UNIVERSIDAD
DE LOS ANDES
VENEZUELA

Facultad de Ingeniería

Postgrado en Modelado y Simulación de Sistemas

Tesis de Maestría

Desarrollo de un portal WEB para el análisis tridimensional de estructuras aporticadas en concreto armado

Por

Ing. Jormany Quintero R.

Tutor: Prof. Jacinto Dávila

Cotutor: Prof. Maylett Uzcátegui

Presentado ante la ilustre Universidad de Los Andes como requisito parcial para obtener el Título de Magister Scientiae en Modelado y Simulación de Sistemas

Marzo 2015

Desarrollo de un portal WEB para el análisis tridimensional de estructuras aporticadas en concreto armado

Ing. Jormany Quintero R.

Tesis de Maestría — Magister Scientiarum, 238 páginas
Postgrado en Modelado y Simulación de Sistemas, Universidad de Los Andes, 2015

Resumen: Para facilitar el análisis y el cálculo en estructuras y construcciones civiles, se han desarrollado diversos programas y aplicaciones que permiten evaluar y diseñar estructuras mediante la teoría de los elementos finitos. En la actualidad existen tanto en el ámbito académico como en el mercado una gama de herramientas computacionales para el análisis de estructuras aporticadas, como lo son Sap2000, Etabs, Portal de Pórticos 2D entre otros. Basada en la tendencia de la "Ciencia Ciudadana" y en el aumento de construcciones en sectores más vulnerables a eventos sísmicos severos, surge la idea desarrollar una herramienta computacional especializada para el alcance de todos que permita la digitalización de sistemas aporticados de concreto armado tridimensionales, con el fin de hacer el análisis y evaluación del daño que puedan presentar las estructuras ante un evento sísmico. Esta herramienta es la primera aplicación de uso específico empleando el programa de elementos finitos PEEF, basado en la Teoría del Daño Concentrado. Con la finalidad de captar un número mayor de usuarios y portabilidad del servicio WEB, se propuso que este portal pueda ser accedido a través de computadores personales (PC). Así mismo como aporte adicional en la evaluación de las estructuras tridimensionales de concreto armado se ha planteado el desarrollo de un elemento finito, para incluir en el análisis elementos tipo placa y así tomar en cuenta el efecto de las losas de entrepiso.

Palabras clave: Servicio WEB, Internet, Computadores Personales, Dispositivos Móviles, Estructuras aporticadas tridimensionales, concreto armado, pórticos

Este trabajo fue procesado en LaTeX.

Índice

Índice de Figuras

Capítulo 1

Introducción

1.1 Antecedentes

Para facilitar el análisis y el cálculo en estructuras y construcciones civiles, se han desarrollado diversos programas y aplicaciones que permiten evaluar y diseñar estructuras mediante la teoría de los elementos finitos. En tal sentido, en la Universidad de Los Andes, Venezuela, se desarrolló un programa de propósito general para el análisis de sólidos en dos y tres dimensiones denominado Programa Endógeno de Elementos Finitos (PEEF),Uzcátegui Flores (2012).

1.1.1 Herramientas computacionales para el análisis de pórticos

En la actualidad existen en el mundo del análisis y simulación de estructuras, una gama de herramientas computacionales para el análisis de estructuras aporticadas. A continuación se explican brevemente dos de ellas, las cuales constituyen los programas de mayor uso por parte de los ingenieros civiles en Venezuela.

1.1.1.1 SAP2000

Es un *software* privado para el análisis y diseño estructural basado en el método de los elementos finitos. Desarrollado por Computers and Structures, INC. (2012b), Berkeley, Estados Unidos. Específicamente su aplicación está enmarcada en el modelado, análisis, diseño y optimización de estructuras como: puentes, estadios, torres, plantas industriales, muros, presas, sólidos, piezas mecánicas, así como edificios aporticados en dos y tres dimensiones.

Para el diseño y análisis estructural el programa cuenta con diversos tipos de elementos finitos: barras, placas, membranas, cáscaras y sólidos. Especificamente para la evaluación de pórticos se usa el modelo de plasticidad concentrada, basado en el modelo de rótula plástica. Así mismo los análisis pueden ser estáticos o dinámicos.

El análisis y diseño se realiza generalmente mediante un proceso iterativo siguiendo los siguientes pasos:

- Creación o modificación de la geometría, propiedades de los materiales, cargas y parámetros de análisis.
- Análisis del modelo.
- Revisión de los resultados.
- Chequeo y mejoramiento de la estructura diseñada.

SAP2000, es un *software* comercial, con licencias privadas que se renuevan anualmente y que funciona solo bajo sistema operativo Windows. Esta herramienta de análisis funciona localmente en la maquina donde se realiza la instalación, la cual debe cumplir con unos requisitos especiales en cuanto al *Hardware* para el buen funcionamiento y desempeño del *software*.

1.1.1.2 ETABS

Es un programa diseñado por los mismos creadores de *SAP2000*, Computers and Structures, INC. (2012a), para el análisis y diseño de edificios así como de naves industriales. Al igual que *SAP2000* es un programa de *software* privativo. Su uso permite crear

modelos 2D y 3D, modificarlos, optimizarlos y visualizar resultados desde una simple interfaz de usuario como se aprecia en la Figura 1.1.

Al igual que el *SAP2000*, puede realizar análisis de estructuras complejas, pero cuenta con opciones extras que simplifican el diseño de edificaciones, como por ejemplo:

- División automática de elementos estructurales (columnas, vigas, placas, membranas entre otros.)
- Opciones de análisis lineal y no lineal, para casos estáticos y dinámicos.
- Cálculo automático de coordenadas de centros de masas.
- Cálculo automático de coordenadas de centros de rigideces.
- Cálculo automático de fuerzas sísmicas y aplicación en el centro de masas.
- Cuenta con elementos cable y elementos de pórtico.

Entre las diferentes opciones de análisis listadas en la página oficial de *ETABS*, se encuentran: Análisis estáticos lineales y no lineales, aplicación de cargas según la secuencia en la construcción, para casos especiales el análisis incluye el efecto del amortiguamiento, análisis considerando el efecto P-delta, análisis modales, de espectros de respuesta dinámicos, análisis historia-tiempo, dinámicos lineales y no lineales.

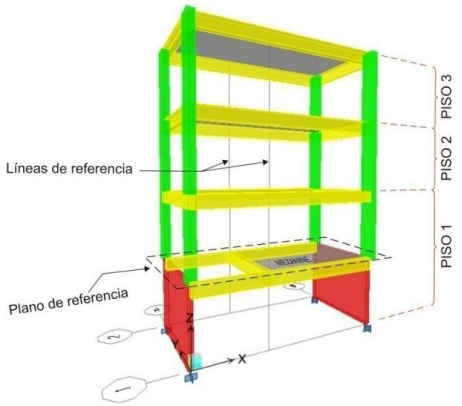

Figura 1.1: Visualización en 3D de una estructura usando *ETABS*. Tomada de Taboada G. (2009)

ETABS al igual que *SAP2000*, comparte las mismas características a nivel de sistema, es decir, es un *software* comercial, con licencias privadas que funciona sólo bajo sistema operativo Windows. Para su funcionamiento y aprovechamiento de sus herramientas de visualización y análisis se requiere una computadora con ciertas especificaciones especiales.

Además de las aplicaciones comerciales citadas también hay una variedad de herramientas académicas para el análisis de estructuras aporticadas como las siguientes:

1.1.1.3 IDARC

Es una herramienta de análisis desarrollada por University at Buffalo (1987) su distribución es gratuita, pero su codigo fuente es privado. Esta aplicación es usada principalmente para evaluar la respuesta de edificios aporticados de concreto armado y acero bajo cargas sísmicas. Por lo tanto, es un programa académico orientado a resolver problemas de la ingeniería estructural.

El programa incluye los tipos de elementos básicos estructurales: Elemento columna, Elemento viga, Elemento de muro de corte. El elemento columna toma en cuenta la deformación inelástica a flexión, el corte elástico y la deformación axial. El elemento viga usa modelos que consideran la rigidez a flexión no lineal con deformaciones por cortante elásticas lineales. El elemento de muro de corte incluye efectos de corte inelásticos y deformaciones de flexión.

Una de las características más resaltantes y novedosas del programa constituye el modelo de plasticidad distribuida, el cual difiere con el modelo de la rótula plástica usado por el programa *SAP2000* para el análisi de pórticos de acero, y el programa calcula el índice de daño propuesto por Park y Ang (1984) para los elementos de la estructura.

Para el caso de cargas dinámica el programa cuenta con modelos histeréticos, los cuales permiten representar el proceso de degradación del material durante cargas reversibles. También están incluidos análisis cuasi-estáticos, pseudo-dinámicos y está considerado el efecto P-Delta.

Para realizar el análisis estructural se debe ejecutar las siguientes tres fases:

1. Sistema de identificación: Análisis estático para determinar propiedades y el

modo de falla última del edificio.

2. Análisis de respuesta dinámico: Análisis inelástico dinámico paso a paso.

3. Análisis de daño: Evaluación del daño.

1.1.1.4 Drain

Desarrollado por University of California (1993), *Drain2D* es un programa académico con licencia y código privado, para el análisis estático y dinámico de estructuras planas, siendo su principal ventaja la simplicidad en el manejo de las herramientas. Una característica resaltante del programa, es que permite al usuario guardar las salidas al final de cada paso, así como también detener el análisis y luego restaurarlo desde cualquier paso guardado. El proceso de integración paso a paso para el análisis dinámico permite variar el tiempo de duración impuesto por el usuario.

Para al comportamiento del concreto armado, el modelo aplica un procedimiento de análisis que permite ver el estado de seguridad de las construcciones existentes antes o después de un movimiento sísmico. Dicho modelo, es capaz de tomar en cuenta la influencia de la fuerza axial en el comportamiento a flexión de los elementos viga-columna. También puede representar la degradación de los esfuerzos durante las cargas cíclicas, simulando la fatiga del material bajo este tipo de cargas.

1.1.1.5 Portal de Pórticos 2D

El programa Portal de Pórticos 2D (PDP) es una herramienta de cálculo desarrollada en la Universidad de Los Andes (1990), su objetivo principal es proporcionar a los ingenieros y analístas de estructuras instrumentos teóricos, numéricos y computacionales para que puedan ser usados en la reducción del riesgo sísmico, mediante la simulación numérica del proceso de daño estructural y del colapso de edificaciones de concreto armado cuyos modelos matemáticos se basan en la Teoría del daño concentrado.

El programa Portal de Pórticos es un programa de elementos finitos no lineal que ha sido diseñado para ser accedido exclusivamente a través de navegadores WEB, el mismo se encuentra disponible en: `http://portaldeporticos.ula.ve`. Es una aplicación de acceso gratuito cuyo código fuente no ha sido liberado. El programa permite la

simulación numérica del proceso de agrietamiento y posible colapso de pórticos 2D de concreto armado sometidos a solicitaciones sísmicas o desplazamientos extraordinarios de sus apoyos.

Para su utilización se requiere que el usuario deba abrir una cuenta, digitalizar el pórtico, generar el archivo de entrada y colocarlo en su cuenta dentro del servidor ya que las simulaciones numéricas no se realizan en la computadora personal del usuario sino en el servidor remoto de alto rendimiento. Una vez realizado el análisis se accede a un post-procesador gráfico para visualizar los resultados.

La estructura general del portal está conformada por tres módulos principales: Preprocesador, Procesador y Postprocesador. Las pantallas principales correspondientes a cada módulo se muestran en la Figura 1.2.

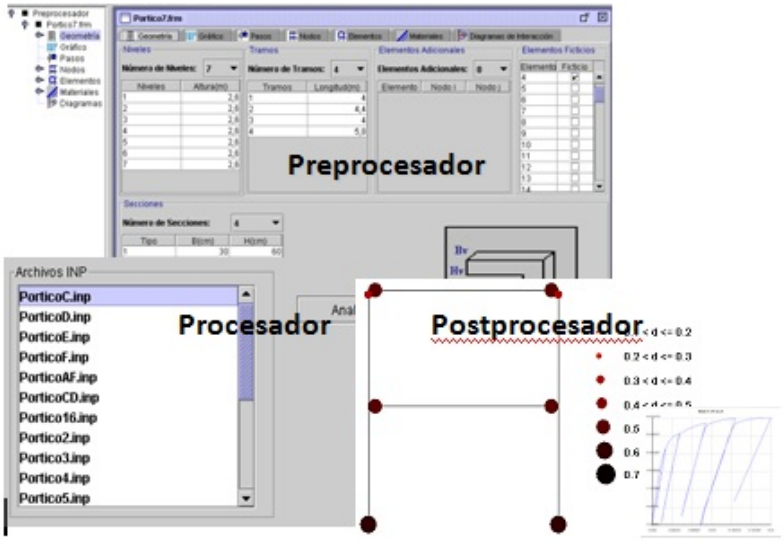

Figura 1.2: Pantallas de cada uno de los módulos del Portal de Pórticos 2D. Tomada de Uzcátegui Flores (2012)

Módulo Preprocesador: Permite la digitalización del pórtico y sus solicitaciones. Esta digitalización consiste en describir la geometría del pórtico, las solicitaciones, las propiedades de las secciones transversales, descripción de los materiales (concreto y el acero) utilizado.

Módulo Procesador: Es una interfaz con el programa de elementos finitos que se encuentra residenciado en el centro de cómputo de alto rendimiento. Este programa de elementos finitos permite el análisis inelástico, estático o dinámico y geométricamente no lineal de la estructura. En el Procesador se accede a la lista de archivos que el usuario tiene en su cuenta y se escoge cual debe ser procesado mediante el programa de elementos finitos.

Módulo Postprocesador: Permite visualizar los resultados del análisis mediante gráficas de variable contra variable, variables contra tiempo y mapas de distribución de daños en cualquier instante del análisis, es decir, permite mostrar al usuario por medio de gráficos, distribuciones y animaciones el comportamiento de la estructura analizada.

