Facultad de Ingeniería Civil

Programacion del metodo de las rigideces apliacado a vigas en el plano

Tesis para obtener el titulo de ingeniero civil

Presenta:
Marcos Jubal Tovar Vázquez

Asesor
M. en I. Enrique Omar Navarro Caballero

Morelia, Michoacán, Mayo de 2008

Índice

Introducción

En la actualidad el uso de las computadoras revolucionó el mundo del análisis estructural debido a que mediante la ayuda de éstas se pueden realizar una serie de operaciones en unos cuantos segundos. En el pasado cuando era necesario analizar ciertas estructuras se utilizaban métodos que no necesitarán realizar una gran cantidad de operaciones muy complejas.

Los métodos exactos o matriciales han tenido un gran auge en las últimas tres décadas debido a la comercialización de las computadoras y a que éstas son cada vez más poderosas y sencillas de usar. Además, hay que sumarle el fenómeno del uso de estructuras con un alto índice de complejidad.

Los métodos matriciales proporcionan un lenguaje matemático muy adecuado para la descripción de un sistema estructural que puede ser resuelto fácilmente por las computadoras.

Como se ha mencionado sin el uso de las computadoras era muy difícil hacer el análisis de estructuras mediante los modelos de matrices, ya que los sistemas son bastante grandes y difíciles de trabajar a mano. Los conceptos aplicados para los métodos matriciales no son nuevos son los mismos ya conocidos de la mecánica de sólidos, de tal manera que para utilizar estos métodos, sólo se requiere conocer las bases de la mecánica de sólidos y el cálculo matricial.

Lo anterior nos lleva a concluir que los métodos matriciales son la mejor opción para el análisis estructural, debido a que presentan una forma fácil de ser programados en una computadora.
Esta parte nos lleva al tema central de este trabajo, el cual es el análisis estructural de vigas por medio de un método matricial conocido como método de los desplazamientos o de las rigideces, tomando en cuenta que las vigas están el plano y que además, estas solamente resisten fuerza cortante y momento flexionante sin presentarse fuerza normal

OBJETIVO

El objetivo de este trabajo es probar un programa realizado para el análisis de vigas en el plano y demostrar lo practico y sencillo que se puede volver el método de los desplazamientos o de las rigideces y lo sencillo que es programarlo con algún lenguaje de programación, que para este caso fue FORTRAN.

Capítulo 1 El Método De Los Desplazamientos o De Las Rigideces

En método de los desplazamientos también conocido como de las rigideces, los desplazamientos son las primeras incógnitas, para encontrar estos se usan un conjunto de ecuaciones simultáneas. Después de resolver estas ecuaciones y determinar los desplazamientos, estos se sustituyen en las relaciones fuerza-deformación de cada elemento para determinar las diversas fuerzas internas.

Hay que establecer una serie de conceptos básicos antes de hablar del método de la rigidez y son:

- Elementos: son las partes que constituyen el sistema estructural que se esta representando.
- Nodos: son los lugares en la estructura donde se conectan los elementos.
- Coordenada: una coordenada estructural es un desplazamiento posible o grado de libertad en un nodo de la estructura. En las estructuras planas hay tres coordenadas en cada nodo. Estos desplazamientos posibles o grados de libertad son dos traslaciones ortogonales y una rotación respecto a un eje perpendicular al plano definido por las traslaciones. En una estructura espacial hay seis coordenadas por cada nodo: tres traslaciones ortogonales y tres rotaciones. No debe de confundirse una coordenada estructural con una coordenada cartesiana, que describe la posición de un punto en el espacio.
- Fuerza: el termino "fuerza" es un termino general que se refiere a una fuerza que actúa en una coordenada traslacional o a un momento que actúa en una coordenada rotacional. No hay distinción respecto a si la fuerza es una carga estructural conocida o una fuerza de reacción desconocida.
- Desplazamiento: el termino "desplazamiento" es un termino general que se refiere a la translación de una coordenada traslacional o a la rotación en una coordenada rotacional. No hay distinción respecto si es un desplazamiento conocido o un desplazamiento desconocido en el nodo.
- Rigidez: la rigidez es la fuerza requerida para generar una deformación unitaria en un material elástico.

En general, un sólido deformable real, como cualquier medio continuo es un sistema físico con un número infinito de grados de libertad. Así sucede que, en general, para describir la deformación de un

sólido es necesario explicitar un campo vectorial de desplazamientos sobre cada uno de sus puntos. Este campo de desplazamientos en general no es reductible a un número finito de parámetros, y por tanto un sólido deformable de forma totalmente general no tiene un número finito de grados de libertad.

Sin embargo, para barras largas elásticas o prismas mecánicos de longitud grande comparada con el área de su sección transversal el campo de desplazamientos viene dado por la llamada curva elástica cuya deformación siempre es reductible a un conjunto finito de parámetros. En concreto, fijados los desplazamientos y giros de las secciones extremas de una barra elástica queda completamente determinada su forma. Así, para una estructura formada por barras largas elásticas, fijados los desplazamientos de los nudos queda completamente determinada la forma deformada de dicha estructura. Esto hace que las estructuras de barras largas puedan ser tratadas muy aproximadamente mediante un número finito de grados de libertad y que puedan ser calculadas resolviendo un número finito de ecuaciones algebraicas. El método matricial proporciona esas ecuaciones en forma de sistema matricial que relaciona los desplazamientos de los extremos de la barras con variables dependientes de las fuerzas exteriores.

Este es un método donde las estructuras están formadas por elementos barra debido a que muchas de las estructuras más comunes tienen este tipo de elementos y por lo tanto este modelo se acerca a la realidad de cómo trabaja la estructura.

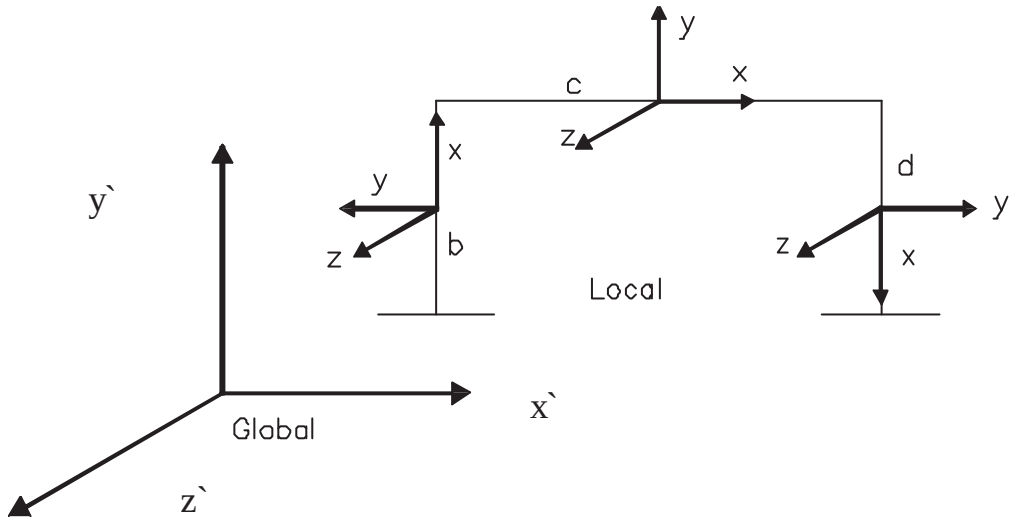
El método de la rigidez o de los desplazamientos establece a través de una serie de ecuaciones las relaciones que hay entre las cargas y los desplazamientos de la estructura. Entonces, al haber planteado esa serie de ecuaciones podemos encontrar el valor de los desplazamientos en los nodos y también los elementos mecánicos de las barras que conforman la estructura en estudio. Por lo tanto, lo primero que tenemos que obtener son los desplazamientos que son generados por las fuerzas externas en la estructura y después obtenemos los elementos de las barras.

Vale la pena mencionar que es el método más adecuado para su programación, y todos los paquetes formales para el análisis estructural en computadora lo utilizan.

SISTEMAS DE REFERENCIA

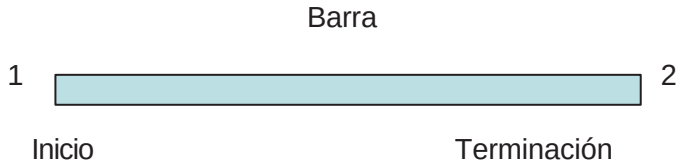
En este método tenemos que usar sistemas de coordenadas para toda la estructura y también un sistema coordenado para cada elemento. Al sistema coordenado de toda la estructura se le conoce como el sistema global y al de cada elemento se le conoce como local. Dado que las estructuras se encuentran en un espacio tridimensional entonces usamos ejes **x', y' ,z'** para el sistema global y (**x, y ,z**) para el sistema local.

Se debe de mencionar que la ubicación del eje x en el sistema local tiene que coincidir con el eje longitudinal de la barra y, dependiendo de éste, los otros ejes se establecen considerando un sistema coordenado derecho, ósea giramos noventa grados hacia la izquierda de donde establecimos el eje x.



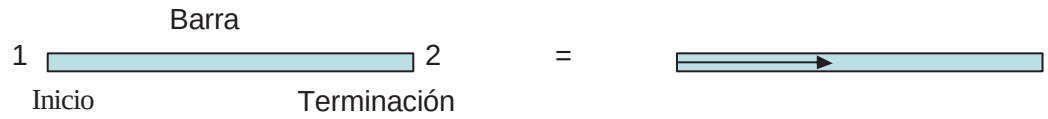
NOTACION DE LAS BARRAS

Tomaremos una barra cualquiera en la cual se indica con el número uno el extremo inicial y con el numero dos indica el extremo final de la barra y se muestra en la figura.

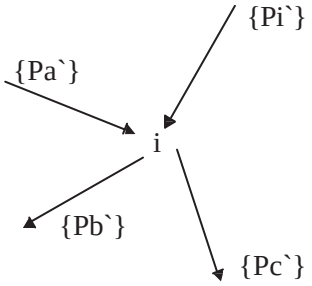


Cuando a un nodo llegan varias barras no se recomienda que se use la notación de la figura anterior debido a que para un nodo tendríamos una gran cantidad de números y por lo tanto se generaría una gran confusión, entonces se opta por utilizar otro tipo de notación la cual consiste en poner en la barra

una flecha donde el inicio de la flecha seria el inicio o el extremo uno y hacia donde indica seria el final o extremo dos de la barra.



Es necesario, para usar el método de los desplazamientos, tomar en cuenta que se debe cumplir con el equilibrio y la compatibilidad en toda la estructura. En la siguiente figura podremos observar esta parte del equilibrio y la continuidad, digamos que hay un nudo "i" al cual llegan ciertas barras y además este esta sometido a un vector de cargas que llamaremos $\{P_i'\}$.



Por equilibrio:

$$\{P_i'\} = \{P_{2a}'\} + \{P_{1b}'\} + \{P_{1c}'\}$$

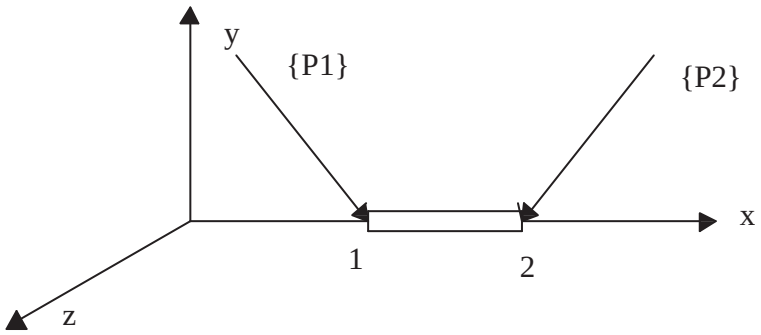
De la ecuación de equilibrio podemos observar que el vector de cargas en el nodo "i" es igual a la sumatoria de las fuerzas que se generan en el extremo de cada barra que llega al nodo "i".

Por compatibilidad:

$$\{d_i'\} = \{d_{2a}'\} = \{d_{1b}'\} = \{d_{1c}'\}$$

De la ecuación de compatibilidad podemos observar que el vector de desplazamiento de un nodo es igual al vector de desplazamiento de los extremos de las barras que llegan a ese mismo nodo.

Ahora si tomamos en cuenta que la barra se encuentra en un espacio tridimensional que esta sujeta a vectores de carga $\{P_1\}$y $\{P_2\}$ el subíndice indica el extremo de la barra donde se encuentran aplicados estos vectores como se ve en la figura siguiente, estos vectores de carga generan desplazamientos los cuales serian $\{d_1\}$ y $\{d_2\}$.



Si encontramos entonces una relación entre las cargas y los desplazamientos de la estructura esto se puede hacer a través de una matriz de coeficientes que se le llama matriz de rigideces de la barra como se observa en la siguiente figura.

$$\begin{Bmatrix} P_1 \\ P_2 \end{Bmatrix} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix}$$

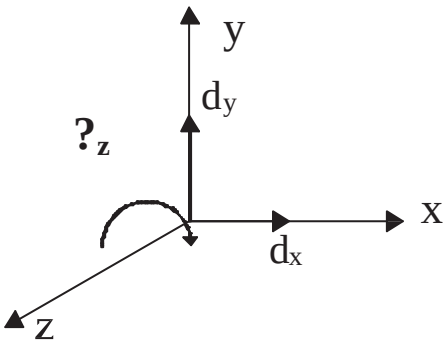
Podemos expresarlo como:

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

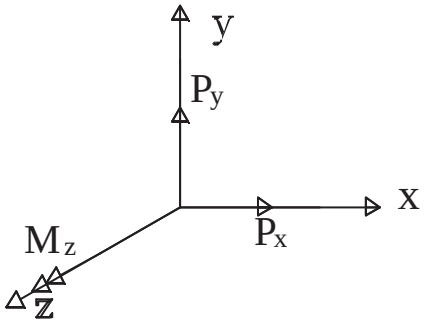
La ecuación anterior es la forma compacta de la otra ecuación que esta presentada en forma matricial y ambas representan la ecuación fuerza-desplazamiento de una barra en el sistema local.

