     294     303     302     ;
375     376     385     384     294     295     304     303     ;
376     377     386     385     295     296     305     304     ;
377     378     387     386     296     297     306     305     ;
379     380     389     388     298     299     308     307     ;
380     381     390     389     299     300     309     308     ;
381     382     391     390     300     301     310     309     ;
```

```
            382     383     392     391     301     302     311     310     ;
            383     384     393     392     302     303     312     311     ;
            384     385     394     393     303     304     313     312     ;
            385     386     395     394     304     305     314     313     ;
            386     387     396     395     305     306     315     314     ;
            388     389     398     397     307     308     317     316     ;
            389     390     399     398     308     309     318     317     ;
            390     391     400     399     309     310     319     318     ;
            391     392     401     400     310     311     320     319     ;
            392     393     402     401     311     312     321     320     ;
            393     394     403     402     312     313     322     321     ;
            394     395     404     403     313     314     323     322     ;
            395     396     405     404     314     315     324     323];
[a]=size(conect,1);

% Cambiando el orden de la conectividad
conect2=zeros(a,8);
conect2(:,1)=conect(:,5);
conect2(:,2)=conect(:,1);
conect2(:,3)=conect(:,4);
conect2(:,4)=conect(:,8);
conect2(:,5)=conect(:,6);
conect2(:,6)=conect(:,2);
conect2(:,7)=conect(:,3);
conect2(:,8)=conect(:,7);
conect=conect2;

%   Información sobre el sistema (Automático)
nne = 8;                    %   Número de nodos por elemento
ndgdl = 3;                  %   Número de GDLs por nodo
[nnd] = size(geom,1);       %   Número de nodos en el sistema
[nel] = size(conect,1);     %   Número de elementos
elgdl = nne*ndgdl;          %   Número de GDLs por elemento
sigdl = nnd*ndgdl;          %   Número de GDLs en el sistema
nglx=2; ngly=2; nglz=2;     %   Número de puntos de Gauss a flexión
E=2.535*10^9;               %   Módulo de elasticidad del sistema
vu=0.20;                    %   Módulo de Poisson

%   Dando numeración a todos los GDLs en el sistema
Numnd = ones(nnd, ndgdl);      % Creación de la Matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para asignar la numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

%   Condiciones de frontera por nodo
nf = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para asignar la numeración
```

```
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end

i=[5 86 167 248 329];   % Aplicando restricciones en los GDLs
[a]=size(i,1);
for ios=1:a
    nf(i,1)=0;
    nf(i,2)=0;
    nf(i,3)=0;
end

i=[401 320 239 158 77];
[a]=size(i,1);
for ios=1:a
    nf(i,1)=0;
    nf(i,2)=0;
    nf(i,3)=0;
end

%   Obteniendo los GDLs restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle para asignar la numeración
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end
[b]=size(ResGDL,2);
bcval=zeros(1,b);

%      Fuerzas aplicadas al sistema
CargNod = zeros(nnd,3);   % Carga en y/o momento aplicado directamente

P=56.25;                  %   Carga aplicada (Valor general-en función del ejemplo)
%   Extremos
CargNod(9,3)=P/4;
CargNod(90,3)=P/2;
CargNod(18,3)=P/2;
CargNod(99,3)=P;

CargNod(333,3)=P/4;
CargNod(252,3)=P/2;
CargNod(342,3)=P/2;
CargNod(261,3)=P;

CargNod(81,3)=P/4;
CargNod(162,3)=P/2;
```

```
CargNod(72,3)=P/2;
CargNod(153,3)=P;

CargNod(405,3)=P/4;
CargNod(324,3)=P/2;
CargNod(396,3)=P/2;
CargNod(315,3)=P;

%   Laterales
CargNod(27,3)=P/2;
CargNod(36,3)=P/2;
CargNod(108,3)=P;
CargNod(117,3)=P;

CargNod(45,3)=P/2;
CargNod(54,3)=P/2;
CargNod(126,3)=P;
CargNod(135,3)=P;

CargNod(351,3)=P/2;
CargNod(360,3)=P/2;
CargNod(270,3)=P;
CargNod(279,3)=P;

CargNod(369,3)=P/2;
CargNod(378,3)=P/2;
CargNod(288,3)=P;
CargNod(297,3)=P;

CargNod(63,3)=P/2;
CargNod(387,3)=P/2;
CargNod(306,3)=P;
CargNod(144,3)=P;

CargNod(243,3)=P/2;
CargNod(171,3)=P/2;
CargNod(234,3)=P;
CargNod(180,3)=P;

%   Centrales
CargNod(189,3)=P;
CargNod(198,3)=P;
CargNod(207,3)=P;
CargNod(216,3)=P;
CargNod(225,3)=P;

%   Limpieza de variables innecesarias
clear c n i j

%   Ejecutando la función para el procesamiento de datos
disp('Procesando Datos');
[delta,SIGMA,stress,conectesf] =
Tri_Elemento_Programacion(nf,nglx,ngly,nglz,nnd,nel,conect,geom,E,vu,elgdl,CargNod,sigdl
,Numnd,ndgdl,ResGDL,nne);
```

```
%%%%%%%%%% ------- ORDENANDO DESPLAZAMIENTOS ------- %%%%%%%%%%
for i=1: nnd %
    if nf(i,1) == 0
        x_disp =0.;
    else
        x_disp = delta(Numnd(i,1));
    end
    %
    if nf(i,2) == 0
        y_disp = 0.;
    else
        y_disp = delta(Numnd(i,2));
    end

    if nf(i,3) == 0
        z_disp = 0.;
    else
        z_disp = delta(Numnd(i,3));
    end
    DISP(i,:) = [x_disp y_disp z_disp];
end

%%%%%%%%%% ------- ORDENANDO ESFUERZOS ------- %%%%%%%%%%
% 1 Punto de integración
Element_Bending=SIGMA(:,1:3);
Element_Shear=SIGMA(:,4:6);
for k = 1:nnd
    mx = 0. ; my = 0.; mxy = 0.; qx = 0.; qy = 0.; qxy = 0.;
    ne = 0; p=0;
    for iel = 1:nel
        for jel=1:nne
            if conect(iel,jel) == k
                ne=ne+1;
                mx = mx + Element_Bending(iel,1);
                my = my + Element_Bending(iel,2);
                mxy = mxy + Element_Bending(iel,3);
                qx = qx + Element_Shear(iel,1);
                qy = qy + Element_Shear(iel,2);
                qxy = qxy + Element_Shear(iel,3);
            end
        end
    end
    ZX(k,1) = mx/ne;
    ZY(k,1) = my/ne;
    ZZ(k,1) = mxy/ne;
    Z1(k,1) = qx/ne;
    Z2(k,1) = qy/ne;
    Z3(k,1) = qxy/ne;
end

% 4 Puntos de integración
Element_Bending=stress(:,1:3);
Element_Shear=stress(:,4:6);
```

```
for k = 1:nnd
    mx = 0. ; my = 0.; mxy = 0.; qx = 0.; qy = 0.; qxy = 0.;
    ne = 0; p=0;
    for iel = 1:nel
        for jel=1:nne
            if conect(iel,jel) == k
                ne=ne+1;
                p=iel*8-8+jel;
                mx = mx + Element_Bending(p,1);
                my = my + Element_Bending(p,2);
                mxy = mxy + Element_Bending(p,3);
                qx = qx + Element_Shear(p,1);
                qy = qy + Element_Shear(p,2);
                qxy = qxy + Element_Shear(p,3);
            end
        end
    end
    ZX2(k,1) = mx/ne;
    ZY2(k,1) = my/ne;
    ZZ2(k,1) = mxy/ne;
    Z12(k,1) = qx/ne;
    Z22(k,1) = qy/ne;
    Z32(k,1) = qxy/ne;
end

%%%%%%%%%% ------- GRÁFICA DESPLAZAMIENTO ------- %%%%%%%%%%
disp('Mostrando Graficos');

% Gráfica de las deformaciones
hold on
U = DISP(:,3);
cmin = min(U);
cmax = max(U);
caxis([cmin cmax]);

p=1;
for i=1: nel    % Bucle para el número de elementos
        s1=[conect(i,1) conect(i,2) conect(i,3) conect(i,4)];
        s2=[conect(i,5) conect(i,6) conect(i,7) conect(i,8)];
        s3=[conect(i,1) conect(i,5) conect(i,8) conect(i,4)];
        s4=[conect(i,2) conect(i,6) conect(i,7) conect(i,3)];
        s5=[conect(i,5) conect(i,6) conect(i,2) conect(i,1)];
        s6=[conect(i,8) conect(i,7) conect(i,3) conect(i,4)];
    connecnew(p,1:4)=s1;
    connecnew(p+1,1:4)=s2;
    connecnew(p+2,1:4)=s3;
    connecnew(p+3,1:4)=s4;
    connecnew(p+4,1:4)=s5;
    connecnew(p+5,1:4)=s6;
    p=p+6;
end

[a,b]=size(geom);
geom3=zeros(a,3);
```

```
patch('Faces', connecnew, 'Vertices', geom, 'FaceVertexCData',U,...
         'Facecolor','interp','Marker','.');
colorbar;
title('Desplazamiento');
ylim([-2 2])
zlim([-1 1])

%%%%%%%%% ------- GRÁFICA ESFUERZOS CENTRO DEL ELEMENTO ------- %%%%%%%%%

figure
cmin = min(ZX);
cmax = max(ZX);
caxis([cmin cmax]);

p=1;
for i=1: nel    % Bucle para el número de elementos
        s1=[conectesf(i,1) conectesf(i,2) conectesf(i,3) conectesf(i,4)];
        s2=[conectesf(i,5) conectesf(i,6) conectesf(i,7) conectesf(i,8)];
        s3=[conectesf(i,1) conectesf(i,5) conectesf(i,8) conectesf(i,4)];
        s4=[conectesf(i,2) conectesf(i,6) conectesf(i,7) conectesf(i,3)];
        s5=[conectesf(i,5) conectesf(i,6) conectesf(i,2) conectesf(i,1)];
        s6=[conectesf(i,8) conectesf(i,7) conectesf(i,3) conectesf(i,4)];
    connecnew2(p,1:4)=s1;
    connecnew2(p+1,1:4)=s2;
    connecnew2(p+2,1:4)=s3;
    connecnew2(p+3,1:4)=s4;
    connecnew2(p+4,1:4)=s5;
    connecnew2(p+5,1:4)=s6;
    p=p+6;
end
patch('Faces',connecnew, 'Vertices', geom, 'FaceVertexCData',ZX,...
'Facecolor','interp','Marker','.');
colorbar;

title('Comparativa ZX - Tridimensional Centro del Elemento')
ylim([-2 2])
zlim([-1 1])

figure
cmin = min(Z1);
cmax = max(Z1);
caxis([cmin cmax]);
patch('Faces',connecnew, 'Vertices', geom, 'FaceVertexCData',Z1,...
'Facecolor','interp','Marker','.');
colorbar;

title('Comparativa Z1 - Tridimensional Centro del Elemento')
ylim([-2 2])
zlim([-1 1])

%%%%%%%%% ------- GRÁFICA ESFUERZOS, EXTREMO DEL ELEMENTO ------- %%%%%%%%%
figure
cmin = min(ZX2);
cmax = max(ZX2);
```

```
caxis([cmin cmax]);

p=1;
for i=1: nel    % Bucle para el número de elementos
        s1=[conectesf(i,1) conectesf(i,2) conectesf(i,3) conectesf(i,4)];
        s2=[conectesf(i,5) conectesf(i,6) conectesf(i,7) conectesf(i,8)];
        s3=[conectesf(i,1) conectesf(i,5) conectesf(i,8) conectesf(i,4)];
        s4=[conectesf(i,2) conectesf(i,6) conectesf(i,7) conectesf(i,3)];
        s5=[conectesf(i,5) conectesf(i,6) conectesf(i,2) conectesf(i,1)];
        s6=[conectesf(i,8) conectesf(i,7) conectesf(i,3) conectesf(i,4)];
    connecnew2(p,1:4)=s1;
    connecnew2(p+1,1:4)=s2;
    connecnew2(p+2,1:4)=s3;
    connecnew2(p+3,1:4)=s4;
    connecnew2(p+4,1:4)=s5;
    connecnew2(p+5,1:4)=s6;
    p=p+6;
end

patch('Faces',connecnew, 'Vertices', geom, 'FaceVertexCData',ZX2,...
'Facecolor','interp','Marker','.');
colorbar;

title('Comparativa ZX2 - Tridimensional Extremos del Elemento')
ylim([-2 2])
zlim([-1 1])

disp('Procesamiento de Datos Terminado');