Al ser una interfaz gráfica que puede ser accedida usando un navegador WEB permite establecer una comunicación entre el usuario y un servidor remoto que contiene el programa principal de elementos finitos. Este servidor remoto se encuentra ubicado y administrado por CeCalCULA (Centro de Cálculo Científico de la Universidad de Los Andes). El programa como servicio WEB está caracterizado por 3 componentes principales: Un Applet (programa que se ejecuta en la maquina del cliente incrustado en una página WEB), el Servlet (que se ejecuta del lado del servidor), y los programas Fortran que ejecutan los cómputos. En la Figura 1.3 se ilustra la arquitectura del servicio.

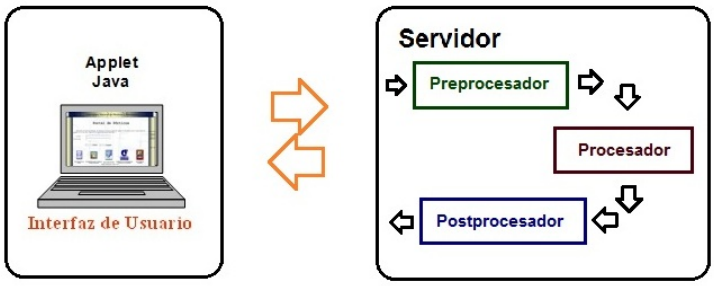

Figura 1.3: Arquitectura del Portal de Pórticos 2D como servicio WEB

En la tabla 1.1 se muestra una breve comparación de las caracteristicas que posee cada una de las herramientas definidas anteriormente.

Tabla 1.1: Comparación entre las herramientas

Características	Sap2000	ETABS	IDARC 2D	DRAIN 2D	PDP 2D
Diseño Estructural	Si	Si	Si	Si	No
Análisis Espacial	Si	Si	No	No	No
Diagnóstico Estruc.	No	No	Si	Si	Si
Daño Estructural	No	No	Si	No	Si
Análisis Estático	Si	Si	Si	Si	Si
Análisis Dinámico	Si	Si	Si	Si	Si
Tipo de Desarrollo	Comercial	Comercial	Académico	Académico	Académico
Sistema Operativo	Win	Win	Win	Win	Win/Unix
Tipo de *Software*	Privativo	Privativo	Privativo	Privativo	Privativo

1.1.2 Programa Endógeno de Elementos Finitos (PEEF)

El Programa Endógeno de Elementos Finitos (PEEF) es un programa escrito en lenguaje de programación Fortran y desarrollado por Uzcátegui (2012). Es un programa de propósito general para el análisis de sólidos, que permite realizar el análisis lineal y no lineal de estructuras en tres y dos dimensiones, bajo condiciones estáticas y dinámicas. PEEF está estructurado de tal manera que permite la inclusión de nuevas subrutinas de elementos finitos, independientemente de la cantidad de nodos y elementos estructurales que posee la geometría del sólido. Actualmente PEEF tiene un conjunto de elementos finitos que permite el análisis de estructuras de concreto armado, sistemas duales, incluyendo la mampostería de relleno.

Para realizar el análisi el programa resuelve un sistema de ecuaciones formado por las ecuaciones de equilibrio, cinemáticas, condiciones externas impuestas y la ley de comportamiento. Básicamente el problema de análisis de la estructura se reduce a la resolución de la ecuación de equilibrio, la cual puede reducirse a un sistema de ecuaciones diferenciales ordinarias no lineales en el tiempo.

Al ser no lineal este problema no puede ser resuelto directamente por metodos lineales, por lo que emplea un método iterativo de tal manera que en cada iteración del método, el problema se pueda resolver como un problema lineal. A la solución del sistema de ecuaciones formado por la ecuación de equilibrio y las condiciones externas

impuestas en cada iteración, se denomina “problema global”.

En cada iteración del problema global es necesario calcular las matrices de fuerzas internas y de rigidez en coordenadas globales para cada uno de los miembros de la estructura. La resolución de este problema para cada elemento se denomina “problema local” Uzcátegui Flores (2012).

Por lo tanto, para cada iteración se tiene un problema global y varios problemas locales por cada elemento (viga, columna, placa, etc) de la estructura. En consecuencia, el programa de elementos finitos se encuentra estructurado de la siguiente manera:

- Problema Global (PEEF): Conjunto de subrutinas que se encargan de resolver la ecuación de equilibrio.
- Problema Local: Subrutina que contiene al elemento fininito a través de la cual se obtiene el comportamiento de la estructura.

PEEF utiliza un procedimiento de resolución paso a paso, donde el intervalo de tiempo continuo [0,T] durante el cual se desea analizar el comportamiento de la estructura es sustituido por un conjunto discreto de instantes (0,t1,t2,...,tr,...,T). Esto significa que las incógnitas del problema no son calculadas para el intervalo completo sino unicamente en los instantes escogidos tr. El cálculo se hace de forma secuencial, es decir, antes de proceder al análisis de la estructura en el instante tr se debe realizar previamente el cálculo en el instante tr-1. Al concluir el cálculo se dice que se ha realizado un paso de tiempo.

Con el fin de aproximar las derivadas con respecto al tiempo, se emplea el algoritmo de integración numérica del método de Newton. Este método consiste en utilizar fórmulas aproximadas que expresan las velocidades y las aceleraciones al final de cada paso como funciones de esas mismas cantidades, y de las deformaciones al principio del paso (conocidas) y de las deformaciones al final del paso.

El uso de este tipo de algoritmos de integración permite junto con las ecuaciones cinemáticas, de equilibrio dinámico y la ley de comportamiento expresar las fuerzas internas en un instante tr como una función de los desplazamientos globales de la estructura.

La ecuación de equilibrio en un instante de tiempo conjuntamente con los desplazamientos impuestos define el llamado "problema global".

La convergencia del método de Newton depende de que tan cerca esté la matriz de desplazamientos utilizada en la primera iteración, con respecto a la solución del sistema de ecuaciones. En otras palabras, la convergencia del método de Newton afecta el tamaño del paso de cálculo. En el procedimiento este paso de cálculo es único y debe satisfacer la convergencia tanto del problema global como de los multiples problemas locales. la Figura 1.4 muestra un diagrama de la solución del problema global.

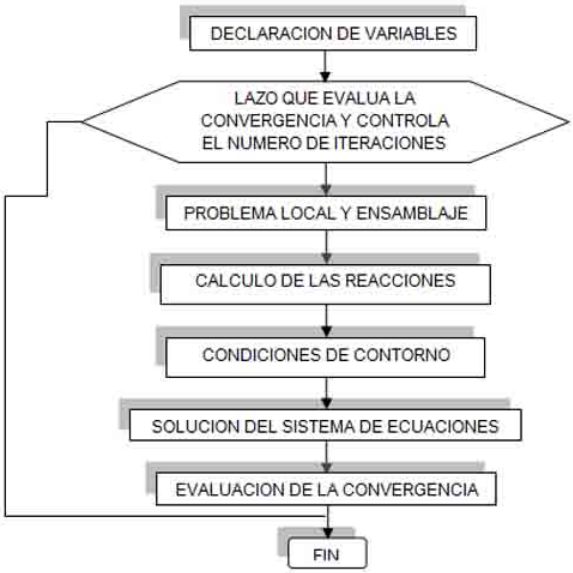

Figura 1.4: Diagrama de flujo para la solución del problema global. Tomada de Uzcátegui Flores (2012)

De esta manera, si la falta de convergencia se da en alguno de los n problemas locales, se reduce el paso de cálculo de los n+1 problemas considerados.

La resolución del problema local consiste en el cálculo de las fuerzas internas, los valores de las variables internas (daño y deformaciones plásticas) y el Jacobiano para cada uno de los n elementos se realiza a través de la subrutina *ELIB* (Subrutina que contiene el elemento finito de la librería). Este cálculo se efectúa utilizando los conceptos de esfuerzos y deformación, cuya definición específica depende del modelo

implementado en el elemento finito. Esta subrutina se acopla al programa para poder realizar el análisis de la estructura.

Posteriormente se determinan los esfuerzos, y variables internas. Para el cálculo de estos últimos se debe resolver un sistema de ecuaciones formado por las leyes de estado y las leyes de evolución de las variables internas. De no lograrse la convergencia después de una cantidad determinada de veces se repite el ciclo con un paso local más pequeño. Este procedimiento se repite hasta alcanzar el valor del paso global.

Una vez lograda la convergencia local se realiza el cálculo de las fuerzas internas a partir de los esfuerzos generalizados. Finalmente, se determina el Jacobiano local en coordenadas globales. La Figura 1.5 ilustra el diagrama flujo para la solución del problema local.

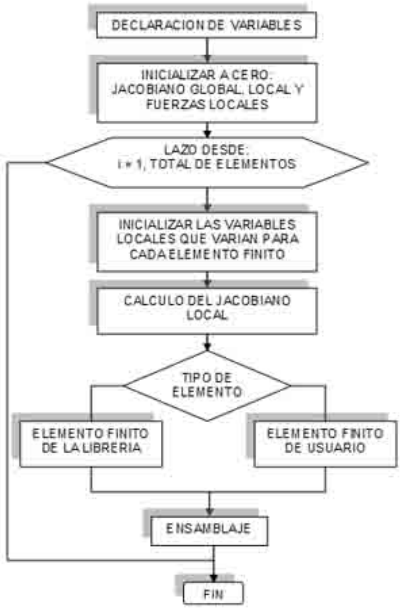

Figura 1.5: Diagrama de flujo para la solución del problema local. Tomada de Uzcátegui Flores (2012)

Anteriormente PEEF con los siguientes elementos finitos:

- VC2DCA: Viga - Columna 2D de concreto armado. Este elemento es usado para definir elementos estructurales tales como vigas y columnas de concreto armado,

para el análisis de pórticos planos. Se tienen 3 grados de libertad activados, desplazamiento en la dirección X (1), desplazamiento en la dirección Z (3), rotación alrededor del eje Y (5). Análisis en el plano (2D).

- VC3DCA: Viga - Columna 3D de concreto armado. Este elemento es usado para definir elementos estructurales tales como vigas y columnas de concreto armado, para el análisis de estructuras tridimensionales. Se tienen 6 grados de libertad activados, desplazamiento en la dirección X (1), desplazamiento en la dirección Y (2), desplazamiento en la dirección Z (3), rotación alrededor del eje X (4), rotación alrededor del eje Y (5), rotación alrededor del eje Z (6). Análisis en el espacio (3D).

- M2DMR: Elemento muro 2D de mampostería de relleno. Este elemento es usado con la finalidad de incluir paneles de relleno en el análisis de estructuras aporticadas en dos dimensiones. Se tienen 2 grados de libertad activados, desplazamiento en la dirección X (1) y desplazamiento en la dirección Z (3). Análisis en el plano (2D).

- VCM2DCA: Elemento Elemento Viga-Columna-Muro de corte 2D de concreto armado. Este elemento es usado para el análisis no lineal de estructuras duales, Viga-Columna y Muro, de concreto armado. Este tipo de estructura está conformada por muros de corte y pórticos dúctiles a flexión. Análisis en el plano (2D).

Actualmente PEEF tiene incorporado un elemento finito adicional, el elemento CSTDKT. Este elemento finito modela el efecto en las vigas de las losas de entrepiso. El desarrollo matemático se puede apreciar en el capítulo 2 del presente documento.

PEEF a futuro podria albergar una cantidad mayor de elementos finítos y convertirse en una herramienta usada en diferetes aplicaciones de uso específico tanto en el área estructural como en otra rama de la ingeniería.

1.1.3 Modelo Cliente/Servidor

Según Comer (1995), el término servidor se aplica a cualquier programa que ofrece un servicio que se puede obtener en una red. Un servidor acepta la petición desde la red, realiza el servicio y devuelve el resultado al solicitante. En el caso de los servicios más sencillos, cada petición llega en un solo datagrama IP y el servidor devuelve una respuesta en otro datagrama.

Un programa ejecutable se convierte en un cliente cuando manda una petición a un servidor y espera una respuesta. Debido a que el modelo Cliente/Servidor es de extensión conveniente y natural en la comunicación de interproceso es una sola maquina, es fácil construir programas que utilicen el modelo para interactuar.

Los servidores se suelen implantar como aplicaciones de programas. La ventaja de implantar los servidores como programas de aplicación es que pueden ejecutarse en cualquier sistema computacional que soporte la comunicación TCP/IP. De este modo el servidor de un servicio en particular puede ejecutarse en un sistema de tiempo compartido junto con otros programas o en una computadora personal.

WIKIPEDIA (2012a) en su artículo en ingles define la arquitectura Cliente/Servidor como un modelo de aplicación distribuida en el cual los procesos se reparten entre los proveedores de recursos, llamados servidores y aquellos que realizan la petición, llamados clientes. Un cliente realiza el llamado de un proceso al servidor encargado en dar respuesta ante la solicitud. El esquema de interacción entre los dos componentes se muestra en la Figura 1.6.

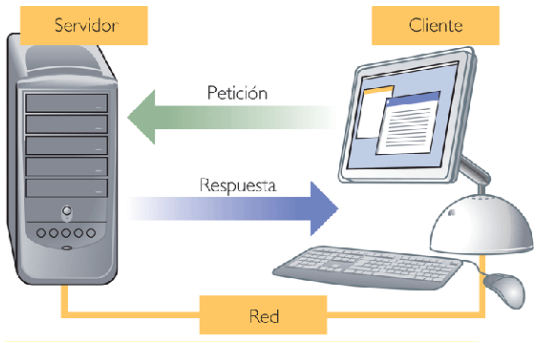

Figura 1.6: Interacción típica entre el Cliente y el Servidor. Tomada de PRISA (2012)

El término Cliente/Servidor fue usado por primera vez en 1980 para referirse a

computadoras en red. Este modelo empezó a ser aceptado a finales de los 80's. Márquez Avendaño (2004)

La arquitectura Cliente/Servidor modela la más básica realización de aplicaciones distribuidas. La principal ventaja de esta arquitectura es que permite separar las funciones según su servicio, permitiendo situar cada función en las plataformas más adecuadas para su ejecución. Uno de los ejemplos de esta arquitectura es el funcionamiento de las páginas WEB, donde un programa cliente, un servidor WEB, solicita servicios (el envío de páginas Web o archivos) a un servidor WEB en otra computadora en algún lugar de la Internet que contiene el código.

Algunas de las características del cliente se mencionan a continuación:

- Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación.
- Espera y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.
- Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario (GUI).