DETERMINACIÓN DE LA MATRIZ DE RIGIDECES

La rigidez es la magnitud de una fuerza para producir un desplazamiento unitario. Para este caso tomaremos sólo una barra que se encuentra en el plano **"xy",** en la cual los extremos de los elementos se pueden desplazar tanto lineal como angularmente y en las direcciones de de cada eje, por lo tanto el vector de desplazamientos de esta barra tendrá tres componentes como se muestra en la siguiente figura



Como también la fuerza en un elemento barra en el plano esta también tiene componentes como el caso del desplazamiento una en cada eje y una de momento en z, por lo cual tenemos tres componentes en cada extremo de la barra para la fuerza como se muestra en la figura, donde cambia un poco la notación siendo las flechas con doble punta desplazamientos angulares y las flechas con una sola punta son desplazamientos lineales.



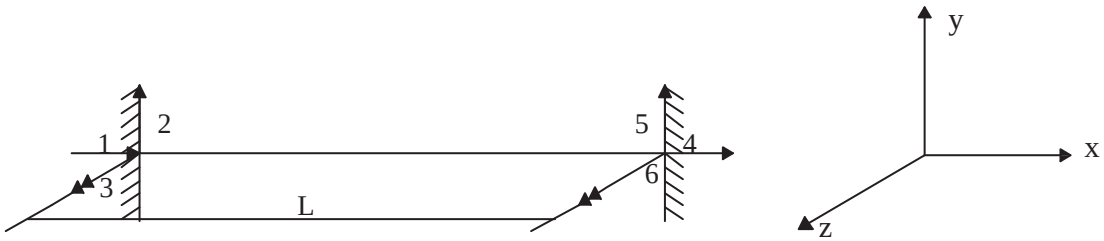
Si lo ponemos de una forma matricial el resultado es el siguiente:

$$\{d\} = \begin{Bmatrix} d_x \\ d_y \\ \phi_z \end{Bmatrix} \qquad \{P\} = \begin{Bmatrix} P_x \\ P_Y \\ M_Z \end{Bmatrix}$$

Entonces por cada tipo de fuerza vamos a tener un tipo de rigidez, por lo cual, nos encontraremos tantos tipos de rigidez como de elementos mecánicos.

Para obtener las rigideces de una manera sencilla se considera un elemento empotrado, al cual se le inducen desplazamientos ya sean lineales o angulares unitarios.

Los signos de los desplazamientos para cada elemento de la estructura se le darán según los ejes de referencia de este.



a b

En la figura anterior las flechas con una sola punta indican que se trata de un desplazamiento lineal y las que tienen dos puntas se trata de un desplazamiento angular. Donde, en el extremo izquierdo los números uno y dos son desplazamientos lineales y el tres es desplazamiento angular. De forma similar se hace lo mismo en el extremo derecho donde los números cuatro y cinco son desplazamientos lineales y el seis es desplazamiento angular. Entonces, el orden es: primero los desplazamientos lineales y después los angulares siguiendo el orden de los ejes **x, y, z** Lo único que indica la figura anterior en "b" es el sistema de referencia.

Las rigideces de obtienen a partir de las relaciones que hay entre los desplazamientos y las fuerzas que lo generan usando la resistencia de materiales. Para la construcción de la matriz de rigidez de un elemento barra es necesario conocer las diferentes tipos de rigideces y a continuación las obtendremos.

Rigidez axial

Se aplica un desplazamiento unitario en el extremo 1 dirección "x" como se indica en la figura

De la figura y aplicando los conceptos de la resistencia de materiales para encontrar la P_{1x} llegamos a que:

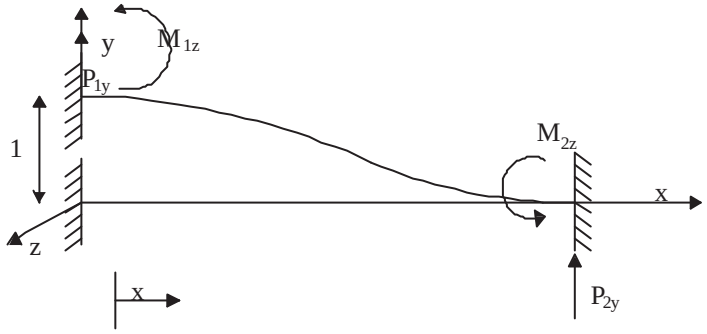
$$P_{1x} = \frac{EA}{L}$$

Por equilibrio

$$P_{2x} = \frac{EA}{L}$$

Rigidez al corte

Se aplica un desplazamiento unitario en el extremo 1 dirección "y" como se indica en la figura



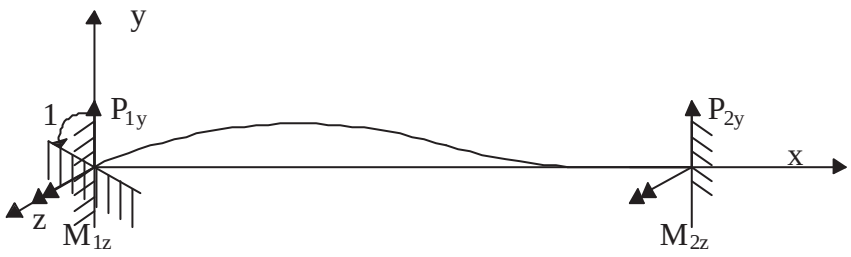
Al aplicar conceptos de la resistencia de materiales y tomando en cuenta la figura tenemos que:

$$P_{1y} = \frac{12EI_z}{L^3} \text{ y } M_{1z} = \frac{6EI_z}{L^2}$$

$$P_{2y} = \frac{12EI_z}{L^3} \text{ y } M_{2z} = \frac{6EI_z}{L^2}$$

Rigidez angular

Se aplica un giro unitario en el extremo 1 dirección "z" como se indica en la figura



Al aplicar conceptos de la resistencia de materiales y tomando en cuenta la figura tenemos que:

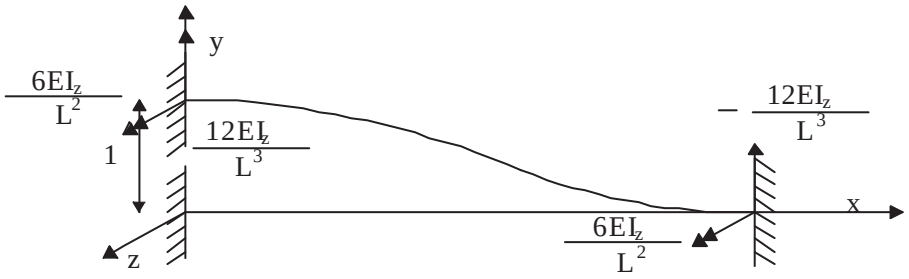
$$P_{1y} = \frac{6EI_z}{L^2} \text{ y } M_{1z} = \frac{4EI_z}{L}$$

$$P_{21y} = \frac{6EI_z}{L^2} \text{ y } M_{2z} = \frac{2EI_z}{L}$$

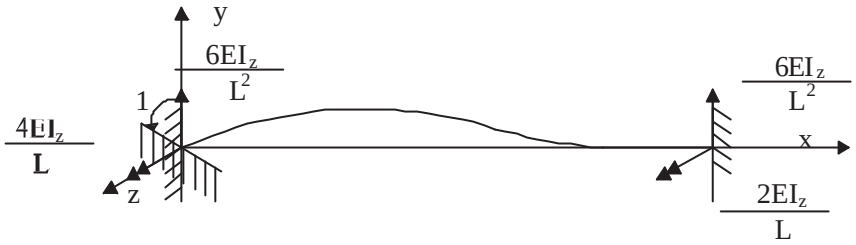
A continuación presentamos cuales son los valores de los diferentes tipos de rigidez que podemos encontrar.

1). Desplazamiento en dirección "x", extremo 1

2). Desplazamiento en dirección "y", extremo 1

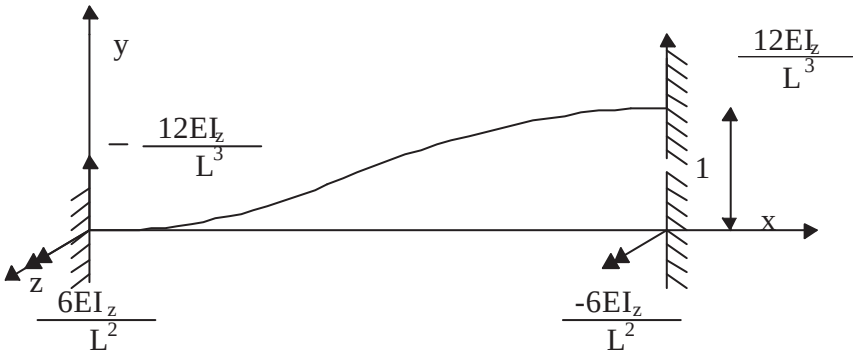


3). Giro alrededor del eje "z", extremo 1

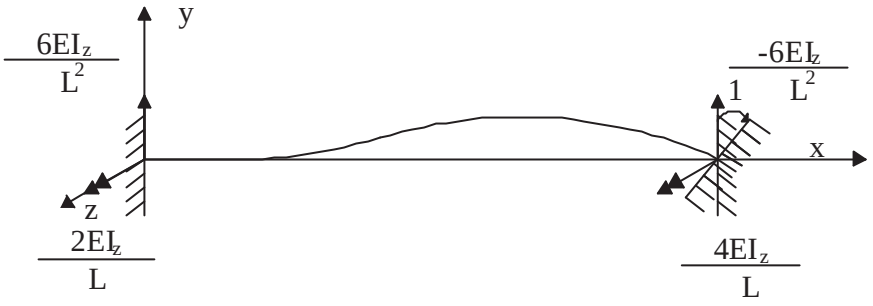


4). Desplazamiento en dirección "x", extremo 2

5). Desplazamiento en dirección "y", extremo 2



6). Giro alrededor del eje "z", extremo 2



La ecuación fuerza-desplazamiento de una barra esta dada por:

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

Sustituyendo el valor del vector {**P**}, los valores de las distintas rigideces se acoplan en la matriz de rigidez general de un elemento barra en el plano esto se ilustra en la figura siguiente.

$$\begin{Bmatrix} N_{x1} \\ V_{y1} \\ M_{z1} \\ N_{x2} \\ V_{y2} \\ M_{z2} \end{Bmatrix} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & \frac{-EA}{L} & 0 & 0 \\ 0 & \frac{12EI_Z}{L^3} & \frac{6EI}{L^2} & 0 & \frac{-12EI_Z}{L^3} & \frac{6EI_Z}{L^2} \\ 0 & \frac{6EI_Z}{L^2} & \frac{4EI_Z}{L} & 0 & \frac{-6EI_Z}{L^2} & \frac{2EI_Z}{L} \\ \frac{-EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & \frac{-12EI_Z}{L^3} & \frac{-6EI_Z}{L^2} & 0 & \frac{12EI_Z}{L^3} & \frac{-6EI_Z}{L^2} \\ 0 & \frac{12EI_Z}{L^2} & \frac{2EI_Z}{L} & 0 & \frac{-6EI_Z}{L^3} & \frac{4EI_Z}{L} \end{bmatrix} \begin{Bmatrix} d_{x1} \\ d_{y1} \\ \phi_{z1} \\ d_{x2} \\ d_{y2} \\ \phi_{z2} \end{Bmatrix}$$

Entonces, si al vector de cargas para el extremo uno le llamamos $\{P_1\}$ y el $\{P_2\}$. para el extremo dos y la matriz de rigideces de toda la barra la subdividimos en cuatro submatrices y además tomamos los desplazamientos en el extremo uno como $\{d_1\}$ y $\{d_2\}$. para el extremo dos tenemos.

$$\begin{Bmatrix} P_1 \\ P_2 \end{Bmatrix} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix}$$

Desarrollando el producto matricial resulta:

$$\{P_1\} = [K_{11}]\{d_1\} + [K_{12}]\{d_2\}$$
$$\{P_2\} = [K_{21}]\{d_1\} + [K_{22}]\{d_2\}$$

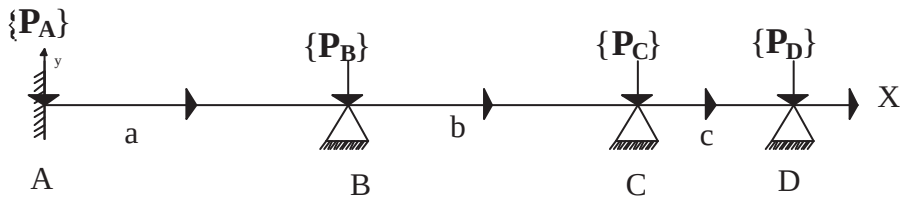
Donde: **K**i,j son submatrices de 3 x 3.

Recordando que $[K_{ij}'] = [K_{ij}]$ para el caso del tipo de estructura que vamos a analizar, ósea no hay rotación y el sistema local coincide con el global, por lo tanto es indistinto si ponemos las comillas para el sistema o no ya que coinciden.

Si ya tenemos la ecuación fuerza-desplazamiento para cada barra en el sistema global, entonces podemos acoplar el resto de las barras de la estructura para generar una ecuación fuerza-desplazamiento para toda la estructura.

ACOPLAMIENTO DE BARRAS

Para la comprensión del acoplamiento tomaremos como base la figura siguiente



$\{P_A\} = \{P_{1a}\}$

$\{P_B\} = \{P_{2a}\} + \{P_{1b}\}$

$\{P_C\} = \{P_{2b}\} + \{P_{1c}\}$

$\{P_D\} = \{P_{2c}\}$

Por compatibilidad

$\{d_{1a}\} = \{0\} \quad \{d_{1a}\} = \{0\}$

$\{d_{1b}\} = \{d_B{}'\} \quad \{d_{2b}\} = \{d_C{}'\}$

$\{d_{1c}\} = \{d_C{}'\} \quad \{d_{2c}\} = \{d_D{}'\}$

$\{P_1\} = [K_{11}]\{d_1\} + [K_{12}]\{d_2\}$ y $\{P_2\} = [K_{21}]\{d_1\} + [K_{22}]\{d_2\}$ son ecuaciones fuerza-desplazamiento.

