UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

FACULTAD DE INGENIERÍA

PROGRAMA ACADÉMICO DE INGENIERÍA CIVIL

Modelo computacional para el análisis matricial de estructuras reticulares

TESIS

Para optar el título profesional de Ingeniero Civil

AUTORES

Díaz Barrantes, Frank Diego (0000-0001-8181-6383)

Guillén Hernández, Álvaro Ricardo (0000-0003-1935-6835)

ASESOR

Ramírez Chaupis, Joseph Edher (0000-0001-8930-3224)

Lima, 06 de enero del 2020

DEDICATORIA

Esta tesis va dedicada a nuestras familias, a los profesionales que nos introdujeron al mundo de la programación y a todas las personas que desinteresadamente nos brindaron su apoyo para la realización de este trabajo.

RESUMEN

Esta tesis desarrolla las teorías y programación de un modelo computacional diseñado para el análisis estructural, mediante el método matricial, de estructuras reticulares. El modelo fue realizado con la finalidad de ser implementado en la universidad como material de apoyo para los estudiantes de ingeniería civil y con la visión de que futuros tesisas sigan desarrollando el programa hasta obtener un software comercial desarrollado en esta casa de estudios.

Este modelo computacional fue programado con la ayuda del IDE (Integrated Development Environment) Matlab, el cual usa su propio lenguaje de programación (lenguaje M). En esta tesis se muestran resultados de cálculo, gráficos además de flujogramas y códigos de la programación para armaduras planas y pórticos planos.

Palabras clave: estructuras reticulares, análisis matricial, Matlab, modelo computacional.

Computational model for matrix analysis of reticular structures

ABSTRACT

This thesis develops theories and programming of a computational model designed for the structural analysis by the matrix method of reticular structures. The model was designed to be implemented at university as a support material for students of civil engineering and with the vision that future testists continue developing the program until obtaining a commercial software developed in this house of studies.

This computational model was programmed with the help of the IDE (Integrated Development Environment) Matlab, which uses its own programming language (M language). This thesis shows calculation results, graphs in addition to flowcharts and programming codes for truss and frames.

Key words: reticular structures; matrix analysis; Matlab; computational model.

TABLA DE CONTENIDOS

1	GENERALIDADES	1
1.1	INTRODUCCIÓN.....	1
1.2	ANTECEDENTES	2
1.3	FUNDAMENTACIÓN.....	3
1.4	OBJETIVOS DE LA INVESTIGACIÓN	3
1.4.1	Objetivo general	3
1.4.2	Objetivos específicos.....	3
1.5	DELIMITACIÓN.....	4
2	MARCO TEÓRICO.....	5
2.1	CONCEPTOS FUNDAMENTALES DE ANÁLISIS ESTRUCTURAL.....	5
2.2	ELEMENTOS ESTRUCTURALES RETICULARES	8
2.3	ANÁLISIS MATRICIAL DE ESTRUCTURAS POR EL MÉTODO DE RIGIDEZ	11
2.4	MATRIZ DE RIGIDEZ DEL ELEMENTO	14
2.4.1	Matriz de rigidez del elemento en coordenadas locales en armaduras.....	14
2.4.2	Matriz de rigidez del elemento en coordenadas locales en pórticos planos	17
2.5	MATRIZ DE TRANSFORMACIÓN DE FUERZA Y DESPLAZAMIENTO.....	18
2.5.1	Matriz de transformación de fuerza y desplazamiento para armaduras	18
2.5.2	Matriz de transformación de fuerza y desplazamiento para pórticos planos...	23
2.6	MATRIZ DE RIGIDEZ GLOBAL DEL ELEMENTO	25
2.7	MATRIZ DE RIGIDEZ GLOBAL DE LA ESTRUCTURA.....	27
3	DESARROLLO DE LA PROGRAMACIÓN.....	32
3.1	LENGUAJE DE PROGRAMACIÓN EMPLEADO	32
3.2	DESARROLLO DEL MODELO MATEMÁTICO	34
3.2.1	Flujogramas	35
3.2.1.1	Flujograma del módulo de inputs de TrussSolver UPC	38
3.2.1.2	Flujograma del módulo de análisis de TrussSolver UPC	44
3.2.1.3	Flujograma del módulo de outputs de TrussSolver UPC	50
3.2.1.4	Flujograma del módulo de inputs de PlaneSolver UPC	56
3.2.1.5	Flujograma del módulo de análisis de PlaneSolver UPC	61
3.2.1.6	Flujograma del módulo de outputs PlaneSolver UPC	72

3.3	CARACTERÍSTICAS DEL PROGRAMA	78
3.4	IMPLEMENTACIÓN DE TRUSSSOLVER Y PLANESOLVER	79
3.4.1	Código de programación	80
3.4.2	Análisis de costo computacional	80
4	VERIFICACIÓN DE RESULTADOS	85
4.1	APLICACIÓN DEL MODELO	85
4.1.1	Ejemplo 1.....	85
4.1.1.1	Ingreso de datos del ejemplo 1	85
4.1.1.2	Resultados obtenidos del ejemplo 1	89
4.1.1.3	Comparación de resultados del TrussSolver vs. Kassimali del ejemplo 1.....	89
4.1.1.4	Salida gráfica	91
4.1.1.5	Observación de la comparativa del ejemplo 1	92
4.1.2	Ejemplo 2.....	93
4.1.2.1	Ingreso de datos del ejemplo 2	93
4.1.2.2	Resultados obtenidos del ejemplo 2	96
4.1.2.3	Comparación de resultados del TrussSolver vs. Kassimali del ejemplo 2.....	97
4.1.2.4	Observación de la comparativa del ejemplo 2.....	99
4.1.2.5	Salida gráfica del ejemplo 2	100
4.1.3	Ejemplo 3.....	101
4.1.3.1	Ingreso de datos del ejemplo 3	101
4.1.3.2	Resultados obtenidos del ejemplo 3	106
4.1.3.3	Comparación de resultados del TrussSolver vs. Kassimali del ejemplo 3.....	106
4.1.3.4	Observación de la comparativa del ejemplo 3.....	108
4.1.3.5	Salida gráfica del ejemplo 5	108
5	CONCLUSIONES Y RECOMENDACIONES	109
5.1	CONCLUSIONES.....	109
5.2	RECOMENDACIONES	109
6	REFERENCIAS.....	111

7	ANEXOS.....	113
7.1	ANEXO 1: CÓDIGO DE PROGRAMACIÓN – TRUSSSOLVER.....	113
7.2	ANEXO 2: CÓDIGO DE PROGRAMACIÓN – PLANE Solver	125

ÍNDICE DE TABLAS

Tabla 1 Fuerzas de empotramiento perfecto.....	28
Tabla 2 Operadores básicos de MATLAB.	32
Tabla 3 Lista de comandos.	33
Tabla 4 Comparativa de costo computacional.....	83
Tabla 5 Comparación de desplazamientos del ejemplo 1.....	89
Tabla 6 Comparación de desplazamientos del ejemplo 1.....	90
Tabla 7 Comparación de fuerzas internas en los elementos del ejemplo 1.	90
Tabla 8 Comparación de desplazamientos del ejemplo 2.....	97
Tabla 9 Comparación de las fuerzas en los apoyos del ejemplo 2.	98
Tabla 10 Comparación de las fuerzas internas en los elementos del ejemplo 2.....	98
Tabla 11 Comparación de desplazamientos del ejemplo 3.....	106
Tabla 12 Comparación de las fuerzas en los apoyos del ejemplo 3.	107
Tabla 12 Comparación de las fuerzas internas en los elementos del ejemplo 3.....	107

ÍNDICE DE FIGURAS

Figura 1. Barra de equilibrio.....	5
Figura 2. Forma deformada de un pórtico.	6
Figura 3. Tipos de estructuras reticulares; viga (a), armadura plana (b), pórtico plano (c), parrilla (d), armadura espacial (e), pórtico espacial (f).	9
Figura 4. Identificación de elementos en una armadura.	13
Figura 5. Elemento orientado en coordenadas locales.	16
Figura 6. Elemento de un pórtico plano con sus fuerzas actuantes	17
Figura 7. Ángulos θ_x e θ_y	18
Figura 8. Elemento ejemplo NF.	19
Figura 9. Desplazamientos del elemento..	21
Figura 10. Elemento bajo fuerza.....	23
Figura 11. Elemento pórtico plano con desplazamientos locales.	24
Figura 12. Etiquetado de nodos, miembros y grados de libertad.	36
Figura 13. Interfaz de resultados gráficos.	78
Figura 14. Matriz de grados de libertad libres obtenidos con el programa.	79
Figura 15. Interfaz del programa	80
Figura 16. Ejemplo 1	85
Figura 16 Enumeración de nodos del ejemplo 1	85
Figura 18 Ingreso de coordenadas x e y de los nodos del ejemplo 1	86
Figura 19 Ingreso de restricciones en x e y de los nodos del ejemplo 1	86
Figura 20 Ingreso del material y módulo de elasticidad del ejemplo 1.....	87
Figura 21 Ingreso de áreas transversales de cada uno de los elementos del ejemplo 1.....	87
Figura 22 Ingreso de conectividad de los elementos del ejemplo 1	88
Figura 23 Ingreso cargas nodales del ejemplo 1.....	88
Figura 24 Resultados obtenidos para el ejemplo 1 con TrussSolver.	89
Figura 25 Salida gráfica TrussSolver para el ejemplo 1.....	91
Figura 26 Ejemplo 2.	93
Figura 27 Enumeración de nodos del ejemplo 2.	93
Figura 28 Ingreso de coordenadas x e y de los nodos del ejemplo 2.	93
Figura 29 Ingreso de restricciones en x e y de los nodos del ejemplo 2.	94
Figura 30 Ingreso del material y módulo de elasticidad del ejemplo 2.....	94
Figura 31 Ingreso de áreas transversales de cada uno de los elementos del ejemplo 2.....	95

Figura 32 Ingreso de conectividad de los elementos del ejemplo 2.	95
Figura 33 Ingreso cargas nodales del ejemplo 2.....	96
Figura 34 Resultados obtenidos para el ejemplo 2 con TrussSolver.....	96
Figura 35 Salida gráfica TrussSolver para el ejemplo 2.....	100
Figura 36 Ejemplo 3.	101
Figura 37 Enumeración de nodos del ejemplo 3.	101
Figura 38 Ingreso de coordenadas x e y de los nodos del ejemplo 3.	102
Figura 39 Ingreso de restricciones en x e y de los nodos del ejemplo 3.	102
Figura 40 Ingreso del material y módulo de elasticidad del ejemplo 3.....	103
Figura 41 Ingreso de áreas transversales de cada uno de los elementos del ejemplo 3.....	103
Figura 42 Ingreso de inercias de los elementos del ejemplo 3.	104
Figura 43 Ingreso de conectividad de los elementos del ejemplo 3.	104
Figura 44 Ingreso cargas nodales del ejemplo 3.....	105
Figura 45 Ingreso cargas en elementos del ejemplo 3.....	105
Figura 46 Resultados obtenidos con el modelo para el ejemplo 3.	106
Figura 47 Salida gráfica TrussSolver para el ejemplo 3.....	108

ÍNDICE DE GRÁFICOS

Gráfico 1 Secuencia lógica del programa.....	35
Gráfico 2 Módulo de inputs TrussSolver 1 de 12.....	38
Gráfico 3 Módulo de inputs TrussSolver 2 de 12.....	38
Gráfico 4 Módulo de inputs TrussSolver 3 de 12.....	39
Gráfico 5 Módulo de inputs TrussSolver 4 de 12.....	39
Gráfico 6 Módulo de inputs TrussSolver 5 de 12.....	40
Gráfico 7 Módulo de inputs TrussSolver 6 de 12.....	40
Gráfico 8 Módulo de inputs TrussSolver 7 de 12.....	41
Gráfico 9 Módulo de inputs TrussSolver 8 de 12.....	41
Gráfico 10 Módulo de inputs TrussSolver 9 de 12.....	42
Gráfico 11 Módulo de inputs TrussSolver 10 de 12.....	42
Gráfico 12 Módulo de inputs TrussSolver 11 de 12.....	43
Gráfico 13 Módulo de inputs TrussSolver 12 de 12.....	43
Gráfico 14 Módulo de análisis TrussSolver 1 de 13.	44
Gráfico 15 Módulo de análisis TrussSolver 2 de 13.	44
Gráfico 16 Módulo de análisis TrussSolver 3 de 13.	45
Gráfico 17 Módulo de análisis TrussSolver 4 de 13.	45
Gráfico 18 Módulo de análisis TrussSolver 5 de 13.	46
Gráfico 19 Módulo de análisis TrussSolver 6 de 13.	46
Gráfico 20 Módulo de análisis TrussSolver 7 de 13.	47
Gráfico 21 Módulo de análisis TrussSolver 8 de 13.	47
Gráfico 22 Módulo de análisis TrussSolver 9 de 13.	48
Gráfico 23 Módulo de análisis TrussSolver 10 de 13.	48
Gráfico 24 Módulo de análisis TrussSolver 11 de 13.	49
Gráfico 25 Módulo de análisis TrussSolver 12 de 13.	49
Gráfico 26 Módulo de análisis TrussSolver 13 de 13.	50
Gráfico 27 Módulo de outputs TrussSolver 1 de 11.....	50
Gráfico 28 Módulo de outputs TrussSolver 2 de 11.....	51
Gráfico 29 Módulo de outputs TrussSolver 3 de 11.....	51
Gráfico 30 Módulo de outputs TrussSolver 4 de 11.....	52
Gráfico 31 Módulo de outputs TrussSolver 5 de 11.....	52
Gráfico 32 Módulo de outputs TrussSolver 6 de 11.....	53

Gráfico 33 Módulo de outputs TrussSolver 7 de 11.....	53
Gráfico 34 Módulo de outputs TrussSolver 8 de 11.....	54
Gráfico 35 Módulo de outputs TrussSolver 9 de 11.....	54
Gráfico 36 Módulo de outputs TrussSolver 10 de 11.....	55
Gráfico 37 Módulo de outputs TrussSolver 11 de 11.....	55
Gráfico 38 Módulo de inputs PlaneSolver 1 de 11.....	56
Gráfico 39 Módulo de inputs PlaneSolver 2 de 11.....	56
Gráfico 40 Módulo de inputs PlaneSolver 3 de 11.....	57
Gráfico 41 Módulo de inputs PlaneSolver 4 de 11.....	57
Gráfico 42 Módulo de inputs PlaneSolver 5 de 11.....	58
Gráfico 43 Módulo de inputs PlaneSolver 6 de 11.....	58
Gráfico 44 Módulo de inputs PlaneSolver 7 de 11.....	59
Gráfico 45 Módulo de inputs PlaneSolver 8 de 11.....	59
Gráfico 46 Módulo de inputs PlaneSolver 9 de 11.....	60
Gráfico 47 Módulo de inputs PlaneSolver 10 de 11.....	60
Gráfico 48 Módulo de inputs PlaneSolver 11 de 11.....	61
Gráfico 49 Módulo de análisis PlaneSolver 1 de 21.	61
Gráfico 50 Módulo de análisis PlaneSolver 2 de 21.	62
Gráfico 51 Módulo de análisis PlaneSolver 3 de 21.	62
Gráfico 52 Módulo de análisis PlaneSolver 4 de 21.	63
Gráfico 53 Módulo de análisis PlaneSolver 5 de 21.	63
Gráfico 54 Módulo de análisis PlaneSolver 6 de 21.	64
Gráfico 55 Módulo de análisis PlaneSolver 7 de 21.	64
Gráfico 56 Módulo de análisis PlaneSolver 8 de 21.	65
Gráfico 57 Módulo de análisis PlaneSolver 9 de 21.	65
Gráfico 58 Módulo de análisis PlaneSolver 10 de 21.	66
Gráfico 59 Módulo de análisis PlaneSolver 11 de 21.	66
Gráfico 60 Módulo de análisis PlaneSolver 12 de 21.	67
Gráfico 61 Módulo de análisis PlaneSolver 13 de 21.	67
Gráfico 62 Módulo de análisis PlaneSolver 14 de 21.	68
Gráfico 63 Módulo de análisis PlaneSolver 15 de 21.	68
Gráfico 64 Módulo de análisis PlaneSolver 16 de 21.	69
Gráfico 65 Módulo de análisis PlaneSolver 17 de 21.	69

Gráfico 66 Módulo de análisis PlaneSolver 18 de 21.	70
Gráfico 67 Módulo de análisis PlaneSolver 19 de 21.	70
Gráfico 68 Módulo de análisis PlaneSolver 20 de 21.	71
Gráfico 69 Módulo de análisis PlaneSolver 21 de 21.	71
Gráfico 70 Módulo de outputs PlaneSolver 1 de 11.....	72
Gráfico 71 Módulo de outputs PlaneSolver 2 de 11.....	72
Gráfico 72 Módulo de outputs PlaneSolver 3 de 11.....	73
Gráfico 73 Módulo de outputs PlaneSolver 4 de 11.....	73
Gráfico 74 Módulo de outputs PlaneSolver 5 de 11.....	74
Gráfico 75 Módulo de outputs PlaneSolver 6 de 11.....	74
Gráfico 76 Módulo de outputs PlaneSolver 7 de 11.....	75
Gráfico 77 Módulo de outputs PlaneSolver 8 de 11.....	75
Gráfico 78 Módulo de outputs PlaneSolver 9 de 11.....	76
Gráfico 79 Módulo de outputs PlaneSolver 10 de 11.....	76
Gráfico 80 Módulo de outputs PlaneSolver 11 de 11.....	77

1 GENERALIDADES

En este capítulo se desarrolla los antecedentes y alcances del modelo computacional para análisis de estructuras planas que se busca implementar en la universidad como herramienta de análisis para los estudiantes de la facultad de ingeniería civil.

1.1 Introducción

El método matricial de la rigidez sigue siendo en la actualidad una de las herramientas más eficaces y eficientes, tanto para el análisis como para el diseño estructural en distintas áreas de la ingeniería, empezando con la ingeniería civil como la automotriz, aeroespacial, mecánica, etc. Una de las mayores ventajas y particularidades del método es que los problemas con geometrías y condiciones de frontera complejas pueden ser tratados individualmente reduciendo su complejidad al dividirlos en pequeños subdominios, llamados “elementos finitos”. La implementación del método ha permitido reducir tiempo y costos en el diseño y rediseño de estructuras, máquinas y dispositivos mecánicos.

Debido a que en la actualidad el uso de la programación se ha extendido a numerosas ramas de la ingeniería con gran éxito, resulta fundamental que los estudiantes de una carrera de ingeniería, sobre todo los que se dedican al diseño de sistemas estructurales, mecánicos, de transporte de fluidos, y eléctricos, conozcan sus fundamentos y diferentes aplicaciones.

Por lo anterior, en esta tesis se presenta una de las aplicaciones básicas del método de análisis matricial, que es el cálculo de estructuras en dos dimensiones, que son tan ampliamente utilizadas para la construcción de sistemas de soporte mecánicos, armaduras, la construcción de tableros en puentes y plataformas.

En adición, líneas abajo encontrarán la utilidad práctica de la programación en Matlab para modelar problemas ingenieriles como en el presente caso la modelación de un software que resuelva estructuras en dos dimensiones; con la finalidad de que exista una mayor comprensión del funcionamiento del método matricial sin tener que utilizar un software comercial especializado.

1.2 Antecedentes

El fundamento teórico de los métodos de matriz de análisis estructural fue puesto por James C. Maxwell, que introdujo el método de las deformaciones consistentes en 1864; y George A. Maney, quien desarrolló el método de deflexión pendiente de en 1915. Se considera que estos métodos clásicos que son los precursores de los métodos matriz de flexibilidad y rigidez, respectivamente. En la era precomputacional, la principal desventaja de estos métodos anteriores era que requerían solución directa de las ecuaciones algebraicas, una tarea formidable simultánea por los cálculos a mano en caso de más de un par de incógnitas.

