UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA CIVIL

“PROGRAMA PARA SOLUCIONAR VIGAS”

TESINA

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO CIVIL

PRESENTA:

ITZAGUERI GARCÍA RODRÍGUEZ

ASESOR: M.I. ALMA ROSA SÁNCHEZ IBARRA

MORELIA, MICHOACÁN, JULIO 2007.

FACULTAD
DE
INGENIERÍA CIVIL

ÍNDICE

Página

Página

3. Programa de Cómputo para resolver Vigas.

3.1 Diagrama de flujo.

3.2 Manual de usuario.

3.3 Ejemplos de Aplicación.

Conclusiones.

Glosario.

Bibligrafía.

INTRODUCCIÓN

Desde hace varias décadas se ha dado gran importancia a la aplicación de la computación, sobretodo en las ramas de las ingenierías, para realizar cálculos de manera más rápida y eficaz. Sin embargo, no sólo se deben aprender las reglas y formas para utilizar un programa, sino que debe de adquirirse el conocimiento de la ciencia de la computadora o el del procesamiento de datos.

A los profesionistas en el área de la construcción, les ha sido de gran ayuda contar con programas de cómputo para agilizar el cálculo y así resolver problemas de manera rápida y exacta.

Es necesario reconocer que las computadoras están llegando a ser una de las influencias más persuasivas tanto técnica como económicamente en la industria de la construcción, por lo que el Ingeniero Civil debe reconocer y adquirir el conocimiento relevante sobre computadoras y sus aplicaciones.

Las computadoras pueden asistirnos en la mayoría de los aspectos de la ingeniería Civil, por ejemplo, haciendo estimaciones, planificación, análisis estructural entre otras.

El conocer los diferentes sistemas de cómputo y los lenguajes para desarrollo de aplicaciones, así como el desarrollo de software, hará que el Ingeniero Civil pueda tomar decisiones adecuadas para implementar, adquirir o desarrollar aplicaciones que satisfagan sus necesidades.

Así, si no existe un software comercial para satisfacer sus necesidades, podrá desarrollar el programa que requiera, tomando en cuenta las características de desarrollo de software.

En la presente tesina se explicará el Método de las Rigideces para análisis de estructuras, además se verá que éste método, bastante laborioso para hacerse a mano, es muy aceptable para su uso en computadora, para lo cual se desarrollará un software para el análisis de vigas continuas por medio de éste método. Se darán ejemplos de aplicaciones específicas para la resolución de vigas.

OBJETIVO

El objetivo de este trabajo es el desarrollo de un programa para calcular los elementos mecánicos de una viga usando el método de las Rigideces, ya que realizar el cálculo en la computadora nos proporciona los resultados de manera rápida y con una gran aproximación.

Así la finalidad de este programa es facilitar el análisis de vigas, ya que al realizar el cálculo de manera manual se pueden cometer errores y se tardaría más tiempo, además de que probablemente no se tendría una buena aproximación en los resultados.

1. MÉTODO DE LAS RIGIDECES

1.1 FUNDAMENTOS DEL MÉTODO DE LAS RIGIDECES

El Método de las Rigideces ó Método de la Rigidez se basa en el análisis matricial, y es un Método de análisis de Desplazamientos. Algo importante de este método es que puede usarse para analizar estructuras formadas por elementos barra, tanto determinadas como indeterminadas, además nos da los desplazamientos y las fuerzas directamente.

Este método tiene la gran ventaja de que puede ser fácilmente programado y por lo mismo analizar una estructura con mayor rapidez.

Para la aplicación del Método de la Rigidez se requiere subdividir la estructura en elementos finitos e identificar sus extremos como nodos.

En términos muy generales este método se basa en establecer mediante el equilibrio y la compatibilidad, la relación que existe entre las cargas y los desplazamientos que estas generan en la estructura. Con base en esto podemos encontrar los desplazamientos en los nodos, para finalmente encontrar los elementos mecánicos en cada barra que compone la estructura.

1.2 IDENTIFICACIÓN DE MIEMBROS Y NODOS

Uno de los primeros pasos para aplicar este método es identificar los miembros de la estructura y sus nodos.

Lo anterior consiste en dividir la estructura en elementos barra que, dependiendo del tipo de carga a la que se encuentren sometidos, se analizarán primero de manera

individual para después conjuntarlos y analizar la estructura de manera global y obtener los desplazamientos en los nodos. Finalmente con los desplazamientos calculados se encuentran los elementos mecánicos en cada barra.

1.3 COORDENADAS

Como las cargas y los desplazamientos son cantidades vectoriales, debemos establecer un sistema coordenado para poder especificar su dirección y así establecer la posición de los elementos de la estructura.

Para ello se utilizan dos tipos diferentes de sistemas coordenados. Un sistema coordenado global o de la estructura, usando ejes x', y', en el cual especificaremos el sentido de las componentes externas de fuerza y desplazamiento de los nodos; y un sistema coordenado local o de miembro, el cual se usará en cada elemento para especificar el sentido de sus desplazamientos y cargas internas; este sistema se identifica usando ejes x, y, con origen en el nodo que de antemano queramos identificar como "nodo inicial", y el eje x señalando hacia el extremo alejado que llamaremos "nodo final".

1.4 GRADOS DE LIBERTAD

Cuando se carga una estructura, puntos específicos de ella, llamados nodos, sufrirán desplazamientos. El grado de libertad es el número de posibles desplazamientos que definen la posición deformada de la estructura.

Los desplazamientos que no están restringidos nos representan las incógnitas en el Método de la Rigidez, por lo tanto deben identificarse.

Al definir el grado de libertad debe tomarse en cuenta el tipo de estructura y sus apoyos, para así identificar los desplazamientos restringidos y los no restringidos. El grado de libertad de la estructura será igual al número total de desplazamientos no restringidos.

En el caso de vigas, si despreciamos los efectos de la fuerza axial y la fuerza cortante y consideramos sólo deflexiones causadas por flexión, la matriz de rigidez de la estructura será pequeña. Además, si los apoyos no tienen desplazamiento transversal por asentamientos, entonces cada nodo tiene solo un grado de libertad, representado como un desplazamiento angular.

Aunque en forma más general una viga puede tener dos desplazamientos posibles uno vertical y uno angular, ésto depende del tipo de apoyo.

1.5 MATRIZ DE RIGIDEZ DEL MIEMBRO

Entendemos como rigidez a la magnitud de la fuerza necesaria para producir un desplazamiento unitario, entendiendo que el desplazamiento puede ser lineal o angular, y que nos denota una posición.

Para un sistema coordenado tridimensional el vector que representa el desplazamiento tiene seis componentes, ésto a su vez nos indica que el vector fuerza en este sistema coordenado también tiene seis componentes.

Lo anterior representado matricialmente nos da:

$$\{d\} = \begin{Bmatrix} d_x \\ d_y \\ d_z \\ \phi_x \\ \phi_y \\ \phi_z \end{Bmatrix} \qquad \{P\} = \begin{Bmatrix} P_x \\ P_y \\ P_z \\ M_x \\ M_y \\ M_z \end{Bmatrix}$$

Para poder determinar la matriz **K** de rigidez de la estructura se debe primero establecer una matriz de rigidez del miembro **K'**, para cada miembro de la estructura, el cual será necesario calcular para usarse en la siguiente ecuación que es conocida como ecuación fuerza-desplazamiento:

$$\{P'\} = [K']\{d'\}$$

Particionando el vector de cargas, la matriz de rigidez y el vector de desplazamientos tomando en cuenta ambos extremos de la barra, la ecuación fuerza-desplazamiento nos queda de la siguiente manera:

$$\begin{Bmatrix} P'_1 \\ P'_2 \end{Bmatrix} = \begin{bmatrix} K'_{11} & K'_{12} \\ K'_{21} & K'_{22} \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix}$$

De acuerdo con la ecuación anterior podemos obtener una matriz de rigidez de un elemento barra en tres dimensiones, para el cual se calcularán los desplazamientos en sus nodos con sus tres componentes lineales y sus tres componentes angulares, además de los elementos mecánicos en los diferentes ejes.

1.6 Ecuación Fuerza-Desplazamiento para una barra tridimensional.

$$\begin{Bmatrix} N_{1x} \\ V_{1y} \\ V_{1z} \\ M_{1x} \\ M_{1y} \\ M_{1z} \\ N_{2x} \\ V_{2y} \\ V_{2z} \\ M_{2x} \\ M_{2y} \\ M_{2z} \end{Bmatrix} = \begin{Bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EI_Z}{L^3} & 0 & 0 & 0 & \frac{6EI_Z}{L^2} & 0 & -\frac{12EI_Z}{L^3} & 0 & 0 & 0 & \frac{6EI_Z}{L^2} \\ 0 & 0 & \frac{12EI_Y}{L^3} & 0 & -\frac{6EI_Y}{L^2} & 0 & 0 & 0 & -\frac{12EI_Y}{L^3} & 0 & -\frac{6EI_Y}{L^2} & 0 \\ 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 \\ 0 & 0 & -\frac{6EI_Y}{L^2} & 0 & \frac{4EI_Y}{L} & 0 & 0 & 0 & \frac{6EI_Y}{L^2} & 0 & \frac{2EI_Y}{L} & 0 \\ 0 & \frac{6EI_Z}{L^2} & 0 & 0 & 0 & \frac{4EI_Z}{L} & 0 & -\frac{6EI_Z}{L^2} & 0 & 0 & 0 & \frac{2EI_Z}{L} \\ -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{12EI_Z}{L^3} & 0 & 0 & 0 & -\frac{6EI_Z}{L^2} & 0 & \frac{12EI_Z}{L^3} & 0 & 0 & 0 & -\frac{6EI_Z}{L^2} \\ 0 & 0 & -\frac{12EI_Y}{L^3} & 0 & \frac{6EI_Y}{L^2} & 0 & 0 & 0 & \frac{12EI_Y}{L^3} & 0 & \frac{6EI_Y}{L^2} & 0 \\ 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \\ 0 & 0 & -\frac{6EI_Y}{L^2} & 0 & \frac{2EI_Y}{L} & 0 & 0 & 0 & \frac{6EI_Y}{L^2} & 0 & \frac{4EI_Y}{L} & 0 \\ 0 & \frac{6EI_Z}{L^2} & 0 & 0 & 0 & \frac{2EI_Z}{L} & 0 & -\frac{6EI_Z}{L^2} & 0 & 0 & 0 & \frac{4EI_Z}{L} \end{Bmatrix} \begin{Bmatrix} d_{1x} \\ d_{1y} \\ d_{1z} \\ \phi_{1x} \\ \phi_{1y} \\ \phi_{1z} \\ d_{2x} \\ d_{2y} \\ d_{2z} \\ \phi_{2x} \\ \phi_{2y} \\ \phi_{2z} \end{Bmatrix}$$

Como en una estructura no todos los miembros que la componen tienen la misma dirección, se debe desarrollar un medio para transformar esas cantidades de cada sistema coordenado local a un sistema coordenado global de la estructura. Esto se hace usando matrices de transformación de fuerzas y desplazamientos.

1.7 MATRIZ DE ROTACIÓN

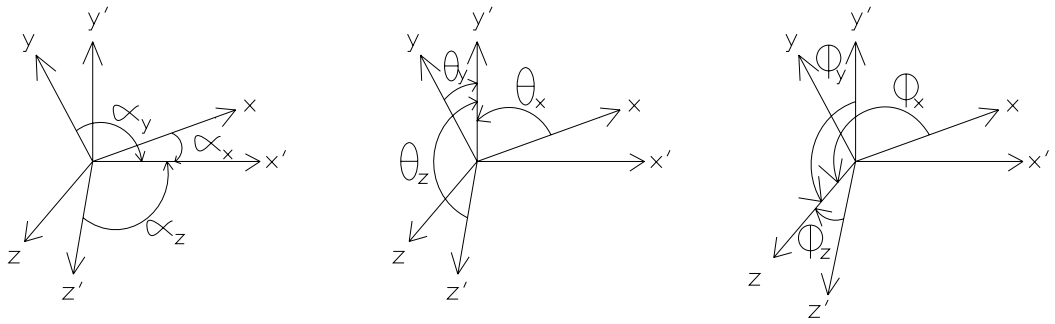
Haciendo uso de los cosenos directores y de acuerdo con las siguiente figuras podemos obtener la matriz de rotación en donde:

$$Cos\alpha_i = \lambda_i$$

$$Cos\theta_i = m_i$$

$$Cos\varphi_i = n_i$$

a) b) c)



La figura del inciso a) nos muestra el ángulo que forman los ejes locales del miembro con respecto al eje x' global de la estructura, de igual forma los incisos b) y c) nos muestran los ángulos que forman los ejes locales del miembro con respecto a los ejes y' y z' globales, respectivamente.

