ANÁLISIS ESTRUCTURAL Y OPTIMIZACIÓN DE ESTRUCTURAS TIPO CASCARÓN

JORGE HUMBERTO ARGUELLO RODRIGUEZ
DANIEL FEDERICO BUITRAGO VILLAMIZAR

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍA CIVIL
BUCARAMANGA
2014

ANÁLISIS ESTRUCTURAL Y OPTIMIZACIÓN DE ESTRUCTURAS TIPO CASCARÓN

JORGE HUMBERTO ARGUELLO RODRIGUEZ
DANIEL FEDERICO BUITRAGO VILLAMIZAR

Plan de Proyecto de Grado para optar al título de
Ingeniero Civil

Director
OSCAR JAVIER BEGAMBRE CARRILLO
Ingeniero Civil

Codirector
LEONARDO MORENO DE LUCA
Ingeniero Civil

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍA CIVIL
BUCARAMANGA
2014

DEDICATORIA

Este trabajo de grado está dedicado a ustedes... a mi familia. Mis padres Humberto Arguello y mi querida madre Herminia Rodríguez, los cuales me han brindado esta oportunidad confiando en mí y apoyándome en este largo camino de formación universitaria, gracias por sus consejos, el apoyo incondicional y los valores inculcados en casa, ya que estos me han servido para ser un buen alumno, compañero y ahora en esta nueva etapa que comienza en mi vida, sé que me servirán para ser un excelente profesional.

Espero con la ayuda de Dios retribuirles tanto esfuerzo y sacrificio echo por mí, con la ilusión de verme salir adelante.

No puedo dejar de nombrar a mis hermanos; Liliana, Patricia y Fernando, igualmente a mi linda sobrina Camila, quienes han sido un apoyo, una voz de aliento. Gracias familia porque han influido en gran medida para ser quien soy, para madurar y superarme personal y profesionalmente, gracias porque me han brindado su amor, tolerancia y el apoyo incondicional a lo largo de todos estos años; sin ustedes esto no habría sido posible. Por estoy y mucho más...

¡¡¡¡Gracias!!!!

JORGE

DEDICATORIA

Este trabajo de grado está dedicado a mi familia. Mis padres Luis Felipe Buitrago y mi querida madre Gloria Beatriz Villamizar, mis Hermanos Christian, Leidy, Fabián y Carolina, a mis amigos Gustavo, Jorge, Víctor, Brayam, Jaiber, Sandra, Stephany, por el apoyo incondicional que me brindaron en este largo camino de formación universitaria, gracias por sus consejos ya que estos me han servido para ser un buena persona, compañero y ahora en esta nueva etapa que comienza en mi vida, sé que me servirán para ser un excelente profesional.

Espero con la ayuda de Dios retribuirles tanto esfuerzo y sacrificio echo por mí, con la ilusión de verme salir adelante.

Gracias familia porque han influido en gran medida para ser quien soy, para madurar y superarme personal y profesionalmente, gracias porque me han brindado su amor, tolerancia y el apoyo incondicional a lo largo de todos estos años; sin ustedes esto no habría sido posible. Por estoy y mucho más...

¡¡¡¡Gracias!!!!

DANIEL

AGRADECIMIENTOS

Las autores expresamos nuestros más sinceros agradecimientos a:

A la ***Universidad Industrial de Santander*** *por ser nuestra alma máter.*

Al ***Ingeniero Oscar Javier Begambre Carrillo*** *y al* ***Ingeniero Leonardo Moreno de Luca*** *por guiarnos en el desarrollo de nuestro proyecto.*

A la ***Escuela de Ingeniería de Civil (EIC)*** *por su colaboración en nuestra formación como ingenieros.*

TABLA DE CONTENIDO

Pag.

LISTA DE FIGURAS

Pag.

RESUMEN

TITULO: ANÁLISIS ESTRUCTURAL Y OPTIMIZACIÓN DE ESTRUCTURAS TIPO CASCARÓN*

AUTORES: JORGE HUMBERTO ARGUELLO RODRIGUEZ
DANIEL FEDERICO BUITRAGO VILLAMIZAR**

PALABRAS CLAVE:

ESTRUCTURAS TIPO CASCARÓN, UPSO (*UNIFIED PARTICLE SWARM OPTIMIZATION*), ENERGIA DE DEFORMACION, ESPESOR.

DESCRIPCIÓN:

La utilización de los métodos numéricos y computacionales para análisis de estructuras tipo cascaron han venido tomando fuerza debido a sus excelentes condiciones estructurales y arquitectónicas. Prueba de esto es el auge en el diseño y construcción de este tipo de estructuras que se dio en la época de 1920.

El artículo presenta un estudio sobre la influencia que podría tener una sección transversal variable de los cascarones en concreto reforzado, sobre la relación entre rigidez y peso. En este sentido, lo que se pretende es aportar en el campo de diseño estructural de cascarones en concreto, implementando un proceso de optimización heurístico que tenga como variable el espesor de la sección transversal, optimice la energía de deformación y el peso de la estructura, partiendo de una configuración geométrica predeterminada. Para lograrlo será necesaria la programación del método de elementos finitos para estructuras tipo cascaron en Matlab, en combinación de un algoritmo heurístico de optimización como el *Unified Particle Swarm Optimization (UPSO)*. Vale la pena resaltar que el procedimiento de optimización pretende resolver un conflicto entre el peso y rigidez, ya que la estructura más liviana posible (*cuyo espesor tendera a ser cero*) va a tener una energía de deformación muy grande y una estructura con una energía de deformación pequeña (*con gran espesor)* va a tener un peso bastante alto. Por otro lado, con el fin de continuar con el desarrollo del proyecto de investigación titulado "*Optimización Geométrica de Cubiertas con Estructura Tipo Cascarón para Minimizar la Intensidad de la Radiación Solar*", presentado por el estudiante Olman Rincón Gaviria de la Escuela de Ingeniería Civil de la Universidad Industrial de Santander, en año 2013. En este proyecto se logró obtener la geometría óptima para una cubierta con dichas características, en lo referente a la disminución de la incidencia de la radiación solar.

* Trabajo de grado

** Facultad físico-mecánicas, Escuela de Ingeniería Civil, Director: Oscar Javier Begambre Carrillo. Codirector: Leonardo Moreno de Luca

ABSTRACT

TITLE: STRUCTURAL ANALYSIS AND OPTIMIZATION OF SHELL TYPE STRUCTURES*

AUTHORS: JORGE RODRIGUEZ HUMBERTO ARGUELLO
DANIEL FEDERICO BUITRAGO VILLAMIZAR**

KEYWORDS:

SHELL TYPE STRUCTURES, UPSO (UNIFIED PARTICLE SWARM OPTIMIZATION) STRAIN ENERGY, THICKNESS.

DESCRIPTION

The use of numerical and computing methods for shell-typed structures analysis have become more common due to their excellent structure and architecture conditions. Evidence of this situation is the growth of design and construction of this type of structures starting in the 1920's.

The article presents a study about the influence that could have a perpendicular section variable of the shells in reinforced concrete, over the relation between stiffness and weight. In this sense, what one pretends to contribute to structural's design field of concrete shells. Implementing an heuristic optimization process that has as a variable the thickness of the perpendicular section, optimizes the deformation energy and the structures' weight, beginning from a predetermined geometric setting. To accomplish that, it will be necessary the finite elements method programming for shell-typed structures in Matlab, in combination of an optimization heuristic algorithm like the Unified Particle Swarm Optimization (UPSO). It is important to emphasize that the optimization procedure pretends to solve a conflict between weight and stiffness, because the lightest structure possible (whose thickness will tend to be "0") is going to have a very big deformation energy and a structure with a small deformation energy (with a great thickness) is going to be very heavy. On the other hand in order to continue with the research project named "Covers' Geometric optimization with shell-typed structure to minimize solar radiation's intensity", presented by the student Olman Rincon Gaviria from the school of Civil Engineering of the Industrial University of Santander, in 2013. In this project it was obtained the ideal geometry for a cover with these characteristics, in terms of reducing the solar radiation's impact.

* Work Dregree

** Physical-Mechanical Faculty. Faculty, School of Civil Engineering. Director: Oscar Javier Begambre Carrillo. Codirector: Leonardo Moreno de Luca

INTRODUCCIÓN

Una tarea de ingeniería típica durante el desarrollo de cualquier sistema estructural es, mejorar su rendimiento en términos de costo de construcción y la respuesta estructural. Las mejoras pueden lograrse ya sea por el simple uso de criterios de diseño basados en códigos existentes o en una forma automatizada mediante el uso de métodos de optimización que conducen a un diseño estructural que puede ser considerado como el óptimo. En sentido estricto, el diseño óptimo significa que no existe una mejor solución bajo ciertas restricciones. Sin embargo, encontrar la solución óptima global es una tarea muy difícil, debido a la incertidumbre o de dispersión involucrada en varios parámetros estructurales tales como las propiedades del material, las imperfecciones geométricas, las variaciones de carga, condiciones de contorno inciertos, etc. Un cascarón es una estructura tridimensional delgada cuya resistencia se obtiene dando forma al material según las cargas que deben soportar, son lo suficientemente delgadas para no desarrollar flexión, pero también suficientemente gruesas para resistir cargas, que según el caso pueden ser de compresión, corte y/o tracción.

Teniendo en cuenta lo mencionado anteriormente, lo planteado en este proyecto de investigación, es profundizar en la influencia que podría tener una sección transversal variable de los cascarones en concreto reforzado, sobre la relación entre rigidez y peso. En este sentido, lo que se pretende es aportar al desarrollo en el campo del diseño estructural de cascarones en concreto, implementando un proceso de optimización heurístico que tenga como variable el espesor de la sección transversal, optimice la energía de deformación y el peso de la estructura, partiendo de una configuración geométrica predeterminada con la ayuda de *RHINOCEROS 3D*, programa con el cual se define la forma y geometría de la superficie. Para lograrlo, será necesaria la programación del método de elementos

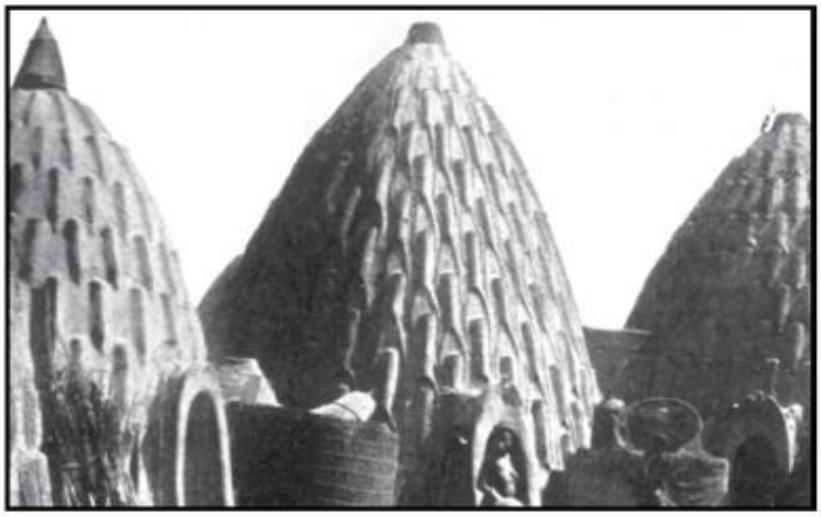
finitos para estructuras tipo cascarón, y la implementación de un algoritmo heurístico de optimización como el *Unified Particle Swarm Optimization* (UPSO) [1]. Vale la pena resaltar que el procedimiento de optimización pretende resolver un conflicto entre peso y rigidez, ya que la estructura más liviana posible (cuyo espesor tenderá a ser cero) va a tener una energía de deformación muy grande, y una estructura con una energía de deformación pequeña (con gran espesor) va a tener un peso bastante alto.

1. CONTEXTUALIZACION DE ESTRUCTURAS TIPO CASCARON

1.1. ANTECEDENTES HISTÓRICOS

En construcciones hechas por el hombre han aparecido hace ya varios siglos atrás, utilizando diferentes formulaciones como las teorías de membrana y de flexión de cascarones formuladas por Geckeler [2] y Aron [3]. Love (1927) presentó las bases generales de la teoría de la cáscara y fue seguido más tarde por muchos autores "tener en cuenta las referencias [4] , [5], [6] , [7], [8] " Se encontraron soluciones analíticas de forma cerrada para los casos simples. En las últimas cuatro décadas, desde la introducción del MEF, esta técnica se aplicó a obtener soluciones para los problemas más complejos. Esto con el fin de proveer a la sociedad de templos, techos, catedrales, entre otros [9].

Figura 1. Adobe Ringdome Houses in Cameroon



Fuente: BERGER, Horst. Structural form in architecture**;** domes arches and shells. Structure Magazine. [Online] November, 2007. [Cited: 19 Feb. 2014] Available from Internet: <http://www.ciccp.es/ImgWeb/Castilla%20y%20Leon/Ingenier%C3%ADaHumanismo/Structural%20Form%20Architecture.pdf>

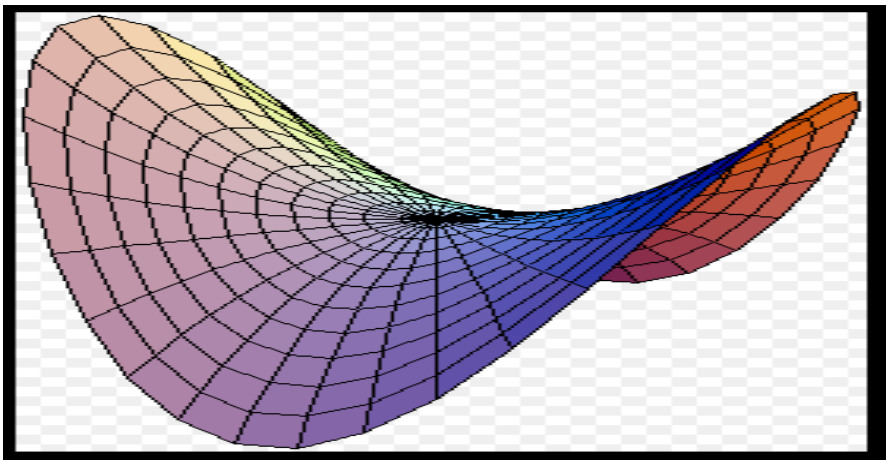
En la (***Figura 1),*** se observa la capacidad que tenían las culturas pasadas donde se construían casas en forma de cascarones los cuales eran capaces de soportar las cargas impuestas. Este tipo de estructuras capaces de adoptar cualquier forma empezaron a llamar la atención de ingenieros, arquitectos y diseñadores de todo tipo, en la actualidad se pueden observar grandes proyectos realizados, que han logrado llegar más allá de lo común, entre ellos está el arquitecto *Félix Candela*, de nacionalidad española y mexicana, famoso por la creación de estructuras basadas en el uso extensivo del *paraboloide hiperbólico* [10] ,(***Figura 2).*** Se destaca la gran variedad de formas que se puede llegar a construir, también se debe hacer énfasis en la aplicación de los cascarones en la industria automotriz y aeroespacial, debido al avance considerable de estas ramas.

Figura 2. Félix Candela Xochimilco Restaurant

Fuente: CANDELA Félix, Xochimilco Restaurant [Online]. [Cited 19 de febrero 2014]. Available from internet: <http://www.taschen.com/pages/es/catalogue/architecture/all/05120/facts.candela.htm>

El paraboloide hiperbólico es una superficie doblemente reglada por lo que se puede construir a partir de rectas. Por su apariencia, también se lo denomina superficie de silla de montar (Figura 3).

Figura 3. Paraboloide Hiperbólico



Fuente: ISLER, Heinz**,** Paraboloide Hiperbólico. [Online]. [Cited: 22 Feb. 2014] Available from Internet:
< http://es.wikipedia.org/wiki/Paraboloide#Paraboloide_hiperb.C3.*B3lico* >

Otro gran especialista en este tema fue *Heinz Isle*r ingeniero civil Suizo, famoso por sus delgadas láminas de hormigón en forma de cascaron (***Figura 4)***.

Figura 4. Techo de concreto cáscara del centro de jardinería Wyss en Zuchwil

Fuente: Techo de concreto cáscara del centro de jardinería. Wyss en Zuchwil [Online]. [Cited: 22 Feb. 2014] Available from Internet:
<http://en.wikipedia.org/wiki/Heinz_Isler#mediaviewer/File:GartencenterWyss Zuchwil_01_09.jpg>

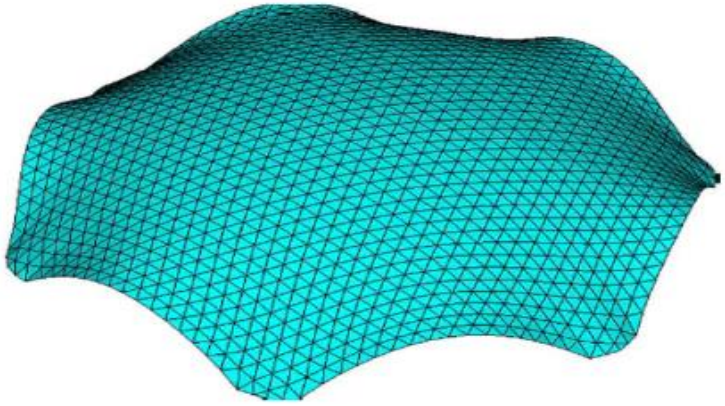
1.2. DEFINICIÓN DE ESTRUCTURAS TIPO CASCARON

Los cascarones son estructuras laminares curvas, en las que el material está agrupado alrededor de una superficie directriz curva, de tal manera que el espesor en sentido normal a la superficie directriz es pequeño en comparación con las otras dimensiones transversales [11]. Una característica fundamental de los cascarones es precisamente su curvatura, ya que si la superficie directriz fuese plana, la estructura podría tratarse utilizando los elementos planos, tipo membrana o placa, ya conocidos. Otra característica muy importante es que, por el hecho de ser curvas, los efectos de esfuerzo axial y de flexión están acoplados, es decir que el esfuerzo axial provoca esfuerzos de flexión y viceversa. En una estructura plana ambos efectos son independientes [12]. Ejemplos de cascarones son los depósitos, las bóvedas, laminares, las carrocerías, etc. y en general cualquier pieza de superficie curva [13]. Para el tratamiento de las láminas pueden plantearse varias alternativas: aproximar el cascarón mediante una serie de facetas planas y utilizar elemento planos, utilizar elementos sólidos tridimensionales y prescindir de la naturaleza real del cascarón, o utilizar una formulación específica para cáscaras curvas [12].

Las características estructurales de este tipo de cascarones permite una gran rigidez, bajo peso, y a la vez realizar un excelente diseño para aprovechar las capacidades de la misma, estas superficies debido a sus dimensiones permite representar un modelo geométrico sin espesor con curvaturas variables a lo largo de las mismas en cualquier dirección, sin embargo el espesor se tendrá en cuenta en las ecuaciones para el cálculo de los esfuerzos y deformaciones. Estas estructuras, por lo general están compuestas por diversas superficies curvas, para lograr realizar más práctico el estudio, se puede discretizar la cubierta tipo cascarón en elementos planos de cualquier forma, pero que al ser ensamblados

adopten una forma aproximada a la superficie curva [12]. Por lo tanto, entre más se divida la cubierta en elementos planos está será más aproximada a la geometría original (***Figura 5),*** teniendo en cuenta las propiedades de cada elemento.

Figura 5. Modelo de Cubierta en Forma de Cascarón.



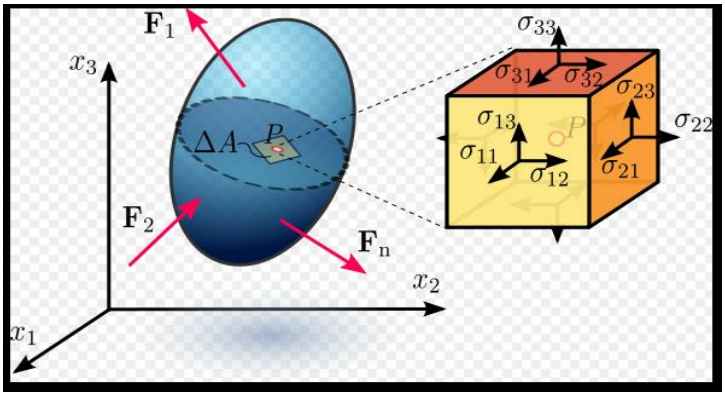
Fuente: Computational Generation of Free Form Shells in Architectural Design and Civil Engineering. [Online]. [Cited: 22 Feb. 2014] Available from Internet: < www.elsevier.com/locate/autcon>

Los elementos planos (***Figura 6)*** tienen características parecidas a las de las estructuras tipo cascaron debido a que una de sus dimensiones es pequeña en comparación con las otras dos. Su análisis está comprendido por esfuerzos y flexión [12].

1.3. ESFUERZOS Y DEFORMACIONES DE UN ELEMENTO PLANO

Primero es necesario conocer las ecuaciones de los elementos que la Conforman y así aplicar el *Método de los Elementos Finitos*, detallando el desarrollo de las ecuaciones constitutivas para esfuerzos y flexión [14].

Figura 6. Elemento Plano Sometido a Esfuerzo y a Flexión



Fuente: Elemento Plano Sometido a Esfuerzo y a Flexión [Online]. [Cited: 22 Feb. 2014] Available from Internet:< http://en.wikipedia.org/wiki/Mohr%27s_circle#mediaviewer/File:Stress_in_a_continuum.svg)>

En la (***Figura 6),*** se representa un elemento a esfuerzos en el plano. Este elemento solo puede generar deformaciones en el plano en dirección "x, y", obteniéndose la siguiente ecuación matricial [14]:

$$\begin{pmatrix} \in_x \\ \in_y \\ \in_{xy} \end{pmatrix} = \begin{pmatrix} 1/E & -v/E & 0 \\ -v/E & 1/E & 0 \\ 0 & 0 & 1/G \end{pmatrix} \begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} \qquad \textbf{(1)}$$

Dónde:

E es el Modulo de Young,

$G = \frac{E}{2*(1+V)}$ es el Módulo de Cortante,

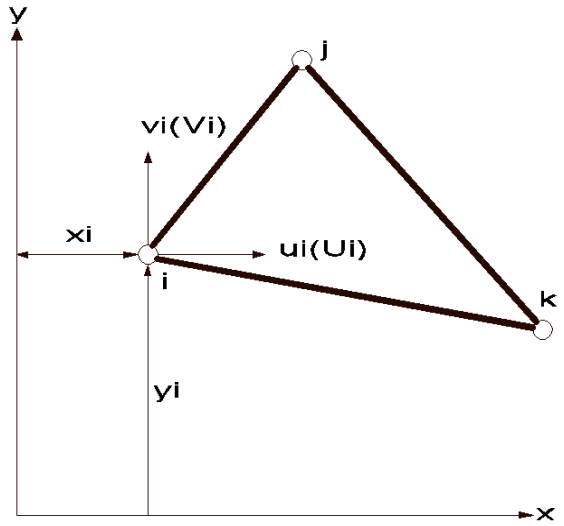
ν es el coeficiente de Poisson,

σi es el esfuerzo normal en dirección i

$$\begin{pmatrix} \in_x \\ \in_y \\ \in_{xy} \end{pmatrix} = \begin{pmatrix} 1/E & -v/E & 0 \\ -v/E & 1/E & 0 \\ 0 & 0 & 1/G \end{pmatrix} * \begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} \qquad \textbf{(2)}$$

En las ecuaciones (1) y (2) se debe tener en cuenta que existen dos grados libertad por cada nodo del elemento, por lo que las ecuaciones tienen en total seis grados de libertad: tres *deformaciones* y tres *esfuerzos*. En el sistema se obtienen las deformaciones de los desplazamientos en dirección $\boldsymbol{x}$ y $\boldsymbol{y}$ respectivamente en función de la ecuación de posición dentro del elemento debido a las cargas sobreimpuestas en el elemento.

Figura 7. Desplazamientos de un elemento Plano Sometido a Esfuerzo



Fuente: ZIENKIEWICZ. O.C. y TAYLOR. R.L. The Finite Element Method Volume 2, Solid Mechanics. 5 edition. [Online]. [Cited: 22 Feb. 2014] Available from Internet: <http://www.academia.edu/208845/Zienkiewicz_y_Taylor_2004_The_finite_element_method_fifth_edition._Volume_2_Solid_Mechanics.pdf>

En la (**Figura 7*),*** Se muestra los desplazamientos definidos como $u\,[x, y]\; y\; v\,[x, y]$:

- Solucionando el sistema las deformaciones y los esfuerzos se obtienen las siguientes ecuaciones:

$$\in_{\boldsymbol{x}} = \frac{\boldsymbol{d_u}}{\boldsymbol{d_x}}; \quad \in_{\boldsymbol{y}} = \frac{\boldsymbol{d_v}}{\boldsymbol{d_y}}; \quad \in_{\boldsymbol{xy}} = \frac{\boldsymbol{d_u}}{\boldsymbol{d_y}} + \frac{\boldsymbol{d_v}}{\boldsymbol{d_x}} \qquad \textbf{(3)}$$

$$\begin{pmatrix} \boldsymbol{\sigma_x} \\ \boldsymbol{\sigma_y} \\ \boldsymbol{\tau_{xy}} \end{pmatrix} = \frac{\boldsymbol{E}}{\boldsymbol{1-v^2}} \begin{pmatrix} \boldsymbol{1} & \boldsymbol{v} & \boldsymbol{0} \\ \boldsymbol{v} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \frac{\boldsymbol{1-v}}{\boldsymbol{2}} \end{pmatrix} * \begin{pmatrix} \in_{\boldsymbol{x}} \\ \in_{\boldsymbol{y}} \\ \in_{\boldsymbol{xy}} \end{pmatrix} \quad \textbf{(4)}$$

Donde

$$\boldsymbol{D} = \begin{pmatrix} \frac{\boldsymbol{E}}{\boldsymbol{1-v^2}} & \frac{\boldsymbol{vE}}{\boldsymbol{1-v^2}} & \boldsymbol{0} \\ \frac{\boldsymbol{vE}}{\boldsymbol{1-v^2}} & \frac{\boldsymbol{E}}{\boldsymbol{1-v^2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \frac{\boldsymbol{E(1-v)}}{\boldsymbol{2(1-v^2)}} \end{pmatrix} \quad \textbf{(5)}$$

1.4. ESFUERZOS Y DEFORMACIONES A FLEXIÓN

Para entender el comportamiento de una placa bajo flexión es necesario tener en cuenta la teoría de Kirchhoff y Timoshenko, que es apropiada para placas delgadas en donde los esfuerzos cortantes son despreciables, esta teoría plantea que la línea normal al plano medio no varía después de la deformación, donde el plano xy Coincide con el plano medio [12]. En la teoría clásica de placas delgadas se hacen las siguientes suposiciones:

- El espesor de la placa es pequeño en comparación con las otras dimensiones
- Las deflexiones son pequeñas
- El plano medio de la placa no está sometido a deformaciones en el plano
- La deformación del corte transversal es cero

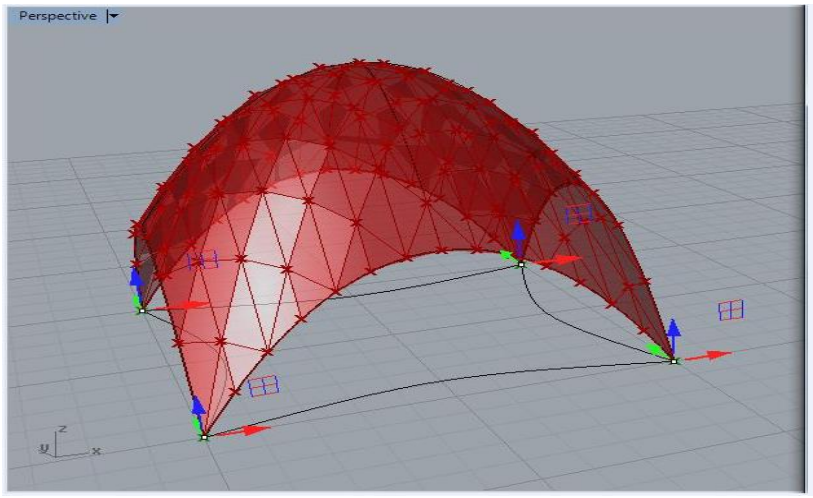
Los esfuerzos inducidos en el elemento de placa plana están sujetos a fuerzas a flexión [12].

2. INTRODUCCION A RHINOCEROS

Es un software computacional capaz de generar formas en tres dimensiones basadas en las NURBS tomando como base los B-Splines que hacen que haya un punto de control y polígonos de control. El programa es generalmente usado para el diseño industrial, arquitectura, diseño naval, diseño automotriz, e ingeniería.

Por medio de esta herramienta de modelado 3D será obtenida la geometría de la superficie tipo cascarón en estudio, exportando las coordenadas de cada uno de los nodos que conforman los elementos triangulares que definen la superficie ver (***Figura 8),*** al código de programación de elementos finitos en Matlab, estas coordenadas obtenidas por medio de Rhinoceros son la base para el inicio de la optimización.

Figura 8. Superficie NURBS obtenida de Rhinoceros 3D



Fuente: Imagen tomada de Rhinoceros 3D

3. OPTIMIZACION DE ESTRUCTURAS TIPO CASCARON

La optimización en general consiste en buscar la solución más satisfactoria posible, por medio de un chequeo de valores óptimos para las variables que la definen teniendo en cuenta las restricciones, es decir existe una metodología de optimización que se divide en tres conceptos [1]:

- Variables de diseño
- Función objetivo
- Restricciones

3.1. VARIABLES DE DISEÑO

Las variables de diseño consideran, los parámetros fijos durante el proceso y las variables que van evolucionando durante la optimización como por ejemplo:

- La geometría
- Propiedades de la sección de la pieza
- Forma de la estructura

3.2. FUNCIÓN OBJETIVO

Generalmente, una función objetivo es donde intervienen las variables, teniendo en cuenta que los criterios más relevantes en este caso para las estructuras tipo

cascaron son el peso y la energía de deformación siendo estas las funciones objetivo a optimizar.

3.3. RESTRICCIONES

Se deben establecer restricciones para delimitar el campo de soluciones para cumplir ciertas condiciones que determinarán si el diseño es válido o no, las restricciones suelen estar asociadas con las limitaciones impuestas a las respuestas de la estructura, como los esfuerzos y desplazamientos.

4. METODO DE LOS ELEMENTOS FINITOS

Es un método de análisis que permite determinar los esfuerzos y deformaciones en los cuerpos sólidos y en los elementos estructurales, a la vez comprender lo que se encuentra en el entorno a partir de la subdivisión de sistemas en dos tipos de problemas: los "*discretos*" es decir son analizados en partes hasta conformar un modelo de un número finito de componentes bien definidos; y los "*continuos*" que se analizan por medio de modelos matemáticos, existe una gran variedad de técnicas de aproximación aplicada en diversos campos, pero también hay diferentes hipótesis de elementos discretos y finitos [15].

Para resolver este problema, el Método de Elementos Finitos se basa en las siguientes hipótesis:

- Un elemento continuo se divide por medio de líneas imaginarias de una diversidad de regiones continuas en formas geométricas sencillas
- Los elementos finitos se unen entre sí en un número finito de puntos, llamados nodos
- Los desplazamientos de los nodos son variables del problema y estos determinan las deformaciones de la estructura

El término de "elemento finito" se empezó a usar por los estudios realizados por Clough [4], quien propuso una metodología a sistemas discretos para demostrar que un cuerpo se puede dividir en finitas partes haciendo que estas partes se ajusten a la forma obteniendo resultados favorables en la solución de problemas de sólidos elásticos continuos [15].

4.1. DESCRIPCIÓN DEL MEF

Es un método numérico capaz de obtener soluciones aproximadas de diversas ecuaciones de manera simplificada desde un enfoque computacional, a problemas geométricos complejos en 3D [16]. Las ecuaciones diferenciales se expresan de manera equivalente, y buscan analizar cada elemento por partes y después ensamblarlos para obtener la solución total de los sistemas. Por ejemplo se puede dividir una superficie continua en varios elementos triangulares o rectangulares en donde de nodos son los que a su vez denotan los grados de libertad que dependen del orden de las derivadas, esto quiere decir que los grados de libertad son las variables de la solución de las ecuaciones constitutivas del elemento.

4.2. MEF PARA ESTRUCTURAS TIPO CASCARÓN

Para el MEF en estructuras complejas, hay una gran variedad de elementos de acuerdo a su geometría, número de nodos por elemento y número de grados de libertad por nodo. En este caso la superficie en estudio será dividida en elementos triangulares los cuales están conformados por dos métodos de análisis:

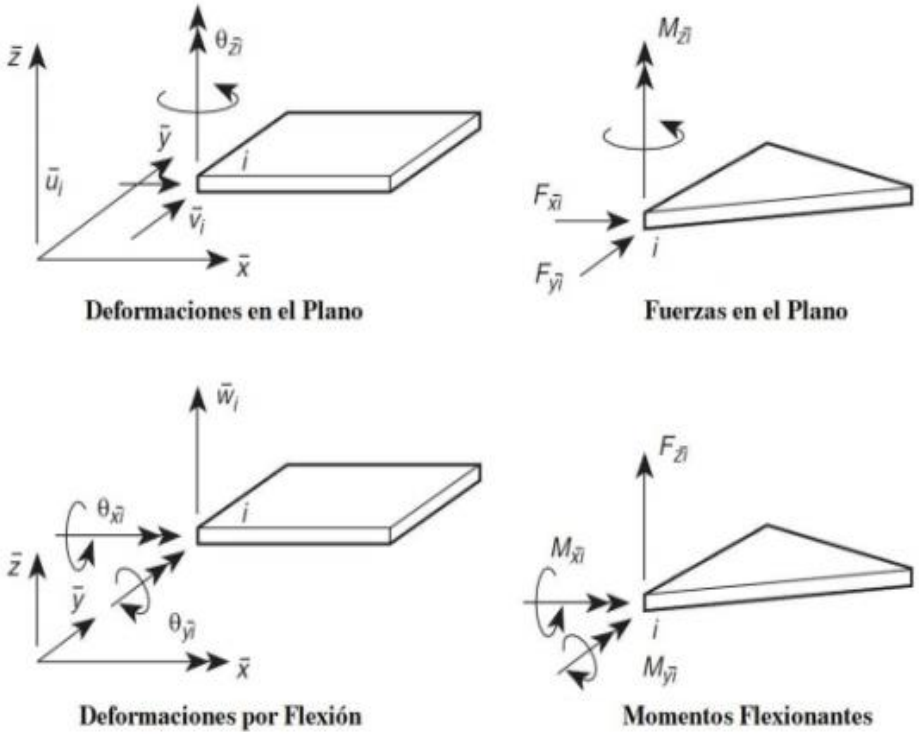
- Análisis de elementos triangulares como membrana
- Análisis de placas triangulares a flexión

Estos dos métodos deben acoplarse ya que la membrana resiste únicamente esfuerzos en el plano y rotación en el eje **z** mientras que la placa a flexión resiste esfuerzos fuera del plano y momentos alrededor de los ejes **x** y **y** ver ***(Figura 9)***;

es decir, los efectos del esfuerzo plano no afectan las deformaciones por flexión y los efectos de flexión no influyen en las deformaciones por esfuerzos [12].

Se debe hacer énfasis en el tema, y se recomienda tener en cuenta la referencia [14] y [12], la interacción entre estos dos se lleva a cabo cuando estos elementos sean ensamblados con diferentes orientaciones, formando así estructuras tipo cascarón. El elemento finito está definido por un triángulo con tres nodos localizados en sus vértices y con seis grados de libertad por nodo.