Con la invención las computadoras hacia finales de los años 40, se inició una revolución en el análisis estructura, ya que con la ayuda de estas se podía resolver grandes sistemas de ecuaciones al mismo tiempo. Estas ecuaciones se podían expresar de forma matricial y a su vez ser programadas, de manera conveniente, para que sean solucionadas con la ayuda del ordenador.

Los métodos clásicos de análisis estructural desarrollado a fines del siglo XIX, tienen las cualidades de la generalidad, simplicidad lógica y elegancia matemática. Desgraciadamente, conducían a menudo a cálculos muy laboriosos cuando se los aplicaba en casos prácticos, y en aquella época, esto era un gran defecto. Por esta razón sucesivas generaciones de ingenieros se dedicaron a tratar de reducir el conjunto de cálculos. Muchas técnicas ingeniosas de gran valor práctico fueron apareciendo (Método de Cross), pero la mayoría de las mismas eran aplicable sólo a determinados tipos de estructuras. La principal objeción a los primeros métodos de análisis fue que los mismos conducían a sistemas con un gran número de ecuaciones lineales, difíciles de resolver manualmente. Con los computadores, capaces de realizar el trabajo numérico, esta objeción no tiene ahora sentido, mientras que la generalidad de los métodos permanece. Esto explica por qué los métodos matriciales deben en su tratamiento básico de las estructuras más al siglo XIX que al XX.

En el Perú, las investigaciones relacionadas al análisis matricial no son recurrentes a nivel de tesis, más aún la creación de modelos computacionales que permitan analizar el comportamiento de estructuras.

La presente tesis busca ser pionera en la creación de un modelo computacional desarrollado en esta casa de estudios, utilizando el análisis matricial de estructuras reticulares delimitado a armaduras y pórticos.

1.3 Fundamentación

Nuestra casa de estudios en sus más de 20 años, no ha presentado ningún estudio relacionado a modelamientos computacionales de la rama de estructuras, considerando la relevante importancia de ello, consideramos que el proyecto de tesis a desarrollar responde a esta ausencia de investigación.

El presente proyecto de tesis quedará en código libre para los futuros tesisas de nuestra casa de estudios. Esperamos que esta pueda ser continuada por las promociones venideras. Tenemos en consideración que toda gran caminata empieza por un primer paso, y es lo que esperamos sea este proyecto de tesis, el primer gran paso para el desarrollo de un poderoso software desarrollado íntegramente en nuestra casa de estudios.

Además, el proyecto de tesis responde a la problemática del cálculo estructural de armaduras y pórticos. Permitiendo conocer, al usuario del modelo, no solo los resultados del cálculo estructural sino los componentes que permitieron alcanzar dichos resultados.

1.4 Objetivos de la investigación

1.4.1 Objetivo general

Crear un modelo computacional con el software Matlab, que efectúe el análisis de una estructura reticular empleando el método matricial para armaduras y pórticos en dos dimensiones.

1.4.2 Objetivos específicos

- Obtener resultados con un margen de error no mayor al 0.05% en comparación a otros softwares que usen la misma metodología de análisis, el software de Aslam Kassimali por ejemplo.
- Brindar una salida gráfica al modelo de cómputo para realizar una inspección visual del comportamiento de las estructuras en análisis.
- Obtener resultados con un margen de error no mayor al 0.05% en comparación a otros softwares que usen la misma metodología de análisis, el software de Aslam Kassimali por ejemplo.
- Brindar una herramienta de comparación y validación de cálculo para estudiantes de los cursos de análisis estructural.

1.5 Delimitación

El alcance de la siguiente tesis, es el desarrollo de un modelo computacional que resuelva estructuras reticulares del tipo, armaduras y pórticos en dos dimensiones.

El programa arrojará resultados de desplazamiento de nodos, fuerza de reacción en los apoyos, fuerzas internas de los miembros, además de una salida gráfica en la que se pueda observar el desplazamiento de la estructura.

El programa no brindará los diagramas de fuerzas en los elementos (axial, cortante y momentos), pues consideramos que esto requiere una investigación adicional.

Dentro de las armaduras, el modelo está orientado a resolver las estructuras que estén conformadas por barras rectas, no curvas.

En el caso de pórticos, el modelo resuelve aquellas que estén conformados por elementos rectos (vigas y columnas). Además, de momento sólo se han implementado la resolución de problemas con cargas del tipo axial, cortante y momento en los nodos; y cargas axiales, cortantes y distribuidas rectangulares en los miembros de la estructura.

En ambos casos de análisis, el número de miembros que pueda resolver el modelo estará limitado, en parte, por la capacidad de memoria del computador en el cual se use.

2 MARCO TEÓRICO

2.1 Conceptos fundamentales de análisis estructural

Para abordar los conceptos fundamentales que formar parte del análisis estructural, es necesario conocer que éstos implican tres tipos de relaciones:

- Ecuaciones de equilibrio
- Condiciones de compatibilidad
- Relaciones constitutivas

Ecuaciones de equilibrio

En estática, se dice que una estructura, o uno de sus miembros, se encuentran en equilibrio cuando mantiene un balance de fuerzas y momentos. En general, se requiere que las ecuaciones de equilibrio por fuerzas y momentos se satisfagan a lo largo de tres ejes independientes, de acuerdo a las ecuaciones 2.1, 2.2 y 2.3.

$$\sum F_x = 0 \quad 2.1$$

$$\sum F_y = 0 \quad 2.2$$

$$\sum M_o = 0 \quad 2.3$$

De las ecuaciones anteriores, $\sum F_x$ y $\sum F_y$, representan las sumas algebraicas de las componentes x e y de todas las fuerzas que actúan sobre la estructura o sobre uno de sus miembros. Además, $\sum M_o$ representa la suma algebraica de los momentos de las componentes de fuerza x e y respecto a un eje perpendicular al plano x-y (eje z) que pasa por el punto O.

Cada vez que se apliquen estas ecuaciones, será necesario primero dibujar un diagrama de cuerpo libre de la estructura o de sus miembros. En el caso de seleccionar un miembro, se debe aislar a este de sus soportes y entorno como se muestra en la figura 1.

Figura 1. Barra de equilibrio.

Fuente propia

Condición de compatibilidad (condiciones de continuidad)

Las condiciones de compatibilidad relacionan las deformaciones de una estructura de manera que varias partes (miembros, articulaciones y soportes) encajan entre sí sin huecos o superposiciones. Estas condiciones (también conocidas como las condiciones de continuidad) aseguran que la forma deformada de la estructura es continua (excepto en los lugares donde existan bisagras internas o rodillos), y es consistente con las condiciones de apoyo.

Figura 2. Forma deformada de un pórtico.

Fuente propia

Consideremos, por ejemplo, el marco plano de dos miembros se muestra en la figura 2. La forma deformada del pórtico debido a una carga arbitraria se representa, utilizando una escala exagerada. Cuando se analiza una estructura, las condiciones de compatibilidad se utilizan para relacionar los desplazamientos finales de los miembros con desplazamientos conjuntos que, a su vez, están relacionados con las condiciones de apoyo. Por ejemplo, debido a que el nodo 1 de unión de la trama en la figura 2 está unido a un soporte de rodillos que no se puede trasladar en la dirección vertical, el desplazamiento vertical de esta junta debe ser cero. Del mismo modo, como el nodo 3 está unido a un soporte fijo que no puede ni girar ni trasladar en cualquier dirección, la rotación y los desplazamientos verticales y horizontales del nodo 3 deben ser cero.

Los desplazamientos de los extremos de los miembros están relacionados con los desplazamientos del nodo por el requisito de compatibilidad. De esta forma, los desplazamientos del extremo del miembro deben ser los mismos que los desplazamientos de la articulación donde el extremo del miembro está conectado. Por lo tanto, como se muestra en la figura 2, debido a que el nodo 1 del pórtico se desplaza hacia la derecha por una distancia d_1 y gira en sentido horario un ángulo θ_1 , el extremo izquierdo del elemento horizontal (miembro 1) que está unido al nodo 1 también debe trasladarse a la derecha una distancia d_1 y girar en sentido horario por θ_1 ángulo. Del mismo modo, debido a que los desplazamientos del nodo 2 consisten en desplazamientos d_2 hacia la derecha, d_3 hacia abajo y un giro θ_2 anti horario, el extremo derecho del miembro horizontal y el extremo superior del miembro vertical que están conectados en el nodo 2 deben también someterse a los mismos desplazamientos (es decir, D_2 , D_3 , y θ_2).

El extremo inferior del miembro vertical, sin embargo, no se somete a ningún desplazamiento, ya que el nodo 3 está empotrado, es decir, no puede girar ni trasladarse en cualquier dirección.

Por último, la compatibilidad exige que las formas deformadas de los miembros de una estructura sean continuas (excepto en algún bisagras internas o rodillos) y ser coherente con los desplazamientos en los extremos correspondientes de los miembros.

Relaciones constitutivas

Las relaciones constitutivas (también conocidas como las relaciones de tensión-deformación) describen las relaciones entre las deformaciones y tensiones de una estructura de acuerdo con las propiedades de tensión-deformación del material estructural. Como se discutió anteriormente, las ecuaciones de equilibrio proporcionan relaciones entre las fuerzas, mientras que las condiciones de compatibilidad implican sólo deformaciones. Las relaciones constitutivas proporcionan el vínculo entre las ecuaciones de equilibrio y compatibilidad, condiciones que son necesarias para establecer las relaciones carga-deformación para una estructura o un miembro.

En el análisis de las estructuras aporticadas, las relaciones básicas de tensión-deformación son utilizadas, junto con las ecuaciones de equilibrio y de compatibilidad, para establecer las relaciones entre las fuerzas y desplazamientos en los extremos de un miembro.

De este modo las relaciones de fuerza-desplazamiento obtenidos del miembro son tratados entonces como las relaciones constitutivas de toda la estructura, y se utilizan para unir las ecuaciones de equilibrio y de compatibilidad de la estructura, dando de ese modo la relación carga-deformación para toda la estructura. Esta relación carga-deformación puede ser luego resuelta para determinar las deformaciones de la estructura debido a una carga dada.

En el caso de estructuras estáticamente determinadas, las ecuaciones de equilibrio pueden ser resueltas independientemente de las relaciones de compatibilidad y constitutivas para obtener las reacciones y fuerzas en los miembros. Las deformaciones de la estructura, si es deseado, a continuación, se pueden determinar mediante el empleo de las relaciones de compatibilidad y constitutivas. En el análisis de estructuras estáticamente indeterminadas, sin embargo, las ecuaciones de equilibrio por sí solos no son suficientes para determinar las reacciones y fuerzas en los miembros. Por lo tanto, se hace necesario para satisfacer simultáneamente los tres tipos de relaciones fundamentales (es decir, el equilibrio, la compatibilidad, y las relaciones constitutivas) para determinar la respuesta estructural.

Los métodos de análisis estructural matricial se formulan habitualmente por aplicación directa de las tres relaciones fundamentales como se describe en términos generales en los párrafos anteriores. Sin embargo, los métodos matriciales también se pueden formular mediante el uso de principios de trabajo con la energía que satisfagan las tres relaciones fundamentales indirectamente.

Los principios de trabajo-energía son generalmente preferidos en la formulación de métodos de elementos finitos, ya que pueden ser más convenientemente aplicados a derivar las relaciones de fuerza-desplazamiento aproximadas para los elementos de estructuras de superficie y sólidas. Para efectos de esta tesis; en el caso de análisis de armaduras se toma en consideración la rigidez axial, mientras que para pórticos se toma en consideración la rigidez axial, rigidez flexional, rigidez frente al cortante y la rigidez flexión cortante.

Los métodos de matriz que se presenta en esta tesis se formulan mediante la aplicación directa del equilibrio, compatibilidad y relaciones constitutivas.

2.2 Elementos estructurales reticulares

Las estructuras reticulares independientemente de su material son categorizadas por su forma geométrica: vigas, armaduras planas, armaduras espaciales, pórticos planos, parrillas y

pórticos espaciales. Sin embargo, su diferencia geométrica le da características especiales a cada tipo de estructura reticular.

Estos tipos de estructuras se muestran en las figuras 3 (a-f). Además, los dos principios fundamentales mencionados al inicio de esta sección, ecuaciones de equilibrio y condiciones de compatibilidad, aplican los mismos principios básicos para cada una de estas estructuras, los análisis para cada tipo de armadura son diferentes en detalles, es por eso que, a pesar de ser estructuras reticulares se deben analizar cada una de manera independiente.

Figura 3. Tipos de estructuras reticulares; viga (a), armadura plana (b), pórtico plano (c), parrilla (d), armadura espacial (e), pórtico espacial (f).

Fuente propia

Cada estructura reticular está formada por miembros que son mayores en dimensión longitudinal comparados con su dimensión transversal. Los puntos donde los miembros se

intersectan o conectan se llaman “nodos”, así como los puntos de apoyo de los elementos y los puntos de los extremos de un elemento.

Ejemplos de nodos son los puntos A, B, C y D de las figuras 3(a) y 3(c). Los apoyos pueden ser del tipo “empotrado”, como el soporte A de la viga de la figura 3(a), o “articulado” como se muestra en el soporte A del pórtico plano de la figura 3(c), o puede ser “deslizante”, ilustrado por los apoyos B y D de la figura 3(a). En casos especiales, las conexiones entre los elementos o entre los elementos y apoyos, pueden ser elásticas (o semirígidas).

Las “cargas” en una estructura reticular pueden ser fuerzas concentradas, cargas distribuidas o pares (flectores o torsores). A continuación, se detallará la característica principal de cada tipo de estructura mostrada en las figuras 3(a-f).

Viga: Una viga mostrada en la figura 3(a) es un elemento recto con uno o más puntos de apoyo, como los graficados (A, B y D) en la figura 3(a). Las fuerzas aplicadas a una viga están bajo el supuesto de que estas actúan en un eje de simetría contenidas en un plano de la sección transversal de la viga, entiéndase por eje de simetría a un eje principal de la sección transversal. Adicionándole propiedades, en una viga todos los pares exteriores que actúan sobre la misma tienen vectores de momento normales a ese plano, y la deformación que sufre la viga está contenida en el mismo plano (flexión) sin sufrir torsión. En cualquier corte transversal a la viga existirán fuerzas internas, fuerzas cortantes y un momento flector.

Armadura plana: Una armadura plana mostrada en la figura 3(b) está idealizada como un sistema compuesta por elementos interconectados en nodos articulados en un plano. En el caso de armaduras planas se asume que todas las fuerzas actúan en el plano de la estructura, y todos los pares externos tienen vectores normales al plano. Con respecto a las cargas que puedan actuar en la armadura plana, estas pueden ser cargas puntuales concentradas en los nodos de la estructura o cargas que actúan sobre los elementos, considerando que estos elementos están sometidos únicamente a fuerzas de tensión y compresión.

Pórtico Plano: Un pórtico plano mostrada en la figura 3(c) se encuentra constituido por elementos en un único plano con eje de simetría en el mismo plano. Los nodos como los B y C, son conexiones rígidas. Por este motivo, tanto las fuerzas actuantes en un pórtico como sus desplazamientos ocurren en el mismo plano estructural. Cada par que actúe sobre un pórtico tienen sus vectores de momento normales al plano. Por ende, las fuerzas internas

resultantes que actúan en cualquier sección de un elemento están compuestas, en general, por una fuerza axial, un par flector y una fuerza cortante.

Parrilla: Una parrilla mostrada en la figura 3(d) está compuesta por elementos continuos, pero estos se intersectan o cruzan. En el caso la armadura, las conexiones entre los elementos se consideran, a menudo, como articuladas, en tanto que, en la viga, las conexiones se consideran rígidas. Mientras que en el pórtico plano las fuerzas externas están, todas, en el plano de la estructura, en el caso de una parrilla, todas las fuerzas actúan perpendicularmente al plano de la misma. Debido a eso, en una parrilla se puede identificar resultados de torsión como de flexión en algunos de sus miembros, considerando que estos tienen dos ejes de simetría transversalmente permitiendo que la flexión y torsión sean independientes. Es decir, en la sección transversal de un miembro que forma parte de una parrilla, en general, se genera fuerza cortante, momento flector y momento torsor.

Armadura espacial: Una armadura espacial mostrada en la figura 3(e) guarda similitud a una armadura plana con la particularidad que sus elementos pueden estar orientados en cualquier dirección del espacio. Esta condición influye en el comportamiento de sus fuerzas puesto que tendrán direcciones arbitrarias, sin embargo, cualquier par aplicado tendrá su vector momento perpendicular al eje del elemento. Este comportamiento ocurre porque un miembro de esta armadura es incapaz de soportar momento torsor.

Pórtico espacial: Un pórtico espacial mostrada en la figura 3(f) es el tipo de estructura más general de las reticulares, no tiene restricciones en la posición de sus nodos, dirección de los miembros o de sus cargas. Estos miembros pueden soportar fuerzas axiales, pares torsores, pares flectores en cualquiera de sus dos direcciones de sus secciones transversales, y fuerzas cortantes en las direcciones principales. Al igual que un elemento de parrilla se considera que este tiene dos planos de simetría.

2.3 Análisis matricial de estructuras por el método de rigidez

En esencia, hay dos formas de analizar las estructuras utilizando métodos matriciales. El método de la rigidez, es un método de análisis de desplazamientos. Para analizar estructuras también puede emplearse un método de fuerza, llamado el método de la flexibilidad. Sin embargo, el método de la rigidez puede usarse tanto para analizar estructuras estáticamente determinadas como indeterminadas. Mientras que el método de la flexibilidad necesita un procedimiento diferente para cada uno de estos dos casos. Inclusive en el método de la

rigidez se obtienen los desplazamientos y fuerzas de forma directa. Además, por lo general es mucho más fácil formular las matrices necesarias para realizarlas operaciones en computadoras mediante el método de la rigidez; y una vez hecho esto, los cálculos en computadora pueden realizarse de modo eficiente.

La aplicación del método de la rigidez requiere subdividir las estructuras en una serie de elementos finitos discretos e identificar sus puntos extremos como nodos. Para el análisis de la estructura los elementos finitos se representan mediante cada uno de los elementos que la componen y los nodos representan las juntas. Se determinan las propiedades de la fuerza-desplazamiento en cada elemento y después se relacionan entre sí usando las ecuaciones de equilibrio de fuerzas escritas en los nodos. Luego, estas relaciones, para toda la estructura, se agrupan en lo que se denomina matriz de rigidez de la estructura **K**. Una vez establecido esto, se pueden determinar los desplazamientos desconocidos de los conocidos para cualquier carga dada sobre la estructura. Al conocer estos desplazamientos pueden calcularse las fuerzas externas e internas en la estructura utilizando las relaciones de fuerza-desplazamiento para cada elemento.

Antes de desarrollar un procedimiento formal para aplicar el método de la rigidez es necesario establecer primero algunas definiciones y conceptos preliminares.

Elemento y Nodo

Uno de los primeros pasos para aplicar el método de la rigidez consiste en identificar los elementos o miembros de la estructura y sus nodos. Cada elemento se especificará por un número encerrado en un cuadrado, y para identificar los nodos se usará un número dentro de un círculo. También se identificarán los extremos “cercano” y “lejano” de cada elemento mediante una flecha indicada a lo largo del elemento, con la punta de la flecha dirigida hacia el extremo lejano. En la figura 4 (a) se muestran algunos ejemplos con la identificación del elemento, el nodo y la dirección para una armadura. Estas asignaciones se hicieron de manera arbitraria.

Figura 4. Identificación de elementos en una armadura.

Fuente propia

Coordenadas globales y del elemento

Dado que las cargas y los desplazamientos son cantidades vectoriales, es necesario establecer un sistema de coordenadas a fin de precisar el sentido correcto de la dirección. Aquí se usarán dos tipos de diferentes sistemas coordenados. Se usará un sistema de coordenadas de la estructura global x e y , el cual es único, y sirve para especificar el sentido de cada uno de los componentes de la fuerza externa y el desplazamiento de los nodos, figura 4(a). Se empleará un sistema de coordenadas locales o de elementos para especificar el sentido de dirección de sus desplazamientos y las cargas internas del elemento. Este sistema se identificará con ejes x' , y' con el origen en el nodo cercano y el eje x extendiéndose hacia nodo lejano. En la figura 4(b) se muestra un ejemplo para el elemento 5.