Algunas de las características del servidor se listan:

- Al iniciarse el servidor se esperan las solicitudes de los clientes, por lo tanto desempeñan entonces un papel pasivo en la comunicación.
- Tras la recepción de una solicitud, se procesan y luego se envía la respuesta al cliente.
- Aceptan conexiones desde un gran número de clientes.

Márquez Avendaño (2004) define algunas de las características de la arquitectura Cliente/Servidor como:

- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, capacidad y velocidad del disco duro así como sus dispositivos de entrada y salida.

- No existe otra relación entre clientes y servidores que no sea la de establecer un intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.
- El ambiente es heterogéneo. La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre la misma. Precisamente una de las principales ventajas de la arquitectura.

Entre las múltiples ventajas que aporta la arquitectura Cliente/Servidor se mencionan las más representativas:

- Existencia de plataformas de hardware cada vez más baratas. Esto constituye a su vez una de las más palpables ventajas de este modelo, la posibilidad de utilizar máquinas considerablemente más baratas que las requeridas por una solución centralizada, basada en sistemas grandes.
- El uso de interfaces gráficas para el usuario en este modelo presenta ventaja con respecto a uno centralizado en el sentido de la necesidad de no transmitir información gráfica por la red, pues esta información puede residir en el cliente permitiendo aprovechar mejor el ancho de banda de la red.
- El mantenimiento y desarrollo de aplicaciones es más rápido, pues se pueden emplear herramientas existentes (Servidores SQL, herramientas de bajo nivel).
- La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.
- Se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado o mejorado en cualquier momento, como también se pueden añadir nuevos nodos a la red (clientes y/o servidores).

La arquitectura Cliente/Servidor presenta algunas desventajas, como:

- La congestión del tráfico ha sido siempre un problema en el paradigma de C/S. Cuando una gran cantidad de clientes envían peticiones simultaneas al mismo

servidor, puede ser que cause muchos problemas, es decir, a mayor número de clientes, más problemas para el servidor.

- Cuando un servidor está caído o no operativo, las peticiones de los clientes no pueden ser satisfechas.

Ibarra A. (2012) clasifica la evolución del modelo Cliente/Servidor de la siguiente forma:

- Mono-capa
- Data Base Server - Computación centralizada
- Two-Tier - Proceso de transacciones
- Multi-tier Client/Server
- Three-tier
- Multi-tier
- N-tier

Definiendo los tres modelos más comunes Ibarra A. (2012)

Aplicaciones mono-capa: Una aplicación mono-capa, es aquella que tanto la propia aplicación como los datos que maneja se encuentran en la misma máquina y son administradas por la misma herramienta.

Modelo En Dos Capas (Two-Tier Model): Es una arquitectura Cliente/Servidor clásica, donde se tienen dos "capas" (two-tier): Una donde está el cliente que implementa la interfaz y otra donde se encuentra el gestor de bases de datos que trata las peticiones recibidas desde el cliente. La lógica de la aplicación se encuentra por tanto repartida entre el cliente y servidor. Un ejemplo de esta configuración es el funcionamiento de una pagina Web, donde el cliente solicita la información al servidor que la contiene.

Una de las ventajas de trabajar en un modelo de dos capas es que permite mantener una conexión con la base de datos, además se minimizan las peticiones en el servidor trasladándose la mayor parte del trabajo al cliente y se gana en rendimiento gracias a la conexión directa y permanente con la base de datos. A través de una única conexión se realiza el envío y recepción de varios datos.

Modelo en Tres Capas (Three-Tier Model): Con la arquitectura Cliente/Cervidor en tres capas (three-tier) se añade una nueva capa entre el cliente y el servidor donde se implementa la lógica de la aplicación. De esta forma el cliente es básicamente una interfaz quedando aislado completamente del acceso a los datos.

Así un applet de JAVA se aloja en el navegador del cliente y este invoca un servlet que corre en la máquina servidor; o bien accedemos a la base de datos a través de un formulario HTML. El servlet establece una conexión a la base de datos mediante JDBC.

En este caso se tiene total libertad para escoger dónde se coloca la lógica de la aplicación: en el cliente, en el servidor de base de datos, o en otro servidor. También se tiene total libertad para la elección del lenguaje a utilizar. Cada uno de los componentes de la aplicación en una arquitectura three-tier se separa en una sola entidad. Esto te permite implementar componentes de una manera más flexible. En esta Arquitectura todas las peticiones de los clientes se controlan en la capa correspondiente a la capa intermedia. Cuando el cliente necesita hacer una petición se la hace a la capa intermedia con el propósito de que el cliente no tenga inconvenientes de comunicación producto a la instalación de controladores (drivers) adicionales.

1.1.4 Servicio Web

Según The World Wide Web Consortium (W3C) (2012) un servicio Web es un sistema de *software* diseñado para apoyar la interoperabilidad entre máquinas que interactúan sobre una red. Tiene una interfaz descrita en un formato procesable por máquina (específicamente Web Services Description Language, conocido como WSDL). Otros

sistemas interactúan con el servicio Web de una manera prescrita por su descripción utilizando mensajes SOAP, transportados usando HTTP con una serialización XML en conjunto con otros estándares Web relacionados.

WIKIPEDIA (2012b) en su artículo en español define los servicios Web como conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones de *software* desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma.

Saffirio (2006), en su blog hace la siguiente definición: un Servicio Web describe una forma estandarizada de integrar aplicaciones WEB mediante el uso de XML, SOAP, WSDL y UDDI sobre los protocolos de la Internet. XML es usado para describir los datos, SOAP se ocupa para la transferencia de los datos, WSDL se emplea para describir los servicios disponibles y UDDI se ocupa para conocer cuáles son los servicios disponibles. Uno de los usos principales es permitir la comunicación entre las empresas y entre las empresas y sus clientes.

Los servicios Web están construidos por varias tecnologías que trabajan conjuntamente con los estándares que están emergiendo para asegurar la seguridad y operatividad. Estos estándares se definen brevemente a continuación y se ilustra su conexion en la Figura 1.7:

XML: Abreviación de Extensible Markup Language. El XML es un lenguaje desarrollada por el *World Wide Web Consortium* (W3C)WIKIPEDIA (2012b). Permite a los desarrolladores crear sus propios tags (comando que se inserta en un documento con el fin de especificar el formato del documento), que les permiten habilitar definiciones, transmisiones, validaciones, e interpretación de los datos entre aplicaciones y entre organizaciones.

SOAP: Abreviación de Simple Object Access Protocol, es un protocolo de mensajería basado en XML que se usa para codificar información de los requerimientos de los servicio Web y para responder los mensajes antes de enviarlos por la red. Los mensajes SOAP son independientes de los sistemas operativos y pueden ser transportados por los protocolos que funcionan en la Internet, como ser: SMTP, MIME y HTTP.

WSDL: Abreviación de Web Services Description Language, es un lenguaje especificado en XML que se ocupa para definir los servicios Web como colecciones de punto de comunicación capaces de intercambiar mensajes. El WSDL es parte integral de UDDI y parte del registro global de XML, en otras palabras es un estándar abierto (no se requiere pagar licencias ni regalías para usarlo).

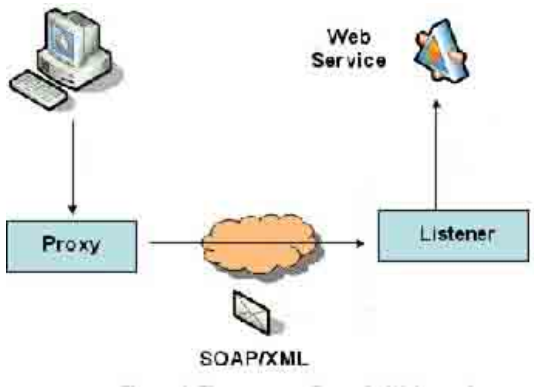

Figura 1.7: Diagrama de un Servicio Web.

Los servicios Web permiten a distintas aplicaciones, de diferentes orígenes, comunicarse entre ellos sin necesidad de escribir programas costosos. Esta comunicación es posible ya que se hace con XML.

A diferencia de los modelos Cliente/Servidor, tales como un servidor de páginas Web, los servicios Web no proveen al usuario una interfaz gráfica (GUI). En vez de ello, los Web Services comparten la lógica del negocio, los datos y los procesos, por medio de una interfaz de programas a través de la red. Es decir conectan programas que no interactúan directamente con los usuarios.

Entre las ventajas que ofrecen los servicios Web se mencionan algunas, consideradas como las más representativas de esta tecnología:

- Aportan interoperabilidad entre aplicaciones de *software* independientemente de sus propiedades o de las plataformas sobre las que se instalen. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de

los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.

- Permiten a las organizaciones intercambiar datos sin necesidad de conocer los detalles de sus respectivos Sistemas de Información.

1.1.5 Portales de Cálculo

Dadas las nuevas herramientas, la tendencia es que las aplicaciones científicas dispongan de una interfaz WEB que permite su manipulación y, sobre todo su ubicuidad de acceso. Esta facilidad tiene un importante beneficio adicional que es la popularización del uso de sofisticadas aplicaciones científicas mediante el uso de Portales.

Los portales son interfaces que permiten a las comunidades de usuarios acceder de forma segura, a las aplicaciones, a los datos, además permiten compartir y difundir los conocimientos. La utilidad misma de los portales como herramientas de interacción y producción de conocimiento ha ido cambiando la definición misma de portales y existe una amplia variedad. Los portales de aplicaciones científicas se constituyen en un importante tipo. Si bien los portales de aplicaciones científicas poseen algunas características similares a los sitios Web comerciales (autenticación, personalización del ambiente de trabajo, registro histórico de su actividad y ubicuidad de uso, por mencionar algunos de los más comunes), son sistemas mucho más sofisticados que van más allá de una simple interfaz WEB. Uzcátegui Flores (2007)

Entre, los servicios a los cuales se tiene acceso a través de este tipo de interfaces se pueden nombrar:

- Seguridad en el acceso: Los usuarios son autenticados, bien sea a través de claves de acceso o certificados digitales.

- Manejo de datos: Permite el acceso a una estructura de archivos, datos y metadatos, tanto locales como remotos. Igualmente permite la transferencia de

archivos de datos entre los sistemas remotos y locales.

- Ejecución y seguimiento de procesos: Permite ubicar y reubicar procesos en distintos sistemas computacionales remotos, así como el seguimiento del estado de su ejecución.

- Servicios de información: Provee información sobre la disponibilidad de recursos (CPU, almacenamiento, comunicación entre los distintos nodos).

- Herramientas de colaboración: algunos portales incorporan herramientas (videoconferencias, chats, Voz sobre IP) para que la comunidad pueda interactuar y compartir la información.

- Servicios de visualización y análisis de datos: Permite analizar los datos a través de ambientes de visualización, minería de datos y/o análisis estadísticos de datos, compartiendo estos resultados con otros miembros de la comunidad.

- Gestión de Usuarios: Es un proceso que incluye la adición, el mantenimiento y la eliminación de perfiles de los diferentes usuarios que hacen vida dentro del sistema. Además el usu y modificaciones del password, llave necesaria para la autenticación del usuario.

1.2 Planteamiento del Problema

La filosofía clásica de los programas de elementos finitos conlleva al uso de sofisticadas herramientas y modelos matemáticos, para la evaluación de un amplio rango de problemas de la ingeniería estructural. Su concepción se basa en programas de aplicaciones generales, que comparten el mismo preproceso, proceso y postproceso, independientemente del tipo de análisis. Esto trae como consecuencia que su uso requiera de un personal altamente capacitado para el manejo y evaluación de resultados, es decir están diseñados para especialistas. Adicionalmente los requerimientos computacionales pueden ser en ocasiones bastante elevados al igual que los tiempos de ejecución, aumentando así los costos de la evaluación.

Específicamente uno de los problemas de la ingeniería estructural es el colapso masivo de edificios bajo cargas sísmicas, este colapso se debe principalmente a deficiencias en los materiales o debilidades en el diseño y/o construcción. Muy probablemente existen centenares de miles de edificaciones en el mundo, miles de ellas en Venezuela que no son seguras y necesitan ser reforzadas. Por lo tanto, la identificación de estas estructuras inseguras es una tarea urgente. Uno de los procedimientos que pueden ser usados para el diagnostico de la seguridad de edificaciones es la simulación numérica. Sin embargo, dada la magnitud del problema a resolver, este es un trabajo que no puede ser resuelto por un pequeño grupo de especialista, por el contrario debe ser solucionado por un grupo numeroso de ingenieros y técnicos localizados en cada zona a evaluar.

En consecuencia y con el fin de proveer de herramientas con características de libre acceso, simples, y con capacidad tecnológica y científica de alto nivel, en esta tesis se propone el Portal de Pórticos 3D. El cual constituye la primera aplicación de uso específico empleando el programa de elementos finitos PEEF. El cual permite la simulación de estructuras aporticadas tridimensionales de concreto armado.

Así mismo como aporte adicional en la evaluación de las estructuras tridimensionales de concreto armado se plantea el desarrollo de un elemento finito (DKTCST3D), con el fin de incluir elementos tipo placa y así tomar en cuenta el efecto de las losas de entrepiso.

1.3 Justificación

Las Tecnologías de Información y Comunicación (TIC) forman parte de la nueva cultura tecnológica, pues constituyen un factor de cambio de la sociedad actual. Las TIC se mueven al ritmo de los avances tecnológicos, y dentro de un mundo globalizado provocan continuas transformaciones en las estructuras económicas, sociales y culturales, permitiendo una redefinición radical del funcionamiento de la sociedad.

De todos los elementos que conforman las TIC, sin duda el más poderoso y revolucionario es Internet, ya que la información es accedida, procesada y consolidada mediante técnicas de representación del conocimiento y minería de datos por intermedio de interfaces Web. En la actualidad la Tecnología de la Información y Comunicación

(TIC) juega un papel importante en el desarrollo de las actividades científico-técnicas, favoreciendo que un mayor número de usuarios tengan acceso a la investigación global para su manipulación y sus aplicaciones en las diferentes áreas de la sociedad.

Los portales son estas interfaces que permiten a las comunidades de usuarios acceder de forma segura, a las aplicaciones, a los datos, además permiten compartir y difundir los conocimientos. Adicionalmente esta facilidad tiene un importante beneficio, que consiste en la popularización del uso de sofisticadas aplicaciones científicas mediante el uso de Portales.