Figura 9*. Elemento Plano Sometido a Esfuerzo y Flexión*



Fuente: ZIENKIEWICZ. O.C. y TAYLOR. R.L. The Finite Element Method Volume 2, Solid Mechanics. 5 edition. [Online]. [Cited: 22 Feb. 2014] Available from Internet: <http://www.academia.edu/208845/Zienkiewicz_y_Taylor_2004_The_finite_element_method_fifth_edition._Volume_2_Solid_Mechanics.pdf>

La acción de los esfuerzos y deformaciones están dadas en términos de los desplazamientos $\boldsymbol{u}$ y $\boldsymbol{v}$ y las fuerzas están dadas por la acción de los desplazamientos [12]. La ecuación para esfuerzos es:

$$\boldsymbol{K}.\boldsymbol{U} = \boldsymbol{F} \tag{6}$$

Dónde:

$$\boldsymbol{U} = \begin{pmatrix} \overline{\boldsymbol{u}_i} \\ \overline{\boldsymbol{v}_i} \end{pmatrix} ; \ \boldsymbol{F} = \begin{pmatrix} \overline{\boldsymbol{u}_\iota} \\ \overline{\boldsymbol{u}_J} \\ \overline{\boldsymbol{u}_k} \end{pmatrix} ; \overline{\boldsymbol{F}_\iota} = \begin{pmatrix} \boldsymbol{F}_{\bar{x}i} \\ \boldsymbol{F}_{\bar{y}i} \end{pmatrix} ; \ \overline{\boldsymbol{F}_\iota} = \begin{pmatrix} \overline{\boldsymbol{F}_\iota} \\ \overline{\boldsymbol{F}_J} \\ \overline{\boldsymbol{F}_k} \end{pmatrix} \tag{7}$$

Para la flexión, las deformaciones están dadas por el desplazamiento $\boldsymbol{w}$ alrededor del eje $\boldsymbol{z}$ y dos rotaciones ubicadas respectivamente alrededor del eje [x, y], en coordenadas locales, la siguiente ecuación para flexión es [12]:

$$\overline{\boldsymbol{k}}^{\boldsymbol{e}}.\overline{\boldsymbol{U}}^{\boldsymbol{e}} = \boldsymbol{F}^{\boldsymbol{e}} \tag{8}$$

Dónde:

$$\overline{\boldsymbol{U}_\iota}^{\boldsymbol{e}} = \begin{pmatrix} \overline{\boldsymbol{w}_i} \\ \boldsymbol{\theta}_{\bar{x}i} \\ \boldsymbol{\theta}_{\bar{y}i} \end{pmatrix} ; \ \overline{\boldsymbol{F}}^{\boldsymbol{e}} = \begin{pmatrix} \overline{\boldsymbol{u}_\iota}^{\boldsymbol{e}} \\ \overline{\boldsymbol{u}_J}^{\boldsymbol{e}} \\ \overline{\boldsymbol{u}_k}^{\boldsymbol{e}} \end{pmatrix} \tag{9}$$

$$\overline{\boldsymbol{F}_\iota}^{\boldsymbol{e}} = \begin{pmatrix} \boldsymbol{F}_{\bar{x}i} \\ \boldsymbol{M}_{\bar{x}i} \\ \boldsymbol{M}_{\bar{y}i} \end{pmatrix} ; \ \overline{\boldsymbol{F}}^{\boldsymbol{e}} = \begin{pmatrix} \overline{\boldsymbol{F}_\iota}^{\boldsymbol{e}} \\ \overline{\boldsymbol{F}_J}^{\boldsymbol{e}} \\ \overline{\boldsymbol{F}_k}^{\boldsymbol{e}} \end{pmatrix} \tag{10}$$

$\boldsymbol{Fxi}$, $\boldsymbol{Mxi}$ y $\boldsymbol{Myi}$ son la fuerza y los momentos debidos al desplazamiento $\boldsymbol{w}$, así obteniendo resultados satisfactorios, una vez que se ensamblan las ecuaciones en un sistema de referencia *global*, se presenta la interacción entre la deformación y la flexión entonces quedaría:

$$\overline{\boldsymbol{U}}_{\boldsymbol{i}} = \left(\overline{\boldsymbol{u}}_{\boldsymbol{i}}\ \overline{\boldsymbol{v}}_{\boldsymbol{i}}\ \overline{\boldsymbol{w}}_{\boldsymbol{i}}\ \boldsymbol{\theta}_{\overline{\boldsymbol{x}}\boldsymbol{i}}\ \boldsymbol{\theta}_{\overline{\boldsymbol{y}}\boldsymbol{i}}\right)^{\boldsymbol{T}}; \qquad \textbf{(11)}$$

$$\overline{\boldsymbol{F}}_{\boldsymbol{i}} = (\boldsymbol{F}_{\overline{\boldsymbol{x}}\boldsymbol{i}}\ \boldsymbol{F}_{\overline{\boldsymbol{y}}\boldsymbol{i}}\ \boldsymbol{F}_{\overline{\boldsymbol{z}}\boldsymbol{i}}\ \boldsymbol{M}_{\overline{\boldsymbol{x}}\boldsymbol{i}}\ \boldsymbol{M}_{\overline{\boldsymbol{y}}\boldsymbol{i}})^{\boldsymbol{T}} \qquad \textbf{(12)}$$

Y la matriz de rigidez para un elemento tipo cascarón está ensamblada a partir de las sub-matrices correspondientes a los nodos $\boldsymbol{i}$, $\boldsymbol{j}$ y $\boldsymbol{k}$ de la siguiente forma:

$$\overline{\boldsymbol{k}} = \begin{pmatrix} \overline{\boldsymbol{k}}_{\boldsymbol{i}} & 0 & 0 \\ 0 & \overline{\boldsymbol{k}}_{\boldsymbol{j}} & 0 \\ 0 & 0 & \overline{\boldsymbol{k}}_{\boldsymbol{k}} \end{pmatrix} \qquad \textbf{(13)}$$

Donde la matriz de rigidez y el vector de fuerzas para esfuerzos en el plano y flexión, respectivamente son:

$$\overline{\boldsymbol{k}}^{\boldsymbol{ep}} = \boldsymbol{tA}(\boldsymbol{B}.\boldsymbol{C}.\boldsymbol{B}^{\boldsymbol{T}}); \qquad \textbf{(14)}$$

$$\overline{\boldsymbol{k}}^{\boldsymbol{f}} = \iint_{\boldsymbol{A}} \boldsymbol{B}.\boldsymbol{D}.\boldsymbol{B}^{\boldsymbol{T}}\, \boldsymbol{dA}; \qquad \overline{\boldsymbol{r}_{\boldsymbol{q}}}^{\boldsymbol{f}} = \iint_{\boldsymbol{A}} \boldsymbol{qN}\, \boldsymbol{dA} \qquad \textbf{(15)}$$

Dónde:

$\boldsymbol{A}$ es el área del elemento finito

$\boldsymbol{C}$ y $\boldsymbol{D}$ son las matrices de propiedades del material para esfuerzo plano y flexión respectivamente

$\boldsymbol{N}$ son las funciones de interpolación

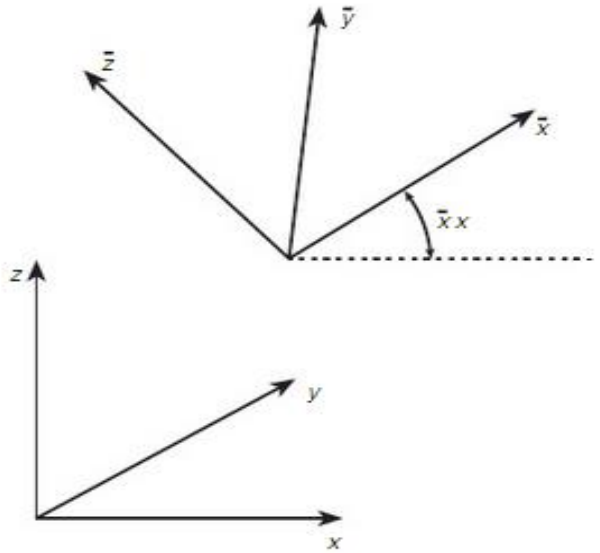
$\boldsymbol{B}$ son las derivadas de las funciones de interpolación

$\boldsymbol{q}$ es la carga normal distribuida sobre la superficie del elemento

4.3. TRANSFORMACIÓN A COORDENADAS GLOBALES Y ENSAMBLE DE ELEMENTOS

La matriz de rigidez anterior ecuación (15), utiliza un sistema de coordenadas locales, para poder lograr el ensamble apropiado de los elementos y ecuaciones de equilibrio será necesario una trasformación de las coordenadas *locales* (designadas por $\boldsymbol{x', y', z'}$), a un sistema *global* (designadas $\boldsymbol{x, y, z}$) [12], ver ***(Figura 10).***

Figura 10. Coordenadas Locales y Globales



Fuente: ZIENKIEWICZ. O.C. y TAYLOR. R.L. The Finite Element Method Volume 2, Solid Mechanics. 5 edition. [Online]. [Cited: 22 Feb. 2014] Available from Internet: <http://www.academia.edu/208845/Zienkiewicz_y_Taylor_2004_The_finite_element_method_fifth_edition._Volume_2_Solid_Mechanics.pdf>

Teniendo en cuenta las reglas de transformaciones ortogonales la matriz de rigidez de un elemento en coordenadas globales sería [12]:

$$\bar{K}^e = T^T * \bar{K}^e{}_l * T \quad \textbf{(16)}$$

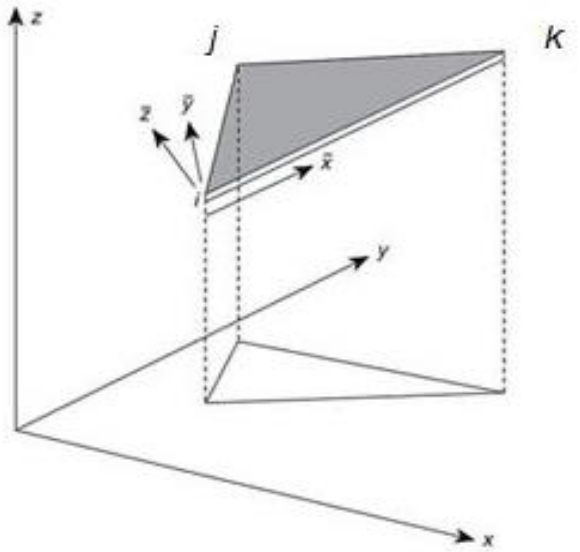
Donde $\bar{K}^e{}_l$ es la matriz de rigidez en el sistema local para cada elemento, la matriz de transformación T que a su vez está formada por la matriz de rotación R (también conocida como matriz de cosenos directores).

$$T = \begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda \end{bmatrix} \quad \textbf{(17)}$$

$$\Lambda = \begin{bmatrix} \Lambda_{\bar{x}x} & \Lambda_{\bar{x}y} & \Lambda_{\bar{x}z} \\ \Lambda_{\bar{y}x} & \Lambda_{\bar{y}y} & \Lambda_{\bar{y}z} \\ \Lambda_{\bar{z}x} & \Lambda_{\bar{z}y} & \Lambda_{\bar{z}z} \end{bmatrix} \quad \textbf{(18)}$$

En la (***Figura 11),*** se ilustran las coordenadas *locales* a *globales* para un elemento triangular

Figura 11. Coordenadas Locales y Globales para un Elemento Triangular



Fuente: ZIENKIEWICZ. O.C. y TAYLOR. R.L. The Finite Element Method Volume 2, Solid Mechanics. 5 edition. [Online]. [Cited: 22 Feb. 2014] Available from Internet: <http://www.academia.edu/208845/Zienkiewicz_y_Taylor_2004_The_finite_element_method_fifth_edition._Volume_2_Solid_Mechanics.pdf>

Donde el sistema de referencia *local* ($\bar{x}$, $\bar{y}$, $\bar{z}$) de un elemento triangular arbitrario se evidencia dentro de un sistema de referencia *global* (x, y, z) que es ensamblado con otros elementos finitos con sistemas de referencia propios referenciados también al sistema x, y, z.

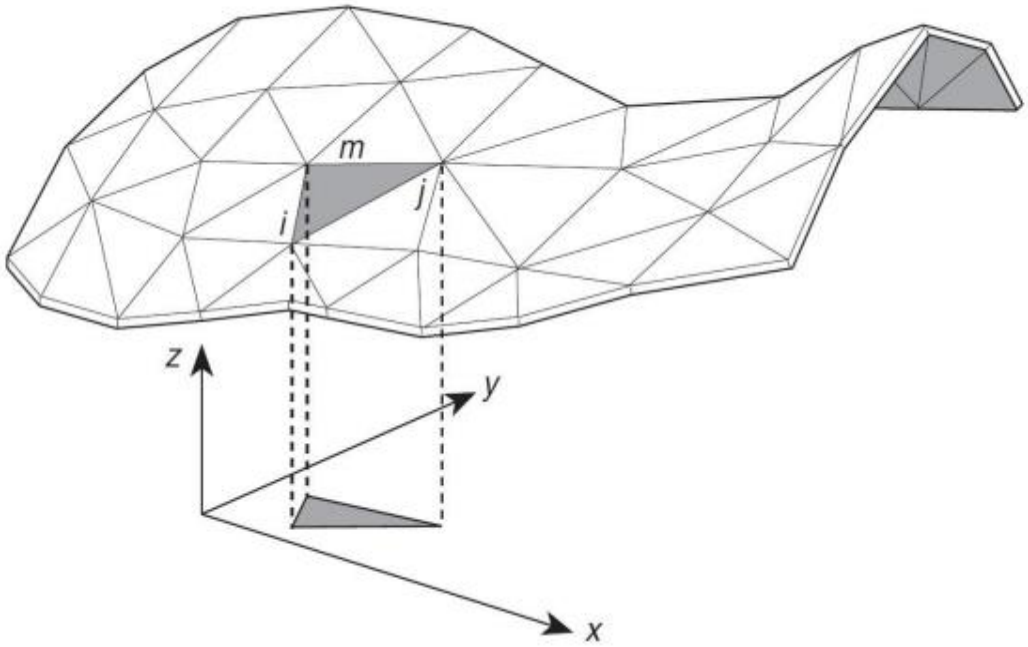
4.4. ENSAMBLE DE ELEMENTOS TRIANGULARES

Luego de tener las coordenadas globales de las matrices de rigidez K^e, los grados de libertad de cada nodo y los vectores de fuerzas de cada elemento se pueden obtener la solución del sistema de la siguiente forma:

$$\boldsymbol{K} * \boldsymbol{U} = \boldsymbol{F} \qquad \textbf{(19)}$$

Este paso consiste en ensamblar las matrices y vectores de cada uno de los elementos en un solo sistema de ecuaciones, en el que interviene la matriz de rigidez, el vector de grados de libertad y el vector de fuerzas de una estructura teniendo en cuenta la conectividad. Ensamble de una estructura tipo cascarón de forma gráfica se ve en la (***Figura 12),*** a partir de una subdivisión de elementos triangulares con sistemas de referencia *local* y a la su vez están definidos en un sistema de referencia *global*.

Figura 12 . Ensamble de Elementos Triangulares Representando una Superficie



Fuente: ZIENKIEWICZ. O.C. y TAYLOR. R.L. The Finite Element Method Volume 2, Solid Mechanics. 5 edition. [Online]. [Cited: 22 Feb. 2014] Available from Internet: <http://www.academia.edu/208845/Zienkiewicz_y_Taylor_2004_The_finite_element_method_fifth_edition._Volume_2_Solid_Mechanics.pdf>

4.5. COSENOS DIRECTORES PARA ELEMENTOS TRIANGULARES

En la (***Figura 12),*** se observa una superficie cualquiera subdividida en varios elementos triangulares orientados de forma arbitraria [12]. Las coordenadas *locales* y *globales* se pueden ver en la (***Figura 13).***

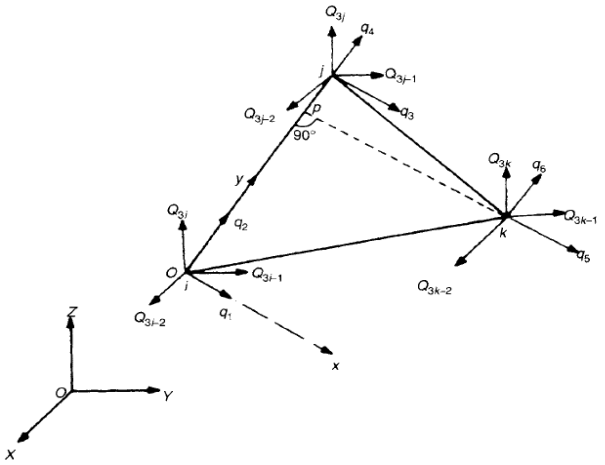
En donde la dirección de la línea $\boldsymbol{ox'}$ y $\boldsymbol{oy'}$, corresponden a los ejes *globales* $\boldsymbol{x, y, z}$, en donde la línea $\boldsymbol{oy'}$ estará orientado en la dirección del lado nodos $\boldsymbol{ij}$ del triángulo, obteniendo:

$$\boldsymbol{l_{ij} = \frac{x_j - x_i}{d_{ij}}\ ;\ m_{ij} = \frac{y_j - y_i}{d_{ij}};\ n_{ij} = \frac{z_j - z_i}{d_{ij}}} \qquad \textbf{(20)}$$

Donde la distancia entre los puntos $\boldsymbol{ij}$ corresponde a $\boldsymbol{dij}$**:**

$$\boldsymbol{d_{ij} = \left[\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2 + \left(z_j - z_i\right)^2\right]^{1/2}} \qquad \textbf{(21)}$$

Figura 13. Cosenos Directores de un Elemento Triangular



Fuente: RAO, S.S.The finite element method in engineering. Oxford: Pergamon Press, 1989. 643p

La línea ox' no se puede usar para hallar los cosenos directores a menos que conozcamos sus coordenadas, por consiguiente se trazara una línea paralela al eje local x', que denominaremos $k'p$, que va desde el nodo k' hasta un punto en la línea ij que llamaremos p como se ve en la (***Figura 12)***. Los cosenos directores de la línea ox' son los mismos de la línea $k'p$, entonces la ecuación seria:

$$l_{pk} = \frac{x_k - x_p}{d_{pk}} \; ; \; m_{pk} = \frac{y_k - y_p}{d_{pk}} ; \; n_{pk} = \frac{z_k - z_p}{d_{pk}} \qquad (22)$$

Donde la dpk' es la distancia entre los puntos $k'y\,p$, y las coordenadas del punto p en el sistema *global* (x_p, y_p, z_p), serian:

$$x_p = x_i + l_{ij} * d_{ip} \qquad (23)$$

$$y_p = y_i + m_{ij} * d_{ip} \qquad (24)$$

$$z_p = z_i + n_{ij} * d_{ip} \qquad (25)$$

Para determinar dpk' necesitamos conocer dip es decir la distancia ente el nodo i al punto p entonces:

$$d_{ip} = l_{ij} * (x_k - x_i) + m_{ij} * (y_k - y_i) + n_{ij} * (z_k - z_i) \qquad (26)$$

Pero también se debe cumplir la siguiente condición donde la suma de los cosenos directores al cuadrado del vector ij es igual a uno:

$$l_{ij}^{\,2} + m_{ij}^{\,2} + n_{ij}^{\,2} = 1 \qquad (27)$$

Donde finalmente la distancia $\boldsymbol{dpk'}$ es:

$$\boldsymbol{d_{pk} = \left(d_{ik}{}^2 - d_{ip}{}^2\right)^{\frac{1}{2}} = \left[(x_k - x_i)^2 + (y_k - y_i)^2 + (z_k - z_i)^2 - d_{ip}{}^2\right]^{1/2}} \quad \textbf{(28)}$$

Lo anterior para obtener la matriz de transformación, con seis grados de libertad por cada nodo, es decir se obtendrá una matriz de $\boldsymbol{18x18}$ de la siguiente forma [14]:

$$[\boldsymbol{\lambda}]_{\boldsymbol{18x18}} = \begin{bmatrix} [\boldsymbol{\lambda}] & [\boldsymbol{0}] & [\boldsymbol{0}] \\ [\boldsymbol{0}] & [\boldsymbol{\lambda}] & [\boldsymbol{0}] \\ [\boldsymbol{0}] & [\boldsymbol{0}] & [\boldsymbol{\lambda}] \end{bmatrix} \quad \textbf{(29)}$$

Donde cada [λ] es igual a:

$$[\boldsymbol{\lambda}]_{\boldsymbol{6x6}} = \begin{bmatrix} \boldsymbol{l_{ox}} & \boldsymbol{m_{ox}} & \boldsymbol{n_{ox}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{l_{oy}} & \boldsymbol{m_{oy}} & \boldsymbol{n_{oy}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{l_{oz}} & \boldsymbol{m_{oz}} & \boldsymbol{n_{oz}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{l_{ox}} & \boldsymbol{m_{ox}} & \boldsymbol{n_{ox}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{l_{oy}} & \boldsymbol{m_{oy}} & \boldsymbol{n_{oy}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{l_{oz}} & \boldsymbol{m_{oz}} & \boldsymbol{n_{oz}} \end{bmatrix} \quad \textbf{(30)}$$

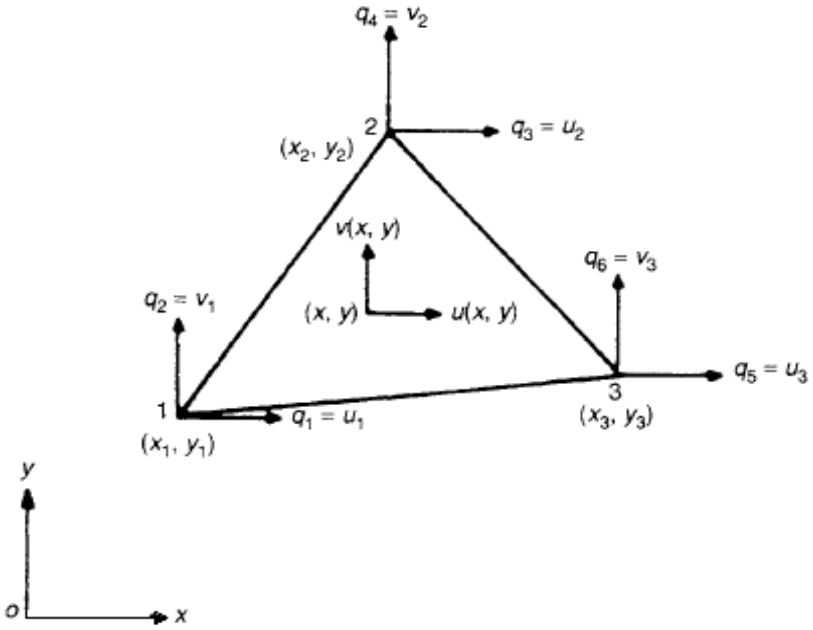
Para obtener los cosenos directores de la línea $\boldsymbol{oz'}$**,** se realiza producto cruz entre $\boldsymbol{v_{ij}}$ **y** $\boldsymbol{v_{ik}}$**,** dando como resultado un vector perpendicular al plano $\boldsymbol{v_z}$:

$$\boldsymbol{v_{ij}\ x\ v_{ik} = v_z} \quad \textbf{(31)}$$

4.6. MATRIZ DE RIGIDEZ PARA ELEMENTOS TRIANGULARES COMO MEMBRANA

El elemento de membrana triangular se considera en un plano $\boldsymbol{xy}$ de coordenadas locales como se muestra en la (***Figura 14)***.

Figura 14. Elemento de membrana Triangular



Fuente: RAO, S.S.The finite element method in engineering. Oxford: Pergamon Press, 1989. 643p

Asumiendo una variación de desplazamiento lineal en el interior del elemento, el modelo de desplazamiento se puede expresar como:

$$[\boldsymbol{K^e}]_{\boldsymbol{m}} = [\boldsymbol{Kn}^{(e)}] + [\boldsymbol{Ks}^{(e)}] \qquad \textbf{(32)}$$

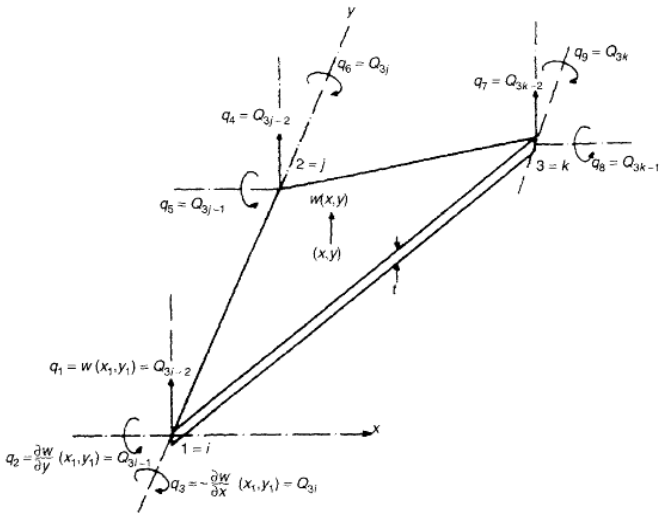
Donde la matriz $[\boldsymbol{K^e}]_{\boldsymbol{m}}$ es separada en dos partes; una la matriz debido a los esfuerzos normales, $[\boldsymbol{Kn^{(e)}}]$ y la otra debido a esfuerzos cortantes $[\boldsymbol{Ks^{(e)}}]$ Los componentes de las matrices $[\boldsymbol{Kn^{(e)}}]$ y $[\boldsymbol{Ks^{(e)}}]$ están dados por:

$$[\boldsymbol{Kn^{(e)}}] = \frac{\boldsymbol{Et}}{\boldsymbol{4A(1-v^2)}} * \begin{bmatrix} \boldsymbol{y_{32}^2} & \boldsymbol{-vy_{32}x_{32}} & \boldsymbol{-y_{32}x_{31}} & \boldsymbol{vy_{32}x_{31}} & \boldsymbol{y_{32}y_{21}} & \boldsymbol{-vy_{32}x_{21}} \\ \boldsymbol{-vy_{32}x_{32}} & \boldsymbol{x_{32}^2} & \boldsymbol{vx_{32}y_{31}} & \boldsymbol{-x_{32}x_{31}} & \boldsymbol{-vx_{32}y_{21}} & \boldsymbol{x_{32}x_{21}} \\ \boldsymbol{-y_{32}x_{31}} & \boldsymbol{vx_{32}y_{31}} & \boldsymbol{y_{31}^2} & \boldsymbol{-y_{31}x_{31}} & \boldsymbol{-y_{31}y_{21}} & \boldsymbol{y_{31}x_{21}} \\ \boldsymbol{vy_{32}x_{31}} & \boldsymbol{-x_{32}x_{31}} & \boldsymbol{-y_{31}x_{31}} & \boldsymbol{x_{31}^2} & \boldsymbol{x_{31}y_{21}} & \boldsymbol{-x_{31}x_{21}} \\ \boldsymbol{y_{32}y_{21}} & \boldsymbol{-vx_{32}y_{21}} & \boldsymbol{-y_{31}y_{21}} & \boldsymbol{x_{31}y_{21}} & \boldsymbol{y_{21}^2} & \boldsymbol{vy_{21}x_{21}} \\ \boldsymbol{-vy_{32}x_{21}} & \boldsymbol{x_{32}x_{21}} & \boldsymbol{y_{31}x_{21}} & \boldsymbol{-x_{31}x_{21}} & \boldsymbol{-vy_{21}x_{21}} & \boldsymbol{x_{21}^2} \end{bmatrix} \quad \textbf{(33)}$$

$$[\boldsymbol{Ks^{(e)}}] = \frac{\boldsymbol{Et}}{\boldsymbol{8A(1+v)}} * \begin{bmatrix} \boldsymbol{x_{32}^2} & \boldsymbol{-x_{32}y_{32}} & \boldsymbol{-x_{32}x_{31}} & \boldsymbol{x_{32}y_{31}} & \boldsymbol{x_{32}x_{21}} & \boldsymbol{x_{32}y_{21}} \\ \boldsymbol{-x_{32}y_{32}} & \boldsymbol{y_{32}^2} & \boldsymbol{y_{32}x_{31}} & \boldsymbol{-y_{32}y_{31}} & \boldsymbol{-y_{32}x_{21}} & \boldsymbol{y_{32}y_{21}} \\ \boldsymbol{-x_{32}x_{31}} & \boldsymbol{y_{32}x_{31}} & \boldsymbol{x_{31}^2} & \boldsymbol{-x_{31}y_{31}} & \boldsymbol{-x_{31}x_{21}} & \boldsymbol{x_{31}y_{21}} \\ \boldsymbol{x_{32}y_{31}} & \boldsymbol{-y_{32}y_{31}} & \boldsymbol{-x_{31}y_{31}} & \boldsymbol{y_{31}^2} & \boldsymbol{y_{31}x_{21}} & \boldsymbol{-y_{31}y_{21}} \\ \boldsymbol{x_{32}x_{21}} & \boldsymbol{-y_{32}x_{21}} & \boldsymbol{-x_{31}x_{21}} & \boldsymbol{y_{31}x_{21}} & \boldsymbol{x_{21}^2} & \boldsymbol{-x_{21}y_{21}} \\ \boldsymbol{x_{32}y_{21}} & \boldsymbol{y_{32}y_{21}} & \boldsymbol{x_{31}y_{21}} & \boldsymbol{-y_{31}y_{21}} & \boldsymbol{-x_{21}y_{21}} & \boldsymbol{y_{21}^2} \end{bmatrix} \quad \textbf{(34)}$$

4.7. MATRIZ DE RIGIDEZ PARA PLACA TRIANGULAR A FLEXIÓN

Figura 15. Grados de Libertad del Elemento Triangularen Rotaciones



Fuente: RAO, S.S.The finite element method in engineering. Oxford: Pergamon Press, 1989. 643p

En la (***Figura 15),*** se observa los desplazamientos transversales $\boldsymbol{w}$ y las rotaciones alrededor de $\boldsymbol{x}\ \boldsymbol{y}$ respectivamente, se obtendrá tres grados de libertad por cada nodo, donde la matriz de rigidez del elemento seria:

$$[\boldsymbol{K}^{(e)}] = ([\boldsymbol{N}]^{-1})^{T} \left\{ \frac{\boldsymbol{Et^3}}{\boldsymbol{12(1-v^2)}} * \iint \boldsymbol{dx.dy}\,[\boldsymbol{M}] \right\} [\boldsymbol{N}]^{-1} \qquad \textbf{(35)}$$

La matriz [N] es igual a:

$$\begin{bmatrix}
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{1} & \mathbf{0} & \boldsymbol{y_2} & \mathbf{0} & \mathbf{0} & \boldsymbol{y_2^2} & \mathbf{0} & \mathbf{0} & \boldsymbol{y_2^2} \\
\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \boldsymbol{2y_2} & \mathbf{0} & \boldsymbol{-y_2^2} & \boldsymbol{3y_2^2} \\
\mathbf{0} & \mathbf{-1} & \mathbf{0} & \mathbf{0} & \boldsymbol{-y_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{1} & \boldsymbol{x_3} & \boldsymbol{y_3} & \boldsymbol{x_3^2} & \boldsymbol{x_3y_3} & \boldsymbol{y_3^2} & \boldsymbol{x_3^3} & \boldsymbol{(x_3^2y_3 + x_3y_3^2)} & \boldsymbol{y_3^3} \\
\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \boldsymbol{x_3} & \boldsymbol{2y_3} & \mathbf{0} & \boldsymbol{(2x_3y_3 + x_3^2)} & \boldsymbol{3y_3^2} \\
\mathbf{0} & \mathbf{-1} & \mathbf{0} & \boldsymbol{-2x_3} & \boldsymbol{-y_3} & \mathbf{0} & \boldsymbol{-3x_3^2} & \boldsymbol{(-y_3^2 + 2x_3y_3)} & \mathbf{0}
\end{bmatrix}$$

Y la matriz $[\boldsymbol{M}]$ es:

$$\begin{bmatrix}
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{4} & \mathbf{0} & \boldsymbol{4v} & \boldsymbol{12x} & \boldsymbol{4(vx+y)} & \boldsymbol{12vy} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{2(1-v)} & \mathbf{0} & \mathbf{0} & \boldsymbol{4(1-v)(x+y)} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{4v} & \mathbf{0} & \mathbf{4} & \boldsymbol{12vx} & \boldsymbol{4(x+vy)} & \boldsymbol{12y} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{12x} & \mathbf{0} & \boldsymbol{12vx} & \boldsymbol{36x^2} & \boldsymbol{12x(vx+y)} & \boldsymbol{36vxy} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{4(vx+y)} & \boldsymbol{4(1-v)(x+y)} & \boldsymbol{4(x+vy)} & \boldsymbol{12x(vx+y)} & \boldsymbol{[(12-8v)(x+y)^2 - 8(1-v)xy]} & \boldsymbol{12y(x+vy)} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{12vy} & \mathbf{0} & \boldsymbol{12y} & \boldsymbol{36vxy} & \boldsymbol{12y(x+vy)} & \boldsymbol{36y^2}
\end{bmatrix}$$

La doble integral es respecto al área del elemento que se solucionara de la siguiente forma para cada valor de la matriz debido a que la integral multiplica a toda la matriz $[\boldsymbol{M}]$, el área del elemento puede ser evaluada en coordenadas globales pero para nuestro caso estará en coordenadas locales y obtener las siguientes ecuaciones:

$$\iint \boldsymbol{dx.dy} = \boldsymbol{A} = \frac{\mathbf{1}}{\mathbf{2}}\, \boldsymbol{x_3}\ \boldsymbol{y_2}$$

$$\iint x \; dx.dy = x_C \, A = \frac{1}{6} \; x_3^2 \; y_2$$

$$\iint y \; dx.dy = y_C \, A = \frac{1}{6} \, x_3 \, y_2 \, (\, y_2 \, + \, y_3)$$

$$\iint x^2 \, dx.dy = \; x_C^2 \; A + \frac{A}{12}\left[(x_i - x_C)^2 + \left(x_j - x_C\right)^2 + (x_k - x_C)^2\right] = \frac{1}{12} \; x_3^3 \; y_2$$

$$\iint x \, y \, dx.dy = y_C \; x_C \; A$$
$$+ \frac{A}{12}\left[(x_i - x_C)(y_i - y_C) + \left(x_j - x_C\right)\left(y_j - y_C\right) + (x_k - x_C)(y_k - y_C)\right]$$
$$= \frac{1}{24} \; x_3^2 \; y_2 \, (\, y_2 \, + \, 2y_3)$$

$$\iint y^2 \, dx.dy = \; y_C^2 \; A + \frac{A}{12}\left[(y_i - y_C)^2 + \left(y_j - y_C\right)^2 + (y_k - y_C)^2\right]$$
$$= \frac{1}{12} \; x_3 \; y_2 \; (\; y_2^2 + \; y_3 \; y_2 \; + \; y_3^2 \,)$$

Dónde:

$$x_C = \left(\frac{x_i \; + \; x_j + \; x_k}{3}\right) \quad ; \quad y_C = \left(\frac{y_i \; + \; y_j + \; y_k}{3}\right)$$

Donde los valores de la matriz de rigidez se multiplican por la integral del área y dando solución se obtendría:

$$\iint 4 \, dx.dy = 2 \, x_3 \, y_2$$

$$\iint 4v \, dx.dy = 2v \, x_3 \, y_2$$

$$\iint 12\, v\, x\ \, dx.dy = 2v\ x_3^2\ y_2$$

$$\iint 12\, x\ \, dx.dy = 2\, x_3^2\ y_2$$

$$\iint 12\, y\ \, dx.dy = 2\, x_3\, y_2\, (\, y_2\ +\ y_3)$$

$$\iint 12v\, y\ \, dx.dy = 2v\, x_3\, y_2\, (\, y_2\ +\ y_3)$$

$$\iint 36v\, xy\ \, dx.dy = \frac{3}{2}\ v\, x_3^2\ y_2\, (\, y_2\ +\ 2y_3)$$

$$\iint 36x^2\ dx.dy = 3\ x_3^3\ y_2$$

$$\iint 36y^2\ dx.dy = 3\ x_3\ y_2\ (\ y_2^2 +\ y_3\ y_2\ +\ y_3^2\ )$$

$$\iint 4(vx + y)\ \, dx.dy = \frac{2}{3} x_3\, y_2[x_3 v + (\, y_2\ +\ y_3)]$$

$$\iint 4(x + vy)\ \, dx.dy = \frac{2}{3} x_3\, y_2[x_3 + v(\, y_2\ +\ y_3)]$$

$$\iint 12y(x + vy)\ \, dx.dy = x_3\, y_2[\frac{1}{2} x_3(\, y_2 + 2y_3) + v(\ y_2^2 +\ y_3\ y_2\ + y_3^2\ )]$$

$$\iint 12x(vx + y)\ \, dx.dy =\ x_3^2\ y_2[vx_3 + \frac{1}{2}(\, y_2\ +\ 2y_3)]$$

$$\iint 2(1 - v)\ \, dx.dy =\ \ (1 - v)\, x_3\ \, y_2$$

$$\iint \mathbf{4}(\mathbf{1}-\boldsymbol{v})\,(\boldsymbol{x}+\boldsymbol{y})\,\boldsymbol{dx}.\boldsymbol{dy} = \frac{\mathbf{2}}{\mathbf{3}}\boldsymbol{x_3}\,\boldsymbol{y_2}(\mathbf{1}-\boldsymbol{v})[\boldsymbol{x_3}+(\,\boldsymbol{y_2}\,+\,\boldsymbol{y_3})]$$

$$\iint (\mathbf{12}-\mathbf{8}\boldsymbol{v})(\boldsymbol{x}+\boldsymbol{y})^{\mathbf{2}}-\mathbf{8}(\mathbf{1}-\boldsymbol{v})\boldsymbol{xy}\,\boldsymbol{dx}.\boldsymbol{dy}$$

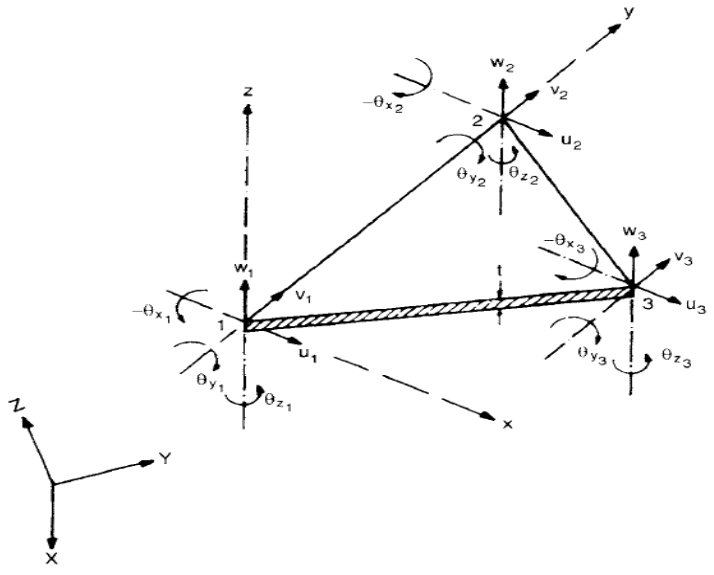
$$= (\mathbf{12}-\mathbf{8}\boldsymbol{v})\left[\frac{\boldsymbol{x_3^3}\;\boldsymbol{y_2}}{\mathbf{12}}+\frac{\mathbf{1}}{\mathbf{12}}\,\boldsymbol{x_3^2}\;\boldsymbol{y_2}(\,\boldsymbol{y_2}\,+\,\mathbf{2}\boldsymbol{y_3})\right.$$

$$\left.+\frac{\mathbf{1}}{\mathbf{12}}\;\boldsymbol{x_3}\;\boldsymbol{y_2}\;(\;\boldsymbol{y_2^2}+\;\boldsymbol{y_3}\;\boldsymbol{y_2}\;+\;\boldsymbol{y_3^2}\,)\right]-\mathbf{8}(\mathbf{1}-\boldsymbol{v})[\frac{\mathbf{1}}{\mathbf{24}}\,\boldsymbol{x_3^2}\;\boldsymbol{y_2}(\,\boldsymbol{y_2}\,+\,\mathbf{2}\boldsymbol{y_3})]$$

4.8. ENSAMBLE DE LA MATRIZ DE RIGIDEZ TOTAL

Una vez obtenidas las ecuaciones (32) y (35), el siguiente paso es ensamblar la matriz de rigidez total y así analizar todo el elemento triangular con seis grados de libertad como se muestra en la (***Figura 16)***.