Indeterminación cinemática

Los grados de libertad no restringidos representan las incógnitas primarias de cualquier método de desplazamiento, y por lo tanto estos deben identificarse. Como regla general, en el caso de las armaduras hay dos grados de libertad o dos posibles desplazamientos, para cada junta (nodo), mientras que en el caso de los pórticos planos estos grados de libertad suben a 3 (por nodo). Para su aplicación, cada grado de libertad se especificará sobre la armadura o pórtico plano mediante un código numérico, mostrado en la junta o nodo y se referirá a su dirección coordenada global positiva con una flecha asociada. Por ejemplo, la armadura de la figura 4 (a) tiene doce grados de libertad, los cuales se han identificado mediante los códigos del 1 al 12, como se muestra. La armadura es cinemáticamente indeterminada de quinto grado debido a estos ocho posibles desplazamientos; del 1 al 9

representan grados de libertad desconocidos o sin restricciones, y del 10 al 12 representan grados de libertad restringidos. Debido a las restricciones aquí los desplazamientos son iguales a cero. Para futuras aplicaciones, los números de código más bajo se usarán siempre para identificar los desplazamientos desconocidos, y los número de código más alto para los conocidos. La razón para elegir este método de identificación tiene que ver con la comodidad posterior al dividir la matriz de rigidez de la estructura, de modo que los desplazamientos desconocidos se puedan encontrar de forma más directa.

Una vez etiquetada la estructura y especificados los números de código, se puede determinar la matriz de rigidez de la estructura **K**. Para hacer esto primero debe establecerse una matriz de rigidez del elemento **k'** para cada elemento de carga-desplazamiento del elemento en términos de las coordenadas locales. Puesto que no todos los elementos de la estructura están en la misma dirección se debe desarrollar una manera de transformar estas cantidades desde el sistema local de coordenadas x',y' de cada elemento al sistema de coordenadas x,y de la estructura global. Esto puede hacerse empleando matrices de transformación (**T**). Una vez establecidos, los elementos de la matriz de rigidez del elemento se transforman de las coordenadas locales a las globales, y luego se juntan para crear la matriz de rigidez de la estructura. Si se usa **K**, como se estableció anteriormente, es posible determinar primero los desplazamientos del nodo para después encontrar las reacciones en los soportes y las fuerzas de los elementos. Ahora se trabajará en el desarrollo de este método.

En este método se obtiene, primero, una estructura modificada, bloqueando los desplazamientos de todos los nudos que son fáciles de analizar. Luego, se superponen otras soluciones complementarias para determinar los verdaderos desplazamientos que ocurren en los nudos. El número de ecuaciones a resolver es igual al número del grado de indeterminación cinemática. Primero se aplica el principio de compatibilidad y luego el de equilibrio.

2.4 Matriz de rigidez del elemento

2.4.1 Matriz de rigidez del elemento en coordenadas locales en armaduras

La matriz de rigidez para un solo elemento de una armadura con coordenadas locales x',y' , orientadas como se muestra en figura 5. Los términos de esta matriz representan las relaciones de carga-desplazamiento para el elemento.

Un elemento de una armadura solo puede desplazarse a lo largo de su eje (eje x') puesto que las cargas se aplican a lo largo de ese eje. Por lo tanto, pueden ocurrir dos desplazamientos independientes. Cuando se impone un desplazamiento positivo d_N sobre el extremo cercano del elemento, mientras el extremo lejano se mantiene articulado como se muestra a la figura 5(a), las fuerzas desarrolladas en los extremos de los elementos se expresan en la de acuerdo a la expresión 2.4.

$$q'_N = \frac{AE}{L} d_N \quad q'_F = -\frac{AE}{L} d_N \quad 2.4$$

Observe que q'_F es negativa para lograr el equilibrio, actúa en la dirección negativa a x' . Del mismo modo, un desplazamiento positivo d_F en el extremo lejano, que mantiene al extremo cercano articulado, figura 5(b), resulta en las siguientes fuerzas de elemento mostradas en las expresiones 2.5

$$q''_N = -\frac{AE}{L} d_F \quad q''_F = \frac{AE}{L} d_F \quad 2.5$$

Por superposición, figura 4(c), las fuerzas resultantes causadas por ambos desplazamientos se expresan en las ecuaciones 2.6 y 2.7

$$q_N = \frac{AE}{L} d_N - \frac{AE}{L} d_F \quad 2.6$$

$$q_F = -\frac{AE}{L} d_N + \frac{AE}{L} d_F \quad 2.7$$

Estas ecuaciones de carga-desplazamiento pueden escribirse en forma matricial como se expresa en la ecuación 2.8

$$\begin{bmatrix} q_N \\ q_F \end{bmatrix} = \frac{AE}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} d_N \\ d_F \end{bmatrix} \quad 2.8$$

O bien

$$\mathbf{q} = \mathbf{k}' \mathbf{d} \quad 2.9$$

Donde $\mathbf{k}'$ se expresa de acuerdo a la matriz 2.10

$$\mathbf{k}' = \frac{AE}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad 2.10$$

Esta matriz, $\mathbf{k}'$, se denomina matriz de rigidez del elemento, y tiene la misma forma para cada elemento de la armadura. Los cuatro elementos que la componen se llaman coeficientes

de influencia de la rigidez del elemento, k'_{ij} . Físicamente, k'_{ij} representa la fuerza en la junta i cuando se impone un desplazamiento unitario en la junta j . Por ejemplo, si $i=j=1$, entonces k'_{11} es la fuerza en la junta cercana cuando la junta lejana se mantiene fija, y la junta cercana experimenta un desplazamiento de $dN=1$, de acuerdo a lo expresado en la ecuación 2.11

$$q_N = k'_{11} = \frac{AE}{L} \quad 2.11$$

Del mismo modo, la fuerza en la junta lejana se determina a partir de $i=2, j=1$, como se muestra en la ecuación 2.12.

$$q_F = k'_{21} = -\frac{AE}{L} \quad 2.12$$

Estos dos términos representan la primera columna de la matriz de rigidez del elemento. De la misma manera, la segunda columna de esta matriz representa las fuerzas en el elemento sólo cuando el extremo lejano del elemento experimenta un desplazamiento unitario.

Figura 5. Elemento orientado en coordenadas locales.

Fuente propia

2.4.2 Matriz de rigidez del elemento en coordenadas locales en pórticos planos

Para el caso de pórticos planos, cada uno de sus elementos está sometido a cargas axiales, de corte y momentos de flexión en sus extremos. Estas cargas actúan todas en las direcciones de coordenadas positivas junto con sus desplazamientos asociados como se muestra en la figura 6.

Figura 6. Elemento de un pórtico plano con sus fuerzas actuantes

Fuente propia.

Por superposición, si se añaden estos resultados, las seis relaciones de carga-desplazamiento resultantes para el miembro se pueden expresar en forma de matriz como la expresión 2-13.

$$k = \begin{bmatrix} \frac{AE}{L} & 0 & 0 & -\frac{AE}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{AE}{L} & 0 & 0 & \frac{AE}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \quad 2.13$$

Donde:

k: Matriz de rigidez del elemento barra de un pórtico plano en coordenadas locales

2.5 Matriz de transformación de fuerza y desplazamiento

2.5.1 Matriz de transformación de fuerza y desplazamiento para armaduras

Como una armadura está compuesta de muchos elementos (miembros), ahora se desarrollará un método para transformar las fuerzas $\mathbf{q}$ del elemento y los desplazamientos $\mathbf{d}$ definidos en coordenadas locales a coordenadas globales. A fin de establecer una convención, se considerará que las coordenadas globales “x” positivas están a la derecha y las coordenadas “y” positivas hacia arriba. Los ángulos menores entre los ejes globales x,y, positivos y el eje local x’ positivo se definirán como θ_x y θ_y tal como se muestra en la figura 7. Los cosenos de estos ángulos se usarán en el análisis matricial que sigue. Estos se identificarán como $\lambda_x = \cos \theta_x$, $\lambda_y = \cos \theta_y$. Los valores numéricos para λ_x y λ_y , pueden generarse fácilmente por computadora una vez que se han especificado las coordenadas x,y del extremo cercano N y del extremo lejano F del elemento. Por ejemplo, considere el elemento NF de la armadura que se muestra en la figura 8. Aquí, las coordenadas de N y F son (X_n, Y_n) y (X_f, Y_f) , respectivamente. Por lo tanto, se expresan en las expresiones 2.14 y 2.15.

$$\lambda_x = \cos \theta_x = \frac{x_F - x_N}{L} = \frac{x_F - x_N}{\sqrt{(x_F - x_N)^2 + (y_F - y_N)^2}} \quad 2.14$$

$$\lambda_y = \cos \theta_y = \frac{y_F - y_N}{L} = \frac{y_F - y_N}{\sqrt{(x_F - x_N)^2 + (y_F - y_N)^2}} \quad 2.15$$

Los signos algebraicos de estas ecuaciones “generalizadas” contarán de forma automática para los elementos orientados en otros cuadrantes del plano x-y.

Figura 7. Ángulos θ_x e θ_y .

Fuente propia.

Figura 8. Elemento ejemplo NF.

Fuente propia.

Matriz de Transformación del desplazamiento

En las coordenadas globales, cada extremo del elemento puede tener dos grados de libertad o desplazamientos independientes; a saber, la junta N tiene D_{Nx} y D_{Ny} , figuras 9(a) y 9(b), y la junta F tiene D_{Fx} y D_{Fy} , figuras 9(c) y 9(d). Ahora se considerarán cada uno de estos desplazamientos por separado, a fin de determinar su desplazamiento de componente a lo largo del elemento. Cuando el extremo lejano se mantiene articulado y al extremo cercano se le da un desplazamiento global D_{Nx} , figura 9(a), el desplazamiento correspondiente (deformación) a lo largo del elemento es $D_{Nx} \cos \theta_x$. Del mismo modo, un desplazamiento D_{Ny} ocasionará que el elemento se desplace $D_{Ny} \cos \theta_y$ a lo largo del eje x' , figura 9(b). El efecto de ambos desplazamientos globales hace que el elemento se desplace de acuerdo a la ecuación 2.16.

$$d_N = D_{Nx} \cos \theta_x + D_{Ny} \cos \theta_y \quad 2.16$$

De manera similar, los desplazamientos positivos D_{Fx} y D_{Fy} , aplicados en forma sucesiva en el extremo lejano F, mientras el extremo cercano se mantiene articulado, figura 9 (c) y 9 (d), hará que el elemento se desplace de acuerdo a lo definido en la ecuación 2.17.

$$d_F = D_{F_x} \cos \theta_x + D_{F_y} \cos \theta_y \quad 2.17$$

Si se considera que $\lambda_x = \cos \theta_x$ y $\lambda_y = \cos \theta_y$ representan los cosenos directores para el elemento, se tienen las ecuaciones 2.18 y 2.19.

$$d_N = D_{N_x} \lambda_x + D_{N_y} \lambda_y \quad 2.18$$

$$d_F = D_{F_x} \lambda_x + D_{F_y} \lambda_y \quad 2.19$$

Que puede escribirse en forma matricial como expresado en 2.20

$$\begin{bmatrix} d_N \\ d_F \end{bmatrix} = \begin{bmatrix} \lambda_x & \lambda_y & 0 & 0 \\ 0 & 0 & \lambda_x & \lambda_y \end{bmatrix} \begin{bmatrix} D_{N_x} \\ D_{N_y} \\ D_{F_x} \\ D_{F_y} \end{bmatrix} \quad 2.20$$

O bien de acuerdo a la expresión 2.21

$$\mathbf{d} = \mathbf{T} \mathbf{D} \quad 2.21$$

Donde T se puede definir de acuerdo a 2.22

$$\mathbf{T} = \begin{bmatrix} \lambda_x & \lambda_y & 0 & 0 \\ 0 & 0 & \lambda_x & \lambda_y \end{bmatrix} \quad 2.22$$

De la deducción anterior, **T** transforma los cuatro desplazamientos **D** globales x,y, en los dos desplazamientos **d** locales x'. Por lo tanto, **T** se conoce como la matriz de transformación del desplazamiento.

Figura 9. Desplazamientos del elemento..

Adaptado de “Structural Analysis 8th edition”, por R.C. Hibbeler, 2009.

Matriz de transformación de fuerza

Considere ahora la aplicación de la fuerza q_N , sobre el extremo cercano del elemento, el extremo lejano se mantiene articulado, figura 10 (a). Aquí las componentes de la fuerza global de q_N en N que se expresan en 2.23 y 2.24

$$Q_{N_x} = q_N \cos \theta_x \quad 2.23$$

$$Q_{N_y} = q_N \cos \theta_y \quad 2.24$$

Del mismo modo, si se aplica q_F a la barra, figura 10 (b), las componentes de la fuerza global en F se expresan en 2.25 y 2.26

$$Q_{F_x} = q_F \cos \theta_x \quad 2.25$$

$$Q_{F_y} = q_F \cos \theta_y \quad 2.26$$

Con base en los cosenos directores $\lambda_x = \cos \theta_x$, $\lambda_y = \cos \theta_y$, estas ecuaciones se convierten en las expresiones 2.27, 2.28, 2.29 y 2.30.

$$Q_{N_x} = q_N \lambda_x \quad 2.27$$

$$Q_{N_y} = q_N \lambda_y \quad 2.28$$

$$Q_{F_x} = q_F \lambda_x \quad 2.29$$

$$Q_{F_y} = q_F \lambda_y \quad 2.30$$

Que puede escribirse en forma matricial de acuerdo a la expresión 2.31.

$$\begin{bmatrix} Q_{N_x} \\ Q_{N_y} \\ Q_{F_x} \\ Q_{F_y} \end{bmatrix} = \begin{bmatrix} \lambda_x & 0 \\ \lambda_y & 0 \\ 0 & \lambda_x \\ 0 & \lambda_y \end{bmatrix} \begin{bmatrix} q_N \\ q_F \end{bmatrix} \quad 2.31$$

O bien de acuerdo a la ecuación 2.32

$$\mathbf{Q} = \mathbf{T}^T \mathbf{q} \quad 2.32$$

Donde $\mathbf{T}^T$ se expresa de acuerdo a 2.33

$$T^T = \begin{bmatrix} \lambda_x & 0 \\ \lambda_y & 0 \\ 0 & \lambda_x \\ 0 & \lambda_y \end{bmatrix} \quad 2.33$$

En este caso, $\mathbf{T}^T$ transforma las dos fuerzas $\mathbf{q}$ locales (x') que actúan en los extremos del elemento en las cuatro componentes de la fuerza $\mathbf{Q}$ global (x,y). Por comparación, puede establecerse que esta matriz de transformación de la fuerza es la transpuesta de la matriz de transformación del desplazamiento.

Figura 10. Elemento bajo fuerza.

Adaptado de “Structural Analysis 8th edition”, por R.C. Hibbeler, 2009.

2.5.2 Matriz de transformación de fuerza y desplazamiento para pórticos planos

Como en el caso de las armaduras, debemos ser capaces de transformar las cargas internas “q” y las deformaciones “d” de las coordenadas locales a las coordenadas globales x, y, z. Por esta razón son necesarias matrices de transformación.

Matriz de transformación de desplazamientos

De la figura 11(a), se puede observar que un desplazamiento global D_{Nx} crea desplazamientos en coordenadas locales según la expresión 2.34

$$d_{Nx'} = D_{Nx} \cos \theta_x \quad d_{Ny'} = -D_{Nx} \cos \theta_y \quad 2.34$$

Y de manera análoga, de la figura 2-11(b), un desplazamiento global D_{Ny} crea desplazamientos en coordenadas locales según la expresión 2.35

$$d_{Nx'} = D_{Ny} \cos \theta_y \quad d_{Ny'} = D_{Ny} \cos \theta_x \quad 2.35$$

Figura 11. Elemento pórtico plano con desplazamientos locales.

Adaptado de “Structural Analysis 8th edition”, por R.C. Hibbeler, 2009.

Para el caso de la rotación, el eje de rotación local coincide con el eje de rotación global ya que pertenecen al mismo plano, esto se traduce de acuerdo a la expresión 2.36

$$d_{Nz'} = D_{Nz} \quad 2.36$$

Al igual que en el caso de las armaduras, si imponemos los desplazamientos globales en la dirección x, en la dirección y, y una rotación en el extremo lejano del miembro, las ecuaciones de transformación resultantes se expresan en 2.37, 2.38, 2.39, 2.40 y 2.41.

$$d_{Fx'} = D_{Fx} \cos \theta_x \quad 2.37$$

$$d_{Fy'} = D_{Fy} \cos \theta_y \quad 2.38$$

$$d_{Fz'} = D_{Fz} \quad 2.39$$

$$d_{Fy'} = -D_{Fx} \cos \theta_y \quad 2.40$$

$$d_{Fx'} = D_{Fy} \cos \theta_x \quad 2.41$$

Si hacemos el siguiente cambio de variable $\lambda_x = \cos \theta_x$ y $\lambda_y = \cos \theta_y$ logramos obtener la ecuación matricial para transformación de desplazamientos.

$$\begin{bmatrix} d_{Nx'} \\ d_{Ny'} \\ d_{Nz'} \\ d_{Fx'} \\ d_{Fy'} \\ d_{Fz'} \end{bmatrix} = \begin{bmatrix} \lambda_x & \lambda_y & 0 & 0 & 0 & 0 \\ -\lambda_y & \lambda_x & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_x & \lambda_y & 0 \\ 0 & 0 & 0 & -\lambda_y & \lambda_x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} D_{Nx} \\ D_{Ny} \\ D_{Nz} \\ D_{Fx} \\ D_{Fy} \\ D_{Fz} \end{bmatrix} \quad 2.42$$

Matriz de transformación de fuerzas

Para la transformación de fuerzas en coordenadas locales a fuerzas en coordenadas globales, basta solo con utilizar la traspuesta de la matriz de transformación de los desplazamientos. Con ello obtenemos la ecuación 2.43

$$\begin{bmatrix} Q_{Nx} \\ Q_{Ny} \\ Q_{Nz} \\ Q_{Fx} \\ Q_{Fy} \\ Q_{Fz} \end{bmatrix} = \begin{bmatrix} \lambda_x & -\lambda_y & 0 & 0 & 0 & 0 \\ \lambda_y & \lambda_x & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_x & -\lambda_y & 0 \\ 0 & 0 & 0 & \lambda_y & \lambda_x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_{Nx'} \\ q_{Ny'} \\ q_{Nz'} \\ q_{Fx'} \\ q_{Fy'} \\ q_{Fz'} \end{bmatrix} \quad 2.43$$

2.6 Matriz de rigidez global del elemento

Ahora se combinarán los resultados de las secciones anteriores y se determinara la matriz de rigidez de un elemento que relaciona los componentes de la fuerza global Q del elemento con sus desplazamientos globales D. Si se sustituye la ecuación 2.21 en la ecuación 2.9, es posible determinar las fuerzas q de los elementos en función de los desplazamientos globales D en sus puntos extremos, y se puede expresar de acuerdo a la ecuación 2.44.

$$\mathbf{q} = \mathbf{k}' \mathbf{T} \mathbf{D} \quad 2.44$$

Al sustituir esta ecuación en la ecuación ($\mathbf{Q} = \mathbf{T}^T \mathbf{q}$), se obtiene la expresión 2.44.

$$\mathbf{Q} = \mathbf{T}^T \mathbf{k}' \mathbf{T} \mathbf{D} \quad 2.45$$

O bien

$$\mathbf{Q} = \mathbf{k} \mathbf{D} \quad 2.46$$

Donde k se expresa en la ecuación 2.47

$$\mathbf{k} = \mathbf{T}^T \mathbf{k}' \mathbf{T} \quad 2.47$$

La matriz **k** es la matriz de rigidez del elemento en coordenadas globales. Dado que **T^T**, **T** y **k'** ya se conocen, como se expresa en 2.48.

$$\mathbf{k} = \begin{bmatrix} \lambda_x & 0 \\ \lambda_y & 0 \\ 0 & \lambda_x \\ 0 & \lambda_y \end{bmatrix} \frac{AE}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_x & \lambda_y & 0 & 0 \\ 0 & 0 & \lambda_x & \lambda_y \end{bmatrix} \quad 2.48$$

(Para armaduras)

$$\mathbf{k} = \begin{bmatrix} \lambda_x & \lambda_y & 0 & 0 & 0 & 0 \\ -\lambda_y & \lambda_x & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_x & \lambda_y & 0 \\ 0 & 0 & 0 & -\lambda_y & \lambda_x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T * \begin{bmatrix} \frac{AE}{L} & 0 & 0 & -\frac{AE}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{AE}{L} & 0 & 0 & \frac{AE}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} * \begin{bmatrix} \lambda_x & \lambda_y & 0 & 0 & 0 & 0 \\ -\lambda_y & \lambda_x & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_x & \lambda_y & 0 \\ 0 & 0 & 0 & -\lambda_y & \lambda_x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad 2.49$$

(Para pórticos planos)

Una vez realizada la multiplicación matricial se obtendrá, en el caso de armaduras, una matriz de 4x4 y para el caso de pórticos planos, una matriz de 6x6.