Por lo tanto, con el fin de ampliar el espectro en el uso de los elementos finitos, generando una mayor simplicidad, en esta tesis se desarrolló un portal web que permite la evaluación de estructuras aporticadas tridimensionales de concreto armado bajo cargas de servicios y sobrecargas.

1.4 Objetivos

1.4.1 Objetivo General

Diseñar y desarrollar un servicio WEB, basado en el modelo cliente servidor, accedible en Internet desde computadores personales (PC), para el análisis, diseño y evaluación de estructuras aporticadas tridimensionales de concreto armado.

1.4.2 Objetivos Específicos

- Precisar el estado del arte en el modelado de estructuras aporticadas y evaluación de riesgos estructural.
- Diseñar y configurar el servidor WEB bajo una plataforma Cliente/Servidor para el análisis y evaluación de estructuras aporticadas tridimensionales de concreto armado.
- Aplicar los protocolos de interacción vía WEB entre el servidor y los clientes.
- Desarrollar el servidor propuesto implementando la plataforma y los protocolos seleccionados.

- Desarrollar el Cliente PC como un módulo autónomo para el análisis y evaluación de estructuras aporticadas tridimensionales de concreto armado.
- Integrar el Cliente PC con el Servidor.
- Escribir el elemento finito tipo placa.
- Incorporar el elemento finito tipo placa a PEEF.
- Probar y Depurar el elemento finito.
- Probar y Depurar el sistema integrado.

1.5 Metodología

Primeramente se precisó el estado del arte en el modelado de estructuras aporticadas y evaluación de riesgos estructural. Para el desarrollo de la aplicación se utilizó la metodología *Scrum*, una metodología de desarrollo ágil. Esta es una estrategia iterativa donde hay que regresar a ciertos puntos para refinar o rediseñar dependiendo de la magnitud del error. A continuación se definen los pasos a seguidos para el desarrollo del proyecto.

1. Se diseñó y configuró un servidor WEB bajo una plataforma Cliente/Servidor para análizar y evaluar estructuras aporticadas tridimensionales de concreto armado utilizando PEEF como programa de elementos finítos de propósitos generales. (Ver detalles en el capítulo 3)
2. Se implementó los protocolos de interacción vía WEB entre el servidor y los clientes.
3. Se desarrolló el servidor propuesto implementando la plataforma y los protocolos seleccionados.
4. Se desarrollaró el Cliente PC como un módulo autónomo para el análisis y evaluación de estructuras aporticadas tridimensionales de concreto armado.

5. Se integraron el cliente PC y el servidor como sistema conjunto.

6. Se probó y depuró el sistema integrado. (Ver detalles en el capítulo 4)

7. Se desarrolló e implementó el elemento finito de tipo placa basado en la teoria de placas de Kirchhoff. En el capítulo 2 se explica con más detalle el proceso.

8. Se integró a PEEF, junto con los otros elementos finitos, el elemento finito de tipo placa.

9. Se realizó algunas pruebas y depuraciónes para el desempeño del elemento finito tipo placa, una sección del capítulo 2 está dedicada a las pruebas .

Capítulo 2

Desarrollo de un elemento finito tipo placa para el análisis de estructuras tridimensionales

El Programa Endógeno de Elementos Finitos (PEEF) cuenta con un elemento para el análisis de pórticos en tres dimensiones, sin embargo presenta una desventaja y es que el efecto de las losas de una edificación no son tomadas en cuenta directamente por el programa. Para ello el usuario debe incluir su efecto como una carga uniformemente repartida sobre el elemento viga.

Para resolver este problema, en este capítulo se describen las ecuaciones usadas para la programación del elemento finito de placa general (CSTDKT), el cual está sometido a fuerzas en el plano y a fuerzas de flexión. Para tomar en cuenta estas dos acciones, se desarrolla el elemento de membrana (CST) y el elemento de placa de flexión (DKT).

2.1 Desarrollo del elemento finito Membrana (CST)

Para desarrollar el elemento finito tipo membrana se utilizan dos elementos finitos, como lo son el triángulo de deformación constante (CST por sus siglas en inglés) y el cuadrilátero isoparamétrico de 4 nodos, Chaves Eduardo (2010). Los cálculos de la

matriz de rigidez y la implementación se realizaron en función del elemento triángulo de deformación.

2.1.1 Esfuerzo y deformación en 2 dimensiones

El problema de Elasticidad en dos dimensiones incluye estructuras muy delgadas y las fuerzas son aplicadas en dirección del plano de la estructura. Se considera una estructura en el plano XY con un espesor t. Cuando las fuerzas son aplicadas a la estructura, se produce un desplazamiento de los puntos de la estructura; este punto está localizado en un par de coordenadas (x,y), donde:

$$U = \{u, v\}^T \tag{2.1}$$

Los vectores u y v son vectores en función de las coordenadas x e y de cada punto. Los esfuerzos (σ) y las deformaciones (ε) del elemento se obtienen de la siguiente ecuación:

$$\begin{aligned} \sigma &= \{\sigma_x, \sigma_y, \sigma_x y\}^T \\ \varepsilon &= \{\varepsilon_x, \varepsilon_y, \varepsilon_x y\}^T \end{aligned} \tag{2.2}$$

Cuando la estructura está sometida a fuerzas en el mismo plano, el estado de los esfuerzos y las deformaciones se llama Condiciones del Plano de Esfuerzo, como la placa es delgada, esta se somete a fuerzas en el plano exclusivamente, los desplazamientos y los esfuerzos normales al plano de la membrana se asumen nulos, además se asume que la placa delgada o membrana esta en el plano xy, por lo tanto, los esfuerzos $\sigma_z = 0$, $\tau_y z = 0$, $\tau_x z = 0$ y la deformación $\varepsilon_z \neq 0$. Para los materiales isotrópicos la relación esfuerzo deformación en el plano de Esfuerzos resulta de la siguiente manera:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_x y \end{Bmatrix} = [E] \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_x y \end{Bmatrix} \tag{2.3}$$

Donde [E] es la Matriz del material y puede expresarse de la siguiente manera:

$$[E] = \frac{E}{1-v^2}\begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix} \tag{2.4}$$

El elemento más simple del plano de esfuerzo es el triángulo de deformación constante (CST). Este elemento tiene dos grados de libertad en el plano y posee 3 nodos, para un total de seis grados de libertad por elemento. El triángulo de deformación constante se utiliza ampliamente para diversos fines analíticos. Los nodos del elemento CST están enumerados en contra de las agujas del reloj, como se muestra en la Figura 2.1.

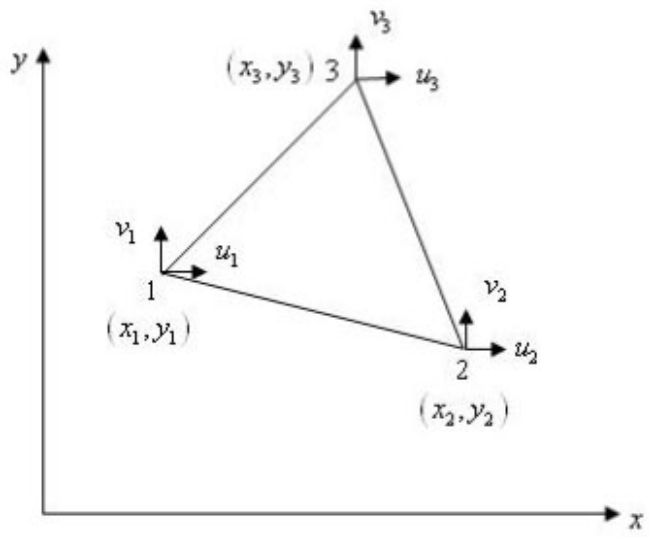

Figura 2.1: Nodos del elemento triangular.

Se procede a calcular la matriz de rigidez para este elemento triangular.

Se asume los desplazamientos:

$$\begin{aligned} u(x,y) &= a_0 + a_1 x + a_2 y \\ v(x,y) &= a_3 + a_4 x + a_5 y \end{aligned} \tag{2.5}$$

Donde u(x,y) es el desplazamiento en la dirección x, y v(x,y) es el desplazamiento en la dirección y.

Las ecuaciones anteriores pueden representarse de forma matricial quedando de la siguiente forma:

$$\begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{Bmatrix} \tag{2.6}$$

o

$$\{U(x,y)\} = [X]\{a\} \tag{2.7}$$

De la ecuación 2.3 y basándose en la relación deformación-desplazamiento se obtiene los siguientes resultados:

$$\varepsilon_x(x,y) = \frac{\partial u}{\partial x} = a_1 \tag{2.8}$$

$$\varepsilon_y(x,y) = \frac{\partial v}{\partial y} = a_5 \tag{2.9}$$

$$\gamma_{xy}(x,y) = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = a_2 + a_4 \tag{2.10}$$

Se observa que de las ecuaciones anteriores que los términos a_0 y a_3 representan las traslaciones del cuerpo rígido, mientras que los términos a_1 representa la constante de deformación en la dirección x y a_5 representa la constante de deformación en la dirección y, de la ecuación 2.10 se puede determinar que los términos a_2 y a_4 representan la deformación de corte uniforme. También se puede observar en la Figura 2.1, que las coordenadas de los nodos 1, 2 y 3 son (x_1, y_1), (x_2, y_2) y (x_3, y_3) respectivamente. Los desplazamientos correspondientes a cada nodo son (u_1, v_1), (u_2, v_2) y (u_3, v_3) respectivamente. Sustituyendo los valores de las coordenadas nodales en la ecuación 2.7, el resultado es el siguiente:

$$\begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{Bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & y_1 \\ 1 & x_2 & y_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_2 & y_2 \\ 1 & x_3 & y_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_3 & y_3 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{Bmatrix} \tag{2.11}$$

o

$$\{u\} = [A]\{a\} \tag{2.12}$$

Los coeficientes de a se obtienen al invertir la Ecuación 2.12

$$\{a\} = [A]^{-1}\{u\} \tag{2.13}$$

En la ecuación 2.7 se sustituye la ecuación 2.13 y resulta la siguiente expresión:

$$\{U(x,y)\} = [X][A]^{-1}\{u\} \tag{2.14}$$

Donde, $[X][A]^{-1}$ representa las funciones de forma $[N]$

$$[N] = [X][A]^{-1} \tag{2.15}$$

Invirtiendo la matriz $[A]$ en la ecuación 2.11 y resolviendo para a se obtiene:

$$\begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{Bmatrix} = \frac{1}{2A} \begin{bmatrix} x_2y_3 - x_3y_2 & 0 & x_3y_1 - x_1y_3 & 0 & x_1y_2 - x_2y_1 & 0 \\ y_2 - y_3 & 0 & y_3 - y_1 & 0 & y_1 - y_2 & 0 \\ x_3 - x_2 & 0 & x_1 - x_3 & 0 & x_2 - x_1 & 0 \\ 0 & x_2y_3 - x_3y_2 & 0 & x_3y_1 - x_1y_3 & 0 & x_1y_2 - x_2y_1 \\ 0 & y_2 - y_3 & 0 & y_3 - y_1 & 0 & y_1 - y_2 \\ 0 & x_3 - x_2 & 0 & x_1 - x_3 & 0 & x_2 - x_1 \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{Bmatrix} \tag{2.16}$$

Donde A es el área del triángulo el cual puede expresarse así:

$$A = \frac{1}{2}[x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)] \tag{2.17}$$

Las ecuaciones de forma se obtienen por la combinación de las ecuaciones 2.15 y 2.16

$$\begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} (x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y \\ (x_3y_1 - x_1y_3) + (y_3 - y_2)x + (x_1 - x_3)y \\ (x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y \end{bmatrix} \tag{2.18}$$

El desplazamiento puede ser escrito en término de la función de forma como:

$$\begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{Bmatrix} u(x,y) \\ v(x,y) \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{Bmatrix} \tag{2.19}$$

Las deformaciones son obtenidas de la relación deformación-desplazamiento

$$\varepsilon_x(x,y) = \frac{\partial u}{\partial x} = \frac{\partial}{\partial x} \sum_{i=1}^{3} (N_i(x,y)u_i) \tag{2.20}$$

$$\varepsilon_y(x,y) = \frac{\partial v}{\partial y} = \frac{\partial}{\partial y} \sum_{i=1}^{3} (N_i(x,y)v_i) \tag{2.21}$$

$$\gamma_{xy}(x,y) = \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} = \frac{\partial}{\partial x} \sum_{i=1}^{3} (N_i(x,y)v_i) + \frac{\partial}{\partial y} \sum_{i=1}^{3} (N_i(x,y)u_i) \tag{2.22}$$

Las anteriores ecuaciones se pueden representar en forma matricial como:

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{Bmatrix} \tag{2.23}$$

O

$$\{\varepsilon(x,y)\}_{3x1} = [B(x,y)]_{3x6}\{u_i\}_{6x1} \tag{2.24}$$

Tomando la derivada de la función de forma, ecuación 2.18, con respecto a x y a y se obtiene la matriz de deformación - desplazamiento $[B]$:

$$[B(x,y)] = \frac{1}{2A}\begin{bmatrix} y_2-y_3 & 0 & y_3-y_1 & 0 & y_1-y_2 & 0 \\ 0 & x_3-x_2 & 0 & x_1-x_3 & 0 & x_2-x_1 \\ x_3-x_2 & y_2-y_3 & x_1-x_3 & y_3-y_1 & x_2-x_1 & y_1-y_2 \end{bmatrix} \tag{2.25}$$

La matriz de rigidez del elemento finito, puede obtenerse usando la matriz de deformación - desplazamiento $[B]$ y la matriz de material $[E]$ de la siguiente ecuación:

$$[k]_{6x6} = \int_v [B]^T_{6x3}[E]_{3x3}[B]_{3x6}dv \tag{2.26}$$

Como el espesor de placa se asume constante, la integral del volumen se reduce a una integral de superficie.

$$[k]_{6x6} = t\int_A [B]^T_{6x3}[E]_{3x3}[B]_{3x6}dA \tag{2.27}$$

Donde t = espesor de la placa.