%   Limpieza de variables innecesarias
clear c n i j a b cmax cmin DR x y xcoord ycoord zcoord vu C coord E eld delta...
    F g k k1 k2 k3 k4 kg KG MCF MD m l intx iel nel nf nnd nne ndgdl node_disp...
    npgf Q ResGDL sigdl Numnd Moment geom elgdl Element_Forces Esp
```

Published with MATLAB® R2019b

ANEXO Nº03
Código de la Comparativa – Capitulo 7.

Código Optimizado / Usuario – Escalera Helicoidal

```
%{
Propósito:
    Analizar un sistema compuesto de un elemento tridimensional con el FEM.
    Código para el ingreso de datos por el usuario.
Variables de Ingreso:
    Especificadas en el codigo.
Diseñador:
    Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Setiembre 2019
%}

%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%

clear           % Limpieza del espacio de trabajo
clc             % Limpieza de la ventana de Comandos
close all       % Cierre de todas las figuras Existentes

%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%

%   Ingreso de Coordenadas de los Nodos. (Geométria)
dataSis='Helicoidal_Coordenadas.txt';
geom = importdata(dataSis);
geom=geom(:,2:4);

%   Ingreso de Datos de conectividad entre nodos. (Conectividad)
dataSis='Helicoidal_Conectividad.txt';
conect = importdata(dataSis);
conect=conect(:,2:9);

% Propiedades del Sistema
nne = 8;                    %   Número de nodos por elemento
ndgdl = 3;                  %   Número de Grados de libertad por nodo
[nnd,~] = size(geom);       %   Número de nodos en el sistema
[nel,~] = size(conect);     %   Número de elementos
elgdl = nne*ndgdl;          %   Número de GDLs por elemento
sigdl = nnd*ndgdl;          %   Número de GDLs en el sistema
nglx=2; ngly=2; nglz=2;     %   Número de puntos de Gauss a flexión
dim = 3;                    %   Dimensión del elemento a analizar

% Propiedades de material del sistema (Uniforme)
E=2.535*10^9;               %   Módulo de elasticidad del sistema
vu=0.20;                    %   Módulo de Poisson

% Cambio de ejes del sistema (Opcional)
geom2=zeros(nnd,3);
geom2(:,1)=geom(:,3);
geom2(:,2)=geom(:,1);
```

```
geom2(:,3)=geom(:,2);
geom=geom2;

% Cambio del orden de la conectividad
conect2=zeros(nel,8);
conect2(:,1)=conect(:,5);
conect2(:,2)=conect(:,6);
conect2(:,3)=conect(:,7);
conect2(:,4)=conect(:,8);
conect2(:,5)=conect(:,1);
conect2(:,6)=conect(:,2);
conect2(:,7)=conect(:,3);
conect2(:,8)=conect(:,4);
conect=conect2;

%   Dando numeración a todos los GDLs en el sistema
Numnd = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

% Condiciones de frontera por nodo
nf = Numnd;      % Creación de la matriz con los GDLs del sistema

% Vector con los nodos con apoyo empotrado (para este ejemplo)
i=[7, 8, 9, 10, 283, 285, 287, 288, 324, 325, 326, 327, 328, 329, 330, 331,...
332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347,...
348, 349, 2198, 2199, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221,
2226,...
2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2250, 2251,
2252, 2253,...
2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267,
2268, 2269,...
2270, 2271, 4514, 4515, 4516, 4517, 4518, 4519, 4520, 4521, 4522, 4523, 4524, 4525,
4526, 4527,...
4528, 4529, 4530, 4531, 4532, 4533, 4534, 4535];

[a,~]=size(i);

% Bucle para la interpletación de caso como empotrado, si se desea asignar
% otro caso modificar el bucle o incorporar uno nuevo para un caso
% combinado como se muestra subsecuentemente
for ios=1:a
    nf(i,1)=0;
    nf(i,2)=0;
    nf(i,3)=0;
end
```

```
%   Obteniendo los GDLs restringidos en un vector aparte (ResGDL)
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end

%      Fuerzas aplicadas al sistema
CargNod = zeros(nnd,3);   % Carga en y/o momento aplicado directamente

i=[172, 176, 179, 180, 183, 186, 187, 189, 193, 194, 198, 201, 1456, 1457, 1458,
1459,...
1460, 1461, 1462, 1463, 1464, 1465, 1466, 1495, 1496, 1509, 1510, 1511, 1512, 1513,
1514, 1515,...
1516, 1517, 1518, 1519, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1540, 1541, 1542,
1543, 1544,...
1545, 1546, 1547, 1575, 1576, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585,
1586, 1587,...
1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1603, 1604, 1628,
1629, 1630,...
1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1640, 3684, 3685, 3686, 3687,
3688, 3689,...
3690, 3691, 3692, 3693, 3694, 3695, 3696, 3697, 3698, 3699, 3700, 3701, 3702, 3703,
3704, 3705,...
3736, 3737, 3738, 3739, 3740, 3741, 3742, 3743, 3744, 3745, 3746, 3747, 3748, 3749,
3750, 3751,...
3752, 3753, 3754, 3755, 3756, 3757, 3788, 3789, 3790, 3791, 3792, 3793, 3794, 3795,
3796, 3797,...
3798, 3799, 3800, 3801, 3802, 3803, 3804, 3805, 3806, 3807, 3808, 3809];

[~,a]=size(i);
for ios=1:a
    CargNod(i(ios),3)=3500;
end

%   Limpieza de variables innecesarias
clear a ios i j n geom2 conect2

%   Ejecutando la función para el procesamiento de datos
disp('Procesando Datos');
[delta,SIGMA,stress,conectesf] =
Tri_Elemento_Programacion(nf,nglx,ngly,nglz,nnd,nel,conect,geom,E,vu,elgdl,CargNod,sigdl
,Numnd,ndgdl,ResGDL,nne);

%%%%%%%%%% ------- ORDENANDO DESPLAZAMIENTOS - ALMACENAMIENTO ------- %%%%%%%%%%
% Inicializando variable que contendra los desplazamientos ordenados
DISP=zeros(nnd,dim);
for i=1: nnd % Bucle para el número total de nodos
    if nf(i,1) == 0 % Condicional si esta restringido en X
        x_disp =0.;
```

```
    else
        x_disp = delta(Numnd(i,1));
    end

    if nf(i,2) == 0 % Condicional si esta restringido en Y
        y_disp = 0.;
    else
        y_disp = delta(Numnd(i,2));
    end

    if nf(i,3) == 0 % Condicional si esta restringido en Z
        z_disp = 0.;
    else
        z_disp = delta(Numnd(i,3));
    end
    DISP(i,:) = [x_disp y_disp z_disp]; % Almacenamiento de los desp. ordenados
end

%%%%%%%%%% ------- ORDENANDO ESFUERZOS ------- %%%%%%%%%%
Element_Bending=SIGMA(:,1:3);
Element_Shear=SIGMA(:,4:6);
for k = 1:nnd
    mx = 0. ; my = 0.; mxy = 0.; qx = 0.; qy = 0.; qxy = 0.; ne = 0;
    for iel = 1:nel
        for jel=1:nne
            if conect(iel,jel) == k
                ne=ne+1;
                mx = mx + Element_Bending(iel,1);
                my = my + Element_Bending(iel,2);
                mxy = mxy + Element_Bending(iel,3);
                qx = qx + Element_Shear(iel,1);
                qy = qy + Element_Shear(iel,2);
                qxy = qxy + Element_Shear(iel,3);
            end
        end
    end
    ZX(k,1) = mx/ne;
    ZY(k,1) = my/ne;
    ZZ(k,1) = mxy/ne;
    Z1(k,1) = qx/ne;
    Z2(k,1) = qy/ne;
    Z3(k,1) = qxy/ne;
end

%%%%%%%%%% ------- GRÁFICA DESPLAZAMIENTO ------- %%%%%%%%%%
disp('Mostrando Graficos');
figure
hold on
U = DISP(:,3);
cmin = min(U);
cmax = max(U);
caxis([cmin cmax]);

p=1;
```

```
for i=1: nel    %Bucle para el número de elementos
        s1=[conect(i,1) conect(i,2) conect(i,3) conect(i,4)];
        s2=[conect(i,5) conect(i,6) conect(i,7) conect(i,8)];
        s3=[conect(i,1) conect(i,5) conect(i,8) conect(i,4)];
        s4=[conect(i,2) conect(i,6) conect(i,7) conect(i,3)];
        s5=[conect(i,5) conect(i,6) conect(i,2) conect(i,1)];
        s6=[conect(i,8) conect(i,7) conect(i,3) conect(i,4)];
    connecnew(p,1:4)=s1;
    connecnew(p+1,1:4)=s2;
    connecnew(p+2,1:4)=s3;
    connecnew(p+3,1:4)=s4;
    connecnew(p+4,1:4)=s5;
    connecnew(p+5,1:4)=s6;
    p=p+6;
end

patch('Faces', connecnew, 'Vertices', geom, 'FaceVertexCData',U,...
         'Facecolor','interp','Marker','.');
colormap(jet(12))
colorbar;
title('Desplazamiento (Uy)');

% Gráfica de desplazamientos en 2
figure
hold on
U = DISP(:,2);
cmin = min(U);
cmax = max(U);
caxis([cmin cmax]);

p=1;
for i=1: nel    %Bucle para el número de elementos
        s1=[conect(i,1) conect(i,2) conect(i,3) conect(i,4)];
        s2=[conect(i,5) conect(i,6) conect(i,7) conect(i,8)];
        s3=[conect(i,1) conect(i,5) conect(i,8) conect(i,4)];
        s4=[conect(i,2) conect(i,6) conect(i,7) conect(i,3)];
        s5=[conect(i,5) conect(i,6) conect(i,2) conect(i,1)];
        s6=[conect(i,8) conect(i,7) conect(i,3) conect(i,4)];
    connecnew(p,1:4)=s1;
    connecnew(p+1,1:4)=s2;
    connecnew(p+2,1:4)=s3;
    connecnew(p+3,1:4)=s4;
    connecnew(p+4,1:4)=s5;
    connecnew(p+5,1:4)=s6;
    p=p+6;
end

patch('Faces', connecnew, 'Vertices', geom, 'FaceVertexCData',U,...
         'Facecolor','interp','Marker','.');
colorbar;
title('Desplazamiento Ux');

figure
hold on
```

```
U = DISP(:,1);
cmin = min(U);
cmax = max(U);
caxis([cmin cmax]);

p=1;
for i=1: nel    %Bucle para el número de elementos
        s1=[conect(i,1) conect(i,2) conect(i,3) conect(i,4)];
        s2=[conect(i,5) conect(i,6) conect(i,7) conect(i,8)];
        s3=[conect(i,1) conect(i,5) conect(i,8) conect(i,4)];
        s4=[conect(i,2) conect(i,6) conect(i,7) conect(i,3)];
        s5=[conect(i,5) conect(i,6) conect(i,2) conect(i,1)];
        s6=[conect(i,8) conect(i,7) conect(i,3) conect(i,4)];
    connecnew(p,1:4)=s1;
    connecnew(p+1,1:4)=s2;
    connecnew(p+2,1:4)=s3;
    connecnew(p+3,1:4)=s4;
    connecnew(p+4,1:4)=s5;
    connecnew(p+5,1:4)=s6;
    p=p+6;
end

patch('Faces', connecnew, 'Vertices', geom, 'FaceVertexCData',U,...
         'Facecolor','interp','Marker','.');
colorbar;
title('Desplazamiento Uz');

%   Limpieza de variables innecesarias
clear c n i j a b cmax cmin DR x y xcoord ycoord zcoord vu C coord E eld delta...
    F g k k1 k2 k3 k4 kg KG MCF MD m l intx iel nel nf nnd nne ndgdl node_disp...
    npgf Q ResGDL sigdl Numnd Moment geom elgdl Element_Forces Esp
```

Published with MATLAB® R2019b

Código sin Optimizar / Usuario – Escalera Helicoidal

Tri_Elem_Usuario_HelicBase.m

```
%{
Propósito:
    Analizar un sistema compuesto de un elemento Tridimensional con el FEM.
    Código para el ingreso de datos por el usuario.
Variables de ingreso:
    Especificadas en el código.
Fecha de última actualización:
    Setiembre 2019
%}

% Inicializando variables globales
global nnd nel nne nodof eldof n nglx ngly nglz sdof
global geom connec dee nf Nodal_loads dim nf2 nf5
```