Aplicando estas ecuaciones para cada barra y sustituyendo las ecuaciones que se obtuvieron por compatibilidad, se tiene;

$\{P_{1a}\} = [K_{21a}]\{d_B\}; \{P_{2a}\} = [K_{22a}]\{d_{2B}\}$

$\{P_{1b}\} = [K_{11b}]\{d_B\} + [K_{12b}]\{d_C\}$

$\{P_{2b}\} = [K_{21b}]\{d_B\} + [K_{22b}]\{d_C\}$

$\{P_{1c}\} = [K_{11c}]\{d_C\} + [K_{12c}]\{d_D\}$

$\{P_{2c}\} = [K_{21c}]\{d_C\} + [K_{22c}]\{d_D\}$

Sustituyendo las anteriores en $\{P_B\} = \{P_{2a}\} + \{P_{1b}\}$, $\{P_C\} = \{P_{2b}\} + \{P_{1c}\}$ y $\{P_D\} = \{P_{2c}\}$ tenemos:

$\{P_B\} = [K_{22a}]\{d_B\} + [K_{11b}]\{d_B\} + [K_{12b}]\{d_C\}$

$$\{P_C\} = [K_{21b}]\{d_B\} + [K_{22b}]\{d_C\} + [K_{11c}]\{d_C\} + [K_{12c}]\{d_D\}$$

$$\{P_D\} = [K_{21c}]\{d_C\} + [K_{22c}]\{d_D\}$$

Si esto lo agrupamos en forma matricial

$$\begin{Bmatrix} P_B \\ P_C \\ P_D \end{Bmatrix} = \begin{bmatrix} K_{22a} + K_{11b} & K_{12b} & 0 \\ K_{21b} & K_{22b} + K_{11c} & K_{12c} \\ 0 & K_{21c} & K_{22c} \end{bmatrix} \begin{Bmatrix} d_B \\ d_C \\ d_D \end{Bmatrix}$$

De una forma simplificada

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

Esta ecuación anterior es la ecuación fuerza-desplazamiento para la estructura mostrada en la figura anterior en el sistema global donde:

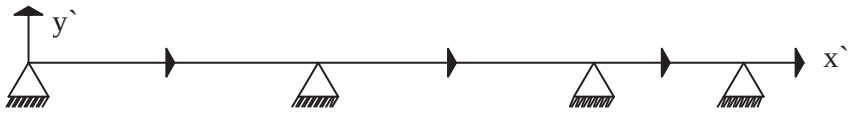
$\{P\}$ es el vector de cargas en los nodos de la estructura

$[K]$ es la matriz de rigideces del sistema estructural

$\{d\}$ es el vector de desplazamientos en los nodos de la estructura.

VIGAS

Para el caso de las vigas son estructuras que por lo general están sujetas a fuerza cortante y momento flexionante. Por lo cual estas no están sujetas a fuerza axial entonces no hay desplazamiento horizontal y de modo que solo puede tener desplazamientos angulares y desplazamientos lineales verticales y sus restricciones están dadas por el tipo de apoyo que tenga la viga.



Por lo anterior, la matriz de rigideces para una viga en el plano queda:

$$\begin{Bmatrix} V_{y1} \\ M_{z1} \\ V_{y2} \\ M_{z2} \end{Bmatrix} = \begin{bmatrix} \frac{12EI_z}{L^3} & \frac{6EI}{L^2} & \frac{-12EI_Z}{L^3} & \frac{6EI_z}{L^2} \\ \frac{6EI_z}{L^2} & \frac{4EI_z}{L} & \frac{-6EI_z}{L^2} & \frac{2EI_z}{L} \\ \frac{-12EI_z}{L^3} & \frac{-6EI_z}{L^2} & \frac{12EI_z}{L^3} & \frac{-6EI_z}{L^2} \\ \frac{12EI_z}{L^2} & \frac{2EI_z}{L} & \frac{-6EI_z}{L^3} & \frac{4EI_z}{L} \end{bmatrix} \begin{Bmatrix} d_{y1} \\ \phi_{z1} \\ d_{y2} \\ \phi_{z2} \end{Bmatrix}$$

Capítulo 2 Programación

INTRODUCCIÓN

En esta sección del trabajo hablaremos de una parte importante, la programación que es un concepto que aunque, se podría decir que es relativamente nuevo no lo es tanto. Se conoce como programación de computadoras a la implementación de un algoritmo en un determinado lenguaje de programación, conformando un programa. El lenguaje de programación puede ser de alto nivel o bajo nivel, en función del grado de abstracción.

Un algoritmo es una secuencia no ambigua, finita y ordenada de instrucciones que han de seguirse para resolver un problema. Un programa normalmente implementa (traduce a un lenguaje de programación concreto) un algoritmo. Nótese que es la secuencia de instrucciones es en sí la que debe ser finita, no el número de pasos realizados como la ejecución de ellas.

Los programas suelen subdividirse en partes menores (módulos), de modo que la complejidad algorítmica de cada una de las partes sea menor que la del programa completo, lo cual ayuda al desarrollo del programa.

Según Niklaus Wirth un programa está formado por algoritmos y estructura de datos.

TECNICAS DE PROGRAMACION

Se han propuesto diversas técnicas de programación, cuyo objetivo es mejorar tanto el proceso de creación de software como su mantenimiento. Entre ellas se pueden mencionar las programaciones lineal, estructurada, modular y orientada a objetos.

La programación lineal es un procedimiento o algoritmo matemático mediante el cual se resuelve un problema indeterminado, formulado a través de ecuaciones lineales, optimizando la función objetivo, también lineal.

Las aplicaciones de este tipo de programación son: Optimización de la combinación de diámetros comerciales en una red ramificada de distribución de agua, aprovechamiento óptimo de los recursos de una cuenca hidrográfica, solución de problemas de transporte, investigación de operaciones entre otros.

La programación estructurada es una forma de escribir programación de computadora de forma clara, para ello utiliza únicamente tres estructuras: secuencial, selectiva e iterativa; siendo innecesario y no permitiéndose el uso de la instrucción o instrucciones de transferencia incondicional.

Hoy en día las aplicaciones informáticas son mucho más ambiciosas, por lo que las técnicas de programación estructurada no son suficientes lo que ha llevado al desarrollo de nuevas técnicas tales como la programación orientada o objetos y el desarrollo de entornos de programación que facilitan la programación de grandes aplicaciones.

Con la programación estructurada, elaborar programas de computador sigue siendo una labor que demanda esfuerzo, creatividad, habilidad y cuidado. Sin embargo, con este estilo podemos obtener las siguientes ventajas:

- Los programas son más fáciles de entender, ya que pueden ser leído de forma secuencial, sin necesidad de hacer seguimiento a saltos de línea (GOTO) dentro de los bloques de código para entender la lógica.
- La estructura del programa es clara puesto que las instrucciones están más ligadas o relacionadas entre sí.
- Reducción del esfuerzo en las pruebas. El seguimiento de los fallos o errores del programa ("debugging") se facilita debido a la estructura más visible, por lo que los errores se pueden detectar y corregir más fácilmente.
- Reducción de los costos de mantenimiento de los programas.
- Programas más sencillos y más rápidos (ya que es más fácil su optimización).
- Los bloques de código son auto explicativos, lo que facilita a la documentación.
- Los GOTO se reservan para construir las instrucciones básicas (selección e iteración)

El principal inconveniente de este método de programación, es que se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar problemático su manejo. Esto se resuelve empleando **la programación modular**. Definiendo módulos interdependientes programados y compilados por separado. Un método un poco más sofisticado es la programación por capas, en la que los módulos tienen una estructura jerárquica muy definida y se denominan capas.

Valiéndose de la técnica del diseño estructurado para el diseño de algoritmos consigue desarrollar programas a partir de un conjunto de módulos, cada uno de los cuales desempeña una tarea necesaria para el correcto funcionamiento del programa global. Los módulos son interdependedientes, y son codificados y compilados por separado.

En el caso de que el conjunto de módulos implicado esté sometido a una jerarquía, estaríamos hablando de un sistema de programación por capas, en la que cada "capa" representaría un nivel en la jerarquía de los módulos.

La Programación Orientada a Objetos (POO u OOP según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo, y encapsulamiento. Su uso se popularizó a principios de la década de 1990. Actualmente son muchos los lenguajes de programación que soportan la orientación a objetos.

Los objetos son entidades que combinan estado, comportamiento e identidad: el estado está compuesto de datos, serán uno o vários atributos a los que se habrán asignado unos valores concretos (datos), el comportamiento está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él, la identidad es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador.

La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

LENGUAJES DE PROGRAMACION

Para realizar un programa es necesario que la maquina entienda la serie de instrucciones que se le dan para ejecutar a esto se le llama lenguaje de programación.

Con la llegada de las computadoras aparecen las secuencias de posiciones de llaves eléctricas que debían conectarse para obtener una acción determinada, una llave conectada era un 1 y una llave desconectada era un 0. Una sucesión de llaves en cualquiera de sus dos posiciones definía una secuencia de ceros y unos (por ejemplo: 0100011010011101...) que venía a representar una instrucción o un conjunto de instrucciones (programa) para el ordenador (o computador) en el que se estaba trabajando. A esta primera forma de especificar programas para una computadora se la denomina lenguaje máquina o código máquina.

La necesidad de recordar secuencias de programación para las acciones usuales llevó a denominarlas con nombres fáciles de memorizar y asociar: ADD (sumar), SUB (restar), MUL (multiplicar), CALL (ejectuar subrutina), etc. A esta secuencia de posiciones se le denominó "instrucciones", y a este conjunto de instrucciones se le llamó lenguaje ensamblador.

Posteriormente aparecieron diferentes lenguajes de programación, los cuales reciben su denominación porque tienen una estructura sintáctica similar a los lenguajes escritos por los humanos.

Un lenguaje de programación consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Aunque muchas veces se usa lenguaje de programación y lenguaje informático como si fuesen sinónimos, no tiene por qué ser así, ya que los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como, por ejemplo, el HTML (lenguaje para el marcado de páginas web).

Los procesadores usados en las computadoras son capaces de entender y actuar según lo indican programas escritos en un lenguaje fijo llamado lenguaje de máquina. Todo programa escrito en otro lenguaje puede ser ejecutado de dos maneras:

- Mediante un programa que va adaptando las instrucciones conforme son encontradas. A este proceso se lo llama interpretar y a los programas que lo hacen se los conoce como intérpretes.
- Traduciendo este programa al programa equivalente escrito en lenguaje de máquina. A ese proceso se lo llama compilar y al traductor se lo conoce como compilador.

Hay distintas clasificaciones para los lenguajes de programación entre los que tenemos

Según su nivel de abstracción:

Lenguajes de bajo nivel

Los lenguajes de bajo nivel son lenguajes de programación que se acercan al funcionamiento de una computadora. El lenguaje de más bajo nivel es, por excelencia, el código máquina. A éste le sigue el lenguaje ensamblador, ya que al programar en ensamblador se trabajan con los registros de memoria de la computadora de forma directa.

Lenguajes de medio nivel

Hay lenguajes de programación que son considerados por algunos expertos como lenguajes de medio nivel (como es el caso del lenguaje C) al tener ciertas características que los acercan a los lenguajes de bajo nivel pero teniendo, al mismo tiempo, ciertas cualidades que lo hacen un lenguaje más cercano al humano y, por tanto, de alto nivel.

Lenguajes de alto nivel

Los lenguajes de alto nivel son normalmente fáciles de aprender porque están formados por elementos de lenguajes naturales, como el inglés. En BASIC, el lenguaje de alto nivel más conocido, los comandos como "IF CONTADOR = 10 THEN STOP" pueden utilizarse para pedir a la computadora que pare si CONTADOR es igual a 10. Por desgracia para muchas personas esta forma de trabajar es

un poco frustrante, dado que a pesar de que las computadoras parecen comprender un lenguaje natural, lo hacen en realidad de una forma rígida y sistemática.

Según la forma de ejecución:

Lenguajes compilados

Naturalmente, un programa que se escribe en un lenguaje de alto nivel también tiene que traducirse a un código que pueda utilizar la máquina. Los programas traductores que pueden realizar esta operación se llaman compiladores. Éstos, como los programas ensambladores avanzados, pueden generar muchas líneas de código de máquina por cada proposición del programa fuente. Se requiere una corrida de compilación antes de procesar los datos de un problema.

Al usar un lenguaje compilado (como lo son los lenguajes del popular Visual Studio de Microsoft), el programa desarrollado nunca se ejecuta mientras haya errores, sino hasta que luego de haber compilado el programa, ya no aparecen errores en el código, en este podríamos poner a **FORTAN 90** debido a que usa un compilador.

Lenguajes interpretados

Se puede también utilizar una alternativa diferente de los compiladores para traducir lenguajes de alto nivel. En vez de traducir el programa fuente y grabar en forma permanente el código objeto que se produce durante la corrida de compilación para utilizarlo en una corrida de producción futura, el programador sólo carga el programa fuente en la computadora junto con los datos que se van a procesar. A continuación, un programa intérprete, almacenado en el sistema operativo del disco, o incluido de manera permanente dentro de la máquina, convierte cada proposición del programa fuente en lenguaje de máquina conforme vaya siendo necesario durante el proceso de los datos. No se graba el código objeto para utilizarlo posteriormente.

Según el paradigma de programación

Un paradigma de programación representa un enfoque particular o filosofía para la construcción del software. No es mejor uno que otro sino que cada uno tiene ventajas y desventajas. También hay situaciones donde un paradigma resulta más apropiado que otro.

Atendiendo al paradigma de programación, se pueden clasificar los lenguajes en:

Lenguajes imperativos

- BASIC

- C
- C++
- Java
- C#
- Perl

Lenguajes Funcionales

Puros:

- Haskell
- Miranda

Híbridos:

- Lisp
- Scheme
- Ocaml
- Standard ML
- ML
- Scala

Lenguajes Logicos

- Prolog

Entre los lenguajes orientados a objetos destacan los siguientes:

- ActionScript
- ActionScript 2
- ActionScrip t 3
- Ada
- C++
- C#
- Clarion
- Lenguaje de programación D
- Delphi
- Harbour
- Eiffel
- Java

- Lexico (en castellano)
- Objective-C
- Ocaml
- Oz
- Lenguaje de programación R
- Perl
- PHP (en su versión 5)
- PowerBuilder
- Python
- Ruby
- Smalltalk
- Turbo Pascal 7
- Magik (SmallWorld)
- VB.NET
- Visual FoxPro
- XBase++
- Gambas

Algunos ejemplos típicos de lenguajes compilados:

- **Fortran**
- La familia de lenguajes de C, incluyendo C++ y Objective C
- Ada, Pascal (incluyendo su dialecto Delphi)
- Algol

El leguaje que se uso para la realización del programa de análisis de vigas fue el **FORTRAN.**

Fortran (Acrónimo de "Formula Translator".) es un lenguaje de programación desarrollado en los años 50 y activamente utilizado desde entonces.