La ubicación de cada elemento de estas matrices obtenidas está referenciada con cada grado de libertad global asociado con el extremo cercano N, seguido por el extremo lejano F. Esto se indica mediante la notación del número de código a lo largo de las filas y columnas, es decir, N_x, N_y, F_x, F_y (para armaduras) y N_x, N_y, M_x, F_x, F_y, M_y (para pórticos planos). Aquí **k** representa las relaciones de fuerza-desplazamiento para el elemento cuando las componentes de la fuerza y el desplazamiento en los extremos del elemento están en las direcciones globales “x y”, Por lo tanto, cada uno de los términos de la matriz es un coeficiente de influencia de la rigidez **k_{ij}**, lo que denota la componente de fuerza x, y ó z en i, necesaria para originar una componente unitaria de desplazamiento asociada x, y ó z. Como resultado, cada columna identificada de la matriz representa las componentes de fuerza desarrolladas en los extremos del elemento cuando el extremo identificado se somete a un desplazamiento unitario, el cuál creará las componentes de fuerza sobre el elemento que se muestran en la primera columna de la matriz

2.7 Matriz de rigidez global de la estructura

Una vez que se forman todas las matrices de rigidez de los elementos en coordenadas globales, es necesario ensamblarlas en el orden correcto para que se pueda determinar la matriz de rigidez K de toda la estructura. Este proceso de combinación de matrices de los elementos depende de una cuidadosa identificación de los miembros de cada matriz del elemento. Como se analizó en la sección anterior esto se hace por la designación de filas y columnas de la matriz mediante los cuatro (o seis) números de código N_x, N_y, F_x, F_y ó $N_x, N_y, M_x, F_x, F_y, M_y$ que se utilizan para identificar los dos grados de libertad globales que pueden ocurrir en cada extremo del elemento. La matriz de rigidez de la estructura tendrá entonces un orden que será igual al número de código mayor asignado a la armadura, ya que representa la cantidad total de grados de libertad para la estructura. Cuando se ensamblen las matrices k , cada elemento en k se pondrá entonces en su misma designación de fila y columna en la matriz de rigidez de la estructura K . En particular, cuando dos o más elementos están conectados a la misma junta o nodo, entonces algunos de los elementos de cada matriz k del elemento se asignarán a la misma posición en la matriz K . Cuando esto ocurre, los elementos asignados a la ubicación común deben sumarse algebraicamente. La razón de esto se hace evidente si se tiene en cuenta que cada elemento de la matriz k representa la resistencia del elemento a una fuerza aplicada en su extremo. De esta manera, la suma de estas resistencias en la dirección x , y o z al formar la matriz K determina la resistencia total de cada junta a un desplazamiento unitario en la dirección x , y o z .

Este método de ensamble de las matrices de los elementos para formar la matriz de rigidez de las estructuras es tedioso para realizar de manera manual, resulta más fácil y con resultados precisos como confiables, al ser programados en una computadora.

Dicho esto, en el siguiente capítulo se mostrará cómo se soluciona estos problemas mediante el presente modelo computacional.

Una vez obtenida la matriz de rigidez de la estructura, las componentes de fuerza global F que actúan sobre la **armadura o cercha** pueden relacionarse con los desplazamientos globales D utilizando la relación mencionada en 2.50.

$$F=KD \qquad 2.50$$

Para el caso de los pórticos planos, no solo pueden actuar sobre estos, fuerzas puntuales en los nodos, sino que también se pueden ver afectados por cargas puntuales a los largos de sus

miembros, así como cargas distribuidas a lo largo de los mismos. Para proceder con el análisis, es necesario llevar el efecto de estas cargas a los extremos del elemento al cual afectan, generando en él las conocidas fuerzas de empotramiento perfecto. A continuación, se muestra las fuerzas de empotramiento perfecto generado por cada tipo de carga en la tabla 1.

Tabla 1 Fuerzas de empotramiento perfecto.

Elaborado por R.C. Hibbeler, 2009.

	Loading	Equations for Fixed-End Moments, Shears, and Axial Forces
1.		$FS_b = \frac{Wl_2^2}{L^3}(3l_1 + l_2)$ $FM_b = \frac{Wl_1l_2^2}{L^2}$ $FS_c = \frac{Wl_1^2}{L^3}(l_1 + 3l_2)$ $FM_c = -\frac{Wl_1^2l_2}{L^2}$
2.		$FS_b = -\frac{6Ml_1l_2}{L^3}$ $FM_b = \frac{Ml_2}{L^2}(l_2 - 2l_1)$ $FS_c = \frac{6Ml_1l_2}{L^3}$ $FM_c = \frac{Ml_1}{L^2}(l_1 - 2l_2)$
3.		$FS_b = \frac{wL}{2} \left[1 - \frac{l_1}{L^4}(2L^3 - 2l_1^2L + l_1^3) - \frac{l_2^3}{L^4}(2L - l_2) \right]$ $FM_b = \frac{wL^2}{12} \left[1 - \frac{l_1^2}{L^4}(6L^2 - 8l_1L + 3l_1^2) - \frac{l_2^3}{L^4}(4L - 3l_2) \right]$ $FS_c = \frac{wL}{2} \left[1 - \frac{l_1^3}{L^4}(2L - l_1) - \frac{l_2}{L^4}(2L^3 - 2l_2^2L + l_2^3) \right]$ $FM_c = -\frac{wL^2}{12} \left[1 - \frac{l_1^3}{L^4}(4L - 3l_1) - \frac{l_2^2}{L^4}(6L^2 - 8l_2L + 3l_2^2) \right]$

Nº	Carga	Fuerzas de empotramiento perfecto
4.		$FS_b = \frac{w_1(L-l_1)^3}{20L^3} \left\{ (7L+8l_1) - \frac{l_2(3L+2l_1)}{(L-l_1)} \right. \\ \times \left[1 + \frac{l_2}{L-l_1} + \frac{l_2^2}{(L-l_1)^2} \right] + \frac{2l_2^4}{(L-l_1)^3} \left. \right\} \\ + \frac{w_2(L-l_1)^3}{20L^3} \left\{ (3L+2l_1) \left[1 + \frac{l_2}{L-l_1} \right. \right. \\ \left. \left. + \frac{l_2^2}{(L-l_1)^2} \right] - \frac{l_2^3}{(L-l_1)^2} \left[2 + \frac{15L-8l_2}{L-l_1} \right] \right\}$ $FM_b = \frac{w_1(L-l_1)^3}{60L^2} \left\{ 3(L+4l_1) - \frac{l_2(2L+3l_1)}{L-l_1} \right. \\ \times \left[1 + \frac{l_2}{L-l_1} + \frac{l_2^2}{(L-l_1)^2} \right] + \frac{3l_2^4}{(L-l_1)^3} \left. \right\} \\ + \frac{w_2(L-l_1)^3}{60L^2} \left\{ (2L+3l_1) \left[1 + \frac{l_2}{L-l_1} \right. \right. \\ \left. \left. + \frac{l_2^2}{(L-l_1)^2} \right] - \frac{3l_2^3}{(L-l_1)^2} \left[1 + \frac{5L-4l_2}{L-l_1} \right] \right\}$ $FS_c = \left(\frac{w_1+w_2}{2} \right) (L-l_1-l_2) - FS_b$ $FM_c = \frac{L-l_1-l_2}{6} [w_1(-2L+2l_1-l_2) \\ -w_2(L-l_1+2l_2)] + FS_b(L) - FM_b$
5.		$FA_b = \frac{Wl_2}{L}$ $FA_c = \frac{Wl_1}{L}$
6.		$FA_b = \frac{w}{2L} (L-l_1-l_2)(L-l_1+l_2)$ $FA_c = \frac{w}{2L} (L-l_1-l_2)(L+l_1-l_2)$
7.		$FT_b = \frac{M_T l_2}{L}$ $FT_c = \frac{M_T l_1}{L}$

Estas fuerzas de empotramiento perfecto deben ser incluidas en la ecuación 2.50 según se muestra en la ecuación 2.51

$$Q = KD + P \quad 2.51$$

Donde P representa a las fuerzas de empotramiento perfecto en la estructura. Sin embargo, se puede simplificar la expresión anterior como las expresiones 2.52 y 2.53.

$$Q - P = KD \quad 2.52$$

$$F = KD \quad 2.53$$

“F” representa la fuerza actuante en la estructura, con la cual podemos continuar con el análisis estructural.

Esta ecuación es conocida como la ecuación de rigidez de la estructura. Al haber realizado el reacomodo de nuestra matriz de rigidez, colocando en las primeras filas y columnas las etiquetas de los grados de libertad de los elementos no restringidos, nos permite hacer una partición de la anterior ecuación de acuerdo a la expresión 2.54.

$$\begin{bmatrix} Fk \\ Fu \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} Du \\ Dk \end{bmatrix} \quad 2.54$$

Donde:

F_k y D_k : Cargas externas y desplazamientos conocidos, aquí las cargas existen en la armadura como parte del problema y los desplazamientos suelen especificarse como iguales a cero debido a las limitaciones de soportes fijos o móviles.

F_u y D_u : Cargas y desplazamientos desconocidos, aquí las cargas representan las reacciones desconocidas en los soportes y los desplazamientos se presentan en las juntas donde el movimiento no está restringido en una dirección en particular.

K : Matriz de rigidez de la estructura, que se parte para ser compatible con las particiones de Q y D .

Al realizar una expansión de la ecuación 2.54 se obtiene:

$$Fk = K_{11}Du + K_{12}Dk \quad 2.55$$

$$Fu = K_{21}Du + K_{22}Dk \quad 2.56$$

Comúnmente el desplazamiento en los nudos restringidos es cero, es decir $D_k = 0$. Cada que sucede lo anterior mencionado la ecuación 2.55 se puede expresar de acuerdo a la ecuación 2.57.

$$Fk = K_{11}Du \quad 2.57$$

Como los elementos de la matriz partida K_{11} representan la resistencia total en una junta de armadura a un deslizamiento unitario, ya sea en la dirección x o y, entonces la ecuación anterior simboliza la colección de todas las ecuaciones de equilibrio de fuerzas aplicadas a las juntas donde las cargas externas son cero o tienen valor conocido (F_k). Si despejamos D_u obtendremos:

$$Du = [K_{11}]^{-1} Qk \quad 2.58$$

De esta ecuación puede obtenerse una solución directa para todos los desplazamientos de junta desconocidos; entonces a partir de la ecuación 2.56, con $D_k = 0$, se obtiene la ecuación 2.59.

$$Fu = K_{21} Du \quad 2.59$$

De esta se pueden obtener las reacciones desconocidas en los soportes (nodos restringidos). Las fuerzas en los miembros en los miembros pueden determinarse mediante la ecuación 2.60.

$$q = k' TD \quad 2.60$$

3 DESARROLLO DE LA PROGRAMACIÓN

3.1 Lenguaje de programación empleado

El desarrollo de la programación de TrussSolver y PlaneSolver se llevó a cabo utilizando el entorno de desarrollo integrado (IDE), Matlab. Este entorno de programación cuenta con su propio lenguaje de programación, denominado lenguaje M. Este lenguaje es interpretado y puede ejecutarse en la ventana Command Window (para ejecución directa) o a través de la creación de *Scripts* (en la que desarrollan ciertas indicaciones que se requiere que haga el programa). Para dar un panorama del lenguaje de programación usado por MATLAB, este se asemeja en cierta medida a los lenguajes de alto nivel como *BASIC* o *C*. Este software tiene una poderosa función que son, los anteriormente mencionados, *Scripts*; estos permiten agrupar sentencias que se usan con cierta frecuencia dentro de un programa para ser llamados posteriormente.

Además, cabe mencionar que este programa tiene incorporados ciertas expresiones, funciones y comandos que fueron necesarios conocer para iniciar en la programación de este modelo. Las expresiones básicas se muestran en la tabla 2.

Tabla 2 Operadores básicos de MATLAB.

Fuente propia.

Expresión en MATLAB	Operación
+	Suma aritmética
-	Resta aritmética o cambio de signo
*	Multiplicación aritmética
/	División
<	Relación "menor que"
>	Relación "mayor que"
<=	Relación "menor o igual que"
>=	Relación "mayor o igual que"
==	Relación "igual que"
~=	Relación "distinto que"

&	Producto l-ógico (operación "y")
I	Suma lógica (operación "o")
~	Negación (operación "no")

En cuanto a los comandos usados, la tabla 3 muestra una lista con algunos de los más importantes usados en la programación del modelo realizado.

Tabla 3 Lista de comandos.

Fuente propia.

Comando	Descripción
clear all	Borra todas las variables del espacio de trabajo, pero no borra lo de las demás ventanas, es decir, no desaparece lo que hay escrito en la ventana de comandos.
clc	Borra lo que hay en la ventana de comandos pero no borra las variables de la memoria del espacio de trabajo.
clf	Borra la figura.
delete(1)	Borra la variable.
file_name=	Ingresa nombre a la variable.
exist(	Busca si la variable existe.
S=strcat(	Extrae datos de una matriz.
system(for	Busca en todas las ventanas el comando for.
for	Condiciona haciendo repeticiones hasta el límite que se le ingrese a la variable.
end	Termina condiciones.
if	Condiciona para un los casos descritos.
cos	Coseno.
sen	Seno.
else	Caso contrario a la condición dada por cierta en el uso del comando if.

zeros	Genera una matriz de ceros.
save	Guarda los datos de la variable.
fprintf	Da como resultado.
plot	Dibuja.
hold	Mantiene el valor de la variable.
set	Configura una matriz con los valores descritos.
line(a,b)	Dibuja una línea con las coordenadas a,b.
Color	Selecciona un color
LineStyle	Selecciona el tipo de línea
mean	Medida de los valores de la matriz
eval	Evalua en la matriz
elseif	Da opción de agregarle otro caso if a la condición inicial
sortrows	Ordena las filas de una matriz

En suma, la ventaja de este software es que permite, entre otros, realizar operaciones con matrices y vectores, que, para el análisis matricial de estructuras, son fundamentales.

3.2 Desarrollo del modelo matemático

El programa o modelo computacional, fue desarrollado con la finalidad de servir como herramienta de estudio para los alumnos de ingeniería civil que cursan el curso de análisis estructural. Además, de ser un sólido precedente para la investigación en software de programación en nuestra casa de estudios.

Resuelve estructuras del tipo pórtico plano y armadura plana.

Todo el desarrollo del programa se encuentra esquematizado en los flujogramas presentados en el siguiente punto.

3.2.1 Flujogramas

El modelo realizado sigue, como todo programa, una secuencia lógica para el análisis de estructuras que se le ingresa. Esta secuencia está comprendida por tres módulos, la primera que es la encargada de dar al programa todos los *inputs*¹, una segunda que es la encargada de procesar estos inputs, realizar el análisis y, finalmente el tercer módulo es el encargado de arrojar los *outputs*² del análisis y salida gráfica que refleje estos resultados.

Gráfico 1 Secuencia lógica del programa.

Fuente propia.

Módulo de inputs

Este módulo abarca el ingreso de datos de la estructura que será analizada. Los datos que requiere el programa para su ejecución se ingresarán en la ventana de ingreso de datos, la cual se muestra al ejecutar el programa. Los datos a ingresar son los siguientes:

- Unidades de medida de longitud y fuerza (metros, pulgadas, Newtons, etc.)
- Coordenadas de los nodos de la estructura.
- Nodo y dirección (x, y o z) de las restricciones.
- Módulo de elasticidad de los materiales.
- Áreas de las secciones transversales de las barras que conforman la estructura.
- Número de elementos (barras), nodo de inicio y fin, tipo de material y sección.

¹ Inputs: Datos entrada otorgados por el usuario.

² Outputs: Salida de datos ya procesados.

- Magnitud de las cargas actuantes, nodo de aplicación y módulo de la fuerza (en dirección x, y o z).
- Cargas actuantes a lo largo de los elementos (distribuida o puntual, esto para el caso de los pórticos).

Módulo de análisis

Este segundo módulo es el encargado de procesar los datos ingresados en el módulo de *inputs*, realizar los cálculos respectivos y arrojar los resultados.

Con los datos ingresados en los pasos previos, el programa crea una matriz que cuenta con las siguientes características:

Matriz miembro:

Nº de miembro	Nodo de inicio del miembro	Nodo de fin del miembro	Valor de Elasticidad	Valor de área de sección	Valor de la inercia
---------------	----------------------------	-------------------------	----------------------	--------------------------	---------------------

Es a partir de estás matriz que crea la numeración de los elementos o barras de la estructura y el sentido que tienen (de izquierda a derecha, de arriba hacia abajo, etc.)

Figura 12. Etiquetado de nodos, miembros y grados de libertad.

Fuente propia.

El programa asigna por defecto los grados de libertad “ $3n-2$ ”, “ $3n-1$ ” y “ n ” al “enésimo” nodo, tal y como se puede observar en el gráfico anterior.

Es a partir de la *matriz miembro* y de las etiquetas de los grados de libertad, que se forman las matrices de rigidez de cada elemento según la ecuación 2.49.

Cada una de estas matrices aporta a la matriz global de la estructura según el índice de su grado de libertad, según se mostró en el capítulo anterior.

Con la matriz global de la estructura creada, se toma de esta solo los aportes de los índices de los grados de libertad libres y se crea la matriz K_f . Es con esta matriz y la matriz de cargas, la cual se crea en el ingreso de datos, que se aplican las ecuaciones 2.57 y 2.58 con las que se obtienen los desplazamientos.

Posteriormente las reacciones se obtendrán con la ecuación 2.59 y las fuerzas internas en los miembros, con la ecuación 2.60.

La lógica más detallada que sigue el programa, en el módulo de análisis, se puede apreciar en los flujogramas desarrollados en los puntos 3.2.1.2 y 3.2.1.5.

Módulo de outputs

En este último módulo es el encargado de tomar los datos analizados en el módulo anterior y presentarlos de manera gráfica. Adicional a esto, muestra en una interfaz, todos los resultados obtenidos en el módulo anterior. En suma, la función principal de este módulo es la de presentar de manera ordenada los resultados al usuario.

3.2.1.1 Flujograma del módulo de inputs de TrussSolver UPC

Gráfico 2 Módulo de inputs TrussSolver 1 de 12.

Fuente propia.

Gráfico 3 Módulo de inputs TrussSolver 2 de 12.

Fuente propia.

Gráfico 4 Módulo de inputs TrussSolver 3 de 12.

Fuente propia.

Gráfico 5 Módulo de inputs TrussSolver 4 de 12.

Fuente propia.

Gráfico 6 Módulo de inputs TrussSolver 5 de 12.