Matriz de rotación de ejes locales a ejes globales.

$$\begin{Bmatrix} P'_X \\ P'_Y \\ P'_Z \\ M'_X \\ M'_Y \\ M'_Z \end{Bmatrix} = \begin{bmatrix} \ell_X & \ell_Y & \ell_Z & 0 & 0 & 0 \\ m_x & m_y & m_z & 0 & 0 & 0 \\ n_x & n_y & n_z & 0 & 0 & 0 \\ 0 & 0 & 0 & \ell_x & \ell_y & \ell_z \\ 0 & 0 & 0 & m_x & m_y & m_z \\ 0 & 0 & 0 & n_x & n_y & n_z \end{bmatrix} \begin{Bmatrix} P_X \\ P_Y \\ P_Z \\ M_X \\ M_Y \\ M_Z \end{Bmatrix}$$

Una vez planteadas estas matrices de transformación, los elementos de la matriz de rigidez del miembro pueden transformarse de coordenadas locales a globales y luego usarlas para generar la matriz de rigidez de la estructura.

Ensamblada la matriz de rigidez **K** de la estructura, podemos primero determinar los desplazamientos nodales y luego las reacciones y las fuerzas en los miembros.

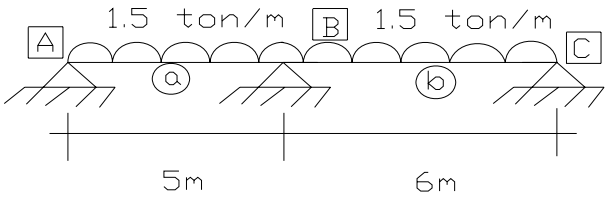
En el caso de vigas, se ha establecido que si los apoyos no sufren desplazamientos transversales, entonces cada uno tendrá solo un grado de libertad, siendo este un desplazamiento angular. En consecuencia, la matriz de rigidez de una viga queda representada por cuatro elementos, que son:

$$[K'] = \begin{bmatrix} \frac{4EI}{L} & \frac{2EI}{L} \\ \frac{2EI}{L} & \frac{4EI}{L} \end{bmatrix}$$

Además debemos notar que esta matriz de rigidez es equivalente a la matriz de rigidez global, ya que las rotaciones no se transforman por que coinciden los ejes locales del miembro con los ejes globales de la estructura.

1.8 EJEMPLO

Se resolverá por el Método de las Rigideces la siguiente viga, en la cual tomaremos el valor de EI = 1:



Como no existe carga horizontal aplicada no se genera carga axial, por lo tanto los desplazamientos horizontales en los apoyos B y C son nulos, además de que los desplazamientos verticales para los tres apoyos valen cero. Así el grado de libertad por nodo es uno, un desplazamiento angular.

Ecuación fuerza-desplazamiento:

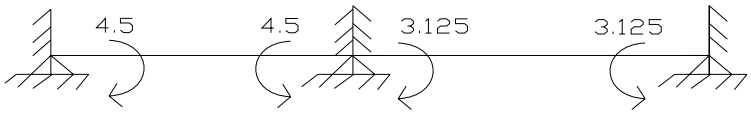
$$\begin{Bmatrix} P'_A \\ P'_B \\ P'_C \end{Bmatrix} = \begin{bmatrix} K'_{1,1a} & K'_{1,2a} & 0 \\ K'_{2,1a} & K'_{2,2a}+K'_{1,1b} & K'_{1,2b} \\ 0 & K'_{2,1b} & K'_{2,2b} \end{bmatrix} \begin{Bmatrix} d'_A \\ d'_B \\ d'_C \end{Bmatrix}$$

Determinando el vector de cargas tenemos:

$$M_{AB} = M_{BA} = \frac{WL_a^2}{12} = \frac{1.5(6)^2}{12} = 4.5\ ton-m$$

$$M_{BC} = M_{CB} = \frac{WL_b^2}{12} = \frac{1.5(5)^2}{12} = 3.125\ ton-m$$

En la siguiente figura se representan las fuerzas de empotramiento perfecto, considerando positivos a los momentos con sentido antihorario:



El vector de cargas para la viga resulta:

$$\begin{Bmatrix} P'_A \\ P'_B \\ P'_C \end{Bmatrix} = \begin{Bmatrix} M'_A \\ M'_B \\ M'_C \end{Bmatrix} = \begin{Bmatrix} -4.5 \\ 4.5-3.125 \\ 3.125 \end{Bmatrix} = \begin{Bmatrix} -4.5 \\ 1.325 \\ 3.125 \end{Bmatrix} \qquad \begin{Bmatrix} d'_A \\ d'_B \\ d'_C \end{Bmatrix} = \begin{Bmatrix} \phi_A \\ \phi_B \\ \phi_C \end{Bmatrix}$$

Por las condiciones de la viga, la matriz de rigidez para cada barra es la siguiente:

$$[K'] = \begin{bmatrix} \frac{4EI}{L} & \frac{2EI}{L} \\ \frac{2EI}{L} & \frac{4EI}{L} \end{bmatrix}$$

La matriz anterior la podemos dividir en submatrices, las cuales nos resultan de la siguiente manera:

$$[K'_{1,1}] = [K'_{2,2}] = \frac{4EI}{L} \qquad [K'_{1,2}] = [K'_{2,1}] = \frac{2EI}{L}$$

Sustituyendo los valores para cada barra:

$$\left[K'_{1,1a}\right] = \frac{4EI}{L} = 0.667EI \qquad \left[K'_{1,2a}\right] = \frac{2EI}{L} = 0.333EI$$

$$\left[K'_{1,1b}\right] = \frac{4EI}{L} = 0.80EI \qquad \left[K'_{1,2b}\right] = \frac{2EI}{L} = 0.40EI$$

Sustituyendo en la ecuación fuerza-desplazamiento de la estructura:

$$\begin{Bmatrix} -4.5 \\ 1.375 \\ 3.125 \end{Bmatrix} = EI \begin{bmatrix} 0.667 & 0.333 & 0 \\ 0.333 & 1.467 & 0.40 \\ 0 & 0.40 & 0.80 \end{bmatrix}$$

Resolviendo el sistema se obtiene:

$$\begin{Bmatrix} \phi'_A \\ \phi'_B \\ \phi'_C \end{Bmatrix} = \frac{1}{EI} \begin{bmatrix} -7.68 \\ 1.87 \\ 2.97 \end{bmatrix}$$

Por compatibilidad:

$$\left\{d'_{1a}\right\} = \left\{d'_A\right\} \qquad \left\{d'_{2a}\right\} = \left\{d'_B\right\}$$

$$\left\{d'_{1b}\right\} = \left\{d'_B\right\} \qquad \left\{d'_{2b}\right\} = \left\{d'_C\right\}$$

Sustituyendo en la ecuación fuerza-desplazamiento para cada barra:

$$\{P'_1\} = [K'_{1,1}]\{d'_1\} + [K'_{1,2}]\{d'_2\}$$
$$\{P'_2\} = [K'_{2,1}]\{d'_1\} + [K'_{2,2}]\{d'_2\}$$

Barra “a”.

$$\{P'_{1a}\} = [K'_{1,1a}]\{d'_A\} + [K'_{1,2a}]\{d'_B\}$$
$$M'_{Z1} = (0.667EI)\frac{-7.68}{EI} + (0.333EI)\frac{1.87}{EI} = -4.5\ ton-m$$
$$\{P'_{2a}\} = [K'_{2,1a}]\{d'_A\} + [K'_{2,2a}]\{d'_B\}$$
$$M'_{Z2} = (0.333EI)\frac{-7.68}{EI} + (0.667EI)\frac{1.87}{EI} = -1.31\ ton-m$$

Barra “b”.

$$\{P'_{1b}\} = [K'_{1,1b}]\{d'_B\} + [K'_{1,2b}]\{d'_C\}$$
$$M'_{Z1} = (0.80EI)\frac{1.87}{EI} + (0.40EI)\frac{2.97}{EI} = 2.685\ ton-m$$
$$\{P'_{2b}\} = [K'_{2,1b}]\{d'_B\} + [K'_{2,2b}]\{d'_C\}$$
$$M'_{Z2} = (0.40EI)\frac{1.87}{EI} + (0.80EI)\frac{2.97}{EI} = 3.125\ ton-m$$

Comprobación del equilibrio:

Nodo “a”.

$$\{P'_A\} = \{P'_{1a}\}$$

$$-4.5 = -4.5$$

Nodo “b”.

$$\{P'_B\} = \{P'_{2a}\} + \{P'_{1b}\}$$

$$1.375 = -1.31 + 2.685$$

$$1.375 = 1.375$$

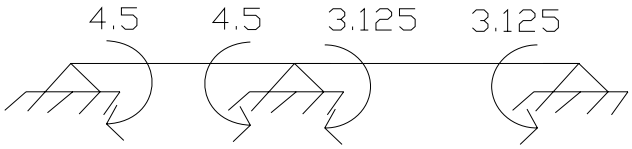
Nodo “c”.

$$\{P'_B\} = \{P'_{2b}\}$$

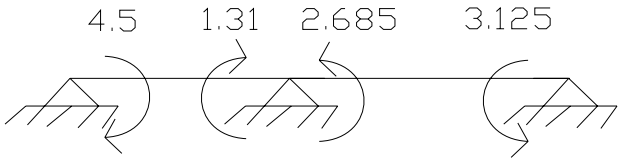
$$3.125 = 3.125$$

Para obtener los momentos finales, debemos sumar las acciones de empotramiento perfecto (cambiándoles el signo) a las reacciones que se obtienen con ecuación fuerza-desplazamiento:

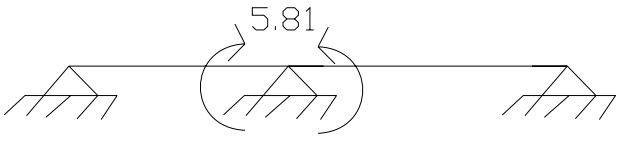
Fuerzas de Empotramiento perfecto



Momentos obtenidos en la ecuación fuerza-desplazamiento



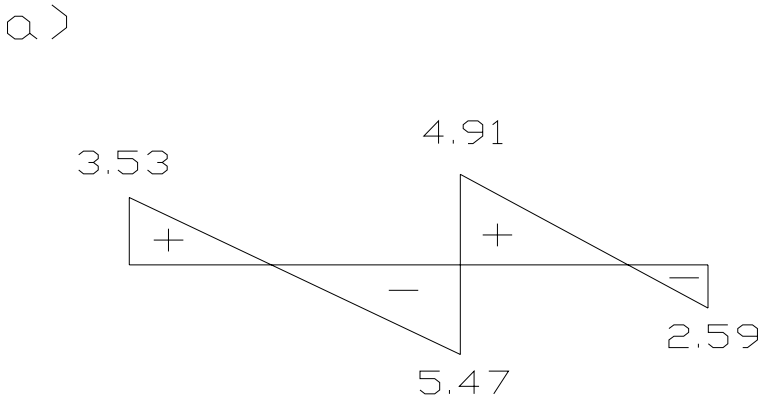
De acuerdo a lo explicado anteriormente, multiplicamos las fuerzas de empotramiento perfecto por menos uno, y las sumamos a los momentos obtenidos en la ecuación, así encontramos los momentos finales:

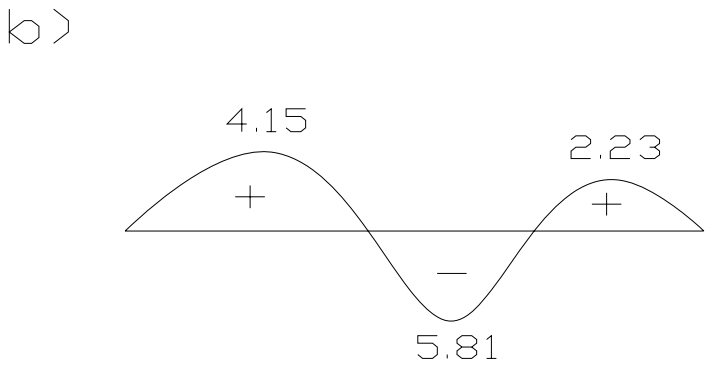


Finalmente trazamos los diagramas de:

a) Fuerza cortante (ton).

b) Momento flexionante (ton-m).