Dado que todos los elementos de la matriz $[B]$ y de la matriz $[E]$ son constantes, estas pueden salir de la integral y reescribirse de la siguiente forma:

$$[k]_{6x6} = t[B]^T_{6x3}[E]_{3x3}[B]_{3x6}\int_A dA \tag{2.28}$$

o

$$[k]_{6x6} = tA[B]^T_{6x3}[E]_{3x3}[B]_{3x6} \tag{2.29}$$

Ya obtenida la matriz de rigidez $[k]$, la cual es necesaria para realizar el cálculo de los desplazamientos del elemento finito ante fuerzas aplicadas en el plano, se debe utilizar la siguiente ecuación para dicho cálculo.

$$\{P\} = [k]\{U\} \tag{2.30}$$

O

$$\{U\} = [k]^{-1}\{P\} \tag{2.31}$$

Donde $\{P\}$ es el vector de Fuerzas externas, $[k]$ es la matriz de rigidez y el vector $\{U\}$ es el vector de desplazamientos.

Posteriormente obtenidos los desplazamientos de la Ecuación 2.31 se procede a implementar este elemento finito en *Fortran 90*, para incorporarlo al Programa Endógeno de Elementos Finitos (PEEF).

Realizando algunas pruebas y comparando los resultados de este elemento finito con los resultados obtenidos de programas de simulación como lo son *ABAQUS CAE*, *LISA* y programas de cálculo matricial como *MATLAB*, para verificar el algoritmo, se pudo obtener en los resultados una muy buena precisión, demostrando así que el elemento finito para la membrana funciona correctamente.

2.1.2 Pruebas y Verificación del Elemento Finito tipo Membrana

A continuación se ilustra 3 pruebas realizadas al elemento finito membrana para comprobar y comparar sus resultados usando diferentes softwares de simulación.

2.1.2.1 Prueba 1: Simulación de un elemento triangular

En la Figura 2.2 se ilustra el elemento finito a comprobar, el cual consta de un triángulo rectángulo sometido a una fuerza y a ciertas restricciones.

Especificaciones: Espesor = 1cm, E=215000, v=0.2

Los resultados obtenidos se muestran en la Tabla 2.1 y en el apéndice C se detalla el código en Matlab para este Elemento Finito (rigidezMembrana.m):

En la Figuras 2.3 y 2.4 se observan graficamente los resultados obtenido por los sofware *Lisa* y *Abaqus* respectivamente.

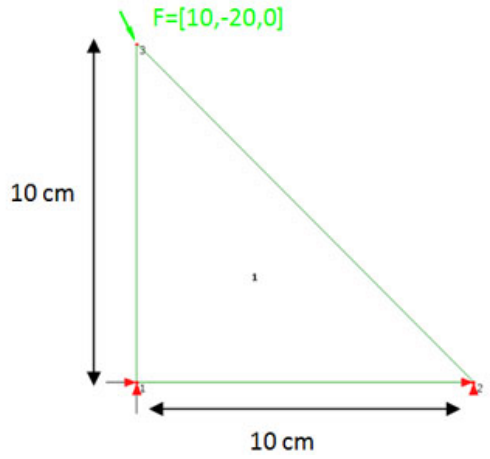

Figura 2.2: Elemento triangular y solicitaciones.

Tabla 2.1: Resultados de la simulación.

Variables	Matlab	Lisa	Abaqus	PEEF
$U_3(cm)$	$2.2326\ 10^{-4}$	$2.232558\ 10^{-4}$	$2.23256\ 10^{-4}$	$2.2326\ 10^{-4}$
$V_3(cm)$	$-1.7860\ 10^{-4}$	$-1.786046\ 10^{-4}$	$-1.78605\ 10^{-4}$	$-1.7860\ 10^{-4}$
$Rx_1(Kg)$	$-6.0\ 10^{0}$	$-6.0\ 10^{0}$	$-6.0\ 10^{0}$	$-6.0000\ 10^{0}$
$Ry_1(Kg)$	$1.0\ 10^{1}$	$1.0\ 10^{1}$	$1.0\ 10^{1}$	$1.00000\ 10^{1}$
$Rx_2(Kg)$	$-4.0\ 10^{0}$	$-4.0\ 10^{0}$	$-4.0\ 10^{0}$	$-4.0000\ 10^{0}$
$Ry_2(Kg)$	$1.0\ 10^{1}$	$1.0\ 10^{1}$	$1.0\ 10^{1}$	$1.0000\ 10^{1}$

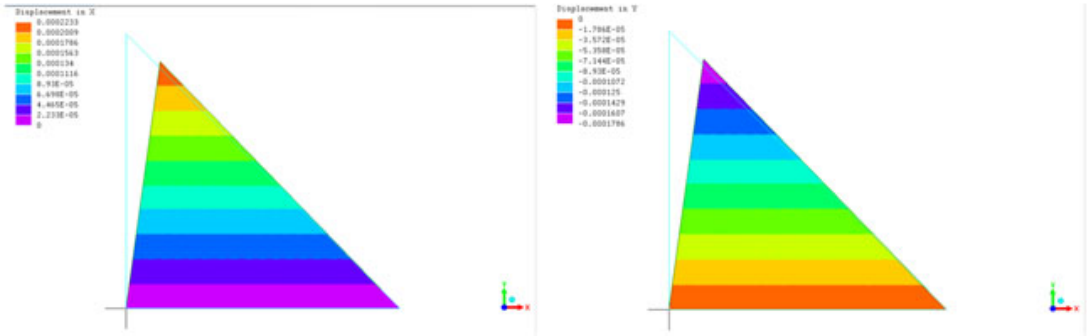

Figura 2.3: Resultados Gráficos de Lisa. *Ux y Uy* para Un Elemento Finito CST

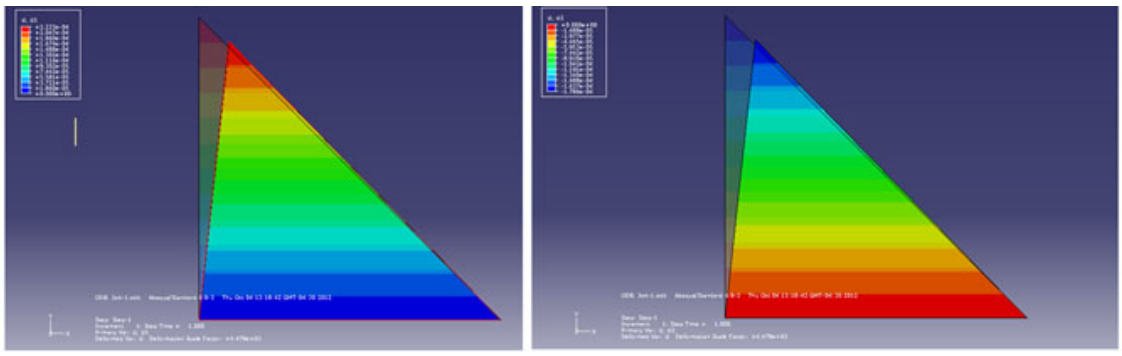

Figura 2.4: Resultados Gráficos de Abaqus CAE. *Ux y Uy* Un Elemento Finito CST

2.1.2.2 Prueba 2: Simulación de dos elementos triangulares

En la Figura 2.5 se ilustra el elemento finito a comprobar, el cual consta de dos triángulos rectángulos sometidos a fuerzas y a ciertas restricciones.

Especificaciones: Espesor = 1cm, E=215000, v=0.2

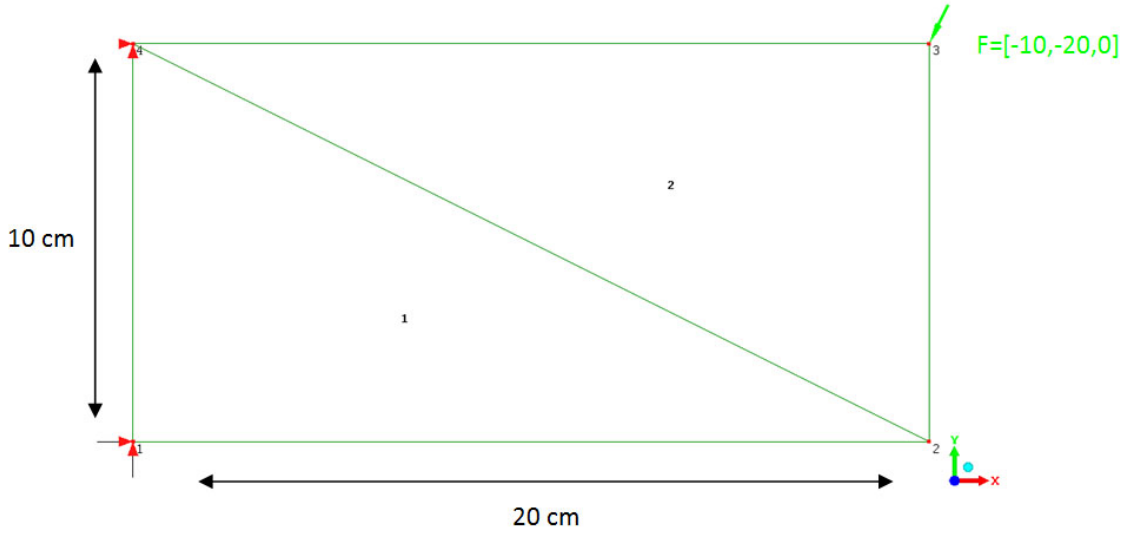

Figura 2.5: Dos elementos triangulares y solicitaciones.

Los resultados obtenidos se muestran en la Tabla 2.2:

En la Figuras 2.6 y 2.7 se observan graficamente los resultados obtenido por los sofware *Lisa* y *Abaqus* respectivamente.

Tabla 2.2: Resultados de la simulación.

Variables	Matlab	Lisa	Abaqus	PEEF
$U_2(cm)$	$-1.94477\ 10^{-4}$	$-1.944774\ 10^{-4}$	$-1.94477\ 10^{-4}$	$-1.9448\ 10^{-4}$
$V_2(cm)$	$-6.49926\ 10^{-4}$	$-6.499264\ 10^{-4}$	$-6.49926\ 10^{-4}$	$-6.4993\ 10^{-4}$
$U_3(cm)$	$4.36144\ 10^{-5}$	$4.361443\ 10^{-5}$	$4.36144\ 10^{-5}$	$4.3614\ 10^{-5}$
$V_3(cm)$	$-7.19281\ 10^{-4}$	$-7.192806\ 10^{-4}$	$-7.19281\ 10^{-4}$	$-7.1928\ 10^{-4}$
$Rx_1(Kg)$	$4.0\ 10^{1}$	$4.0\ 10^{1}$	$4.0\ 10^{1}$	$4.0000\ 10^{1}$
$Ry_1(Kg)$	$1.89111\ 10^{1}$	$1.89111289031225\ 10^{1}$	1.89111	$1.8911\ 10^{1}$
$Rx_4(Kg)$	$-3.0\ 10^{1}$	$-3.0\ 10^{1}$	$-3.0\ 10^{1}$	$-3.0000\ 10^{1}$
$Ry_4(Kg)$	$1.08887\ 10^{0}$	$1.08887109687751\ 10^{0}$	$1.08887\ 10^{0}$	$1.0889\ 10^{0}$

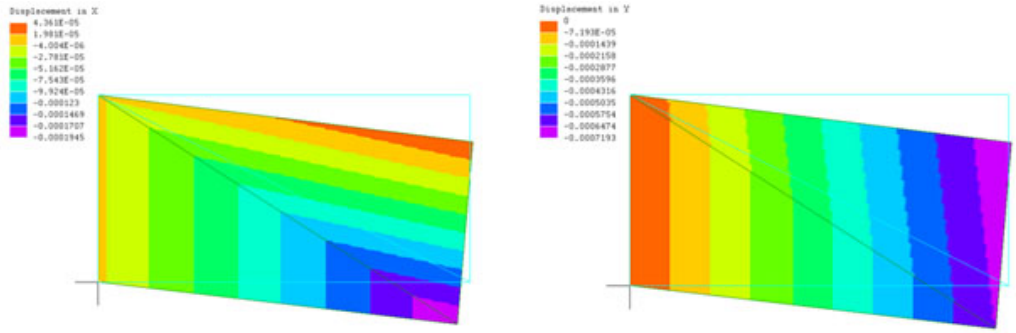

Figura 2.6: Resultados Gráficos de Lisa. *Ux y Uy* para dos Elemento Finito CST

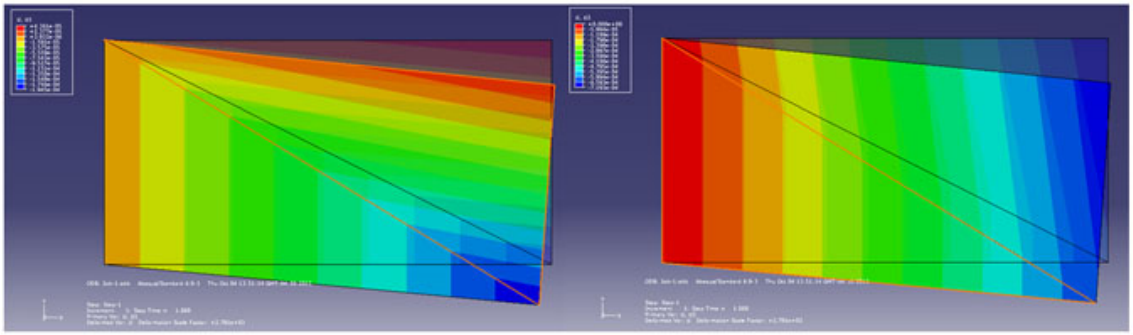

Figura 2.7: Resultados Gráficos de Abaqus. *Ux y Uy* para dos Elemento Finito CST

2.1.2.3 Prueba 3: Simulación de cuatro elementos triangulares

En la Figura 2.8 se ilustra el elemento finito a comprobar, el cual consta de cuatro triángulos conectados de manera que formen una placa rectangular, los cuales son sometidos a fuerzas y a ciertas restricciones.