```
% Propiedades del sistema
nnd = 6123;                 %   Número total de nodos en el sistema
nel = 3888;                 %   Número de elementos
nne = 8;                    %   Número de nodos por elemento
nodof =3;                   %   Número de grados de libertad por nodo
eldof = nne*nodof;          %   Número de GDLs por elemento
sdof=nnd*nodof;             %   Número total de grados de libertad en el sistema
nglx=2; ngly=2; nglz=2;     %   Número de puntos de Gauss a flexión
dim = 3;                    %   Dimensión del elemento a analizar

% Propiedades del material
E=2.535*10^9;               %   Módulo de Young del sistema
vu=0.20;                    %   Módulo de Poisson
P=56.25;                    %   Carga aplicada en la parte superior del solido

% Matriz constitutiva del sistema
dee= E/((1+vu)*(1-2*vu))* ...
    [(1-vu)  vu     vu      0         0       0;
    vu     (1-vu)   vu      0         0       0;
    vu       vu   (1-vu)    0         0       0;
    0        0      0    (1-2*vu)/2   0       0;
    0        0      0       0    (1-2*vu)/2   0;
    0        0      0       0         0   (1-2*vu)/2];

%   Ingreso de coordenadas de los nodos. (Geométria)
dataSis='Helicoidal_Coordenadas.txt';
geom = importdata(dataSis);

%   Ingreso de datos de conectividad entre nodos. (Conectividad)
dataSis='Helicoidal_Conectividad.txt';
connec = importdata(dataSis);
connec=connec(:,2:9);

% Cambio de ejes del sistema (opcional)
geom2=zeros(6123,3);
geom2(:,1)=geom(:,3);
geom2(:,2)=geom(:,1);
geom2(:,3)=geom(:,2);
geom=geom2;

% Condiciones de borde
nf = ones(nnd, nodof); % Inicio de la matriz sin restricciones(1)

% Aplicando restricciones a los nodos correspondientes
i=[7, 8, 9, 10, 283, 285, 287, 288, 324, 325, 326, 327, 328, 329, 330, 331,...
332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347,...
348, 349, 2198, 2199, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221,
2226,...
2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2250, 2251,
2252, 2253,...
2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267,
2268, 2269,...
2270, 2271, 4514, 4515, 4516, 4517, 4518, 4519, 4520, 4521, 4522, 4523, 4524, 4525,
4526, 4527,...
```

```
4528, 4529, 4530, 4531, 4532, 4533, 4534, 4535];
[a,~]=size(i);

for ios=1:a
    nf(i,1)=0;
    nf(i,2)=0;
    nf(i,3)=0;
end

% Variables para ingresar la data correspondiente a las condiciones de
% borde
nf2 = zeros(nnd, nodof);
n=0;
for i=1:nnd
    for j=1:nodof
        n=n+1;
        nf2(i,j)=n;
    end
end

a=max(max(nf2));
nf3=zeros(a,1);
n=0;
for i=1:nnd
    for j=1:nodof
        if nf(i,j) == 0
            n=n+1;
            nf3(n)=nf2(i,j);
        end
    end
end

nf3=nf3';
b=1;
nf4=0;
while 1
    j=nf3(b);
    if j==0
        break
    else
        nf4(b)=nf3(b);
        b=b+1;
    end
end

nf5 = zeros(nnd, nodof);
n=1;
for i=1:nnd
    for j=1:nodof
        pq=nf(i,j);
        if pq==0
            nf5(i,j)=0;
        else
            nf5(i,j)=n;
```

```
            n=n+1;
        end
    end
end

%      Asignación de fuerzas aplicadas al sistema
Nodal_loads = zeros(nnd,3);

i=[172, 176, 179, 180, 183, 186, 187, 189, 193, 194, 198, 201, 1456, 1457, 1458,
1459,...
1460, 1461, 1462, 1463, 1464, 1465, 1466, 1495, 1496, 1509, 1510, 1511, 1512, 1513,
1514, 1515,...
1516, 1517, 1518, 1519, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1540, 1541, 1542,
1543, 1544,...
1545, 1546, 1547, 1575, 1576, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585,
1586, 1587,...
1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1603, 1604, 1628,
1629, 1630,...
1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1640, 3684, 3685, 3686, 3687,
3688, 3689,...
3690, 3691, 3692, 3693, 3694, 3695, 3696, 3697, 3698, 3699, 3700, 3701, 3702, 3703,
3704, 3705,...
3736, 3737, 3738, 3739, 3740, 3741, 3742, 3743, 3744, 3745, 3746, 3747, 3748, 3749,
3750, 3751,...
3752, 3753, 3754, 3755, 3756, 3757, 3788, 3789, 3790, 3791, 3792, 3793, 3794, 3795,
3796, 3797,...
3798, 3799, 3800, 3801, 3802, 3803, 3804, 3805, 3806, 3807, 3808, 3809];

[~,a]=size(i);
for ios=1:a
    Nodal_loads(i(ios),3)=3500;
end

% Ordenando las fuerzas aplicadas en forma de vector
b=1;
s=1;
ps=1;
ff=0;
while 2
    if b>sdof
        break
    else
        if s>3
            s=1;
            ps=ps+1;
        else
            ff(b)=Nodal_loads(ps,s);
            s=s+1;
            b=b+1;
        end
    end
end
[a,b]=size(nf4);
```

```
bcval=zeros(1,b);
ff=ff';
```

Published with MATLAB® R2019b

Código sin Optimizar / Programa – Escalera Helicoidal

Tri_Elem_ProgBas_Helicoidal.m

```
%{
Propósito:
    Analizar un sistema compuesto de un elemento Tridimensional con el FEM.
    Código para el procesamiento de datos interno
Descripción de datos de ingreso:
    - Ninguno, se utilizaran datos del codigo Tri_Elem_Usuario_HelicBase.m
Diseñador:
    Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Setiembre 2019
%}

% Aplicando comandos de limpieza
clc
clear variables
close all

% Asignando varibales globales para que puedan ser usadas en otras funciones

global nnd nel nne nodof eldof n nglx ngly nglz sdof
global geom connec dee nf Nodal_loads nf2 nf5

% A continuación se ingresara los datos de la escalera mediante otro
% programa.

Tri_Elem_Usuario_HelicBase

%   Obtención de los puntos de Gauss y los Pesos
[point3,weight3]=NumPInteg3D(nglx,ngly,nglz);

%   Creación de las matrices bases
kk=zeros(sdof,sdof);        % Matriz de rigidez general
disp=zeros(sdof,1);         % Vector de desplazamientos general
nd=zeros(1,nne);            % Nodos en el sistema
xcoord=zeros(1,nne);        % Coordenadas en X
ycoord=zeros(1,nne);        % Coordenadas en Y
zcoord=zeros(1,nne);        % Coordenadas en Z

for iel=1:nel           % Bucle para el número total de elementos
    iel
    k=zeros(eldof,eldof) ;      % Inicialización de la matriz de tigidez
    [coord,g] = dataelem(iel) ; % Coodernadas geométricas y GDLs libres del Elemento
     for i=1:nne
```

```
        nd(i)=connec(iel,i);        % Datos de conectividad
        xcoord(i)=geom(nd(i),1);    % Coordenadas en X
        ycoord(i)=geom(nd(i),2);    % Coordenadas en Y
        zcoord(i)=geom(nd(i),3);    % Coordenadas en Z
     end

    for intx=1:nglx
        x=point3(intx,1);       % Puntos de Int. X
        wtx=weight3(intx,1);    % Coef Peso X
        for inty=1:ngly
            y=point3(inty,2);       % Puntos de Int. Y
            wty=weight3(inty,2);    % Coef Peso Y
            for intz=1:nglz
                z=point3(intz,3);       % Puntos de Int. Z
                wtz=weight3(intz,3);    % Coef Peso Z
                % Cálculo de las funciones de forma y derivadas en
                % coordenadas naturales
                [shape,dhdr,dhds,dhdt]=Elem3D_8nodos(x,y,z);
                % Cálculo del Jacobiano del Sistema
                jacob3=jacob3d(nne,dhdr,dhds,dhdt,xcoord,ycoord,zcoord);
                % Determinante del Jacobiano
                detjacob=det(jacob3);
                % Inversa del Jacobiano
                invjacob=inv(jacob3);
                % Derivadas de las funciones de forma en coord. globales
                [dhdx,dhdy,dhdz]=Deriv3DGlobal(nne,dhdr,dhds,dhdt,invjacob);
                kinmtx=matdeform3d(nne,dhdx,dhdy,dhdz); % Matriz de deform.
                k=k+kinmtx'*dee*kinmtx*wtx*wty*wtz*detjacob; % M.Rigid general
            end
        end
    end
    index=gdls_elem(nd,nne,nodof); % Obtención de los GDLs del elemento
    kk=m_rigidglobal(kk,k,index);  % Ensamblaje / matriz de rigidez general

end

% Aplicando restricciones a la matriz de rigidez global
[kk,fg]=restring_rigid(kk,ff,nf4,bcval);

delta = kk\fg;      %   Obtención de los desplazamientos y giros correspondientes
num=1:1:sdof;
% Asignando numeración a los desplazamientos
displace=[num' delta];

for i=1: nnd
    if nf(i,1) == 0
        x_disp =0.;
    else
        x_disp = delta(nf2(i,1));
    end
    %
    if nf(i,2) == 0 %
        y_disp = 0.; %
    else
```

```
        y_disp = delta(nf2(i,2));
    end

    if nf(i,3) == 0
        z_disp = 0.;
    else
        z_disp = delta(nf2(i,3));
    end
    % Ordenando los desplazamientos
    DISP(i,:) = [x_disp y_disp z_disp];
end

%Gráfica de las deformaciones
hold on
U = DISP(:,3);
cmin = min(U);
cmax = max(U);
caxis([cmin cmax]);

p=1;
for i=1: nel    %Bucle para el número de elementos
        s1=[connec(i,1) connec(i,2) connec(i,3) connec(i,4)];
        s2=[connec(i,5) connec(i,6) connec(i,7) connec(i,8)];
        s3=[connec(i,1) connec(i,5) connec(i,8) connec(i,4)];
        s4=[connec(i,2) connec(i,6) connec(i,7) connec(i,3)];
        s5=[connec(i,5) connec(i,6) connec(i,2) connec(i,1)];
        s6=[connec(i,8) connec(i,7) connec(i,3) connec(i,4)];
    connecnew(p,1:4)=s1;
    connecnew(p+1,1:4)=s2;
    connecnew(p+2,1:4)=s3;
    connecnew(p+3,1:4)=s4;
    connecnew(p+4,1:4)=s5;
    connecnew(p+5,1:4)=s6;
    p=p+6;
end

patch('Faces', connecnew, 'Vertices', geom, 'FaceVertexCData',U,...
         'Facecolor','interp','Marker','.');
colorbar;
title('Desplazamiento');

%-- CÁLCULO DE ESFUERZOS Y DESPLAZAMIENTOS EN EL CENTRO DE CADA ELEMENTO --%
% Número de puntos de integración
nglx=1;
ngly=1;
nglz=1;
% Obteniendo los puntos de Int. y coef de pesos
[point3,weight3]=NumPInteg3D(nglx,ngly,nglz);

for iel=1:nel
    k=zeros(eldof,eldof) ; % Inicialización de la matriz de Rigidez
    [coord,g] = dataelem(iel) ; % Coodernadas geométricas y GDLs libres del elemento
    for i=1:nne
        nd(i)=connec(iel,i);            % Datos de conectividad
```

```
        xcoord(i)=geom(nd(i),1);        % Coordenadas en X
        ycoord(i)=geom(nd(i),2);        % Coordenadas en Y
        zcoord(i)=geom(nd(i),3);        % Coordenadas en Z
    end
    % Inicializando variables vacia de los desplazamientos del elemento
    eld=zeros(eldof,1);
    % Bucle para ingresar los desplazamientos obtenidos anteriormente
    for m=1:eldof
        if g(m)==0
            eld(m)=0.;
        else
            eld(m)=displace(g(m),2);
        end
    end

    for intx=1: nglx
        x=point3(intx,1);       % Puntos de Int. X
        wtx=weight3(intx,1); % Coef Peso X
        for inty=1: ngly
            y=point3(inty,2);       % Puntos de Int. Y
            wty=weight3(inty,2) ;    % Coef Peso Y
            for intz=1: nglz
                z=point3(intz,3);       % Puntos de Int. Z
                wtz=weight3(intz,3) ;       % Coef Peso Z
                % Cálculo de las funciones de forma y derivadas en
                % coordenadas naturales
                [shape,dhdr,dhds,dhdt]=Elem3D_8nodos(x,y,z);
                % Cálculo del Jacobiano del Sistema
                jacob3=jacob3d(nne,dhdr,dhds,dhdt,xcoord,ycoord,zcoord);
                % Determinante del Jacobiano
                detjacob=det(jacob3);
                % Inversa del Jacobiano
                invjacob=inv(jacob3);
                % Derivadas de las funciones de forma en coord. globales
                [dhdx,dhdy,dhdz]=Deriv3DGlobal(nne,dhdr,dhds,dhdt,invjacob);
                % Matriz de deformación del elemento
                kinmtx=matdeform3d(nne,dhdx,dhdy,dhdz);
                % Calculando las deformaciones del elemento
                eps=kinmtx*eld;
                % Calculando los esfuerzos del elemento
                sigma=dee*eps;
            end
        end
    end
% Almacenando el historial de esfuerzos del elemento
SIGMA(iel,:)=sigma;
end