2. FORTRAN

2.1 INTRODUCCION

Hoy en día la programación es de vital importancia cuando se trata de realizar cálculos mecanizados en los que los datos del problema suelen ser las únicas variables y que con base en estos datos, el procedimiento a seguir para el análisis y resolución del problema es igual para cualquier posible caso.

Por lo anterior es necesario desarrollar programas que realicen todos los cálculos de manera más rápida y con mejor aproximación en los resultados.

2.2 DEFINICIONES

Definición de programa:
Es una serie de instrucciones que se le indican a una computadora para que esta realice una tarea específica.

Definición de lenguaje de programación:

Conjunto de normas "lingüísticas" (palabras y símbolos) que permiten escribir un programa y que éste sea entendido por el ordenador y pueda ser trasladado a ordenadores similares para su funcionamiento en otros sistemas.

Es complicado definir qué es y qué no es un lenguaje de programación. Se asume generalmente que la traducción de las instrucciones a un código que comprende la computadora debe ser completamente sistemática. Normalmente es la computadora la que realiza la traducción.

Un lenguaje de programación es una notación para escribir programas, a través de los cuales podemos comunicarnos con el hardware y dar así las órdenes adecuadas

para la realización de un determinado proceso. Un lenguaje esta definido por una gramática o conjunto de reglas que se aplican a un alfabeto constituido por el conjunto de símbolos utilizados.

Definición de lenguaje de alto nivel:

Estos lenguajes son los más utilizados por los programadores. Están diseñados para que las personas escriban y entiendan los programas de un modo mucho más fácil que los lenguajes máquina y ensamblador. Un programa escrito en lenguaje de alto nivel es independiente de la máquina (las instrucciones no dependen del diseño del hardware o de una computadora en particular), por lo que estos programas son portables o transportables. Los programas escritos en lenguaje de alto nivel pueden ser ejecutados con poca o ninguna modificación en diferentes tipos de computadoras. Son lenguajes de programación en los que las instrucciones enviadas para que el ordenador ejecute ciertas órdenes son similares al lenguaje humano. Dado que el ordenador no es capaz de reconocer estas ordenes, es necesario el uso de un intérprete que traduzca el lenguaje de alto nivel a un lenguaje de bajo nivel que el sistema pueda entender.

Por lo general se piensa que los ordenadores son máquinas que realizan tareas de cálculos o procesamiento de texto. La descripción anterior es sólo una forma muy esquemática de ver una computadora. Hay un alto nivel de abstracción entre lo que se pide a la computadora y lo que realmente comprende. Existe también una relación compleja entre los lenguajes de alto nivel y el código máquina.

FORTRAN

Fortran acrónimo de "Formula Translation", es un lenguaje de programación desarrollado en 1957 y activamente utilizado desde entonces.
Fortran fue uno de los primeros lenguajes de alto nivel para computadoras. Desde entonces, se han desarrollado unos 450 lenguajes de programación, pero aún así, el FORTRAN sigue siendo un lenguaje poderoso que además es utilizado para la enseñanza de programación debido a su flexibilidad y facilidad de aprendizaje.

Fortran es un lenguaje compilado. Esto significa que, una vez escrito el programa, éste ha de ser traducido en bloque al lenguaje máquina, o sea, el lenguaje que entiende el procesador del ordenador, mediante un proceso llamado compilación; por contra, en un lenguaje interpretado las líneas de programa se van traduciendo según el flujo va pasando por ellas, con lo que el proceso de ejecución se hace más lento.

Fortran es el más antiguo de los lenguajes de alto nivel. Antes de él, todos los programas se escribían en lenguaje ensamblador o en lenguaje máquina.

Por convención, una versión de Fortran es acompañada con los últimos dos dígitos del año en que se propuso la estandarización. Por lo que se tiene:

- Fortran 66
- Fortran 77
- Fortran 90 (95)

La versión más común de Fortran actualmente es todavía Fortran 77, sin embargo Fortran 90 esta creciendo en popularidad. Fortran 95 es una versión revisada de Fortran 90 la cual fue aprobada por ANSI en 1996.

2.3 RAZONES PARA APRENDER FORTRAN

Fortran es un lenguaje de programación dominante, usado en aplicaciones de ingeniería y matemáticas, por lo que es importante que se tenga bases para poder leer y modificar un código de Fortran. Algunas opiniones de expertos han dicho que Fortran será un lenguaje que pronto decaerá en popularidad y se extinguirá, lo que no ha sucedido todavía.

Una de las razones para esta sobrevivencia es la inercia del software, ya que una vez que una compañía ha gastado muchos millones de dólares y de años en el desarrollo de software, no le es conveniente traducir el software a un lenguaje diferente, por el costo que implica y por ser una tarea difícil y laboriosa.

2.4 ORGANIZACIÓN DEL PROGRAMA

Un programa de Fortran por lo general consiste de un programa principal o main (manejador) y posiblemente varios subprogramas (procedimientos o subrutinas). Por el momento se considerará que todas las sentencias están en el programa principal; los subprogramas se revisarán más adelante. La estructura del programa principal es:

```
program (nombre)

    declaraciones

    sentencias

    stop

end program
```

La sentencia stop es opcional y podría ser vista como redundante ya que el programa terminará cuando alcance el fin, pero se recomienda que en el programa se incluya la sentecia stop para resaltar que la ejecución del programa ahí termina.

2.5 LENGUAJE FORTRAN

Comentarios.

Una línea que inicia con una letra "c" o un asterisco (*) en la primera columna es un comentario. El comentario puede aparecer en cualquier lugar del programa. El colocarlos en el lugar preciso incrementan la legibilidad. Los códigos comerciales de Fortran contienen un 50% de comentarios. Se pueden encontrar también programas que usan el signo de exclamación (!) para comentarios, en este caso el comentario puede iniciar en cualquier columna.

Espacios en Blanco.

Los espacios en blanco son ignorados en Fortran 90. Por lo tanto si se remueven todos los espacios en blanco en un programa de Fortran 90, el programa sintácticamente es correcto, pero no es legible para los humanos.

Declaración y tipos de Variables

Nombre de Variables.

Los nombres de variables en Fortran consisten en caracteres escogidos de la a “a” la “z” y de los dígitos del 0 al 9. El primer caracter debe ser una letra, en F90 se

permiten nombres de longitud arbitraria, además de que no diferencía entre mayúsculas y minúsculas.

Tipos y declaraciones.

Cada variable debería ser definida con una declaración. Esto indica el tipo de la variable. Las declaraciones más comunes son:

`Integer :: lista de variables;` para variables enteras.
`Real :: lista de variables;` para variables reales.
`Double precision :: lista de variables;` para variables que requieran doble precisión.
`Complex :: lista de variables;` para variables complejas.
`Character :: lista de variables;` para variables de tipo caracter.

La lista de variables consiste de nombres de variables separadas por comas. Cada variable deberá ser declarada solamente una vez. Si una variable no esta declarada, F90 usa un conjunto implícito de reglas para establecer el tipo. Con lo anterior todas las variables que comiencen con el conjunto de letras i-n son enteros y el resto tipo real.
La probabilidad de errores en el programa crece exponencialmente si no se declaran las variables explícitamente.

Variables Enteras y de punto flotante.

F90 sólo tiene un tipo para variables enteras. Los enteros son usualmente guardados en 32 bits (4 bytes).

F90 tiene dos tipos diferentes para punto flotantes conocidos como real y doble precisión. Mientras el tipo real es por lo general adecuado, algunos cálculos numéricos requieren de una mayor precisión por lo que *double precision* deberá ser usado.

El tamaño por lo general es para el tipo real de 4 bytes y el de doble precisión es de 8 bytes, pero lo anterior depende de la máquina y el compilador. Algunas versiones no estandarizadas de Fortran usan la sintaxis real*8 para indicar una variable de punto flotante de 8 bytes.

La sentencia *parameter***.**

Algunas constantes aparecen varias veces en un programa, por lo que es deseable que se definan una sola vez al principio del program. Esto se puede hacer con la sentencia *parameter*, a la vez que hace que los programas sean más legibles.

Expresiones

La precedencia de los operadores aritméticos en Fortran es (de la más alta a la más baja):

** {exponenciación}

*/ {multiplicación, división}

+ - {suma, resta}

Todos los operadores son evaluados de izquierda a derecha, excepto el operador de exponenciación, el cual tiene precedencia de derecha a izquierda. Si se desea cambiar el orden de evaluación, se pueden usar paréntesis.

Expresiones Lógicas

Una expresión lógica puede tener solamente el valor de .TRUE. o de .FALSE.. Un valor lógico puede ser obtenido al comparar expresiones aritméticas usando los siguientes operadores relacionales:

```
.LT.   <
.LE.   <=
.GT.   >
.GE.   >=
.EQ.   =
.NE.   /=
```

Las expresiones lógicas pueden ser combinadas con los operadores lógicos .AND. .OR. y .NOT. que corresponden a los operadores lógicos conocidos Y, O y negación respectivamente.

Las expresiones lógicas son usadas frecuentemente en sentencias condicionales.

La sentencia *if*.

Una parte importante de cualquier lenguaje de programación son las sentencias condicionales. La sentencia más común en Fortran es *if*, la cual tiene varias formas de uso. La forma más simple de la sentencia *if* es:

```
if (expresión lógica) sentencia
```

En este caso la sentencia se ejecuta si la expresión lógica resulta verdadera, si no es así, el programa la ignora.

Lo anterior tiene que ser escrito en una sola línea.

Si más de una sentencia necesita ser ejecutada dentro de la sentencia if, entonces la siguiente sintaxis deberá ser usada:

```
if (expresión lógica) then
sentencias
end if
```

La forma más general más general de la sentencia if tiene la siguiente forma:

```
if (expresión lógica) then
sentencias
else if (expresión lógica) then
sentencias
:
:
else
sentencias
end if
```

El flujo de ejecución es de arriba hacia abajo. Las expresiones condicionales son evaluadas en secuencia hasta que se encuentra una que es verdadera. Entonces el

código asociado es ejecutado y el control salta a la siguiente línea después de la sentencia *end if*.

Sentencias *if* anidadas.

La sentencia *if* puede ser anidada varios niveles. Para asegurar la legibilidad es importante sangrar las sentencias:

```
if (x > 0) then
     if (x >= y) then
     write(*,*) 'x es positivo y x >= y'
     else
     write(*,*) 'x es positivo pero, x < y'
     end if
else if (x < 0) then
write(*,*) 'x es negativo'
else
write(*,*) 'x es cero'
end if
```

Como se observa en el código anterior, las sentencias *if* anidadas resultan útiles cuando se deben evaluar varias condiciones antes de ejecutar una sentencia, sin embargo, se debe evitar anidar muchos niveles de sentencias *if* ya que es difícil de seguir.

Ciclos

Para repetir la ejecución de sentencias se usan los ciclos. F90 tiene solamente una construcción de ciclo, conocida como el *ciclo-do*. El *ciclo-do* corresponde al *ciclo-for* que existe en otros lenguajes de programación. Otros ciclos pueden ser simulados usando las sentencias *if* y *goto*.

Ciclos-do.

El *ciclo-do* es usado para repetir un conjunto de sentencias una determinada cantidad de veces.

```
Do var = expr1, expr2, expr3
sentencias
end do
```

var: es la variable del ciclo (conocida con frecuencia como el índice del ciclo) el cual deberá ser del tipo integer.

expr1: indica el valor inicial de var,
expr2: es el valor hasta el que llegará el índice, y
expr3: es el incremento (step).