Figura 16. Grados de Libertad del Elemento Triangularen sometido a fuerzas y rotaciones



Fuente: RAO, S.S.The finite element method in engineering. Oxford: Pergamon Press, 1989. 643p

En coordenadas locales esta seria la martiz de rigidez total [12]:

$$\underset{\underset{18\times18}{=}}{\tilde{K}}^{m+b}_{e} = \left[\begin{array}{c|c|c|c|c|c} \left\{\tilde{k}^{m}_{11}\right\}_{2\times2} & \begin{matrix}0&0&0\\0&0&0\end{matrix} & \begin{matrix}0\\0\end{matrix} & \left\{\tilde{k}^{m}_{12}\right\}_{2\times2} & \begin{matrix}0&0&0\\0&0&0\end{matrix} & \begin{matrix}0\\0\end{matrix} & \left\{\tilde{k}^{m}_{13}\right\}_{2\times2} & \begin{matrix}0&0&0\\0&0&0\end{matrix} & \begin{matrix}0\\0\end{matrix} \\ \hline \begin{matrix}0&0\\0&0\\0&0\end{matrix} & \left\{\tilde{k}^{b}_{11}\right\}_{3\times3} & \begin{matrix}0\\0\\0\end{matrix} & \begin{matrix}0&0\\0&0\\0&0\end{matrix} & \left\{\tilde{k}^{b}_{12}\right\}_{3\times3} & \begin{matrix}0\\0\\0\end{matrix} & \begin{matrix}0&0\\0&0\\0&0\end{matrix} & \left\{\tilde{k}^{b}_{13}\right\}_{3\times3} & \begin{matrix}0\\0\\0\end{matrix} \\ \hline 0\ 0 & 0\ 0\ 0 & 0 & 0\ 0 & 0\ 0\ 0 & 0 & 0\ 0 & 0\ 0\ 0 & 0 \\ \hline \left\{\tilde{k}^{m}_{21}\right\}_{2\times2} & \begin{matrix}0&0&0\\0&0&0\end{matrix} & \begin{matrix}0\\0\end{matrix} & \left\{\tilde{k}^{m}_{22}\right\}_{2\times2} & \begin{matrix}0&0&0\\0&0&0\end{matrix} & \begin{matrix}0\\0\end{matrix} & \left\{\tilde{k}^{m}_{23}\right\}_{2\times2} & \begin{matrix}0&0&0\\0&0&0\end{matrix} & \begin{matrix}0\\0\end{matrix} \\ \hline \begin{matrix}0&0\\0&0\\0&0\end{matrix} & \left\{\tilde{k}^{b}_{21}\right\}_{3\times3} & \begin{matrix}0\\0\\0\end{matrix} & \begin{matrix}0&0\\0&0\\0&0\end{matrix} & \left\{\tilde{k}^{b}_{22}\right\}_{3\times3} & \begin{matrix}0\\0\\0\end{matrix} & \begin{matrix}0&0\\0&0\\0&0\end{matrix} & \left\{k^{b}_{23}\right\}_{3\times3} & \begin{matrix}0\\0\\0\end{matrix} \\ \hline 0\ 0 & 0\ 0\ 0 & 0 & 0\ 0 & 0\ 0\ 0 & 0 & 0\ 0 & 0\ 0\ 0 & 0 \\ \hline \left\{\tilde{k}^{m}_{31}\right\}_{2\times2} & \begin{matrix}0&0&0\\0&0&0\end{matrix} & \begin{matrix}0\\0\end{matrix} & \left\{\tilde{k}^{m}_{32}\right\}_{2\times2} & \begin{matrix}0&0&0\\0&0&0\end{matrix} & \begin{matrix}0\\0\end{matrix} & \left\{\tilde{k}^{m}_{33}\right\}_{2\times2} & \begin{matrix}0&0&0\\0&0&0\end{matrix} & \begin{matrix}0\\0\end{matrix} \\ \hline \begin{matrix}0&0\\0&0\\0&0\end{matrix} & \left\{\tilde{k}^{b}_{31}\right\}_{3\times3} & \begin{matrix}0\\0\\0\end{matrix} & \begin{matrix}0&0\\0&0\\0&0\end{matrix} & \left\{\tilde{k}^{b}_{32}\right\}_{3\times3} & \begin{matrix}0\\0\\0\end{matrix} & \begin{matrix}0&0\\0&0\\0&0\end{matrix} & \left\{\tilde{k}^{b}_{33}\right\}_{3\times3} & \begin{matrix}0\\0\\0\end{matrix} \\ \hline 0\ 0 & 0\ 0\ 0 & 0 & 0\ 0 & 0\ 0\ 0 & 0 & 0\ 0 & 0\ 0\ 0 & 0 \end{array}\right]$$

Donde $[\boldsymbol{K^e}]_{\boldsymbol{m}}$ es la matriz tipo membrana de rigidez de fuerzas y desplazamientos nodales que se puede denotar de una forma simplificada en coordenadas locales como:

$$[\boldsymbol{K^e}]_{\underset{6x6}{m}} = \begin{bmatrix} [\boldsymbol{k}_{\underset{2x2}{11}}]_{\boldsymbol{m}} & [\boldsymbol{k}_{\underset{2x2}{12}}]_{\boldsymbol{m}} & [\boldsymbol{k}_{\underset{2x2}{13}}]_{\boldsymbol{m}} \\ [\boldsymbol{k}_{\underset{2x2}{21}}]_{\boldsymbol{m}} & [\boldsymbol{k}_{\underset{2x2}{22}}]_{\boldsymbol{m}} & [\boldsymbol{k}_{\underset{2x2}{23}}]_{\boldsymbol{m}} \\ [\boldsymbol{k}_{\underset{2x2}{31}}]_{\boldsymbol{m}} & [\boldsymbol{k}_{\underset{2x2}{32}}]_{\boldsymbol{m}} & [\boldsymbol{k}_{\underset{2x2}{33}}]_{\boldsymbol{m}} \end{bmatrix} \qquad \textbf{(36)}$$

Y las submatrices $[\boldsymbol{k_{ij}}]_{\boldsymbol{m}}$ corresponden a la rigidez asociada a los coeficientes con el nodo $\boldsymbol{i}$ y $\boldsymbol{j}$, el subíndice $\boldsymbol{m}$ es usado para indicar las acciones que tiene en elemento es decir que el elemento está sometido a fuerzas y desplazamientos nodales se puede escribir de la siguiente forma:

$$\begin{Bmatrix} \boldsymbol{P_{x1}} \\ \boldsymbol{P_{y1}} \\ \boldsymbol{P_{x2}} \\ \boldsymbol{P_{y2}} \\ \boldsymbol{P_{x3}} \\ \boldsymbol{P_{y3}} \end{Bmatrix} = [\boldsymbol{K^e}]_{\boldsymbol{m}} = \begin{Bmatrix} \boldsymbol{u_1} \\ \boldsymbol{v_1} \\ \boldsymbol{u_2} \\ \boldsymbol{v_2} \\ \boldsymbol{u_3} \\ \boldsymbol{v_3} \end{Bmatrix} \qquad \textbf{(37)}$$

En la matriz de rigidez total también aparece $[\boldsymbol{K^e}]_{\boldsymbol{b}}$ que representa la matriz de rotaciones o resistente a flexión, que se denota por el subíndice $\boldsymbol{b}$ que se puede denotar de forma simplificada en coordenadas locales de la siguiente forma:

$$[\boldsymbol{K^e}]_{\underset{9x9}{\boldsymbol{b}}} = \begin{bmatrix} [\boldsymbol{k}_{\underset{3x3}{11}}]_{\boldsymbol{b}} & [\boldsymbol{k}_{\underset{3x3}{12}}]_{\boldsymbol{b}} & [\boldsymbol{k}_{\underset{3x3}{13}}]_{\boldsymbol{b}} \\ [\boldsymbol{k}_{\underset{3x3}{21}}]_{\boldsymbol{b}} & [\boldsymbol{k}_{\underset{3x3}{22}}]_{\boldsymbol{b}} & [\boldsymbol{k}_{\underset{3x3}{23}}]_{\boldsymbol{b}} \\ [\boldsymbol{k}_{\underset{3x3}{31}}]_{\boldsymbol{b}} & [\boldsymbol{k}_{\underset{3x3}{32}}]_{\boldsymbol{b}} & [\boldsymbol{k}_{\underset{3x3}{33}}]_{\boldsymbol{b}} \end{bmatrix} \qquad \textbf{(38)}$$

Para el análisis de estructuras tridimensionales en el plano deben combinarse las rotaciones de acuerdo con la siguiente observación:

- Para desplazamientos pequeños, el plano y las rotaciones están desacopladas, en el plano la rotación $\boldsymbol{\theta z}$ no es necesaria para un solo elemento, sin embargo $\boldsymbol{\theta z}$ esta conjugada de la fuerza $\boldsymbol{Mz}$ que está considerada en el análisis [12].

5. VALIDACIÓN DEL MEF PROGRAMADO EN MATLAB

En este paso se realizó la validación del método de elementos finitos para estructuras tipo cascarón programado en MATLAB, donde se realizó por aparte el análisis de placas.

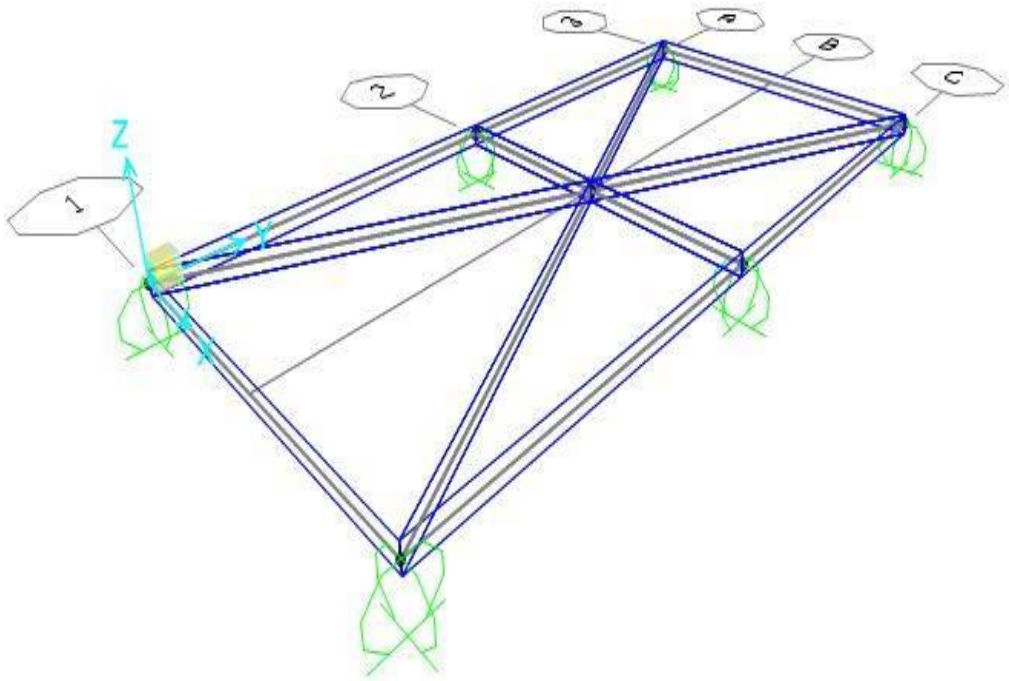
El primer ejemplo se desarrolló por medio del método análisis de placas triangulares a flexión y el segundo por el método de análisis de elementos triangulares como membrana que se evidenciaran en este numeral respectivamente.

5.1. ANÁLISIS DE PLACAS TRIANGULARES A FLEXIÓN

Para esto, se modelo una placa plana sencilla en el software de análisis estructural SAP2000, con unas determinadas dimensiones y características del material, sin cargas externas teniendo en cuenta únicamente el peso propio ver (***Figura 17)***, las especificaciones de placa son las siguientes:

Propiedades del Material:	**Propiedades del Elemento:**	**Restricciones:**
Módulo de elasticidad E= 2,1538e10 N/m^2 *Peso específico del concreto* P= 23544 N/m^3 *Coeficiente Poisson* v= 0,2	*Largo* b= 2m *Ancho* a= 1m *Espesor:* t= 0,05m	*Simplemente apoyado nodos* (1, 2, 3, 4, 5 y 6)

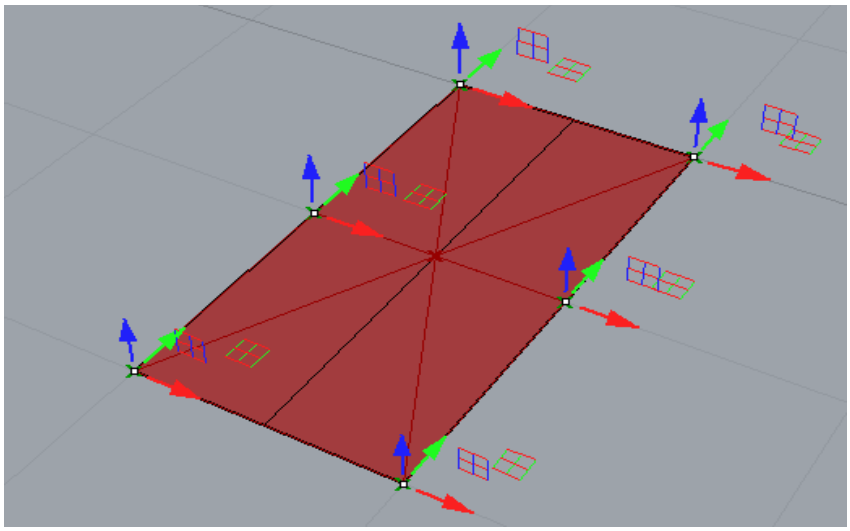
Figura 17. Modelo en SAP2000 de una Placa sometida a flexión



Fuente: Imagen tomada de SAP2000

En la (***Figura 18),*** se puede observar la placa plana modelada por el software de RHINOCEROS 3D, también se puede observar los apoyos o restricciones de la placa.

Figura 18. Modelo en Rhinoceros 3D de una Placa sometida a flexión



Fuente: Imagen tomada de Rhinoceros 3D

Los resultados obtenidos del análisis fueron los siguientes:

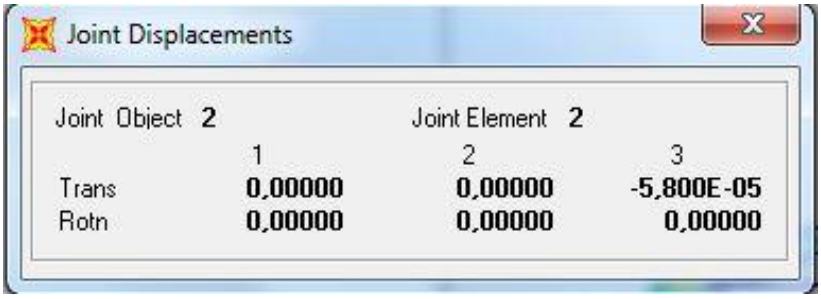
- Deformaciones según el software SAP2000, en la (***Figura 19)*** *se muestra la deformación que es igual a:*

δz=- 5,8e-5 m

θx= 0 rad

θy= 0 rad

Figura 19. Modelo en SAP2000 de una Placa sometida a flexión (deformaciones)



Fuente: Imagen tomada de SAP2000

- Deformaciones según el Método de Elementos Finitos programado en el Software MATLAB 7.0 que se muestra en la (***Figura 20)*** *se muestra la deformación que es igual a:*
-

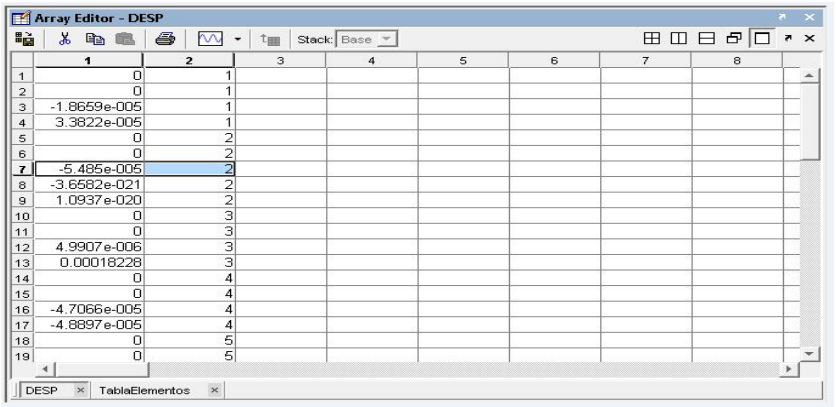
δz= -5,485e-5 m

θx= -3,6582e-21 rad

θy= -1,0937e-20 rad

- Esta deformación máxima está ubicada en el centro de la placa es causada por el peso propio del elemento.

Figura 20. Resultados del Método de Elementos Finitos programado en MATLAB 7.0 de una Placa sometida a flexión (deformaciones)



	1	2	3	4	5	6	7	8
1	0	1						
2	0	1						
3	-1.8659e-005	1						
4	3.3822e-005	1						
5	0	2						
6	0	2						
7	-5.485e-005	2						
8	-3.6582e-021	2						
9	1.0937e-020	2						
10	0	3						
11	0	3						
12	4.9907e-006	3						
13	0.00018228	3						
14	0	4						
15	0	4						
16	-4.7066e-005	4						
17	-4.8897e-005	4						
18	0	5						
19	0	5						

Fuente: Imagen tomada de SAP2000

- El cuadro de azul de la (***Figura 20)*** hace referencia a la deformación máxima del elemento.

- En la siguiente tabla se puede apreciar la diferencia de error en porcentaje de los resultados obtenidos por medio de “SAP2000” y “Matlab7.0”:

Tabla 1. *Resultados de las deformaciones para una placa sometida a flexión*

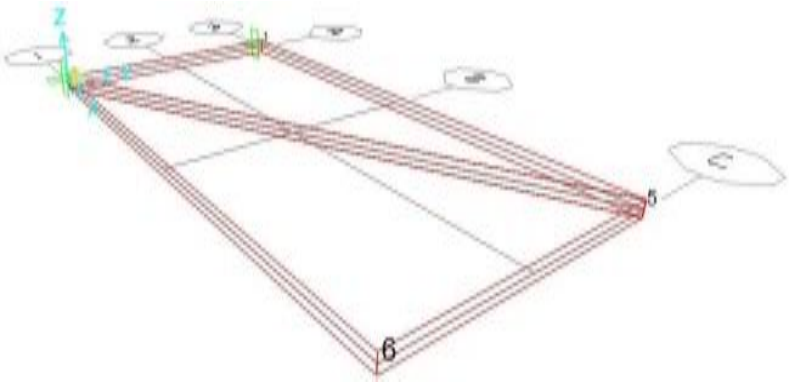
	Deformaciones [m]		
Reacciones	**SAP2000**	**Matlab7.0**	**Diferencia**
δz	-5,80E-05	-5,49E-05	5,37%
θx	0,00E+00	-3,66E-21	0,00%
θy	0,00E+00	-1,09E-20	0,00%

5.2. ANÁLISIS DE PLACAS TRIANGULARES COMO MEMBRANA

Para esto ejemplo, se modelo una placa plana sencilla como membrana en el software de análisis estructural SAP2000 (***Figura 21)***, con unas determinadas dimensiones, características del material y aplicación de cargas, las especificaciones de placa son las siguientes:

Propiedades del material:	**Propiedades del elemento:**	**Restricciones:**
Módulo de elasticidad E= 2,1538e10 N/m^2 Peso específico del concreto P= 23544 N/m^3 Coeficiente Poisson v= 0,2	Largo b= 2m Ancho a= 1m Espesor: t= 0,05m	Empotrado en dos extremos **Cargas** Nodo 5: 1000 N en direccion x Nodo 6: 1000 N en direccion x

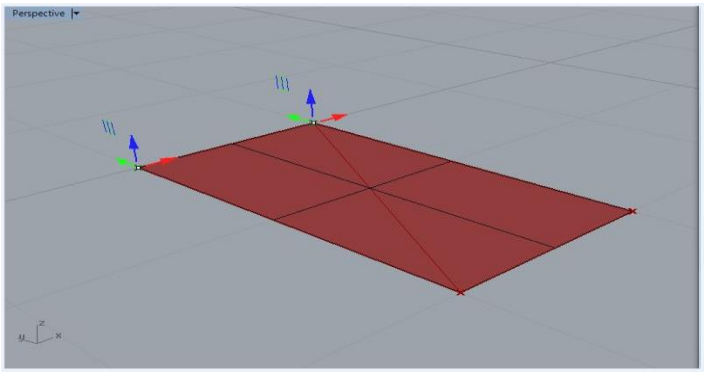
Figura 21. Modelo en SAP2000 de una Placa como membrana



Fuente: Imagen tomada de SAP2000

En la (***Figura 22),*** se puede observar la placa plana modelada como membrana por el software de RHINOCEROS 3D, también se puede observar los apoyos o restricciones de la placa

Figura 22. Modelo en RHINOCEROS 3D de una Placa como membrana



Fuente: Imagen tomada de Rhinoceros 3D

Los resultados obtenidos del análisis fueron los siguientes:

- ✓ Deformaciones según el software SAP2000, en la (***Figura 23)*** *se muestra las deformaciones igual a:*

δ5x= 3,561e-6 m

δ5y= 0 m

δ6x= 3,717e-6 m

δ6y= 0 m

Figura 23. Resultados SAP2000 de una Placa sometida como Membrana (deformaciones)

Fuente: Imagen tomada de SAP2000

- ✓ Deformaciones según el Método de Elementos Finitos programado en el Software MATLAB 7.0 con las mismas características de material, restricciones y cargas ver (***Figura 24):***

δ5x= 3,55e-6 m

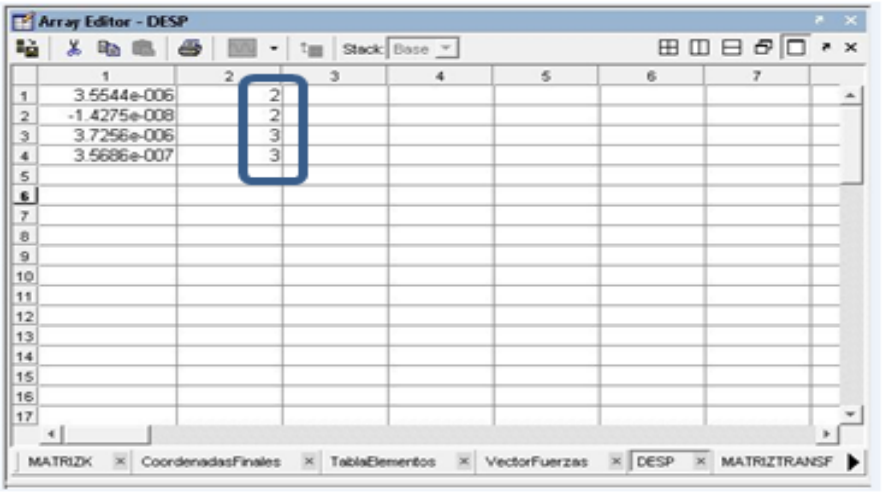
δ5y= -1,427e-8 m

δ6x= 3,72e-6 m

δ6y= 3,56e-7 m

Estas deformaciones están ubicadas en los nodos 5(2) y 6(3) de la placa (***Figura 21).***

Figura 24. Resultados del Método de Elementos Finitos programado en MATLAB 7.0 de una Placa sometida como membrana (deformaciones)



Fuente: Imagen tomada de SAP2000

- La (***Figura 24),*** el cuadro azul hace referencia a los nodos donde se aplica la carga.

- En la siguiente tabla se puede apreciar la diferencia de error en porcentaje de los resultados obtenidos por medio de “SAP2000” y “Matlab7.0”:

Tabla 2. Resultados de las deformaciones para una placa sometida como membrana

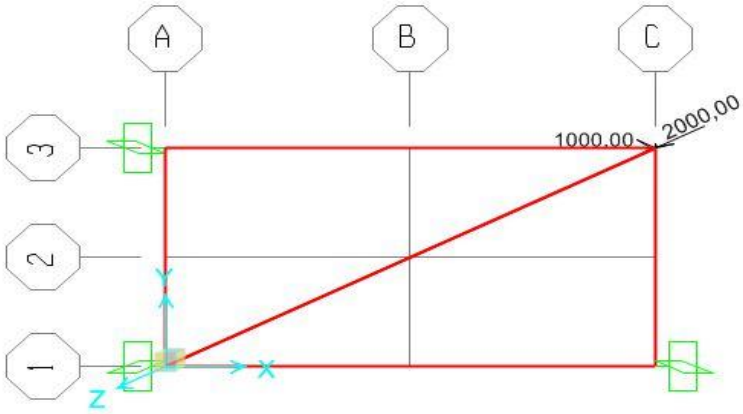
		Deformaciones [m]		
Fuerza aplicada en Dos Extremos	**Reacciones**	**SAP2000**	**Matlab7.0**	**Diferencia**
1	**δx**	3,56E-06	3,55E-06	0,31%
	δy	0,00E+00	-1,43E-08	0,00%
2	**δx**	3,72E-06	3,72E-06	0,08%
	δy	0,00E+00	3,56E-07	0,00%

5.3. ANÁLISIS REALIZANDO LA COMBINACIÓN DE MEMBRANA Y FLEXIÓN (SHELL) PARA UNA PLACA

Para esto, se modelo una placa plana sencilla en el software de análisis estructural SAP2000, con unas determinadas dimensiones, características del material y de aplicación de cargas (***Figura 25)***, las especificaciones de placa son las siguientes:

Propiedades del material:	**Propiedades del elemento:**	**Restricciones:**
Módulo de elasticidad E= 2,1538e10 N/m^2 Peso específico del concreto P= 23544 N/m^3 Coeficiente Poisson v= 0,2	Largo b= 2m Ancho a= 1m Espesor: t= 0,05m	Empotrado en tres extremos **Cargas** Nodo 5: 1000 N en direccion x y -2000N en direccion -z

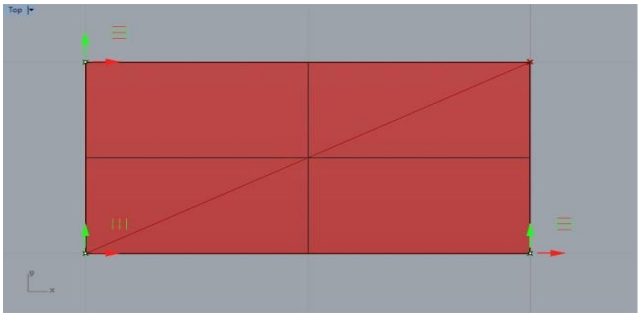
Figura 25. Modelo en SAP2000 de una Placa sometida como membrana y flexión (Shell)



Fuente: Imagen tomada de SAP2000

En la (***Figura 26),*** se puede observar la placa plana modelada por el software de RHINOCEROS 3D, también se puede observar los apoyos o restricciones de la placa.

Figura 26. Modelo en Rhinoceros 3D de una Placa sometida como membrana y flexión (Shell)



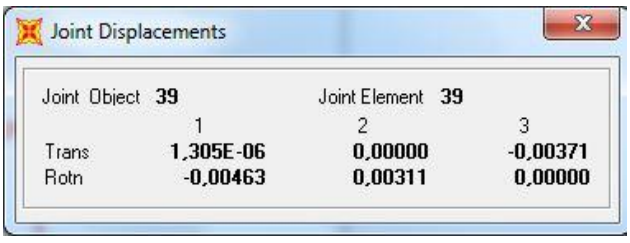
Fuente: Imagen tomada de Rhinoceros 3D

Los resultados obtenidos del análisis fueron los siguientes:

- Deformaciones según el software SAP2000, en la (***Figura 27),*** se muestra la deformación que es igual a:
-

δ5x= 1,305e-6 m
δ5y= 0 m
δ5z= -3,71e-3 m
θ5x= -4,63e-3 rad
θ5y= 3,11e-3 rad

Figura 27. Resultado SAP2000 de una Placa sometida como membrana y flexión (deformaciones)

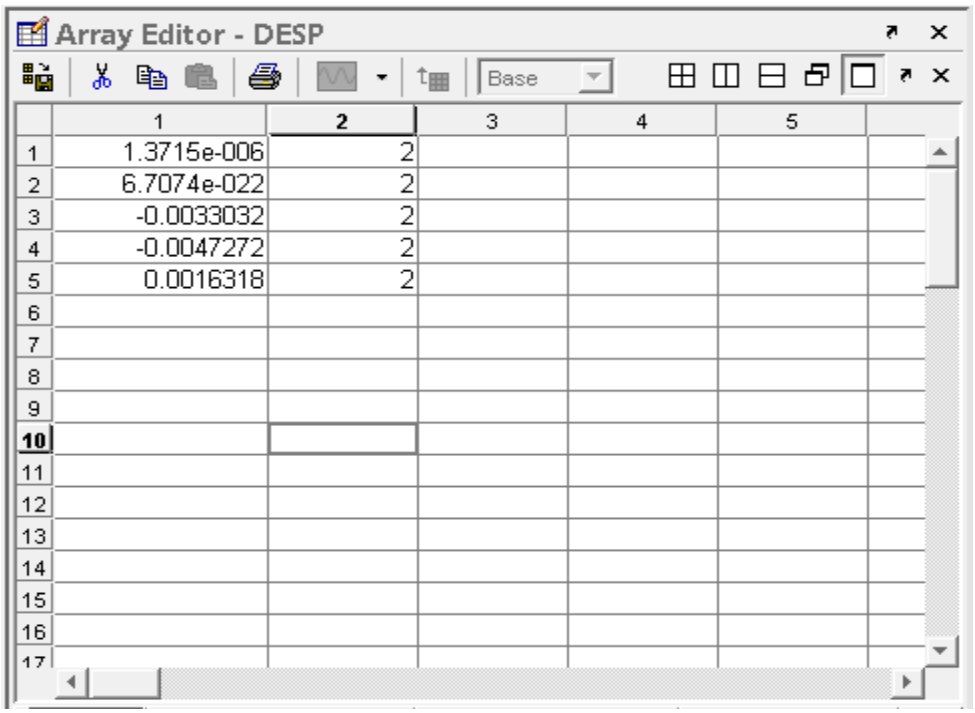


Fuente: Imagen tomada de SAP2000

- ✓ Deformaciones según el Método de Elementos Finitos programado en el Software MATLAB 7.0 con las mismas características de material, restricciones y cargas ver (***Figura 28):***

δ2x= 1,3715e-6 m
δ2y= 6,70745e-22 m
δ2z= -3,3032e-3 m
θ2x= -4,7272e-3 rad
θ2y= 1,6318e-3 rad

Figura 28. Resultados del Método de Elementos Finitos programado en MATLAB 7.0 de una Placa sometida como membrana y flexión (deformaciones)



	1	2	3	4	5
1	1.3715e-006	2			
2	6.7074e-022	2			
3	-0.0033032	2			
4	-0.0047272	2			
5	0.0016318	2			
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					

Fuente: Imagen tomada de SAP2000

- En la siguiente tabla se puede apreciar la diferencia de error en porcentaje de los resultados obtenidos por medio de “SAP2000” y “Matlab7.0”:

Tabla 3. Resultados de las deformaciones para una placa sometida como membrana y flexión (Shell)

		Deformaciones [m] y Rotaciones[rad]		
Fuerza aplicada en un Extremo	**Reacciones**	**SAP2000**	**Matlab7.0**	**Diferencia**
1	**δx**	1,31E-06	1,37E-06	5,10%
	δy	0,00E+00	6,71E-22	0,00%
	δz	-3,71E-03	-3,30E-03	10,96%
	θx	-4,63E-03	-4,73E-03	2,06%
	θy	3,11E-03	1,63E-03	47,53%

6. VALIDACION DEL ALGORITMO MUPSO

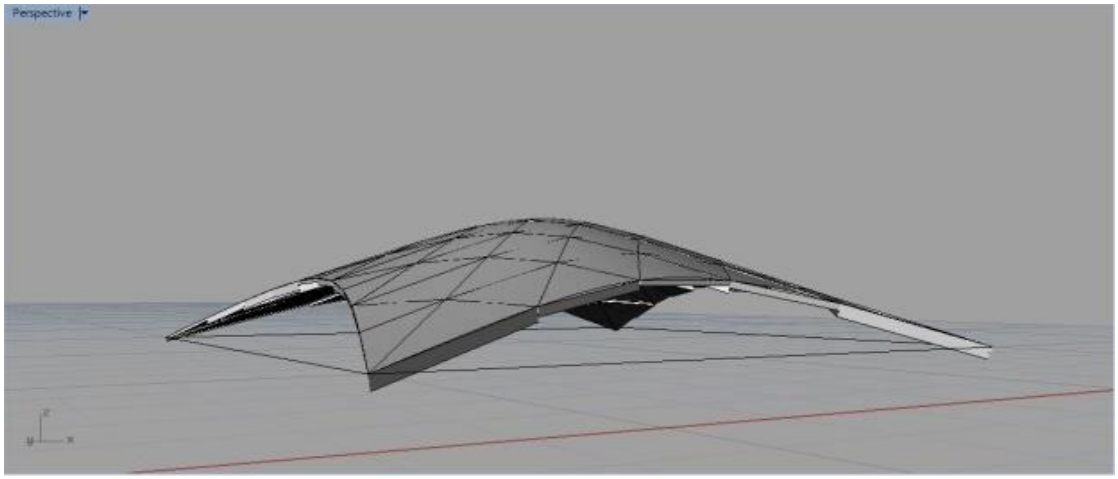
Para la optimización de estructuras tipo cascaron se utilizó el algoritmo de optimización “MUPSO”, el cual se encuentra validado en el proyecto de maestría titulado “Computación Evolutiva Aplicada al Diseño de Estructuras Reticulares” realizado por el ingeniero Leonardo Moreno de Luca de la Universidad Industrial de Santander en el año 2014.