% Subdiviendo los esfuerzos
ZX=zeros(8,1);
ZY=zeros(8,1);
ZZ=zeros(8,1);
Z1=zeros(8,1);
Z2=zeros(8,1);
```

```
Z3=zeros(8,1);
p=1;
while 1
    if p>8
        break
    else
        ZX(p,1)=SIGMA(1,1);
        ZY(p,1)=SIGMA(1,2);
        ZZ(p,1)=SIGMA(1,3);
        Z1(p,1)=SIGMA(1,4);
        Z2(p,1)=SIGMA(1,5);
        Z3(p,1)=SIGMA(1,6);
        p=p+1;
    end
end
```

Published with MATLAB® R2019b

Funciones extra código sin optimizar

Dataelem.m

```
%{
Propósito:
    Obtener las coordenadas y conectividad del elementos que se este
    analizando
Descripción de datos de ingreso:
    - nel = Número de elemento que se esta analizando
Fecha de última actualización:
    Enero 2019
%}

function[coord,conect] = dataelem(i)

% Coordenadas globales
global nne nodof geom connec nf dim nf5

% Inicializando variable de coordenadas del elemento
coord=zeros(nne,dim);
for k=1: nne
    for j=1:dim
        coord(k,j)=geom(connec(i,k),j);
    end
end

% Incorporando data de conectividad
l=0;
for k=1: nne
    for j=1:nodof
        l=l+1;
        conect(l)=nf(connec(i,k),j);
    end
```

```
end
end
```

Published with MATLAB® R2019b

Gdls_elem.m

```
%{
Propósito:
    Obener los grados de libertad de los nodos del elemento que se este
    analizando.
Descripción de datos de ingreso:
    - nd = Vector que contienen los nodos numerados globalmente del elemento.
    - nne = Número de nodos en el elemento.
    - ndof = Número de GDLs por nodo.
Fecha de última actualización:
    Enero 2019
%}

function [index]=gdls_elem(nd,nnel,ndof)
% Calculando los grados de Libertad totales en el elemento
edof = nnel*ndof;
% Bucle para obtener los GDLs del elemento en todos los nodos
k=0;
for i=1:nnel    % Bucle para cada uno de los nodos
    start = (nd(i)-1)*ndof;
    for j=1:ndof    % Bucle para cada uno de los GDLs por nodo
        k=k+1;
        index(k)=start+j;
    end
end
end
```

Published with MATLAB® R2019b

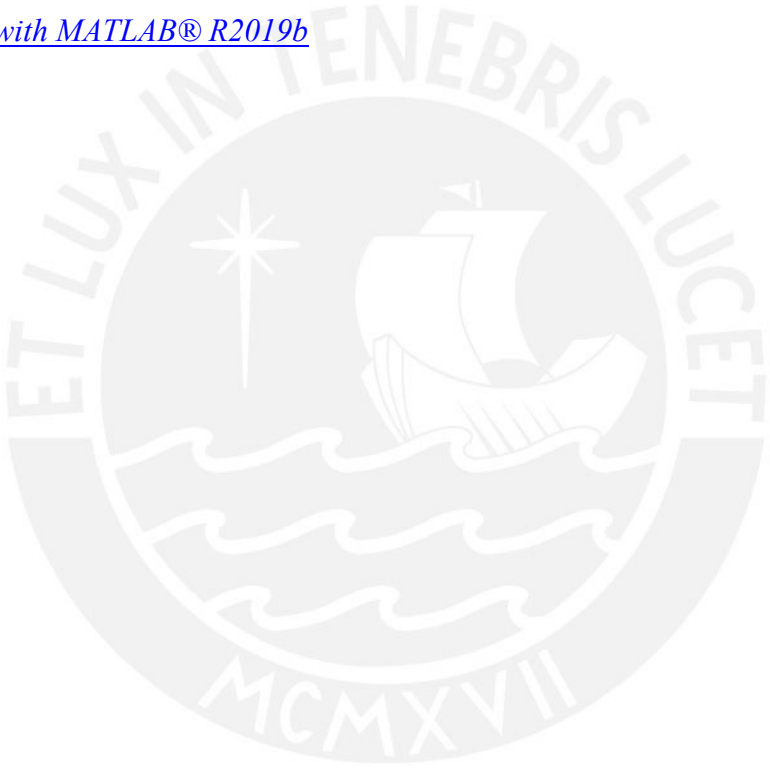
M_rigidglobal.m

```
%{
Propósito:
    Ensamblar la rigidez individual de cada elemento en una global.
Descripción de Datos de Ingreso:
    - kk = Matriz de rigidez global.
    - k = Matriz de rigidez individual por elemento.
    - index = GDLs del elemento analizado.
Fecha de última actualización:
    Setiembre 2019
%}

function [kk]=m_rigidglobal(kk,k,index)
% Calculando el numero total de GDLs del elemento
```

```
edof = length(index);
% Bucle para la asignacion de la data de matriz de rigidez individual a la
% global a partir de su numeracion en los GDLs globales establecidos
% anteriormente.
for i=1:edof
    ii=index(i);
    for j=1:edof
        jj=index(j);
        kk(ii,jj)=kk(ii,jj)+k(i,j);
    end
end
end
```

Published with MATLAB® R2019b



ANEXO Nº04
Código de los ejemplos aplicativos. Datos de Usuario

Estructura Articulada

```
%{
Propósito:
    Analizar un sistema compuesto de Estructuras Articuladas con el FEM.
    Código para el Ingreso de Datos por el Usuario.
Descripción de Datos de Ingreso:
    - Coordenadas de las Barras.
    - Cargas Impuestas en los diversos Nodos.
    - Condiciones de Borde en el Sistema.
Variables de Ingreso:
    Especificadas en el código.
Diseñador:
    Ing.Jorge Enrique Alvarez Ruffrán
Fecha de Ultima Actualización:
    Noviembre 2019
%}

%%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%%

clear           % Limpieza del Espacio de Trabajo
clc             % Limpieza de la ventana de Comandos
close all       % Cierre de todas las figuras Existentes
format short e  % Formato Numérico a Trabajar

%%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%%

% Ingreso de Coordenadas de los Nodos. (Geométria)
% Coordenadas X/Y por Nodo
geom = [0. 0.; ...
        3. 0.; ...
        3. 4.; ...
        6. 4.];

% Ingreso de Datos de Conectividad entre Nodos. (Conectividad)
% Conectividad Nodo Inicial/ Nodo Final por Elemento
conect = [1 2;...
          1 3;...
          2 3;...
          2 4;...
          3 4];

% Propiedades Geométricas del Material por Barra (Propiedades)
% Propiedades: Modulo de Elasticidad/ Área por Elemento
prop = [2*10^7 0.001;...
        2*10^7 0.001;...
        2*10^7 0.001;...
        2*10^7 0.001;...
        2*10^7 0.001];
```

```
%   Información sobre el Sistema (Automático)
nne = 2;        % Numero de Nodos por Elemento
ndgdl = 2;      % Numero de GDLs por Nodo
[nnd,~] = size(geom);       % Número de Nodos en el Sistema
[nel,~] = size(conect);     % Número de Elementos
elgdl = nne*ndgdl;          % Número de GDLs por Elemento
sigdl = nnd*ndgdl;          % Número de GDLs en el Sistema

%   Dando Numeración a todos los GDLs en el Sistema
Numnd = ones(nnd, ndgdl);      % Creación de la Matriz con GDLs Libres (1)
n=0;
for i=1:nnd     % Bucle para asignar la numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

%   Condiciones de Frontera por Nodo
nf = ones(nnd, ndgdl);      % Creación de la Matriz con GDLs Libres (1)
n=0;
for i=1:nnd     % Bucle para asignar la numeración
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
nf(1,1) = 0;                % Aplicando Restricciones en los GDLs
nf(1,2) = 0;
nf(2,2) = 0;

%   Obteniendo los GDLs Restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle para extraer los GDLs restringidos
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end

%      Cargas/Fuerzas Aplicadas al Sistema
Carg = zeros(nnd, ndgdl);   % Creación de la Matriz de Cargas Vacías (0)
Carg(4,:)=[0. 3.];          % Aplicando Cargas a los Nodos

%   Limpieza de Variables Innecesarias
clear c n i j

%   Ejecución de la función para el Procesamiento de Datos
```

```
disp('Procesando Datos');
[desp,Fuerza] =
Barra_Elemento_Programacion(nnd,nel,conect,geom,prop,elgdl,Carg,sigdl,Numnd,ndgdl,ResGDL
);
disp('Procesamiento de Datos Terminado');

%   Ordenamiento de Resultados}
disp('Resultados');
for i=1:nnd
    for j=1:ndgdl
        nodo_desp(i,j) = 0;
        if nf(i,j)~= 0
            nodo_desp(i,j) = desp(nf(i,j));
        end
    end
end
display(nodo_desp);

for i=1:nel
    if Fuerza(i) > 0
        fprintf(' %g, %9.2f, %s\n',i, Fuerza(i), 'Tension');
    else
        fprintf(' %g, %9.2f, %s\n',i, Fuerza(i), 'Compression');
    end
end

%   Limpieza de Variables Innecesarias
clear Carg conect elgdl geom i j ResGDL prop n ndgdl nel nf nnd nne desp...
    Numnd sigdl
```

Published with MATLAB® R2019b

Losa Empotrada con Carga Central

```
%{
Propósito:
    Analizar un sistema compuesto de elementos cuadriláteros uniforme tipo
    Plate mediante el método de Kirchhoff y en base a 4 nodos por elemento.
    Código para el Ingreso de Datos por el Usuario.
Descripción de Datos de Ingreso:
    - Coordenadas de los Nodos en el elemento Plate.
    - Cargas Impuestas en los diversos Nodos.
    - Condiciones de Borde en el Sistema.
Variables de Ingreso:
    Especificadas en el código.
Diseñador:
    Ing.Jorge Enrique Alvarez Ruffrán
Fecha de Ultima Actualización:
    Junio 2019
%}

%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%
```

```
clear           % Limpieza del Espacio de Trabajo
clc             % Limpieza de la ventana de Comandos
close all       % Cierre de todas las figuras Existentes
format short e  % Formato Numérico a Trabajar

%%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%%

%   Ingreso de Coordenadas de los Nodos. (Geométria)
geom = [0.0  0.0;...    % Coordenadas en X,Y del Elem. Plate.
        2.0  0.0;...
        2.0  2.0;...
        0.0  2.0];

%   Ingreso de Datos de Conectividad entre Nodos. (Conectividad)
conect = [1 2 3 4];  % Conectividad de Nodos en el Extremo del elemento

%   Propiedades del Material (Elasticidad/Poisson/Espesor)
E=2*10^10;      % Modulo de elasticidad del material
vu=0.3;         % Coeficiente de Poisson del material
Esp=0.10;       % Espesor del elemento plate

%   Información sobre el Sistema (Automático)
nne = 4;                    % Número de nodos por elemento
ndgdl = 3;                  % Número de GDLs por nodo
[nnd,~] = size(geom);       % Número de nodos en el sistema
[nel,~] = size(conect);     % Número de elementos
elgdl = nne*ndgdl;          % Número de GDLs por elemento
sigdl = nnd*ndgdl;          % Número de GDLs en el sistema
npgf = 3;                   % Número de puntos de Gauss a flexión

%   Dando Numeración a todos los GDLs en el Sistema
Numnd = ones(nnd, ndgdl);      % Creación de la Matriz con GDLs Libres (1)
n=0;
for i=1:nnd     % Bucle para asignar la numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

%   Condiciones de Frontera por Nodo
nf = ones(nnd, ndgdl);      % Creación de la Matriz con GDLs Libres (1)
n=0;
for i=1:nnd     % Bucle para asignar la numeración
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
```

```
nf(1,1) = 0; nf(1,3)=0;
nf(1,2)=0;
nf(2,1) = 0; nf(2,3)=0;
nf(2,2)=0;
nf(3,3) = 0; %nf(3,3)=0;
nf(3,2)=0;
nf(4,1) = 0; nf(4,3)=0;
nf(4,2)=0;