Especificaciones: Espesor = 1cm, E=215000, v=0.2

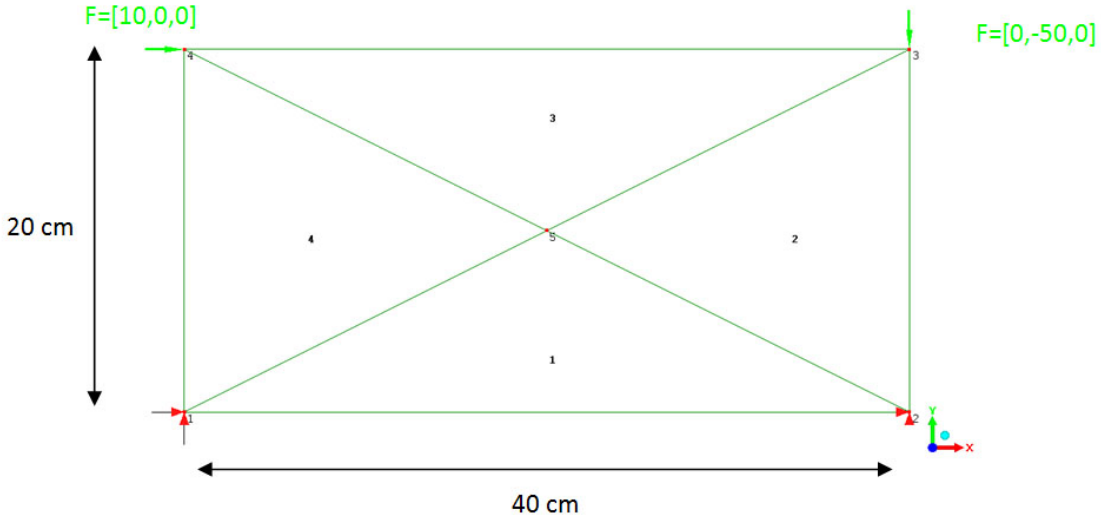

Figura 2.8: Cuatro elementos triangulares y solicitaciones.

Los resultados obtenidos se muestran en la Tabla 2.3:

En la Figuras 2.9 y 2.10 se observan graficamente los resultados obtenido por los sofware *Lisa* y *Abaqus* respectivamente.

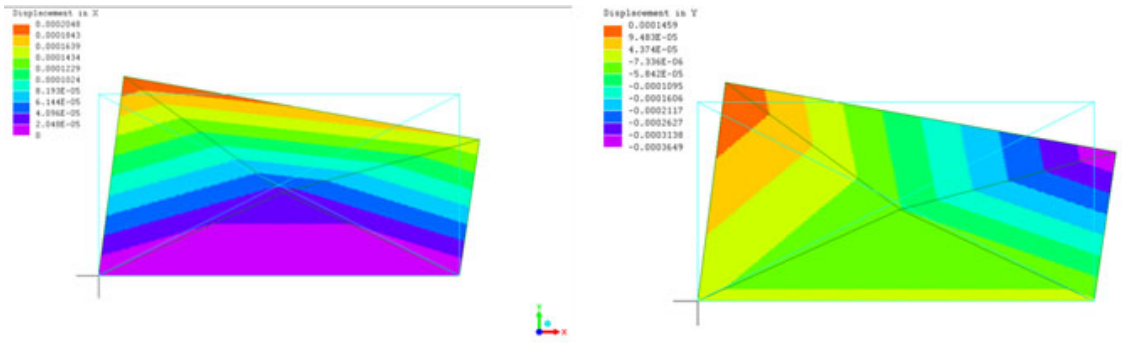

Figura 2.9: Resultados Gráficos de Lisa. *Ux y Uy* para cuatro Elementos Finitos CST

Tabla 2.3: Resultados de la simulación.

Variables	Matlab	Lisa	Abaqus	PEEF
$U_3(cm)$	$1.62217\ 10^{-4}$	$1.622169\ 10^{-4}$	$1.62217\ 10^{-4}$	$1.6222\ 10^{-4}$
$V_3(cm)$	$-3.64903\ 10^{-4}$	$-3.649026\ 10^{-4}$	$-3.64903\ 10^{-4}$	$-3.6490\ 10^{-4}$
$U_4(cm)$	$2.04816\ 10^{-4}$	$2.048156\ 10^{-4}$	$2.04816\ 10^{-4}$	$2.0482\ 10^{-4}$
$V_4(cm)$	$1.45907\ 10^{-4}$	$1.459067\ 10^{-4}$	$1.45907\ 10^{-4}$	$1.4591\ 10^{-4}$
$U_5(cm)$	$3.28186\ 10^{-5}$	$-5.765344\ 10^{-5}$	$-5.76534\ 10^{-5}$	$-5.7653\ 10^{-5}$
$V_5(cm)$	$-7.19281\ 10^{-4}$	$-7.192806\ 10^{-4}$	$-7.19281\ 10^{-4}$	$-7.1928\ 10^{-4}$
$Rx_1(Kg)$	-2.71032	-2.7103209019948	-2.71032	-2.7103
$Ry_1(Kg)$	-5.0	-5.0	-5.0	-5.0
$Rx_2(Kg)$	-7.28968	-7.28967909800521	-7.28968	-7.2897
$Ry_2(Kg)$	55	55	55	55

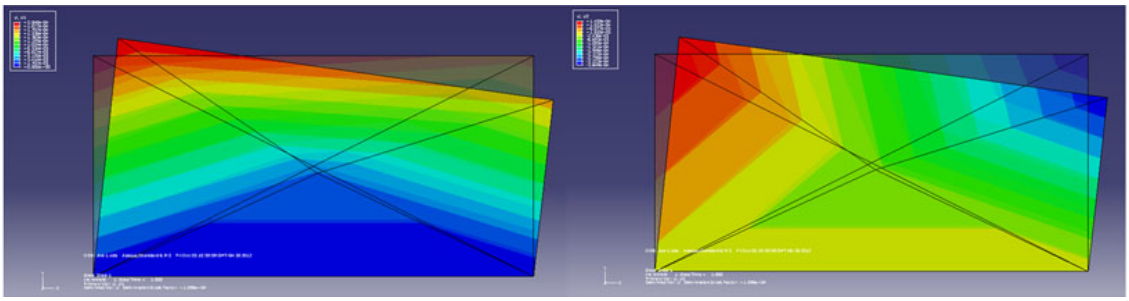

Figura 2.10: Resultados Gráficos de Abaqus. *Ux y Uy* para cuatro Elementos Finitos CST

2.2 Desarrollo del elemento finito Placa Flexión (DKT)

Para representar el elemento finito tipo placa flexión (DKT por sisglas en inglés) se utiliza el elemento triangular discreto de *Kirchhoff*, al cual se hallará su matriz de rigidez para la realización de los cálculos de los desplazamientos nodales.

2.2.1 Flexión de Placas

La flexión de placas planas es similar a la flexión de las vigas; sólo que la placa de flexión es en dos dimensiones, mientras que la flexión de la viga es una dimensión. El comportamiento de las placas depende principalmente del espesor de la misma. Las placas pueden ser:

- Placas delgadas con pequeñas deformaciones.
- Placas delgadas con grandes deformaciones.
- Placas gruesas.

En este estudio se considera placas delgadas con pequeñas deformaciones.

Hay tres supuestos básicos de la teoría de flexión para placas delgadas (Timoshenko, 1959).

1. La mitad de la superficie de la placa permanece sin estirar durante las deformaciones.
2. Puntos, rectas y normales a la mitad de la superficie de la placa antes de doblar permanecen rectos y normal a mediados de la superficie después de la flexión.
3. Los esfuerzos cortantes transversales son pequeños en comparación con las tensiones normales y por lo tanto pueden ser omitidos.

Estos supuestos son conocidos como hipótesis de *Kirchhoff* y son aplicables a la flexión de las placas delgadas con pequeñas deflexiones. Se considera una placa isotrópica de espesor uniforme t con el plano XY como el plano principal. De acuerdo con la teoría de la flexión para una placa delgada, la placa está en la condición de tensión plana y por lo tanto todas las tensiones varían linealmente en el espesor de la placa.

Los momentos en la placa pueden ser representados como:

$$\begin{aligned} M_x &= \int_{-t/2}^{t/2} \sigma_x z dz \\ M_y &= \int_{-t/2}^{t/2} \sigma_y z dz \\ M_{xy} &= \int_{-t/2}^{t/2} \tau_{xy} z dz \end{aligned} \tag{2.32}$$

Donde M_x y M_y son los momentos en la dirección x y la dirección y respectivamente, el M_{xy} es el momento de torsión.

Si w es el desplazamiento transversal de la placa, la relación curvatura - desplazamiento para la palca delgada se pude escribir como: S

$$\begin{aligned} K_x &= -\frac{\partial^2 w}{\partial x^2} \\ K_y &= -\frac{\partial^2 w}{\partial y^2} \\ K_{xy} &= -2\frac{\partial^2 w}{\partial x \partial y} \end{aligned} \tag{2.33}$$

2.2.2 Relaciones básicas para una placa delgada de flexión

Considere una pequeña sección de la placa de longitud dx en la dirección x. Cuando una carga se aplica en la dirección z, el punto o en la mitad de la superficie de la placa y se mueve en dirección z debido a la flexión presente en el plano, en la Figura 2.11, se ilustra mejor este efecto.

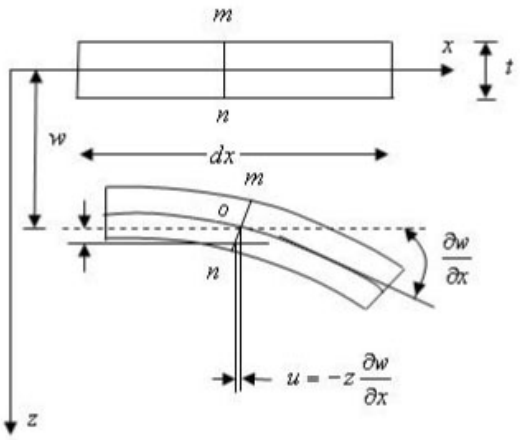

Figura 2.11: Flexión de placa

De acuerdo con la suposición de *Kirchhoff*, una línea que es recta y normal a la mitad de la superficie antes de doblar, se mantiene recta y normal a la superficie media después de doblar.

Los desplazamientos pueden ser escritos como:

$$u = -z\frac{\partial w}{\partial x}; \quad v = -z\frac{\partial w}{\partial y} \tag{2.34}$$

De la Ecuación 2.34, y de la relación de deformación - desplazamiento, las deformaciones correspondientes pueden calcularse como:

$$\begin{aligned} \varepsilon_x &= \frac{\partial u}{\partial x} = -z\frac{\partial^2 w}{\partial x^2} \\ \varepsilon_y &= \frac{\partial v}{\partial y} = -z\frac{\partial^2 w}{\partial y^2} \\ \gamma_{xy} &= \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = -2z\frac{\partial^2 w}{\partial x \partial y} \end{aligned} \tag{2.35}$$

Para el plano de esfuerzos, las relaciones esfuerzo - deformación es la siguiente:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \frac{E}{1-v^2} \begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \tag{2.36}$$

Sustituyendo los valores de esfuerzo de la ecuación 2.46 en la 2.36, resulta:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \frac{E}{1-v^2} \begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix} \begin{Bmatrix} -z\frac{\partial^2 w}{\partial x^2} \\ -z\frac{\partial^2 w}{\partial y^2} \\ -2z\frac{\partial^2 w}{\partial x \partial y} \end{Bmatrix} \tag{2.37}$$

De la Ecuación 2.37 los esfuerzos en la placa se pueden representar como:

$$\begin{aligned} \sigma_x &= -\tfrac{zE}{1-v^2}\left[\frac{\partial^2 w}{\partial x^2} + v\frac{\partial^2 w}{\partial y^2}\right] \\ \sigma_y &= -\tfrac{zE}{1-v^2}\left[v\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right] \\ \gamma_{xy} &= -\tfrac{2zE}{1-v^2}\left(\tfrac{1-v}{2}\right)\left[\frac{\partial^2 w}{\partial x \partial y}\right] \end{aligned} \tag{2.38}$$

Sustituyendo los esfuerzos de la Ecuación 2.38 en la Ecuación 2.32 e integrando sobre el espesor de la placa, se obtiene la siguiente relación para los momentos:

$$M_x = \frac{-Et^3}{12(1-v^2)}\left[\frac{\partial^2 w}{\partial x^2} + v\frac{\partial^2 w}{\partial y^2}\right]$$

$$M_y = \frac{-Et^3}{12(1-v^2)}\left[v\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right] \tag{2.39}$$

$$M_{xy} = \frac{-Et^3}{12(1-v^2)}(1-v)(\frac{\partial^2 w}{\partial x \partial y})$$

Donde $\frac{-Et^3}{12(1-v^2)} = D$ es la rigidez a la flexión de la placa.

La ecuación 2.39 se puede escribir de forma matricial de la siguiente forma:

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \begin{bmatrix} D & v & 0 \\ v & D & 0 \\ 0 & 0 & D(\frac{1-v}{2}) \end{bmatrix} \begin{Bmatrix} -z\frac{\partial^2 w}{\partial x^2} \\ -z\frac{\partial^2 w}{\partial y^2} \\ -2z\frac{\partial^2 w}{\partial x \partial y} \end{Bmatrix} \tag{2.40}$$

De la Ecuación 2.40 y de la Ecuación 2.33, se obtiene la relación entre el momento y la curvatura

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \begin{bmatrix} D & v & 0 \\ v & D & 0 \\ 0 & 0 & D(\frac{1-v}{2}) \end{bmatrix} \begin{Bmatrix} K_x \\ K_y \\ K_{xy} \end{Bmatrix} \tag{2.41}$$

2.2.3 Elemento triangular de flexión basado en la teoría discreta de *Kirchhoff*

Batoz (1980), desarrolló un elemento triangular de flexión (elemento DKT) basado en la teoría discreta de *Kirchhoff*. De acuerdo con los supuestos de *Kirchhoff*, la energía de curvatura presente en el elemento es mucho mayor en comparación con la energía de deformación de corte, y por lo tanto, el término de energía de corte transversal se puede despreciar.

El elemento DKT es una placa triangular ampliamente utilizado por los programas de análisis de elemento finito. En la Figura 2.12 se ilustra el modelo del elemento finito DKT usado.