Este algoritmo se programó en combinación con el método de elementos finitos en Matlab para así realizar el proceso de optimización de estructuras tipo cascaron.

6.1. EJEMPLO DE OPTIMIZACIÓN PARA ESTRUCTURAS TIPO CASCARON

A continuación se desarrolló un ejemplo de optimización de estructuras tipo cascaron, realizando un proceso de 5 corridas, cada una con 50 iteraciones con el fin de determinar los mejores resultados. En la (***Figura 29),*** se puede observar una cubierta con 2,25m de alto, 10m de largo y 10m de ancho con los mejores espesores después de realizada la optimización del método de elementos finitos para estructuras tipo cascaron (MUPSO).

Figura 29. Resultados del Método de Elementos Finitos programado



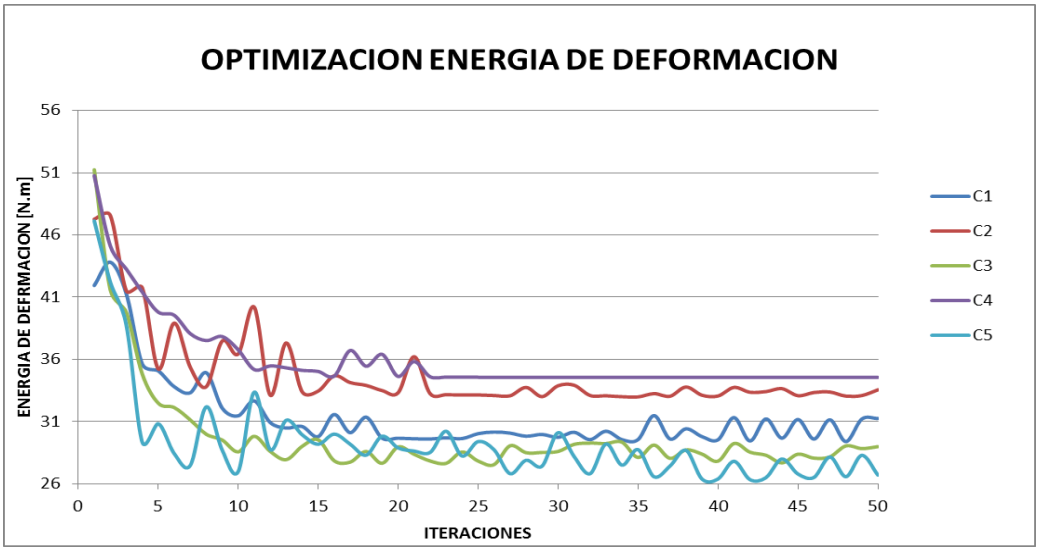
Fuente: Imagen tomada de Rhinoceros 3D

El rango de variación en el proceso de optimización para el espesor de la cubierta se definió entre cinco centímetros y veinticinco centímetros, obteniéndose espesores variables dentro de este, las funciones Fo1 y Fo2 de la tabla 1 hacen referencia a los mejores valores de energía de deformación y peso respectivamente obtenidos en la estructura.

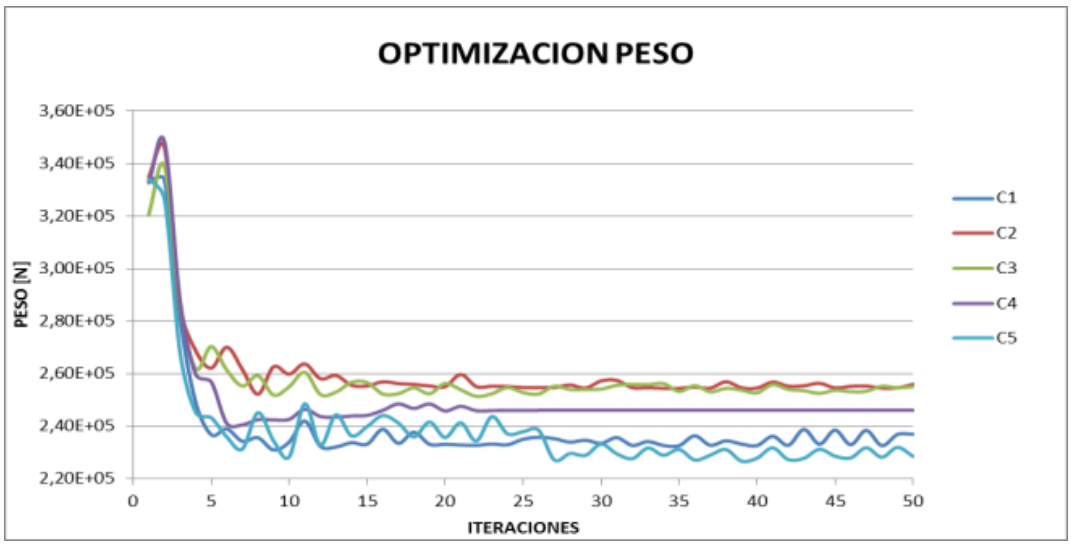
Tabla 4. Resultados de la optimización (MUPSO)

Corridas	**Energía Deformación [N.m] "Fo1"**	**Peso [N] "Fo2"**
1	29,3952	2,2365e5
2	32,9861	2,5446e5
3	27,5440	2,5234e5
4	34,5583	2,4609e5
5	26,3374	2.2734e5

Grafica 1. Resultados de la optimización de energía de deformación para 5 corridas, cada una de 50 iteraciones (donde C1, C2, C3, C4, C5, hacen referencia las cinco corridas en el proceso de optimización).



Grafica 2. Resultados de la optimización del peso para 5 corridas, cada una de 50 iteraciones (donde C1, C2, C3, C4, C5, hacen referencia las cinco corridas en el proceso de optimización).



Con el objetivo de realizar una comparación se corrió la misma cubierta del ejemplo anterior con el método de elementos finitos programado, variando únicamente el espesor, tomando un espesor constante de 10 cm obteniendo los siguientes resultados de Peso y Energía de Deformación:

- Peso: 1.9202 e5 N
- Energía de Deformación: 158.9965 N.m

6.2. DIAGRAMA DE OPTIMIZACIÓN DE ESTRUCTURAS TIPO CASCARON

El ***Diagrama 1*** (ver anexos) hace referencia al proceso Iterativo de optimización para Estructuras tipo Cascaron programado en Matlab.

7. CONCLUSIONES

- En el proceso de optimización Multi-objetivo (MUPSO) para estructuras tipo cascarón fue posible introducir la programación del método de elementos finitos (MEF) en Matlab 7.0, capaz de generar de una forma sencilla la geometría teniendo en cuenta los puntos de los nodos de la estructura y el número de divisiones para la evaluación total de la cubierta, que con la programación realizada se obtuvo la energía de deformación y el peso de la cubierta. Vale la pena resaltar que con respecto a los valores usados para los parámetros del algoritmo de optimización (UPSO) “C1, C2, X y U”, fueron tomados del proyecto de investigación en maestría titulado “Computación Evolutiva Aplicada al Diseño de Estructuras Reticuladas” realizada por el Ingeniero civil Leonardo Moreno de Luca de la Universidad Industrial de Santander en el año 2014, que aplica para este caso en el que se optimiza la relación entre “Energía-Peso”. Este proceso de optimización se describe paso a paso en el numeral 6.2.

- Vale la pena resaltar el uso del Software llamado Rhinoceros 3D que en combinación con Matlab 7.0 se logró obtener los datos para así realizar la programación del método de elementos finitos para estructuras tipo cascarón, en el numeral 2 , 3, 4, se habla de las bases para cumplir a cabalidad con la realización de la programación del método. En este sentido se logró calcular para cualquier tipo de cubierta las deformaciones, la energía de deformación y el peso de la misma.

- En la sección correspondiente al marco teórico, se presenta la teoría básica necesaria para realizar un primer acercamiento a la utilización del método de elementos finitos para estructuras tipo cascaron o tipo “Shell”,

presentando en forma generalizada las matrices correspondientes a las placas tipo membrana y placas resistentes a esfuerzos a flexión, así como su respectivo ensamble para formar la matriz de rigidez que me define una estructura tipo cascarón. Es necesario para una correcta implementación del método de elementos finitos tener claro cada concepto de la teoría, empezando por una correcta enumeración tanto de cada uno de los nodos como elementos “triangulares”, la ubicación de las coordenadas locales de cada elemento y globales de la estructura, la obtención de los cosenos directores y las correspondientes matrices de transformación de coordenadas, así igualmente tener claro los valores que definen las matrices como son las coordenadas geométricas de los nodos que conforman cada elemento, su área, espesor y propiedades del material, es muy importante también el proceso de ensamble ya que es muy fácil cometer errores si no se tiene cuidado de realizar una correcta enumeración de los nodos y de revisar cuidadosamente la conectividad de estos con cada uno de los elementos ya sean cuadrados, rectangulares o como en es este caso fueron utilizados triangulares que conforman la totalidad de la superficie de la cubierta en estudio. Tener en cuenta los numerales 1, 2, 4.

- Los programas computacionales de análisis estructural son herramientas muy poderosas para obtener la solución de problemas complejos o de análisis largos y dispendiosos como lo es el método de elementos finitos, es cierto que es muy importante saber manejar estos software, pero así mismo también es importante entender que hacen, como lo hacen y que métodos de solución emplean. Gracias a este proyecto de investigación se profundizo más en el tema al tener la necesidad de programar el método de elementos finitos partiendo desde cero, logrando así entender varios conceptos y afianzando conocimientos. Así mismo se comprendió un poco

más el funcionamiento de una herramienta muy utilizada para el análisis estructural como lo es SAP2000 la cual se basa en la teoría clásica de Timoshenko y Kirchhoff, teoría utilizada para la programación del método de elementos finitos para estructuras tipo cascarón.

- Se desarrolló correctamente el procedimiento computacional para llevar a cabo el análisis estructural y la optimización de estructuras tipo cascarón, implementando el método de elementos finitos en combinación con el algoritmo de optimización Multi-objetivo (MUPSO), ambas programadas en Matlab. Para la aplicación de este algoritmo se desarrolló el ejemplo de optimización de una cubierta tipo cascarón ver (***Figura 29),*** en la cual se realizó el proceso de cinco corridas cada una de estas con cincuenta iteraciones obteniendo así los mejores valores en cuanto a Energía-Peso ver (***Tabla 3).*** Como se puede observar en las gráficas de optimización de energía de deformación ver (***Grafica 1)*** y optimización de peso ver (***Grafica 2)*** , donde se graficó cada valor obtenido en cada corrida y en cada una de las iteraciones los resultados obtenidos son optimizados y tienden a converger minimizando tanto el peso total de la estructura así como la energía de deformación de esta, dando como resultado una estructura optima desde el punto de vista estructural y arquitectónico siendo a la vez llamativo a la vista, así como también obteniéndose una estructura optima reduciendo la cantidad de material necesaria para su construcción.

CITAS BIBLIOGRAFICAS

[1] *K. a. M. Parsopoulos, «UPSO: A Unified Particle Swarm Optimization Scheme,» de Lecture Series on Computer and Computational Sciences, Vol. 1, pp. 868-873., 2004.*

[2] *J. O. Geckeler, de Statics of elastic bodies, 1932.*

[3] *Arón, «´´Shell equations in curvilinear coordinates´´,» 1874.*

[4] *R. Clough, « "The finite element in plane stress analysis"».*

[5] *Finsterwalder, de Teoría general de láminas cilíndricas., 1935.*

[6] *E. Freyssinet, de Paraboloides elípticos , 1916.*

[7] *P. L. Nervi, «Elementos laminares prefabricados,» 1949.*

[8] *E. C.-Z.-P. Jena, «. Planetario más viejo del mundo,» Alemania, estado de Turingia . , 1926.*

[9] *H. Berger, « Structures of Light, the art and engineering of tensile Architecture.,» Boston, 1996.*

[10] *Félix Candela, «´´ Estructuras basadas en el uso extensivo del paraboloide hiperbólico´´,» Instituto Juan de Herrer Madrid, 1994.*

[11] *H. ISLER'S, « INFINITE SPECTRUM OF NEW SHAPES FOR SHELLS,» United Kingdom, Universidad Politécnica de Valencia, Noviembre 25 del 2009.*

[12] *S. RAO, «The Element Method in Engineering,» 1989.*

[13] H. Berger, «STRUCTURAL FORM IN ARCHITECTURE; domes arches and shells,» Enero 2008.

[14] Z. O.C., « Taylor R.L. The Finite Element Method Volume 2: Solid Mechanics. 5 edition».

[15] SERGIO GALLEGOS CÁZERES, de Análisis de sólidos y estructural mediante el método de elementos finitos, Editorial LIMUSA SA , 2008.

[16] Z. O.C., «The Finite Element Method Edition 1,» de The Finite Element Method Edition1: The Basis.by Zienkiewicz O.C..

[17] A. Tomás, «Shape and size optimisation of concrete shells».

[18] N. Newmark, « "Numerical methods of analysis in bars, plates and elastic bodies", in Numerical Methods in Analysis in Engineering».

[19] U. o. C. (UNICAMP), «COMPUTATIONAL GENERATION OF FREE-FORM SHELLS IN ARCHITECTURAL DESIGN AND CIVIL ENGINEERING,» Campinas SP,Brazil, Septiembre 9 del 2010.

[20] H. ISLER, de 50 YEARS OF "NEW SHAPES FOR SHELLS, "JOURNAL OF THE INTERNATIONAL ASSOCIATION FOR SHELL AND SPATIAL STRUCTURES; formerly bulletin of the international association for shell and spatial structures. Guest Editors: J. F. Abel and J. C. Chilton;, 3 de septimebre del 2011.

[21] H. Isler, « "The engineer's contribution to contemporary architecture",» Editorial: thomas telford, 2000.

BIBLIOGRAFIA

BERGER, Horst. Structural form in architecture**;** domes arches and shells. Structure Magazine. [Online] November, 2007. [Cited: 19 Feb. 2014] Available from Internet: http://www.ciccp.es/ImgWeb/Castilla%20y%20Leon/Ingenier%C3%ADaHumanismo/Structural%20Form%20Architecture.pdf

ISLER, Heinz – 50 Years of "New Shapes for Shells"**;** journal of the international association for shell and spatial structures**;** formerly bulletin of the international association for shell and spatial structures. . [Online]. [Cited: 22 Feb. 2014] Available from Internet: http://www.schwartz.arch.ethz.ch/Publikationen/Dokumente/Isler.pdf

ISLER, Heinz. INFINITE SPECTRUM OF NEW SHAPES FOR SHELLS**;** Nottingham Trent University School of Architecture Design and Built Environment, Nottingham Trent University. Universidad Politécnica de Valencia. [Online] Noviembre 25 de 2009. [Cited: 15 Feb. 2014] Available from Internet: http://riunet.upv.es/bitstream/handle/10251/6465/PAP_CHILTON_51.pdf?sequence=1&isAllowed=y

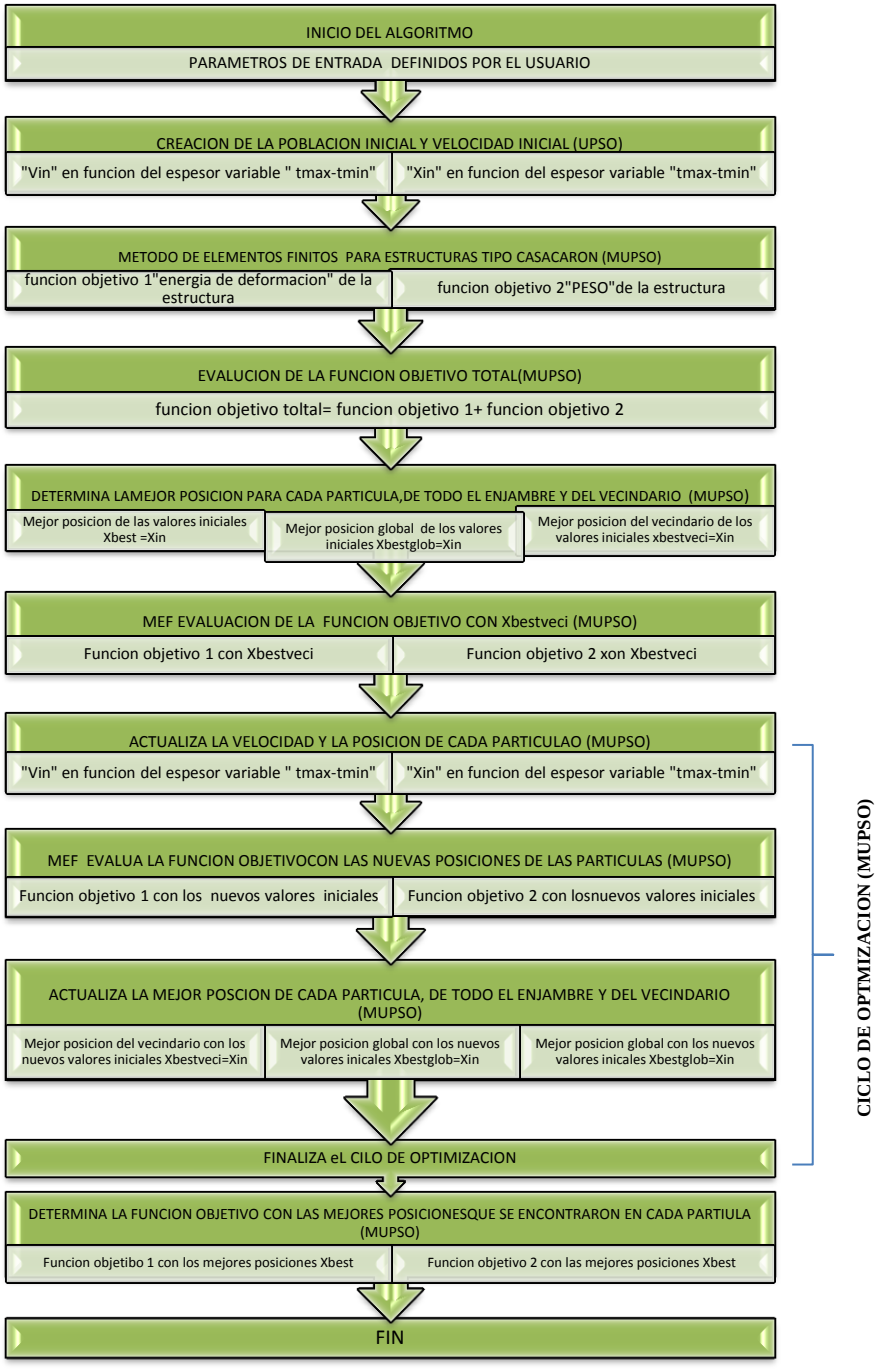
MORENO DE LUCA, Leonardo. Computación evolutiva aplicada al diseño de estructuras reticulares. Universidad Industrial de Santander, Escuela de Ingeniería Civil – UIS, 2013. 229p.

RAO, S.S.The finite element method in engineering. Oxford: Pergamon Press, 1989. 643p.

VIZOTTO, I. COMPUTATIONAL GENERATION OF FREE-FORM SHELLS IN ARCHITECTURAL DESIGN AND CIVIL ENGINEERING**;** University of Campinas (UNICAMP), Faculty of Civil Engineering, Architecture and Urban Planning. Elsevier Automation in Construction. [Online] September 2010. [Cited: 15 Feb. 2014] Available from Internet: http://www.sciencedirect.com/science/article/pii/S0926580510001330

ANEXOS

ANEXO A: DIAGRAMA 1. OPTIMIZACIÓN DE ESTRUCTURAS TIPO CASCARON



ANEXO B: CÓDIGO DE PROGRAMACION MATLAB 7.0 “MÉTODO DE ELEMENTOS FINITOS PARA PLACA TIPO SHELL"

```
%-------------------------------------------------------------------------------------------------
%NODOS DE LA CUBIERTA:
%-------------------------------------------------------------------------------------------------
%Los nodos creados en Grasshopper como una lista de números se emplean para crear un archivo .txt con las coordenadas de los nodos.
Nodos=load ('CoordenadasNodos.txt');
%-------------------------------------------------------------------------------------------------
%ORGANIZACIÓN Y ENUMERACIÓN DE NODOS POR COORDENADAS X, Y, Z
%-------------------------------------------------------------------------------------------------
%Inicia la matriz que almacenará las coordenadas de los nodos de la cubierta, con los datos de la lista del archivo  “CoordenadasNodos.txt”. El for  empieza desde 1 hasta la longitud total de la lista de nodos dividida en tres ya que se va a organizar por cada fila tres coordenadas "X, Y, Z", el 3i-2, 3i-1, 3i, hacen referencia a los índices correspondientes de cada coordenada.
CoordenadaNodos= zeros ((length (Nodos) /3), 4);

for i= 1: (length (Nodos) /3)
    CoordenadaNodos (i, 1) = 0; %Enumera los nodos de la cubierta.
    CoordenadaNodos (i, 2)=Nodos ((3*i) - 2); %Llena el vector de las coordenadas X de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-2 es porque debe tomar el número 1, 4, 7,...,  de la lista, es decir  el correspondiente a la coordenada X.
    CoordenadaNodos (i, 3)=Nodos ((3*i) - 1); %Llena el vector de las coordenadas Y de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-1 es porque debe tomar el número 2, 5, 8,... de la lista, es decir, el correspondiente a la coordenada Y.
    CoordenadaNodos (i, 4)=Nodos (3*i); %Llena el vector de las coordenadas Z de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i es porque debe tomar el número 3, 6, 9,... de la lista, es decir, el correspondiente a la coordenada Z.
end
```

```
%-----------------------------------------------------------------------------------------------------
%SE CONVIERTE EN CEROS LOS DATOS DE LAS FILAS 4, 8, 12, 16….., (MÚLTIPLOS DE 4) Y POSTERIORMENTE SE ELIMINAN DE LA LISTA.
%-----------------------------------------------------------------------------------------------------
CoordenadasCeros=CoordenadaNodos; %matriz donde se convertirá el ceros todos las filas de vectores múltiplos de 4 de la lista (4*i, :)=0, esto debido a que las tres primeras filas de coordenadas de la lista 1, 2, 3, forman el triángulo siendo la cuarta fila un nodo repetido de estos tres, así sucesivamente para las filas 5, 6, 7, donde la numero 8 se eliminara.

pos=zeros ((length (CoordenadaNodos (: 1)) /4), 1);

for i=1:(length (CoordenadaNodos (:,1))/4) %For desde 1 hasta la longitud total de la lista de coordenadas dividida en 4 para abarcar los múltiplos de la lista completa.
    CoordenadasCeros (4*i, :)=0; %Se asigna el valor de cero a toda esta posición.
    pos (i)=4*i;
end
CoordenadasFinales=CoordenadasCeros;
CoordenadasFinales (pos, :)= []; %Se eliminan todas las filas que contienen ceros, donde la matriz CoordenadasFinales contendrá los nodos, siendo cada tres filas los nodos correspondientes de un elemento que conforma la superficie.

%-----------------------------------------------------------------------------------------------------
%CAMBIO DE ÍNDICE A NODOS REPETIDOS EN LA LISTA
%-----------------------------------------------------------------------------------------------------

for i=1: (length (CoordenadasFinales (: 1))) % For desde 1 hasta la longitud total de la lista de Nodos dividida en tres.

    a=CoordenadasFinales (1: i, 1); %Vector que contiene las enumeraciones ya asignadas cada que se cumple el ciclo.
    b=unique(a); %vector que contiene las enumeraciones ya asignadas sin repetir enumeración.
```

```
c=length(a)-length (b); % largo del vector "a" menos el vector "b", esto con el fin de saber el número de índices repetidos ya asignados.

if  CoordenadasFinales (i, 1) ==0
        CoordenadasFinales (i, 1)=i-c; %si las coordenadas ubicadas en la fila i del proceso de enumeración aun no tienen índice asignado ósea tienen cero "0", el índice que se le asigna es el valor "i" del ciclo for en que va el proceso menos "c"
end

for j= (i+1) :(length (CoordenadasFinales (:,1))) %El (-1) se debe a que en el condicional se utiliza (i+1), luego el ciclo no puede ir hasta el último (i), sino hasta el penúltimo.
        ifCoordenadasFinales(j,2)==CoordenadasFinales(i,2))&&(CoordenadasFinales(j,3)==CoordenadasFinales(i,3)&&(CoordenadasFinales(j,4)==CoordenadasFinales(i,4)));  %Se evalúa si las coordenadas X (CoordenadaNodos(i,2)),Y(CoordenadaNodos(i,3)),ZCoordenadaNodos(i,4)) del Vector (i) son iguales a las coordenadas X (CoordenadaNodos(j+1,2)), Y (CoordenadaNodos(j+1,3)), Z (CoordenadaNodos(j+4,4)) del Vector (j+1).De esta manera se compara vector por vector con el resto de vectores de la lista.
        CoordenadasFinales (j, 1)=CoordenadasFinales (i, 1); % Al cumplirse la condición se cambia el índice del vector repetido por el índice designado al primer vector exactamente igual en la lista.
        end
end
end

%-----------------------------------------------------------------------------------------------------
%ORGANIZACION DE TABLA POR NUMERO DE ELEMENTOS "TRIANGULARES" QUE CONFORMAN LA SUPERFICIE TOTAL
%-----------------------------------------------------------------------------------------------------

TablaElementos=zeros ((length (CoordenadasFinales)/3) ,7); %Matriz que contendrá los elementos que conforman la superficie con sus tres nodos y los respetivos cosenos directores.
```

```
for j=1:((length (CoordenadasFinales (:,1)))/3)
    TablaElementos (j, 1)=j; %Enumera los elementos de la superficie.
    TablaElementos (j, 2)=CoordenadasFinales ((3*j-2) ,1); %Llena el vector de los Nodos (i) de los elementos  de la cubierta, con el índice del Nodo que conforma el elemento de los datos de la primer columna de la matriz "CoordenadasFinales". El 3i-2 es porque debe tomar el número 1, 4, 7,... de la lista, es decir, el correspondiente al Nodo (i) del elemento.
    TablaElementos (j, 3)=CoordenadasFinales ((3*j-1) ,1); %Llena el vector de los Nodos (j) de los elementos  de la cubierta, con el índice del Nodo que conforma el elemento de los datos de la primer columna de la matriz "CoordenadasFinales". El 3i-1 es porque debe tomar el número 2, 5, 8,... de la lista, es decir, el correspondiente al nodo (j) del elemento.
    TablaElementos (j, 4)=CoordenadasFinales ((3*j) ,1); %Llena el vector de los Nodos (k) de los elementos  de la cubierta, con el índice del Nodo que conforma el elemento de los datos de la primer columna de la matriz "CoordenadasFinales". El 3i es porque debe tomar el número 3, 6, 9,... de la lista, es decir, el correspondiente al nodo (k) del elemento.
End

%-----------------------------------------------------------------------------------------------
%Cosenos Directores Y Coordenadas Locales Por Elemento
%-----------------------------------------------------------------------------------------------

CoordpuntoP=zeros (length (TablaElementos (:,1)) ,4); %matriz que contendrá las coordenadas de un punto "p" contenido por el elemento y dentro del plano del mismo, este será perpendicular al eje y local asignado a cada elemento

CoordenadasLocales (:,1)=CoordenadasFinales (:,1); %enumeración local igual a la global.
CoordenadasLocales (:,2:3)=0;
CoordenadasLocales (:,4)=CoordenadasFinales (:,4); %coordenadas de la elevación en el eje Z.

Distancias (:,1)=TablaElementos (:,1);
```

for i=1: length (TablaElementos (:,1))

[P1]=find (CoordenadasFinales (:,1) ==TablaElementos (i, 2)); %Busco el índice de las Coordenadas X, Y, Z que necesito del nodo (i) en la toda la columna 1 de la matriz Coordenadas finales.

[P2]=find (CoordenadasFinales (:,1) ==TablaElementos (i, 3)); %Busco el índice de las Coordenadas X, Y, Z que necesito del Nodo (j) en la toda la columna 1 de la matriz Coordenadas finales.

[P3]=find (CoordenadasFinales (:,1) ==TablaElementos (i, 4)); %Busco el índice de las coordenadas X, Y, Z que necesito del nodo (k) en la toda la columna 1 de la matriz coordenadas finales.

U1=P1 (1,1); %Debido a que con el comando "find" obtengo un vector con todas la posiciones de las filas donde se encuentra el índice buscado U1 me arroja la primer fila donde se encuentran las coordenadas del nodo (i).

U2=P2 (1,1); %Debido a que con el comando "find" obtengo un vector con todas la posiciones de las filas donde se encuentra el índice buscado U2 me arroja la primer fila donde se encuentran las coordenadas del nodo (j).

U3=P3 (1,1); %Debido a que con el comando "find" obtengo un vector con todas la posiciones de las filas donde se encuentra el índice buscado U3 me arroja la primer fila donde se encuentran las coordenadas del Nodo (k).

%COSENOS DIRECTORES "EJE Y"

TablaElementos (i, 8)= (CoordenadasFinales (U2, 2) –
CoordenadasFinales(U1,2))/(((CoordenadasFinales(U2,2)-
CoordenadasFinales(U1,2))^2+(CoordenadasFinales(U2,3)–
CoordenadasFinales (U1, 3)) ^2+ (CoordenadasFinales (U2, 4) –
CoordenadasFinales (U1, 4)) ^2) ^ (1/2)); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (l).

TablaElementos(i,9)=(CoordenadasFinales(U2,3)-
CoordenadasFinales(U1,3))/(((CoordenadasFinales(U2,2)-
CoordenadasFinales(U1,2))^2+(CoordenadasFinales(U2,3)-
CoordenadasFinales(U1,3))^2+(CoordenadasFinales(U2,4)-

```
CoordenadasFinales(U1,4))^2)^(1/2)); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (m).
TablaElementos(i,10)=(CoordenadasFinales(U2,4)-
CoordenadasFinales(U1,4))/(((CoordenadasFinales(U2,2)-
CoordenadasFinales(U1,2))^2+(CoordenadasFinales(U2,3)-
CoordenadasFinales(U1,3))^2+(CoordenadasFinales(U2,4)-
CoordenadasFinales(U1,4))^2)^(1/2)); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el Coseno Director (n).

dip=TablaElementos(i,8)*(CoordenadasFinales(U3,2)-
CoordenadasFinales(U1,2))+TablaElementos(i,9)*(CoordenadasFinales(U3,3)-
CoordenadasFinales(U1,3))+TablaElementos(i,10)*(CoordenadasFinales(U3,4)-
CoordenadasFinales(U1,4));% Distancia del nodo "i" o nodo 1 del elemento al punto "P"
```

% ALMACENA LAS COORDENADAS X Y Z DEL PUNTO "P" DE CADA ELEMENTO

```
CoordpuntoP (i, 1)=i;
CoordpuntoP (i, 2)=CoordenadasFinales (U1, 2)+TablaElementos (i, 8)*dip; %Coordenada X del nodo i mas el coseno director lij del eje y por la distancia dip.
CoordpuntoP (i, 3)=CoordenadasFinales (U1, 3)+TablaElementos (i, 9)*dip; %Coordenada Y del nodo i mas el coseno director mij del eje y por la distancia dip.
CoordpuntoP (i, 4)=CoordenadasFinales (U1, 4)+TablaElementos (i, 10)*dip; %Coordenada Z del nodo i mas el coseno director nij del eje y por la distancia dip.

dik=((CoordenadasFinales(U3,2)-
CoordenadasFinales(U1,2))^2+(CoordenadasFinales(U3,3)-
CoordenadasFinales(U1,3))^2+(CoordenadasFinales(U3,4)-
CoordenadasFinales(U1,4))^2)^(1/2);% Distancia del nodo i al k, del "1 al 2"
dpk= ((dik) ^2-(dip) ^2) ^ (1/2); %Distancia del nodo k al punto p.
```

%COSENOS DIRECTORES "EJE X"

TablaElementos (i, 5)= (CoordenadasFinales (U3, 2)-CoordpuntoP (i, 2))/dpk; %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (l).

TablaElementos (i, 6)= (CoordenadasFinales (U3, 3)-CoordpuntoP (i, 3))/dpk; %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (m)

TablaElementos (i, 7)= (CoordenadasFinales (U3, 4)-CoordpuntoP (i, 4))/dpk; %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (n).

a= [TablaElementos (i, 5), TablaElementos (i, 6), TablaElementos (i, 7)]; %vector con los cosenos directores del eje x.

b=[TablaElementos(i,8),TablaElementos(i,9),TablaElementos(i,10)];%vector con los cosenos directores del eje x.

vectZ=cross(a, b); %producto cruz entre los vectores a, b con el fin de obtener los cosenos directores del eje z.

magnitud=norm (vectZ);

vectorunitarioZ=vectZ/magnitud;

%COSENOS DIRECTORES "EJE Z"

TablaElementos (i,11)=vectorunitarioZ (1); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (l).

TablaElementos(i,12)=vectorunitarioZ (2); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (m).