%   Obteniendo los GDLs Restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle para obtener los GDLs restringidos
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end

%      Cargas/Fuerzas Aplicadas al Sistema
Carg = zeros(nnd, ndgdl);   % Creación de la Matriz de Cargas Vacío (0)
Carg(3,1) = 5;              % Aplicando Cargas a los Nodos

%   Ejecución de la función para el Procesamiento de Datos
disp('Procesando Datos');
[desp,Element_Forces] =
Placa_Kirchhoff_Programacion(nf,nne,nnd,nel,conect,geom,E,vu,Esp,elgdl,Carg,sigdl,Numnd,
ndgdl,ResGDL);

%%%%%%%%%%%%%%%%%% ------- MOSTRANDO GRÁFICOS ------- %%%%%%%%%%%%%%%%%%

% Desplazamientos
UZ=desp(:,1);
UX=desp(:,2);
UY=desp(:,3);

figure
cmin = min(UZ);
cmax = max(UZ);
caxis([cmin cmax]);
patch('Faces',conect, 'Vertices', geom, 'FaceVertexCData',UZ,...
'Facecolor','interp','Marker','.');
colorbar;
title('Desplazamiento en Z (UZ)')
xlabel('B')
ylabel('H')

% Momentos
MX=Element_Forces(:,1);
MY=Element_Forces(:,2);
MXY=Element_Forces(:,3);

figure
```

```
cmin = min(MX);
cmax = max(MX);
caxis([cmin cmax]);
patch('Faces',conect, 'Vertices', geom, 'FaceVertexCData',MX,...
'Facecolor','interp','Marker','.');
colorbar;
title('Momento en X (MX)')
xlabel('B')
ylabel('H')

figure
cmin = min(MY);
cmax = max(MY);
caxis([cmin cmax]);
patch('Faces',conect, 'Vertices', geom, 'FaceVertexCData',MY,...
'Facecolor','interp','Marker','.');
colorbar;
title('Momento en Y (MX)')
xlabel('B')
ylabel('H')

disp('Procesamiento de Datos Terminado');

%   Limpieza de Variables Innecesarias
clear c n i j a b cmax cmin DR x y xcoord ycoord zcoord vu C coord E eld delta...
    F g k k1 k2 k3 k4 kg KG MCF MD m l intx iel nel nf nnd nne ndgdl node_disp...
    npgf Q ResGDL sigdl Numnd Moment geom elgdl Element_Forces Esp
```

Published with MATLAB® R2019b

Viga Multidimensional - 1D

```
%{
Propósito:
    Analizar un sistema compuesto de una viga con la teoria de
    Euler-Bernoulli mediante el FEM.
    Código para el ingreso de datos por el usuario.
Variables de Ingreso:
    Especificadas en el código.
Diseño: Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Setiembre 2019
%}

%%%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%%%

clear           % Limpieza del espacio de trabajo
clc             % Limpieza de la ventana de comandos
close all       % Cierre de todas las figuras existentes
format short e  % Formato numérico a trabajar

%%%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%%%
```

```
%   Ingreso de Coordenadas de los nodos. (Geométria)
geom = [0.; ...     % Coordenadas en X de la viga
        1.5;...
        3.0];

%   Ingreso de datos de conectividad entre Nodos. (Conectividad)
conect = [1 2;2 3] ; % Conectividad de nodos en el extremo del elemento

%   Propiedades del sistema (Elasticidad/Inercia)
prop = [2.535*10^9 0.0072;...
        2.535*10^9 0.0072];

%   Información sobre el sistema (Automático)
nne = 2;        % Número de nodos por elemento
ndgdl = 2;      % Número de GDLs por nodo
[nnd] = size(geom,1);       % Número de nodos en el sistema
[nel] = size(conect,1);     % Número de elementos
elgdl = nne*ndgdl;          % Número de GDLs por elemento
sigdl = nnd*ndgdl;          % Número de GDLs en el sistema

%   Dando numeración a todos los GDLs en el sistema
Numnd = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para asignar de numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

%   Condiciones de frontera por nodo
nf = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para asignar de numeración
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end
nf(1,1) = 0;    % Aplicando restricciones en los GDLs
nf(3,1) = 0;

%   Obteniendo los GDLs Restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle asignar de numeración
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
```

```
        end
    end
end

%   Aplicación de rotulas al sistema
Rot = ones(nel, 2);

%      Fuerzas aplicadas al sistema
CargNod = zeros(nnd, 2);    % Carga en y/o momento aplicado directamente

CargElem = zeros(nnd, elgdl);   % Creación de la matriz de cargas vacias (0)
CargElem(1,:)=[ -450 -112.5 -450 112.5];         % Aplicando cargas a los nodos
CargElem(2,:)=[ -450 -112.5 -450 112.5];

%   Limpieza de variables innecesarias
clear c n i j

%   Ejecutando la función para el procesamiento de datos
disp('Procesando Datos');
[desp,Fuerza] =
Viga_EB_Programacion(nnd,nel,conect,geom,prop,elgdl,CargNod,CargElem,sigdl,Numnd,ndgdl,R
esGDL,Rot);

%   Ordenamiento de resultados
disp('Resultados');

for i=1:nnd
    for j=1:ndgdl
        nodo_desp(i,j) = 0;
        if nf(i,j)~= 0
            nodo_desp(i,j) = desp(nf(i,j));
        end
    end
end
display(nodo_desp);

for i=1:nel
    fprintf(' %g, %9.2f, %9.2f, %9.2f, %9.2f\n',i, ...
    Fuerza(i,1),Fuerza(i,2),Fuerza(i,3),Fuerza(i,4));
end

disp('Procesamiento de Datos Terminado');

%   Limpieza de variables innecesarias
clear Carg conect elgdl geom i j ResGDL prop n ndgdl nel nf nnd nne desp...
    Numnd sigdl Rot
```

Published with MATLAB® R2019b

Viga Multidimensional - 2D

```
%{
Propósito:
    Analizar un sistema compuesto de elementos bidimensioanles bajo tensión
    Plana. Elementos con 8 Nodos, bajo integración numérica por Gauss-Legendre.
    Código para el ingreso de datos por el usuario.
Variables de Ingreso:
    Especificadas en el código.
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Setiembre 2019
%}

%%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%%

clear           % Limpieza del espacio de trabajo
clc             % Limpieza de la ventana de comandos
close all       % Cierre de todas las figuras existentes
format short e  % Formato numérico a trabajar

%%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%%

%   Ingreso de coordenadas de los nodos. (Geométria)
dataSis='VIGA2D_AltoMesh_Coord.txt';
geom = importdata(dataSis);
geom=geom(:,2:4);
[a,b]=size(geom);
geom2=zeros(a,2);
geom2(:,1)=geom(:,1);
geom2(:,2)=geom(:,3);
geom=geom2;

%   Ingreso de datos de conectividad entre nodos. (Conectividad)
dataSis='VIGA2D_AltoMesh_Conec.txt';
conect = importdata(dataSis);
conect=conect(:,2:9);

[nnd]=size(conect,1);

conect2=zeros(nnd,8);
conect2(:,1)=conect(:,1);
conect2(:,2)=conect(:,5);
conect2(:,3)=conect(:,2);
conect2(:,4)=conect(:,6);
conect2(:,5)=conect(:,3);
conect2(:,6)=conect(:,7);
conect2(:,7)=conect(:,4);
conect2(:,8)=conect(:,8);
conect=conect2;


%   Propiedades del sistema
```

```
E = 2.535*10^9;          % Módulo de elasticidad del sistema
vu = 0.20;               % Coeficiente de Poisson
thick = 0.40;            % Espesor del sistema

%   Información sobre el sistema (Automático)
nne = 8;             % Número de nodos por elemento
ndgdl = 2;           % Número de GDLs por nodo
ngp = 2;             % Número de puntos de Gauss
[nnd] = size(geom,1);        % Número de nodos en el sistema
[nel] = size(conect,1);      % Número de elementos
elgdl = nne*ndgdl;           % Número de GDLs por elemento
sigdl = nnd*ndgdl;           % Número de GDLs en el sistema

%   Dando numeración a todos los GDLs en el sistema
Numnd = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle asginar numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

%   Condiciones de frontera por nodo
nf = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle asginar numeración
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end

i=[367, 427];
[a]=size(i,1);
for ios=1:a
    nf(i,1)=0;
    nf(i,2)=0;
end

%   Obteniendo los GDLs restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle asginar numeración
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end
```

```
%      Fuerzas aplicadas al sistema
CargNod = zeros(nnd, 2);    % Carga en y/o momento aplicados directamente

%   Bordes
P=-14.8760;
i=[734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749,...
    750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765,...
    766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781,...
    782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 2186, 2189, 2191, 2193,
2195,...
    2197, 2199, 2201, 2203, 2205, 2207, 2209, 2211, 2213, 2215, 2217, 2219, 2221, 2223,
2225, 2227,...
    2229, 2231, 2233, 2235, 2237, 2239, 2241, 2243, 2245, 2247, 2249, 2251, 2253, 2255,
2257, 2259,...
    2261, 2263, 2265, 2267, 2269, 2271, 2273, 2275, 2277, 2279, 2281, 2283, 2285, 2287,
2289, 2291,...
    2293, 2295, 2297, 2299, 2301, 2303, 2305, 733, 793];

[a,b]=size(i);
for ios=1:a
    CargNod(i,2)=P;
end

%   Limpieza de variables innecesarias
clear c n i j

%   Ejecutando la función para el procesamiento de datos
disp('Procesando Datos');
[delta,SIGMA] =
Bi_TenPlana_Gauss_Programacion(nf,ngp,nnd,nel,conect,geom,E,vu,thick,elgdl,CargNod,sigdl
,Numnd,ndgdl,ResGDL,nne);

%%%%%%%%% ------- DESPLAZAMIENTOS ORDENADOS ------- %%%%%%%%%

for i=1:nnd
    for j=1:ndgdl
        node_disp(i,j) = 0;
        if nf(i,j)~= 0
            node_disp(i,j) = delta(Numnd(i,j)) ;
        end
    end
end
desp=node_disp;

%%%%%%%%% ------- ESFUERZOS ORDENADOS ------- %%%%%%%%%

for k = 1:nnd
    sigx = 0. ;sigy = 0.; tau = 0.;
    ne = 0;
    for iel = 1:nel
        for jel=1:nne
            if conect(iel,jel) == k
                ne=ne+1;
```

```
                sigx = sigx+SIGMA(iel,1);
                sigy = sigy + SIGMA(iel,2);
                tau = tau + SIGMA(iel,3);
            end
        end
    end
    ZX(k,1) = sigx/ne;
    ZY(k,1) = sigy/ne;
    ZT(k,1)=tau/ne;
    Z1(k,1)= ((sigx+sigy)/2 + sqrt(((sigx+sigy)/2)^2 +tau^2))/ne;
    Z2(k,1)= ((sigx+sigy)/2 - sqrt(((sigx+sigy)/2)^2 +tau^2))/ne;
end

%%%%%%%%%%%%%%%%% ------- MOSTRANDO GRÁFICOS ------- %%%%%%%%%%%%%%%%%

% ----- DESPLAZAMIENTO EN Z ----- %
figure('Renderer', 'painters', 'Position', [3 388 465 294]);
U2 = desp(:,2);
cmin = min(U2);
cmax = max(U2);
caxis([cmin cmax])
patch('Faces', conect, 'Vertices', geom, 'FaceVertexCData',U2,...
'Facecolor','interp','Marker','.');
colormap(jet(12))
colorbar
title('Desplazamiento Uz(m)');
xlabel('L = 3m')
ylabel('H = 0.60m')
xlim([0 3])
ylim([0 0.60])

% ----- DESPLAZAMIENTO EN X ----- %

figure('Renderer', 'painters', 'Position', [3 44 465 260])
U1 = desp(:,1);
cmin = min(U1);
cmax = max(U1);
caxis([cmin cmax])
patch('Faces', conect, 'Vertices', geom, 'FaceVertexCData',U1,...
'Facecolor','interp','Marker','.');
colormap(jet(12))
colorbar
title('Desplazamiento Ux(m)');
xlabel('L = 3m')
ylabel('H = 0.60m')
xlim([0 3])
ylim([0 0.60])

% ----- ESFUERZO EN SXX ----- %

figure('Renderer', 'painters', 'Position', [470 388 421 294])
cmin = min(ZX);
cmax = max(ZX);
caxis([cmin cmax]);
```