Fortran se utiliza principalmente en aplicaciones científicas y análisis numérico. Desde 1958 ha pasado por varias versiones, entre las que destacan FORTRAN II, FORTRAN IV, FORTRAN 77, Fortran 90, Fortran 95 y Fortran 2003. Si bien el lenguaje era inicialmente un lenguaje imperativo, las últimas versiones incluyen elementos de la programación orientada a objetos y también tiene parte de programación modular.

El primer compilador de FORTRAN se desarrolló para una IBM 704 entre 1954 y 1957 , por un grupo liderado por John W. Backus. En la época se consideró imprescindible que los programas escritos en FORTRAN corrieran a velocidad comparable a la del lenguaje ensamblado; de otra forma, nadie lo tomaría en cuenta.

El lenguaje ha sido ampliamente adoptado por la comunidad científica para escribir aplicaciones con cómputos intensivos. La inclusión en el lenguaje de la aritmética de números complejos amplió la gama de aplicaciones para las cuales el lenguaje se adapta especialmente y muchas técnicas de compilación de lenguajes han sido creadas para mejorar la calidad del código generado por los compiladores de Fortran.

PRINCIPALES CARACTERÍSTICAS

El lenguaje fue diseñado tomando en cuenta que los programas serían escritos en tarjetas perforadas de 80 columnas. Así por ejemplo, las líneas debían ser numeradas y la única alteración posible en el orden de ejecución era producida con la instrucción goto. Estas características han evolucionado de versión en versión. Las versiones actuales contienen subprogramas, recursión y una variada gama de estructuras de control.

Dado que fue la primera tentativa de proyección de un lenguaje de programación de alto nivel, tiene una sintaxis considerada arcaica por muchos programadores que aprenden lenguajes más modernos. Algunas de las versiones anteriores no poseían facilidades que son consideradas como útiles en las máquinas modernas, como la colocación dinámica de memoria. Se debe tener en cuenta que la sintaxis de Fortran fue afinada para el uso en trabajos numéricos y científicos y que muchas de sus deficiencias han sido abordadas en revisiones más recientes del lenguaje. Por ejemplo, Fortran 95 posee comandos mucho más breves para efectuar operaciones matemáticas con matrices y dispone de tipos. Esto no sólo mejora mucho la lectura del programa sino que además aporta información útil al compilador. Por estas razones Fortran no es muy usado fuera de los campos de la informática y el análisis numérico, pero permanece como el lenguaje a escoger para desempeñar tareas de computación numérica de alto rendimiento.

Haciendo una comparación entre un software realizado en visual Basic 6 vs fortran, se ha podido llegar a ciertas conclusiones que se podrían considerar: Visual Basic 6 Muy rápida implementación Pocas líneas para generar una idea Pero desgraciadamente muy lento para hacer cálculos a gran escala Fortran 95 Muy larga su implementación Pero afortunadamente muy rápido para hacer calculos. El resultado en

tiempo fue el siguiente Visual Basic 80 Horas Fortran 10 segundos. El fortran para calcular es excelente para implementar es muy complejo comparado con los lenguajes de punta.

La versión de fortran que se utilizo para hacer el programa de las vigas fue el fortran 90, el cual presenta las siguientes características

Formato libre en el código fuente.

En Fortran 90 se puede usar el formato de entrada de Fortran 77 o el formato libre. Si se usa el formato libre, la extensión .90 deberá ser usada en el nombre del archivo.

Apuntadores y asignación dinámica.

Es posible usar almacenamiento dinámico, con lo que se puede hacer que todos los arreglos "trabajen" no importando su tamaño.

Tipos de datos definidos por el usuario.

Se pueden definir sus propios tipos compuestos de datos, de forma parecida a como se hace en C con struct o en Pascal con record.

Módulos.

Los módulos permiten hacer una programación en un estilo orientado a objetos parecido a como se hace en C++. Los módulos pueden también ser usados para ocultar variables globales, por lo que hace a la construcción common caiga en desuso.

Funciones recursivas.

Un procedimiento recursivo es un procedimiento que puede llamarse a si mismo. Suficiente de declararlo con la palabra clave recursive delante de la identificación del procedimiento.

Por ejemplo:

```
recursive function factorial(n)
integer::factorial
integer,intent(in)
if(n==0)then
factorial=1
else
factorial=n*factorial(n-1)
end if
end function factorial
```

La utilización de procedimientos (funciones y subrutinas) recursivas es poco eficiente. Es más conveniente utilizar ciclos.

Operaciones con arreglos construidas internamente

Las sentencias como `A=0` y `C=A+B` son ahora válida cuando `A, B y C` son arreglos. También hay una función para la multiplicación de matrices (`matmul`).

Hay otras muchas características, bastante numerosas para ser mencionadas. Fortran 90 es muy diferente a las primeras versiones de Fortran. Pero guarda compatibilidad con las anteriores, Fortran 77 ha sido incluido como un subconjunto de Fortran 90.

Ahora mencionaremos a grandes rasgos la sintaxis y algunas de las palabras reservadas que utiliza fortran 90.

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
real       lista de variables
double precision  lista de variables
complex    lista de variables
logical    lista de variables
character lista de variables
```

La lista de variables consiste de nombres de variables separadas por comas. Cada variable deberá ser declarada exactamente una vez. Si una variable no esta declarada, Fortran 90 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. Varios programas viejos de Fortran usan estas reglas implícitas, pero no se recomienda su uso. La probabilidad de errores en el programa crece exponencialmente si no se declaran las variables explícitamente.

Variables enteras y de punto flotante

Fortran sólo tiene un tipo para variables enteras. Los enteros son usualmente guardados en 32 bits (4 bytes). Por lo que el rango de valores que pueden tomar los enteros es de (-231,231-1).

Fortran 90 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 programa. Esto se puede hacer con la sentencia `parameter`, a la vez que hace los programas más legibles. Por ejemplo el programa visto anteriormente podría haberse escrito de la siguiente forma:

```
program circulo
      real r, area, pi
      parameter (pi=3.14159)
```

Este programa lee un número real r y muestra el área del círculo con radio r.

```
      write (*,*) 'Escribe el radio r:'
      read  (*,*) r
      area = pi*r*r
      write (*,*) 'Area = ', area
      stop
      end
```

La sintaxis de la sentencia parameter es

```
parameter (nombre = constante, ... , nombre = constante)
```

Las reglas para la sentencia parameter son:

La "variable" definida en la sentencia parameter no es una variable, es una constante por lo que su valor nunca cambia.

Una "variable" puede aparecer a lo más una vez en la sentencia parameter.

Las sentencias parameter deberán estar antes que cualquier sentencia de ejecución.

Algunas de las razones para usar parameter son:

Ayuda a recordar más fácilmente el uso de constantes. Es fácil cambiar una constante si aparece muchas veces en el programa.

También hay la posibilidad de tener sentencias 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:

SIMBOLO		SIGNIFICADO
<	.LT.	Menor que
>	.GT.	Mayor que
==	.EQ.	Igual que
<=	.LE.	Menor o igual que
>=	.GE.	Mayor o igual que
/=	.NE.	Diferente a

Por lo que no se pueden usar símbolos como < or = para comparación en Fortran 77, por lo que se tienen que usar abreviaturas de dos letras encerradas con puntos. Sin embargo en Fortran 90 ya pueden ser usados.

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

Asignación de Variables Lógicas

Los valores booleanos pueden ser guardados en variables lógicas. La asignación es de forma análoga a la asignación aritmética. Ejemplo:

```
logical a, b
a = .TRUE.
b = a .AND. 3 .LT. 5/2
```

El orden de precedencia es importante, como se muestra en el último ejemplo. La regla es que las expresiones aritméticas son evaluadas primero, después las que contienen operadores relacionales, y finalmente las de operadores lógicos. Por lo que a b se le asigna .FALSE. en el ejemplo anterior.

Las expresiones lógicas son usadas frecuentemente en sentencias condicionales como 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
```

Lo anterior tiene que ser escrito en una sola línea. El siguiente ejemplo obtiene el valor absoluto de x:

```
if (x .LT. 0) x = -x
```

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
elseif (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 sentencia 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. Se muestra un ejemplo:

```
if (x .GT. 0) then
        if (x .GE. y) then
            write(*,*) 'x es positivo y x >= y'
        else
            write(*,*) 'x es positivo pero, x < y'
        endif
    elseif (x .LT. 0) then
        write(*,*) 'x es negativo'
    else
        write(*,*) 'x es cero'
    endif
```

Se debe evitar anidar muchos niveles de sentencias if ya que es difícil de seguir.

Ciclos en Fortran 90

Fortran 90 ha adoptado la construcción do-enddo como su ciclo.

Por lo que el ejemplo de decrementar de dos en dos queda como:

```
do i = 10, 1, -2
    write(*,*) 'i =', i
end do
```

Para simular los ciclos while y until se puede usar la construcción do-enddo, pero se tiene que agregar una sentencia condicional de salida exit (salida). El caso general es:

```
do
    sentencias
    if (expr lógica) exit
    sentencias
end do
```

Si se tienen la condición de salida al principio es un ciclo while, y si esta al final se tiene un ciclo until.

Arreglos

Muchos cálculos científicos usan vectores y matrices. El tipo de dato usado en Fortran para representar tales objetos es el array. Un arreglo unidimensional corresponde a un vector, mientras que un arreglo bidimensional corresponde a una matriz. Para entender como son usados en Fortran 77, no solamente se requiere conocer la sintaxis para su uso, sino también como son guardados estos objetos en la memoria.

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 d(20)
```

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. Por convención, los arreglos en Fortran están indexados a partir del valor 1. Por lo tanto el primer elemento en el arreglo es `d(1)` y el último es `d(20)`. Sin embargo, se puede definir un rango de índice arbitrario para los arreglos como se observa en los siguientes ejemplos:

```
real b(0:19), c(-162:237)
```

En el caso de `b` es similar con el arreglo d del ejemplo previo, excepto que el índice corre desde el 0 hasta el 19. El arreglo `c` es un arreglo de longitud `237-(-162)+1 = 400`.

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

Ejemplos:

```
integer i(10)
logical aa(0:1)
double precision x(100)
```

Cada elemento de un arreglo puede ser visto como una variable separada. Se referencia al i-ésimo elemento de un arreglo `a` por `a(i)`. A continuación se muestra un segmento de código que guarda los primeros 10 cuadrados en un arreglo `cuad`

```
integer i, cuad(10)
do i=1, 10, 1
    cuad(i) = i**2;
    write(*,*) cuad(i)
end do
```

Un error común en Fortran es hacer que el programa intente acceder elementos del arreglo que están fuera de los límites. Lo anterior es responsabilidad del programador, ya que tales errores no son detectados por el compilador.

Arreglos Bidimensionales

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

```
real A(3,5)
```

Define un arreglo bidimensional de 3 x 5 = 15 números del tipo real. Es útil pensar que el primer índice es el índice del renglón, y el segundo índice corresponde a la columna. Por lo tanto se vería como:

	1	2	3	4	5
1					
2					
3					

Un arreglo bidimensional podría también tener índices de rango arbitrario. La sintaxis general para declarar el arreglo es:

```
nombre (índice1_inf : indice1_sup, índice2_inf : indice2_sup)
```

El tamaño total del arreglo es de

```
tamaño = (indice1_sup − indice1_inf + 1) x (indice2_sup − indice2_inf + 1)
```

Es muy común en Fortran declarar arreglos que son más grandes que la matriz que se va a guardar. ejemplo:

```
real A(3,5)
    integer i,j
```

Solamente se usará una submatriz de 3 x 3 del arreglo

```
    do i=1, 3
        do j=1, 3
            a(i,j) = real(i)/real(j)
        end do
    end do
```

Los elementos en la submatriz A(1:3,4:5) no estan definidas. No se debe considerar que estos elementos están inicializados a cero por el compilador (algunos compiladores lo hacen, pero otros no).

Subprogramas

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. Al inicio de la sección se

comento de los subprogramas definidas por el usuario, pero Fortran 90 tiene también funciones incorporadas.

Un ejemplo simple muestra como usar una función:

```
x = cos(pi/3.0)
```

En este caso la función coseno cos de 60º, asignará a la variable `x` el valor de 0.5 (si `pi` ha sido definido correctamente; Fortran 90 no tiene constantes incorporadas). Hay varias funciones incorporadas en Fortran 90. 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)
```

En general, una función siempre tiene un tipo. Varias de las funciones incorporadas mencionadas anteriormente son sin embargo genéricas. Por lo tanto en el ejemplo anterior `pi` y `x` podrían ser del tipo real o del tipo doble precisión. El compilador revisará los tipos y usará la versión correcta de la función cos (real o doble precisión). Desafortunadamente, Fortran no es un lenguaje polimórfico, por lo que en general, el programador debe hacer coincidir los tipos de las variables y las funciones.

Se revisa a continuación como implementar las funciones escritas por el usuario. Supongamos el siguiente problema: un meteorólogo ha estudiado los niveles de precipitación en el área de una bahía y ha obtenido un modelo (función) `ll(m,t)` donde `ll` es la cantidad de lluvia, m es el mes, y t es un parámetro escalar que depende de la localidad. Dada la fórmula para `ll` y el valor de `t`, calcular la precipitación anual

La forma obvia de resolver el problema es escribir un ciclo que corra sobre todos los meses y sume los valores de `ll`. Como el cálculo del valor de `ll` es un subproblema independiente, es conveniente implementarlo como una función. El siguiente programa principal puede ser usado:

```
program lluvia
      real r, t, suma
      integer m
      read (*,*) t
      suma = 0.0
      do m = 1, 12
          suma = suma + ll(m, t)
      end do
      write (*,*) 'La precipitación Anual es ', suma, 'pulgadas'
      stop
      end
```

Además, la función ll tiene que ser definida como una función de Fortran. La fórmula del meteorólogo es:

ll(m,t) = t/10 * (m**2 + 14*m + 46) si la expresión es positiva

ll(m,t) = 0 otro caso

La correspondiente función en Fortran es

```
real function ll(m,t)
        integer m
        real t
        ll = 0.1*t * (m**2 + 14*m + 46)
        if (ll .LT. 0) ll = 0.0
        return
      end
```

Se puede observar que la estructura de una función es parecida a la del programa principal. Las diferencias son:

Las funciones tienen un tipo. El tipo debe coincidir con el tipo de la variable que recibirá el valor.