TablaElementos(i,13)=vectorunitarioZ (3); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (n).

Distancias (i, 2)=dip

Distancias (i, 3)=dik

Distancias (i, 4)=dpk

end

```
for j=1:((length (CoordenadasFinales (:,1)))/3)
        CoordenadasLocales((3*j-1),3)=((CoordenadasFinales((3*j-1),2)-CoordenadasFinales((3*j-
        2),2))^2+(CoordenadasFinales((3*j-1),3)-CoordenadasFinales((3*j-
        2),3))^2+(CoordenadasFinales((3*j-1),4)-CoordenadasFinales((3*j-2),4))^2)^(1/2);
        CoordenadasLocales (3*j, 2)=Distancias (j, 4);
        CoordenadasLocales (3*j, 3)=Distancias (j, 2);
 end

%-------------------------------------------------------------------------------------------------
GENERA LA MATRIZ DE RIGIDEZ DEL ELEMENTO
%-------------------------------------------------------------------------------------------------

MATRIZRIGIDEZ=zeros (max (CoordenadasLocales (:,1))*6, max (CoordenadasLocales (:,1))*6);

for i=1: length (TablaElementos (:,1))
      A= [CoordenadasLocales (3*i-2,2), CoordenadasLocales (3*i-2,3), CoordenadasLocales (3*i-
      2,4)];
      B=[CoordenadasLocales(3*i-1,2),CoordenadasLocales(3*i-1,3),CoordenadasLocales(3*i-
      1,4)];
      C=[CoordenadasLocales(3*i,2),CoordenadasLocales(3*i,3),CoordenadasLocales(3*i,4)];
      AB=B-A;
      AC=C-A;
      r=cross (AB, AC);
      AR=norm(r)/2;
      TablaElementos (i, 14)=AR; % Asigno el área para cada elemento de la superficie.
      CoordenadasLocales ((3*i-2):3*i, 5)=TablaElementos (i, 14)/3;
```

```
%-------------------------------------------------------------------------------------------------
%DATOS DE ENTRADA (PROPIEDADES DEL MATERIAL):
%-------------------------------------------------------------------------------------------------

E=2.15381e10; % Modulo de elasticidad (N/m2)
t=0.05; % Espesor placa (m)
v=0.2; % Coeficiente de Poisson
p=23544; % Peso específico del concreto (N/m3)
a= (E*t)/ (4*AR*(1-v^2));
b= (E*t)/ (8*AR*(1+v));
c= (E*(t^3))/ (12*(1-v^2));

% MATRIZ DE RIGIDEZ PLACA

MATRIZPLACA=zeros (9,9);

%FILA 4
MATRIZPLACA (4,4)=2*c*C (1)*B (2);
MATRIZPLACA (4,6)=2*c*v*C (1)*B (2);
MATRIZPLACA (4,7)=2*c*(C (1)) ^2*B (2);
MATRIZPLACA(4,8)=(2/3)*c*v*(C(1))^2*B(2)+(2/3)*c*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA (4,9)=2*c*v*C (1)*B (2)*(B (2)+C (2));
%FILA 5
MATRIZPLACA (5,5)= (1-v)*c*C (1)*B (2);
MATRIZPLACA(5,8)=(2/3)*(1-v)*c*(C(1))^2*B(2)+(2/3)*(1-
v)*c*C(1)*B(2)*(B(2)+C(2));
%FILA 6
MATRIZPLACA (6,4)=2*c*v*C (1)*B (2);
MATRIZPLACA (6,6)=2*c*C (1)*B (2);
MATRIZPLACA (6,7)=2*c*v*(C (1)) ^2*B (2);
MATRIZPLACA(6,8)=(2/3)*c*(C(1))^2*B(2)+(2/3)*c*v*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA (6,9)=2*c*C (1)*B (2)*(B (2)+C (2));
```

%FILA 7

```
MATRIZPLACA (7,4)=2*c*(C (1)) ^2*B (2);
MATRIZPLACA (7,6)=2*c*v*(C (1)) ^2*B (2);
MATRIZPLACA (7,7)=3*c*(C (1)) ^3*B (2);
MATRIZPLACA(7,8)=c*v*(C(1))^3*B(2)+(1/2)*c*(C(1))^2*B(2)*(B(2)+(2*C(2)));
MATRIZPLACA (7,9)= (3/2)*c*v*(C (1)) ^2*B (2)*(B (2)+2*C (2));
```

%FILA 8

```
MATRIZPLACA(8,4)=(2/3)*c*v*(C(1))^2*B(2)+(2/3)*c*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA(8,5)=(2/3)*(1-v)*c*(C(1))^2*B(2)+(2/3)*(1-
v)*c*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA(8,6)=(2/3)*c*(C(1))^2*B(2)+(2/3)*c*v*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA(8,7)=c*v*(C(1))^3*B(2)+(1/2)*c*(C(1))^2*B(2)*(B(2)+(2*C(2)));
MATRIZPLACA(8,8)=c*((128*v)*((((C(1))^3*B(2))/12)+(1/12)*(C(1))^2*B(2)*(B(2)+2*
(C(2)))+(1/12)*C(1)*B(2)*((B(2))^2+B(2)*C(2)+(C(2))^2))-(8*(1-
v))*((1/24)*(C(1))^2*B(2)*(B(2)+2*(C(2)))));
MATRIZPLACA(8,9)=(c/2)*(C(1))^2*B(2)*(B(2)+2*C(2))+c*v*C(1)*B(2)*((B(2))^2+B(
2)*C(2)+(C(2))^2);
```

%FILA 9

```
MATRIZPLACA (9,4)=2*c*v*C (1)*B (2)*(B (2)+C (2));
MATRIZPLACA (9,6)=2*c*C (1)*B (2)*(B (2)+C (2));
MATRIZPLACA (9,7)= (3/2)*c*v*(C (1)) ^2*B (2)*(B (2)+2*C (2));
MATRIZPLACA(9,8)=(c/2)*(C(1))^2*B(2)*(B(2)+2*C(2))+c*v*C(1)*B(2)*((B(2))^2+B(
2)*C(2)+(C(2))^2);
MATRIZPLACA (9,9)=3*c*C (1)*B (2)*((B (2)) ^2+B (2)*C (2)+C (2) ^2);
```

```
MATRIZN=zeros (9,9);
```

%FILA1

```
MATRIZN (1,1)=1;
```

%FILA2

```
MATRIZN (2,3)=1;
```

%FILA3

```
MATRIZN (3,2)=-1;
```

%FILA4

MATRIZN (4,1)=1;

MATRIZN (4,3)=B (2);

MATRIZN (4,6)= (B (2)) ^2;

MATRIZN (4,9)= (B (2)) ^3;

%FILA5

MATRIZN (5,3)=1;

MATRIZN (5,6)=2*B (2);

MATRIZN (5,9)=3*(B (2)) ^2;

%FILA6

MATRIZN (6,2)=-1;

MATRIZN (6,5)=-B (2);

MATRIZN (6,8)=-(B (2)) ^2;

%FILA7

MATRIZN (7,1)=1;

MATRIZN (7,2)=C (1);

MATRIZN (7,3)=C (2);

MATRIZN (7,4)=(C (1)) ^2;

MATRIZN (7,5)=C (1)*C (2);

MATRIZN (7,6)=(C (2)) ^2;

MATRIZN (7,7)=(C (1)) ^3;

MATRIZN (7,8)=(C (1)) ^2*C (2)+C (1)*(C (2)) ^2;

MATRIZN (7,9)=(C (2)) ^3;

%FILA8

MATRIZN (8,3)=1;

MATRIZN (8,5)=C (1);

MATRIZN (8,6)=2*C (2);

MATRIZN (8,8)=2*C (1)*C (2)+(C (1)) ^2;

MATRIZN (8,9)=3*(C (2)) ^2;

%FILA9

MATRIZN (9,2)=-1;

MATRIZN (9,4)=-2*C (1);

MATRIZN (9,5)=-C (2);

MATRIZN (9,7)=-3*(C (1)) ^2;

```
MATRIZN (9,8)=-((C (2)) ^2+2*C (1)*C (2));

MATRIZNINV=inv (MATRIZN);
MATRIZNTRANSP=transp (MATRIZNINV);
KPLACA=MATRIZNTRANSP*MATRIZPLACA*MATRIZNINV;

% MATRIZ DE RIGIDEZ MEMBRANA
KMEMBRANA=zeros (6,6);
%FILA 1
KMEMBRANA (1,1)=a*(B (2)-C (2)) ^2+b*(B (1)-C (1)) ^2;
KMEMBRANA (1,2)=-a*v*(B (2)-C (2))*(B (1)-C (1))-b*(B (1)-C (1))*(B (2)-C (2));
KMEMBRANA (1,3)=-a*(B (2)-C (2))*(A (2)-C (2))-b*(B (1)-C (1))*(A (1)-C (1));
KMEMBRANA (1,4)=a*v*(B (2)-C (2))*(A (1)-C (1))+b*(B (1)-C (1))*(A (2)-C (2));
KMEMBRANA (1,5)=a*(B (2)-C (2))*(A (2)-B (2))+b*(B (1)-C (1))*(A (1)-B (1));
KMEMBRANA (1,6)=-a*v*(B (2)-C (2))*(A (1)-B (1))-b*(B (1)-C (1))*(A (2)-B (2));
%FILA 2
KMEMBRANA (2,1)=-a*v*(B (2)-C (2))*(B (1)-C (1))-b*(B (1)-C (1))*(B (2)-C (2));
KMEMBRANA (2,2)=b*(B (2)-C (2)) ^2+a*(B (1)-C (1)) ^2;
KMEMBRANA (2,3)=a*v*(B (1)-C (1))*(A (2)-C (2))+b*(B (2)-C (2))*(A (1)-C (1));
KMEMBRANA (2,4)=-a*(B (1)-C (1))*(A (1)-C (1))-b*(B (2)-C (2))*(A (2)-C (2));
KMEMBRANA (2,5)=-a*v*(B (1)-C (1))*(A (2)-B (2))-b*(B (2)-C (2))*(A (1)-B (1));
KMEMBRANA (2,6)=a*(B (1)-C (1))*(A (1)-B (1))+b*(B (2)-C (2))*(A (2)-B (2));
%FILA 3
KMEMBRANA (3,1)=-a*(B (2)-C (2))*(A (2)-C (2))-b*(B (1)-C (1))*(A (1)-C (1));
KMEMBRANA (3,2)=a*v*(B (1)-C (1))*(A (2)-C (2))+b*(B (2)-C (2))*(A (1)-C (1));
KMEMBRANA (3,3)=a*(A (2)-C (2)) ^2+b*(A (1)-C (1)) ^2;
KMEMBRANA (3,4)=-a*v*(A (2)-C (2))*(A (1)-C (1))-b*(A (1)-C (1))*(A (2)-C (2));
KMEMBRANA (3,5)=-a*(A (2)-C (2))*(A (2)-B (2))-b*(A (1)-C (1))*(A (1)-B (1));
KMEMBRANA (3,6)=a*v*(A (2)-C (2))*(A (1)-B (1))+b*(A (1)-C (1))*(A (2)-B (2));
%FILA 4
KMEMBRANA (4,1)=a*v*(B (2)-C (2))*(A (1)-C (1))+b*(B (1)-C (1))*(A (2)-C (2));
KMEMBRANA (4,2)=-a*(B (1)-C (1))*(A (1)-C (1))-b*(B (2)-C (2))*(A (2)-C (2));
KMEMBRANA (4,3)=-a*v*(A (2)-C (2))*(A (1)-C (1))-b*(A (1)-C (1))*(A (2)-C (2));
```

```
KMEMBRANA (4,4)=b*(A (2)-C (2)) ^2+a*(A (1)-C (1)) ^2;
KMEMBRANA (4,5)=a*v*(A (1)-C (1))*(A (2)-B (2))+b*(A (2)-C (2))*(A (1)-B (1));
KMEMBRANA (4,6)=-a*(A (1)-C (1))*(A (1)-B (1))-b*(A (2)-C (2))*(A (2)-B (2));
%FILA 5
KMEMBRANA (5,1)=a*(B (2)-C (2))*(A (2)-B (2))+b*(B (1)-C (1))*(A (1)-B (1));
KMEMBRANA (5,2)=-a*v*(B (1)-C (1))*(A (2)-B (2))-b*(B (2)-C (2))*(A (1)-B (1));
KMEMBRANA (5,3)=-a*(A (2)-C (2))*(A (2)-B (2))-b*(A (1)-C (1))*(A (1)-B (1));
KMEMBRANA (5,4)=a*v*(A (1)-C (1))*(A (2)-B (2))+b*(A (2)-C (2))*(A (1)-B (1));
KMEMBRANA (5,5)=a*(A (2)-B (2)) ^2+b*(A (1)-B (1)) ^2;
KMEMBRANA (5,6)=-a*v*(A (2)-B (2))*(A (1)-B (1))-b*(A (1)-B (1))*(A (2)-B (2));
%FILA 6
KMEMBRANA (6,1)=-a*v*(B (2)-C (2))*(A (1)-B (1))-b*(B (1)-C (1))*(A (2)-B (2));
KMEMBRANA (6,2)=a*(B (1)-C (1))*(A (1)-B (1))+b*(B (2)-C (2))*(A (2)-B (2));
KMEMBRANA (6,3)=a*v*(A (2)-C (2))*(A (1)-B (1))+b*(A (1)-C (1))*(A (2)-B (2));
KMEMBRANA (6,4)=-a*(A (1)-C (1))*(A (1)-B (1))-b*(A (2)-C (2))*(A (2)-B (2));
KMEMBRANA (6,5)=-a*v*(A (2)-B (2))*(A (1)-B (1))-b*(A (1)-B (1))*(A (2)-B (2));
KMEMBRANA (6,6)=b*(A (2)-B (2)) ^2+a*(A (1)-B (1)) ^2;

% MATRIZ RIGIDEZ TOTAL ELEMENTO
MATRIZK=zeros (18,18);

%FILA 1-2
MATRIZK (1:2,1:2)=KMEMBRANA (1:2,1:2);
MATRIZK (1:2,7:8)=KMEMBRANA (1:2,3:4);
MATRIZK (1:2,13:14)=KMEMBRANA (1:2,5:6);
%FILA 3-5
MATRIZK (3:5,3:5)=KPLACA (1:3,1:3);
MATRIZK (3:5,9:11)=KPLACA (1:3,4:6);
MATRIZK (3:5,15:17)=KPLACA (1:3,7:9);
%FILA 7-8
MATRIZK (7:8,1:2)=KMEMBRANA (3:4,1:2);
MATRIZK (7:8,7:8)=KMEMBRANA (3:4,3:4);
MATRIZK (7:8,13:14)=KMEMBRANA (3:4,5:6);
```

```
%FILA 9-11
MATRIZK (9:11,3:5)=KPLACA (4:6,1:3);
MATRIZK (9:11,9:11)=KPLACA (4:6,4:6);
MATRIZK (9:11,15:17)=KPLACA (4:6,7:9);
%FILA 13-14
MATRIZK (13:14,1:2)=KMEMBRANA (5:6,1:2);
MATRIZK (13:14,7:8)=KMEMBRANA (5:6,3:4);
MATRIZK (13:14,13:14)=KMEMBRANA (5:6,5:6);
%FILA 15-17
MATRIZK (15:17,3:5)=KPLACA (7:9,1:3);
MATRIZK (15:17,9:11)=KPLACA (7:9,4:6);
MATRIZK (15:17,15:17)=KPLACA (7:9,7:9);

% MATRIZ DE TRANSFORMACION
MATRIZTRANSF=zeros (18,18);
MATRIZTRANSF (1,1:3)=TablaElementos (i, 5:7);
MATRIZTRANSF (2,1:3)=TablaElementos (i, 8:10);
MATRIZTRANSF (3,1:3)=TablaElementos (i, 11:13);
MATRIZTRANSF (4:6,4:6)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (7:9,7:9)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (10:12,10:12)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (13:15,13:15)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (16:18,16:18)=MATRIZTRANSF (1:3,1:3);

TRANSPUESTA=MATRIZTRANSF';
MATRIZELEMENTOi=TRANSPUESTA*MATRIZK*MATRIZTRANSF;
R=MATRIZK;
v=genvarname (['K', num2str (i)]);
Eval ([v,'=R']);
```

```
%-------------------------------------------------------------------------------------------------
%ENSAMBLE MATRIZ DE RIGIDEZ TOTAL
%-------------------------------------------------------------------------------------------------

    %FILA 1-6
    for m=5:-1:0
        n=6-m;

         MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        5)+MATRIZELEMENTOi(n,1);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        4)+MATRIZELEMENTOi(n,2);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        3)+MATRIZELEMENTOi(n,3);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        2)+MATRIZELEMENTOi(n,4);
         MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        1)+MATRIZELEMENTOi(n,5);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-
        m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
        m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        5)+MATRIZELEMENTOi(n,7);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        4)+MATRIZELEMENTOi(n,8);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
5)+MATRIZELEMENTOi(n,13);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
4)+MATRIZELEMENTOi(n,14);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
3)+MATRIZELEMENTOi(n,15);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
2)+MATRIZELEMENTOi(n,16);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
1)+MATRIZELEMENTOi(n,17);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
end
```

```
%FILA 7-12
for m=5:-1:0
    n=12-m;
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
5)+MATRIZELEMENTOi(n,1);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
4)+MATRIZELEMENTOi(n,2);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
3)+MATRIZELEMENTOi(n,3);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
2)+MATRIZELEMENTOi(n,4);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
1)+MATRIZELEMENTOi(n,5);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
5)+MATRIZELEMENTOi(n,7);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
4)+MATRIZELEMENTOi(n,8);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
```

```
        MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
        1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
        1)+MATRIZELEMENTOi(n,11);
        MATRIZRIGIDEZ(6*TablaElementos(i,3)-
        m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
        m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
        MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        5)+MATRIZELEMENTOi(n,13);
        MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        4)+MATRIZELEMENTOi(n,14);
        MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        3)+MATRIZELEMENTOi(n,15);
        MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        2)+MATRIZELEMENTOi(n,16);
        MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        1)+MATRIZELEMENTOi(n,17);
        MATRIZRIGIDEZ(6*TablaElementos(i,3)-
        m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
        m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
end

%FILA 13-18
for m=5:-1:0
        n=18-m;
        MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        5)+MATRIZELEMENTOi(n,1);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
4)+MATRIZELEMENTOi(n,2);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
3)+MATRIZELEMENTOi(n,3);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
2)+MATRIZELEMENTOi(n,4);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
1)+MATRIZELEMENTOi(n,5);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
5)+MATRIZELEMENTOi(n,7);
 MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
4)+MATRIZELEMENTOi(n,8);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
```

```
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                5)+MATRIZELEMENTOi(n,13);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                4)+MATRIZELEMENTOi(n,14);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                3)+MATRIZELEMENTOi(n,15);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                2)+MATRIZELEMENTOi(n,16);
                 MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                1)+MATRIZELEMENTOi(n,17);
                 MATRIZRIGIDEZ(6*TablaElementos(i,4)-
                m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
                m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
        end
end

VectorFuerzas=zeros (max (CoordenadasLocales (:,1))*6,1);

for j=1: max (CoordenadasLocales (:,1))
        z=find (CoordenadasLocales (:,1) ==j);
                VectorFuerzas (6*j-3)=sum (CoordenadasLocales (z, 5))*t*p*-1; % Vector que
        contiene las fuerzas por peso propio por área aferente soportadas por cada nodo en
        direccion –Z.
end

PESO=abs (sum (VectorFuerzas)) %Peso total cubierta
```

```
%-------------------------------------------------------------------------------------------------
%NODOS DE LOS APOYOS
%-------------------------------------------------------------------------------------------------

Apoyos=load ('NodosApoyos.txt'); %Organización y enumeración de nodos por coordenadas X, Y, Z.

CoordenadaApoyos=zeros ((length (Apoyos)/3) ,4); %Inicia la matriz que almacenará las coordenadas de los nodos de la cubierta.

for i=1:(length (Apoyos)/3) % For desde 1 hasta la longitud total de la lista de nodos dividida en tres ya que se va a organizar por cada fila tres coordenadas "X, Y, Z".
    CoordenadaApoyos (i, 1)=0; %Enumera los nodos de la cubierta.
    CoordenadaApoyos (i, 2)=Apoyos ((3*i)-2); %Llena el vector de las coordenadas X de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-2 es porque debe tomar el número 1, 4, 7,... de la lista, es decir, el correspondiente a la coordenada X.
    CoordenadaApoyos (i, 3)=Apoyos ((3*i)-1); %Llena el vector de las coordenadas Y de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-1 es porque debe tomar el número 2, 5, 8,... de la lista, es decir, el correspondiente a la coordenada Y.
    CoordenadaApoyos (i, 4)=Apoyos (3*i); %Llena el vector de las coordenadas Z de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i es porque debe tomar el número 3, 6, 9,... de la lista, es decir, el correspondiente a la coordenada Z.

    ind=find(CoordenadasFinales(:,2)==CoordenadaApoyos(i,2)&CoordenadasFinales(:,3)==C
    oordenadaApoyos(i,3)&CoordenadasFinales(:,4)==CoordenadaApoyos(i,4));
    CoordenadaApoyos (i, 1)=CoordenadasFinales (ind (1) ,1);
end
```

```
%ELIMINACION ROTACION EN Z

Nnodos=length (MATRIZRIGIDEZ (1, :))/6

for i=Nnodos:-1:1
   MATRIZRIGIDEZ (6*i, :)= [];
   MATRIZRIGIDEZ (:,6*i)= [];
   VectorFuerzas (6*i, :)= [];
end

NodosAp=sort ((CoordenadaApoyos (:,1)),'descend'); % Se organizan los nodos de los apoyos en orden descendente.
k=str2double (inputdlg ('EMPOTRADO: Digite 1; ARTICULADO: Digite 2; SIMPLEMENTE APOYADO: Digite 3','RESTRICCIONES EN LOS APOYOS'));

%EMPOTRADO
if k==1
   for i=1:(length (Apoyos)/3)
      MATRIZRIGIDEZ (5*(NodosAp (i))-4:5*(NodosAp (i)), :)= [];
      MATRIZRIGIDEZ (:,5*(NodosAp (i))-4:5*(NodosAp (i)))= [];
      VectorFuerzas (5*(NodosAp (i))-4:5*(NodosAp (i)), :)= [];
   End

%MEMBRANA
   VectorFuerzas (1)=1000;
   VectorFuerzas (3)=-2000;
   DESP=linsolve (MATRIZRIGIDEZ, VectorFuerzas);
   e1=unique (CoordenadasFinales (:,1));
   e1 (NodosAp)= [];
   for i=1: length (e1)
      DESP (5*i-4:5*i, 2)=e1 (i);
   end
end
```

```
%ARTICULADO
if k==2
   for i=1:(length (Apoyos)/3)
      MATRIZRIGIDEZ (5*(NodosAp (i))-4:5*(NodosAp (i))-2, :)= [];
      MATRIZRIGIDEZ (:,5*(NodosAp (i))-4:5*(NodosAp (i))-2)= [];
      VectorFuerzas (5*(NodosAp (i))-4:5*(NodosAp (i))-2, :)= [];
   end
   DESP=linsolve (MATRIZRIGIDEZ, VectorFuerzas);
end
%SIMPLEMENTE APOYADO
if k==3
   for i=1:(length (Apoyos)/3)
      MATRIZRIGIDEZ (5*(NodosAp (i))-2, :)= [];
      MATRIZRIGIDEZ (:,5*(NodosAp (i))-2)= [];
      VectorFuerzas (5*(NodosAp (i))-2, :)= [];
   end
   DESP=linsolve (MATRIZRIGIDEZ, VectorFuerzas);
end
%ENERGIA DE DEFORMACION
DESPLAZAMIENTOS=DESP (:,1);
ENERGIA=transp (VectorFuerzas)*DESPLAZAMIENTOS
```

ANEXO C: CÓDIGO DE PROGRAMACION MATLAB 7.0 “OPTIMIZACIÓN DE CUBIERTAS TIPO CASCARON”

```
%-------------------------------------------------------------------------------------------------
%PARÁMETROS DE ENTRADA
%-------------------------------------------------------------------------------------------------
numpar =10; %Número de Particulas (Superficies).
numit =50; %Número de iteraciones.
X=0.5; %Factor de constricción.
c1=1.1; %Parámetro cognitivo.
c2=1.75; %Parámetro social.
u=0.62; %Factor de unificación.
w1=0.5; %Porcentaje de optimización función objetivo 1.
w2=0.5; %Porcentaje de optimización función objetivo 2.
tmax=0.25; %Límite superior del rango en donde se generará aleatoriamente la población inicial.
tmin=0.05; %Límite inferior del rango en donde se generará aleatoriamente la población inicial.

%-------------------------------------------------------------------------------------------------
%NODOS DE LA CUBIERTA:
%-------------------------------------------------------------------------------------------------
%Los nodos creados en Grasshopper como una lista de números se emplean para crear un archivo
.txt con las coordenadas de los nodos.
Nodos=load ('CoordenadasNodos.txt');

%-------------------------------------------------------------------------------------------------
%ORGANIZACIÓN Y ENUMERACIÓN DE NODOS POR COORDENADAS X, Y, Z
%-------------------------------------------------------------------------------------------------
%Inicia la matriz que almacenará las coordenadas de los nodos de la cubierta, con los datos de la
lista del archivo  “CoordenadasNodos.txt”. El for  empieza desde 1 hasta la longitud total de la lista
de nodos dividida en tres ya que se va a organizar por cada fila tres coordenadas "X, Y, Z", el 3i-2,
3i-1, 3i, hacen referencia a los índices correspondientes de cada coordenada.
CoordenadaNodos= zeros ((length (Nodos) /3), 4);
```

```
for i= 1: (length (Nodos) /3)
    CoordenadaNodos (i, 1) = 0; %Enumera los nodos de la cubierta.
    CoordenadaNodos (i, 2)=Nodos ((3*i) - 2); %Llena el vector de las coordenadas X de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-2 es porque debe tomar el número 1, 4, 7,...,  de la lista, es decir  el correspondiente a la coordenada X.
    CoordenadaNodos (i, 3)=Nodos ((3*i) - 1); %Llena el vector de las coordenadas Y de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-1 es porque debe tomar el número 2, 5, 8,... de la lista, es decir, el correspondiente a la coordenada Y.
    CoordenadaNodos (i, 4)=Nodos (3*i); %Llena el vector de las coordenadas Z de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i es porque debe tomar el número 3, 6, 9,... de la lista, es decir, el correspondiente a la coordenada Z.
end

%-----------------------------------------------------------------------------------------------------
%SE CONVIERTE EN CEROS LOS DATOS DE LAS FILAS 4, 8, 12, 16….., (MÚLTIPLOS DE 4) Y POSTERIORMENTE SE ELIMINAN DE LA LISTA.
%-----------------------------------------------------------------------------------------------------

CoordenadasCeros=CoordenadaNodos; %matriz donde se convertirá el ceros todos las filas de vectores múltiplos de 4 de la lista (4*i, :)=0, esto debido a que las tres primeras filas de coordenadas de la lista 1, 2, 3, forman el triángulo siendo la cuarta fila un nodo repetido de estos tres, así sucesivamente para las filas 5, 6, 7, donde la numero 8 se eliminara.

pos=zeros ((length (CoordenadaNodos (: 1)) /4), 1);

for i=1:(length (CoordenadaNodos (:,1))/4) %For desde 1 hasta la longitud total de la lista de coordenadas dividida en 4 para abarcar los múltiplos de la lista completa.
    CoordenadasCeros (4*i, :)=0; %Se asigna el valor de cero a toda esta posición.
    pos (i)=4*i;
end
CoordenadasFinales=CoordenadasCeros;
```

CoordenadasFinales (pos, :)= []; %Se eliminan todas las filas que contienen ceros, donde la matriz CoordenadasFinales contendrá los nodos, siendo cada tres filas los nodos correspondientes de un elemento que conforma la superficie.

%----------

%CAMBIO DE ÍNDICE A NODOS REPETIDOS EN LA LISTA

%----------

for i=1: (length (CoordenadasFinales (: 1))) % For desde 1 hasta la longitud total de la lista de Nodos dividida en tres.

a=CoordenadasFinales (1: i, 1); %Vector que contiene las enumeraciones ya asignadas cada que se cumple el ciclo.

b=unique(a); %vector que contiene las enumeraciones ya asignadas sin repetir enumeración.

c=length(a)-length (b); % largo del vector "a" menos el vector "b", esto con el fin de saber el número de índices repetidos ya asignados.

if CoordenadasFinales (i, 1) ==0

CoordenadasFinales (i, 1)=i-c; %si las coordenadas ubicadas en la fila i del proceso de enumeración aun no tienen índice asignado ósea tienen cero "0", el índice que se le asigna es el valor "i" del ciclo for en que va el proceso menos "c"

end

for j= (i+1) :(length (CoordenadasFinales (:,1))) %El (-1) se debe a que en el condicional se utiliza (i+1), luego el ciclo no puede ir hasta el último (i), sino hasta el penúltimo.

ifCoordenadasFinales(j,2)==CoordenadasFinales(i,2))&&(CoordenadasFinales(j,3) ==CoordenadasFinales(i,3)&&(CoordenadasFinales(j,4)==CoordenadasFinales(i,4))); %Se evalúa si las coordenadas X (CoordenadaNodos(i,2)),Y(CoordenadaNodos(i,3)),ZCoordenadaNodos(i,4)) del Vector (i) son iguales a las coordenadas X (CoordenadaNodos(j+1,2)), Y (CoordenadaNodos(j+1,3)), Z (CoordenadaNodos(j+4,4)) del Vector (j+1).De esta manera se compara vector por vector con el resto de vectores de la lista.

```
            CoordenadasFinales (j, 1)=CoordenadasFinales (i, 1); % Al cumplirse la condición se cambia el índice del vector repetido por el índice designado al primer vector exactamente igual en la lista.
            end
        end
end

%----------------------------------------------------------------------------------------------------
%ORGANIZACION DE TABLA POR NUMERO DE ELEMENTOS "TRIANGULARES" QUE CONFORMAN LA SUPERFICIE TOTAL
%----------------------------------------------------------------------------------------------------

TablaElementos=zeros ((length (CoordenadasFinales)/3) ,7); %Matriz que contendrá los elementos que conforman la superficie con sus tres nodos y los respetivos cosenos directores.

for j=1:((length (CoordenadasFinales (:,1)))/3)
        TablaElementos (j, 1)=j; %Enumera los elementos de la superficie.
        TablaElementos (j, 2)=CoordenadasFinales ((3*j-2) ,1); %Llena el vector de los Nodos (i) de los elementos  de la cubierta, con el índice del Nodo que conforma el elemento de los datos de la primer columna de la matriz "CoordenadasFinales". El 3i-2 es porque debe tomar el número 1, 4, 7,... de la lista, es decir, el correspondiente al Nodo (i) del elemento.
        TablaElementos (j, 3)=CoordenadasFinales ((3*j-1) ,1); %Llena el vector de los Nodos (j) de los elementos  de la cubierta, con el índice del Nodo que conforma el elemento de los datos de la primer columna de la matriz "CoordenadasFinales". El 3i-1 es porque debe tomar el número 2, 5, 8,... de la lista, es decir, el correspondiente al nodo (j) del elemento.
        TablaElementos (j, 4)=CoordenadasFinales ((3*j) ,1); %Llena el vector de los Nodos (k) de los elementos  de la cubierta, con el índice del Nodo que conforma el elemento de los datos de la primer columna de la matriz "CoordenadasFinales". El 3i es porque debe tomar el número 3, 6, 9,... de la lista, es decir, el correspondiente al nodo (k) del elemento.
End
```

```
%-------------------------------------------------------------------------------------------------
%Cosenos Directores Y Coordenadas Locales Por Elemento
%-------------------------------------------------------------------------------------------------
CoordpuntoP=zeros (length (TablaElementos (:,1)) ,4); %matriz que contendrá las coordenadas de un punto "p" contenido por el elemento y dentro del plano del mismo, este será perpendicular al eje y local asignado a cada elemento

CoordenadasLocales (:,1)=CoordenadasFinales (:,1); %enumeración local igual a la global.
CoordenadasLocales (:,2:3)=0;
CoordenadasLocales (:,4)=CoordenadasFinales (:,4); %coordenadas de la elevación en el eje Z.

Distancias (:,1)=TablaElementos (:,1);

for i=1: length (TablaElementos (:,1))

    [P1]=find (CoordenadasFinales (:,1) ==TablaElementos (i, 2)); %Busco el índice de las Coordenadas X, Y, Z que necesito del nodo (i) en la toda la columna 1 de la matriz Coordenadas finales.
    [P2]=find (CoordenadasFinales (:,1) ==TablaElementos (i, 3)); %Busco el índice de las Coordenadas X, Y, Z que necesito del Nodo (j) en la toda la columna 1 de la matriz Coordenadas finales.
    [P3]=find (CoordenadasFinales (:,1) ==TablaElementos (i, 4)); %Busco el índice de las coordenadas X, Y, Z que necesito del nodo (k) en la toda la columna 1 de la matriz coordenadas finales.
    U1=P1 (1,1); %Debido a que con el comando “find” obtengo un vector con todas la posiciones de las filas donde se encuentra el índice buscado U1 me arroja la primer fila donde se encuentran las coordenadas del nodo (i).
    U2=P2 (1,1); %Debido a que con el comando “find” obtengo un vector con todas la posiciones de las filas donde se encuentra el índice buscado U2 me arroja la primer fila donde se encuentran las coordenadas del nodo (j).
    U3=P3 (1,1); %Debido a que con el comando “find” obtengo un vector con todas la posiciones de las filas donde se encuentra el índice buscado U3 me arroja la primer fila donde se encuentran las coordenadas del Nodo (k).
```

%COSENOS DIRECTORES "EJE Y"

```
TablaElementos (i, 8)= (CoordenadasFinales (U2, 2) –
CoordenadasFinales(U1,2))/(((CoordenadasFinales(U2,2)-
CoordenadasFinales(U1,2))^2+(CoordenadasFinales(U2,3)–
CoordenadasFinales (U1, 3)) ^2+ (CoordenadasFinales (U2, 4) –
CoordenadasFinales (U1, 4)) ^2) ^ (1/2)); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (l).
TablaElementos(i,9)=(CoordenadasFinales(U2,3)-
CoordenadasFinales(U1,3))/(((CoordenadasFinales(U2,2)-
CoordenadasFinales(U1,2))^2+(CoordenadasFinales(U2,3)-
CoordenadasFinales(U1,3))^2+(CoordenadasFinales(U2,4)-
CoordenadasFinales(U1,4))^2)^(1/2)); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (m).
TablaElementos(i,10)=(CoordenadasFinales(U2,4)-
CoordenadasFinales(U1,4))/(((CoordenadasFinales(U2,2)-
CoordenadasFinales(U1,2))^2+(CoordenadasFinales(U2,3)-
CoordenadasFinales(U1,3))^2+(CoordenadasFinales(U2,4)-
CoordenadasFinales(U1,4))^2)^(1/2)); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el Coseno Director (n).

dip=TablaElementos(i,8)*(CoordenadasFinales(U3,2)-
CoordenadasFinales(U1,2))+TablaElementos(i,9)*(CoordenadasFinales(U3,3)-
CoordenadasFinales(U1,3))+TablaElementos(i,10)*(CoordenadasFinales(U3,4)-
CoordenadasFinales(U1,4));% Distancia del nodo "i" o nodo 1 del elemento al punto "P"
```