```
patch('Faces', conect, 'Vertices', geom, 'FaceVertexCData',ZX,...
'Facecolor','interp','Marker','.');
colormap(jet(12))
colorbar
title('Esfuerzo SXX(kgf/m^2)');
xlabel('L = 3m')
ylabel('H = 0.60m')
xlim([0 3])
ylim([0 0.60])

% ----- ESFUERZO EN SYY ----- %

figure('Renderer', 'painters', 'Position', [470 44 421 260])
cmin = min(ZY);
cmax = max(ZY);
caxis([cmin cmax]);
patch('Faces', conect, 'Vertices', geom, 'FaceVertexCData',ZY,...
'Facecolor','interp','Marker','.');
colormap(jet(12))
colorbar
title('Esfuerzo SYY(kgf/m^2)');
xlabel('L = 3m')
ylabel('H = 0.60m')
xlim([0 3])
ylim([0 0.60])

% ----- ESFUERZO EN SZZ ----- %

figure('Renderer', 'painters', 'Position', [892 388 471 294])
cmin = min(ZT);
cmax = max(ZT);
caxis([cmin cmax]);
patch('Faces', conect, 'Vertices', geom, 'FaceVertexCData',ZT,...
'Facecolor','interp','Marker','.');
colormap(jet(12))
colorbar
title('Esfuerzo SZZ(kgf/m^2)');
xlabel('L = 3m')
ylabel('H = 0.60m')
xlim([0 3])
ylim([0 0.60])

disp('Procesamiento de Datos Terminado');

%   Limpieza de variables innecesarias
clear c n i j a b cmax cmin DR x y xcoord ycoord zcoord vu C coord E eld delta...
    F g k k1 k2 k3 k4 kg KG MCF MD m l intx iel nel nf nnd nne ndgdl node_disp...
    npgf Q ResGDL sigdl Numnd Moment geom elgdl Element_Forces Esp
```

Published with MATLAB® R2019b

Viga Multidimensional - 3D

```
%{
Propósito:
    Analizar un sistema compuesto de un elemento tridimensional con el FEM.
    Código para el ingreso de datos por el usuario.
Variables de ingreso:
    Especificadas en el código.
Diseño: Ing.Jorge Enrique Alvarez Ruffrán
Fecha de última actualización:
    Junio 2019
%}

%%%%%%%%%%%%%%%%% ------- COMANDOS INICIALES ------- %%%%%%%%%%%%%%%%%

clear           % Limpieza del espacio de trabajo
clc             % Limpieza de la ventana de comandos
close all       % Cierre de todas las figuras existentes
format long     % Formato numérico a trabajar

%%%%%%%%%%%%%%%%% ------- INGRESO DE DATOS ------- %%%%%%%%%%%%%%%%%

%   Ingreso de coordenadas de los nodos. (Geométria)
dataSis='VIGA3D_AltoMesh_Coord.txt';
geom = importdata(dataSis);
geom=geom(:,2:4);

%   Ingreso de datos de conectividad entre nodos. (Conectividad)
dataSis='VIGA3D_AltoMesh_Conec.txt';
conect = importdata(dataSis);
conect=conect(:,2:9);

%   Propiedades del sistema
nne = 8;                    %   Número de nodos por elemento
ndgdl =3;                   %   Número de grados de libertad por nodo
[nnd] = size(geom,1);       %   Número de nodos en el sistema
[nel] = size(conect,1);     %   Número de elementos
elgdl = nne*ndgdl;          %   Número de GDLs por elemento
sigdl = nnd*ndgdl;          %   Número de GDLs en el sistema
nglx=2; ngly=2; nglz=2;     %   Número de puntos de Gauss a flexión
dim = 3;                    %   Dimensión del elemento a analizar
E=2.535*10^9;               %   Módulo de elasticidad del sistema
vu=0.20;                    %   Módulo de Poisson

%   Cambio de los ejes globales si es necesario (Eje X,Y,Z)
geom2=zeros(nnd,3);
geom2(:,1)=geom(:,3);
geom2(:,2)=geom(:,1);
geom2(:,3)=geom(:,2);
geom=geom2;

%   Dando numeración a todos los GDLs en el sistema
Numnd = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
```

```
n=0;
for i=1:nnd     % Bucle para la asignación de numeración
    for j=1:ndgdl
        if Numnd(i,j) ~= 0
            n=n+1;
            Numnd(i,j)=n;
        end
    end
end

%   Condiciones de frontera por nodo
nf = ones(nnd, ndgdl);      % Creación de la matriz con GDLs libres (1)
n=0;
for i=1:nnd     % Bucle para asignar numeración
    for j=1:ndgdl
        if nf(i,j) ~= 0
            n=n+1;
            nf(i,j)=n;
        end
    end
end

i=[5, 6, 7, 8, 11, 12, 15, 16, 26, 27]; % Vector con los nodos con GDLs restringidos

[a]=size(i,1);  % Asignación de los GDLs restringidos por nodo
for ios=1:a
    nf(i,1)=0;
    nf(i,2)=0;
    nf(i,3)=0;
end

%   Obteniendo los GDLs restringidos (ResGDL)
n=0;
for i=1:nnd     % Bucle para asignar numeración
    for j=1:ndgdl
        if nf(i,j) == 0
            n=n+1;
            ResGDL(n)=Numnd(i,j);
        end
    end
end

[a,b]=size(ResGDL);
bcval=zeros(1,b);

%       Fuerzas aplicadas al sistema
CargNod = zeros(nnd,3);   % Carga en y/o Momento aplicado directamente

%   Esquinas
P=-2.8125;
i=[39, 40, 47, 48];

[a,b]=size(i);
for ios=1:a
```

```
    CargNod(i,3)=P;
end

%   Bordes
P=-5.625;
i=[37, 38, 41, 42, 43, 44, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827,...
   828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843,...
    844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 985, 986, 987,...
    988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002,
1003,...
    1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019,...
    1020, 1021, 1022, 1023  ];

[a,b]=size(i);
for ios=1:a
    CargNod(i,3)=P;
end

%   Centrales
P=-11.25;
i=[779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794,...
    795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810,...
    811, 812, 813, 814, 815, 816, 817, 861, 862, 863, 864, 865, 866, 867, 868, 869,...
    870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885,...
    886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 902, 903,...
    904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919,...
    920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935,...
    936, 937, 938, 939, 940];

[a,b]=size(i);
for ios=1:a
    CargNod(i,3)=P;
end

%   Limpieza de variables innecesarias
clear c n i j

%   Ejecutando la función para el procesamiento de datos
disp('Procesando Datos');
[delta,SIGMA,stress,conectesf] =
Tri_Elemento_Programacion(nf,nglx,ngly,nglz,nnd,nel,conect,geom,E,vu,elgdl,CargNod,sigdl
,Numnd,ndgdl,ResGDL,nne);

%%%%%%%%%% ------- ORDENANDO DESPLAZAMIENTOS ------- %%%%%%%%%%
for i=1: nnd
    if nf(i,1) == 0
        x_disp =0.;
    else
        x_disp = delta(Numnd(i,1));
    end
    %
    if nf(i,2) == 0 %
        y_disp = 0.; %
```

```
    else
        y_disp = delta(Numnd(i,2));
    end

    if nf(i,3) == 0 %
        z_disp = 0.; %
    else
        z_disp = delta(Numnd(i,3));
    end
    DISP(i,:) = [x_disp y_disp z_disp];
end

%%%%%%%%%% ------- ORDENANDO ESFUERZOS ------- %%%%%%%%%%
% 1 punto de Integración
Element_Bending=SIGMA(:,1:3);
Element_Shear=SIGMA(:,4:6);
for k = 1:nnd
    mx = 0. ; my = 0.; mxy = 0.; qx = 0.; qy = 0.; qxy = 0.;
    ne = 0; p=0;
    for iel = 1:nel
        for jel=1:nne
            if conect(iel,jel) == k
                ne=ne+1;
                mx = mx + Element_Bending(iel,1);
                my = my + Element_Bending(iel,2);
                mxy = mxy + Element_Bending(iel,3);
                qx = qx + Element_Shear(iel,1);
                qy = qy + Element_Shear(iel,2);
                qxy = qxy + Element_Shear(iel,3);
            end
        end
    end
    ZX(k,1) = mx/ne;
    ZY(k,1) = my/ne;
    ZZ(k,1) = mxy/ne;
    Z1(k,1) = qx/ne;
    Z2(k,1) = qy/ne;
    Z3(k,1) = qxy/ne;
end

% 4 puntos de Integración
Element_Bending=stress(:,1:3);
Element_Shear=stress(:,4:6);
for k = 1:nnd
    mx = 0. ; my = 0.; mxy = 0.; qx = 0.; qy = 0.; qxy = 0.;
    ne = 0; p=0;
    for iel = 1:nel
        for jel=1:nne
            if conect(iel,jel) == k
                ne=ne+1;
                p=iel*8-8+jel;
                mx = mx + Element_Bending(p,1);
                my = my + Element_Bending(p,2);
                mxy = mxy + Element_Bending(p,3);
```

```
                qx = qx + Element_Shear(p,1);
                qy = qy + Element_Shear(p,2);
                qxy = qxy + Element_Shear(p,3);
            end
        end
    end
    ZX2(k,1) = mx/ne;
    ZY2(k,1) = my/ne;
    ZZ2(k,1) = mxy/ne;
    Z12(k,1) = qx/ne;
    Z22(k,1) = qy/ne;
    Z32(k,1) = qxy/ne;
end

%%%%%%%%% ------- GRAFICA DESPLAZAMIENTO ------- %%%%%%%%%
disp('Mostrando Graficos');

% ----- DESPLAZAMIENTO EN Z ----- %
figure('Renderer', 'painters', 'Position', [3 388 465 294]);
view(-24.26299,40.80);
hold on
U = DISP(:,3);
cmin = min(U);
cmax = max(U);
caxis([cmin cmax]);

p=1;
for i=1: nel
        s1=[conect(i,1) conect(i,2) conect(i,3) conect(i,4)];
        s2=[conect(i,5) conect(i,6) conect(i,7) conect(i,8)];
        s3=[conect(i,1) conect(i,5) conect(i,8) conect(i,4)];
        s4=[conect(i,2) conect(i,6) conect(i,7) conect(i,3)];
        s5=[conect(i,5) conect(i,6) conect(i,2) conect(i,1)];
        s6=[conect(i,8) conect(i,7) conect(i,3) conect(i,4)];
    connecnew(p,1:4)=s1;
    connecnew(p+1,1:4)=s2;
    connecnew(p+2,1:4)=s3;
    connecnew(p+3,1:4)=s4;
    connecnew(p+4,1:4)=s5;
    connecnew(p+5,1:4)=s6;
    p=p+6;
end

patch('Faces', connecnew, 'Vertices', geom, 'FaceVertexCData',U,...
         'Facecolor','interp','Marker','.');
colormap(flipud(jet(12)))
colorbar
title('Desplazamiento en z (UZ)');

%xlim([0 3])
ylim([-2 2])
zlim([-1 1])

% ----- DESPLAZAMIENTO EN X ----- %
```

```
figure('Renderer', 'painters', 'Position', [3 44 465 260])
view(-24.26299,40.80);
hold on
U = DISP(:,1);
cmin = min(U);
cmax = max(U);
caxis([cmin cmax]);

p=1;
for i=1: nel
        s1=[conect(i,1) conect(i,2) conect(i,3) conect(i,4)];
        s2=[conect(i,5) conect(i,6) conect(i,7) conect(i,8)];
        s3=[conect(i,1) conect(i,5) conect(i,8) conect(i,4)];
        s4=[conect(i,2) conect(i,6) conect(i,7) conect(i,3)];
        s5=[conect(i,5) conect(i,6) conect(i,2) conect(i,1)];
        s6=[conect(i,8) conect(i,7) conect(i,3) conect(i,4)];
    connecnew(p,1:4)=s1;
    connecnew(p+1,1:4)=s2;
    connecnew(p+2,1:4)=s3;
    connecnew(p+3,1:4)=s4;
    connecnew(p+4,1:4)=s5;
    connecnew(p+5,1:4)=s6;
    p=p+6;
end

patch('Faces', connecnew, 'Vertices', geom, 'FaceVertexCData',U,...
         'Facecolor','interp','Marker','.');
colormap(flipud(jet(12)))
colorbar
title('Desplazamiento en x (UX)');

%xlim([0 3])
ylim([-2 2])
zlim([-1 1])