Fuente propia.

Gráfico 7 Módulo de inputs TrussSolver 6 de 12.

Fuente propia.

Gráfico 8 Módulo de inputs TrussSolver 7 de 12.

Fuente propia.

Gráfico 9 Módulo de inputs TrussSolver 8 de 12.

Fuente propia.

Gráfico 10 Módulo de inputs TrussSolver 9 de 12.

Fuente propia.

Gráfico 11 Módulo de inputs TrussSolver 10 de 12.

Fuente propia.

Gráfico 12 Módulo de inputs TrussSolver 11 de 12.

Fuente propia.

Gráfico 13 Módulo de inputs TrussSolver 12 de 12.

Fuente propia.

3.2.1.2 Flujograma del módulo de análisis de TrussSolver UPC

Gráfico 14 Módulo de análisis TrussSolver 1 de 13.

Fuente propia.

Gráfico 15 Módulo de análisis TrussSolver 2 de 13.

Fuente propia.

Gráfico 16 Módulo de análisis TrussSolver 3 de 13

Fuente propia.

Gráfico 17 Módulo de análisis TrussSolver 4 de 13.

Fuente propia.

Gráfico 18 Módulo de análisis TrussSolver 5 de 13.

Fuente propia.

Gráfico 19 Módulo de análisis TrussSolver 6 de 13.

Fuente propia.

Gráfico 20 Módulo de análisis TrussSolver 7 de 13.

Fuente propia.

Gráfico 21 Módulo de análisis TrussSolver 8 de 13.

Fuente propia.

Gráfico 22 Módulo de análisis TrussSolver 9 de 13.

Fuente propia.

Gráfico 23 Módulo de análisis TrussSolver 10 de 13.

Fuente propia.

Gráfico 24 Módulo de análisis TrussSolver 11 de 13.

Fuente propia.

Gráfico 25 Módulo de análisis TrussSolver 12 de 13.

Fuente propia.

Gráfico 26 Módulo de análisis TrussSolver 13 de 13.

Fuente propia.

3.2.1.3 Flujograma del módulo de outputs de TrussSolver UPC

Gráfico 27 Módulo de outputs TrussSolver 1 de 11.

Fuente propia.

Gráfico 30 Módulo de outputs TrussSolver 4 de 11.

Fuente propia.

Gráfico 31 Módulo de outputs TrussSolver 5 de 11.

Fuente propia.

Gráfico 32 Módulo de outputs TrussSolver 6 de 11.

Fuente propia.

Gráfico 33 Módulo de outputs TrussSolver 7 de 11.

Fuente propia.

Gráfico 34 Módulo de outputs TrussSolver 8 de 11.

Fuente propia.

Gráfico 35 Módulo de outputs TrussSolver 9 de 11.

Fuente propia.

Gráfico 36 Módulo de outputs TrussSolver 10 de 11.

Fuente propia.

Gráfico 37 Módulo de outputs TrussSolver 11 de 11.

Fuente propia.

3.2.1.4 Flujograma del módulo de inputs de PlaneSolver UPC

Gráfico 38 Módulo de inputs PlaneSolver 1 de 11.

Fuente propia.

Gráfico 39 Módulo de inputs PlaneSolver 2 de 11.

Fuente propia.

Gráfico 40 Módulo de inputs PlaneSolver 3 de 11.

Fuente propia.

Gráfico 41 Módulo de inputs PlaneSolver 4 de 11.

Fuente propia.

Gráfico 42 Módulo de inputs PlaneSolver 5 de 11.

Fuente propia.

Gráfico 43 Módulo de inputs PlaneSolver 6 de 11.

Fuente propia.

Gráfico 44 Módulo de inputs PlaneSolver 7 de 11.

Fuente propia.

Gráfico 45 Módulo de inputs PlaneSolver 8 de 11.

Fuente propia.

Gráfico 46 Módulo de inputs PlaneSolver 9 de 11.

Fuente propia.

Gráfico 47 Módulo de inputs PlaneSolver 10 de 11.

Fuente propia.

Gráfico 48 Módulo de inputs PlaneSolver 11 de 11.

Fuente propia.

3.2.1.5 Flujograma del módulo de análisis de PlaneSolver UPC

Gráfico 49 Módulo de análisis PlaneSolver 1 de 21.

Fuente propia.

Gráfico 50 Módulo de análisis PlaneSolver 2 de 21.

Fuente propia.

Gráfico 51 Módulo de análisis PlaneSolver 3 de 21.

Fuente propia.

Gráfico 52 Módulo de análisis PlaneSolver 4 de 21.

Fuente propia.

Gráfico 53 Módulo de análisis PlaneSolver 5 de 21.

Fuente propia.

Gráfico 54 Módulo de análisis PlaneSolver 6 de 21.

Fuente propia.

Gráfico 55 Módulo de análisis PlaneSolver 7 de 21.

Fuente propia.

Gráfico 56 Módulo de análisis PlaneSolver 8 de 21.

Fuente propia.

Gráfico 57 Módulo de análisis PlaneSolver 9 de 21.

Fuente propia.

Gráfico 58 Módulo de análisis PlaneSolver 10 de 21.

Fuente propia.

Gráfico 59 Módulo de análisis PlaneSolver 11 de 21.

Fuente propia.

Gráfico 60 Módulo de análisis PlaneSolver 12 de 21.

Fuente propia.

Gráfico 61 Módulo de análisis PlaneSolver 13 de 21.

Fuente propia.

Gráfico 62 Módulo de análisis PlaneSolver 14 de 21.

Fuente propia.

Gráfico 63 Módulo de análisis PlaneSolver 15 de 21.

Fuente propia.

Gráfico 64 Módulo de análisis PlaneSolver 16 de 21.

Fuente propia.

Gráfico 65 Módulo de análisis PlaneSolver 17 de 21.

Fuente propia.

Gráfico 66 Módulo de análisis PlaneSolver 18 de 21.

Fuente propia.

Gráfico 67 Módulo de análisis PlaneSolver 19 de 21.

Fuente propia.

Gráfico 68 Módulo de análisis PlaneSolver 20 de 21.

Fuente propia.

Gráfico 69 Módulo de análisis PlaneSolver 21 de 21.

Fuente propia.

3.2.1.6 Flujograma del módulo de outputs PlaneSolver UPC

Gráfico 70 Módulo de outputs PlaneSolver 1 de 11.

Fuente propia.

Gráfico 71 Módulo de outputs PlaneSolver 2 de 11.

Fuente propia.

Gráfico 72 Módulo de outputs PlaneSolver 3 de 11.

Fuente propia.

Gráfico 73 Módulo de outputs PlaneSolver 4 de 11.

Fuente propia.

Gráfico 74 Módulo de outputs PlaneSolver 5 de 11.

Fuente propia.

Gráfico 75 Módulo de outputs PlaneSolver 6 de 11.

Fuente propia.

Gráfico 76 Módulo de outputs PlaneSolver 7 de 11.

Fuente propia.

Gráfico 77 Módulo de outputs PlaneSolver 8 de 11.

Fuente propia.

Gráfico 78 Módulo de outputs PlaneSolver 9 de 11.

Fuente propia.

Gráfico 79 Módulo de outputs PlaneSolver 10 de 11.

Fuente propia.

Gráfico 80 Módulo de outputs PlaneSolver 11 de 11.

Fuente propia.

3.3 Características del programa

Incluirá características intuitivas para el ingreso de datos y clara identificación de los datos necesarios para los modelos tipo armaduras o tipo pórticos.

Además, brindará gráficos de la estructura deformada permitiendo asignarles un valor de escala.

Figura 13. Interfaz de resultados gráficos.

Fuente propia.

Los resultados serán fiables puesto que el margen de error será inferior al 0.05% con respecto a software comerciales.

Añadiendo valores agregado a la fiabilidad de los resultados, también tendrán rutinas dentro del código que permitan reducir el costo computacional.

Finalmente, el valor añadido de este modelo, que no posee ningún modelo comercial, es que brindará al usuario la posibilidad de mostrar las matrices de rigideces global de la estructura, matriz de grados de libertad libres y la matriz de rigidez global del elemento que el desee, lo cual será de gran ayudará para la verificación de sus cálculos.

Figura 14. Matriz de grados de libertad libres obtenidos con el programa.

Fuente propia.

3.4 Implementación de TrussSolver y PlaneSolver

Para el desarrollo de los programas presentados en esta tesis, TrussSolver y PlaneSolver, buscamos basarnos en la metodología de resolución aprendida a través de los cursos de Análisis Estructural I y Análisis Estructural II. Lo anterior con la finalidad de poder mostrar, dentro de los resultados que brinde el programa, datos que permitan a los estudiantes de nuestra casa de estudios, verificar los resultados que obtengan al realizar sus cálculos dentro de su proceso de aprendizaje.

Para el desarrollo de estos programas se ha construido una interfaz, la cual hace que su uso sea muy intuitivo para los usuarios, por lo que contamos con que los estudiantes no tengan dificultad en adoptarlo como programa de análisis.

Esperamos que, con el tiempo, estos programas sean usados por todos los estudiantes de la facultad de ingeniería civil y que este se convierta en el primer programa desarrollado por estudiantes para beneficio de otros estudiantes.

Figura 15. Interfaz del programa

Fuente propia.

3.4.1 Código de programación

Los códigos de desarrollo de los programas realizados se presentan en los Anexos **1 y 2**.

3.4.2 Análisis de costo computacional

En búsqueda de la eficiencia de los algoritmos presentes en esta tesis, nos basamos en que el éxito de un algoritmo o subrutina no debe de depender de la capacidad del sistema operativo en el que se ejecute. Sino, en disminuir el costo computacional buscando alternativas de “solvers” (manera en que el programa resuelve lo que se le asigna).

En el caso del presente modelo computacional, logramos reducir el costo computacional del programa cambiando el “solver” que invertía la matriz de rigidez para luego multiplicarla por la matriz de fuerzas y de esa manera obtener los desplazamientos.

El comando se ejecutaba de la siguiente manera:

```
%-----VECTOR DE DESPLAZAMIENTO DE LOS NUDOS LIBRE-----

%DNL: Desplazamiento de los nodos libres

DNL=inv(Kf)*VFR';

DNL;
```

Se cambió el *solver* de invertir matrices al *solver* conocido como eliminación de Gauss-Jordan, el cual trabaja con matrices bandas que reducen la matriz de rigidez analizando solo las diagonales evitando analizar todos los ceros que se crean dentro de esta matriz.

El comando se ejecuta de la siguiente manera:

```
%-----VECTOR DE DESPLAZAMIENTO DE LOS NUDOS LIBRE-----

%DNL: Desplazamiento de los nodos libres

DNL=Kf\VFR';

DNL;
```

Una vez implementada esta mejora en el *solver* del modelo computacional se generaron porcentajes de optimización del costo computacional desde 9% hasta 40% .

Se realizó la medición del tiempo en segundo que demora la ejecución de las rutinas en el programa, se dividió el código en tres zonas estratégicas.

Primera zona:

Desde el inicio del código hasta dónde inicia la salida con interfaz para el usuario, en esta zona se encuentran todas las matrices y algoritmos que hacen posibles poder mostrar los resultados en la interfaz para el usuario.

Segunda zona:

Desde el inicio del código hasta el final de la programación de la salida con interfaz para el usuario, es decir, incluye la programación necesaria para que el usuario pueda conocer los resultados del análisis dando clic en los botones de la interfaz para el usuario.

Tercera zona:

Desde el inicio del código hasta el final de la salida gráfica del modelo computacional. Esta zona incluye el código que hace posible obtener una salida gráfica de la estructura en análisis.

Una vez identificadas las zonas se utilizaron las funciones de MatLab conocida como “tic-toc”

tic - Start stopwatch timer: Esta función de MATLAB inicia un stopwatch timer para medir el costo computacional.

toc - Read elapsed time from stopwatch: Esta función de MATLAB lee el tiempo desde que el stopwatch timer empezó por la función tic.

Ejemplo de uso del tic-toc dentro del código del programa:

```
tic

%----VECTOR DE FUERZAS APLICADAS Y CONOCIDAS (ORGANIZADO)-----

%VFR: Vector de Fuerzas Reacomodadas
for i=1:2*Num_nodos-AUX
    VFR(i)=VF(VRM(i));
end
VFR;

%-----VECTOR DE DESPLAZAMIENTO DE LOS NUDOS LIBRE-----

%DNL: Desplazamiento de los nodos libres
DNL=Kf\VFR';
```

DNL;

toc

Dando como resultado lo siguiente:

Elapsed time is 0.007650 seconds.

De la misma manera se realizó el análisis del costo computacional tomando como puntos de evaluación las tres zonas descritas líneas arriba, se analizaron tres estructuras con el solver de invertir matriz “inv(x)” y con el solver de reducción de Gauss “A\B” con la finalidad de mostrar la importancia de la eficiencia de un algoritmo y la optimización de las rutinas. Los resultados del análisis se muestran en la tabla 4.

Tabla 4 Comparativa de costo computacional.

Fuente propia.

COSTO COMPUTACIONAL					
ESTRUCTURA	ESQUEMA	ZONA DEL CÓDIGO EN EVALUACIÓN	COSTO USANDO EL SOLVER "inv(x)"	COSTO USANDO EL SOLVER "A\B"	%COSTO OPTIMIZADO
Modelo 1		Desde el inicio hasta el inicio de la salida interactiva del usuario	Elapsed time is 0.227976 seconds.	Elapsed time is 0.189336 seconds.	17%
		Desde el inicio hasta el fin de la salida interactiva del usuario	Elapsed time is 1.751222 seconds.	Elapsed time is 1.244623 seconds.	29%
		Desde el inicio hasta el fin de la salida grafica	Elapsed time is 2.695107 seconds.	Elapsed time is 2.1757075 seconds.	19%

Modelo 2		Desde el inicio hasta el inicio de la salida interactiva del usuario	Elapsed time is 0.326376 seconds.	Elapsed time is 0.308445 seconds.	5%
		Desde el inicio hasta el fin de la salida interactiva del usuario	Elapsed time is 1.837041 seconds.	Elapsed time is 1.108642 seconds.	40%
		Desde el inicio hasta el fin de la salida grafica	Elapsed time is 3.550535 seconds.	Elapsed time is 2.461288 seconds.	31%
Modelo 3		Desde el inicio hasta el inicio de la salida interactiva del usuario	Elapsed time is 0.348608 seconds.	Elapsed time is 0.315707 seconds.	9%
		Desde el inicio hasta el fin de la salida interactiva del usuario	Elapsed time is 1.853107 seconds.	Elapsed time is 1.257951 seconds.	32%
		Desde el inicio hasta el fin de la salida grafica	Elapsed time is 3.498046 seconds.	Elapsed time is 2.939671 seconds.	16%

4 VERIFICACIÓN DE RESULTADOS

4.1 Aplicación del modelo

Para poner a prueba el modelo realizado, se buscó la verificación de resultados del modelo realizado usando 3 ejemplos:

4.1.1 Ejemplo 1

Figura 16. Ejemplo 1

Fuente propia.

4.1.1.1 Ingreso de datos del ejemplo 1

Figura 17 Enumeración de nodos del ejemplo 1

Fuente propia.

- Nodos

Ingreso de Datos - NODOS

Nodos: 10

x:

y:

nodo	coordenada x	coordenada y
1	0	0
2	0	4.8000
3	3	1.2000
4	3	4.8000
5	6	2.4000
6	6	4.8000

Figura 18 Ingreso de coordenadas x e y de los nodos del ejemplo 1

Fuente propia.

- Restricciones

restricciones

Nodo restringido:

Rx:

Ry:

Nodo restringido	X	Y
1	1	1
2	1	1

Figura 19 Ingreso de restricciones en x e y de los nodos del ejemplo 1

Fuente propia.

- Tipos de material

elasticidad

Elasticidad

Num. Material: 2

Elasticidad:

Material	Elasticidad (E)
1	70000000

Agregar Corregir Actualizar Guardar y Cerrar Regresar

Figura 20 Ingreso del material y módulo de elasticidad del ejemplo 1.

Fuente propia.

- Tipo de sección

area_transversal

Área

Num. Sección: 2

Área:

N°	Área
1	0.0100

Agregar Corregir Actualizar Guardar y Cerrar Regresar

Figura 21 Ingreso de áreas transversales de cada uno de los elementos del ejemplo 1.

Fuente propia.

- Conectividad de miembros

The 'Miembros_asignar' window contains a form for assigning members. On the left, there are input fields for 'Num. Miembro' (16), 'Nodo Inicial', 'Nodo Final', 'N° E', and 'N° Sección'. On the right, a table displays the assigned members:

N° Miembro	Nodo Inicial	Nodo Final	N° Elasticidad	N° Sección
1	1	2	1	
2	3	4	1	
3	5	6	1	
4	7	8	1	
5	2	4	1	
6	4	6	1	

Below the table are buttons: 'Agregar', 'Corregir', 'Actualizar', 'Guardar y Cerrar', and 'Regresar'.

Figura 22 Ingreso de conectividad de los elementos del ejemplo 1

Fuente propia.

- Cargas nodales

The 'ingreso_cargas' window contains a form for entering nodal loads. On the left, there are input fields for 'Nodo Cargado', 'Carga en X', and 'Carga en Y'. On the right, a table displays the entered loads:

Nodo Cargado	X	Y
6	-3	3
9	0	-18

Below the table are buttons: 'Agregar', 'Corregir', 'Actualizar', 'Guardar y Cerrar', and 'Regresar'.

Figura 23 Ingreso cargas nodales del ejemplo 1.

Fuente propia.

4.1.1.2 Resultados obtenidos del ejemplo 1

Figura 24 Resultados obtenidos para el ejemplo 1 con TrussSolver.

Fuente propia.

4.1.1.3 Comparación de resultados del TrussSolver vs. Kassimali del ejemplo 1

- Desplazamientos

Tabla 5 Comparación de desplazamientos del ejemplo 1.

Fuente propia.

Desplazamiento de los nodos						
Nodo	TrussSolver		Kassimali		Variación	
	X	Y	X	Y	X	Y
1	0.0000	0.0000	0.0000	0.0000	0.00%	0.00%
2	0.0000	0.0000	0.0000	0.0000	0.00%	0.00%
3	-1.71E-04	-1.25E-04	-1.71E-04	-1.25E-04	0.00%	0.00%
4	1.69E-04	-1.15E-04	1.69E-04	-1.15E-04	0.00%	0.00%
5	-2.19E-04	-5.73E-04	-2.19E-04	-5.73E-04	0.00%	0.00%
6	3.49E-04	-5.62E-04	3.49E-04	-5.62E-04	0.00%	0.00%

7	-5.76E-05	-1.58E-03	-5.76E-05	-1.58E-03	0.00%	0.00%
8	5.42E-04	-1.58E-03	5.42E-04	-1.58E-03	0.00%	0.00%
9	7.35E-04	-4.16E-03	7.35E-04	-4.16E-03	0.00%	0.00%

- Fuerzas en los apoyos

Tabla 6 Comparación de desplazamientos del ejemplo 1.

Fuente propia.

Fuerzas en los apoyos					
Nodo	TrussSolver		Kassimali		Variación
	Rx	Ry	Rx	Ry	
1	4.22E+01	1.65E+01	4.22E+01	1.65E+01	0.00%
2	-3.83E+01	-1.50E+00	-3.83E+01	-1.50E+00	0.00%

- Fuerzas internas en los elementos

Tabla 7 Comparación de fuerzas internas en los elementos del ejemplo 1.

Fuente propia.