El valor que devolverá la función, deberá ser asignado en una variable que tenga el mismo nombre que la función.

Las funciones son terminadas con la sentencia return en vez de la sentencia stop.

Para resumir, la sintaxis general de una función en Fortran es:

```
tipo function nombre (lista_de parámetros)
    declaraciones
    sentencias
    return
end
```

La función es llamada simplemente usando el nombre de la función y haciendo una lista de argumentos entre paréntesis.

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
    return
end
```

Observar que las subrutinas no tienen tipo y por consecuencia no pueden hacerse asignación al momento de llamar al procedimiento.

Se da un ejemplo de una subrutina muy sencilla. El propósito de la subrutina es intercambiar dos valores enteros.

```
subroutine iswap (a, b)
    integer a, b
 Variables Locales
    integer tmp
    tmp = a
    a = b
    b = tmp
    return
end
```

Se debe observar que hay 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.

Capítulo 3 Presentación Del Programa

DESCRIPCIÓN DEL PROGRAMA

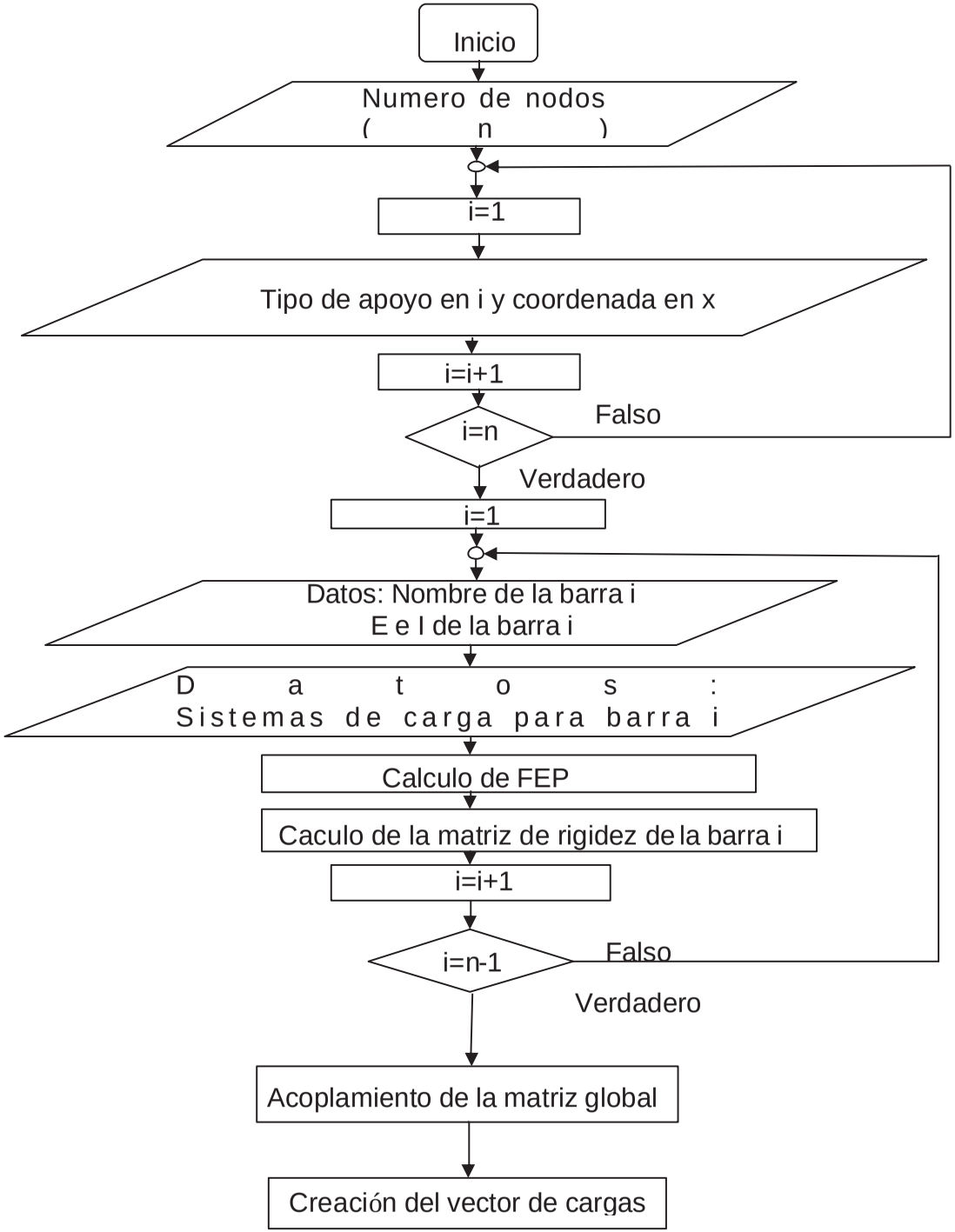
Este es un programa hecho en FORTRAN 90 por lo tanto no cuenta con una interfase grafica y tal vez no sea del agrado de muchos usuarios. En esencia, el programa calcula los elementos mecánicos para vigas en el plano, bajo ciertos sistemas de cargas y apoyos, usando el método de las rigideces. Los apoyos que permite el programa son los más comunes que se suponen para el análisis como son empotrado, articulado fijo , articulado móvil y también permite que el nodo de la estructura este libre. Es necesario mencionar que los sistemas de carga permitidos en el programa no producen fuerza normal por lo tanto no se produce desplazamiento vertical.

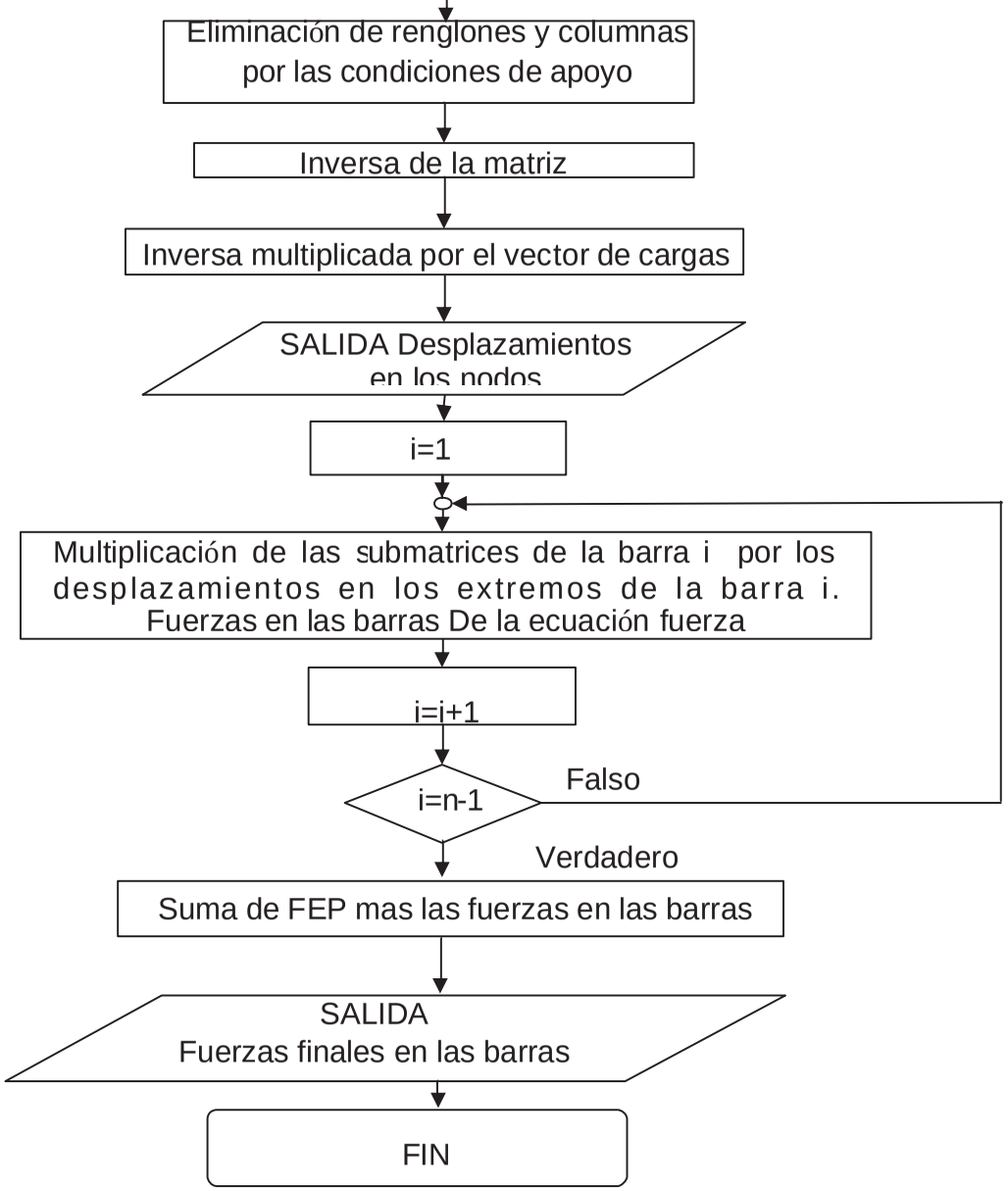
La forma en que trabaja el programa es pidiendo en pantalla, al usuario la información, necesaria para realizar el análisis. Después de procesar los datos, los resultados del análisis se muestran en un archivo de texto llamado final.txt, donde se imprimen los datos de las barras y los elementos mecánicos de cada una de las barras de la viga. La manera de hacer la introducción de datos para el programa se mencionara en el apartado de “manual del usuario”.

DIAGRAMA DE FLUJO

Una definición más formal de diagrama de flujo es un esquema para representar gráficamente un algoritmo. Se basan en la utilización de diversos símbolos para representar operaciones específicas. Se les llama diagramas de flujo porque los símbolos utilizados se conectan por medio de flechas para indicar la secuencia de operación. Para hacer comprensibles los diagramas a todas las personas, los símbolos se someten a una normalización; es decir, se hicieron símbolos casi universales, ya que, en un principio cada usuario podría tener sus propios símbolos para representar sus procesos en forma de Diagrama de flujo. Esto trajo como consecuencia que sólo aquel que conocía sus símbolos, los podía interpretar. La simbología utilizada para la elaboración de diagramas de flujo es variable y debe ajustarse a un patrón definido previamente."

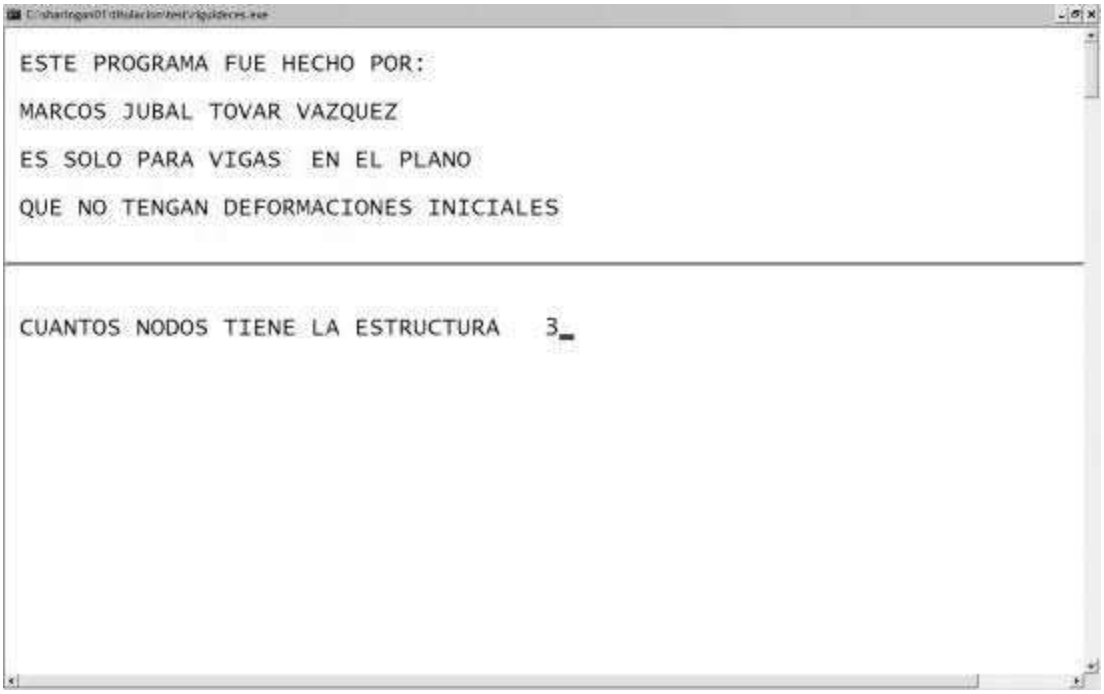
A continuación presentamos el diagrama de flujo del programa


Inicio
Numero de nodos
(n)
i=1
Tipo de apoyo en i y coordenada en x
i=i+1
i=n
Falso
Verdadero
i=1
Datos: Nombre de la barra i
E e I de la barra i
D a t o s :
Sistemas de carga para barra i
Calculo de FEP
Caculo de la matriz de rigidez de la barra i
i=i+1
i=n-1
Falso
Verdadero
Acoplamiento de la matriz global
Creación del vector de cargas

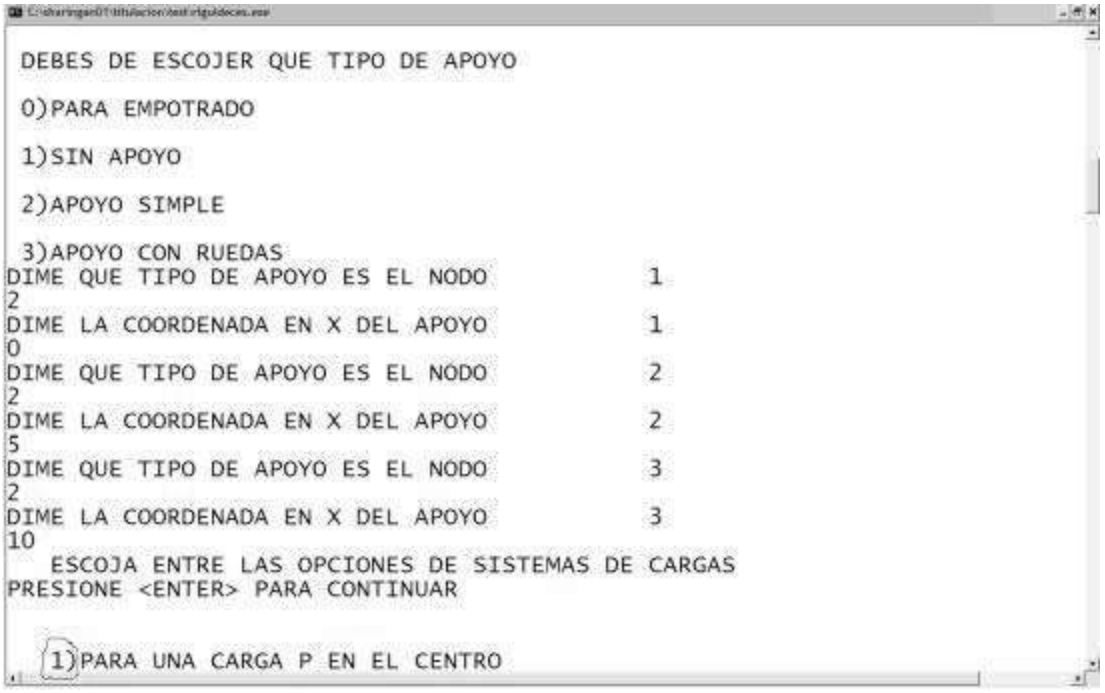

Eliminación de renglones y columnas por las condiciones de apoyo
Inversa de la matriz
Inversa multiplicada por el vector de cargas
SALIDA Desplazamientos en los nodos
i=1
Multiplicación de las submatrices de la barra i por los desplazamientos en los extremos de la barra i. Fuerzas en las barras De la ecuación fuerza
i=i+1
i=n-1
Falso
Verdadero
Suma de FEP mas las fuerzas en las barras
SALIDA
Fuerzas finales en las barras
FIN

MANUAL DEL USUARIO

En primer lugar, cuando usas el programa aparece una pantalla como la de MS-DOS, y entonces el programa te pregunta primero cuantos nodos tiene la estructura, este valor debe de ser entero ya que ninguna estructura puede tener nodos fraccionarios, considerando un nodo al punto donde llegan dos barras, como se muestra en la figura.

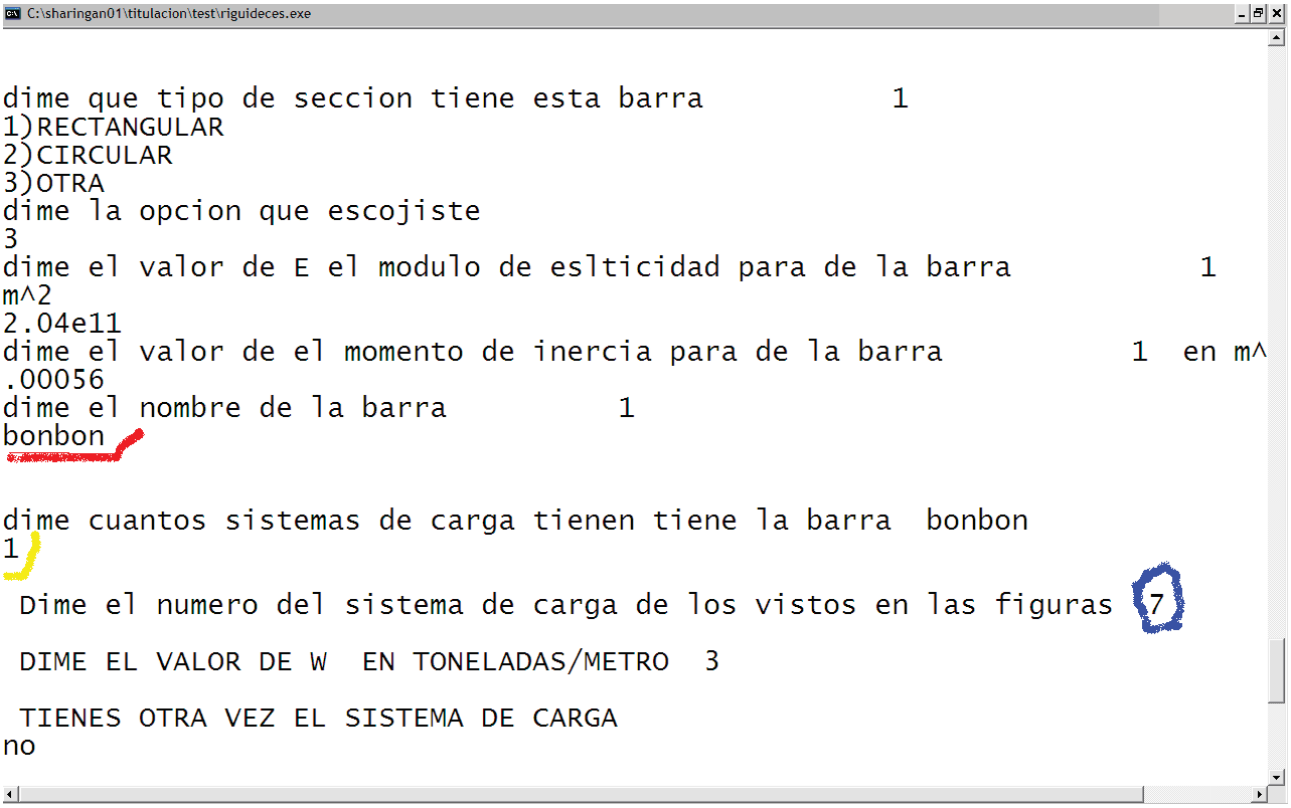


Después de que tecleamos el número de nodos ahora se despliega en pantalla un menú que nos indica los tipos de apoyos que se permiten en el programa, después nos pide que introduzcamos que tipo de apoyo es el primer nodo y su coordenada en “x”. Para los apoyos el usuario tiene que teclear un numero entero correspondiente a tipo de apoyo, como por ejemplo para el apoyo empotrado el usuario deberá de teclear el numero cero, para las coordenadas se acostumbra tomar el primer nodo como la cero. Este paso se ve el la imagen que sigue.



El siguiente paso es que imprime en pantalla los sistemas de carga admisibles para el programa y a cada uno le asigna un número clave es el numero que en la imagen esta encerado en azul.