Ciclos while.

La forma más intuitiva para escribir un ciclo *while* es:

```
do while (expr lógica)
sentencias
end do
```

Las sentencias en el cuerpo serán repetidas mientras la condición en el ciclo while sea verdadera.

Arreglos

Muchos cálculos científicos usan vectores y matrices. El tipo de dato usado en Fortran para representar tales objetos es el array (arreglo). Un arreglo unidimensional corresponde a un vector, mientras que un arreglo bidimensional corresponde a una matriz.

Arreglos Unidimensionales.

El arreglo más sencillo es el de una dimensión, el cual es sólo un conjunto de elementos almacenados secuencialmente en memoria. Por ejemplo, la declaración:

```
real, dimension (20) :: d
```

Declara a d como un arreglo del tipo real con 20 elementos. Esto es, d consiste de 20 números del tipo real almacenados en forma contigua en memoria.

El tipo de los elementos de un arreglo puede ser cualquiera de los tipos básicos de datos ya vistos. Ejemplos:

```
integer :: i(10)
double precision :: x(100)
```

Arreglos Bidimensionales.

Las matrices son muy importantes en álgebra lineal. Las matrices son usualmente representadas por arreglos bidimensionales. Por ejemplo, la declaración:

```
real, dimension (3, 5) :: Arreglo
```

En la declaración anterior el número 3 indica los renglones de la matriz y el número 5 indica las columnas.

Una característica importante de Fortran 90 es que es posible usar almacenamiento en memoria dinámica, con lo que se puede hacer que todos los arreglos "trabajen" no importando su tamaño.

Subprogramas

Cuando un programa tiene más de cien líneas, es difícil de seguir. Los códigos de Fortran que resuelven problemas reales de ingeniería por lo general tienen decenas de miles de líneas. La única forma para manejar códigos tan grandes, es usar una aproximación modular y dividir el programa en muchas unidades independientes pequeñas llamadas *subprogramas*.

Un subprograma es una pequeña pieza de código que resuelve un subproblema bien definido. En un programa grande, se tiene con frecuencia que resolver el mismo subproblema con diferentes tipos de datos. En vez de replicar el código, estas tareas pueden resolverse con subprogramas. El mismo *subprograma* puede ser llamado varias veces con distintas entradas de datos.

En Fortran se tienen dos tipos diferentes de *subprogramas*, conocidas como funciones y subrutinas.

Funciones

Las *funciones* en Fortran son bastante similares a las funciones matemáticas: ambas toman un conjunto de variables de entrada (parámetros) y regresan un valor de algún tipo. Los subprogramas son definidos por el usuario, pero F90 tiene también *funciones incorporadas*.

Algunas de las más comunes son:

`abs:` valor absoluto
`min:` valor mínimo
`max:` valor máximo
`sqrt:` raíz cuadrada
`sin:` seno
`cos:` coseno
`tan:` tangente
`atan:` arco tangente
`exp:` exponencial (natural)
`log:` logaritmo (natural)

Subrutinas.

Una función de Fortran puede devolver únicamente un valor. En ocasiones se desean regresar dos o más valores y en ocasiones ninguno. Para este propósito se usa la construcción *subrutina*. La sintaxis es la siguiente:

```
subroutine nombre (lista_de_parámetros)
declaraciones
:
sentencias
:

end subroutine
```

Se deben crear dos bloques de declaración de variables en el código. Primero, se declaran los parámetros de entrada/salida, es decir, las variables que son comunes al que llama y al que recibe la llamada. Después, se declaran las variables locales, esto es, las variables que serán sólo conocidas dentro del subprograma. Se pueden usar los mismos nombres de variables en diferentes subprogramas.

Entradas/Salidas (E/S)

Una parte importante del cualquier programa de cómputo es manejar la entrada y la salida de datos. Las dos construcciones más comunes de Fortran son: *read* y *write.* La E/S con Fortran puede ser un poco complicada.

La sentencia *read* es usada para la entrada y la sentencia *write* para la salida. El formato es:

```
read (núm_unidad, núm_formato) lista_de_variables
write(núm_unidad, núm_formato) lista_de_variables
```

El número de unidad se puede referir a la salida estándar, entrada estándar o a un archivo.
El número de formato se refiere a una etiqueta para la sentencia *format*. Es posible simplificar estas sentencias usando asteriscos (*) para algunos argumentos.

```
read (*,*) lista_de_variables
write(*,*) lista_de_variables
```

La primera sentencia leerá valores de la entrada estándar y asignará los valores a las variables que aparecen en la lista, y la segunda escribe la lista en la salida estándar.

Sentencia Format

Por lo general un programador desea indicar algún formato de entrada o salida, por ejemplo, el número de decimales que tendrá un número real. Para este propósito F90 tiene la sentencia *format*. La misma sentencia *format* puede ser usada para la entrada o salida.

Sintaxis

```
write(*, etiqueta) lista_de_variables
etiqueta format códigos_de_formato
```

La sentencia *format* puede estar en cualquier lugar dentro del programa. Hay dos estilos de programación: agrupar por parejas las sentencias, o poner el grupo de sentencias *format* al final del (sub)programa.

Códigos comunes de formato:

Las letras para códigos de formato más comunes son:

```
A - cadena de texto
D - números de doble precisión, notación científica
E - números reales, notación científica
F - números reales, formato de punto fijo
I - entero
X - salto horizontal (espacio)
/ - salto vertical (nueva línea)
```

El código de formato `F` (y similarmente `D` y `E`) tiene la forma general "`Fd.a`", donde "`d`" es una constante entera indicando el ancho del campo y "`a`" es un entero constante que indica el número de dígitos significativos. Por ejemplo: "`F8.3`".
Para los enteros solamente el campo de ancho es indicado, por lo que la sintaxis es "`Ia`". En forma parecida las cadenas de caracteres pueden ser especificadas como "`A`" ó "`a`".

Para un espaciado horizontal, el código usado es "`nX`". Donde n indica el número de espacios horizontales.
Para espaciado vertical (nuevas líneas) se usa el código "/". Cada diagonal corresponde a una nueva línea.

E/S de Archivos

La salida/entrada se puede realizar en los dispositivos estándares de entrada/salida. También es posible leer o escribir de archivos los cuales son guardados en algún dispositivo externo de almacenamiento, por lo general un disco (disco duro, floppy) o una cinta. En Fortran cada archivo esta asociado con un número de unidad, un entero entre 1 y 99. Algunos números están reservados: 5 es la entrada estándar, 6 es la salida estándar.

Abriendo y cerrando un archivo.

Antes de que pueda usarse un archivo se requiere que sea abierto (open).

El comando es:

```
open (lista_de_especificadores)
```

Donde los especificadores más comunes son:

```
[UNIT=] u
IOSTAT= ios
ERR= err
FILE= nomb_arch
STATUS= sta
ACCESS= acc
FORM= frm
RECL= rl
```

u: es el número de unidad para algún archivo, el programador lo escoge debiendo ser un número único.

ios: es el identificador del estado de la E/S y debe ser una variable entera. El valor que regresa ios es cero si la sentencia fue exitosa y sino, regresa un valor diferente de cero.

err: es una etiqueta a la cual el programa saltará si hay un error.

nomb_arch: es una cadena de caracteres que contiene el nombre del archivo.

sta: es una cadena de caracteres que tiene que ser NEW, OLD o SCRATCH. Esta muestra el estatus del archivo. Un archivo scratch es aquel que es creado y borrado cuando el archivo es cerrado (o el programa termina).

acc: deberá ser SEQUENTIAL o DIRECT. El valor predefinido es SEQUENTIAL.

frm: deberá ser FORMATTED o UNFORMATTED. El valor predefinido es UNFORMATTED.

rl: indica la longitud de cada registro en un archivo de acceso directo.

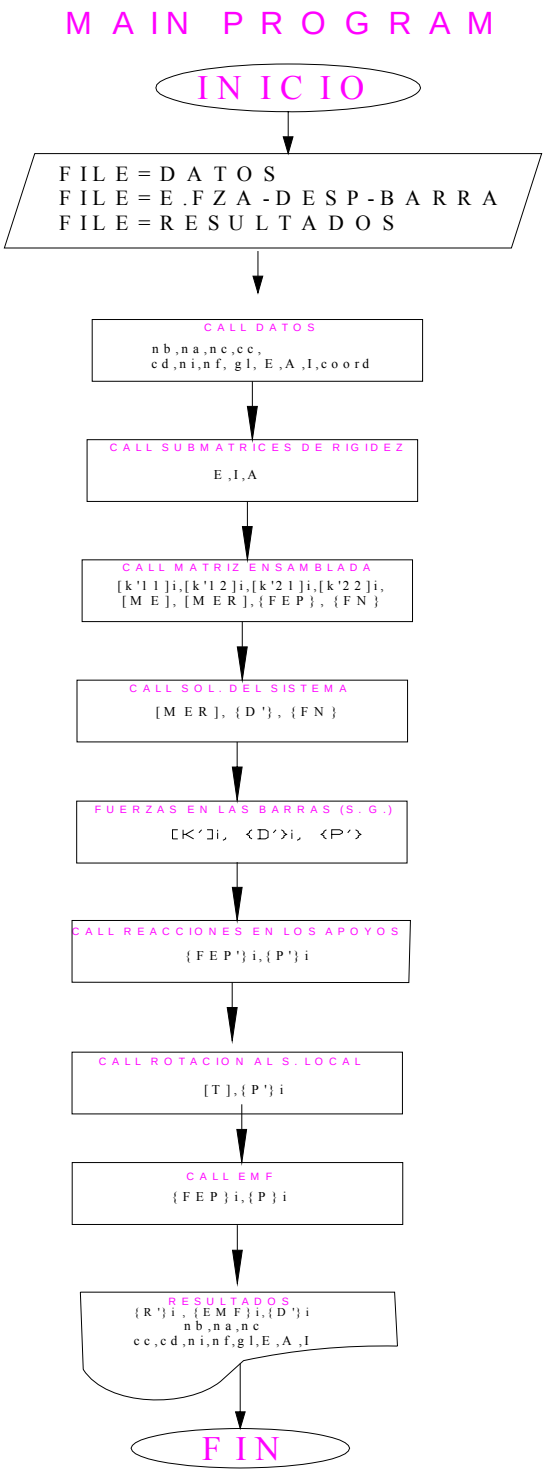
Una vez que un archivo ha sido abierto, se puede acceder con sentencias de lectura y escritura. Cuando se manipula un archivo y se termina de usar, deberá ser cerrado usando la sentencia:

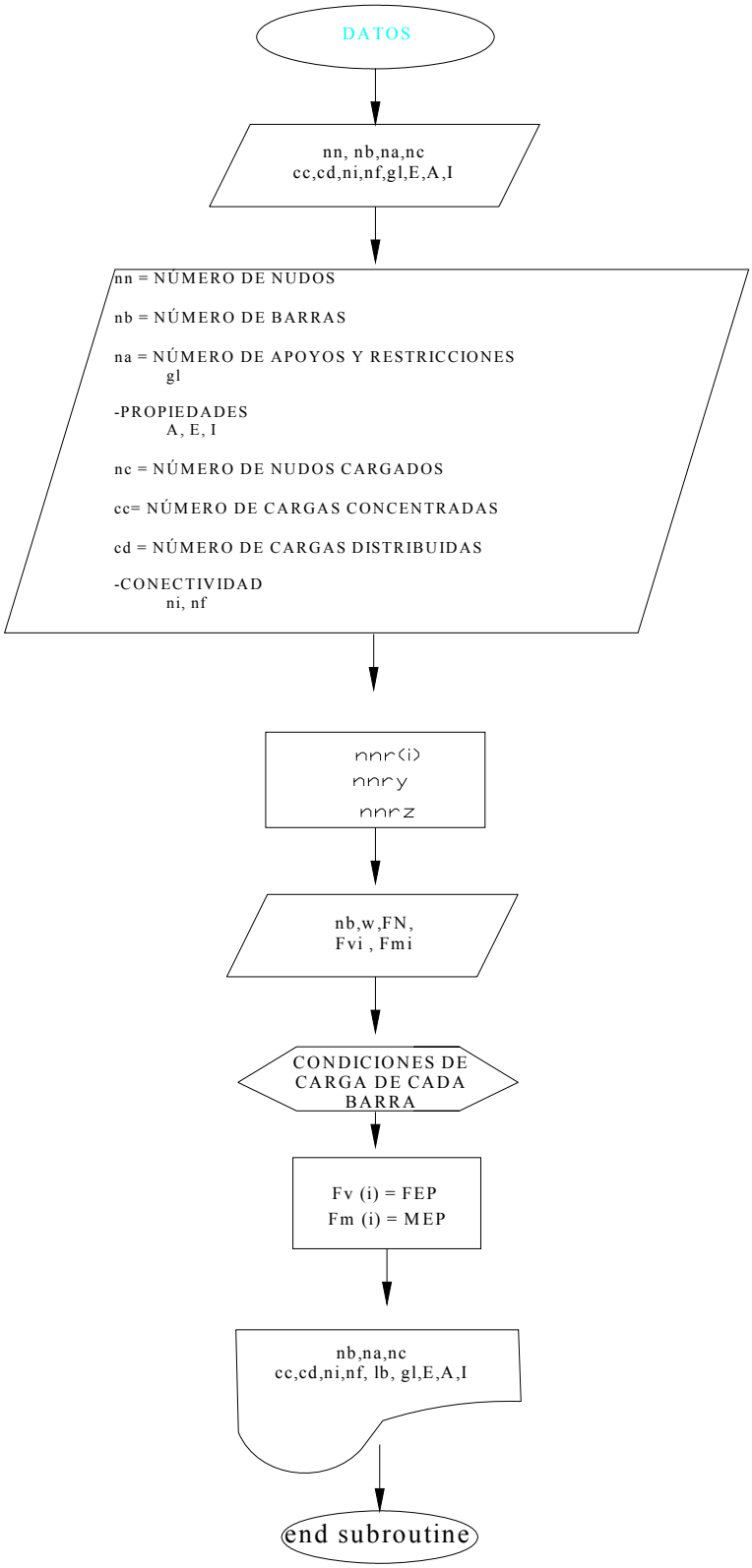
```
close ([UNIT=]u[,IOSTAT=ios,ERR=err,STATUS=sta])
```

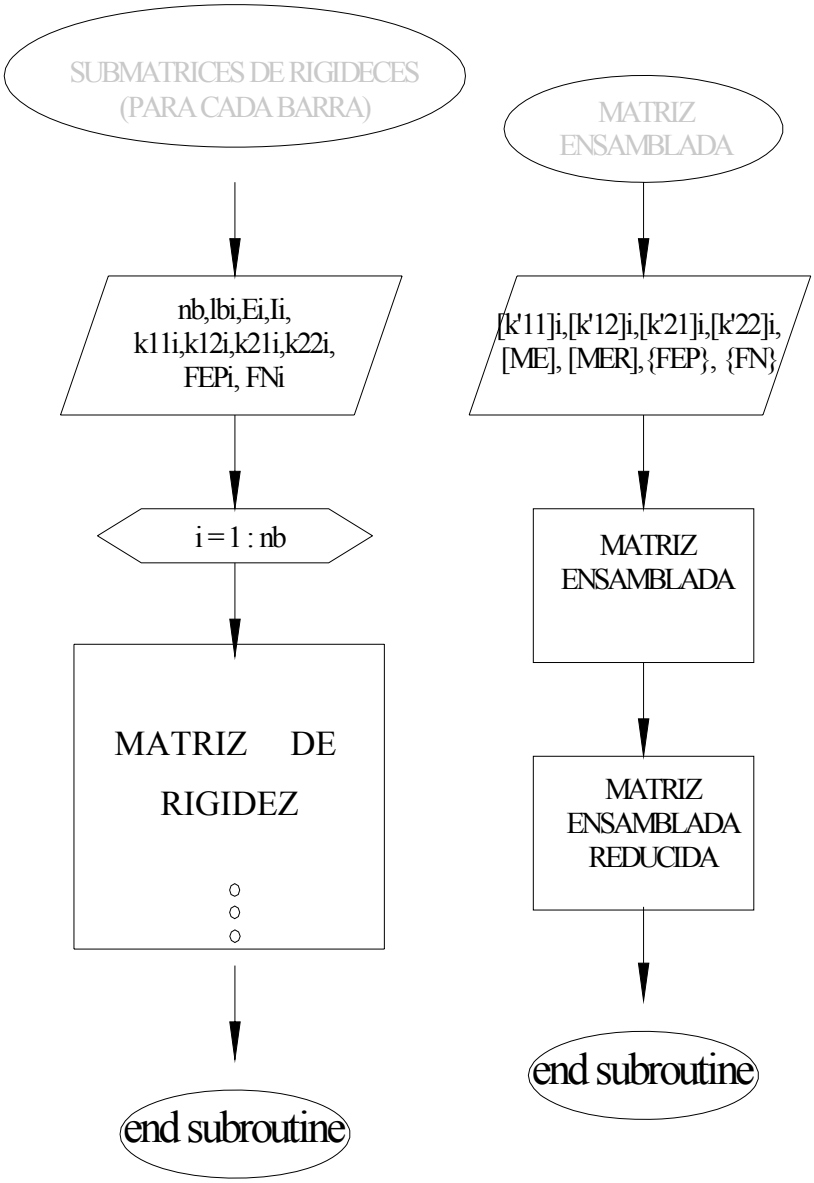
.

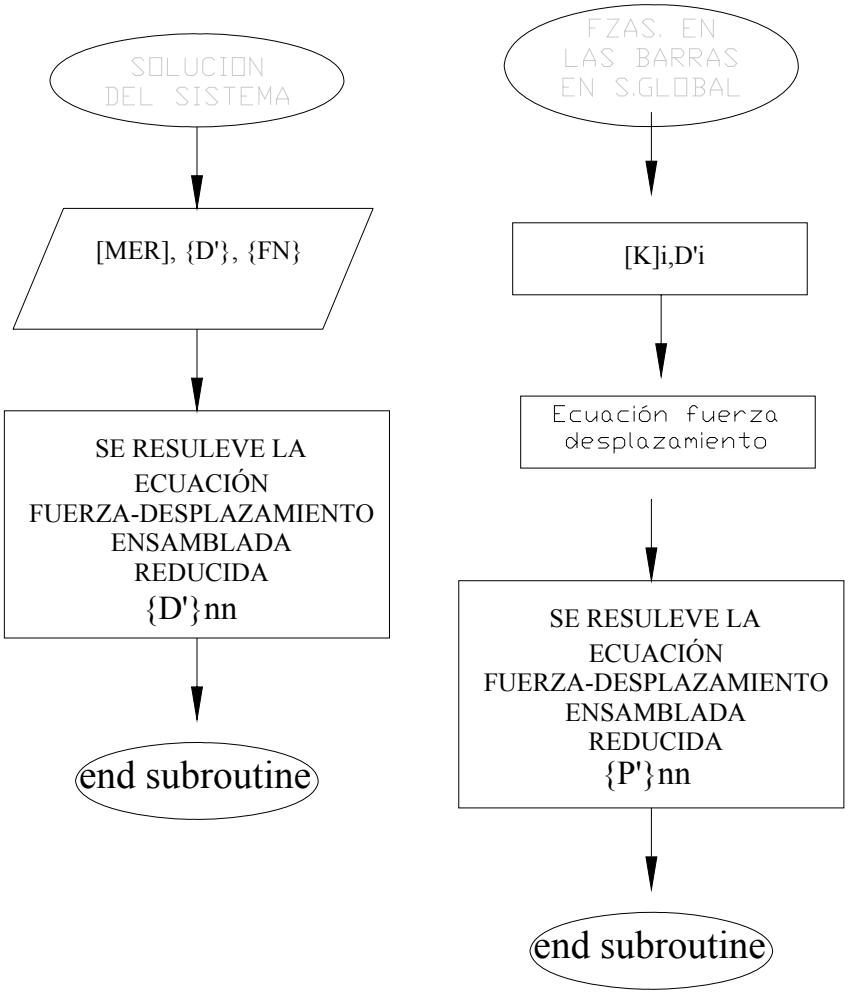
3. PROGRAMA DE CÓMPUTO PARA RESOLVER VIGAS