Fuerzas internas			
Elemento	TussSolver	Kassimali	Variación
	Fuerza	Fuerza	
1	0.0000	0.0000	0.00%
2	2.00E+00	2.00E+00	0.00%
3	3.00E+00	3.00E+00	0.00%
4	-2.49E-14	-2.49E-14	0.00%
5	3.95E+01	3.95E+01	0.00%
6	4.20E+01	4.20E+01	0.00%
7	4.50E+01	4.50E+01	0.00%

8	4.50E+01	4.50E+01	0.00%
9	-4.44E+01	-4.44E+01	0.00%
10	-4.58E+01	-4.58E+01	0.00%
11	-4.85E+01	-4.85E+01	0.00%
12	-4.85E+01	-4.85E+01	0.00%
13	-1.95E+00	-1.95E+00	0.00%
14	-3.20E+00	-3.20E+00	0.00%
15	-2.93E-14	-2.93E-14	0.00%

4.1.1.4 Salida gráfica

Figura 25 Salida gráfica TrussSolver para el ejemplo 1.

Fuente propia.

4.1.1.5 Observación de la comparativa del ejemplo 1

A partir de la comparación de datos, se puede observar que la precisión del modelo computacional creado está a la talla del software comercial, obteniéndose resultados sin variación.

4.1.2 Ejemplo 2

Figura 26 Ejemplo 2.

Fuente propia.

4.1.2.1 Ingreso de datos del ejemplo 2

Figura 27 Enumeración de nodos del ejemplo 2.

Fuente propia.

- Nodos

nodo	coordenada x	coordenada y
1	0	0
2	6	0
3	12	0
4	18	0
5	24	0
6	30	0

Figura 28 Ingreso de coordenadas x e y de los nodos del ejemplo 2.

Fuente propia.

- Restricciones

The 'restricciones' window contains the following elements:

- Input Fields:**
 - Nodo restringido: 9
 - Rx: 0
 - Ry: 1
- Table:**

Nodo restringido	X	Y
1	1	1
5	0	1
- Buttons:** Agregar, Correguir, Actualizar, Guardar y Cerrar, and Regresar.

Figura 29 Ingreso de restricciones en x y y de los nodos del ejemplo 2.

Fuente propia.

- Tipo de material

The 'elasticidad' window contains the following elements:

- Input Fields:**
 - Num. Material: 2
 - Elasticidad: (empty)
- Table:**

Material	Elasticidad (E)
1	70000000
- Buttons:** Agregar, Corregir, Actualizar, Guardar y Cerrar, and Regresar.

Figura 30 Ingreso del material y módulo de elasticidad del ejemplo 2.

Fuente propia.

- Tipo de sección

area_transversal

Área

Num. Sección: 2

Área:

N°	Área
1	0.0100

Figura 31 Ingreso de áreas transversales de cada uno de los elementos del ejemplo 2.

Fuente propia.

- Conectividad de miembros

Miembros_asignar

Miembros

Num. Miembro: 29

Nodo Inicial: 13

Nodo Final: 9

N° E: 1

N° Sección: 1

N° Miembro	Nodo Inicial	Nodo Final	N° Elasticidad	N° Sección
1	1	2	1	
2	2	3	1	
3	3	4	1	
4	4	5	1	
5	5	6	1	
6	6	7	1	

Figura 32 Ingreso de conectividad de los elementos del ejemplo 2.

Fuente propia.

- Cargas nodales

ingreso_cargas

Cargas

Nodo Cargado:

Carga en X:

Carga en Y:

Nodo Cargado	X	Y
2	0	-120
3	0	-120
4	0	-120
6	0	-120
7	0	-120
8	0	-120

Figura 33 Ingreso cargas nodales del ejemplo 2.

Fuente propia.

4.1.2.2 Resultados obtenidos del ejemplo 2

programa1

TrussSolverUPC Análisis

	Desplazamiento X	Desplazamiento Y
1	0	0
2	0	0
3	-1.7076e-04	-1.2528e-04
4	1.6929e-04	-1.1500e-04
5	-2.1936e-04	-5.7268e-04
6	3.4929e-04	-5.6239e-04
7	-5.7568e-05	-0.0016
8	5.4214e-04	-0.0016
9	7.3500e-04	-0.0042

Nodo	Reacción en X	Reacción en Y
1	41.2500	16.5000
2	-38.2500	-1.5000

	Fuerzas internas (+): Tracción (-): Compresión
1	0
2	2.0000
3	3.0000
4	0
5	39.5000
6	42
7	45.0000
8	45.0000
9	-44.4276
10	-45.7739
11	-48.4665
12	-48.4665
13	-1.9526
14	-3.2016

UPC
Universidad Peruana
de Ciencias Aplicadas

Figura 34 Resultados obtenidos para el ejemplo 2 con TrussSolver.

Fuente propia.

4.1.2.3 Comparación de resultados del TrussSolver vs. Kassimali del ejemplo 2

- Desplazamientos

Tabla 8 Comparación de desplazamientos del ejemplo 2.

Fuente propia.

Desplazamiento de todos los nodos						
Nodo	TrussSolver		Kassimali		Variación	
	X	Y	X	Y	X	Y
1	0.0000	0.0000	0.0000	0.0000	0.00%	0.00%
2	0.0000	-0.0067	0.0000	-0.0067	0.00%	0.00%
3	0.0000	-0.0099	0.0000	-0.0099	0.00%	0.00%
4	0.0000	-0.0126	0.0000	-0.0126	0.00%	0.00%
5	0.0000	-0.0150	0.0000	-0.0150	0.00%	0.00%
6	0.0000	-0.0126	0.0000	-0.0126	0.00%	0.00%
7	0.0000	-0.0099	0.0000	-0.0099	0.00%	0.00%
8	0.0000	-0.0067	0.0000	-0.0067	0.00%	0.00%
9	0.0000	0.0000	0.0000	0.0000	0.00%	0.00%
10	0.0021	-0.0066	0.0021	-0.0066	0.01%	0.00%
11	0.0024	-0.0095	0.0024	-0.0095	0.02%	0.00%
12	0.0000	-0.0150	0.0000	-0.0150	0.00%	0.00%
13	-0.0024	-0.0095	-0.0024	-0.0095	0.02%	0.00%
14	-0.0021	-0.0066	-0.0021	-0.0066	0.01%	0.00%
15	0.0012	-0.0125	0.0012	-0.0125	-0.04%	0.00%
16	-0.0012	-0.0125	-0.0012	-0.0125	-0.04%	0.00%

- Fuerzas en los apoyos

Tabla 9 Comparación de las fuerzas en los apoyos del ejemplo 2.

Fuente propia.

Vector de fuerzas			
GL	TrussSolver	Kassimali	Variación
	Fuerza	Fuerza	
1	720.0000	720	0.00%
2	420.0000	420	0.00%
17	-720.0000	-720	0.00%
18	420.0000	420	0.00%

- Fuerzas internas en los elementos

Tabla 10 Comparación de las fuerzas internas en los elementos del ejemplo 2.

Fuente propia.

Fuerzas internas			
Elemento	TrussSolver	Kassimali	Variación
	Fuerza	Fuerza	
1	0.0000	0.0000	0.00%
2	0.0000	0.0000	0.00%
3	0.0000	0.0000	0.00%
4	0.0000	0.0000	0.00%
5	0.0000	0.0000	0.00%
6	0.0000	0.0000	0.00%
7	0.0000	0.0000	0.00%
8	0.0000	0.0000	0.00%
9	-833.5466	-833.5466	0.00%

10	-714.4685	-714.4685	0.00%
11	-822.8571	-822.8571	0.00%
12	-822.8571	-822.8571	0.00%
13	-714.4685	-714.4685	0.00%
14	-833.5466	-833.5466	0.00%
15	120.000000	120.000000	0.00%
16	240.000000	240.000000	0.00%
17	120.000000	120.000000	0.00%
18	0.000000	0.000000	0.00%
19	120.000000	120.000000	0.00%
20	240.000000	240.000000	0.00%
21	120.000000	120.000000	0.00%
22	-119.078091	-119.078091	0.00%
23	238.156183	238.156183	0.00%
24	119.078091	119.078091	0.00%
25	-119.078091	-119.078091	0.00%
26	-119.078091	-119.078091	0.00%
27	119.078091	119.078091	0.00%
28	238.156183	238.156183	0.00%
29	-119.078091	-119.078091	0.00%

4.1.2.4 Observación de la comparativa del ejemplo 2

Al igual que en el ejemplo 1, se puede observar que la precisión del modelo computacional creado está a la talla del software comercial, obteniéndose resultados una precisión mínima de 0.02%.

4.1.2.5 Salida gráfica del ejemplo 2

Figura 35 Salida gráfica TrussSolver para el ejemplo 2.

Fuente propia.

- Nodos

Ingreso de Datos - NODOS

Nodos

Nodos:

x:

y:

nodo	coordenada x	coordenada y
1	0	0
2	0	240
3	240	336
4	480	240
5	480	0

Agregar Corregir Actualizar Guardar y Cerrar

UPC
Universidad Peruana
de Ciencias Aplicadas

Figura 38 Ingreso de coordenadas x e y de los nodos del ejemplo 3.

Fuente propia.

- Restricciones

restricciones_Plane

Restricciones

Nodo restringido:

Rx:

Ry:

Rz:

Nodo restringido	X	Y	Giro
1	1	1	1
5	1	1	0

Agregar Corregir Actualizar Guardar y Cerrar

Nota:
Digitar "1", si el nodo está restringido y "0" si el nodo está libre.

Regresar

UPC
Universidad Peruana
de Ciencias Aplicadas

Figura 39 Ingreso de restricciones en x e y de los nodos del ejemplo 3.

Fuente propia.

- Tipo de material

The window titled 'elasticidad_Plane' contains a red-bordered area with the title 'Elasticidad'. Inside, there are two input fields: 'Num. Material' with the value '3' and 'Elasticidad' which is empty. To the right is a table with two columns: 'Material' and 'Elasticidad (E)'. The table contains two rows: Material 1 with Elasticidad 29000, and Material 2 with Elasticidad 10000. Below the table is a red logo and the text 'LIDER' and 'Gestión de Recursos'. At the bottom of the window are five buttons: 'Agregar' (highlighted with a blue border), 'Corregir', 'Actualizar', 'Guardar y Cerrar', and 'Regresar'.

Material	Elasticidad (E)
1	29000
2	10000

Figura 40 Ingreso del material y módulo de elasticidad del ejemplo 3.

Fuente propia.

- Tipo de sección

The window titled 'area_transversal_Plane' contains a red-bordered area with the title 'Área'. Inside, there are two input fields: 'Num. Sección' with the value '3' and 'Área' which is empty. To the right is a table with two columns: 'N°' and 'Área'. The table contains two rows: N° 1 with Área 29.8000, and N° 2 with Área 30.6000. Below the table is a red logo and the text 'LIDER' and 'Gestión de Recursos'. At the bottom of the window are five buttons: 'Agregar' (highlighted with a blue border), 'Corregir', 'Actualizar', 'Guardar y Cerrar', and 'Regresar'.

N°	Área
1	29.8000
2	30.6000

Figura 41 Ingreso de áreas transversales de cada uno de los elementos del ejemplo 3.

Fuente propia.

- Inercia

Inercia

Num. Sección: 3

Inercia:

N°	Inercia
1	2420
2	3100

Buttons: Agregar, Corregir, Actualizar, Guardar y Cerrar, Regresar

Figura 42 Ingreso de inercias de los elementos del ejemplo 3.

Fuente propia.

- Conectividad de miembros

Miembros

Num. Miembro: 5

Nodo Inicial:

Nodo Final:

N° E:

N° Sección:

N° Inercia:

N° Miembro	Nodo Inicial	Nodo Final	N° Elasticidad	N° Sección	N° Inercia
1	1	2	1	1	1
2	2	3	2	2	2
3	4	3	2	2	2
4	5	4	1	1	1

Buttons: Agregar, Corregir, Actualizar, Guardar y Cerrar, Regresar

Figura 43 Ingreso de conectividad de los elementos del ejemplo 3.

Fuente propia.

- Cargas nodales

The window titled "ingreso_cargas_Plane" contains a section labeled "Cargas". On the left, there are input fields for "Nodo Cargado", "Carga en X", "Carga en Y", and "Carga en Z". To the right is a table with the following data:

Nodo Cargado	X	Y	Momento
2	75	0	0

Below the table are buttons: "Agregar" (highlighted with a dashed border), "Corregir", "Actualizar", "Guardar y Cerrar", and "Regresar". A "Nota:" box contains the following text:

El signo de la carga deberá ser digitado en los casilleros correspondiente, considerando negativo si la fuerza está en sentido opuesto al eje correspondiente.
Si no existe ningún nodo cargado, se deberá agregar como mínimo un nodo colocándos cargas de magnitud "0".

Figura 44 Ingreso cargas nodales del ejemplo 3.

Fuente propia.

- Cargas en elementos

The window titled "cargas_miembros_Plane" contains a section labeled "Cargas en los miembros". On the left, there are input fields for "Miembro Cargado", "Tipo de Carga", "Magnitud (+/-)", "Distancia des de el inicio", and "Distancia desde el final". To the right is a table with the following data:

Miembro Cargado	Tipo de Carga	Magnitud	L1	L2
2	3	0.2500	0	0
3	1	-45	129.2440	129.2440
3	2	20	129.2440	129.2440

Below the table are buttons: "Agregar" (highlighted with a dashed border), "Correguir", "Actualizar", "Guardar y Cerrar", and "Regresar". A "Nota:" box contains the following text:

En el casillero de "Tipo de Caga", escribir:
1 Para cargas cortante.
2 Para cargas axial.
3 Para cargas distribuidas.

Figura 45 Ingreso cargas en elementos del ejemplo 3.

Fuente propia.

4.1.3.2 Resultados obtenidos del ejemplo 3

Figura 46 Resultados obtenidos con el modelo para el ejemplo 3.

Fuente propia.

4.1.3.3 Comparación de resultados del TrussSolver vs. Kassimali del ejemplo 3

- Desplazamientos

Tabla 11 Comparación de desplazamientos del ejemplo 3.

Fuente propia.

Desplazamiento de todos los nodos									
Nodo	PlaneSolver			Kassimali			Variación		
	X	Y	Z	X	Y	Z	X	Y	Z
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.00%	0.00%	0.00%
2	3.4472	-0.0092	-0.0195	3.4472	-0.0092	-0.0195	0.00%	0.00%	0.00%
3	3.952	-1.3152	0.0071	3.952	-1.3152	0.0071	0.00%	0.00%	0.00%
4	4.4247	-0.0212	-0.0093	4.4247	-0.0212	-0.0093	0.00%	0.00%	0.00%
5	0.0000	0.0000	-0.023	0.0000	0.0000	-0.023	0.00%	0.00%	0.00%

- Fuerzas en los apoyos

Tabla 12 Comparación de las fuerzas en los apoyos del ejemplo 3.

Fuente propia.

Vector de fuerzas			
GL	PlaneSolver	Kassimali	Variación
	Fuerza	Fuerza	
1	-67.36	-67.36	0
2	33.01	33.01	0
3	13789	13789	0
13	-33.5	-33.5	0
14	76.19	76.19	0

- Fuerzas en los miembros

Tabla 13 Comparación de las fuerzas internas en los elementos del ejemplo 3.

Fuente propia.

Fuerzas internas										
Elemento	Nodo	FrameSolver			Kassimali			Variación		
		X	Y	Z	X	Y	Z	X	Y	Z
1	Inicial	33.0145	67.3554	13789	33.0145	67.3554	13789	0.00%	0.00%	0.00%
	Final	-33.0145	-67.3554	2376.6	-33.0145	-67.3554	2376.6	0.00%	0.00%	0.00%
2	Inicial	19.3592	-4.4969	-3768.6	19.3592	-4.4969	-3768.6	0.00%	0.00%	0.00%
	Final	-19.3592	4.4969	2606.2	-19.3592	4.4969	2606.2	0.00%	0.00%	0.00%
3	Inicial	49.4035	-35.8029	-6586.4	49.4035	-35.8029	-6586.4	0.00%	0.00%	0.00%
	Final	-49.4035	35.8029	-2668.2	-49.4035	35.8029	-2668.2	0.00%	0.00%	0.00%
4	Inicial	76.1948	33.5016	1.82E-12	76.1948	33.5016	1.82E-12	0.00%	0.00%	0.00%
	Final	-76.1948	-33.5016	8040.4	-76.1948	-33.5016	8040.4	0.00%	0.00%	0.00%

4.1.3.4 Observación de la comparativa del ejemplo 3

En este ejemplo realizado para pórticos, se puede observar que la precisión del modelo computacional desarrollado, ya que tuvo una variación del 0.00% en comparación del software modelo.

4.1.3.5 Salida gráfica del ejemplo 5

Figura 47 Salida gráfica TrussSolver para el ejemplo 3.

Fuente propia.

5 CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- El presente modelo computacional brinda salidas interactivas de los resultados de desplazamientos, fuerzas internas y reacciones en estructuras del tipo armadura plana y de pórtico plano. Además, logra que el usuario pueda analizar la salida gráfica de la deformación en la estructura, brindando al estudiante la opción de poder asignar la escala de dibujo de acuerdo a sus necesidades.
- PlaneSolverUPC y TrusSolverUPC, brindan resultados con un margen de error 0.00%, comparado y registrado en la presente tesis, con el software comercial Aslam Kassimali que usa la misma metodología de análisis (matricial). Esto se puede corroborar según los resultados obtenidos en los puntos 4.1.1.3, 4.1.2.3 y 4.1.3.3 de la presente tesis.
- Los estudiantes de cursos de análisis estructural podrán usar el presente modelo computacional como una herramienta fiable, para la comparación y validación de sus cálculos. En adición a esto, la nueva interfaz interactiva de la versión más reciente del programa; le brinda al usuario mayor fluidez en el ingreso de datos y corrección de los mismos.
- Finalmente, la herramienta computacional desarrollada en la presente tesis de grado, se diferencia del resto de programas computacionales comerciales, porque permite visualizar las matrices de rigidez global y de grados de libertad libres del análisis matricial de las estructuras tipo pórticos y armaduras planas.

5.2 Recomendaciones

- Profesionales cuya formación sea la programación, serán capaces de encontrar rutinas más simplificadas con el fin de volver el presente software en un modelo comercial. Sin embargo, la esencia del presente modelo es incentivar a que cualquier futuro tesista pueda comprender, analizar y continuar con mejoras en futuras versiones del modelo.
- La programación de la muestra de diagramas de fuerza axial, cortante y momento de cada elemento, por su complejidad resultaría en una investigación adicional. Por tal motivo, el presente código quedará de uso libre para cualquier tesista que plantee investigar la programación de distintos tipos de diagramas.

- El código de este programa puede nutrirse con más teorías de cálculo para que así se obtenga un programa aún completo para los estudiantes que lo usen.

6 REFERENCIAS

- Abaqus, U. M. (2006). Abaqus.
- Bathe, K. J. (2006). Finite element procedures. Klaus-Jurgen Bathe.
- Char, B. W. (1988). MAPLE reference manual. Watcom Publications.
- Díaz, E. B., Ruiz, M. C., & Suárez, B. (2007). Análisis matricial de estructuras. CIMNE, Centro Internacional de Métodos Numéricos en Ingeniería.
- Ferreira, A. J. (2008). MATLAB codes for finite element analysis: solids and structures (Vol. 157). Springer Science & Business Media.
- Hibbett, Karlsson, & Sorensen. (1998). Abaqus/standard: User's Manual (Vol. 1).
- Hurtado Gómez, J. E. Análisis matricial de estructuras: curso con MATLAB.
- Kassimali, A. (2012). Matrix Analysis of Structures SI Version. Cengage Learning.
- Kwon, Y. W., & Bang, H. (2000). The finite element method using MATLAB. CRC press.
- Manual, Matlab. (1995). the MathWorks. Inc., Natick, MA.
- Mase, G. T., Smelser, R. E., & Mase, G. E. (2009). Continuum mechanics for engineers. CRC press.
- Mcguire, W., Gallagher, R. H., & Ziemian, R. D. (2000). Matrix structural analysis.
- Nakamura, S. (1997). Análisis numérico y visualización gráfica con MATLAB (No. T385. N35 1997.). R. H. R. Gutiérrez (Ed.). Prentice Hall.
- Orces Pareja, E., & Blum Velasco, D. (2009). Desarrollo de un programa didáctico de elementos finitos para el análisis de estructuras reticulares y cuerpos sometidos a esfuerzos planos.
- Przemieniecki, J. S. (2012). Theory of matrix structural analysis. Dover Publications.
- Reddy, J. N. (2005). An introduction to the finite element method (3rd edition). New York: McGraw-Hill.
- Terrence, W., & Zelkowitz, M. (1998). Lenguajes de programación. Diseño e implementación. Pearson Prentice-Hall.