Entonces el programa te pide datos para la primer barra y puede escoger la sección transversal circular, rectangular u otra, cabe mencionar las secciones circular y rectangular son de tipo macizo, ya que si se escoge una de esta opciones el programa pedirá datos para la circular el radio y para la rectangular la base y altura adicionalmente se pedirá el modulo de elasticidad del material. Por otro lado si se escoge el caso de otra seccion, entonces tendremos que teclear el valor de momento de inercia de forma directa, se debe de tener especial cuidado en las unidades ya que el programa te dice en que unidades debes de teclear los valores.



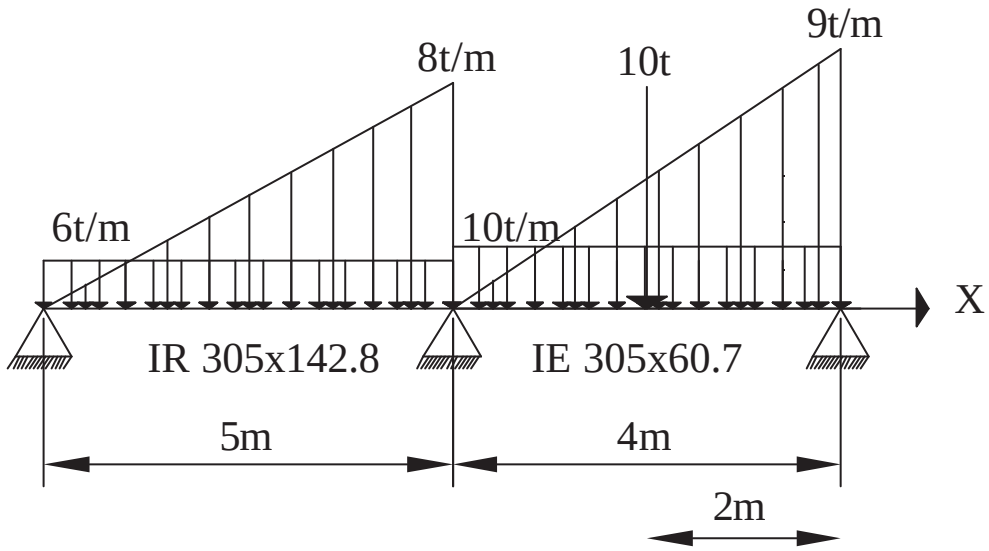
Después de haber hecho todo lo anterior, nos pide asignarle un nombre a la primer barra, el cual no debe ser mayor de quince caracteres. En este caso esta subrayado de rojo el nombre en la figura. A continuación nos pregunta cuantos sistemas de carga tenemos, sin contar los repetidos, en esta barra, entonces tecleamos un número entero que en la figura esta subrayado de amarillo. Después nos despliega otra pregunta acerca del número clave de esos sistemas de carga, ósea el número clave que corresponde al sistema que tenemos en la viga. Este numero en la figura esta encerrado en azul. Para cada sistema de carga nos pregunta algunos datos extra, una vez tecleados los datos del sistema de cargas, nos dice si tenemos otra vez ese sistema y debemos responder con “si” o “no” con letras minúsculas y así sigue el proceso para cada sistema de cargas que tengamos y después para cada barra.

Al realizar todo lo ya mencionado el programa genera un archivo de texto llamado final.txt el cual contiene los datos de las barras, desplazamientos en los nodos y los elementos mecánicos finales de cada barra.

Ejemplos de aplicación

Ejemplo 1

Obtener las reacciones y los diagramas de la viga que se muestra en la siguiente figura recordando que es una viga en el plano y sin fuerza normal.



El ejemplo de forma manual, para este ejemplo le llamaremos a la primer barra hidan.txt y a la segunda barra le llamaremos kakasu.txt.

Tenemos para la barra hidan.txt

E(t/m^2)	2.04E+11
I(m^4)	0.00034672
L(m)	5

$$[k_{11}]=\begin{bmatrix} 6{,}790{,}164.\ 48 & 16{,}975{,}411\ .20 \\ 16{,}975{,}411\ .20 & 56{,}584{,}704\ .00 \end{bmatrix} \quad [k_{12}]=\begin{bmatrix} -6{,}790{,}164.\ 48 & 16{,}975{,}411\ .20 \\ -16{,}975{,}411\ .20 & 28{,}292{,}352\ .00 \end{bmatrix}$$

$$[k_{21}]=\begin{bmatrix} -6{,}790{,}164.\ 48 & -16{,}975{,}411\ .20 \\ 16{,}975{,}411\ .20 & 28{,}292{,}352\ .00 \end{bmatrix} \quad [k_{22}]=\begin{bmatrix} 6{,}790{,}164.\ 48 & -16{,}975{,}411\ .20 \\ -16{,}975{,}411\ .20 & 56{,}584{,}704\ .00 \end{bmatrix}$$

FEP

	Ext 1	Ext 2
F(t)=	-21.357	-29.357
M(t-m)=	-19.46416667	22.7975

Para la barra kakasu.txt

E(t/m^2)	2.04E+11
I(m^4)	0.00011321
L(m)	4

$$[k_{11}] = \begin{bmatrix} 4{,}330{,}282.\ 50 & 8{,}660{,}565.\ 00 \\ 8{,}660{,}565.\ 00 & 23{,}094{,}840\ .00 \end{bmatrix} \quad [k_{12}] = \begin{bmatrix} -4{,}330{,}282.\ 50 & 8{,}660{,}565.\ 00 \\ -8{,}660{,}565.\ 00 & 11{,}547{,}420\ .00 \end{bmatrix}$$

$$[k_{21}] = \begin{bmatrix} -4{,}330{,}282.\ 50 & -8{,}660{,}565.\ 00 \\ 8{,}660{,}565.\ 00 & 11{,}547{,}420\ .00 \end{bmatrix} \quad [k_{22}] = \begin{bmatrix} 4{,}330{,}282.\ 50 & -8{,}660{,}565.\ 00 \\ -8{,}660{,}565.\ 00 & 23{,}094{,}840\ .00 \end{bmatrix}$$

FEP

	Ext 1	Ext 2
F(t)=	-30.5214	-37.7214
M(t-m)=	-23.21426667	25.61426667

La ecuación fuerza desplazamiento es

$$\begin{Bmatrix} P_A \\ P_B \\ P_C \end{Bmatrix} = \begin{bmatrix} \mathrm{k}_{11\mathrm{hidan.txt}} & \mathrm{k}_{12\mathrm{hidan.txt}} & 0 \\ \mathrm{k}_{21\mathrm{hidan.txt}} & \mathrm{k}_{22\mathrm{hidan.txt}} + \mathrm{k}_{11\mathrm{kakasu.txt}} & \mathrm{k}_{12\mathrm{kakasu.txt}} \\ 0 & \mathrm{k}_{21\mathrm{kakasu.txt}} & \mathrm{k}_{22\mathrm{kakasu.txt}} \end{bmatrix} \begin{Bmatrix} d_A \\ d_B \\ d_C \end{Bmatrix}$$

Y sabemos que solo se permiten los desplazamientos angulares en los nodos.

Haciendo la inversa y multiplicando por el vector de cargas tenemos los desplazamientos en radianes

$$\begin{Bmatrix} \phi_{\mathrm{zA}} \\ \phi_{\mathrm{zB}} \\ \phi_{\mathrm{zC}} \end{Bmatrix} = \begin{Bmatrix} -3.14767\mathrm{E}-07 \\ -5.8431\mathrm{E}-08 \\ 1.13831\mathrm{E}-06 \end{Bmatrix}$$

Aplicando las ecuaciones $\{P_1\} = [K_{11}]\{d_1\} + [K_{12}]\{d_2\}$ y $\{P_2\} = [K_{21}]\{d_1\} + [K_{22}]\{d_2\}$

Las primeras fuerzas son las de las ecuaciones de fuerza-desplazamiento y después para a obtener los elementos mecánicos le sumamos las FEP multiplicadas por menos uno.

Para la barra hidan.txt

	EXTREMO 1	EXTREMO 2
V(t)=	15.0218	35.6922
M(t-m)=	0.0000	-35.0093

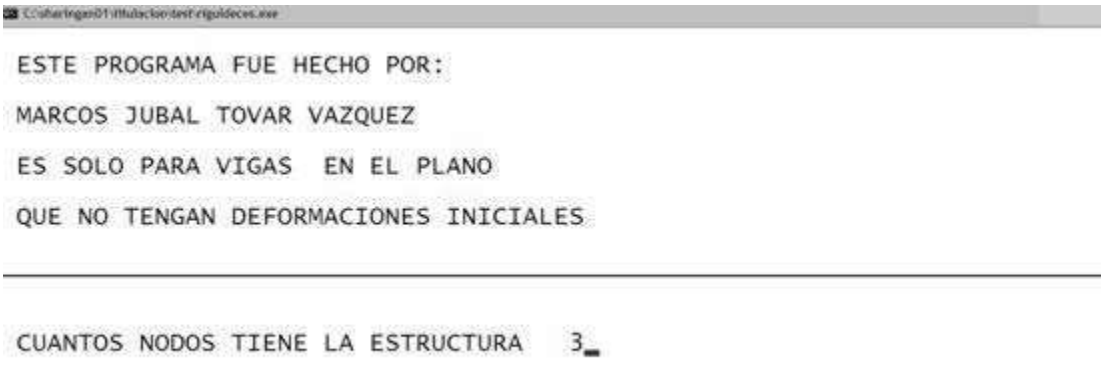
Para la barra kakasu.txt

	EXTREMO 1	EXTREMO 2
V(t)=	39.8737	28.3691
M(t-m)=	35.0093	0.0000

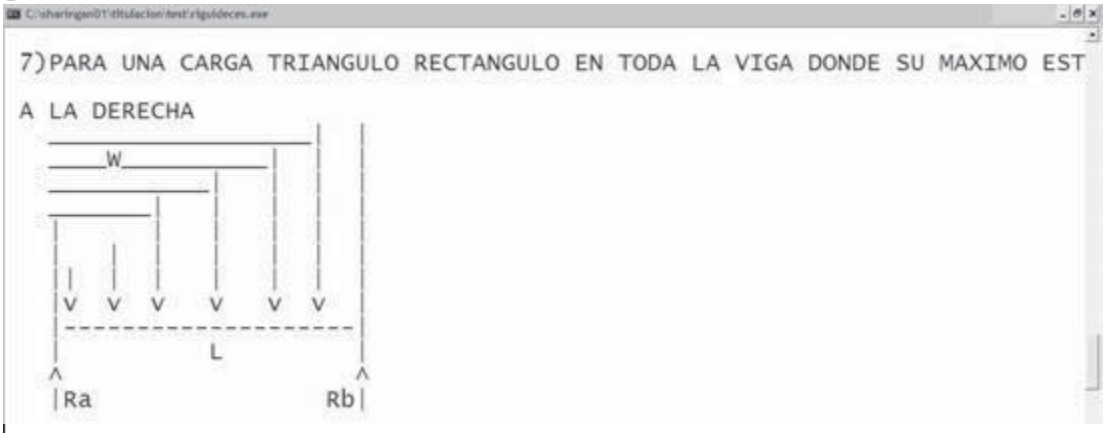
Se observa que las ultimas columnas representan el cortante y el momento en el extremo uno y dos respectivamente de cada barra.

Ahora resolvemos el mismo ejemplo anterior con el programa.

A continuación se muestra como se introdujeron los datos del problema en el programa.



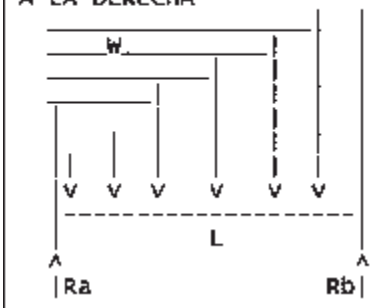
```
DEBES DE ESCOJER QUE TIPO DE APOYO