% ALMACENA LAS COORDENADAS X Y Z DEL PUNTO "P" DE CADA ELEMENTO

```
CoordpuntoP (i, 1)=i;
CoordpuntoP (i, 2)=CoordenadasFinales (U1, 2)+TablaElementos (i, 8)*dip; %Coordenada X del nodo i mas el coseno director lij del eje y por la distancia dip.
CoordpuntoP (i, 3)=CoordenadasFinales (U1, 3)+TablaElementos (i, 9)*dip; %Coordenada Y del nodo i mas el coseno director mij del eje y por la distancia dip.
CoordpuntoP (i, 4)=CoordenadasFinales (U1, 4)+TablaElementos (i, 10)*dip; %Coordenada Z del nodo i mas el coseno director nij del eje y por la distancia dip.
```

```
dik=((CoordenadasFinales(U3,2)-
CoordenadasFinales(U1,2))^2+(CoordenadasFinales(U3,3)-
CoordenadasFinales(U1,3))^2+(CoordenadasFinales(U3,4)-
CoordenadasFinales(U1,4))^2)^(1/2);% Distancia del nodo i al k, del "1 al 2"
dpk= ((dik) ^2-(dip) ^2) ^ (1/2); %Distancia del nodo k al punto p.
```

%COSENOS DIRECTORES "EJE X"

```
TablaElementos (i, 5)= (CoordenadasFinales (U3, 2)-CoordpuntoP (i, 2))/dpk; %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (l).
TablaElementos (i, 6)= (CoordenadasFinales (U3, 3)-CoordpuntoP (i, 3))/dpk; %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (m)
TablaElementos (i, 7)= (CoordenadasFinales (U3, 4)-CoordpuntoP (i, 4))/dpk; %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (n).

a= [TablaElementos (i, 5), TablaElementos (i, 6), TablaElementos (i, 7)]; %vector con los cosenos directores del eje x.
b=[TablaElementos(i,8),TablaElementos(i,9),TablaElementos(i,10)];%vector con los cosenos directores del eje x.
vectZ=cross(a, b); %producto cruz entre los vectores a, b con el fin de obtener los cosenos directores del eje z.
magnitud=norm (vectZ);
vectorunitarioZ=vectZ/magnitud;
```

%COSENOS DIRECTORES "EJE Z"

```
TablaElementos(i,11)=vectorunitarioZ (1); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (l).
TablaElementos(i,12)=vectorunitarioZ (2); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (m).
TablaElementos(i,13)=vectorunitarioZ (3); %Ubico en la posición correspondiente al elemento de la superficie en la tabla de elementos el coseno director (n).
```

```
        Distancias (i, 2)=dip
        Distancias (i, 3)=dik
        Distancias (i, 4)=dpk

end

for j=1:((length (CoordenadasFinales (:,1)))/3)
        CoordenadasLocales((3*j-1),3)=((CoordenadasFinales((3*j-1),2)-CoordenadasFinales((3*j-
        2),2))^2+(CoordenadasFinales((3*j-1),3)-CoordenadasFinales((3*j-
        2),3))^2+(CoordenadasFinales((3*j-1),4)-CoordenadasFinales((3*j-2),4))^2)^(1/2);
        CoordenadasLocales (3*j, 2)=Distancias (j, 4);
        CoordenadasLocales (3*j, 3)=Distancias (j, 2);
 End

%-------------------------------------------------------------------------------------------------
%POBLACIÓN INICIAL MOUPSO
%-------------------------------------------------------------------------------------------------

xin=zeros(numpar, Elementos); %Inicia el vector que contendrá la población inicial.
for i=1:numpar
   for j=1:Elementos
        xin(i,j)=(tmin+rand*(tmax-tmin)); %Llena aleatoriamente el vector de posiciones iniciales,
        dependiendo del rango definido por tmax y tmin.
   end
end

%-------------------------------------------------------------------------------------------------
%VELOCIDAD INICIAL MOUPSO
%-------------------------------------------------------------------------------------------------
vin=zeros(numpar, Elementos); %Inicia el vector que contendrá las velocidades iniciales de las
partículas.
for i=1:numpar
   for j=1:Elementos
```

```
        vin(i,j)=tmin+rand*(tmax-tmin); %Llena aleatoriamente el vector de velocidades  iniciales,
        dependiendo del rango definido por tmax y tmin.
    end
end

G=vin;
L=vin;

%-----------------------------------------------------------------------------------------------
%EVALUACIÓN INICIAL DE FUNCIÓN OBJETIVO MOUPSO
%-----------------------------------------------------------------------------------------------

fo1=zeros(numpar,1); %Inicia el vector que contendrá los valores de la evaluación de la función
objetivo 1.
fo2=zeros(numpar,1); %Inicia el vector que contendrá los valores de la evaluación de la función
objetivo 2.
FO=zeros(numpar,1);  %Inicia el vector que contendrá los valores de la evaluación de la función
objetivo que reúne las dos funciones objetivo.

%Para las restricciones
TABLAOPTIMIZACION=zeros (numit, 3);

%-----------------------------------------------------------------------------------------------
%PROGRAMACION DEL METODO DE ELEMENTOS FINITOS PARA SHELL
%-----------------------------------------------------------------------------------------------

for k=1:numpar
MATRIZRIGIDEZ=zeros(max(CoordenadasLocales(:,1))*6,max(CoordenadasLocales(:,1))*6);
    for i=1:length(TablaElementos(:,1))
        A=[CoordenadasLocales(3*i-2,2),CoordenadasLocales(3*i-2,3),CoordenadasLocales(3*i-
        2,4)];
        B=[CoordenadasLocales(3*i-1,2),CoordenadasLocales(3*i-1,3),CoordenadasLocales(3*i-
        1,4)];
```

```
C=[CoordenadasLocales(3*i,2),CoordenadasLocales(3*i,3),CoordenadasLocales(3*i,4)];
AB=B-A;
AC=C-A;
r=cross(AB,AC);
AR=norm(r)/2;
TablaElementos(i,14)=AR;% Asigno el área para cada elemento de la superficie
CoordenadasLocales((3*i-2):3*i,5)=TablaElementos(i,14)/3;
CoordenadasLocales((3*i-2):3*i,6)=CoordenadasLocales((3*i-2):3*i,5)*xin(k,i);
E=2.15381e10;%N/m2
v=0.2;
p=23544 ; % Peso específico del concreto N/m3
a=(E*(xin(k,i)))/(4*AR*(1-v^2));
b=(E*(xin(k,i)))/(8*AR*(1+v));
c=(E*(xin(k,i)^3))/(12*(1-v^2));

% MATRIZ DE RIGIDEZ PLACA

MATRIZPLACA=zeros (9,9);

%FILA 4
MATRIZPLACA (4,4)=2*c*C (1)*B (2);
MATRIZPLACA (4,6)=2*c*v*C (1)*B (2);
MATRIZPLACA (4,7)=2*c*(C (1)) ^2*B (2);
MATRIZPLACA(4,8)=(2/3)*c*v*(C(1))^2*B(2)+(2/3)*c*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA (4,9)=2*c*v*C (1)*B (2)*(B (2)+C (2));
%FILA 5
MATRIZPLACA (5,5)= (1-v)*c*C (1)*B (2);
MATRIZPLACA(5,8)=(2/3)*(1-v)*c*(C(1))^2*B(2)+(2/3)*(1-
v)*c*C(1)*B(2)*(B(2)+C(2));
%FILA 6
MATRIZPLACA (6,4)=2*c*v*C (1)*B (2);
MATRIZPLACA (6,6)=2*c*C (1)*B (2);
MATRIZPLACA (6,7)=2*c*v*(C (1)) ^2*B (2);
```

MATRIZPLACA(6,8)=(2/3)*c*(C(1))^2*B(2)+(2/3)*c*v*C(1)*B(2)*(B(2)+C(2));

MATRIZPLACA (6,9)=2*c*C (1)*B (2)*(B (2)+C (2));

%FILA 7

MATRIZPLACA (7,4)=2*c*(C (1)) ^2*B (2);

MATRIZPLACA (7,6)=2*c*v*(C (1)) ^2*B (2);

MATRIZPLACA (7,7)=3*c*(C (1)) ^3*B (2);

MATRIZPLACA(7,8)=c*v*(C(1))^3*B(2)+(1/2)*c*(C(1))^2*B(2)*(B(2)+(2*C(2)));

MATRIZPLACA (7,9)= (3/2)*c*v*(C (1)) ^2*B (2)*(B (2)+2*C (2));

%FILA 8

MATRIZPLACA(8,4)=(2/3)*c*v*(C(1))^2*B(2)+(2/3)*c*C(1)*B(2)*(B(2)+C(2));

MATRIZPLACA(8,5)=(2/3)*(1-v)*c*(C(1))^2*B(2)+(2/3)*(1-
v)*c*C(1)*B(2)*(B(2)+C(2));

MATRIZPLACA(8,6)=(2/3)*c*(C(1))^2*B(2)+(2/3)*c*v*C(1)*B(2)*(B(2)+C(2));

MATRIZPLACA(8,7)=c*v*(C(1))^3*B(2)+(1/2)*c*(C(1))^2*B(2)*(B(2)+(2*C(2)));

MATRIZPLACA(8,8)=c*((128*v)*((((C(1))^3*B(2))/12)+(1/12)*(C(1))^2*B(2)*(B(2)+2*
(C(2)))+(1/12)*C(1)*B(2)*((B(2))^2+B(2)*C(2)+(C(2))^2))-(8*(1-
v))*((1/24)*(C(1))^2*B(2)*(B(2)+2*(C(2)))));

MATRIZPLACA(8,9)=(c/2)*(C(1))^2*B(2)*(B(2)+2*C(2))+c*v*C(1)*B(2)*((B(2))^2+B(
2)*C(2)+(C(2))^2);

%FILA 9

MATRIZPLACA (9,4)=2*c*v*C (1)*B (2)*(B (2)+C (2));

MATRIZPLACA (9,6)=2*c*C (1)*B (2)*(B (2)+C (2));

MATRIZPLACA (9,7)= (3/2)*c*v*(C (1)) ^2*B (2)*(B (2)+2*C (2));

MATRIZPLACA(9,8)=(c/2)*(C(1))^2*B(2)*(B(2)+2*C(2))+c*v*C(1)*B(2)*((B(2))^2+B(
2)*C(2)+(C(2))^2);

MATRIZPLACA (9,9)=3*c*C (1)*B (2)*((B (2)) ^2+B (2)*C (2)+C (2) ^2);

MATRIZN=zeros (9,9);

%FILA1

MATRIZN (1,1)=1;

%FILA2

MATRIZN (2,3)=1;

%FILA3

MATRIZN (3,2)=-1;

%FILA4

MATRIZN (4,1)=1;

MATRIZN (4,3)=B (2);

MATRIZN (4,6)= (B (2)) ^2;

MATRIZN (4,9)= (B (2)) ^3;

%FILA5

MATRIZN (5,3)=1;

MATRIZN (5,6)=2*B (2);

MATRIZN (5,9)=3*(B (2)) ^2;

%FILA6

MATRIZN (6,2)=-1;

MATRIZN (6,5)=-B (2);

MATRIZN (6,8)=-(B (2)) ^2;

%FILA7

MATRIZN (7,1)=1;

MATRIZN (7,2)=C (1);

MATRIZN (7,3)=C (2);

MATRIZN (7,4)=(C (1)) ^2;

MATRIZN (7,5)=C (1)*C (2);

MATRIZN (7,6)=(C (2)) ^2;

MATRIZN (7,7)=(C (1)) ^3;

MATRIZN (7,8)=(C (1)) ^2*C (2)+C (1)*(C (2)) ^2;

MATRIZN (7,9)=(C (2)) ^3;

%FILA8

MATRIZN (8,3)=1;

MATRIZN (8,5)=C (1);

MATRIZN (8,6)=2*C (2);

MATRIZN (8,8)=2*C (1)*C (2)+(C (1)) ^2;

MATRIZN (8,9)=3*(C (2)) ^2;

%FILA9

MATRIZN (9,2)=-1;

```
MATRIZN (9,4)=-2*C (1);
MATRIZN (9,5)=-C (2);
MATRIZN (9,7)=-3*(C (1)) ^2;
MATRIZN (9,8)=-((C (2)) ^2+2*C (1)*C (2));

MATRIZNINV=inv (MATRIZN);
MATRIZNTRANSP=transp (MATRIZNINV);
KPLACA=MATRIZNTRANSP*MATRIZPLACA*MATRIZNINV;

% MATRIZ DE RIGIDEZ MEMBRANA

KMEMBRANA=zeros (6,6);
%FILA 1
KMEMBRANA (1,1)=a*(B (2)-C (2)) ^2+b*(B (1)-C (1)) ^2;
KMEMBRANA (1,2)=-a*v*(B (2)-C (2))*(B (1)-C (1))-b*(B (1)-C (1))*(B (2)-C (2));
KMEMBRANA (1,3)=-a*(B (2)-C (2))*(A (2)-C (2))-b*(B (1)-C (1))*(A (1)-C (1));
KMEMBRANA (1,4)=a*v*(B (2)-C (2))*(A (1)-C (1))+b*(B (1)-C (1))*(A (2)-C (2));
KMEMBRANA (1,5)=a*(B (2)-C (2))*(A (2)-B (2))+b*(B (1)-C (1))*(A (1)-B (1));
KMEMBRANA (1,6)=-a*v*(B (2)-C (2))*(A (1)-B (1))-b*(B (1)-C (1))*(A (2)-B (2));
%FILA 2
KMEMBRANA (2,1)=-a*v*(B (2)-C (2))*(B (1)-C (1))-b*(B (1)-C (1))*(B (2)-C (2));
KMEMBRANA (2,2)=b*(B (2)-C (2)) ^2+a*(B (1)-C (1)) ^2;
KMEMBRANA (2,3)=a*v*(B (1)-C (1))*(A (2)-C (2))+b*(B (2)-C (2))*(A (1)-C (1));
KMEMBRANA (2,4)=-a*(B (1)-C (1))*(A (1)-C (1))-b*(B (2)-C (2))*(A (2)-C (2));
KMEMBRANA (2,5)=-a*v*(B (1)-C (1))*(A (2)-B (2))-b*(B (2)-C (2))*(A (1)-B (1));
KMEMBRANA (2,6)=a*(B (1)-C (1))*(A (1)-B (1))+b*(B (2)-C (2))*(A (2)-B (2));
%FILA 3
KMEMBRANA (3,1)=-a*(B (2)-C (2))*(A (2)-C (2))-b*(B (1)-C (1))*(A (1)-C (1));
KMEMBRANA (3,2)=a*v*(B (1)-C (1))*(A (2)-C (2))+b*(B (2)-C (2))*(A (1)-C (1));
KMEMBRANA (3,3)=a*(A (2)-C (2)) ^2+b*(A (1)-C (1)) ^2;
KMEMBRANA (3,4)=-a*v*(A (2)-C (2))*(A (1)-C (1))-b*(A (1)-C (1))*(A (2)-C (2));
KMEMBRANA (3,5)=-a*(A (2)-C (2))*(A (2)-B (2))-b*(A (1)-C (1))*(A (1)-B (1));
KMEMBRANA (3,6)=a*v*(A (2)-C (2))*(A (1)-B (1))+b*(A (1)-C (1))*(A (2)-B (2));
```

```
%FILA 4
KMEMBRANA (4,1)=a*v*(B (2)-C (2))*(A (1)-C (1))+b*(B (1)-C (1))*(A (2)-C (2));
KMEMBRANA (4,2)=-a*(B (1)-C (1))*(A (1)-C (1))-b*(B (2)-C (2))*(A (2)-C (2));
KMEMBRANA (4,3)=-a*v*(A (2)-C (2))*(A (1)-C (1))-b*(A (1)-C (1))*(A (2)-C (2));
KMEMBRANA (4,4)=b*(A (2)-C (2)) ^2+a*(A (1)-C (1)) ^2;
KMEMBRANA (4,5)=a*v*(A (1)-C (1))*(A (2)-B (2))+b*(A (2)-C (2))*(A (1)-B (1));
KMEMBRANA (4,6)=-a*(A (1)-C (1))*(A (1)-B (1))-b*(A (2)-C (2))*(A (2)-B (2));
%FILA 5
KMEMBRANA (5,1)=a*(B (2)-C (2))*(A (2)-B (2))+b*(B (1)-C (1))*(A (1)-B (1));
KMEMBRANA (5,2)=-a*v*(B (1)-C (1))*(A (2)-B (2))-b*(B (2)-C (2))*(A (1)-B (1));
KMEMBRANA (5,3)=-a*(A (2)-C (2))*(A (2)-B (2))-b*(A (1)-C (1))*(A (1)-B (1));
KMEMBRANA (5,4)=a*v*(A (1)-C (1))*(A (2)-B (2))+b*(A (2)-C (2))*(A (1)-B (1));
KMEMBRANA (5,5)=a*(A (2)-B (2)) ^2+b*(A (1)-B (1)) ^2;
KMEMBRANA (5,6)=-a*v*(A (2)-B (2))*(A (1)-B (1))-b*(A (1)-B (1))*(A (2)-B (2));
%FILA 6
KMEMBRANA (6,1)=-a*v*(B (2)-C (2))*(A (1)-B (1))-b*(B (1)-C (1))*(A (2)-B (2));
KMEMBRANA (6,2)=a*(B (1)-C (1))*(A (1)-B (1))+b*(B (2)-C (2))*(A (2)-B (2));
KMEMBRANA (6,3)=a*v*(A (2)-C (2))*(A (1)-B (1))+b*(A (1)-C (1))*(A (2)-B (2));
KMEMBRANA (6,4)=-a*(A (1)-C (1))*(A (1)-B (1))-b*(A (2)-C (2))*(A (2)-B (2));
KMEMBRANA (6,5)=-a*v*(A (2)-B (2))*(A (1)-B (1))-b*(A (1)-B (1))*(A (2)-B (2));
KMEMBRANA (6,6)=b*(A (2)-B (2)) ^2+a*(A (1)-B (1)) ^2;

% MATRIZ RIGIDEZ TOTAL ELEMENTO
MATRIZK=zeros (18,18);

%FILA 1-2
MATRIZK (1:2,1:2)=KMEMBRANA (1:2,1:2);
MATRIZK (1:2,7:8)=KMEMBRANA (1:2,3:4);
MATRIZK (1:2,13:14)=KMEMBRANA (1:2,5:6);
%FILA 3-5
MATRIZK (3:5,3:5)=KPLACA (1:3,1:3);
MATRIZK (3:5,9:11)=KPLACA (1:3,4:6);
MATRIZK (3:5,15:17)=KPLACA (1:3,7:9);
```

```
%FILA 7-8
MATRIZK (7:8,1:2)=KMEMBRANA (3:4,1:2);
MATRIZK (7:8,7:8)=KMEMBRANA (3:4,3:4);
MATRIZK (7:8,13:14)=KMEMBRANA (3:4,5:6);
%FILA 9-11
MATRIZK (9:11,3:5)=KPLACA (4:6,1:3);
MATRIZK (9:11,9:11)=KPLACA (4:6,4:6);
MATRIZK (9:11,15:17)=KPLACA (4:6,7:9);
%FILA 13-14
MATRIZK (13:14,1:2)=KMEMBRANA (5:6,1:2);
MATRIZK (13:14,7:8)=KMEMBRANA (5:6,3:4);
MATRIZK (13:14,13:14)=KMEMBRANA (5:6,5:6);
%FILA 15-17
MATRIZK (15:17,3:5)=KPLACA (7:9,1:3);
MATRIZK (15:17,9:11)=KPLACA (7:9,4:6);
MATRIZK (15:17,15:17)=KPLACA (7:9,7:9);

% MATRIZ DE TRANSFORMACION
MATRIZTRANSF=zeros (18,18);
MATRIZTRANSF (1,1:3)=TablaElementos (i, 5:7);
MATRIZTRANSF (2,1:3)=TablaElementos (i, 8:10);
MATRIZTRANSF (3,1:3)=TablaElementos (i, 11:13);
MATRIZTRANSF (4:6,4:6)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (7:9,7:9)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (10:12,10:12)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (13:15,13:15)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (16:18,16:18)=MATRIZTRANSF (1:3,1:3);

TRANSPUESTA=MATRIZTRANSF';
MATRIZELEMENTOi=TRANSPUESTA*MATRIZK*MATRIZTRANSF;
R=MATRIZK;
v=genvarname (['K', num2str (i)]);
Eval ([v,'=R']);
```

```
%-------------------------------------------------------------------------------------------------
%ENSAMBLE MATRIZ DE RIGIDEZ TOTAL
%-------------------------------------------------------------------------------------------------

    %FILA 1-6
    for m=5:-1:0
        n=6-m;

         MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        5)+MATRIZELEMENTOi(n,1);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        4)+MATRIZELEMENTOi(n,2);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        3)+MATRIZELEMENTOi(n,3);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        2)+MATRIZELEMENTOi(n,4);
         MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        1)+MATRIZELEMENTOi(n,5);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-
        m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
        m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        5)+MATRIZELEMENTOi(n,7);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        4)+MATRIZELEMENTOi(n,8);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
5)+MATRIZELEMENTOi(n,13);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
4)+MATRIZELEMENTOi(n,14);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
3)+MATRIZELEMENTOi(n,15);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
2)+MATRIZELEMENTOi(n,16);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
1)+MATRIZELEMENTOi(n,17);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
End
```

```
%FILA 7-12
for m=5:-1:0
    n=12-m;
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
5)+MATRIZELEMENTOi(n,1);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
4)+MATRIZELEMENTOi(n,2);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
3)+MATRIZELEMENTOi(n,3);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
2)+MATRIZELEMENTOi(n,4);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
1)+MATRIZELEMENTOi(n,5);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
5)+MATRIZELEMENTOi(n,7);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
4)+MATRIZELEMENTOi(n,8);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
```

```
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
5)+MATRIZELEMENTOi(n,13);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
4)+MATRIZELEMENTOi(n,14);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
3)+MATRIZELEMENTOi(n,15);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
2)+MATRIZELEMENTOi(n,16);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
1)+MATRIZELEMENTOi(n,17);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
end

%FILA 13-18
for m=5:-1:0
    n=18-m;
    MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
    5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
    5)+MATRIZELEMENTOi(n,1);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
4)+MATRIZELEMENTOi(n,2);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
3)+MATRIZELEMENTOi(n,3);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
2)+MATRIZELEMENTOi(n,4);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
1)+MATRIZELEMENTOi(n,5);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
5)+MATRIZELEMENTOi(n,7);
 MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
4)+MATRIZELEMENTOi(n,8);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
```

```
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                5)+MATRIZELEMENTOi(n,13);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                4)+MATRIZELEMENTOi(n,14);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                3)+MATRIZELEMENTOi(n,15);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                2)+MATRIZELEMENTOi(n,16);
                 MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                1)+MATRIZELEMENTOi(n,17);
                 MATRIZRIGIDEZ(6*TablaElementos(i,4)-
                m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
                m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
        end
    end
VectorFuerzas=zeros (max (CoordenadasLocales (:,1))*6,1);

for j=1: max (CoordenadasLocales (:,1))
                z=find (CoordenadasLocales (:,1) ==j);
                VectorFuerzas (6*j-3)=sum (CoordenadasLocales (z, 5))*t*p*-1; % Vector que
                contiene las fuerzas por peso propio por área aferente soportadas por cada nodo en
                direccion –Z.
end

PESO=abs (sum (VectorFuerzas)) %Peso total cubierta
```

```
%-------------------------------------------------------------------------------------------------
%NODOS DE LOS APOYOS
%-------------------------------------------------------------------------------------------------

Apoyos=load ('NodosApoyos.txt'); %Organización y enumeración de nodos por coordenadas X, Y, Z.

CoordenadaApoyos=zeros ((length (Apoyos)/3) ,4); %Inicia la matriz que almacenará las coordenadas de los nodos de la cubierta.

for i=1:(length (Apoyos)/3) % For desde 1 hasta la longitud total de la lista de nodos dividida en tres ya que se va a organizar por cada fila tres coordenadas "X, Y, Z".
    CoordenadaApoyos (i, 1)=0; %Enumera los nodos de la cubierta.
    CoordenadaApoyos (i, 2)=Apoyos ((3*i)-2); %Llena el vector de las coordenadas X de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-2 es porque debe tomar el número 1, 4, 7,... de la lista, es decir, el correspondiente a la coordenada X.
    CoordenadaApoyos (i, 3)=Apoyos ((3*i)-1); %Llena el vector de las coordenadas Y de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-1 es porque debe tomar el número 2, 5, 8,... de la lista, es decir, el correspondiente a la coordenada Y.
    CoordenadaApoyos (i, 4)=Apoyos (3*i); %Llena el vector de las coordenadas Z de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i es porque debe tomar el número 3, 6, 9,... de la lista, es decir, el correspondiente a la coordenada Z.

    ind=find(CoordenadasFinales(:,2)==CoordenadaApoyos(i,2)&CoordenadasFinales(:,3)==C
    oordenadaApoyos(i,3)&CoordenadasFinales(:,4)==CoordenadaApoyos(i,4));
    CoordenadaApoyos (i, 1)=CoordenadasFinales (ind (1) ,1);
End
```

```
%ELIMINACION ROTACION EN Z

Nnodos=length (MATRIZRIGIDEZ (1, :))/6

for i=Nnodos:-1:1
   MATRIZRIGIDEZ (6*i, :)= [];
   MATRIZRIGIDEZ (:,6*i)= [];
   VectorFuerzas (6*i, :)= [];
end
NodosAp=sort ((CoordenadaApoyos (:,1)),'descend'); % Se organizan los nodos de los apoyos en orden descendente.
for i=1:(length(Apoyos)/3)
      MATRIZRIGIDEZ(5*(NodosAp(i))-4:5*(NodosAp(i)),:)=[];
      MATRIZRIGIDEZ(:,5*(NodosAp(i))-4:5*(NodosAp(i)))=[];
      VectorFuerzas(5*(NodosAp(i))-4:5*(NodosAp(i)),:)=[];
end
DESP=linsolve(MATRIZRIGIDEZ,VectorFuerzas);
   e1=unique(CoordenadasFinales(:,1));
   e1(NodosAp)=[];

   for i=1:length(e1)
      DESP(5*i-4:5*i,2)=e1(i);
   end

DESPLAZAMIENTOS=DESP(:,1);
ENERGIA=transp(VectorFuerzas)*DESPLAZAMIENTOS

%-------------------------------------------------------------------------------------------------
%FIN PROGRAMACION DEL METODO DE ELEMENTOS FINITOS PARA SHELL
%-------------------------------------------------------------------------------------------------

fo1(k,1)=ENERGIA; %Evalúa la función objetivo 1 con cada partícula.
```

```
fo2(k,1)=PESO; %Evalúa la función objetivo 2 con cada partícula.
end % end de partícula

%-----------------------------------------------------------------------------------------------------
%EVALUACION DE LA FUNCION OBJETIVO TOTAL
%-----------------------------------------------------------------------------------------------------
for i=1:numpar
        esc=(min(fo1))/(min(fo2))
        FO(i,1)=w1*fo1(i,1)+w2*fo2(i,1)*esc; %Las funciones objetivo se suman en una sola con
        pesos iguales a 0.5, esto se hace solo en esta primera evaluación de la función objetivo,
        posteriormente los pesos son dinámicos y los encuentra el PSO.
end

%-----------------------------------------------------------------------------------------------------
%MEJOR POSICIÓN DE CADA PARTÍCULA MOUPSO
%-----------------------------------------------------------------------------------------------------

xbest=zeros(numpar, Elementos); %Inicia el vector que contendrá las mejores posiciones
encontradas por cada partícula.

for i=1:numpar
   for j=1:Elementos
        xbest(i,j)=xin(i,j); %Llena el vector de mejores posiciones de cada partícula con las mismas
        posiciones iniciales (por lo que es la primera iteración).
   end
end

FOxbest=FO
```

```
%-------------------------------------------------------------------------------------------------
%MEJOR POSICIÓN DEL ENJAMBRE MOUPSO
%-------------------------------------------------------------------------------------------------

xbestglob=zeros(1,Elementos);
[bestglob,ibestglob]=min(FO) %Obtiene el valor mínimo de la función objetivo y el índice en
donde se localiza.

for j=1:Elementos
   xbestglob(1,j)=xin(ibestglob,j);
end

%-------------------------------------------------------------------------------------------------
%MEJOR POSICIÓN DEL VECINDARIO MOUPSO
%-------------------------------------------------------------------------------------------------

xbestveci=zeros(numpar, Elementos); %Inicia el vector que contendrá las mejores posiciones
encontradas en el vecindario de cada partícula (vecindario=partícula i-1, i, i+1)

if FO(1,1)<FO(2,1)
   for j=1:Elementos
         xbestveci(1,j)=xin(1,j); %Si la función objetivo evaluada en la partícula 1 es menor que la
         función objetivo evaluada en la partícula 2, la mejor posición del vecindario de la partícula
         1 es xin(1,j).
   end
else
   for j=1:Elementos
         xbestveci(1,j)=xin(2,j); %Si la función objetivo evaluada en la partícula 1 es mayor que la
         función objetivo evaluada en la partícula 2, la mejor posición del vecindario de la partícula
         1 es xin(2,j).
   end
end
```

```
for i=2:(numpar-1)
   if FO(i,1)<FO(i-1,1)&&FO(i,1)<FO(i+1,1)
      for j=1:Elementos
         xbestveci(i,j)=xin(i,j); %Si la función objetivo evaluada en la partícula i es menor que la
         función objetivo evaluada en la partícula i-1 y en la partícula i+1, la mejor posición del
         vecindario es la de la partícula i.
      end
   else
      if FO(i-1,1)<FO(i+1,1)
         for j=1:Elementos
                  xbestveci(i,j)=xin(i-1,j); %Si la mejor posición del vecindario NO es la de la
                  partícula i, puede ser entonces la de la partícula i-1 o la de la partícula i+1. Luego,
                  si la función objetivo evaluada en la partícula i-1 es menor que la función objetivo
                  evaluada en la partícula i+1, la mejor posición del vecindario es la de la partícula i-
                  1.
         end
      else
         for j=1:Elementos
                  xbestveci(i,j)=xin(i+1,j); %Si la mejor partícula del vecindario no es ni la i, ni la i-
                  1, será la i+1.
         end
      end
   end
end

if FO(numpar,1)<FO(numpar-1,1)
   for j=1:Elementos
         xbestveci(numpar,j)=xin(numpar,j); %Si la función objetivo evaluada en la partícula
         numpar es menor que la función objetivo evaluada en la partícula numpar-1, la mejor
         posición del vecindario de la partícula numpar es xin(numpar,j).
   end
else
   for j=1:Elementos
```

```
            xbestveci(numpar,j)=xin(numpar-1,j) %Si la función objetivo evaluada en la partícula numpar es mayor que la función objetivo evaluada en la partícula numpar-1, la mejor posición del vecindario de la partícula numpar es xin(numpar-1,j).
        end
    end

%-----------------------------------------------------------------------------------------------------
%Evaluación de la función objetivo con las mejores posiciones del vecindario
%-----------------------------------------------------------------------------------------------------

fo1xbestveci=zeros(numpar,1); %Inicia el vector que contendrá los valores de la evaluación de la función objetivo 1.
fo2xbestveci=zeros(numpar,1); %Inicia el vector que contendrá los valores de la evaluación de la función objetivo 2.
FOxbestveci=zeros(numpar,1);  %Inicia el vector que contendrá los valores de la evaluación de la función objetivo que reúne las dos funciones objetivo.


%-----------------------------------------------------------------------------------------------------
%PROGRAMACION DEL METODO DE ELEMENTOS FINITOS PARA SHELL
%-----------------------------------------------------------------------------------------------------
for k=1:numpar
MATRIZRIGIDEZ=zeros(max(CoordenadasLocales(:,1))*6,max(CoordenadasLocales(:,1))*6);
    for i=1:length(TablaElementos(:,1))
        A=[CoordenadasLocales(3*i-2,2),CoordenadasLocales(3*i-2,3),CoordenadasLocales(3*i-2,4)];
        B=[CoordenadasLocales(3*i-1,2),CoordenadasLocales(3*i-1,3),CoordenadasLocales(3*i-1,4)];
        C=[CoordenadasLocales(3*i,2),CoordenadasLocales(3*i,3),CoordenadasLocales(3*i,4)];
        AB=B-A;
        AC=C-A;
        r=cross(AB,AC);
        AR=norm(r)/2;
```

```
TablaElementos(i,14)=AR;% Asigno el área para cada elemento de la superficie
CoordenadasLocales((3*i-2):3*i,5)=TablaElementos(i,14)/3;
CoordenadasLocales((3*i-2):3*i,6)=CoordenadasLocales((3*i-2):3*i,5)* xbestveci(k,i);
E=2.15381e10;%N/m2
v=0.2;
p=23544 ; % Peso específico del concreto N/m3
a=(E*(xbestveci(k,i)))/(4*AR*(1-v^2));
b=(E*(xbestveci(k,i)))/(8*AR*(1+v));
c=(E*(xbestveci(k,i)^3))/(12*(1-v^2));


% MATRIZ DE RIGIDEZ PLACA

MATRIZPLACA=zeros (9,9);

%FILA 4
MATRIZPLACA (4,4)=2*c*C (1)*B (2);
MATRIZPLACA (4,6)=2*c*v*C (1)*B (2);
MATRIZPLACA (4,7)=2*c*(C (1)) ^2*B (2);
MATRIZPLACA(4,8)=(2/3)*c*v*(C(1))^2*B(2)+(2/3)*c*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA (4,9)=2*c*v*C (1)*B (2)*(B (2)+C (2));
%FILA 5
MATRIZPLACA (5,5)= (1-v)*c*C (1)*B (2);
MATRIZPLACA(5,8)=(2/3)*(1-v)*c*(C(1))^2*B(2)+(2/3)*(1-
v)*c*C(1)*B(2)*(B(2)+C(2));
%FILA 6
MATRIZPLACA (6,4)=2*c*v*C (1)*B (2);
MATRIZPLACA (6,6)=2*c*C (1)*B (2);
MATRIZPLACA (6,7)=2*c*v*(C (1)) ^2*B (2);
MATRIZPLACA(6,8)=(2/3)*c*(C(1))^2*B(2)+(2/3)*c*v*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA (6,9)=2*c*C (1)*B (2)*(B (2)+C (2));
%FILA 7
MATRIZPLACA (7,4)=2*c*(C (1)) ^2*B (2);
```

```
MATRIZPLACA (7,6)=2*c*v*(C (1)) ^2*B (2);
MATRIZPLACA (7,7)=3*c*(C (1)) ^3*B (2);
MATRIZPLACA(7,8)=c*v*(C(1))^3*B(2)+(1/2)*c*(C(1))^2*B(2)*(B(2)+(2*C(2)));
MATRIZPLACA (7,9)= (3/2)*c*v*(C (1)) ^2*B (2)*(B (2)+2*C (2));
%FILA 8
MATRIZPLACA(8,4)=(2/3)*c*v*(C(1))^2*B(2)+(2/3)*c*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA(8,5)=(2/3)*(1-v)*c*(C(1))^2*B(2)+(2/3)*(1-
v)*c*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA(8,6)=(2/3)*c*(C(1))^2*B(2)+(2/3)*c*v*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA(8,7)=c*v*(C(1))^3*B(2)+(1/2)*c*(C(1))^2*B(2)*(B(2)+(2*C(2)));
MATRIZPLACA(8,8)=c*((128*v)*((((C(1))^3*B(2))/12)+(1/12)*(C(1))^2*B(2)*(B(2)+2*
(C(2)))+(1/12)*C(1)*B(2)*((B(2))^2+B(2)*C(2)+(C(2))^2))-(8*(1-
v))*((1/24)*(C(1))^2*B(2)*(B(2)+2*(C(2)))));
MATRIZPLACA(8,9)=(c/2)*(C(1))^2*B(2)*(B(2)+2*C(2))+c*v*C(1)*B(2)*((B(2))^2+B(
2)*C(2)+(C(2))^2);
%FILA 9
MATRIZPLACA (9,4)=2*c*v*C (1)*B (2)*(B (2)+C (2));
MATRIZPLACA (9,6)=2*c*C (1)*B (2)*(B (2)+C (2));
MATRIZPLACA (9,7)= (3/2)*c*v*(C (1)) ^2*B (2)*(B (2)+2*C (2));

MATRIZPLACA(9,8)=(c/2)*(C(1))^2*B(2)*(B(2)+2*C(2))+c*v*C(1)*B(2)*((B(2))^2+B(
2)*C(2)+(C(2))^2);
MATRIZPLACA (9,9)=3*c*C (1)*B (2)*((B (2)) ^2+B (2)*C (2)+C (2) ^2);

 MATRIZN=zeros (9,9);
%FILA1
 MATRIZN (1,1)=1;
%FILA2
MATRIZN (2,3)=1;
%FILA3
MATRIZN (3,2)=-1;
%FILA4
MATRIZN (4,1)=1;
```