Weaver, W., & Gere, J. M. (2012). *Matrix Analysis Framed Structures*. Springer Science & Business Media.

7 ANEXOS

7.1 Anexo 1: Código de programación – TrussSolver

```
function varargout = TrussSolver_V1(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @TrussSolver_V1_OpeningFcn, ...
                  'gui_OutputFcn',  @TrussSolver_V1_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

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

function TrussSolver_V1_OpeningFcn(hObject, eventdata, handles, varargin)
set(handles.text2, 'visible', 'off');
set(handles.Local, 'visible', 'off');
set(handles.Global, 'visible', 'off');
set(handles.GLF, 'visible', 'off');
handles.output = hObject;
guidata(hObject, handles);

function varargout = TrussSolver_V1_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
a=imread('LOGO-UPC-VERTICAL-ROJO (1).jpg');
image(a)
axis off
axes(handles.axes7);
set(handles.axes7, 'visible', 'off');

function Ingreso_datos_Callback(hObject, eventdata, handles)
delete('k.mat');
delete('Fee.mat');
ingreso_unidades_Truss

function Calcular_Callback(hObject, eventdata, handles)
set(handles.text2, 'visible', 'on');
set(handles.Local, 'visible', 'on');
set(handles.Global, 'visible', 'on');
set(handles.GLF, 'visible', 'on');
handles.output = hObject;
guidata(hObject, handles);

global M MSUP EM CP MPRP1 PJ MPRP COORD

COORD=M(:,2:3);
Num_nodos=M(end,1);
```

```

D=size(MSUP);
Num_soportes=D(1,1);

Num_material= EM(end,1);

Num_areas= CP(end,1);

MPRP=MPRP1(:,2:5);
Num_miembros= MPRP1(end,1);

PJ1=size(PJ);
Num_cargas=PJ1(1,1);

%Escribir el nombre del archivo del que se extraeran los datos:

%file_name=input('INGRESE NOMBRE DEL ARCHIVO: ','s');
%if exist('temp.m')~=0
    %dos('del temp.m');

%end
%S=strcat('copy',{' '},file_name,' temp.m');
%system(char(S),'-echo');
%temp;
%dos('del temp.m');

%alpha= 50;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  INGRESO DE DATOS  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%JOINT COORDINATE MATRIX

%for i=1:Num_nodos
    % COORD(i,:)=eval(strcat('joint', num2str(i)));
%end
%COORD;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%SUPPORT DATA MATRIX

%for i=1:Num_soportes
    %MSUP(i,:)=eval(strcat('support', num2str(i)));
%end
%MSUP;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ELASTIC MODULUS VECTOR

%for i=1:Num_material
    %EM(i,:)=eval(strcat('Material', num2str(i)));
%end
%EM;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% CROSS SECTIONAL PROPERTY VECTOR

%for i=1:Num_areas
    %CP(i,:)=eval(strcat('section', num2str(i)));
%end
%CP;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% MEMBER DATA MATRIX

%for i=1:Num_miembros
    % MPRP(i,:)=eval(strcat('member', num2str(i)));
%end
%MPRP;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% LOAD DATA MATRICES

%for i=1:Num_cargas
    %PJ(i,:)=eval(strcat('load', num2str(i)));
%end
%PJ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  PROGRAMACION  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%global L

Ejes=1:2*Num_nodos;

for i=1:Num_nodos;
    GL(i,1)=2*i-1;
    GL(i,2)=2*i;
end

%Longitud, angulo de inclinación y Matriz de rigidez de cada elemento

K=zeros(2*Num_nodos);

for i=1:Num_miembros

    xi=COORD(MPRP(i,1),1);
    xf=COORD(MPRP(i,2),1);
    yi=COORD(MPRP(i,1),2);
    yf=COORD(MPRP(i,2),2);

    L(i,1)=( (xf-xi)^2+(yf-yi)^2)^0.5;

    c=(xf-xi)/L(i,1);    %c: Coseno
    s=(yf-yi)/L(i,1);    %s: Seno

```



```

%%%%%%%%

E=EM(MPRP(i,3),2);
A=CP(MPRP(i,4),2);

%%%%%%%%
kn=E*A*[1 0 -1 0; 0 0 0 0; -1 0 1 0; 0 0 0 0]/L(i,1);
T=[c,s,0,0;-s,c,0,0;0,0,c,s;0,0,-s,c];
k=T'*kn*T;

Elemento(i,1:2)=GL(MPRP(i,1),:);
Elemento(i,3:4)=GL(MPRP(i,2),:);
%%%%%%%%
K(Elemento(i,:),Elemento(i,:))=K(Elemento(i,:),Elemento(i,:))+k;

end
K;

%REORGANIZACIÓN DE LA MATRIZ

MSUPX=zeros(Num_soportes,3);
for i=1:Num_soportes;
    MSUPX(i,1)=MSUP(i,1);
    MSUPX(i,2)=2*MSUP(i,1)-1;
    MSUPX(i,3)=2*MSUP(i,1);
end

M0=MSUP(:,[2,3]);
Num_restringidos=sum(sum(M0));
GLr=zeros(1, Num_restringidos);

P=1;
for i=1:Num_soportes

    for j=2:3

        if MSUP(i,j)==1

            GLr(1,P)=MSUPX(i,j);

            else P=P-1;

        end
        P=P+1;

    end

end

GLr;

GLt=[1:2*Num_nodos];

GLf=[GLt,GLr];

for i=1:Num_restringidos;

```

```

        [elemento, repitencia]=find(GLf==GLr(i));
        GLf(repitencia)=[];
end

VRM=[GLf, GLr];

%MATRIZ DEL SISTEMA REORGANIZADA

K0=zeros(2*Num_nodos); %K0: Matriz auxiliar

for i=1:2*Num_nodos
    K0(i,:)=K(VRM(i),:);
end
K0;

KG=zeros(2*Num_nodos);
for i=1:2*Num_nodos;
    KG(:,i)=K0(:,VRM(i));
end
KG; %KG: Matriz de rigidez global

%-----MATRIZ DE LOS NUDOS LIBRES Y FUERZAS CONOCIDAS-----

Kf=zeros(2*Num_nodos-Num_restringidos);
for i=1:2*Num_nodos-Num_restringidos
    Kf(i,:)=KG(i,1:2*Num_nodos-Num_restringidos);
end
Kf; %Kf: Matriz de rigidez de los grados de libertad libres
(También conocida conocida como K11)

%----MATRIZ DE LOS DESPLAZAMIENTOS Y REACCIONES EN LOS APOYOS----

% Kr=zeros([Num_restringidos,2*Num_nodos-Num_restringidos]);
% %Kf: Matriz de rigidez de los grados de libertad restringidos (También
conocida conocida como K21)
%
% AUX1=1;
% for i=2*Num_nodos-Num_restringidos+1:2*Num_nodos
%     Kr(AUX1,:)=KG(i,1:2*Num_nodos-Num_restringidos);
%     AUX1=AUX1+1;
% end
% Kr;

%----VECTOR DE FUERZAS APLICADAS Y CONOCIDAS-----
VF=zeros(2*Num_nodos,1);
%VF: Vector de Fuerzas
for i=1:Num_cargas;
    VF(2*PJ(i,1)-1,1)=PJ(i,2);
    VF(2*PJ(i,1),1)=PJ(i,3);
end
VF;

%----VECTOR DE FUERZAS APLICADAS Y CONOCIDAS (ORGANIZADO)-----

%VFR: Vector de Fuerzas Reacomodadas
for i=1:2*Num_nodos-Num_restringidos
    VFR(i)=VF(VRM(i));
end

```

```

VFR;

%-----VECTOR DE DESPLAZAMIENTO DE LOS NUDOS LIBRE-----

%DNL: Desplazamiento de los nodos libres
%DNL=Kf\VFR';
DNL=inv(Kf)*VFR';
DNL;

%----VECTOR DE DESPLAZAMIENTO DE TODOS LOS NUDOS-----

%DTN0: Matriz auxiliar
DTN0=zeros(2*Num_nodos,1);
for i=1:2*Num_nodos-Num_restringidos;
    DTN0(i)=DNL(i);
end
DTN0;

VRMt=VRM';
DTN1=zeros(2*Num_nodos, 2);
DTN1=[VRM' DTN0];

DTN2=sortrows(DTN1,1);

DTN=DTN2(:,2);

%DTN: Desplazamiento de todos los nodos
%DTN=zeros(2*Num_nodos,1);
for i=1:2*Num_nodos
    %DTN(i)=DTN0(VRM(i));
end
%DTN;

%GUARDANDO VALOR DE DTN
%save DTN
%global DTN
%-----

fprintf('\n Desplazamiento de todos los nudos \n\n');

%DTNf=zeros(Num_nodos,2);
for i=1:Num_nodos
    %DTNf(i,1)=i;
    fprintf('%d      %f      %f\n',i,DTN(2*i-1),DTN(2*i))
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%-----VECTOR DE LAS REACCIONES EN LOS APOYOS-----

KGR=zeros(2*Num_nodos);

for i=1:2*Num_nodos
    KGR(i,:)=K(VRM(i),:);
end

```

```

VFA0=KGR(2*Num_nodos-Num_restringidos+1:2*Num_nodos,:)*DTN;

% fprintf('\n Vector de fuerzas en los apoyos \n\n');
% %VFA: Vector de Fuerzas en los apoyos
% VFA0=Kr*DNL;
% VFA0;

M0=MSUP(:, [2,3]);
    Num_restringidos=sum(sum(M0));
    AUX3=zeros(1, Num_restringidos); %AUX3: Matriz auxiliar de grados de
    libertad

P=1;
    for i=1:Num_soportes;
        for j=2:3

            if MSUP(i,j)==1

                AUX3(1,P)=MSUPX(i,j);

            end
            P=P+1;

        end

    end
    AUX3;

VFA=zeros(2*Num_soportes,1);
GLrt=GLr.';
AUX4=AUX3.';
Q=1;
% for i=1:2*Num_soportes;
for i=1:2*Num_soportes;

    %if Q<=size(GLrt)

        if AUX4(i,1)==GLrt(Q,1);

            VFA(i,1)=VFA0(Q,1);
            Q=Q+1;

        else
            VFA(i,1)=0;
            %Q=i;
        end

    %else
        %VFA(i,1)=0;
        %Q=i;

    end

    fprintf('%f\n',VFA)
    %GUARDANDO EL VALOR DE VFA
    %save VFA
    %global VFA

```

```

%FIN DEL GUARDADO

%----VECTOR DE FUERZAS INTERNAS DE LOS ELEMENTOS-----

d=zeros(4,1);
FuerzasInternas=zeros(Num_miembros,2);
Esfuerzos=zeros(Num_miembros,2);

fprintf('\n Vector de fuerzas internas en los elementos (+):Tracción (-) :Compresión \n\n');

for i=1:Num_miembros;
    ElementoAux=Elemento(i,:);
    for x=1:4
        d(x)=DTN(ElementoAux(x),1);
    end
    d;

    E=EM(MPRP(i,3),2);
    A=CP(MPRP(i,4),2);
    xi=COORD(MPRP(i,1),1);
    xf=COORD(MPRP(i,2),1);
    yi=COORD(MPRP(i,1),2);
    yf=COORD(MPRP(i,2),2);
    c=(xf-xi)/L(i,1);
    s=(yf-yi)/L(i,1);

    Fe=A*E/L(i,1)*[-c -s c s]*d;

    FuerzasInternas(i,1)=i;
    FuerzasInternas(i,2)=Fe;

    Esfuerzos(i,1)=i;
    Esfuerzos(i,2)=Fe/A;

    fprintf('%d      %f\n',i,Fe)
    Fee(i,1)=Fe;
end
Fee;
%GUARDANDO FEE
save Fee
%global Fee

%FIN DEL GUARDADO
%GUI SALIDA CON INTERFAZ PARA USUARIO DE DESPLAZAMIENTOS
programa1
%FIN DE SALIDA GRAFICA DE INTERFAZ DE USUARIO

SalidaTxtArmadura;

%%%%SALIDA GRAFICA DE DESPLAZAMIENTOS%%%

% --- Executes on button press in Graficar.
function Graficar_Callback(hObject, eventdata, handles)
% hObject      handle to Graficar (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PROGRAMACIÓN GRAFICA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cla
global alpha

load ('Fee.mat', 'L', 'DTN')
global M MSUP EM CP MPRP1 PJ

COORD=M(:,2:3);
Num_nodos=M(end,1);

D=size(MSUP);
Num_soportes=D(1,1);

Num_material= EM(end,1);

Num_areas= CP(end,1);

MPRP=MPRP1(:,2:5);
Num_miembros= MPRP1(end,1);

PJ1=size(PJ);
Num_cargas=PJ1(1,1);

%GRAFICA DE LOS NUDOS

for i=1:Num_nodos;
    x=COORD(i,1);
    y=COORD(i,2);

    plot(x,y,'o')
    axis('equal')
    hold on
end

%GRAFICA DE LOS ELEMENTOS

for i=1:Num_miembros
    a=[ (COORD(MPRP(i,1),1)), (COORD(MPRP(i,2),1)) ];
    b=[ (COORD(MPRP(i,1),2)), (COORD(MPRP(i,2),2)) ];

    line(a,b)
    hold on
end

NodoDesplazado=zeros(Num_nodos,2);
for i=1:Num_nodos;
    NodoDesplazado(i,1)=COORD(i,1)+alpha*DTN(2*i-1);
    NodoDesplazado(i,2)=COORD(i,2)+alpha*DTN(2*i);
end
NodoDesplazado;

%GRAFICA DE LOS NUDOS DESPLAZADOS

```

```

for i=1:Num_nodos;
    x=NodeDesplazado(i,1);
    y=NodeDesplazado(i,2);

    plot(x,y,'s')
    hold on
end

%GRAFICA DE LOS ELEMENTOS DESPLAZADOS

for i=1:Num_miembros
    a=[(NodeDesplazado(MPRP(i,1),1)),(NodeDesplazado(MPRP(i,2),1))];
    b=[(NodeDesplazado(MPRP(i,1),2)),(NodeDesplazado(MPRP(i,2),2))];

    set(line(a,b),'Color','r','LineStyle','--');
    hold on
end

%GRAFICO DE LOS APOYOS

for i=1:Num_soportes;
    %Apoyo Fijo
    if MSUP(i,2)==1 && MSUP(i,3)==1

        n=3;
        R=mean(L)*0.1;
        tita=[0:(2*pi/n):2*pi]+pi/2;
        Xapoyo=R*cos(tita)+COORD(MSUP(i,1),1);
        Yapoyo=R*sin(tita)-R+COORD(MSUP(i,1),2);

        plot(Xapoyo,Yapoyo)

    else
        %Apoyo Movil
        R = mean(L)*0.07;
        tita = (0:0.01:2.01*pi);
        Xapoyo= R*cos(tita)+COORD(MSUP(i,1),1);
        Yapoyo= R*sin(tita)-R+COORD(MSUP(i,1),2);
        plot(Xapoyo,Yapoyo)

    end

end

%Dibujo de los vectores de fuerza

for i=1:Num_cargas;
    quiver(COORD(PJ(i,1),1),COORD(PJ(i,1),2),PJ(i,2)*0.02,PJ(i,3)*0.02)
    hold on
end

% --- Executes on button press in Verificar.
function Verificar_Callback(hObject, eventdata, handles)
    ingreso_cargas

```

```

% --- Executes on button press in Local.
function Local_Callback(hObject, eventdata, handles)
matriz_elemento

% --- Executes on button press in Global.
function Global_Callback(hObject, eventdata, handles)
kg

% --- Executes on button press in GLF.
function GLF_Callback(hObject, eventdata, handles)
kf

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
XP=get(handles.popupmenu1,'Value');
global alpha
switch XP
    case 1
        axes(handles.axes7);
        delete axes7
        alpha=1

    case 2

        alpha=10

    case 3

        alpha=20

    case 4

        alpha=50

    case 5

        alpha=100

    case 6

        alpha=200

end

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to selector (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

```



```
% Hint: popupmenu controls usually have a white background on Windows.  
%     See ISPC and COMPUTER.  
if ispc && isequal(get(hObject,'BackgroundColor'),  
get(0,'defaultUicontrolBackgroundColor'))  
    set(hObject,'BackgroundColor','white');  
end
```

7.2 Anexo 2: Código de programación – PlaneSolver

```
function varargout = PlaneSolver_V1(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @PlaneSolver_V1_OpeningFcn, ...
                  'gui_OutputFcn',  @PlaneSolver_V1_OutputFcn, ...
                  'gui_LayoutFcn',   [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

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

function PlaneSolver_V1_OpeningFcn(hObject, eventdata, handles, varargin)
set(handles.text2,'visible','off');
set(handles.KG,'visible','off');
set(handles.GLF,'visible','off');
%set(handles.GLR,'visible','off');
set(handles.local,'visible','off');
handles.output = hObject;
guidata(hObject, handles);

function varargout = PlaneSolver_V1_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
a=imread('LOGO-UPC-VERTICAL-ROJO (1).jpg');
image(a)
axis off
axes(handles.axes7);
set(handles.axes7,'visible','off');

function Ingreso_datos_Callback(hObject, eventdata, handles)
delete('VFA.mat');
delete('k.mat');
ingreso_unidades_Plane

function Calcular_Callback(hObject, eventdata, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  INGRESO DE DATOS  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global M MSUP EM CP IV MPRP1 PJ MP

COORD=M(:,2:3);
Num_nodos=M(end,1);

D=size(MSUP);
Num_soportes=D(1,1);
```

```

Num_material= EM(end,1);

Num_areas= CP(end,1);

Num_inercia= IV(end,1);

MPRP=MPRP1(:,2:6);
Num_miembros= MPRP1(end,1);

PJ1=size(PJ);
Num_cargas=PJ1(1,1);

Num_tiposdecargas=size(MP,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  PROGRAMACION  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Ejes=1:3*Num_nodos;
for i=1:Num_nodos;
    GLx=3*i-2;
    GLy=3*i-1;
    GLz=3*i;
    GL(i,1)=GLx;
    GL(i,2)=GLy;
    GL(i,3)=GLz;
end

%Longitud, angulo de inclinación y Matriz de rigidez de cada elemento

K=zeros(3*Num_nodos);

for i=1:Num_miembros

    xi=COORD(MPRP(i,1),1);
    xf=COORD(MPRP(i,2),1);
    yi=COORD(MPRP(i,1),2);
    yf=COORD(MPRP(i,2),2);

    L(i,1)=((xf-xi)^2+(yf-yi)^2)^0.5;

    c=(xf-xi)/L(i,1);    %c: Coseno
    s=(yf-yi)/L(i,1);    %s: Seno

    %%%%%

    E=EM(MPRP(i,3),2);
    A=CP(MPRP(i,4),2);
    I=IV(MPRP(i,5),2);

    %%%%%

    kn=[A*E/L(i,1) 0 0 -A*E/L(i,1) 0 0; 0 12*E*I/L(i,1)^3 6*E*I/L(i,1)^2
0 -12*E*I/L(i,1)^3 6*E*I/L(i,1)^2; 0 6*E*I/L(i,1)^2 4*E*I/L(i,1) 0 -
6*E*I/L(i,1)^2 2*E*I/L(i,1); -A*E/L(i,1) 0 0 A*E/L(i,1) 0 0; 0 -
12*E*I/L(i,1)^3 -6*E*I/L(i,1)^2 0 12*E*I/L(i,1)^3 -6*E*I/L(i,1)^2; 0
6*E*I/L(i,1)^2 2*E*I/L(i,1) 0 -6*E*I/L(i,1)^2 4*E*I/L(i,1)];