0)PARA EMPOTRADO

1)SIN APOYO

2)APOYO SIMPLE

3)APOYO CON RUEDAS
DIME QUE TIPO DE APOYO ES EL NODO           1
2
DIME LA COORDENADA EN X DEL APOYO           1
0
DIME QUE TIPO DE APOYO ES EL NODO           2
2
DIME LA COORDENADA EN X DEL APOYO           2
5
DIME QUE TIPO DE APOYO ES EL NODO           3
2
DIME LA COORDENADA EN X DEL APOYO           3
9
```



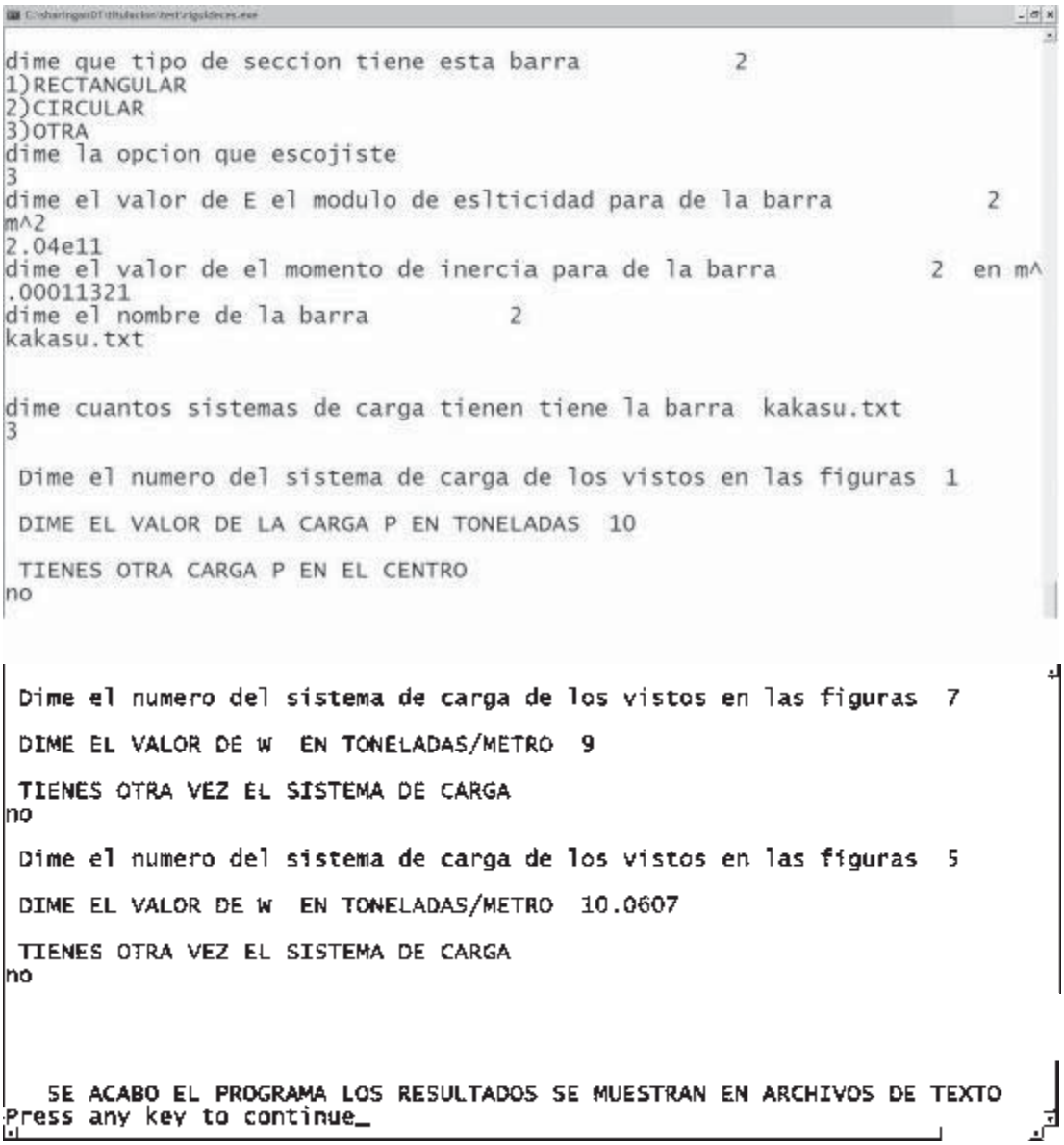

```
5)PARA UNA CARGA W REPARTIDA EN TODA LA VIGA
          W
          L
   Ra                  Rb
```


```
5)PARA UNA CARGA W REPARTIDA EN TODA LA VIGA
          W
          L
   Ra                  Rb
```

```
7)PARA UNA CARGA TRIANGULO RECTANGULO EN TODA LA VIGA DONDE SU MAXIMO EST
A LA DERECHA
```




```
dime que tipo de seccion tiene esta barra          1
1)RECTANGULAR
2)CIRCULAR
3)OTRA
dime la opcion que escojiste
3
dime el valor de E el modulo de eslticidad para de la barra          1
m^2
2.04e11
dime el valor de el momento de inercia para de la barra          1  en m^
.00034672
dime el nombre de la barra          1
hidan.txt