La energía de curvatura se puede representarse por la siguiente ecuación:

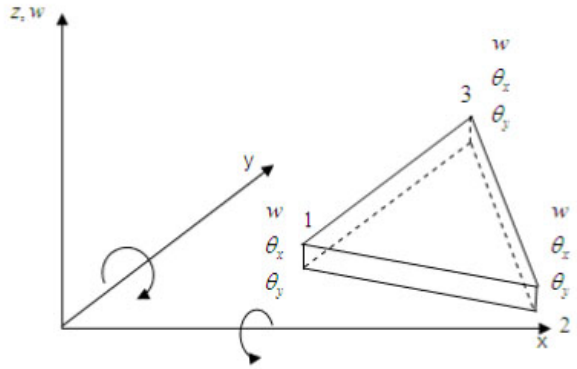

Figura 2.12: Elemento finito DKT

$$U_b = \frac{1}{2}\int_A K^T D_b K dxdy \tag{2.42}$$

Donde

$$D_b = \frac{Et^3}{12(1-v^2)}\begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix} \tag{2.43}$$

y, t = espesor de la placa.

La curvatura es obtenida de la siguiente manera:

$$k = \begin{bmatrix} \beta_{x,x} \\ \beta_{y,y} \\ \beta_{x,y} + \beta_{y,x} \end{bmatrix} \tag{2.44}$$

De acuerdo con las hipótesis formuladas en la teoría de la flexión de placas delgadas con pequeños desplazamientos, los componentes de los vectores U, V y W en cualquier punto puede ser representado como:

$$u = z\beta_x(x,y), \quad v = z\beta_y(x,y), \quad w = w(x,y) \tag{2.45}$$

Donde, w es el desplazamiento transversal, β_x y β_y son las rotaciones en dirección

normal al plano XZ y al plano YZ respectivamente. Se debe tener algunas consideraciones para correlacionar la rotación normal de la superficie media de la placa y el desplazamiento w:

1. El elemento triangular debe tener sólo nueve grados de libertad; es decir, el desplazamiento transversal w y las rotaciones θ_x y θ_y en cada nodo del elemento.

2. De acuerdo a la teoría de *Kirchhoff*, las rotaciones se definen como:

$$\theta_x = \frac{\partial_w}{\partial_x}, \quad \theta_y = \frac{\partial_w}{\partial_y} \tag{2.46}$$

3. La teoría de Kirchhoff puede ser aplicada a cualquier punto discreto en el elemento

4. La compatibilidad de las rotaciones θ_x y θ_y no se puede perder

Además se debe tomar en cuenta los siguientes supuestos,

1. La relación entre las rotaciones en los seis puntos nodales incluyendo superficie media y las funciones de forma en cada seis nodos está en forma de cuadrática

$$\beta_x = \sum_{i=1}^{6} N_i \beta_{xi}, \quad \beta_y = \sum_{i=1}^{6} N_i \beta_{yi} \tag{2.47}$$

   Donde β_{xi} y β_{yi} son las rotaciones en cada uno de los nodos. Como se muestra en la Figura 2.13

   Para $i = 1$ hasta 6, N_i, son las funciones de forma para el elemento DKT, en coordenadas de área ξ y η, y puede expresarse como:

$$\begin{aligned} N_1 &= 2(1-\xi-\eta)(\tfrac{1}{2}-\xi-\eta) \\ N_2 &= \xi(2\xi-1) \\ N_3 &= \eta(2\eta-1) \\ N_4 &= 4\xi\eta \\ N_5 &= 4\eta(1-\xi-\eta) \\ N_6 &= 4\xi(1-\xi-\eta) \end{aligned} \tag{2.48}$$

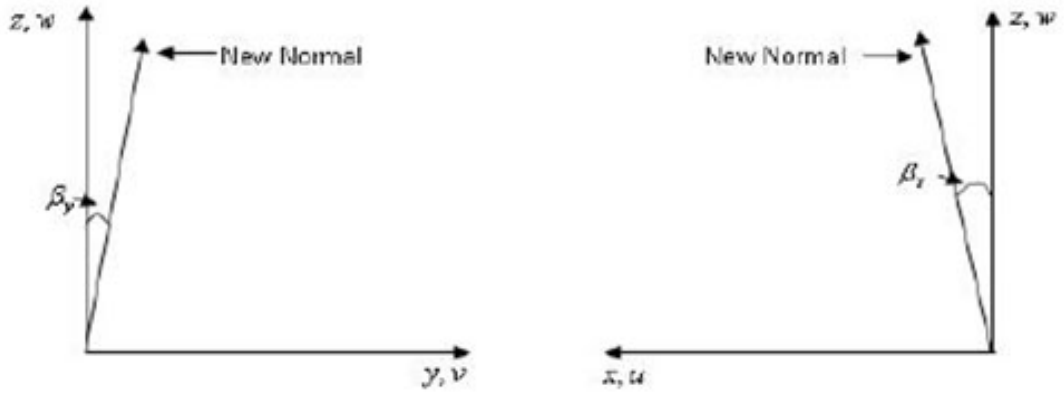

Figura 2.13: Direcciones positiva de β_x y β_y

2. Se aplica la hipótesis de *Kirchhoff* para eliminar las deformación del corte transversal dada a las siguientes ecuaciones.

   En los nodos de esquina,

$$\gamma = \begin{bmatrix} \beta_x + w_x \\ \beta_y + w_y \end{bmatrix} = 0 \tag{2.49}$$

   Donde, γ = deformación del corte transversal.

   En el nodo del medio,

$$\beta_{xk} + w_{xk} = 0k = 4, 5, 6 \tag{2.50}$$

   Donde k es el número del nodo.

3. La variación del desplazamiento transversal está representada por la siguiente expresión:

$$w_{sk} = \frac{-3}{2l_{ij}} w_i - \frac{1}{4} w_{si} + \frac{3}{2l_{ij}} w_j + \frac{1}{4} w_{sj} \tag{2.51}$$

   Donde k es nodo intermedio del lado ij del triángulo y lij representa la longitud del lado ij del triangulo.

4. La variación en la rotación a lo largo del lado del triángulo es representada por la siguiente ecuación lineal:

$$\beta_{nk} = \frac{1}{2}(\beta_{ni} + \beta_{nj}) \tag{2.52}$$

Donde $k = 4, 5, 6$ representan los nodos intermedio de los lados $2-3, 3-1$ y $2-1$ respectivamente.

Como consecuencia de las anteriores cuatro supuestos, la condición de la deformación transversal de corte a lo largo de los lados del triángulo, $\gamma_s = \beta_s + w_s = 0$ es satisfecha.

Los desplazamientos en cada uno de los nodos puede escribirse como:

$$U^T = \{ \; w_1 \quad \theta_{x1} \quad \theta_{y1} \quad w_1 \quad \theta_{x1} \quad \theta_{y1} \quad w_1 \quad \theta_{x1} \quad \theta_{y1} \; \} \tag{2.53}$$

La relación entre los desplazamientos nodales y β_x y β_y se obtiene por:

$$\begin{aligned} \beta_x &= H_x^T(\xi, \eta)U \\ \beta_y &= H_y^T(\xi, \eta)U \end{aligned} \tag{2.54}$$

Donde Hx y Hy son los componentes del vector de las funciones de forma.

$$H_x(\xi, \eta) = \begin{bmatrix} 1.5(a_6N_6 - a_5N_5) \\ ((b_5N_5 + a_6N_6)) \\ N_1 - c_5N_5 - c_6N_6 \\ 1.5(a_4N_4 - a_6N_6) \\ (b_6N_6 + a_4N_4) \\ N_1 - c_6N_6 - c_4N_4 \\ 1.5(a_5N_5 - a_4N_4) \\ (b_4N_4 + b_5N_5) \\ N_3 - c_4N_4 - c_5N_5 \end{bmatrix} \tag{2.55}$$

$$H_y(\xi,\eta) = \begin{bmatrix} 1.5(d_6N_6 - d_5N_5) \\ -N_1 + e_5N_5 + e_6N_6 \\ -b_5N_5 - b_6N_6 \\ 1.5(d_4N_4 - d_6N_6) \\ -N_2 + e_6N_6 + e_4N_4 \\ -b_6N_6 - b_4N_4 \\ 1.5(d_5N_5 - d_4N_4) \\ -N_3 + e_4N_4 + e_5N_5 \\ -b_4N_4 - b_5N_5 \end{bmatrix} \tag{2.56}$$

Donde

$$\begin{aligned} a_k &= -\tfrac{x_{ij}}{l_{ij}^2} \\ e_k &= \tfrac{\frac{1}{4}y_{ij}^2 - \frac{1}{2}x_{ij}^2}{l_{ij}^2} \\ x_{ij} &= x_i - x_j \\ y_{ij} &= y_i - y_j \\ l_{ij}^2 &= (x_{ij}^2 + y_{ij}^2) \end{aligned} \tag{2.57}$$

y $k = 4, 5, 6$ para los lados $ij = 23, 31, 12$ respectivamente.

La matriz de deformación - desplazamiento del elemento DKT puede representarse de la siguiente forma:

$$B(\xi,\eta) = \frac{1}{2A}\begin{bmatrix} y_{31}H_{x,\xi}^T + y_1 2H_{x,\eta}^T \\ -x_{31}H_{y,\xi}^T - x_{12}H_{y,\eta}^T \\ -x_{31}H_{x,\xi}^T - x_{12}H_{x,\eta}^T + y_{31}H_{y,\xi}^T + y_{12}H_{y,\eta}^T \end{bmatrix} \tag{2.58}$$

Donde $2A = x_{31}y_{31} - x_{12}y_{31}$

Las derivadas con respecto a ξ y η del componente del vector de las funciones de forma puede representarse de la siguiente manera:

La derivada de la componente del vector con respecto a ξ es:

$$H_{x,\xi} = \begin{bmatrix} P_6(1-2\xi) + (P_5 - P_6)\eta \\ q_6(1-2\xi) - (q_5 + q_6)\eta \\ -4 + 6(\xi + \eta) + r_6(1-2\xi) - \eta(r_5 + r_6) \\ -P_6(1-2\xi) + \eta(P_4 + P_6) \\ q_5(1-2\xi) - \eta(q_6 - q_4) \\ -2 + 6\xi + r_6(1-2\xi) + \eta(r_4 - r_6) \\ -\eta(P_5 + P_4) \\ \eta(q_4 - q_5) \\ -\eta(r_5 - r_4) \end{bmatrix} \tag{2.59}$$

$$H_{y,\xi} = \begin{bmatrix} t_6(1-2\xi) + (t_5 - t_6)\eta \\ 1 + r_6(1-2\xi) - (r_5 + r_6)\eta \\ -q_6(1-2\xi) - \eta(q_5 - q_6) \\ -t_6(1-2\xi) + \eta(t_4 + t_6) \\ -1 + r_6(1-2\xi) - \eta(r_4 - r_6) \\ q_6(1-2\xi) + \eta(q_4 - q_6) \\ -\eta(t_4 + t_5) \\ \eta(r_4 - r_5) \\ -\eta(q_4 - q_5) \end{bmatrix} \tag{2.60}$$

La derivada de la componente del vector con respecto a η son:

$$H_{x,\eta} = \begin{bmatrix} -P_5(1-2\eta) + (P_6 - P_5)\xi \\ q_5(1-2\eta) - (q_5 + q_6)\xi \\ -4 + 6(\eta + \xi) + r_5(1-2\eta) - \xi(r_5 + r_6) \\ \xi(P_4 + P_6) \\ \xi(q_4 - q_6) \\ -\xi(r_6 - r_4) \\ P_5(1-2\eta) - \xi(P_4 + P_5) \\ q_5(1-2\eta) + \xi(q_4 - q_5) \\ -2 + 6\eta + r_5(1-2\eta) + \xi(r_4 - r_5) \end{bmatrix} \tag{2.61}$$

$$H_{y,\eta} = \begin{bmatrix} -t_5(1-2\eta) - (t_6 - t_5)\xi \\ 1 + r_5(1-2\eta) - \xi(r_5 + r_6) \\ -q_5(1-2\eta) + \xi(q_5 + q_6) \\ \xi(t_4 + t_6) \\ \xi(r_4 - r_6) \\ -\xi(q_4 - q_6) \\ t_5(1-2\eta) - \xi(t_4 + t_5) \\ 1 - r_5(1-2\eta) + \xi(r_4 - r_5) \\ -q_5(1-2\eta) + \xi(q_4 - q_5) \end{bmatrix} \tag{2.62}$$

Donde

$$\begin{aligned} P_k &= -6\frac{x_{ij}}{l_{ij}^2} \\ q_k &= \frac{3x_{ij}y_{ij}}{l_{ij}^2} \\ r_k &= \frac{3y_{ij}}{l_{ij}^2} \\ t_k &= -6\frac{y_{ij}}{l_{ij}^2} \end{aligned} \tag{2.63}$$

Tabla 2.4: Coordenadas y funciones de peso para la cuadratura de *Gauss*

Punto de Integración	Coordenada	Función de Peso
1	$(\frac{1}{2}, 0)$	$\frac{1}{3}$
2	$(\frac{1}{2}, \frac{1}{2})$	$\frac{1}{3}$
3	$(0, \frac{1}{2})$	$\frac{1}{3}$

Y $k = 4, 5, 6$ para $ij = 23, 31, 12$ respectivamente.

La matriz de deformación - desplazamiento puede ser calculada usando la ecuación 2.58 hasta 2.62. Sustituyendo la matriz de deformación - desplazamiento en la ecuación 2.64, la matriz de rigidez para el elemento DKT se obtiene de la siguiente manera:

$$K_{DKT} = 2A \int_0^1 \int_0^{1-\eta} B^T D_b B d_\xi d_\eta \tag{2.64}$$

Recordando que la Matriz D_b es la matriz de material.

Se asume que el elemento tiene espesor constante. La matriz de rigidez para el elemento DKT se puede obtener usando un esquema de la cuadratura de *Gauss*. Los tres puntos de integración numérica se encuentran en los puntos medios de los lados del triángulo. Puesto que la ecuación de la matriz de rigidez tiene términos cuadráticos es una condición suficiente para aplicar el esquema de tres puntos de *Gauss*. Las coordenadas y las funciones de peso para el esquema de los tres puntos se dan en la Tabla 2.4.