% ----- DESPLAZAMIENTO EN Y ----- %
figure('Renderer', 'painters', 'Position', [470 388 421 294])
view(-24.26299,40.80);
hold on
U = DISP(:,2);
cmin = min(U);
cmax = max(U);
caxis([cmin cmax]);

p=1;
for i=1: nel
        s1=[conect(i,1) conect(i,2) conect(i,3) conect(i,4)];
        s2=[conect(i,5) conect(i,6) conect(i,7) conect(i,8)];
        s3=[conect(i,1) conect(i,5) conect(i,8) conect(i,4)];
        s4=[conect(i,2) conect(i,6) conect(i,7) conect(i,3)];
        s5=[conect(i,5) conect(i,6) conect(i,2) conect(i,1)];
        s6=[conect(i,8) conect(i,7) conect(i,3) conect(i,4)];
    connecnew(p,1:4)=s1;
    connecnew(p+1,1:4)=s2;
```

```
    connecnew(p+2,1:4)=s3;
    connecnew(p+3,1:4)=s4;
    connecnew(p+4,1:4)=s5;
    connecnew(p+5,1:4)=s6;
    p=p+6;
end

patch('Faces', connecnew, 'Vertices', geom, 'FaceVertexCData',U,...
         'Facecolor','interp','Marker','.');
colormap(flipud(jet(12)))
colorbar
title('Desplazamiento en x (UX)');

%xlim([0 3])
ylim([-2 2])
zlim([-1 1])

% %%%%%%%%%% ------- GRÁFICA ESFUERZOS, EXTREMO DEL ELEMENTO ------- %%%%%%%%%%

% ----- ESFUERZO SXX2 ----- %
figure('Renderer', 'painters', 'Position', [470 44 421 260])
view(-24.26299,40.80);
cmin = min(ZX2);
cmax = max(ZX2);
caxis([cmin cmax]);

p=1;
for i=1: nel
        s1=[conectesf(i,1) conectesf(i,2) conectesf(i,3) conectesf(i,4)];
        s2=[conectesf(i,5) conectesf(i,6) conectesf(i,7) conectesf(i,8)];
        s3=[conectesf(i,1) conectesf(i,5) conectesf(i,8) conectesf(i,4)];
        s4=[conectesf(i,2) conectesf(i,6) conectesf(i,7) conectesf(i,3)];
        s5=[conectesf(i,5) conectesf(i,6) conectesf(i,2) conectesf(i,1)];
        s6=[conectesf(i,8) conectesf(i,7) conectesf(i,3) conectesf(i,4)];
    connecnew2(p,1:4)=s1;
    connecnew2(p+1,1:4)=s2;
    connecnew2(p+2,1:4)=s3;
    connecnew2(p+3,1:4)=s4;
    connecnew2(p+4,1:4)=s5;
    connecnew2(p+5,1:4)=s6;
    p=p+6;
end

patch('Faces',connecnew, 'Vertices', geom, 'FaceVertexCData',ZX2,...
'Facecolor','interp','Marker','.');
colormap(flipud(jet(12)))
colorbar

title('Esfuerzo SXX (kgf/m^2)')
ylim([-2 2])
zlim([-1 1])

% ----- ESFUERZO SYY2 ----- %
figure('Renderer', 'painters', 'Position', [892 388 471 294])
```

```
view(-24.26299,40.80);
cmin = min(ZY2);
cmax = max(ZY2);
caxis([cmin cmax]);

p=1;
for i=1: nel
        s1=[conectesf(i,1) conectesf(i,2) conectesf(i,3) conectesf(i,4)];
        s2=[conectesf(i,5) conectesf(i,6) conectesf(i,7) conectesf(i,8)];
        s3=[conectesf(i,1) conectesf(i,5) conectesf(i,8) conectesf(i,4)];
        s4=[conectesf(i,2) conectesf(i,6) conectesf(i,7) conectesf(i,3)];
        s5=[conectesf(i,5) conectesf(i,6) conectesf(i,2) conectesf(i,1)];
        s6=[conectesf(i,8) conectesf(i,7) conectesf(i,3) conectesf(i,4)];
    connecnew2(p,1:4)=s1;
    connecnew2(p+1,1:4)=s2;
    connecnew2(p+2,1:4)=s3;
    connecnew2(p+3,1:4)=s4;
    connecnew2(p+4,1:4)=s5;
    connecnew2(p+5,1:4)=s6;
    p=p+6;
end

patch('Faces',connecnew, 'Vertices', geom, 'FaceVertexCData',ZY2,...
'Facecolor','interp','Marker','.');
colormap(flipud(jet(12)))
colorbar

title('Esfuerzo SYY (kgf/m^2)')
ylim([-2 2])
zlim([-1 1])

% ----- ESFUERZO SZZ2 ----- %
f=figure('Renderer', 'painters', 'Position', [892 44 471 260])
view(-24.26299,40.80);
cmin = min(ZZ2);
cmax = max(ZZ2);
caxis([cmin cmax]);

p=1;
for i=1: nel
        s1=[conectesf(i,1) conectesf(i,2) conectesf(i,3) conectesf(i,4)];
        s2=[conectesf(i,5) conectesf(i,6) conectesf(i,7) conectesf(i,8)];
        s3=[conectesf(i,1) conectesf(i,5) conectesf(i,8) conectesf(i,4)];
        s4=[conectesf(i,2) conectesf(i,6) conectesf(i,7) conectesf(i,3)];
        s5=[conectesf(i,5) conectesf(i,6) conectesf(i,2) conectesf(i,1)];
        s6=[conectesf(i,8) conectesf(i,7) conectesf(i,3) conectesf(i,4)];
    connecnew2(p,1:4)=s1;
    connecnew2(p+1,1:4)=s2;
    connecnew2(p+2,1:4)=s3;
    connecnew2(p+3,1:4)=s4;
    connecnew2(p+4,1:4)=s5;
    connecnew2(p+5,1:4)=s6;
    p=p+6;
end
```

```
patch('Faces',connecnew, 'Vertices', geom, 'FaceVertexCData',zz2,...
'Facecolor','interp','Marker','.');
colormap(flipud(jet(12)))
colorbar

title('Esfuerzo SZZ (kgf/m^2)')
ylim([-2 2])
zlim([-1 1])

disp('Procesamiento de Datos Terminado');
```

Published with MATLAB® R2019b

ANEXO Nº05
Data sobre coordenadas de los nodos y conectividad de diversos ejemplos.

Debido a la extensión del código este se subirá a Google Drive y estará disponible desde el siguiente enlace:

https://drive.google.com/drive/folders/1UYQApcj5MIWtmTOmp6Gz_PEK6HfpsOuh?usp=sharing

ANEXO Nº06
Funciones extras, incorporadas a los códigos.

Deriv3DGlobal.m

```
%{
Propósito:
    Obtener los derivadas de las funciones de forma para un elemento
    tridimensional en coordenadas globales.
Descripción de Datos de Ingreso:
    - nglx,ngly,nglz = Número de puntos de integración a tratar en cada uno
                       de los ejes cartesianos.
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Julio 2018
%}

function [dhdx,dhdy,dhdz]=Deriv3DGlobal(nnel,dhdr,dhds,dhdt,invjacob)
% Bucle para asignar la data en diferentes casillas de las variables de
% funciones de forma globales.
for i=1:nnel
    dhdx(i)=invjacob(1,1)*dhdr(i)+invjacob(1,2)*dhds(i)+invjacob(1,3)*dhdt(i);
    dhdy(i)=invjacob(2,1)*dhdr(i)+invjacob(2,2)*dhds(i)+invjacob(2,3)*dhdt(i);
    dhdz(i)=invjacob(3,1)*dhdr(i)+invjacob(3,2)*dhds(i)+invjacob(3,3)*dhdt(i);
end
end
```

Published with MATLAB® R2019b

Elem2D_4_nodos.m

```
%{
Propósito:
    Obtener las funciones de forma y sus derivadas en función a coordenadas
    naturales, para un elemento bidimensional con 4 nodos por elemento.
Descripción de datos de ingreso:
    - rvalue, svalue = Coordenadas del punto de integracion trabajado.
Orden de los Nodos trabajados:
    1er Nodo (-1,-1)
```

```
    2do Nodo (1,-1)
    3er Nodo (1,1)
    4to Nodo (-1,1)
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Julio 2018
%}

function [shapeq4,dhdrq4,dhdsq4]=Elem2D_4nodos(rvalue,svalue)

% Ingresando data de las funciones de forma
 shapeq4(1)=0.25*(1-rvalue)*(1-svalue);
 shapeq4(2)=0.25*(1+rvalue)*(1-svalue);
 shapeq4(3)=0.25*(1+rvalue)*(1+svalue);
 shapeq4(4)=0.25*(1-rvalue)*(1+svalue);

% Ingresando data de las derivadas

 dhdrq4(1)=-0.25*(1-svalue);
 dhdrq4(2)=0.25*(1-svalue);
 dhdrq4(3)=0.25*(1+svalue);
 dhdrq4(4)=-0.25*(1+svalue);

 dhdsq4(1)=-0.25*(1-rvalue);
 dhdsq4(2)=-0.25*(1+rvalue);
 dhdsq4(3)=0.25*(1+rvalue);
 dhdsq4(4)=0.25*(1-rvalue);
```

Published with MATLAB® R2019b

Elem2D_8_nodos.m

```
%{
Propósito:
    Obtener las funciones de forma y sus derivadas en funcion a coordenadas
    naturales, para un elemento bidimensional con 8 nodos por elemento.
Descripción de datos de ingreso:
    - samp, ig,jg = Coordenadas del punto de integracion trabajado y peso.
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Julio 2018
%}

function[der,fun] = Elem2D_8nodos(samp, ig,jg)
xi=samp(ig,1);
eta=samp(jg,1);
etam=(1.-eta);
etap=(1.+eta);
xim=(1.-xi);
xip=(1.+xi);

fun(1) = -0.25*xim*etam*(1.+ xi + eta);
```

```
fun(2) = 0.5*(1.- xi^2)*etam;
fun(3) = -0.25*xip*etam*(1. - xi + eta);
fun(4) = 0.5*xip*(1. - eta^2);
fun(5) = -0.25*xip*etap*(1. - xi - eta);
fun(6) = 0.5*(1. - xi^2)*etap;
fun(7) = -0.25*xim*etap*(1. + xi - eta);
fun(8) = 0.5*xim*(1. - eta^2);

der(1,1)=0.25*etam*(2.*xi + eta); der(1,2)=-1.*etam*xi;
der(1,3)=0.25*etam*(2.*xi-eta); der(1,4)=0.5*(1-eta^2);
der(1,5)=0.25*etap*(2.*xi+eta); der(1,6)=-1.*etap*xi;
der(1,7)=0.25*etap*(2.*xi-eta); der(1,8)=-0.5*(1.-eta^2);

der(2,1)=0.25*xim*(2.*eta+xi); der(2,2)=-0.5*(1. - xi^2);
der(2,3)=-0.25*xip*(xi-2.*eta); der(2,4)=-1.*xip*eta;
der(2,5)=0.25*xip*(xi+2.*eta); der(2,6)=0.5*(1.-xi^2);
der(2,7)=-0.25*xim*(xi-2.*eta); der(2,8)=-1.*xim*eta;
```

Published with MATLAB® R2019b

Elem3D_8_nodos.m

```
%{
Propósito:
    Obtener las funciones de forma y sus derivadas en función a coordenadas
    naturales, para un elemento tridimensional con 8 nodos por elemento.
Descripción de datos de ingreso:
    - rvalue,svalue,tvalue = Coordenadas del punto de integración trabajado.
Orden de los Nodos trabajados:
    1er Nodo (-1,-1,-1)
    2do Nodo (1,-1,-1)
    3er Nodo (1,1,-1)
    4to Nodo (-1,1,-1)
    5to Nodo (-1,-1,1)
    6to Nodo (1,-1,1)
    7mo Nodo (1,1,1)
    8vo Nodo (-1,1,1)
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Julio 2018
%}

function [shapes8,dhdrs8,dhdss8,dhdts8]=Elem3D_8nodos(rvalue,svalue,tvalue)

% Ingresando data de las funciones de forma
 shapes8(1)=0.125*(1-rvalue)*(1-svalue)*(1-tvalue);
 shapes8(2)=0.125*(1+rvalue)*(1-svalue)*(1-tvalue);
 shapes8(3)=0.125*(1+rvalue)*(1+svalue)*(1-tvalue);
 shapes8(4)=0.125*(1-rvalue)*(1+svalue)*(1-tvalue);
 shapes8(5)=0.125*(1-rvalue)*(1-svalue)*(1+tvalue);
 shapes8(6)=0.125*(1+rvalue)*(1-svalue)*(1+tvalue);
 shapes8(7)=0.125*(1+rvalue)*(1+svalue)*(1+tvalue);
```

```
 shapes8(8)=0.125*(1-rvalue)*(1+svalue)*(1+tvalue);