```

```

T=[c s 0 0 0 0; -s c 0 0 0 0; 0 0 1 0 0 0; 0 0 0 c s 0; 0 0 0 -s c 0;
0 0 0 0 0 1]; %%Matriz de transformación%%
k=T'*kn*T; %(6.29) Kassimali

Elemento(i,1:3)=GL(MPRP(i,1),:);
Elemento(i,4:6)=GL(MPRP(i,2),:);

K(Elemento(i,:),Elemento(i,:))=K(Elemento(i,:),Elemento(i,:))+k;

end
K;

%REORGANIZACIÓN DE LA MATRIZ

MSUPX=zeros(Num_soportes,4);
for i=1:Num_soportes;
    MSUPX(i,1)=MSUP(i,1);
    MSUPX(i,2)=3*MSUP(i,1)-2;
    MSUPX(i,3)=3*MSUP(i,1)-1;
    MSUPX(i,4)=3*MSUP(i,1);
end

M0=MSUP(:, [2:4]);
Num_restringidos=sum(sum(M0));
GLraux=zeros(1, 3*Num_nodos);

P=1;
for i=1:Num_soportes
    for j=2:4

        if MSUP(i,j)==1

            GLraux(1,P)=MSUPX(i,j);

            end
            P=P+1;

        end

    end

    end

    GLraux;
    GLraux2=unique(GLraux);
    GLr=GLraux2(1,2:(Num_restringidos+1));

    GLt=[1:3*Num_nodos];

    GLf=[GLt, GLr];

    for i=1:Num_restringidos;
        [elemento, repitencia]=find(GLf==GLr(i));
        GLf(repitencia)=[];
    end

    VRM=[GLf, GLr];

    %MATRIZ DEL SISTEMA REORGANIZADA

```

```

K0=zeros(3*Num_nodos); %K0: Matriz auxiliar

for i=1:3*Num_nodos
    K0(i,:)=K(VRM(i),:);
end
K0;

KG=zeros(3*Num_nodos);
for i=1:3*Num_nodos;
    KG(:,i)=K0(:,VRM(i));
end
KG; %KG: Matriz de rigidez global

%FUERZAS DE EMPOTRAMIENTO PERFECTO

FEP=zeros(Num_tiposdecargas,7);

for i=1:Num_tiposdecargas;

%     if MP(i,2)==1
%
%         Fxi=0;
%         Fyi=(MP(i,3)*MP(i,5)^2/L(MP(i,1))^3)*(3*MP(i,4)+MP(i,5));
%         Mi=MP(i,3)*MP(i,4)^2*MP(i,5)/L(MP(i,1))^2;
%         Fxf=0;
%         Fyf=(MP(i,3)*MP(i,4)^2/L(MP(i,1))^3)*(MP(i,4)+3*MP(i,5));
%         Mf=-MP(i,3)*MP(i,4)^2*MP(i,5)/L(MP(i,1))^2;
%
%     elseif MP(i,2)==2
%
%         Fxi=MP(i,3)*MP(i,5)/L(MP(i,1));
%         Fyi=0;
%         Mi=0;
%         Fxf=MP(i,3)*MP(i,4)/L(MP(i,1));
%         Fyf=0;
%         Mf=0;
%
%     elseif MP(i,2)==3
%
%         Fxi=0;
%         Fyi=(MP(i,3)*L(MP(i,1))/2)*(1-MP(i,4)*(2*L(MP(i,1))^3-
% 2*MP(i,4)^2*L(MP(i,1))+MP(i,4)^3)/L(MP(i,1))^4-MP(i,5)^3*(2*L(MP(i,1))-
% MP(i,5))/L(MP(i,1))^4);
%         Mi=MP(i,3)*L(MP(i,1))^2*(1-MP(i,4)^2*(6*L(MP(i,1))^2-
% 8*MP(i,4)*L(MP(i,1))+3*MP(i,4)^2)/L(MP(i,1))^4-MP(i,5)^3*(4*L(MP(i,1))-
% 3*MP(i,5))/L(MP(i,1))^4)/12;
%         Fxf=0;
%         Fyf=(MP(i,3)*L(MP(i,1))/2)*(1-MP(i,4)^3*(2*L(MP(i,1))-
% MP(i,4))/L(MP(i,1))^4-MP(i,5)*(2*L(MP(i,1))^3-
% 2*MP(i,5)^2*L(MP(i,1))+MP(i,5)^3)/L(MP(i,1))^4);
%         Mf=-MP(i,3)*L(MP(i,1))^2*(1-MP(i,4)^3*(4*L(MP(i,1))-
% 3*MP(i,4))/L(MP(i,1))^4-MP(i,5)^2*(6*L(MP(i,1))^2-
% 8*MP(i,5)*L(MP(i,1))+3*MP(i,5)^2)/L(MP(i,1))^4)/12;
%
%     end

```

```

if MP(i,2)==1

    Fxi=0;
    Fyi=(MP(i,3)*MP(i,5)^2/L(MP(i,1))^3)*(3*MP(i,4)+MP(i,5));
    Mi=MP(i,3)*MP(i,4)^2*MP(i,5)/L(MP(i,1))^2;
    Fxf=0;
    Fyf=(MP(i,3)*MP(i,4)^2/L(MP(i,1))^3)*(MP(i,4)+3*MP(i,5));
    Mf=-MP(i,3)*MP(i,4)^2*MP(i,5)/L(MP(i,1))^2;

elseif MP(i,2)==2

    Fxi=MP(i,3)*MP(i,5)/L(MP(i,1));
    Fyi=0;
    Mi=0;
    Fxf=MP(i,3)*MP(i,4)/L(MP(i,1));
    Fyf=0;
    Mf=0;

elseif MP(i,2)==3

    Fxi=0;
    Fyi=(MP(i,3)*L(MP(i,1))/2)*(1-MP(i,4)*(2*L(MP(i,1))^3-
2*MP(i,4)^2*L(MP(i,1))+MP(i,4)^3)/L(MP(i,1))^4-MP(i,5)^3*(2*L(MP(i,1))-
MP(i,5))/L(MP(i,1))^4);
    Mi=MP(i,3)*L(MP(i,1))^2*(1-MP(i,4)^2*(6*L(MP(i,1))^2-
8*MP(i,4)*L(MP(i,1))+3*MP(i,4)^2)/L(MP(i,1))^4-MP(i,5)^3*(4*L(MP(i,1))-
3*MP(i,5))/L(MP(i,1))^4)/12;
    Fxf=0;
    Fyf=(MP(i,3)*L(MP(i,1))/2)*(1-MP(i,4)^3*(2*L(MP(i,1))-
MP(i,4))/L(MP(i,1))^4-MP(i,5)*(2*L(MP(i,1))^3-
2*MP(i,5)^2*L(MP(i,1))+MP(i,5)^3)/L(MP(i,1))^4);
    Mf=-MP(i,3)*L(MP(i,1))^2*(1-MP(i,4)^3*(4*L(MP(i,1))-
3*MP(i,4))/L(MP(i,1))^4-MP(i,5)^2*(6*L(MP(i,1))^2-
8*MP(i,5)*L(MP(i,1))+3*MP(i,5)^2)/L(MP(i,1))^4)/12;

end

FEP(i,1)=MP(i,1);
FEP(i,2)=Fxi;
FEP(i,3)=Fyi;
FEP(i,4)=Mi;
FEP(i,5)=Fxf;
FEP(i,6)=Fyf;
FEP(i,7)=Mf;
end

FEP; %Cambiar por Q0

P0=zeros(3*Num_nodos,1); %Cambiar por F0

for i=1:Num_tiposdecargas;

xi=COORD(MPRP(FEP(i,1),1),1);
xf=COORD(MPRP(FEP(i,1),2),1);
yi=COORD(MPRP(FEP(i,1),1),2);
yf=COORD(MPRP(FEP(i,1),2),2);

```

```

c=(xf-xi)/L(FEP(i,1),1);      %c: Coseno
s=(yf-yi)/L(FEP(i,1),1);      %s: Seno

T=[c s 0 0 0 0; -s c 0 0 0 0; 0 0 1 0 0 0; 0 0 0 c s 0; 0 0 0 -s c 0; 0 0
0 0 0 1];

Q0=T'*FEP(i,2:7)';

    for n=1:6;
        P0(Elemento(FEP(i,1),n))=Q0(n)+P0(Elemento(FEP(i,1),n));
    end

end

P0i=zeros(3*Num_nodos,1); %P0i: Vector de las fep ordenado

for i=1:3*Num_nodos;
    P0i(i)=P0(VRM(i));
end

P=zeros(3*Num_nodos,1);

for i=1:Num_cargas;
    P(3*PJ(i,1)-2,1)=PJ(i,2);
    P(3*PJ(i,1)-1,1)=PJ(i,3);
    P(3*PJ(i,1),1)=PJ(i,4);
end
P;

Pr=zeros(3*Num_nodos,1); %Pr: El vector de las fuerzas en los nodos
ordenados
for i=1:3*Num_nodos;
    Pr(i)=P(VRM(i)');
end

%%%%Desplazamiento de todos lo nodos%%%%

P1=Pr-P0i;

Kf=KG(1:3*Num_nodos-Num_restringidos,1:3*Num_nodos-Num_restringidos);
P2=P1(1:3*Num_nodos-Num_restringidos,1);
DNL=Kf\P2; %DESPLAZAMIENTO DE LOS NODOS LIBRES

DTN=zeros(3*Num_nodos,1); %DESPLAZAMIENTO DE TODOS LOS NODOS
for i=1:3*Num_nodos-Num_restringidos;
    DTN(i)=DNL(i);
end

R=zeros(3*Num_nodos,2);
VRMt=VRM';
for i=1:3*Num_nodos;
    R(i,1)=VRMt(i);
    R(i,2)=DTN(i);
end

```

```

C=sortrows(R,1);

set(handles.text2,'visible','on');
set(handles.KG,'visible','on');
set(handles.GLF,'visible','on');
%set(handles.GLR,'visible','on');
set(handles.local,'visible','on');

%%% FUERZAS INTERNAS %%%

%%%FEP1 es una matriz auxiliar que le agrega al FEP filas de ceros%%%
FEP1=[FEP; 0 0 0 0 0 0 0; 0 0 0 0 0 0 0];

FEP2=zeros(Num_miembros,7);

for i=1:size(FEP(:,1),1); %%%% dimensión del FEP%%%

    if FEP1(i,1)==FEP1(i+1,1) & FEP2(FEP1(i,1),:)==[0 0 0 0 0 0 0];
        FEP2(FEP1(i,1),:)= [FEP1(i,1) sum(FEP1([i i+1],2:7),1)];
    elseif FEP2(FEP1(i,1),:)==[0 0 0 0 0 0 0]
        FEP2(FEP1(i,1),:)=FEP1(i,:);
    end
    if FEP1(i,1)==FEP1(i+2,1);
        FEP2(FEP1(i,1),:)= [FEP1(i,1) FEP2(FEP1(i,1),2:7)+FEP1(i+2,2:7)];
    end

end

FEP2;

FI=zeros(Num_miembros,6);

for i=1:Num_miembros

    xi=COORD(MPRP(i,1),1);
    xf=COORD(MPRP(i,2),1);
    yi=COORD(MPRP(i,1),2);
    yf=COORD(MPRP(i,2),2);

    L(i,1)=((xf-xi)^2+(yf-yi)^2)^0.5;

    c=(xf-xi)/L(i,1); %c: Coseno
    s=(yf-yi)/L(i,1); %s: Seno

    E=EM(MPRP(i,3),2);
    A=CP(MPRP(i,4),2);
    I=IV(MPRP(i,5),2);

    kn2=[A*E/L(i,1) 0 0 -A*E/L(i,1) 0 0; 0 12*E*I/L(i,1)^3 6*E*I/L(i,1)^2
0 -12*E*I/L(i,1)^3 6*E*I/L(i,1)^2; 0 6*E*I/L(i,1)^2 4*E*I/L(i,1) 0 -
6*E*I/L(i,1)^2 2*E*I/L(i,1); -A*E/L(i,1) 0 0 A*E/L(i,1) 0 0; 0 -
12*E*I/L(i,1)^3 -6*E*I/L(i,1)^2 0 12*E*I/L(i,1)^3 -6*E*I/L(i,1)^2; 0
6*E*I/L(i,1)^2 2*E*I/L(i,1) 0 -6*E*I/L(i,1)^2 4*E*I/L(i,1)];
    T=[c s 0 0 0 0; -s c 0 0 0 0; 0 0 1 0 0 0; 0 0 0 c s 0; 0 0 0 -s c 0;
0 0 0 0 0 1];

```



```

        D=zeros(6,1);
        for y=1:6
            D(y)=C(Elemento(i,y),2);
        end
        Fe=kn2*T*D + FEP2(i,(2:7))';
        FI(i,:)=Fe;
    end

    %%% VECTOR DE FUERZAS EN LOS APOYOS %%%

    Kr=KG((3*Num_nodos-Num_restringidos+1):3*Num_nodos,1:(3*Num_nodos-
    Num_restringidos));

    VFA=KG*DTN-P1;

    save VFA

    %GUI salida con interfaz para usuario

    programalportico

    %Fin del GUI

    %SALIDA EN FORMATO .txt

    SalidaTxtPortico;

    %%%SALIDA GRAFICA DE DESPLAZAMIENTOS%%

    % --- Executes on button press in Graficar.
    function Graficar_Callback(hObject, eventdata, handles)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %PROGRAMACIÓN GRAFICA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    cla
    global alpha

    load ('VFA.mat', 'L', 'DTN','C','VRM','Elemento','PJ');
    global M MSUP EM CP IV MPRP1 PJ MP

    COORD=M(:,2:3);
    Num_nodos=M(end,1);

    D=size(MSUP);
    Num_soportes=D(1,1);

    Num_material= EM(end,1);

    Num_areas= CP(end,1);

    MPRP=MPRP1(:,2:5);
    Num_miembros= MPRP1(end,1);

    PJ1=size(PJ);

```

```

Num_cargas=PJ1(1,1);

for i=1:Num_nodos;
    x=COORD(i,1);
    y=COORD(i,2);

    plot(x,y,'o');
    axis('equal');
    hold on;
end

%GRAFICA DE LOS ELEMENTOS

for i=1:Num_miembros;
    a=[(COORD(MPRP(i,1),1)),(COORD(MPRP(i,2),1))];
    b=[(COORD(MPRP(i,1),2)),(COORD(MPRP(i,2),2))];

    line(a,b)
    hold on
end

NodoDesplazado=zeros(Num_nodos,2);
for i=1:Num_nodos;
    NodoDesplazado(i,1)=COORD(i,1)+alpha*C(3*i-2,2);
    NodoDesplazado(i,2)=COORD(i,2)+alpha*C(3*i-1,2);
end
NodoDesplazado;

%GRAFICA DE LOS NUDOS DESPLAZADOS

for i=1:Num_nodos;
    x=NodoDesplazado(i,1);
    y=NodoDesplazado(i,2);

    plot(x,y,'s')
    hold on
end

%GRAFICO DE LOS APOYOS

for i=1:Num_soportes;

    if MSUP(i,2)==1 && MSUP(i,3)==1 && MSUP(i,4)==1
        %Empotrado
        n=4;
        R=mean(L)*0.1;
        tita=[0:(2*pi/n):2*pi]+pi/4;
        Xapoyo=R*cos(tita)+COORD(MSUP(i,1),1);
        Yapoyo=R*sin(tita)-R/2^0.5+COORD(MSUP(i,1),2);
        plot(Xapoyo,Yapoyo)

    elseif MSUP(i,2)==1 && MSUP(i,3)==1 && MSUP(i,4)==0
        n=3;
        R=mean(L)*0.1;
        tita=[0:(2*pi/n):2*pi]+pi/2;
    end
end

```

```

Xapoyo=R*cos(tita)+COORD(MSUP(i,1),1);
Yapoyo=R*sin(tita)-R+COORD(MSUP(i,1),2);
plot(Xapoyo,Yapoyo)

elseif MSUP(i,2)==1 && MSUP(i,3)==0 && MSUP(i,4)==0
%Apoyo Movil
R = mean(L)*0.07;
tita = (0:0.01:2.01*pi);
Xapoyo= R*cos(tita)+COORD(MSUP(i,1),1);
Yapoyo= R*sin(tita)-R+COORD(MSUP(i,1),2);

plot(Xapoyo,Yapoyo)

elseif MSUP(i,2)==0 && MSUP(i,3)==1 && MSUP(i,4)==0
%Apoyo Movil
R = mean(L)*0.07;
tita = (0:0.01:2.01*pi);
Xapoyo= R*cos(tita)+COORD(MSUP(i,1),1);
Yapoyo= R*sin(tita)-R+COORD(MSUP(i,1),2);

plot(Xapoyo,Yapoyo)

end

end

%GRAFICA DE LA ESTRUCTURA DEFORMADA

for i=1:Num_miembros;

xi=COORD(MPRP(i,1),1);
xf=COORD(MPRP(i,2),1);
yi=COORD(MPRP(i,1),2);
yf=COORD(MPRP(i,2),2);

L(i,1)=((xf-xi)^2+(yf-yi)^2)^0.5;

c=(xf-xi)/L(i,1); %c: Coseno
s=(yf-yi)/L(i,1); %s: Seno
T=[c s; -s c];

DE=[VRM' DTN];
DR=sortrows(DE,1);

Ma=zeros(6,1);
for m=1:6
Ma(m,1)=DR(Elemento(i,m),2);
end

n=100;
x=[0:L(i,1)/n:L(i,1)];

for p=1:n+1;
N1=1-x(p)/L(i,1);
N2=1-3*(x(p)/L(i,1)).^2+2*(x(p)/L(i,1)).^3;
%N3=-x(p)*(1-x(p)/L(i,1)).^2;
N3=-(-x(p)*(1-x(p)/L(i,1)).^2);

```

```

N4=x(p)/L(i,1);
N5=3*(x(p)/L(i,1)).^2-2*(x(p)/L(i,1)).^3;
%N6=-x(p)*((x(p)/L(i,1)).^2-x(p)/L(i,1));
N6=-(-x(p)*((x(p)/L(i,1)).^2-x(p)/L(i,1)));

N=[N1 0 0 N4 0 0; 0 N2 N3 0 N5 N6];

uv=N*Ma;
UV=T'*uv;

Yi=x(p)*(yf-yi)/L(i,1)+yi;
Xi=x(p)*(xf-xi)/L(i,1)+xi;

Xf(p)=Xi+uv(1,1)*alpha;
Yf(p)=Yi+uv(2,1)*alpha;

    %plot(Xf,Yf)
    %hold on
end

for i=1:n

    a=[Xf(i),Xf(i+1)];
    b=[Yf(i),Yf(i+1)];
    set(line(a,b),'Color','r','LineStyle','-')
end
end

%Dibujo de los vectores de fuerza

for i=1:Num_cargas;
    quiver(COORD(PJ(i,1),1),COORD(PJ(i,1),2),PJ(i,2)*0.02,PJ(i,3)*0.02);
    hold on
end

function Verificar_Callback(hObject, eventdata, handles)
cargas_miembros_Plane

function KG_Callback(hObject, eventdata, handles)
kg_Plane

function GLR_Callback(hObject, eventdata, handles)
kr_Plane

function GLF_Callback(hObject, eventdata, handles)
kf_Plane

% --- Executes on selection change in selector.
function selector_Callback(hObject, eventdata, handles)
XP=get(handles.selector,'Value');
global alpha
switch XP
    case 1
        axes(handles.axes7);

```

```

        delete axes7
        alpha=1

    case 2

        alpha=10

    case 3

        alpha=20

    case 4

        alpha=50

    case 5

        alpha=100

    case 6

        alpha=200

end
% --- Executes during object creation, after setting all properties.
function selector_CreateFcn(hObject, eventdata, handles)
% hObject    handle to selector (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in local.
function local_Callback(hObject, eventdata, handles)
matriz_elemento_plane

```