MATRIZN (4,3)=B (2);

MATRIZN (4,6)= (B (2)) ^2;

MATRIZN (4,9)= (B (2)) ^3;

%FILA5

MATRIZN (5,3)=1;

MATRIZN (5,6)=2*B (2);

MATRIZN (5,9)=3*(B (2)) ^2;

%FILA6

MATRIZN (6,2)=-1;

MATRIZN (6,5)=-B (2);

MATRIZN (6,8)=-(B (2)) ^2;

%FILA7

MATRIZN (7,1)=1;

MATRIZN (7,2)=C (1);

MATRIZN (7,3)=C (2);

MATRIZN (7,4)=(C (1)) ^2;

MATRIZN (7,5)=C (1)*C (2);

MATRIZN (7,6)=(C (2)) ^2;

MATRIZN (7,7)=(C (1)) ^3;

MATRIZN (7,8)=(C (1)) ^2*C (2)+C (1)*(C (2)) ^2;

MATRIZN (7,9)=(C (2)) ^3;

%FILA8

MATRIZN (8,3)=1;

MATRIZN (8,5)=C (1);

MATRIZN (8,6)=2*C (2);

MATRIZN (8,8)=2*C (1)*C (2)+(C (1)) ^2;

MATRIZN (8,9)=3*(C (2)) ^2;

%FILA9

MATRIZN (9,2)=-1;

MATRIZN (9,4)=-2*C (1);

MATRIZN (9,5)=-C (2);

MATRIZN (9,7)=-3*(C (1)) ^2;

MATRIZN (9,8)=-((C (2)) ^2+2*C (1)*C (2));

```
MATRIZNINV=inv (MATRIZN);
MATRIZNTRANSP=transp (MATRIZNINV);
KPLACA=MATRIZNTRANSP*MATRIZPLACA*MATRIZNINV;

% MATRIZ DE RIGIDEZ MEMBRANA

KMEMBRANA=zeros (6,6);
%FILA 1
KMEMBRANA (1,1)=a*(B (2)-C (2)) ^2+b*(B (1)-C (1)) ^2;
KMEMBRANA (1,2)=-a*v*(B (2)-C (2))*(B (1)-C (1))-b*(B (1)-C (1))*(B (2)-C (2));
KMEMBRANA (1,3)=-a*(B (2)-C (2))*(A (2)-C (2))-b*(B (1)-C (1))*(A (1)-C (1));
KMEMBRANA (1,4)=a*v*(B (2)-C (2))*(A (1)-C (1))+b*(B (1)-C (1))*(A (2)-C (2));
KMEMBRANA (1,5)=a*(B (2)-C (2))*(A (2)-B (2))+b*(B (1)-C (1))*(A (1)-B (1));
KMEMBRANA (1,6)=-a*v*(B (2)-C (2))*(A (1)-B (1))-b*(B (1)-C (1))*(A (2)-B (2));
%FILA 2
KMEMBRANA (2,1)=-a*v*(B (2)-C (2))*(B (1)-C (1))-b*(B (1)-C (1))*(B (2)-C (2));
KMEMBRANA (2,2)=b*(B (2)-C (2)) ^2+a*(B (1)-C (1)) ^2;
KMEMBRANA (2,3)=a*v*(B (1)-C (1))*(A (2)-C (2))+b*(B (2)-C (2))*(A (1)-C (1));
KMEMBRANA (2,4)=-a*(B (1)-C (1))*(A (1)-C (1))-b*(B (2)-C (2))*(A (2)-C (2));
KMEMBRANA (2,5)=-a*v*(B (1)-C (1))*(A (2)-B (2))-b*(B (2)-C (2))*(A (1)-B (1));
KMEMBRANA (2,6)=a*(B (1)-C (1))*(A (1)-B (1))+b*(B (2)-C (2))*(A (2)-B (2));
%FILA 3
KMEMBRANA (3,1)=-a*(B (2)-C (2))*(A (2)-C (2))-b*(B (1)-C (1))*(A (1)-C (1));
KMEMBRANA (3,2)=a*v*(B (1)-C (1))*(A (2)-C (2))+b*(B (2)-C (2))*(A (1)-C (1));
KMEMBRANA (3,3)=a*(A (2)-C (2)) ^2+b*(A (1)-C (1)) ^2;
KMEMBRANA (3,4)=-a*v*(A (2)-C (2))*(A (1)-C (1))-b*(A (1)-C (1))*(A (2)-C (2));
KMEMBRANA (3,5)=-a*(A (2)-C (2))*(A (2)-B (2))-b*(A (1)-C (1))*(A (1)-B (1));
KMEMBRANA (3,6)=a*v*(A (2)-C (2))*(A (1)-B (1))+b*(A (1)-C (1))*(A (2)-B (2));
%FILA 4
KMEMBRANA (4,1)=a*v*(B (2)-C (2))*(A (1)-C (1))+b*(B (1)-C (1))*(A (2)-C (2));
KMEMBRANA (4,2)=-a*(B (1)-C (1))*(A (1)-C (1))-b*(B (2)-C (2))*(A (2)-C (2));
KMEMBRANA (4,3)=-a*v*(A (2)-C (2))*(A (1)-C (1))-b*(A (1)-C (1))*(A (2)-C (2));
KMEMBRANA (4,4)=b*(A (2)-C (2)) ^2+a*(A (1)-C (1)) ^2;
```

KMEMBRANA (4,5)=a*v*(A (1)-C (1))*(A (2)-B (2))+b*(A (2)-C (2))*(A (1)-B (1));
KMEMBRANA (4,6)=-a*(A (1)-C (1))*(A (1)-B (1))-b*(A (2)-C (2))*(A (2)-B (2));

%FILA 5

KMEMBRANA (5,1)=a*(B (2)-C (2))*(A (2)-B (2))+b*(B (1)-C (1))*(A (1)-B (1));
KMEMBRANA (5,2)=-a*v*(B (1)-C (1))*(A (2)-B (2))-b*(B (2)-C (2))*(A (1)-B (1));
KMEMBRANA (5,3)=-a*(A (2)-C (2))*(A (2)-B (2))-b*(A (1)-C (1))*(A (1)-B (1));
KMEMBRANA (5,4)=a*v*(A (1)-C (1))*(A (2)-B (2))+b*(A (2)-C (2))*(A (1)-B (1));
KMEMBRANA (5,5)=a*(A (2)-B (2)) ^2+b*(A (1)-B (1)) ^2;
KMEMBRANA (5,6)=-a*v*(A (2)-B (2))*(A (1)-B (1))-b*(A (1)-B (1))*(A (2)-B (2));

%FILA 6

KMEMBRANA (6,1)=-a*v*(B (2)-C (2))*(A (1)-B (1))-b*(B (1)-C (1))*(A (2)-B (2));
KMEMBRANA (6,2)=a*(B (1)-C (1))*(A (1)-B (1))+b*(B (2)-C (2))*(A (2)-B (2));
KMEMBRANA (6,3)=a*v*(A (2)-C (2))*(A (1)-B (1))+b*(A (1)-C (1))*(A (2)-B (2));
KMEMBRANA (6,4)=-a*(A (1)-C (1))*(A (1)-B (1))-b*(A (2)-C (2))*(A (2)-B (2));
KMEMBRANA (6,5)=-a*v*(A (2)-B (2))*(A (1)-B (1))-b*(A (1)-B (1))*(A (2)-B (2));
KMEMBRANA (6,6)=b*(A (2)-B (2)) ^2+a*(A (1)-B (1)) ^2;

% MATRIZ RIGIDEZ TOTAL ELEMENTO

MATRIZK=zeros (18,18);

%FILA 1-2

MATRIZK (1:2,1:2)=KMEMBRANA (1:2,1:2);
MATRIZK (1:2,7:8)=KMEMBRANA (1:2,3:4);
MATRIZK (1:2,13:14)=KMEMBRANA (1:2,5:6);

%FILA 3-5

MATRIZK (3:5,3:5)=KPLACA (1:3,1:3);
MATRIZK (3:5,9:11)=KPLACA (1:3,4:6);
MATRIZK (3:5,15:17)=KPLACA (1:3,7:9);

%FILA 7-8

MATRIZK (7:8,1:2)=KMEMBRANA (3:4,1:2);
MATRIZK (7:8,7:8)=KMEMBRANA (3:4,3:4);

```
MATRIZK (7:8,13:14)=KMEMBRANA (3:4,5:6);
%FILA 9-11
MATRIZK (9:11,3:5)=KPLACA (4:6,1:3);
MATRIZK (9:11,9:11)=KPLACA (4:6,4:6);
MATRIZK (9:11,15:17)=KPLACA (4:6,7:9);
%FILA 13-14
MATRIZK (13:14,1:2)=KMEMBRANA (5:6,1:2);
MATRIZK (13:14,7:8)=KMEMBRANA (5:6,3:4);
MATRIZK (13:14,13:14)=KMEMBRANA (5:6,5:6);
%FILA 15-17
MATRIZK (15:17,3:5)=KPLACA (7:9,1:3);
MATRIZK (15:17,9:11)=KPLACA (7:9,4:6);
MATRIZK (15:17,15:17)=KPLACA (7:9,7:9);

% MATRIZ DE TRANSFORMACION

MATRIZTRANSF=zeros (18,18);
MATRIZTRANSF (1,1:3)=TablaElementos (i, 5:7);
MATRIZTRANSF (2,1:3)=TablaElementos (i, 8:10);
MATRIZTRANSF (3,1:3)=TablaElementos (i, 11:13);
MATRIZTRANSF (4:6,4:6)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (7:9,7:9)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (10:12,10:12)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (13:15,13:15)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (16:18,16:18)=MATRIZTRANSF (1:3,1:3);

TRANSPUESTA=MATRIZTRANSF';
MATRIZELEMENTOi=TRANSPUESTA*MATRIZK*MATRIZTRANSF;
R=MATRIZK;
v=genvarname (['K', num2str (i)]);
Eval ([v,'=R']);
```

```
%-------------------------------------------------------------------------------------------------
%ENSAMBLE MATRIZ DE RIGIDEZ TOTAL
%-------------------------------------------------------------------------------------------------
    %FILA 1-6
    for m=5:-1:0
        n=6-m;

        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        5)+MATRIZELEMENTOi(n,1);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        4)+MATRIZELEMENTOi(n,2);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        3)+MATRIZELEMENTOi(n,3);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        2)+MATRIZELEMENTOi(n,4);
         MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        1)+MATRIZELEMENTOi(n,5);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-
        m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
        m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        5)+MATRIZELEMENTOi(n,7);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        4)+MATRIZELEMENTOi(n,8);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
5)+MATRIZELEMENTOi(n,13);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
4)+MATRIZELEMENTOi(n,14);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
3)+MATRIZELEMENTOi(n,15);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
2)+MATRIZELEMENTOi(n,16);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
1)+MATRIZELEMENTOi(n,17);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
end
```

```
%FILA 7-12
for m=5:-1:0
    n=12-m;
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
5)+MATRIZELEMENTOi(n,1);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
4)+MATRIZELEMENTOi(n,2);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
3)+MATRIZELEMENTOi(n,3);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
2)+MATRIZELEMENTOi(n,4);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
1)+MATRIZELEMENTOi(n,5);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
5)+MATRIZELEMENTOi(n,7);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
4)+MATRIZELEMENTOi(n,8);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
```

```
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
5)+MATRIZELEMENTOi(n,13);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
4)+MATRIZELEMENTOi(n,14);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
3)+MATRIZELEMENTOi(n,15);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
2)+MATRIZELEMENTOi(n,16);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
1)+MATRIZELEMENTOi(n,17);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
end

%FILA 13-18
for m=5:-1:0
    n=18-m;
    MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
    5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
    5)+MATRIZELEMENTOi(n,1);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
4)+MATRIZELEMENTOi(n,2);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
3)+MATRIZELEMENTOi(n,3);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
2)+MATRIZELEMENTOi(n,4);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
1)+MATRIZELEMENTOi(n,5);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
5)+MATRIZELEMENTOi(n,7);
 MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
4)+MATRIZELEMENTOi(n,8);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
```

```
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                5)+MATRIZELEMENTOi(n,13);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                4)+MATRIZELEMENTOi(n,14);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                3)+MATRIZELEMENTOi(n,15);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                2)+MATRIZELEMENTOi(n,16);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                1)+MATRIZELEMENTOi(n,17);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-
                m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
                m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
        end
    end
VectorFuerzas=zeros (max (CoordenadasLocales (:,1))*6,1);

for j=1: max (CoordenadasLocales (:,1))
                z=find (CoordenadasLocales (:,1) ==j);
                VectorFuerzas (6*j-3)=sum (CoordenadasLocales (z, 5))*t*p*-1; % Vector que
                contiene las fuerzas por peso propio por área aferente soportadas por cada nodo en
                direccion –Z.
end

PESO=abs (sum (VectorFuerzas)) %Peso total cubierta
```

```
%-------------------------------------------------------------------------------------------------
%NODOS DE LOS APOYOS
%-------------------------------------------------------------------------------------------------
Apoyos=load ('NodosApoyos.txt'); %Organización y enumeración de nodos por coordenadas X, Y, Z.

CoordenadaApoyos=zeros ((length (Apoyos)/3) ,4); %Inicia la matriz que almacenará las coordenadas de los nodos de la cubierta.

for i=1:(length (Apoyos)/3) % For desde 1 hasta la longitud total de la lista de nodos dividida en tres ya que se va a organizar por cada fila tres coordenadas "X, Y, Z".
    CoordenadaApoyos (i, 1)=0; %Enumera los nodos de la cubierta.
    CoordenadaApoyos (i, 2)=Apoyos ((3*i)-2); %Llena el vector de las coordenadas X de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-2 es porque debe tomar el número 1, 4, 7,... de la lista, es decir, el correspondiente a la coordenada X.
    CoordenadaApoyos (i, 3)=Apoyos ((3*i)-1); %Llena el vector de las coordenadas Y de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-1 es porque debe tomar el número 2, 5, 8,... de la lista, es decir, el correspondiente a la coordenada Y.
    CoordenadaApoyos (i, 4)=Apoyos (3*i); %Llena el vector de las coordenadas Z de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i es porque debe tomar el número 3, 6, 9,... de la lista, es decir, el correspondiente a la coordenada Z.

    ind=find(CoordenadasFinales(:,2)==CoordenadaApoyos(i,2)&CoordenadasFinales(:,3)==C
    oordenadaApoyos(i,3)&CoordenadasFinales(:,4)==CoordenadaApoyos(i,4));
    CoordenadaApoyos (i, 1)=CoordenadasFinales (ind (1) ,1);
end
%ELIMINACION ROTACION EN Z

Nnodos=length (MATRIZRIGIDEZ (1, :))/6
```

```
for i=Nnodos:-1:1
   MATRIZRIGIDEZ (6*i, :)= [];
   MATRIZRIGIDEZ (:,6*i)= [];
   VectorFuerzas (6*i, :)= [];
end
NodosAp=sort ((CoordenadaApoyos (:,1)),'descend'); % Se organizan los nodos de los apoyos en orden descendente.
for i=1:(length(Apoyos)/3)
      MATRIZRIGIDEZ(5*(NodosAp(i))-4:5*(NodosAp(i)),:)=[];
      MATRIZRIGIDEZ(:,5*(NodosAp(i))-4:5*(NodosAp(i)))=[];
      VectorFuerzas(5*(NodosAp(i))-4:5*(NodosAp(i)),:)=[];
end
DESP=linsolve(MATRIZRIGIDEZ,VectorFuerzas);
   e1=unique(CoordenadasFinales(:,1));
   e1(NodosAp)=[];

   for i=1:length(e1)
      DESP(5*i-4:5*i,2)=e1(i);
   end

DESPLAZAMIENTOS=DESP(:,1);
ENERGIA=transp(VectorFuerzas)*DESPLAZAMIENTOS

%-------------------------------------------------------------------------------------------------
%FIN PROGRAMACION DEL METODO DE ELEMENTOS FINITOS PARA SHELL
%-------------------------------------------------------------------------------------------------

fo1xbestveci (k, 1)=ENERGIA; %Evalúa la función objetivo 1 con cada partícula.

fo2xbestveci (k, 1)=PESO; %Evalúa la función objetivo 2 con cada partícula.

end % end de partícula
```

```
%-------------------------------------------------------------------------------------------------
%EVALUACION DE LA FUNCION OBJETIVO TOTAL
%-------------------------------------------------------------------------------------------------
for i=1:numpar
        esc=(min(fo1))/(min(fo2))
        FO (i, 1)=w1*fo1xbestveci (i, 1)+w2*fo2xbestveci (i, 1)*esc; %Las funciones objetivo se
        suman en una sola con pesos iguales a 0.5, esto se hace solo en esta primera evaluación de
        la función objetivo, posteriormente los pesos son dinámicos y los encuentra el PSO.
end
%-------------------------------------------------------------------------------------------------
%CICLO DE OPTIMIZACIÓN
%-------------------------------------------------------------------------------------------------
for j=1:numit
  for k=1:numpar
        MATRIZRIGIDEZ=zeros(max(CoordenadasLocales(:,1))*6,max(CoordenadasLocales(:,1))
        *6);

        for q=1:Elementos
                G(k,q)=X*(vin(k,q)+c1*rand*(xbest(k,q)-xin(k,q))+c2*rand*(xbestglob(1,q)-
                xin(k,q))); %Calcula el componente de la velocidad de la partícula que depende de
                la mejor posición que ha tenido la partícula y de la mejor posición que ha tenido el
                enjambre.

                L(k,q)=X*(vin(k,q)+c1*rand*(xbest(k,q)-xin(k,q))+c2*rand*(xbestveci(k,q)-
                xin(k,q))); %Calcula el componente de la velocidad de la partícula que depende de
                la mejor posición que ha tenido la partícula y de la mejor posición que ha tenido el
                vecindario de la partícula.

                vin(k,q)=u*G(k,q)+(1-u)*L(k,q); %Actualiza la velocidad de la partícula,
                dependiendo del factor de unificación (u) y de los componentes calculados
                anteriormente (G y L).

        xin(k,q)=xin(k,q)+vin(k,q); %Calcula la nueva posición de la partícula.
```

```
        end

        for p=1:Elementos
        if xin(k,p)>tmax
                        v=tmax-tmin;
                        n=v*rand+tmin;
                        xin(k,p)=n;
        end

        if xin(k,p)<tmin
                        v=tmax-tmin;
                        n=v*rand+tmin;
                        xin(k,p)=n;
        end
       end

%-------------------------------------------------------------------------------------------------
%PROGRAMACION DEL METODO DE ELEMENTOS FINITOS PARA SHELL
%-------------------------------------------------------------------------------------------------
        MATRIZRIGIDEZ=zeros(max(CoordenadasLocales(:,1))*6,max(CoordenadasLocales(:,1))
        *6);
   for i=1:length(TablaElementos(:,1))
        A=[CoordenadasLocales(3*i-2,2),CoordenadasLocales(3*i-2,3),CoordenadasLocales(3*i-
        2,4)];
        B=[CoordenadasLocales(3*i-1,2),CoordenadasLocales(3*i-1,3),CoordenadasLocales(3*i-
        1,4)];
        C=[CoordenadasLocales(3*i,2),CoordenadasLocales(3*i,3),CoordenadasLocales(3*i,4)];
        AB=B-A;
        AC=C-A;
        r=cross(AB,AC);
        AR=norm(r)/2;
        TablaElementos(i,14)=AR;% Asigno el área para cada elemento de la superficie
        CoordenadasLocales((3*i-2):3*i,5)=TablaElementos(i,14)/3;
```

```
CoordenadasLocales((3*i-2):3*i,6)=CoordenadasLocales((3*i-2):3*i,5)*xin(k,i);
E=2.15381e10;%N/m2
%t=0.05;
v=0.2;
p=23544 ; % Peso específico del concreto N/m3
a=(E*(xin(k,i)))/(4*AR*(1-v^2));
b=(E*(xin(k,i)))/(8*AR*(1+v));
c=(E*(xin(k,i)^3))/(12*(1-v^2));
% MATRIZ DE RIGIDEZ PLACA

MATRIZPLACA=zeros (9,9);

%FILA 4
MATRIZPLACA (4,4)=2*c*C (1)*B (2);
MATRIZPLACA (4,6)=2*c*v*C (1)*B (2);
MATRIZPLACA (4,7)=2*c*(C (1)) ^2*B (2);
MATRIZPLACA(4,8)=(2/3)*c*v*(C(1))^2*B(2)+(2/3)*c*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA (4,9)=2*c*v*C (1)*B (2)*(B (2)+C (2));
%FILA 5
MATRIZPLACA (5,5)= (1-v)*c*C (1)*B (2);
MATRIZPLACA(5,8)=(2/3)*(1-v)*c*(C(1))^2*B(2)+(2/3)*(1-
v)*c*C(1)*B(2)*(B(2)+C(2));
%FILA 6
MATRIZPLACA (6,4)=2*c*v*C (1)*B (2);
MATRIZPLACA (6,6)=2*c*C (1)*B (2);
MATRIZPLACA (6,7)=2*c*v*(C (1)) ^2*B (2);
MATRIZPLACA(6,8)=(2/3)*c*(C(1))^2*B(2)+(2/3)*c*v*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA (6,9)=2*c*C (1)*B (2)*(B (2)+C (2));
%FILA 7
MATRIZPLACA (7,4)=2*c*(C (1)) ^2*B (2);
MATRIZPLACA (7,6)=2*c*v*(C (1)) ^2*B (2);
MATRIZPLACA (7,7)=3*c*(C (1)) ^3*B (2);
   MATRIZPLACA(7,8)=c*v*(C(1))^3*B(2)+(1/2)*c*(C(1))^2*B(2)*(B(2)+(2*C(2)));
```

MATRIZPLACA (7,9)= (3/2)*c*v*(C (1)) ^2*B (2)*(B (2)+2*C (2));

%FILA 8

MATRIZPLACA(8,4)=(2/3)*c*v*(C(1))^2*B(2)+(2/3)*c*C(1)*B(2)*(B(2)+C(2));

MATRIZPLACA(8,5)=(2/3)*(1-v)*c*(C(1))^2*B(2)+(2/3)*(1-v)*c*C(1)*B(2)*(B(2)+C(2));

MATRIZPLACA(8,6)=(2/3)*c*(C(1))^2*B(2)+(2/3)*c*v*C(1)*B(2)*(B(2)+C(2));

MATRIZPLACA(8,7)=c*v*(C(1))^3*B(2)+(1/2)*c*(C(1))^2*B(2)*(B(2)+(2*C(2)));

MATRIZPLACA(8,8)=c*((128*v)*((((C(1))^3*B(2))/12)+(1/12)*(C(1))^2*B(2)*(B(2)+2*(C(2)))+(1/12)*C(1)*B(2)*((B(2))^2+B(2)*C(2)+(C(2))^2))-(8*(1-v))*((1/24)*(C(1))^2*B(2)*(B(2)+2*(C(2)))));

MATRIZPLACA(8,9)=(c/2)*(C(1))^2*B(2)*(B(2)+2*C(2))+c*v*C(1)*B(2)*((B(2))^2+B(2)*C(2)+(C(2))^2);

%FILA 9

MATRIZPLACA (9,4)=2*c*v*C (1)*B (2)*(B (2)+C (2));

MATRIZPLACA (9,6)=2*c*C (1)*B (2)*(B (2)+C (2));

MATRIZPLACA (9,7)= (3/2)*c*v*(C (1)) ^2*B (2)*(B (2)+2*C (2));

MATRIZPLACA(9,8)=(c/2)*(C(1))^2*B(2)*(B(2)+2*C(2))+c*v*C(1)*B(2)*((B(2))^2+B(2)*C(2)+(C(2))^2);

MATRIZPLACA (9,9)=3*c*C (1)*B (2)*((B (2)) ^2+B (2)*C (2)+C (2) ^2);

MATRIZN=zeros (9,9);

%FILA1

MATRIZN (1,1)=1;

%FILA2

MATRIZN (2,3)=1;

%FILA3

MATRIZN (3,2)=-1;

%FILA4

MATRIZN (4,1)=1;

MATRIZN (4,3)=B (2);

MATRIZN (4,6)= (B (2)) ^2;

MATRIZN (4,9)= (B (2)) ^3;

%FILA5

MATRIZN (5,3)=1;

MATRIZN (5,6)=2*B (2);

MATRIZN (5,9)=3*(B (2)) ^2;

%FILA6

MATRIZN (6,2)=-1;

MATRIZN (6,5)=-B (2);

MATRIZN (6,8)=-(B (2)) ^2;

%FILA7

MATRIZN (7,1)=1;

MATRIZN (7,2)=C (1);

MATRIZN (7,3)=C (2);

MATRIZN (7,4)=(C (1)) ^2;

MATRIZN (7,5)=C (1)*C (2);

MATRIZN (7,6)=(C (2)) ^2;

MATRIZN (7,7)=(C (1)) ^3;

MATRIZN (7,8)=(C (1)) ^2*C (2)+C (1)*(C (2)) ^2;

MATRIZN (7,9)=(C (2)) ^3;

%FILA8

MATRIZN (8,3)=1;

MATRIZN (8,5)=C (1);

MATRIZN (8,6)=2*C (2);

MATRIZN (8,8)=2*C (1)*C (2)+(C (1)) ^2;

MATRIZN (8,9)=3*(C (2)) ^2;

%FILA9

MATRIZN (9,2)=-1;

MATRIZN (9,4)=-2*C (1);

MATRIZN (9,5)=-C (2);

MATRIZN (9,7)=-3*(C (1)) ^2;

MATRIZN (9,8)=-((C (2)) ^2+2*C (1)*C (2));

MATRIZNINV=inv (MATRIZN);

MATRIZNTRANSP=transp (MATRIZNINV);

KPLACA=MATRIZNTRANSP*MATRIZPLACA*MATRIZNINV;

% MATRIZ DE RIGIDEZ MEMBRANA

KMEMBRANA=zeros (6,6);

%FILA 1

KMEMBRANA (1,1)=a*(B (2)-C (2)) ^2+b*(B (1)-C (1)) ^2;

KMEMBRANA (1,2)=-a*v*(B (2)-C (2))*(B (1)-C (1))-b*(B (1)-C (1))*(B (2)-C (2));

KMEMBRANA (1,3)=-a*(B (2)-C (2))*(A (2)-C (2))-b*(B (1)-C (1))*(A (1)-C (1));

KMEMBRANA (1,4)=a*v*(B (2)-C (2))*(A (1)-C (1))+b*(B (1)-C (1))*(A (2)-C (2));

KMEMBRANA (1,5)=a*(B (2)-C (2))*(A (2)-B (2))+b*(B (1)-C (1))*(A (1)-B (1));

KMEMBRANA (1,6)=-a*v*(B (2)-C (2))*(A (1)-B (1))-b*(B (1)-C (1))*(A (2)-B (2));

%FILA 2

KMEMBRANA (2,1)=-a*v*(B (2)-C (2))*(B (1)-C (1))-b*(B (1)-C (1))*(B (2)-C (2));

KMEMBRANA (2,2)=b*(B (2)-C (2)) ^2+a*(B (1)-C (1)) ^2;

KMEMBRANA (2,3)=a*v*(B (1)-C (1))*(A (2)-C (2))+b*(B (2)-C (2))*(A (1)-C (1));

KMEMBRANA (2,4)=-a*(B (1)-C (1))*(A (1)-C (1))-b*(B (2)-C (2))*(A (2)-C (2));

KMEMBRANA (2,5)=-a*v*(B (1)-C (1))*(A (2)-B (2))-b*(B (2)-C (2))*(A (1)-B (1));

KMEMBRANA (2,6)=a*(B (1)-C (1))*(A (1)-B (1))+b*(B (2)-C (2))*(A (2)-B (2));

%FILA 3

KMEMBRANA (3,1)=-a*(B (2)-C (2))*(A (2)-C (2))-b*(B (1)-C (1))*(A (1)-C (1));

KMEMBRANA (3,2)=a*v*(B (1)-C (1))*(A (2)-C (2))+b*(B (2)-C (2))*(A (1)-C (1));

KMEMBRANA (3,3)=a*(A (2)-C (2)) ^2+b*(A (1)-C (1)) ^2;

KMEMBRANA (3,4)=-a*v*(A (2)-C (2))*(A (1)-C (1))-b*(A (1)-C (1))*(A (2)-C (2));

KMEMBRANA (3,5)=-a*(A (2)-C (2))*(A (2)-B (2))-b*(A (1)-C (1))*(A (1)-B (1));

KMEMBRANA (3,6)=a*v*(A (2)-C (2))*(A (1)-B (1))+b*(A (1)-C (1))*(A (2)-B (2));

%FILA 4

KMEMBRANA (4,1)=a*v*(B (2)-C (2))*(A (1)-C (1))+b*(B (1)-C (1))*(A (2)-C (2));

KMEMBRANA (4,2)=-a*(B (1)-C (1))*(A (1)-C (1))-b*(B (2)-C (2))*(A (2)-C (2));

KMEMBRANA (4,3)=-a*v*(A (2)-C (2))*(A (1)-C (1))-b*(A (1)-C (1))*(A (2)-C (2));

KMEMBRANA (4,4)=b*(A (2)-C (2)) ^2+a*(A (1)-C (1)) ^2;

KMEMBRANA (4,5)=a*v*(A (1)-C (1))*(A (2)-B (2))+b*(A (2)-C (2))*(A (1)-B (1));