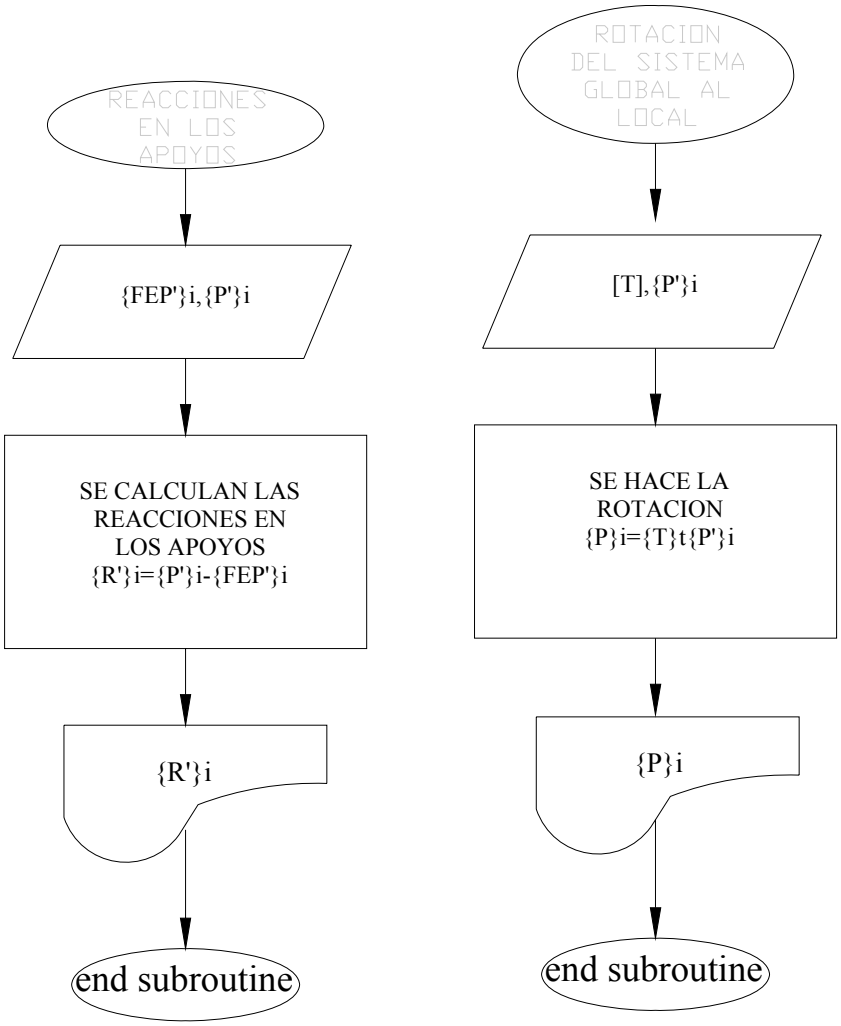
3.1 DIAGRAMAS DE FLUJO

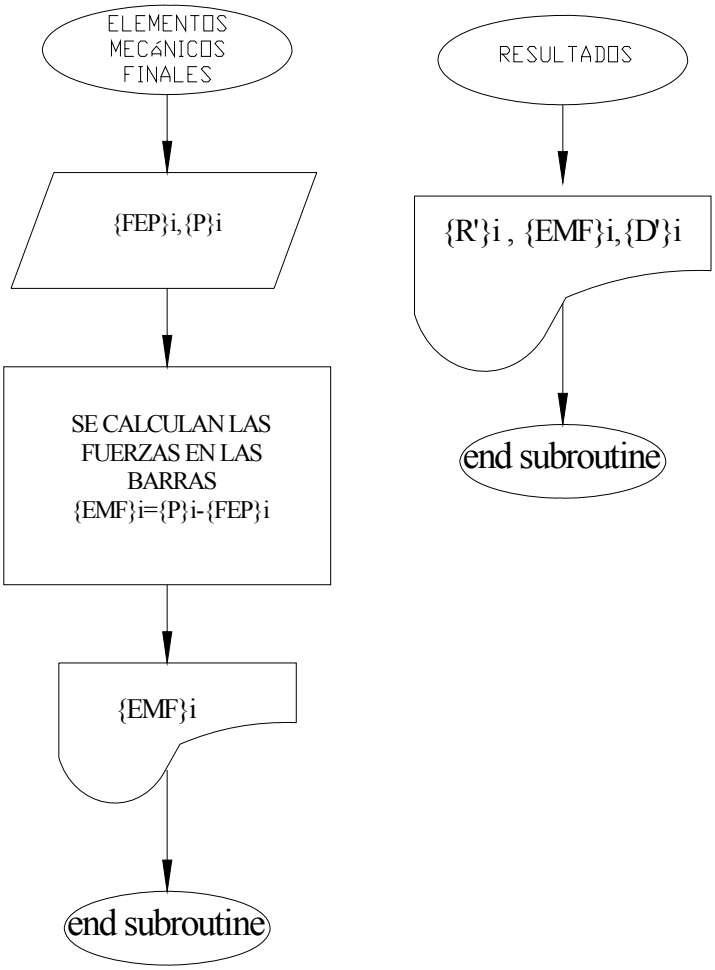

MAIN PROGRAM
INICIO
FILE=DATOS
FILE=E.FZA-DESP-BARRA
FILE=RESULTADOS
CALL DATOS
nb,na,nc,cc,
cd,ni,nf, gl, E,A,I,coord
CALL SUBMATRICES DE RIGIDEZ
E,I,A
CALL MATRIZ ENSAMBLADA
[k'11]i,[k'12]i,[k'21]i,[k'22]i,
[ME], [MER],{FEP}, {FN}
CALL SOL. DEL SISTEMA
[MER], {D'}, {FN}
FUERZAS EN LAS BARRAS (S.G.)
[K']i, <D'>i, <P'>
CALL REACCIONES EN LOS APOYOS
{FEP'}i,{P'}i
CALL ROTACION AL S.LOCAL
[T],{P'}i
CALL EMF
{FEP}i,{P}i
RESULTADOS
{R'}i, {EMF}i,{D'}i
nb,na,nc
cc,cd,ni,nf,gl,E,A,I
FIN


DATOS
nn, nb,na,nc
cc,cd,ni,nf,gl,E,A,I
nn = NÚMERO DE NUDOS
nb = NÚMERO DE BARRAS
na = NÚMERO DE APOYOS Y RESTRICCIONES
gl
-PROPIEDADES
A, E, I
nc = NÚMERO DE NUDOS CARGADOS
cc= NÚMERO DE CARGAS CONCENTRADAS
cd = NÚMERO DE CARGAS DISTRIBUIDAS
-CONECTIVIDAD
ni, nf
nnr(i)
nnry
nnrz
nb,w,FN,
Fvi , Fmi
CONDICIONES DE CARGA DE CADA BARRA
Fv (i) = FEP
Fm (i) = MEP
nb,na,nc
cc,cd,ni,nf, lb, gl,E,A,I
end subroutine


SUBMATRICES DE RIGIDECES
(PARA CADA BARRA)
nb,lbi,Ei,Ii,
k11i,k12i,k21i,k22i,
FEPi, FNi
i = 1 : nb
MATRIZ DE
RIGIDEZ
end subroutine
MATRIZ
ENSAMBLADA
[k'11]i,[k'12]i,[k'21]i,[k'22]i,
[ME], [MER],{FEP}, {FN}
MATRIZ
ENSAMBLADA
MATRIZ
ENSAMBLADA
REDUCIDA
end subroutine


SOLUCION DEL SISTEMA
[MER], {D'}, {FN}
SE RESULEVE LA ECUACIÓN FUERZA-DESPLAZAMIENTO ENSAMBLADA REDUCIDA {D'}nn
end subroutine
FZAS. EN LAS BARRAS EN S.GLOBAL
[K]i,D'i
Ecuación fuerza desplazamiento
SE RESULEVE LA ECUACIÓN FUERZA-DESPLAZAMIENTO ENSAMBLADA REDUCIDA {P'}nn
end subroutine


REACCIONES EN LOS APOYOS
{FEP'}i,{P'}i
SE CALCULAN LAS REACCIONES EN LOS APOYOS {R'}i={P'}i-{FEP'}i
{R'}i
end subroutine
ROTACION DEL SISTEMA GLOBAL AL LOCAL
[T],{P'}i
SE HACE LA ROTACION {P}i={T}t{P'}i
{P}i
end subroutine


ELEMENTOS MECáNICOS FINALES
{FEP}i,{P}i
SE CALCULAN LAS FUERZAS EN LAS BARRAS
{EMF}i={P}i-{FEP}i
{EMF}i
end subroutine
RESULTADOS
{R'}i , {EMF}i,{D'}i
end subroutine

3.2 MANUAL DE USUARIO

PROCEDIMIENTO PARA LA EJECUCIÓN DEL PROGRAMA

El programa RIGIDECES.EXE se puede ejecutar de manera óptima mediante los siguientes pasos:

PASO 1: Se debe abrir el programa ejecutable de RIGIDECES.EXE, el cuál primeramente nos pedirá introducir el nombre del archivo de datos en el cual tenemos todos los datos que el programa requiere para hacer el cálculo correspondiente, el cual debe tener la siguiente forma:

Pasos a seguir para escribir el archivo de datos.

Al escribir se debe dejar un espacio entre cada dato ingresado, además no se deben omitir datos aún en aquellos que se deben especificar poniendo cero.

Antes de ingresar los datos se debe tener cuidado en que éstos sean congruentes en unidades. Además, debemos hacer coincidir el eje local de la viga con el eje global, esto lo hacemos analizando la viga de izquierda a derecha.

Primer paso

En el primer renglón escribiremos los datos siguientes mencionados en orden de izquierda a derecha:

nn nb na np nc

nn: número de nodos.

nb: número de barras.

na: número de apoyos.

np: número de propiedades.

nc: número nodos cargados.

Segundo paso.

En este renglón escribiremos el número cargas a las cuales se encuentran sometidas las barras:

cc cd

cc: número de cargas concentradas.

cd: número de cargas distribuidas.

Tercer paso.

En los siguientes renglones escribiremos el número de nodo con su respectiva coordenada en el eje "x".

Núm. de nodo Coordenada en "x"

Cuarto paso.

En los siguientes renglones escribiremos el número de propiedad y sus correspondientes Módulos de elasticidad e Inercia.

Núm. de propiedad E I

Quinto paso.

En los siguientes renglones escribiremos la conectividad y las propiedades de las barras, primero escribimos el número de la barra, luego su nodo inicial, después su nodo final y finalmente su número de propiedad.

Núm. de barra ni nf Núm. de propiedad

ni: nodo inicial

nf: nodo final

Sexto paso.

En los siguientes renglones escribiremos el número nodo que es apoyo y sus restricciones de desplazamiento en el eje "y" y de giro alrededor del eje "z". Para especificar que el desplazamiento es libre se pondrá un cero en la posición correspondiente y para indicar que está restringido un uno.

Núm. de apoyo Restricción en "y" Restricción en "z"

Séptimo paso. (Este se realizará sólo si existen cargas en los nodos)

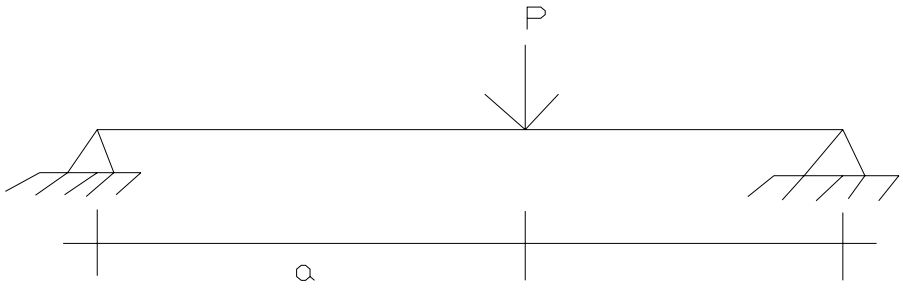
En los siguientes renglones escribiremos el número de nodo sometido a carga primero su carga en dirección del eje "y" y después el momento alrededor del eje "z". En caso de que solo exista un tipo de carga en la posición que no exista se anotará un cero.

Núm. de nodo cargado Py Mz

Octavo paso. (Este se realizará sólo si existen cargas concentradas)

En los siguientes renglones escribiremos el número de barra con carga concentrada, después la distancia entre la carga y el apoyo izquierdo de la barra y finalmente el valor de la carga, esto para cada carga concentrada.

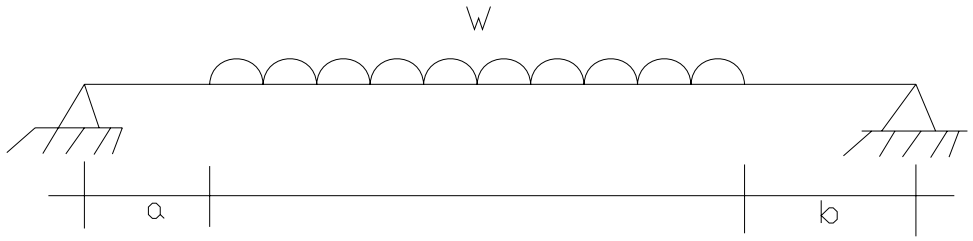
Núm. de barra Distancia a P



Noveno paso. (Este se realizará sólo si existen cargas distribuidas)

En los siguientes renglones escribiremos el número de la barra con carga distribuida, luego la distancia entre la carga y el apoyo izquierdo de la barra, después la distancia de la carga y el apoyo derecho de la carga, finalmente el valor de la carga, esto para cada carga distribuida.
Para el caso cuando tenemos la carga distribuida a todo lo largo de la barra solo tenemos que introducir ceros en ambas distancias.

Núm. de barra **Distancia a** **Distancia b** **W**



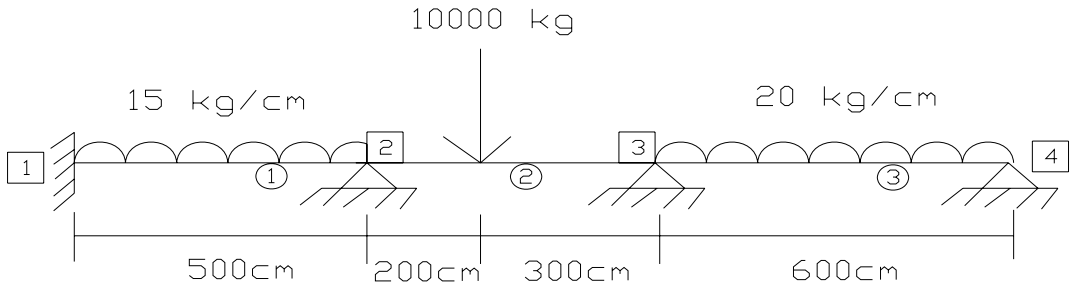
PASO 2: Introducimos el nombre del archivo de datos creado con anticipación.

PASO 3: Introducimos el nombre del archivo en el cual queremos guardar los resultados del calculo.

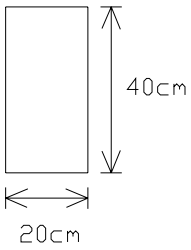
Finalmente, una vez que el programa se ha ejecutado satisfactoriamente podemos ver los resultados en el archivo de resultados.

3.3 EJEMPLOS DE APLICACIÓN

EJEMPLO 1 (Resuelto a mano)



Propiedades:



Base = 20 cm

Peralte= 40 cm

I= 106666.667 cm^4

Barras 1 y 3.

E = 113 137 kg/cm^2

Barra 2.

E= 158 000 kg/cm^2

Matriz de rigidez de cada barra.