dime cuantos sistemas de carga tienen tiene la barra  hidan.txt
2

 Dime el numero del sistema de carga de los vistos en las figuras  7

 DIME EL VALOR DE W  EN TONELADAS/METRO  8

 TIENES OTRA VEZ EL SISTEMA DE CARGA
no

 Dime el numero del sistema de carga de los vistos en las figuras  5

 DIME EL VALOR DE W  EN TONELADAS/METRO  6.1428

 TIENES OTRA VEZ EL SISTEMA DE CARGA
no
```



A continuación se muestra el archivo de resultados que arroja el programa:

BARRA hidan.txt

MOMENTO DE INERCIA EN m4= 3.4672 E-004

MODULO DE ELASTICIDAD EN t/m2 = 2.040 E+011

LONGUITUD EN m = 5.000

RESULTADOS DE LA BARRA hidan.txt

EXTREMO UNO:

El desplazamiento vertical es (en m)= 0.000 E+000

El giro es (en radianes)= -3.147E-007

V(cortante en t)= 15.0218

M(momento en t-m)= .0000

EXTREMO DOS:

El desplazamiento vertical es (en m)= 0.000 E+000

El giro es (en radianes)= -5.843E-008

V(cortante en t)= 35.6922

M(momento en t-m)= -35.0093

BARRA kakasu.txt

MOMENTO DE INERCIA EN m4= 1.1321 E-004

MODULO DE ELASTICIDAD EN t/m2 = 2.04E+011

LONGUITUD EN m = 4.000

RESULTADOS DE LA BARRA kakasu.txt

EXTREMO UNO:

El desplazamiento vertical es (en m)= 0.000 E+000

El giro es (en radianes)= -5.843E-008

V(cortante en t)= 39.8737

M(momento en t-m)= 35.0093

EXTREMO DOS:

El desplazamiento vertical es (en m)= 0.000 E+000

El giro es (en radianes)= 1.138E-006

V(cortante en t)= 28.3691

M(momento en t-m)= .0000

ESTE ES UN PROGRAMA FUE HECHO POR:

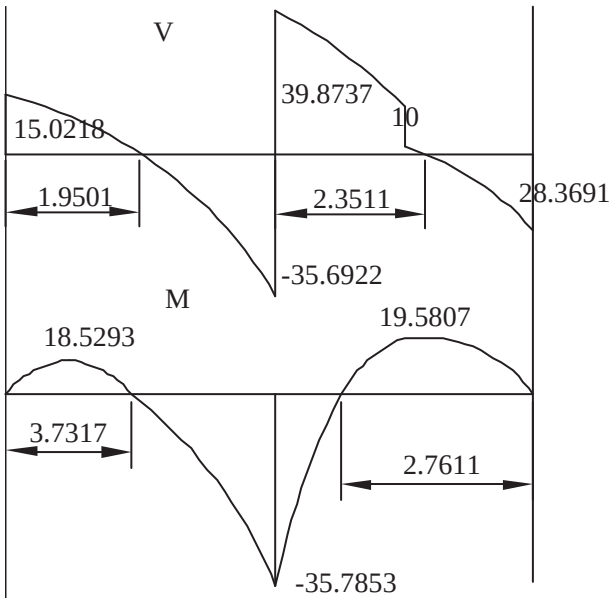
MARCOS JUBAL TOVAR VAZQUEZ

Además de generar este archivo de resultados, el programa genera un archivo con la matriz de rigideces acoplada de la estructura y otro archivo con los desplazamientos en los nodos.

Estos son los resultados que arrojo el programa hecho en FORTRAN 90 las unidades de cortante son toneladas y de momento fexionante son t-m. Para el cálculo también se tomo en cuenta el peso propio de la viga aunque no aparece en la figura.

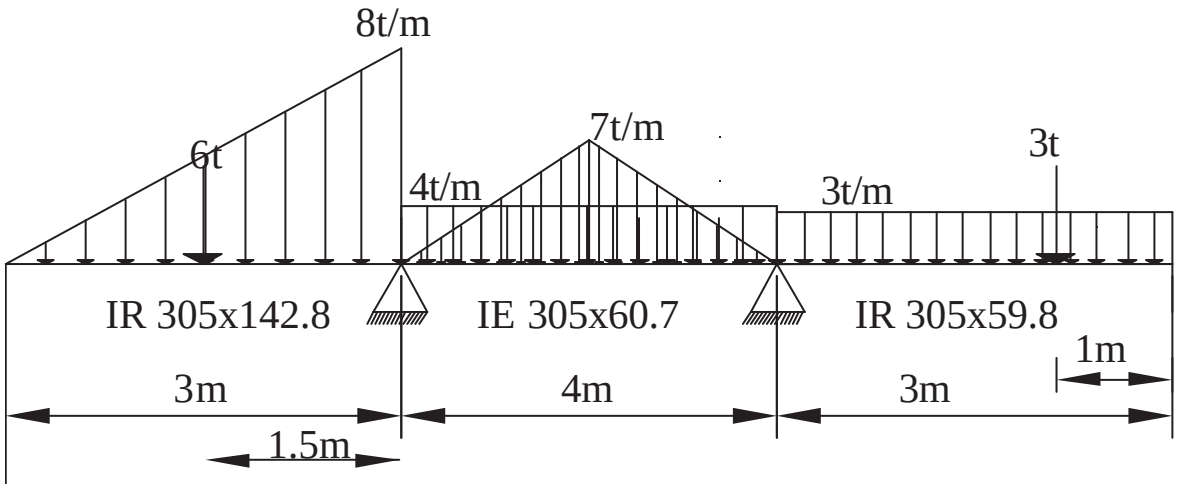
Podemos observar que los resultados emitidos por el programa realizado en FORTRAN 90 son idénticos a los resultados calculados de forma manual, por lo que decimos que este programa es confiable y por lo tanto ya podemos proceder a dibujar los diagramas de fuerza cortante y momento flexionante para la viga del ejemplo.

Ahora presentamos los diagramas.



Ejemplo 2

Obtener las reacciones y los diagramas de la viga que se muestra en la siguiente figura recordando que es una viga en el plano y sin fuerza normal.



Nuevamente no se pone el peso propio de la viga pero si se toma en cuenta.

BARRA a

MOMENTO DE INERCIA EN m4= 3.4672 E-004

MODULO DE ELASTICIDAD EN t/m2 = 2.040 E+011

LONGUITUD EN m = 3.000

RESULTADOS DE LA BARRA a

EXTREMO UNO:

El desplazamiento vertical es (en m)= -1.826 E-006

El giro es (en radianes)= 6.523E-007

V(cortante en t)= .0000

M(momento en t-m)= .0000

EXTREMO DOS:

El desplazamiento vertical es (en m) = 0.000 E+000

El giro es (en radianes)= 4.206E-007

V(cortante en t)= 18.4284

M(momento en t-m)= -21.6426

BARRA b

MOMENTO DE INERCIA EN m4= 1.2903E-004

MODULO DE ELASTICIDAD EN t/m2 = 2.040 E+011

LONGUITUD EN m = 3.000

RESULTADOS DE LA BARRA b

EXTREMO UNO:

El desplazamiento vertical es (en m)= 0.000 E+000

El giro es (en radianes)= 4.206E-007

V(cortante en t)= 19.2747

M(momento en t-m)= 21.6426

EXTREMO DOS:

El desplazamiento vertical es (en m)= 0.000 E+000

El giro es (en radianes)= 5.892E-008

V(cortante en t)= 10.9681

M(momento en t-m)= -5.0295

BARRA c

MOMENTO DE INERCIA EN m4= 5.9800E-02

MODULO DE ELASTICIDAD EN t/m2 = 2.040000000000000E+011

LONGUITUD EN m = 3.000

RESULTADOS DE LA BARRA c

EXTREMO UNO:

El desplazamiento vertical es (en m)= 0.000 E+000

El giro es (en radianes)= 5.892E-008

V(cortante en t)= 6.4015

M(momento en t-m)= 5.0295

EXTREMO DOS:

El desplazamiento vertical es (en m)= 0.000 E+000

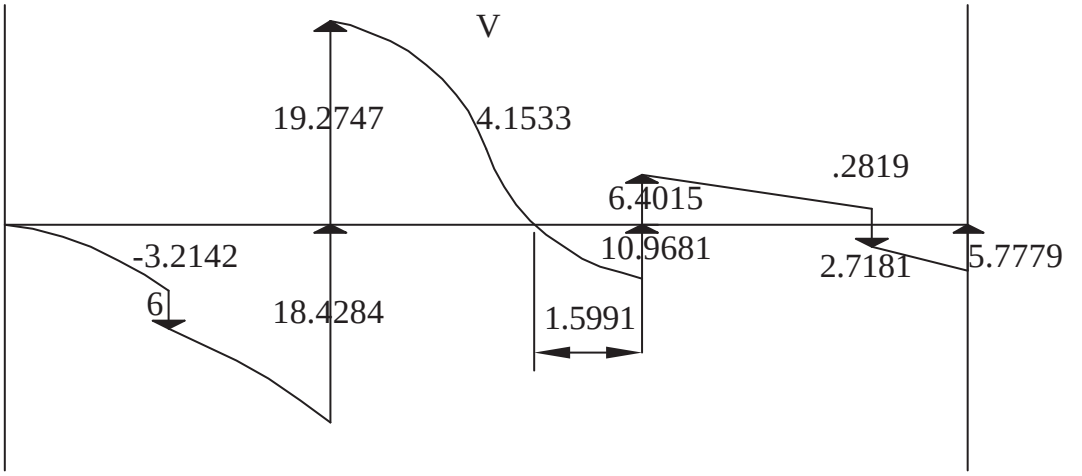
El giro es (en radianes)= 0.000E+000

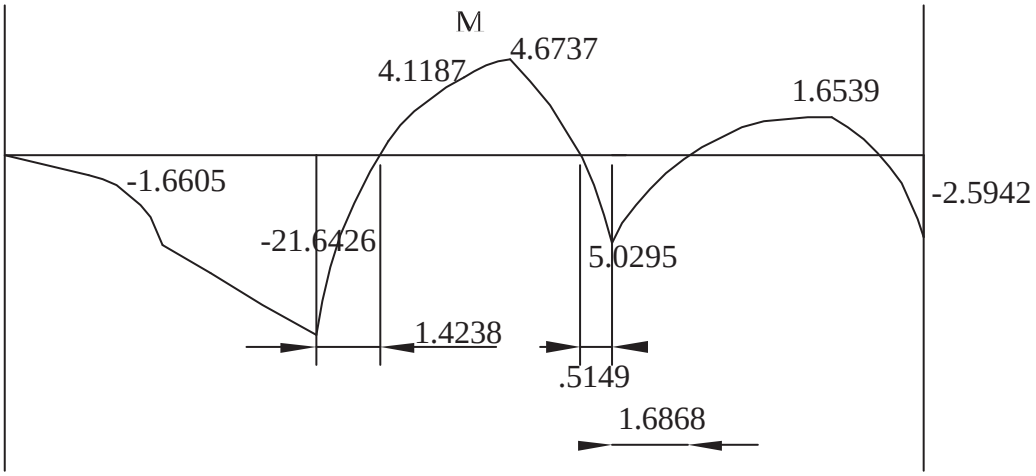
V(cortante en t)= 5.7779

M(momento en t-m)= -2.5942

ESTE ES UN PROGRAMA FUE HECHO POR:

MARCOS JUBAL TOVAR VAZQUEZ

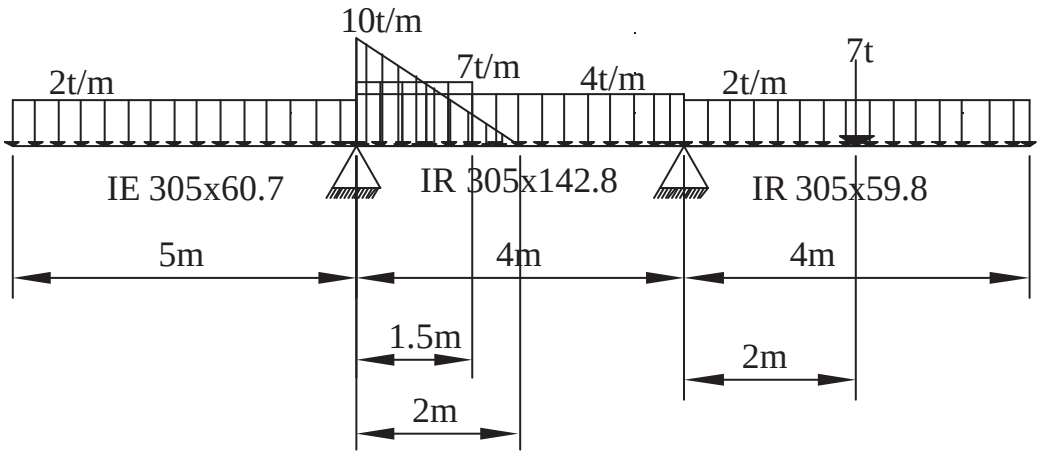




Los diagramas de cortante están en toneladas y las distancias en metros, el de momentos esta en toneladas-metro y distancias en metros.

Ejemplo 3

Obtener las reacciones y los diagramas de la viga que se muestra en la siguiente figura recordando que es una viga en el plano y sin fuerza normal.



DATOS DE LA BARRA a

MOMENTO DE INERCIA EN m4= 1.1321E-004

MODULO DE ELASTICIDAD EN t/m2 = 2.040 E+011

LONGUITUD EN m = 5.000

RESULTADOS DE LA BARRA a

EXTREMO UNO:

El desplazamiento vertical es (en m)= -9.14E-006

El giro es (en radianes)= 2.294E-006

V(cortante en t)= .0000

M(momento en t-m)= .0000

EXTREMO DOS:

El desplazamiento vertical es (en m)= 0.000 E+000

El giro es (en radianes)= 4.353E-007

V(cortante en t)= 10.3035

M(momento en t-m)= -25.7587

DATOS DE LA BARRA b

MOMENTO DE INERCIA EN m4= 3.4672E-004

MODULO DE ELASTICIDAD EN t/m2 = 2.040 E+011

LONGUITUD EN m = 4.000

RESULTADOS DE LA BARRA b

EXTREMO UNO:

El desplazamiento vertical es (en m)= 0.000 E+000

El giro es (en radianes)= 4.353E-007

V(cortante en t)= 23.9703

M(momento en t-m)= 25.7587

EXTREMO DOS:

El desplazamiento vertical es (en m)= 0.000E+000

El giro es (en radianes)= -5.340 E-007

V(cortante en t)= 13.1009

M(momento en t-m)= -30.4784

DATOS DE LA BARRA c

MOMENTO DE INERCIA EN m4= 1.2903 E-004

MODULO DE ELASTICIDAD EN t/m2 = 2.04E+011

LONGUITUD EN m = 4.0000

RESULTADOS DE LA BARRA c

EXTREMO UNO:

El desplazamiento vertical es (en m)= 0.000

El giro es (en radianes)= -5.340 E-007

V(cortante en t)= 15.2392

M(momento en t-m)= 30.4784

EXTREMO DOS:

El desplazamiento vertical es (en m)= -6.413E-006

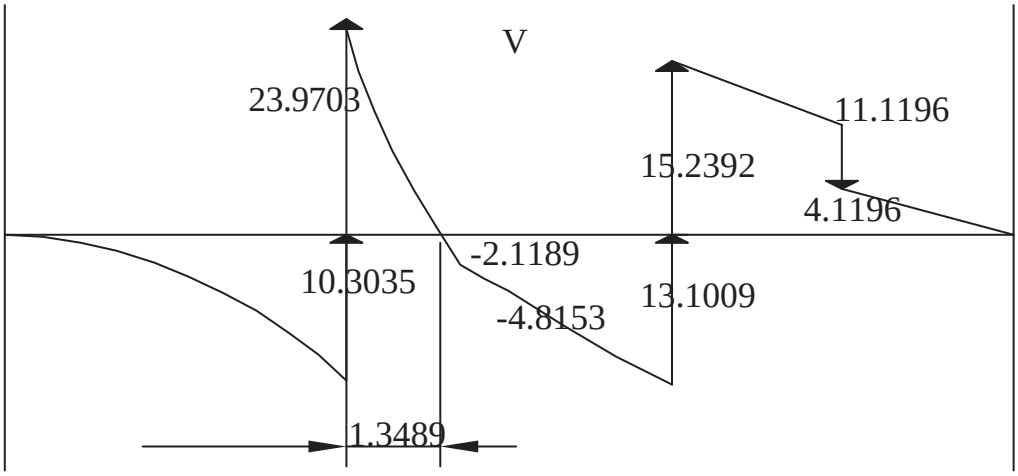
El giro es (en radianes)= -1.900E-006

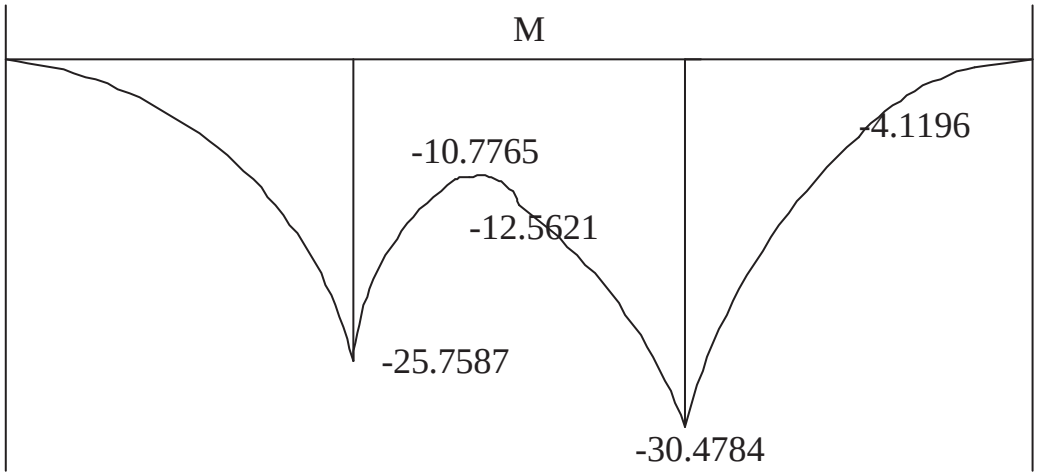
V(cortante en t)= .0000

M(momento en t-m)= .0000

ESTE ES UN PROGRAMA FUE HECHO POR:

MARCOS JUBAL TOVAR VAZQUEZ





Los diagramas de cortante están en toneladas y las distancias en metros, el de momentos esta en toneladas-metro y distancias en metros.

Conclusiones y recomendaciones

Como se observó en los ejemplos de aplicación, el programa realizado en FORTRAN 90 es un programa que nos permite conocer las reacciones de los apoyos en vigas de una manera muy rápida, también podemos ver que los resultados del análisis con el programa son iguales a los calculados de manera manual.

Concluimos también que este programa puede ser usado de manera didáctica para la comprobación paso a paso del método de los desplazamientos, debido a que este programa muestra en archivos de texto los diferentes pasos como los desplazamientos en un archivo llamado desplazamientos y en otro muestra la matriz de rigideces acoplada de toda la viga, por lo cual estos archivos pueden tomarse como una referencia para cuando se este trabajando de una manera didáctica.

También podemos concluir que el lenguaje FORTAN aunque no tenga una interfase grafica, es un gran lenguaje de programación, debido a que la mayoría de las operaciones son programadas por el usuario y este llega a conocer a fondo el lenguaje. Además el usuario que llega tener una noción integral de la programación con este lenguaje.

.

Bibliografía

Nelson y MacCormac**, Análisis de estructuras métodos clásico y matricial,** Editorial Alfaomega, México, 2006

Instituto Mexicano de la Construcción en Acero, A. C.**, Manual de construcción en acero diseño por esfuerzos permisibles,** Editorial Limusa, México, 2005

Rojas, **Apuntes de análisis estructural de Rafael Rojas R.**

Sánchez I., **Apuntes del curso**