KMEMBRANA (4,6)=-a*(A (1)-C (1))*(A (1)-B (1))-b*(A (2)-C (2))*(A (2)-B (2));

```
%FILA 5
KMEMBRANA (5,1)=a*(B (2)-C (2))*(A (2)-B (2))+b*(B (1)-C (1))*(A (1)-B (1));
KMEMBRANA (5,2)=-a*v*(B (1)-C (1))*(A (2)-B (2))-b*(B (2)-C (2))*(A (1)-B (1));
KMEMBRANA (5,3)=-a*(A (2)-C (2))*(A (2)-B (2))-b*(A (1)-C (1))*(A (1)-B (1));
KMEMBRANA (5,4)=a*v*(A (1)-C (1))*(A (2)-B (2))+b*(A (2)-C (2))*(A (1)-B (1));
KMEMBRANA (5,5)=a*(A (2)-B (2)) ^2+b*(A (1)-B (1)) ^2;
KMEMBRANA (5,6)=-a*v*(A (2)-B (2))*(A (1)-B (1))-b*(A (1)-B (1))*(A (2)-B (2));
%FILA 6
KMEMBRANA (6,1)=-a*v*(B (2)-C (2))*(A (1)-B (1))-b*(B (1)-C (1))*(A (2)-B (2));
KMEMBRANA (6,2)=a*(B (1)-C (1))*(A (1)-B (1))+b*(B (2)-C (2))*(A (2)-B (2));
KMEMBRANA (6,3)=a*v*(A (2)-C (2))*(A (1)-B (1))+b*(A (1)-C (1))*(A (2)-B (2));
KMEMBRANA (6,4)=-a*(A (1)-C (1))*(A (1)-B (1))-b*(A (2)-C (2))*(A (2)-B (2));
KMEMBRANA (6,5)=-a*v*(A (2)-B (2))*(A (1)-B (1))-b*(A (1)-B (1))*(A (2)-B (2));
KMEMBRANA (6,6)=b*(A (2)-B (2)) ^2+a*(A (1)-B (1)) ^2;

% MATRIZ RIGIDEZ TOTAL ELEMENTO

MATRIZK=zeros (18,18);
%FILA 1-2
MATRIZK (1:2,1:2)=KMEMBRANA (1:2,1:2);
MATRIZK (1:2,7:8)=KMEMBRANA (1:2,3:4);
MATRIZK (1:2,13:14)=KMEMBRANA (1:2,5:6);
%FILA 3-5
MATRIZK (3:5,3:5)=KPLACA (1:3,1:3);
MATRIZK (3:5,9:11)=KPLACA (1:3,4:6);
MATRIZK (3:5,15:17)=KPLACA (1:3,7:9);
%FILA 7-8
MATRIZK (7:8,1:2)=KMEMBRANA (3:4,1:2);
MATRIZK (7:8,7:8)=KMEMBRANA (3:4,3:4);
MATRIZK (7:8,13:14)=KMEMBRANA (3:4,5:6);
%FILA 9-11
MATRIZK (9:11,3:5)=KPLACA (4:6,1:3);
MATRIZK (9:11,9:11)=KPLACA (4:6,4:6);
```

```
MATRIZK (9:11,15:17)=KPLACA (4:6,7:9);
%FILA 13-14
MATRIZK (13:14,1:2)=KMEMBRANA (5:6,1:2);
MATRIZK (13:14,7:8)=KMEMBRANA (5:6,3:4);
MATRIZK (13:14,13:14)=KMEMBRANA (5:6,5:6);
%FILA 15-17
MATRIZK (15:17,3:5)=KPLACA (7:9,1:3);
MATRIZK (15:17,9:11)=KPLACA (7:9,4:6);
MATRIZK (15:17,15:17)=KPLACA (7:9,7:9);

% MATRIZ DE TRANSFORMACION

MATRIZTRANSF=zeros (18,18);
MATRIZTRANSF (1,1:3)=TablaElementos (i, 5:7);
MATRIZTRANSF (2,1:3)=TablaElementos (i, 8:10);
MATRIZTRANSF (3,1:3)=TablaElementos (i, 11:13);
MATRIZTRANSF (4:6,4:6)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (7:9,7:9)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (10:12,10:12)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (13:15,13:15)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (16:18,16:18)=MATRIZTRANSF (1:3,1:3);

TRANSPUESTA=MATRIZTRANSF';
MATRIZELEMENTOi=TRANSPUESTA*MATRIZK*MATRIZTRANSF;
R=MATRIZK;
v=genvarname (['K', num2str (i)]);
Eval ([v,'=R']);
```

```
%-------------------------------------------------------------------------------------------------
%ENSAMBLE MATRIZ DE RIGIDEZ TOTAL
%-------------------------------------------------------------------------------------------------

    %FILA 1-6
    for m=5:-1:0
        n=6-m;

         MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        5)+MATRIZELEMENTOi(n,1);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        4)+MATRIZELEMENTOi(n,2);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        3)+MATRIZELEMENTOi(n,3);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        2)+MATRIZELEMENTOi(n,4);
         MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        1)+MATRIZELEMENTOi(n,5);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-
        m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
        m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        5)+MATRIZELEMENTOi(n,7);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
        4)+MATRIZELEMENTOi(n,8);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
5)+MATRIZELEMENTOi(n,13);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
4)+MATRIZELEMENTOi(n,14);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
3)+MATRIZELEMENTOi(n,15);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
2)+MATRIZELEMENTOi(n,16);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
1)+MATRIZELEMENTOi(n,17);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
end
```

```
%FILA 7-12
for m=5:-1:0
    n=12-m;
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
5)+MATRIZELEMENTOi(n,1);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
4)+MATRIZELEMENTOi(n,2);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
3)+MATRIZELEMENTOi(n,3);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
2)+MATRIZELEMENTOi(n,4);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
1)+MATRIZELEMENTOi(n,5);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
5)+MATRIZELEMENTOi(n,7);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
4)+MATRIZELEMENTOi(n,8);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
```

```
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
5)+MATRIZELEMENTOi(n,13);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
4)+MATRIZELEMENTOi(n,14);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
3)+MATRIZELEMENTOi(n,15);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
2)+MATRIZELEMENTOi(n,16);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
1)+MATRIZELEMENTOi(n,17);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
end

%FILA 13-18
for m=5:-1:0
    n=18-m;
    MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
    5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
    5)+MATRIZELEMENTOi(n,1);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
4)+MATRIZELEMENTOi(n,2);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
3)+MATRIZELEMENTOi(n,3);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
2)+MATRIZELEMENTOi(n,4);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
1)+MATRIZELEMENTOi(n,5);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
5)+MATRIZELEMENTOi(n,7);
 MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
4)+MATRIZELEMENTOi(n,8);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
```

```
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                5)+MATRIZELEMENTOi(n,13);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                4)+MATRIZELEMENTOi(n,14);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                3)+MATRIZELEMENTOi(n,15);
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                2)+MATRIZELEMENTOi(n,16);
                 MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
                1)+MATRIZELEMENTOi(n,17);
                 MATRIZRIGIDEZ(6*TablaElementos(i,4)-
                m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
                m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
        end
    end
VectorFuerzas=zeros (max (CoordenadasLocales (:,1))*6,1);

for j=1: max (CoordenadasLocales (:,1))
                z=find (CoordenadasLocales (:,1) ==j);
                VectorFuerzas (6*j-3)=sum (CoordenadasLocales (z, 5))*t*p*-1; % Vector que
                contiene las fuerzas por peso propio por área aferente soportadas por cada nodo en
                direccion –Z.
end

PESO=abs (sum (VectorFuerzas)) %Peso total cubierta
```

```
%-------------------------------------------------------------------------------------------------
%NODOS DE LOS APOYOS
%-------------------------------------------------------------------------------------------------
Apoyos=load ('NodosApoyos.txt'); %Organización y enumeración de nodos por coordenadas X, Y, Z.

CoordenadaApoyos=zeros ((length (Apoyos)/3) ,4); %Inicia la matriz que almacenará las coordenadas de los nodos de la cubierta.

for i=1:(length (Apoyos)/3) % For desde 1 hasta la longitud total de la lista de nodos dividida en tres ya que se va a organizar por cada fila tres coordenadas "X, Y, Z".
    CoordenadaApoyos (i, 1)=0; %Enumera los nodos de la cubierta.
    CoordenadaApoyos (i, 2)=Apoyos ((3*i)-2); %Llena el vector de las coordenadas X de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-2 es porque debe tomar el número 1, 4, 7,... de la lista, es decir, el correspondiente a la coordenada X.
    CoordenadaApoyos (i, 3)=Apoyos ((3*i)-1); %Llena el vector de las coordenadas Y de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-1 es porque debe tomar el número 2, 5, 8,... de la lista, es decir, el correspondiente a la coordenada Y.
    CoordenadaApoyos (i, 4)=Apoyos (3*i); %Llena el vector de las coordenadas Z de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i es porque debe tomar el número 3, 6, 9,... de la lista, es decir, el correspondiente a la coordenada Z.

    ind=find(CoordenadasFinales(:,2)==CoordenadaApoyos(i,2)&CoordenadasFinales(:,3)==C
    oordenadaApoyos(i,3)&CoordenadasFinales(:,4)==CoordenadaApoyos(i,4));
    CoordenadaApoyos (i, 1)=CoordenadasFinales (ind (1) ,1);
End

%ELIMINACION ROTACION EN Z

Nnodos=length (MATRIZRIGIDEZ (1, :))/6
```

```
for i=Nnodos:-1:1
   MATRIZRIGIDEZ (6*i, :)= [];
   MATRIZRIGIDEZ (:,6*i)= [];
   VectorFuerzas (6*i, :)= [];
end
NodosAp=sort ((CoordenadaApoyos (:,1)),'descend'); % Se organizan los nodos de los apoyos en orden descendente.
for i=1:(length(Apoyos)/3)
      MATRIZRIGIDEZ(5*(NodosAp(i))-4:5*(NodosAp(i)),:)=[];
      MATRIZRIGIDEZ(:,5*(NodosAp(i))-4:5*(NodosAp(i)))=[];
      VectorFuerzas(5*(NodosAp(i))-4:5*(NodosAp(i)),:)=[];
end
DESP=linsolve(MATRIZRIGIDEZ,VectorFuerzas);
   e1=unique(CoordenadasFinales(:,1));
   e1(NodosAp)=[];

   for i=1:length(e1)
      DESP(5*i-4:5*i,2)=e1(i);
   end

DESPLAZAMIENTOS=DESP(:,1);
ENERGIA=transp(VectorFuerzas)*DESPLAZAMIENTOS

%-----------------------------------------------------------------------------------------------
%FIN PROGRAMACION DEL METODO DE ELEMENTOS FINITOS PARA SHELL
%-----------------------------------------------------------------------------------------------

fo1(k,1)=ENERGIA; %Evalúa la función objetivo 1 con cada partícula.

fo2(k,1)=PESO; %Evalúa la función objetivo 2 con cada partícula.

 end % end de partícula
```

```
%-----------------------------------------------------------------------------------------------
%EVALUACION DE LA FUNCION OBJETIVO TOTAL
%-----------------------------------------------------------------------------------------------

for i=1:numpar
        esc=(min(fo1))/(min(fo2))
        FO(i,1)=w1*fo1(i,1)+w2*fo2(i,1)*esc; %Las funciones objetivo se suman en una sola con
        pesos iguales a 0.5, esto se hace solo en esta primera evaluación de la función objetivo,
        posteriormente los pesos son dinámicos y los encuentra el PSO.
end

%-----------------------------------------------------------------------------------------------
%ACTUALIZAR LA MEJOR POSICIÓN DE CADA PARTÍCULA
%-----------------------------------------------------------------------------------------------

for i=1:numpar
    if FO(i,1)<FOxbest(i,1)
       FOxbest(i,1)=FO(i,1);
       for q=1:Elementos
              xbest(i,q)=xin(i,q); %Si la función objetivo evaluada con la nueva posición es
              menor que la función objetivo evaluada con la mejor posición que ha tenido hasta el
              momento la partícula, se actualiza la mejor posición de la partícula con la nueva
              posición.
       end
    end
end

%-----------------------------------------------------------------------------------------------
%ACTUALIZAR LA MEJOR POSICIÓN DEL ENJAMBRE
%-----------------------------------------------------------------------------------------------

[bestglobtrans,ibestglobtrans]=min(FO); %Obtiene el valor mínimo de la función objetivo y el
índice en donde se localiza.
```

```
if bestglobtrans<bestglob
    bestglob=bestglobtrans;
    for q=1:Elementos
      xbestglob(1,q)=xin(ibestglobtrans,q);
    end
end

%-----------------------------------------------------------------------------------------------------
%ACTUALIZAR LA MEJOR POSICIÓN DEL VECINDARIO
%-----------------------------------------------------------------------------------------------------
 if FO(1,1)<FO(2,1)&&FO(1,1)<FOxbestveci(1,1)
    FOxbestveci(1,1)=FO(1,1);
    for q=1:Elementos
            xbestveci(1,q)=xin(1,q); %Si la función objetivo evaluada en la partícula 1 es
            menor que la función objetivo evaluada en la partícula 2 y menor que la función
            objetivo evaluada en la mejor posición que hasta el momento ha tenido el
            vecindario, la mejor posición del vecindario de la partícula 1 es xin(1,1).
    end
else
     if FO(2,1)<FOxbestveci(1,1)
      FOxbestveci(1,1)=FO(2,1);
      for q=1:Elementos
              xbestveci(1,q)=xin(2,q); %Si la función objetivo evaluada en la partícula 1
              es mayor que la función objetivo evaluada en la partícula 2 y la función
              objetivo evaluada en la partícula 2 es menor que la función objetivo
              evaluada en la mejor posición que hasta ahora ha tenido el vecindario, la
              mejor posición del vecindario de la partícula 1 es xin(2,1).
      end
     end
  end
```

```
for i=2:(numpar-1)
    if FO(i,1)<FO(i-1,1)&&FO(i,1)<FO(i+1,1)&&FO(i,1)<FOxbestveci(i,1)
       FOxbestveci(i,1)=FO(i,1);
       for q=1:Elementos
                xbestveci(i,q)=xin(i,q); %Si la función objetivo evaluada en la partícula i es menor
                que la función objetivo evaluada en la partícula i-1 y en la partícula i+1 y menor
                que la función objetivo evaluada en la mejor posición que ha tenido hasta el
                momento el vecindario, la mejor posición del vecindario es la de la partícula i.
       end
else
        if FO(i-1,1)<FO(i+1,1)&&FO(i-1,1)<FOxbestveci(i,1)
          FOxbestveci(i,1)=FO(i-1,1);
          for q=1:Elementos
                xbestveci(i,q)=xin(i-1,q); %Si la mejor posición del vecindario NO es la de la
                partícula i, puede ser entonces la de la partícula i-1 o la de la partícula i+1 o la que
                ya tenía. Luego, si la función objetivo evaluada en la partícula i-1 es menor que la
                función objetivo evaluada en la partícula i+1 y menor que la función objetivo
                evaluada en la mejor posición que ha tenido el vecindario hasta el momento, la
                mejor posición del vecindario es la de la partícula i-1.
          end
       else
          if FO(i+1,1)<FOxbestveci(i,1)
             FOxbestveci(i,1)=FO(i+1,1);
             for q=1:Elementos
                      xbestveci(i,q)=xin(i+1,q); %Si la mejor partícula del vecindario no es ni la
                      i, ni la i-1, y la función objetivo evaluada en la posición i+1 es menor que la
                      función objetivo evaluada en la mejor posición que ha tenido hasta ahora el
                      vecindario, la mejor posición del vecindario será la i+1.
             end
          end
       end
    end
 end
```

```
    if FO(numpar,1)<FO(numpar-1,1)&&FO(numpar,1)<FOxbestveci(numpar,1)
        FOxbestveci(numpar,1)=FO(numpar,1);
        for q=1:Elementos
                xbestveci(numpar,q)=xin(numpar,q); %Si la función objetivo evaluada en la
                partícula numpar es menor que la función objetivo evaluada en la partícula numpar-
                1 y menor que la función objetivo evaluada en la mejor posición que hasta ahora ha
                tenido el vecindario, la mejor posición del vecindario de la partícula numpar es
                xin(numpar,1).
        end
    else
        if FO(numpar-1,1)<FOxbestveci(numpar,1)
            FOxbestveci(numpar,1)=FO(numpar-1,1);
            for q=1:Elementos
                xbestveci(numpar,q)=xin(numpar-1,q); %Si la función objetivo evaluada en la
                partícula numpar es mayor que la función objetivo evaluada en la partícula numpar-
                1 y es menor que la función objetivo evaluada en la mejor posición que ha tenido
                hasta ahora el vecindario, la mejor posición del vecindario de la partícula numpar es
                xin(numpar-1,1).
            end
        end
    end

d=find(xbest(:,Elementos)==xbestglob(Elementos)&xbest(:,(round(Elementos*(3/4))))==xbestglob
((round(Elementos*(3/4))))&xbest(:,(round(Elementos*(2/4))))==xbestglob((round(Elementos*(2/4
))))&xbest(:,(round(Elementos*(1/4))))==xbestglob((round(Elementos*(1/4)))));

        TABLAOPTIMIZACION (j, 1)=j;
        TABLAOPTIMIZACION (j, 2)=fo1 (d);
        TABLAOPTIMIZACION (j, 3)=fo2 (d);

end
```

```
%Evaluación de las dos funciones objetivo con las mejores posiciones que se encontraron para cada
partícula

fo1best=zeros(numpar,1);
fo2best=zeros(numpar,1);

%-------------------------------------------------------------------------------------------------
%PROGRAMACION DEL METODO DE ELEMENTOS FINITOS PARA SHELL
%-------------------------------------------------------------------------------------------------

        MATRIZRIGIDEZ=zeros(max(CoordenadasLocales(:,1))*6,max(CoordenadasLocales(:,1))
        *6);
    for i=1:length(TablaElementos(:,1))
        A=[CoordenadasLocales(3*i-2,2),CoordenadasLocales(3*i-2,3),CoordenadasLocales(3*i-
        2,4)];
        B=[CoordenadasLocales(3*i-1,2),CoordenadasLocales(3*i-1,3),CoordenadasLocales(3*i-
        1,4)];
        C=[CoordenadasLocales(3*i,2),CoordenadasLocales(3*i,3),CoordenadasLocales(3*i,4)];
        AB=B-A;
        AC=C-A;
        r=cross(AB,AC);
        AR=norm(r)/2;
        TablaElementos(i,14)=AR;% Asigno el área para cada elemento de la superficie
        CoordenadasLocales((3*i-2):3*i,5)=TablaElementos(i,14)/3;
        CoordenadasLocales((3*i-2):3*i,6)=CoordenadasLocales((3*i-2):3*i,5)*xbest(k,i);
        E=2.15381e10;%N/m2
        v=0.2;
        p=23544 ; % Peso específico del concreto N/m3
        a=(E*(xbest(k,i)))/(4*AR*(1-v^2));
        b=(E*(xbest(k,i)))/(8*AR*(1+v));
        c=(E*(xbest(k,i)^3))/(12*(1-v^2));
```

```
% MATRIZ DE RIGIDEZ PLACA
MATRIZPLACA=zeros (9,9);

%FILA 4
MATRIZPLACA (4,4)=2*c*C (1)*B (2);
MATRIZPLACA (4,6)=2*c*v*C (1)*B (2);
MATRIZPLACA (4,7)=2*c*(C (1)) ^2*B (2);
MATRIZPLACA(4,8)=(2/3)*c*v*(C(1))^2*B(2)+(2/3)*c*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA (4,9)=2*c*v*C (1)*B (2)*(B (2)+C (2));
%FILA 5
MATRIZPLACA (5,5)= (1-v)*c*C (1)*B (2);
MATRIZPLACA(5,8)=(2/3)*(1-v)*c*(C(1))^2*B(2)+(2/3)*(1-
v)*c*C(1)*B(2)*(B(2)+C(2));
%FILA 6
MATRIZPLACA (6,4)=2*c*v*C (1)*B (2);
MATRIZPLACA (6,6)=2*c*C (1)*B (2);
MATRIZPLACA (6,7)=2*c*v*(C (1)) ^2*B (2);
MATRIZPLACA(6,8)=(2/3)*c*(C(1))^2*B(2)+(2/3)*c*v*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA (6,9)=2*c*C (1)*B (2)*(B (2)+C (2));
%FILA 7
MATRIZPLACA (7,4)=2*c*(C (1)) ^2*B (2);
MATRIZPLACA (7,6)=2*c*v*(C (1)) ^2*B (2);
MATRIZPLACA (7,7)=3*c*(C (1)) ^3*B (2);
   MATRIZPLACA(7,8)=c*v*(C(1))^3*B(2)+(1/2)*c*(C(1))^2*B(2)*(B(2)+(2*C(2)));
MATRIZPLACA (7,9)= (3/2)*c*v*(C (1)) ^2*B (2)*(B (2)+2*C (2));
%FILA 8
MATRIZPLACA(8,4)=(2/3)*c*v*(C(1))^2*B(2)+(2/3)*c*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA(8,5)=(2/3)*(1-v)*c*(C(1))^2*B(2)+(2/3)*(1-
v)*c*C(1)*B(2)*(B(2)+C(2));
MATRIZPLACA(8,6)=(2/3)*c*(C(1))^2*B(2)+(2/3)*c*v*C(1)*B(2)*(B(2)+C(2));
   MATRIZPLACA(8,7)=c*v*(C(1))^3*B(2)+(1/2)*c*(C(1))^2*B(2)*(B(2)+(2*C(2)));
MATRIZPLACA(8,8)=c*((128*v)*((((C(1))^3*B(2))/12)+(1/12)*(C(1))^2*B(2)*(B(2)+2*
(C(2)))+(1/12)*C(1)*B(2)*((B(2))^2+B(2)*C(2)+(C(2))^2))-(8*(1-
```

v))*((1/24)*(C(1))^2*B(2)*(B(2)+2*(C(2)))));

MATRIZPLACA(8,9)=(c/2)*(C(1))^2*B(2)*(B(2)+2*C(2))+c*v*C(1)*B(2)*((B(2))^2+B(2)*C(2)+(C(2))^2);

%FILA 9

MATRIZPLACA (9,4)=2*c*v*C (1)*B (2)*(B (2)+C (2));

MATRIZPLACA (9,6)=2*c*C (1)*B (2)*(B (2)+C (2));

MATRIZPLACA (9,7)= (3/2)*c*v*(C (1)) ^2*B (2)*(B (2)+2*C (2));

MATRIZPLACA(9,8)=(c/2)*(C(1))^2*B(2)*(B(2)+2*C(2))+c*v*C(1)*B(2)*((B(2))^2+B(2)*C(2)+(C(2))^2);

MATRIZPLACA (9,9)=3*c*C (1)*B (2)*((B (2)) ^2+B (2)*C (2)+C (2) ^2);

MATRIZN=zeros (9,9);

%FILA1

MATRIZN (1,1)=1;

%FILA2

MATRIZN (2,3)=1;

%FILA3

MATRIZN (3,2)=-1;

%FILA4

MATRIZN (4,1)=1;

MATRIZN (4,3)=B (2);

MATRIZN (4,6)= (B (2)) ^2;

MATRIZN (4,9)= (B (2)) ^3;

%FILA5

MATRIZN (5,3)=1;

MATRIZN (5,6)=2*B (2);

MATRIZN (5,9)=3*(B (2)) ^2;

%FILA6

MATRIZN (6,2)=-1;

MATRIZN (6,5)=-B (2);

MATRIZN (6,8)=-(B (2)) ^2;

%FILA7

```
MATRIZN (7,1)=1;
MATRIZN (7,2)=C (1);
MATRIZN (7,3)=C (2);
MATRIZN (7,4)=(C (1)) ^2;
MATRIZN (7,5)=C (1)*C (2);
MATRIZN (7,6)=(C (2)) ^2;
MATRIZN (7,7)=(C (1)) ^3;
MATRIZN (7,8)=(C (1)) ^2*C (2)+C (1)*(C (2)) ^2;
MATRIZN (7,9)=(C (2)) ^3;
%FILA8
MATRIZN (8,3)=1;
MATRIZN (8,5)=C (1);
MATRIZN (8,6)=2*C (2);
MATRIZN (8,8)=2*C (1)*C (2)+(C (1)) ^2;
MATRIZN (8,9)=3*(C (2)) ^2;
%FILA9
MATRIZN (9,2)=-1;
MATRIZN (9,4)=-2*C (1);
MATRIZN (9,5)=-C (2);
MATRIZN (9,7)=-3*(C (1)) ^2;
MATRIZN (9,8)=-((C (2)) ^2+2*C (1)*C (2));
MATRIZNINV=inv (MATRIZN);
MATRIZNTRANSP=transp (MATRIZNINV);
KPLACA=MATRIZNTRANSP*MATRIZPLACA*MATRIZNINV;

% MATRIZ DE RIGIDEZ MEMBRANA
KMEMBRANA=zeros (6,6);
%FILA 1
KMEMBRANA (1,1)=a*(B (2)-C (2)) ^2+b*(B (1)-C (1)) ^2;
KMEMBRANA (1,2)=-a*v*(B (2)-C (2))*(B (1)-C (1))-b*(B (1)-C (1))*(B (2)-C (2));
KMEMBRANA (1,3)=-a*(B (2)-C (2))*(A (2)-C (2))-b*(B (1)-C (1))*(A (1)-C (1));
KMEMBRANA (1,4)=a*v*(B (2)-C (2))*(A (1)-C (1))+b*(B (1)-C (1))*(A (2)-C (2));
KMEMBRANA (1,5)=a*(B (2)-C (2))*(A (2)-B (2))+b*(B (1)-C (1))*(A (1)-B (          1));
```

KMEMBRANA (1,6)=-a*v*(B (2)-C (2))*(A (1)-B (1))-b*(B (1)-C (1))*(A (2)-B (2));

%FILA 2

KMEMBRANA (2,1)=-a*v*(B (2)-C (2))*(B (1)-C (1))-b*(B (1)-C (1))*(B (2)-C (2));

KMEMBRANA (2,2)=b*(B (2)-C (2)) ^2+a*(B (1)-C (1)) ^2;

KMEMBRANA (2,3)=a*v*(B (1)-C (1))*(A (2)-C (2))+b*(B (2)-C (2))*(A (1)-C (1));

KMEMBRANA (2,4)=-a*(B (1)-C (1))*(A (1)-C (1))-b*(B (2)-C (2))*(A (2)-C (2));

KMEMBRANA (2,5)=-a*v*(B (1)-C (1))*(A (2)-B (2))-b*(B (2)-C (2))*(A (1)-B (1));

KMEMBRANA (2,6)=a*(B (1)-C (1))*(A (1)-B (1))+b*(B (2)-C (2))*(A (2)-B (2));

%FILA 3

KMEMBRANA (3,1)=-a*(B (2)-C (2))*(A (2)-C (2))-b*(B (1)-C (1))*(A (1)-C (1));

KMEMBRANA (3,2)=a*v*(B (1)-C (1))*(A (2)-C (2))+b*(B (2)-C (2))*(A (1)-C (1));

KMEMBRANA (3,3)=a*(A (2)-C (2)) ^2+b*(A (1)-C (1)) ^2;

KMEMBRANA (3,4)=-a*v*(A (2)-C (2))*(A (1)-C (1))-b*(A (1)-C (1))*(A (2)-C (2));

KMEMBRANA (3,5)=-a*(A (2)-C (2))*(A (2)-B (2))-b*(A (1)-C (1))*(A (1)-B (1));

KMEMBRANA (3,6)=a*v*(A (2)-C (2))*(A (1)-B (1))+b*(A (1)-C (1))*(A (2)-B (2));

%FILA 4

KMEMBRANA (4,1)=a*v*(B (2)-C (2))*(A (1)-C (1))+b*(B (1)-C (1))*(A (2)-C (2));

KMEMBRANA (4,2)=-a*(B (1)-C (1))*(A (1)-C (1))-b*(B (2)-C (2))*(A (2)-C (2));

KMEMBRANA (4,3)=-a*v*(A (2)-C (2))*(A (1)-C (1))-b*(A (1)-C (1))*(A (2)-C (2));

KMEMBRANA (4,4)=b*(A (2)-C (2)) ^2+a*(A (1)-C (1)) ^2;

KMEMBRANA (4,5)=a*v*(A (1)-C (1))*(A (2)-B (2))+b*(A (2)-C (2))*(A (1)-B (1));

KMEMBRANA (4,6)=-a*(A (1)-C (1))*(A (1)-B (1))-b*(A (2)-C (2))*(A (2)-B (2));

%FILA 5

KMEMBRANA (5,1)=a*(B (2)-C (2))*(A (2)-B (2))+b*(B (1)-C (1))*(A (1)-B (1));

KMEMBRANA (5,2)=-a*v*(B (1)-C (1))*(A (2)-B (2))-b*(B (2)-C (2))*(A (1)-B (1));

KMEMBRANA (5,3)=-a*(A (2)-C (2))*(A (2)-B (2))-b*(A (1)-C (1))*(A (1)-B (1));

KMEMBRANA (5,4)=a*v*(A (1)-C (1))*(A (2)-B (2))+b*(A (2)-C (2))*(A (1)-B (1));

KMEMBRANA (5,5)=a*(A (2)-B (2)) ^2+b*(A (1)-B (1)) ^2;

KMEMBRANA (5,6)=-a*v*(A (2)-B (2))*(A (1)-B (1))-b*(A (1)-B (1))*(A (2)-B (2));

%FILA 6

KMEMBRANA (6,1)=-a*v*(B (2)-C (2))*(A (1)-B (1))-b*(B (1)-C (1))*(A (2)-B (2));

KMEMBRANA (6,2)=a*(B (1)-C (1))*(A (1)-B (1))+b*(B (2)-C (2))*(A (2)-B (2));

KMEMBRANA (6,3)=a*v*(A (2)-C (2))*(A (1)-B (1))+b*(A (1)-C (1))*(A (2)-B (2));

```
KMEMBRANA (6,4)=-a*(A (1)-C (1))*(A (1)-B (1))-b*(A (2)-C (2))*(A (2)-B (2));
KMEMBRANA (6,5)=-a*v*(A (2)-B (2))*(A (1)-B (1))-b*(A (1)-B (1))*(A (2)-B (2));
KMEMBRANA (6,6)=b*(A (2)-B (2)) ^2+a*(A (1)-B (1)) ^2;

% MATRIZ RIGIDEZ TOTAL ELEMENTO
MATRIZK=zeros (18,18);

%FILA 1-2
MATRIZK (1:2,1:2)=KMEMBRANA (1:2,1:2);
MATRIZK (1:2,7:8)=KMEMBRANA (1:2,3:4);
MATRIZK (1:2,13:14)=KMEMBRANA (1:2,5:6);
%FILA 3-5
MATRIZK (3:5,3:5)=KPLACA (1:3,1:3);
MATRIZK (3:5,9:11)=KPLACA (1:3,4:6);
MATRIZK (3:5,15:17)=KPLACA (1:3,7:9);
%FILA 7-8
MATRIZK (7:8,1:2)=KMEMBRANA (3:4,1:2);
MATRIZK (7:8,7:8)=KMEMBRANA (3:4,3:4);
MATRIZK (7:8,13:14)=KMEMBRANA (3:4,5:6);
%FILA 9-11
MATRIZK (9:11,3:5)=KPLACA (4:6,1:3);
MATRIZK (9:11,9:11)=KPLACA (4:6,4:6);
MATRIZK (9:11,15:17)=KPLACA (4:6,7:9);
%FILA 13-14
MATRIZK (13:14,1:2)=KMEMBRANA (5:6,1:2);
MATRIZK (13:14,7:8)=KMEMBRANA (5:6,3:4);
MATRIZK (13:14,13:14)=KMEMBRANA (5:6,5:6);
%FILA 15-17
MATRIZK (15:17,3:5)=KPLACA (7:9,1:3);
MATRIZK (15:17,9:11)=KPLACA (7:9,4:6);
MATRIZK (15:17,15:17)=KPLACA (7:9,7:9);
```

```
% MATRIZ DE TRANSFORMACION
MATRIZTRANSF=zeros (18,18);
MATRIZTRANSF (1,1:3)=TablaElementos (i, 5:7);
MATRIZTRANSF (2,1:3)=TablaElementos (i, 8:10);
MATRIZTRANSF (3,1:3)=TablaElementos (i, 11:13);
MATRIZTRANSF (4:6,4:6)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (7:9,7:9)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (10:12,10:12)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (13:15,13:15)=MATRIZTRANSF (1:3,1:3);
MATRIZTRANSF (16:18,16:18)=MATRIZTRANSF (1:3,1:3);

TRANSPUESTA=MATRIZTRANSF';
MATRIZELEMENTOi=TRANSPUESTA*MATRIZK*MATRIZTRANSF;
R=MATRIZK;
v=genvarname (['K', num2str (i)]);
Eval ([v,'=R']);

%-----------------------------------------------------------------------------------------------
%ENSAMBLE MATRIZ DE RIGIDEZ TOTAL
%-----------------------------------------------------------------------------------------------

%FILA 1-6
for m=5:-1:0
        n=6-m;

         MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        5)+MATRIZELEMENTOi(n,1);
        MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
        4)+MATRIZELEMENTOi(n,2);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
3)+MATRIZELEMENTOi(n,3);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
2)+MATRIZELEMENTOi(n,4);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,2)-
1)+MATRIZELEMENTOi(n,5);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
5)+MATRIZELEMENTOi(n,7);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
4)+MATRIZELEMENTOi(n,8);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
5)+MATRIZELEMENTOi(n,13);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
4)+MATRIZELEMENTOi(n,14);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
3)+MATRIZELEMENTOi(n,15);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
2)+MATRIZELEMENTOi(n,16);
 MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,2)-m,6*TablaElementos(i,4)-
1)+MATRIZELEMENTOi(n,17);
MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,2)-
m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
end

%FILA 7-12
 for m=5:-1:0
      n=12-m;
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
5)+MATRIZELEMENTOi(n,1);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
4)+MATRIZELEMENTOi(n,2);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
3)+MATRIZELEMENTOi(n,3);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
2)+MATRIZELEMENTOi(n,4);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,2)-
1)+MATRIZELEMENTOi(n,5);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
5)+MATRIZELEMENTOi(n,7);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
4)+MATRIZELEMENTOi(n,8);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
 MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
5)+MATRIZELEMENTOi(n,13);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
4)+MATRIZELEMENTOi(n,14);
MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
3)+MATRIZELEMENTOi(n,15);
```

```
        MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        2)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        2)+MATRIZELEMENTOi(n,16);
        MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        1)=MATRIZRIGIDEZ(6*TablaElementos(i,3)-m,6*TablaElementos(i,4)-
        1)+MATRIZELEMENTOi(n,17);
        MATRIZRIGIDEZ(6*TablaElementos(i,3)-
        m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,3)-
        m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
end

%FILA 13-18
for m=5:-1:0
        n=18-m;
        MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        5)+MATRIZELEMENTOi(n,1);
        MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        4)+MATRIZELEMENTOi(n,2);
        MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        3)+MATRIZELEMENTOi(n,3);
        MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        2)+MATRIZELEMENTOi(n,4);
        MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,2)-
        1)+MATRIZELEMENTOi(n,5);
        MATRIZRIGIDEZ(6*TablaElementos(i,4)-
        m,6*TablaElementos(i,2))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
        m,6*TablaElementos(i,2))+MATRIZELEMENTOi(n,6);
```

```
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
5)+MATRIZELEMENTOi(n,7);
 MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
4)+MATRIZELEMENTOi(n,8);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
3)+MATRIZELEMENTOi(n,9);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
2)+MATRIZELEMENTOi(n,10);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,3)-
1)+MATRIZELEMENTOi(n,11);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,3))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
m,6*TablaElementos(i,3))+MATRIZELEMENTOi(n,12);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
5)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
5)+MATRIZELEMENTOi(n,13);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
4)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
4)+MATRIZELEMENTOi(n,14);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
3)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
3)+MATRIZELEMENTOi(n,15);
MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
2)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
2)+MATRIZELEMENTOi(n,16);
 MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
1)=MATRIZRIGIDEZ(6*TablaElementos(i,4)-m,6*TablaElementos(i,4)-
1)+MATRIZELEMENTOi(n,17);
```

```
                MATRIZRIGIDEZ(6*TablaElementos(i,4)-
                m,6*TablaElementos(i,4))=MATRIZRIGIDEZ(6*TablaElementos(i,4)-
                m,6*TablaElementos(i,4))+MATRIZELEMENTOi(n,18);
        end
    end
VectorFuerzas=zeros (max (CoordenadasLocales (:,1))*6,1);

for j=1: max (CoordenadasLocales (:,1))
                z=find (CoordenadasLocales (:,1) ==j);
                VectorFuerzas (6*j-3)=sum (CoordenadasLocales (z, 5))*t*p*-1; % Vector que
                contiene las fuerzas por peso propio por área aferente soportadas por cada nodo en
                direccion –Z.
end

PESO=abs (sum (VectorFuerzas)) %Peso total cubierta

%----------------------------------------------------------------------------------------------------
%NODOS DE LOS APOYOS
%----------------------------------------------------------------------------------------------------

Apoyos=load ('NodosApoyos.txt'); %Organización y enumeración de nodos por coordenadas X, Y,
Z.

CoordenadaApoyos=zeros ((length (Apoyos)/3) ,4); %Inicia la matriz que almacenará las
coordenadas de los nodos de la cubierta.

for i=1:(length (Apoyos)/3) % For desde 1 hasta la longitud total de la lista de nodos dividida en
tres ya que se va a organizar por cada fila tres coordenadas "X, Y, Z".
        CoordenadaApoyos (i, 1)=0; %Enumera los nodos de la cubierta.
        CoordenadaApoyos (i, 2)=Apoyos ((3*i)-2); %Llena el vector de las coordenadas X de los
        nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-2
        es porque debe tomar el número 1, 4, 7,... de la lista, es decir, el correspondiente a la
        coordenada X.
```

```
    CoordenadaApoyos (i, 3)=Apoyos ((3*i)-1); %Llena el vector de las coordenadas Y de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i-1 es porque debe tomar el número 2, 5, 8,... de la lista, es decir, el correspondiente a la coordenada Y.
    CoordenadaApoyos (i, 4)=Apoyos (3*i); %Llena el vector de las coordenadas Z de los nodos de la cubierta, con los datos de la lista del archivo "CoordenadasNodos.txt". El 3i es porque debe tomar el número 3, 6, 9,... de la lista, es decir, el correspondiente a la coordenada Z.

    ind=find(CoordenadasFinales(:,2)==CoordenadaApoyos(i,2)&CoordenadasFinales(:,3)==C
    oordenadaApoyos(i,3)&CoordenadasFinales(:,4)==CoordenadaApoyos(i,4));
    CoordenadaApoyos (i, 1)=CoordenadasFinales (ind (1) ,1);
End

%ELIMINACION ROTACION EN Z

Nnodos=length (MATRIZRIGIDEZ (1, :))/6

for i=Nnodos:-1:1
   MATRIZRIGIDEZ (6*i, :)= [];
   MATRIZRIGIDEZ (:,6*i)= [];
   VectorFuerzas (6*i, :)= [];
end
NodosAp=sort ((CoordenadaApoyos (:,1)),'descend'); % Se organizan los nodos de los apoyos en orden descendente.
for i=1:(length(Apoyos)/3)
      MATRIZRIGIDEZ(5*(NodosAp(i))-4:5*(NodosAp(i)),:)=[];
      MATRIZRIGIDEZ(:,5*(NodosAp(i))-4:5*(NodosAp(i)))=[];
      VectorFuerzas(5*(NodosAp(i))-4:5*(NodosAp(i)),:)=[];
end
DESP=linsolve(MATRIZRIGIDEZ,VectorFuerzas);
   e1=unique(CoordenadasFinales(:,1));
   e1(NodosAp)=[];
```

```
   for i=1:length(e1)
      DESP(5*i-4:5*i,2)=e1(i);
   end

DESPLAZAMIENTOS=DESP(:,1);
ENERGIA=transp(VectorFuerzas)*DESPLAZAMIENTOS

%-------------------------------------------------------------------------------------------------
%FIN PROGRAMACION DEL METODO DE ELEMENTOS FINITOS PARA SHELL
%-------------------------------------------------------------------------------------------------

fo1(k,1)=ENERGIA; %Evalúa la función objetivo 1 con cada partícula.

fo2(k,1)=PESO; %Evalúa la función objetivo 2 con cada partícula.

 end % end de partícula

xbestglob=transp(xbestglob);
dlmwrite('xbestglob.txt',xbestglob);

d=find(xbest(:,Elementos)==xbestglob(Elementos)&xbest(:,(round(Elementos*(3/4))))==xbestglob
((round(Elementos*(3/4))))&xbest(:,(round(Elementos*(2/4))))==xbestglob((round(Elementos*(2/4
))))&xbest(:,(round(Elementos*(1/4))))==xbestglob((round(Elementos*(1/4)))));

ENERGIAfinal=fo1best (d)
PESOfinal=fo2best (d)
```

ANEXO D: CÓDIGO DE PROGRAMACION EN GRASSHOPER