Barra 1

$$K'1=\begin{pmatrix} 1158.523 & 289630.720 & -1158.523 & 289630.720 \\ 289630.720 & 96543573.333 & -289630.720 & 48271786.667 \\ -1158.523 & -289630.720 & 1158.523 & -289630.720 \\ 289630.720 & 48271786.667 & -289630.720 & 96543573.333 \end{pmatrix}$$

Barra 2

$$K'2=\begin{pmatrix} 1617.920 & 404480.000 & -1617.920 & 404480.000 \\ 404480.000 & 134826666.667 & -404480.000 & 67413333.333 \\ -1617.920 & -404480.000 & 1617.920 & -404480.000 \\ 404480.000 & 67413333.333 & -404480.000 & 134826666.667 \end{pmatrix}$$

Barra 3

$$K'3=\begin{pmatrix} 670.441 & 201132.444 & -670.441 & 201132.444 \\ 201132.444 & 80452977.778 & -201132.444 & 40226488.889 \\ -670.441 & -201132.444 & 670.441 & -201132.444 \\ 201132.444 & 40226488.889 & -201132.444 & 80452977.778 \end{pmatrix}$$

Momentos de empotramiento perfecto para cada barra.

Barra 1

$Vy_1 = 3750$

$Vy_2 = 3750$

Barra 2

$Vy_2 = 6480$

$Vy_3 = 3520$

Barra 3

$Vy_3 = 6000$

$Vy_4 = 6000$

Los momentos con sentido antihorario son positivos y los de sentido horario negativos.

Matriz ensamblada reducida del sistema.

$$\begin{array}{c} \begin{matrix} |\varphi 2| & |\varphi 3| & |\varphi 4| \end{matrix} \\ \begin{pmatrix} 231370240.000 & 67413333.333 & 0.000 \\ 67413333.333 & 215279644.444 & 40226488.889 \\ 0.000 & 40226488.889 & 80452977.778 \end{pmatrix} \end{array} \begin{matrix} |\varphi 2| \\ |\varphi 3| \\ |\varphi 4| \end{matrix}$$

Aplicando la ecuación fuerza desplazamiento y despejando los desplazamientos tenemos:

$$P = K\, d$$

Con los momentos de empotramiento perfecto calculados obtenemos el vector de fuerzas:

$$P = \begin{pmatrix} 720000-312500 \\ 600000-480000 \\ -600000 \end{pmatrix} = \begin{pmatrix} 407500 \\ 120000 \\ -600000 \end{pmatrix}$$

$$\begin{pmatrix} -407500 \\ -120000 \\ 600000 \end{pmatrix} = \begin{pmatrix} 231370240.000 & 67413333.333 & 0.000 \\ 67413333.333 & 215279644.444 & 40226488.889 \\ 0.000 & 40226488.889 & 80452977.778 \end{pmatrix} \begin{pmatrix} \varphi 2 \\ \varphi 3 \\ \varphi 4 \end{pmatrix}$$

$$\begin{pmatrix} \varphi 2 \\ \varphi 3 \\ \varphi 4 \end{pmatrix} = \begin{pmatrix} -0.001261149 \\ -0.00171639 \\ 0.008315968 \end{pmatrix}$$

Desplazamientos en radianes

Cálculo de los elementos mecánicos de cada barra.

$$EMF = K' * d + FEP$$

Barra1

$$EMF = \begin{pmatrix} 1158.52288 & 289630.72 & -1158.52288 & 289630.72 \\ 289630.720 & 96543573.333 & -289630.720 & 48271786.66 \\ -1158.523 & -289630.720 & 1158.523 & -289630.720 \\ 289630.720 & 48271786.667 & -289630.720 & 96543573.333 \end{pmatrix} \begin{pmatrix} 0.00000000 \\ 0.00000000 \\ 0.00000000 \\ -0.001261149 \end{pmatrix} + \begin{pmatrix} 3750 \\ 312500 \\ 3750 \\ -312500 \end{pmatrix}$$

$$\begin{pmatrix} Vy_1 \\ M_1 \\ Vy_2 \\ M_2 \end{pmatrix} = \begin{pmatrix} 3384.732434 \\ 251622.0723 \\ 4115.267566 \\ -434255.8554 \end{pmatrix}$$

Barra 2

$$EMF=\begin{pmatrix} 1617.92 & 404480 & -1617.92 & 404480 \\ 404480 & 134826666.7 & -404480 & 67413333.33 \\ -1617.92 & -404480 & 1617.92 & -404480 \\ 404480 & 67413333.33 & -404480 & 134826666.7 \end{pmatrix} \begin{pmatrix} 0.00000000 \\ -0.001261149 \\ 0.000000000 \\ -0.00171639 \end{pmatrix} + \begin{pmatrix} 6480 \\ 720000 \\ 3520 \\ -480000 \end{pmatrix}$$

$$\begin{pmatrix} Vy_2 \\ M_2 \\ Vy_3 \\ M_3 \end{pmatrix} = \begin{pmatrix} 5275.644782 \\ 434255.8554 \\ 4724.355218 \\ -796433.4642 \end{pmatrix}$$

Barra 3

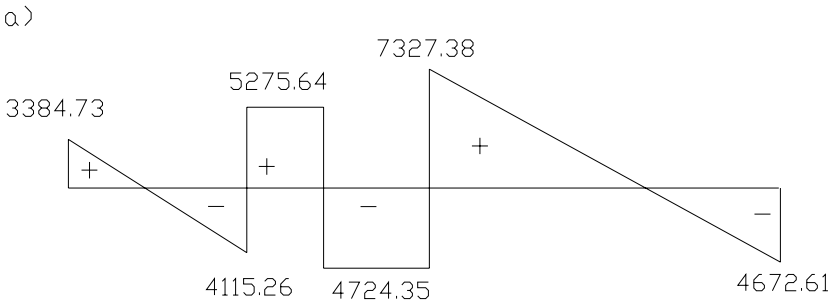
$$EMF=\begin{pmatrix} 670.4414815 & 201132.4444 & -670.4414815 & 201132.4444 \\ 201132.4444 & 80452977.78 & -201132.4444 & 40226488.89 \\ -670.4414815 & -201132.4444 & 670.4414815 & -201132.4444 \\ 201132.4444 & 40226488.89 & -201132.4444 & 80452977.78 \end{pmatrix} \begin{pmatrix} 0.00000000 \\ -0.00171639 \\ 0.00000000 \\ 0.008315968 \end{pmatrix} + \begin{pmatrix} 6000 \\ 600000 \\ 6000 \\ 600000 \end{pmatrix}$$

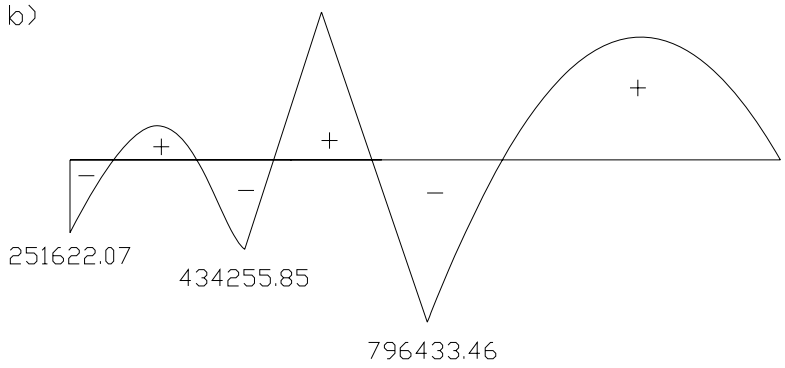
$$\begin{pmatrix} Vy_3 \\ M_3 \\ Vy_4 \\ M_4 \end{pmatrix} = \begin{pmatrix} 7327.389107 \\ 796433.4642 \\ 4672.610893 \\ 0.000 \end{pmatrix}$$

Diagramas de cortante y momento:

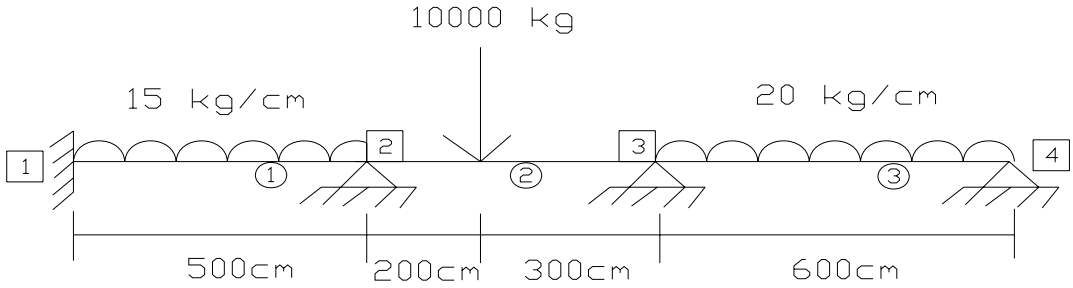
a) Fuerza cortante (kg).

b) Momento flexionante (kg-cm).

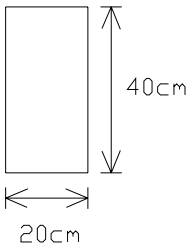




EJEMPLO 1. (Resuelto con el programa)



PROPIEDADES.



Base = 20 cm

Peralte = 40 cm

I=106666.667 cm^4

Barras 1 y 3.

E = 113 137 kg/cm^2

Barra 2.

E= 158 000 kg/cm^2

FORMATO DEL ARCHIVO DE DATOS PARA LA VIGA DEL EJEMPLO 1.

```
4   3   4   2   0
1   2
1   0
2   500
3   1000
4   1600
1   113137   106666.667
2   158000   106666.667
1   1   2   1
2   2   3   2
3   3   4   1
1   1   1
2   1   0
3   1   0
4   1   0
2   200   -10000
1   0   0   -15
3   0   0   -20
```

El nombre del archivo de datos del ejemplo 1 es: VIGA1.TXT

El nombre del archivo de resultados del ejemplo 1 es: RESV1.TXT

ARCHIVO DE DATOS INGRESADO: VIGA1.TXT

TIPO DE ESTRUCTURA: "VIGA"

Nodos..... 4

Barras..... 3

Apoyos..... 4

Propiedades..... 2

Nudos cargados..... 0

Cargas concentradas.. 1

Cargas distribuidas.. 2

COORDENADAS DE LOS NODOS

Nodo	X
1	.0000
2	500.0000
3	1000.0000
4	1600.0000

PROPIEDADES DE LAS BARRAS

BARRA	Nodo Inicial	Nodo Final	E	I
1	1	2	113137.000	106666.667
2	2	3	158000.000	106666.667
3	3	4	113137.000	106666.667

APOYOS

Nodo	Desplazamiento en Y	Giro alrededor de Z
1.	RESTRINGIDO	RESTRINGIDO
2.	RESTRINGIDO	LIBRE
3.	RESTRINGIDO	LIBRE
4.	RESTRINGIDO	LIBRE

CARGAS CONCENTRADAS EN BARRAS

Barra	a	P
2	200.000	-10000.000

CARGAS DISTRIBUIDAS EN LAS BARRAS

Barra	a	b	W
1	.000	.000	-15.000
3	.000	.000	-20.000

DESPLAZAMIENTOS DE LOS NUDOS

NUDO	Y	Z
1	.0000000000	.0000000000
2	.0000000000	-.1681532338
3	.0000000000	-.2288520463
4	.0000000000	1.1087956812

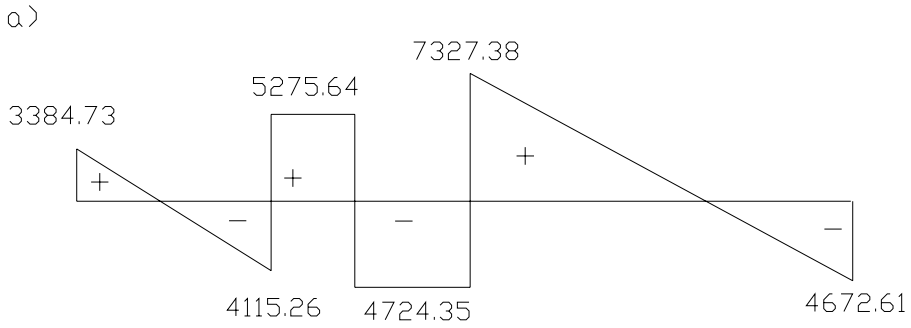
FUERZAS EN LAS BARRAS

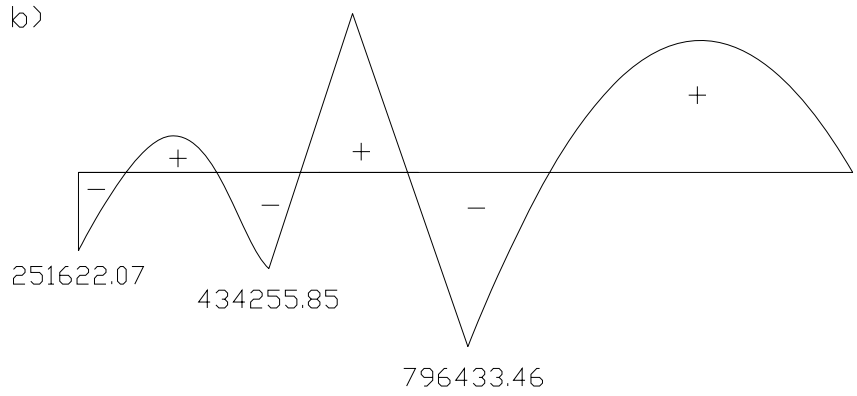
BARRA	NUDO	CORTANTE	MOMENTO
1	1	3384.7324	251622.0723
	2	4115.2676	-434255.8554
2	2	5275.6448	434255.8554
	3	4724.3552	-796433.4642
3	3	7327.3891	796433.4642
	4	4672.6109	.0000