% Ingresando data de las derivadas

 dhdrs8(1)=-0.125*(1-svalue)*(1-tvalue);
 dhdrs8(2)=0.125*(1-svalue)*(1-tvalue);
 dhdrs8(3)=0.125*(1+svalue)*(1-tvalue);
 dhdrs8(4)=-0.125*(1+svalue)*(1-tvalue);
 dhdrs8(5)=-0.125*(1-svalue)*(1+tvalue);
 dhdrs8(6)=0.125*(1-svalue)*(1+tvalue);
 dhdrs8(7)=0.125*(1+svalue)*(1+tvalue);
 dhdrs8(8)=-0.125*(1+svalue)*(1+tvalue);

 dhdss8(1)=-0.125*(1-rvalue)*(1-tvalue);
 dhdss8(2)=-0.125*(1+rvalue)*(1-tvalue);
 dhdss8(3)=0.125*(1+rvalue)*(1-tvalue);
 dhdss8(4)=0.125*(1-rvalue)*(1-tvalue);
 dhdss8(5)=-0.125*(1-rvalue)*(1+tvalue);
 dhdss8(6)=-0.125*(1+rvalue)*(1+tvalue);
 dhdss8(7)=0.125*(1+rvalue)*(1+tvalue);
 dhdss8(8)=0.125*(1-rvalue)*(1+tvalue);

 dhdts8(1)=-0.125*(1-rvalue)*(1-svalue);
 dhdts8(2)=-0.125*(1+rvalue)*(1-svalue);
 dhdts8(3)=-0.125*(1+rvalue)*(1+svalue);
 dhdts8(4)=-0.125*(1-rvalue)*(1+svalue);
 dhdts8(5)=0.125*(1-rvalue)*(1-svalue);
 dhdts8(6)=0.125*(1+rvalue)*(1-svalue);
 dhdts8(7)=0.125*(1+rvalue)*(1+svalue);
 dhdts8(8)=0.125*(1-rvalue)*(1+svalue);
end
```

Published with MATLAB® R2019b

Jacob3d.m

```
%{
Propósito:
    Obtención del jacobiano del elemento que se esta analizando.
Descripción de Datos de Ingreso:
    - nnel = Número de nodos en ele elemento tridimensional(8).
    - dhdr,dhds,dhdt = derivadas de las funciones de forma.
    - xcoord,ycoord,zcoord = coordendas de los nodos del elemento analizado.
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Julio 2018
%}

function [jacob3]=jacob3d(nnel,dhdr,dhds,dhdt,xcoord,ycoord,zcoord)
% Inicializando matriz con el jacobiano del elemento.
jacob3=zeros(3,3);
```

```
% Realizado un bucle for para obtener la sumatoria de todos los nodos por
% elemento.
for i=1:nnel
 jacob3(1,1)=jacob3(1,1)+dhdr(i)*xcoord(i);
 jacob3(1,2)=jacob3(1,2)+dhdr(i)*ycoord(i);
 jacob3(1,3)=jacob3(1,3)+dhdr(i)*zcoord(i);
 jacob3(2,1)=jacob3(2,1)+dhds(i)*xcoord(i);
 jacob3(2,2)=jacob3(2,2)+dhds(i)*ycoord(i);
 jacob3(2,3)=jacob3(2,3)+dhds(i)*zcoord(i);
 jacob3(3,1)=jacob3(3,1)+dhdt(i)*xcoord(i);
 jacob3(3,2)=jacob3(3,2)+dhdt(i)*ycoord(i);
 jacob3(3,3)=jacob3(3,3)+dhdt(i)*zcoord(i);
end
end
```

Published with MATLAB® R2019b

Matdeform2d.m

```
%{
Propósito:
    Crear la matriz de deformación a traves de la asignación
    de las derivadas de las funciones de forma en coordenadas globales.
Descripción de Datos de Ingreso:
    - nne = Número de nodos en el elemento tridimensional(8).
    - eldof = Número de Grados de Libertad en el elemento Analizado.
    - deriv = Derivadas de las funciones de forma en coord globales.
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Julio 2018
%}

function[bee] = matdeform2d(deriv,nne,eldof)

% Asignación de la matriz de deformacion a traves de las funciones de forma
% y sus derivadas.
bee=zeros(3,eldof);
for m=1:nne
    k=2*m;
    l=k-1;
    x=deriv(1,m);
    bee(1,l)=x;
    bee(3,k)=x;
    y=deriv(2,m);
    bee(2,k)=y;
    bee(3,l)=y;
end
```

Published with MATLAB® R2019b

Matdeform2d_cort.m

```
%{
Propósito:
    Crear la matriz de deformación a traves de la asignación
    de las derivadas de las funciones de forma en coordenadas globales.
    Matriz de deformación orientada a corte.
Descripción de datos de ingreso:
    - nne = Número de nodos en el elemento tridimensional(8).
    - eldof = Número de Grados de Libertad en el elemento Analizado.
    - deriv = Derivadas de las funciones de forma en coord globales.
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Julio 2018
%}

function[bees] = matdeform2d_cort(deriv,fun, nne,eldof)

% Asignación de la matriz de deformación a traves de las funciones de forma
% y sus derivadas.
bees=zeros(2,eldof);
for m=1:nne
    i=3*m;
    j=i-1;
    k=i-2;
    x=deriv(1,m); y=deriv(2,m);
    bees(2,i)=-y;
    bees(1,i)=-x;
    bees(1,k) = fun(m);
    bees(2,j) = fun(m);
end
```

Published with MATLAB® R2019b

Matdeform2d_flex.m

```
%{
Propósito:
    Crear la matriz de deformación a traves de la asignación
    de las derivadas de las funciones de forma en coordenadas globales.
    Matriz de deformación orientada a flexión.
Descripción de datos de ingreso:
    - nne = Número de nodos en el elemento tridimensional(8).
    - eldof = Número de Grados de Libertad en el elemento analizado.
    - deriv = Derivadas de las funciones de forma en coord globales.
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Julio 2018
%}

function[beeb] = matdeform2d_flex(deriv,nne,eldof)
```

```
% Asignación de la matriz de deformación a traves de las funciones de forma
% y sus derivadas.
beeb=zeros(3,eldof);
for m=1:nne
    k=3*m-1;
    j=k-1;
    x=deriv(1,m);
    beeb(1,j)=x;
    beeb(3,k)=x;
    y=deriv(2,m);
    beeb(2,k)=y;
    beeb(3,j)=y;
end
```

Published with MATLAB® R2019b

Matdeform3d.m

```
%{
Propósito:
    Crear la matriz de deformación a traves de la asignación
    de las derivadas de las funciones de forma en coordenadas globales.
Descripción de datos de ingreso:
    - nnel = Número de nodos en el elemento tridimensional(8).
    - dhdx,dhdy,dhdz = derivadas de las funciones de forma en coord globales.
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Julio 2018
%}

function [kinmtx3]=matdeform3d(nnel,dhdx,dhdy,dhdz)

% Bucle para determinar la matriz de deformación asignando derivadas de
% funciones de forma en coord. globales para cada nodo.
for i=1:nnel
    i1=(i-1)*3+1;
    i2=i1+1;
    i3=i2+1;
    kinmtx3(1,i1)=dhdx(i);
    kinmtx3(2,i2)=dhdy(i);
    kinmtx3(3,i3)=dhdz(i);
    kinmtx3(4,i1)=dhdy(i);
    kinmtx3(4,i2)=dhdx(i);
    kinmtx3(5,i1)=dhdz(i);
    kinmtx3(5,i3)=dhdx(i);
    kinmtx3(6,i2)=dhdz(i);
    kinmtx3(6,i3)=dhdy(i);
end
end
```

Published with MATLAB® R2019b

NumPInteg1D.m

```
%{
Propósito:
    Obtener los puntos de integración y coeficientes de peso para un
    elemento unidimensional
Descripción de datos de ingreso:
    - npix= Número de puntos de integracion a tratar en el eje
            correspondiente
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Mayo 2019
%}

function [pint1d,coefpes1d]=NumPInteg1D(npix)

% Creando las variables de Puntos y Pesos
pint1d=zeros(npix,1);
coefpes1d=zeros(npix,1);

% Introduciendo la data de la cuadratura de Guass dependiendo del orden del
% elemento (Número de Puntos de integración a trabajar).

switch npix
    case 1      % Cuadratura de Gauss de Primer Orden
        pint1d(1)=0.0;
        coefpes1d(1)=2.0;
    case 2      % Cuadratura de Gauss de Segundo Orden
        pint1d(1)=-1./sqrt(3);
        pint1d(2)=-pint1d(1);
        coefpes1d(1)=1.0;
        coefpes1d(2)=coefpes1d(1);
    case 3      % Cuadratura de Gauss de Tercer Orden
        pint1d(1)=-0.774596669241483;
        pint1d(2)=0.0;
        pint1d(3)=-pint1d(1);
        coefpes1d(1)=0.555555555555556;
        coefpes1d(2)=0.888888888888889;
        coefpes1d(3)=coefpes1d(1);
    case 4      % Cuadratura de Gauss de Cuarto Orden
        pint1d(1)=-0.861136311594053;
        pint1d(2)=-0.339981043584856;
        pint1d(3)=-pint1d(2);
        pint1d(4)=-pint1d(1);
        coefpes1d(1)=0.347854845137454;
        coefpes1d(2)=0.652145154862546;
        coefpes1d(3)=coefpes1d(2);
        coefpes1d(4)=coefpes1d(1);
    otherwise   % Cuadratura de Gauss de Quinto Orden / Maxima
        pint1d(1)=-0.906179845938664;
        pint1d(2)=-0.538469310105683;
        pint1d(3)=0.0;
        pint1d(4)=-pint1d(2);
        pint1d(5)=-pint1d(1);
```

```
        coefpes1d(1)=0.236926885056189;
        coefpes1d(2)=0.478628670499366;
        coefpes1d(3)=0.568888888888889;
        coefpes1d(4)=coefpes1d(2);
        coefpes1d(5)=coefpes1d(1);
end
end
```

Published with MATLAB® R2019b

NumPInteg2D.m

```
%{
Propósito:
    Obtener los puntos de integración y coeficientes de peso para un
    elemento bidimensional
Descripción de datos de ingreso:
    - nglx,ngly = Número de puntos de integración a tratar en cada uno
                  de los ejes cartesianos.
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Mayo 2019
%}

function [point2,weight2]=NumPInteg2D(nglx,ngly)
% Definiendo el mayor N° Punto de Int. para inicializar la variable.
if nglx > ngly
  ngl=nglx;
else
  ngl=ngly;
end

% Inicializando las variables de puntos de integración y pesos
point2=zeros(ngl,2);
weight2=zeros(ngl,2);

% Introduciendo la data de la cuadratura de gauss en los p.integración
[pointx,weightx]=NumPInteg1D(nglx);
[pointy,weighty]=NumPInteg1D(ngly);

% Incorporando la data en una matriz general que contega los 2 ejes

 for intx=1:nglx
   point2(intx,1)=pointx(intx);
   weight2(intx,1)=weightx(intx);
 end

 for inty=1:ngly
   point2(inty,2)=pointy(inty);
   weight2(inty,2)=weighty(inty);
 end
```

Published with MATLAB® R2019b

NumPInteg3D.m

```
%{
Propósito:
    Obtener los puntos de integración y coeficientes de peso para un
    elemento tridimensional
Descripción de datos de ingreso:
    - nglx,ngly,nglz = Número de puntos de integración a tratar en cada uno
                       de los ejes cartesianos.
Diseño: Ing. Jorge Alvarez Ruffrán
Fecha de última actualización:
    Mayo 2019
%}

function [pint3d,coefpes3d]=NumPInteg3D(npix,npiy,npiz)

% Definiendo el mayor N° Punto de Int. para inicializar la variable.
npimax=max([npix npiy npiz]);

% Creando las variables de Puntos y Pesos
pint3d=zeros(npimax,3);
coefpes3d=zeros(npimax,3);

% Introduciendo la data de la cuadratura de gauss en los p.integracion
[pint1dx,coefpes1dx]=NumPInteg1D(npix);
[pint1dy,coefpes1dy]=NumPInteg1D(npiy);
[pint1dz,coefpes1dz]=NumPInteg1D(npiz);

% Incorporando la data en una matriz general que contega los 3 ejes
pint3d(1:npix,1)=pint1dx;       % Eje X
coefpes3d(1:npix,1)=coefpes1dx;

pint3d(1:npiy,2)=pint1dy;       % Eje Y
coefpes3d(1:npiy,2)=coefpes1dy;

pint3d(1:npiz,3)=pint1dz;       % Eje Z
coefpes3d(1:npiz,3)=coefpes1dz;
end
```

Published with MATLAB® R2019b