Diagramas de cortante y momento:

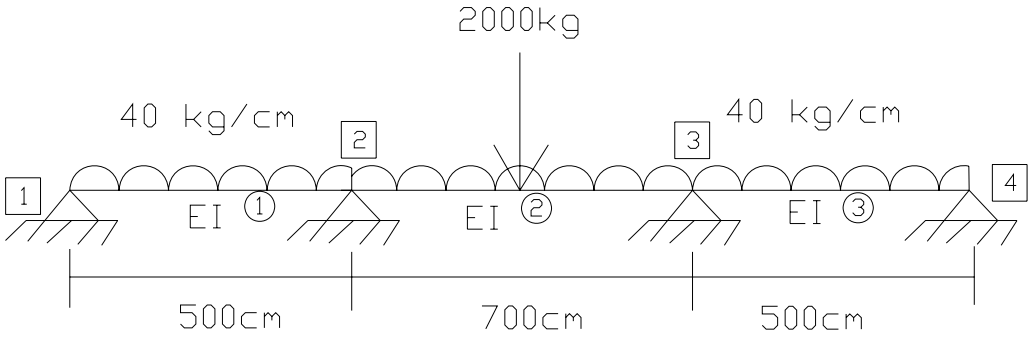
a) Fuerza cortante (kg).

b) Momento flexionante (kg-cm).

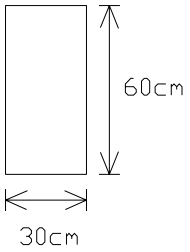




EJEMPLO 2.



PROPIEDADES.



Base = 30 cm

Peralte = 60 cm

I=540000 cm^4

E = 158 000 kg/cm^2

FORMATO DEL ARCHIVO DE DATOS PARA LA VIGA DEL EJEMPLO 2.

```
4   3   4   1   0
1   3
1   0
2   500
3   1200
4   1700
1   158000   540000
1   1   2   1
2   2   3   1
3   3   4   1
1   1   0
2   1   0
3   1   0
4   1   0
2   350   -2000
1   0   0   -40
2   0   0   -40
3   0   0   -40
```

El nombre del archivo de datos del ejemplo 2 es: VIGA2.TXT

El nombre del archivo de resultados del ejemplo 2 es: RESV2.TXT

ARCHIVO DE DATOS INGRESADO: VIGA2.TXT

TIPO DE ESTRUCTURA: "VIGA"

Nodos..... 4

Barras..... 3

Apoyos..... 4

Propiedades..... 1

Nudos cargados..... 0

Cargas concentradas.. 1

Cargas distribuidas.. 3

COORDENADAS DE LOS NODOS

Nodo	X
1	.0000
2	500.0000
3	1200.0000
4	1700.0000

PROPIEDADES DE LAS BARRAS

BARRA	Nodo Inicial	Nodo Final	E	I
1	1	2	158000.000	540000.000
2	2	3	158000.000	540000.000
3	3	4	158000.000	540000.000

APOYOS

Nodo	Desplazamiento en Y	Giro alrededor de Z
1.	RESTRINGIDO	LIBRE
2.	RESTRINGIDO	LIBRE
3.	RESTRINGIDO	LIBRE
4.	RESTRINGIDO	LIBRE

CARGAS CONCENTRADAS EN BARRAS

Barra	a	P
2	350.000	-2000.000

CARGAS DISTRIBUIDAS EN LAS BARRAS

Barra	a	b	W
1	.000	.000	-40.000
2	.000	.000	-40.000
3	.000	.000	-40.000

DESPLAZAMIENTOS DE LOS NUDOS

NUDO	Y	Z
1	.0000000000	-.2554425087
2	.0000000000	-.2216513165
3	.0000000000	.2216513165
4	.0000000000	.2554425087

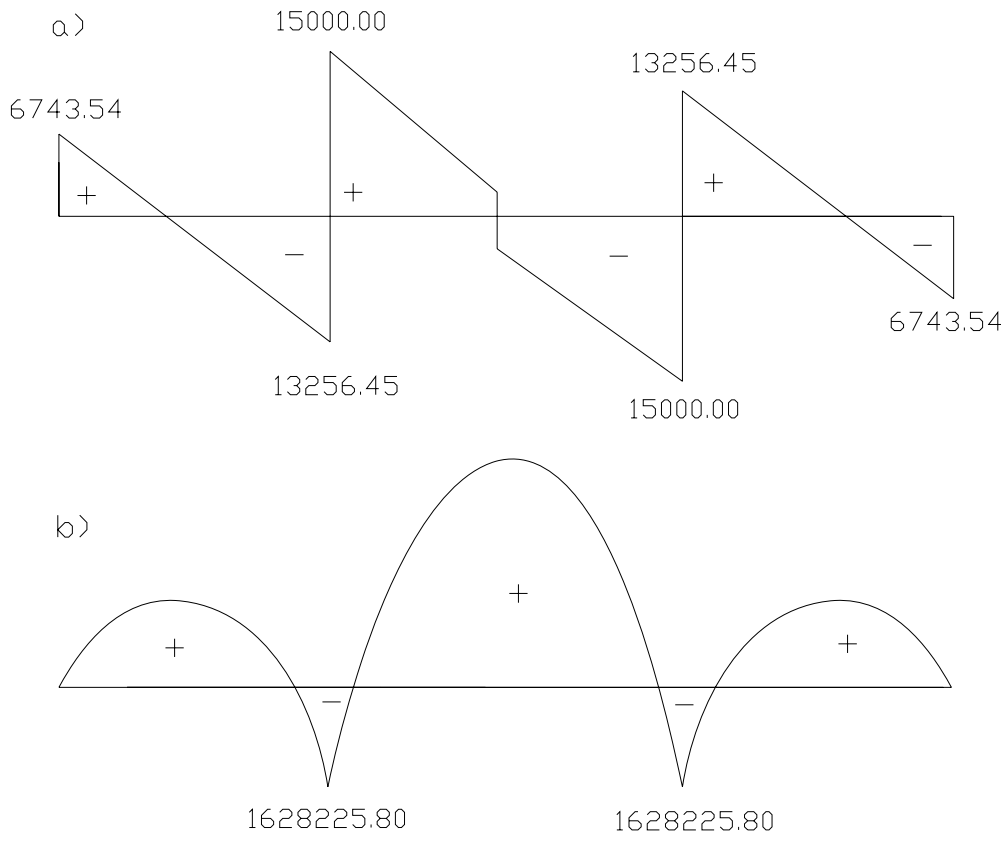
FUERZAS EN LAS BARRAS

BARRA	NUDO	CORTANTE	MOMENTO
1	1	6743.5484	.0000
	2	13256.4516	-1628225.8065
2	2	15000.0000	1628225.8065
	3	15000.0000	-1628225.8065
3	3	13256.4516	1628225.8065
	4	6743.5484	.0000

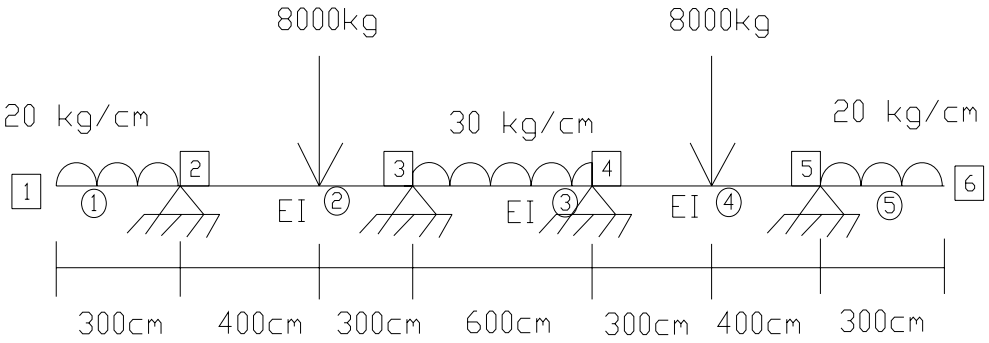
Diagramas de cortante y momento:

a) Fuerza cortante (kg).

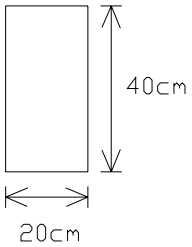
b) Momento flexionante (kg-cm).



EJEMPLO 3.



PROPIEDADES.



Base = 20 cm

Peralte = 40 cm

$I=106666.667\ cm^4$

$E = 158\ 000\ kg/cm^2$

FORMATO DEL ARCHIVO DE DATOS PARA LA VIGA DEL EJEMPLO 3.

```
6   5   4   1   0
2   3
1   0
2   300
3   1000
4   1600
5   2300
6   2600
1   158000   106666.667
1   1   2   1
2   2   3   1
3   3   4   1
4   4   5   1
5   5   6   1
2   1   0
3   1   0
4   1   0
5   1   0
2   400   -8000
4   300   -8000
1   0   0   -20
3   0   0   -30
5   0   0   -20
```

El nombre del archivo de datos del ejemplo 3 es: VIGA3.TXT

El nombre del archivo de resultados del ejemplo 3 es: RESV3.TXT

CONCLUSIONES

Con este programa podemos realizar cálculos en un menor tiempo, con mejor exactitud en cuanto a los valores que obtenemos y de manera fácil calculamos los elementos mecánicos que actúan en una viga, para así poder realizar su diseño.

En la actualidad es de vital importancia el tiempo en se lleva a cabo el cálculo y todo trabajo que al Ingeniero se le asigne, pues la competencia no está sólo en la parte económica sino también en el factor tiempo.

Por lo anterior cabe mencionar que para la Ingeniería Civil la programación es una herramienta que va de la mano con el cálculo y resolución de problemas, por ello es importante que el Ingeniero sepa la importancia de saber programar para que mejore su trabajo y desempeño.

GLOSARIO

{d}	Vector de desplazamientos global.
{d'}	Vector de desplazamientos local.
|K|	Matriz de rigidez de la estructura.
|K'|	Matriz de rigidez del elemento.
Mx	Momento alrededor del eje x.
My	Momento alrededor del eje y.
Mz	Momento alrededor del eje z.
Nx	Fuerza normal en el eje x.
{P}	Vector de cargas global.
{P'}	Vector de cargas del elemento o elementos mecánicos.
Vy	Fuerza cortante en el eje y.
Vz	Fuerza normal en el eje z.
x,y	Ejes coordenados locales de una barra.
x',y'	Ejes coordenados globales o de la estructura.

BIBLIOGRAFÍA

HIBBELER R.C., "Análisis Estructural", 3ª Ed., Editorial Prentice Hall Hispanoamericana S.A., México, 1997.

MEISSNER L.P., "Fortran 90", Editorial PWS Publishing Company, E.U.A., 1995.

ROJAS ROJAS R., "Método de las Rigideces".

SÁNCHEZ IBARRA A., "Programación en Fortran".