La matriz de rigidez usando la cuadratura de *Gauss* se obtiene de:

$$[k]_{9x9} = 2A \sum_{j=1}^{3} \sum_{i=1}^{3} w_j w_i [B(\xi_j, \eta_j]^T_{9x3} [D_b]_{3x3} [B(\xi_i, \eta_i)]_{3x9} d\xi d\eta \tag{2.65}$$

Ya obtenida la matriz de rigidez de este elemento, la cual se es necesaria para realizar el cálculo de los desplazamientos del elemento ante cargas aplicadas en el plano, se procede a aplicar las ecuaciones 2.30 y 2.31, aplicadas para el elemento finito CST.

Donde P es el vector de Fuerzas externas, $[k]$ es la matriz de rigidez del elemento y el vector U es el vector de desplazamientos.

Posteriormente luego de obtenida la ecuación 2.31 se procede a implementar este elemento finito en Fortran 90, para incorporarlo seguidamente al Programa Endógeno de Elementos Finitos (PEEF).

Realizando algunas pruebas y comparando el buen funcionamiento de este elemento finito con programas de simulación como lo son ABAQUS CAE y programas de cálculo como MATLAB, para corroborar el algoritmo, se pudo obtener en los resultados una muy buena precisión, demostrando así que el elemento finito para la flexión de la placa representa correctamente el fenómeno.

2.2.4 Pruebas y Verificación del Elemento Finito placa flexión

Se presenta los resultados obtenidos de el modelado de diferentes triángulos para la comprobación del elemento finito programado "DKT" basado en la teoría de placas de *Kirchhoff*.

2.2.4.1 Prueba 1: Simulación de un elemento triangular

En la Figura 2.14 se ilustra el elemento finito a comprobar, el cual consta de un triángulo rectángulo sometido a una fuerza y a ciertas restricciones.

Especificaciones: Espesor = 1cm, E=215000, v=0.2

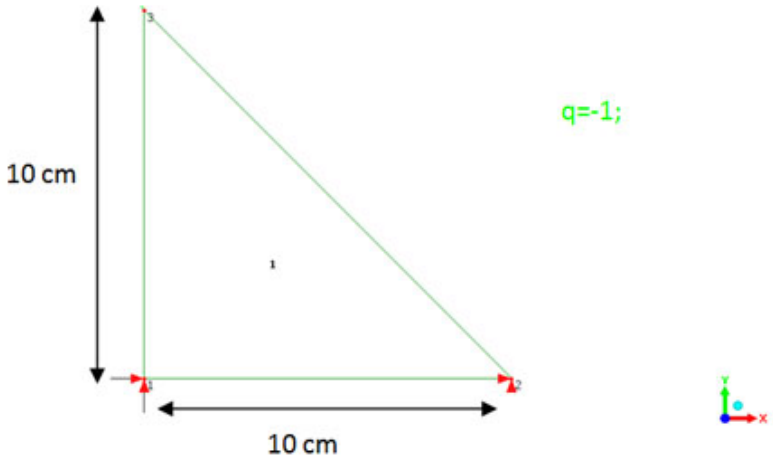

Figura 2.14: Elemento triangular y solicitaciones.

Tabla 2.5: Resultados de la simulación.

Variables	Matlab	Abaqus	PEEF
$Fw_1(Kg)$	33.3477	33.3477	33.348
$Mx_1(Kg.cm)$	121.194	121.194	121.19
$My_1(Kg.cm)$	-23.214	-23.214	-23.214
$Fw_2(Kg)$	16.6523	16.6523	16.652
$Mx_2(Kg.cm)$	45.4731	45.4731	45.473
$My_2(Kg.cm)$	23.07	23.07	23.070
$W_3(cm)$	-0.0591565	-0.0591565	-0.059157
$\theta x_3(cm)$	-0.0095085	-0.0095085	-0.0095085
$\theta y_3(cm)$	-0.00376848	-0.00376848	-0.0037685

Los resultados obtenidos se muestran en la Tabla 2.5 y en el apéndice C se detalla el código en Matlab para este Elemento Finito (rigidDKT.m):

En la Figura 2.15 se observan graficamente los resultados obtenido por el sofware *Abaqus* en el cual se aplicó el elemento finito ***STRI3****: A3 -node triangular facet thin shell* para la simulación.

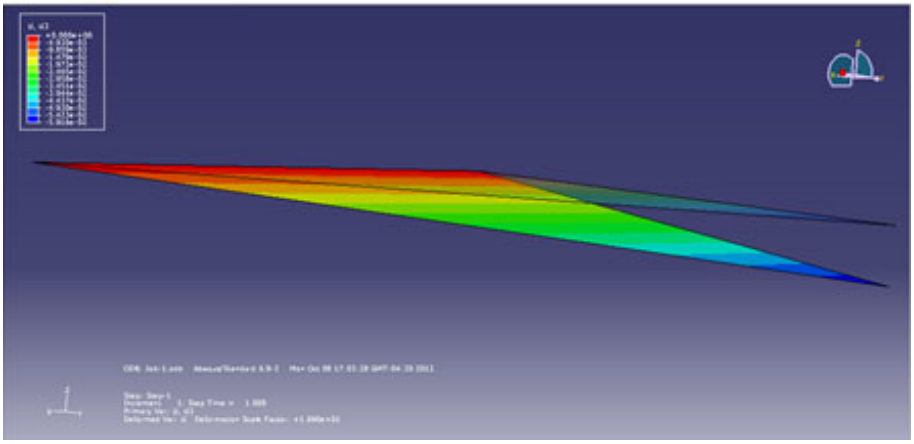

Figura 2.15: Resultados gráficos de *Abaqus*. Un Elemento Finito DKT

2.2.4.2 Prueba 2: Simulación de dos elementos triangulares

En la Figura 2.16 se ilustra el elemento finito a comprobar, el cual consta de dos triángulos rectángulos sometidos a fuerzas y a ciertas restricciones.

Especificaciones: Espesor = 1cm, E=215000, v=0.2

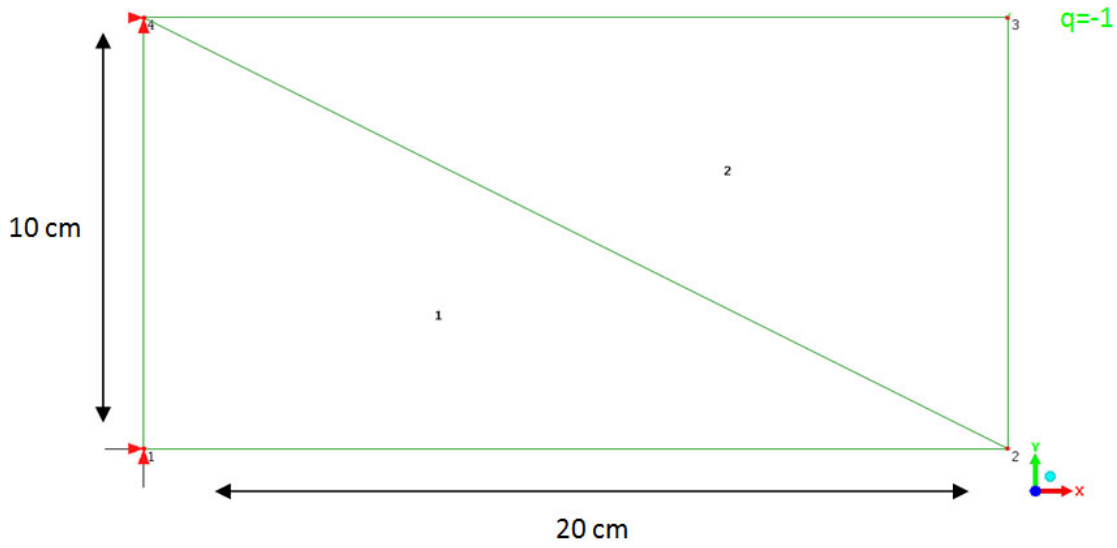

Figura 2.16: Dos elementos triangulares y solicitaciones.

Los resultados obtenidos se muestran en la Tabla 2.6:

En la Figura 2.17 se observa graficamente los resultados obtenido por el sofware *Abaqus* en el cual se aplicó el elemento finito ***STRI3****: A3 -node triangular facet thin shell* para la simulación.

Tabla 2.6: Resultados de la simulación.

Variables	Abaqus	PEEF
$Fw_1(Kg)$	89.0241	89.024
$Mx_1(Kg.cm)$	86.6933	86.693
$My_1(Kg.cm)$	-897.583	-897.58
$W_2(cm)$	-1.5219	-1.5219
$\theta x_2(cm)$	0.00825282	0.0082528
$\theta y_2(cm)$	0.122078	0.12208
$W_3(cm)$	-1.37597	-1.3760
$\theta x_3(cm)$	0.0226767	0.022677
$\theta y_3(cm)$	0.0980174	0.098017
$Fw_4(Kg)$	110.976	110.98
$Mx_4(Kg.cm)$	-196.452	196.45
$My_4(Kg.cm)$	-1102.42	-1102.4

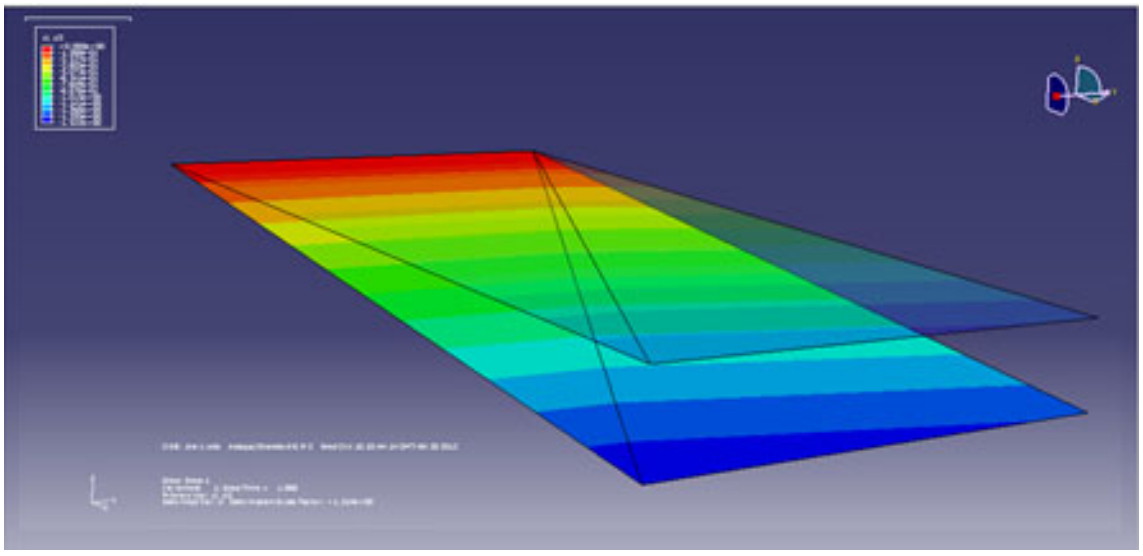

Figura 2.17: Resultados Gráficos de Abaqus. Dos Elementos Finitos DKT

2.2.4.3 Prueba 3: Simulación de cuatro elementos triangulares

En la Figura 2.18 se ilustra el elemento finito a comprobar, el cual consta de cuatro triángulos conectados de manera que formen una placa rectangular, los cuales son

sometidos a fuerzas y a ciertas restricciones.

Especificaciones: Espesor = 1cm, E=215000, v=0.2

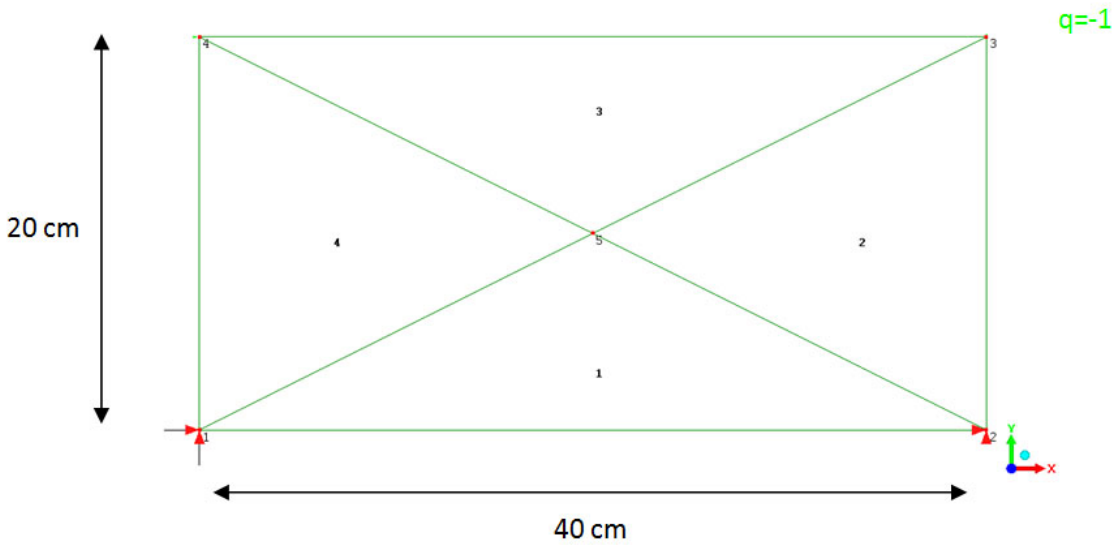

Figura 2.18: Cuatro elementos triangulares y solicitaciones.

Los resultados obtenidos se muestran en la Tabla 2.7:

En la Figura 2.19 se observa graficamente los resultados obtenido por el sofware *Abaqus* en el cual se aplicó el elemento finito ***STRI3****: A3 -node triangular facet thin shell* para la simulación.

Tabla 2.7: Resultados de la simulación.

Variables	Abaqus	PEEF
$Fw_1(Kg)$	400	400.00
$Mx_1(Kg.cm)$	4000	4000.0
$My_1(Kg.cm)$	-1551.04	-1551.0
$Fw_2(Kg)$	400	400.00
$Mx_2(Kg.cm)$	4000	4000.0
$My_2(Kg.cm)$	1551.04	1551.0
$W_3(cm)$	-1.48695	-1.4869
$\theta x_3(cm)$	-0.09593	-0.090593
$\theta y_3(cm)$	-0.0129072	-0.012907
$W_4(cm)$	-1.48695	-1.4869
$\theta x_4(cm)$	-0.090593	-0.090593
$\theta y_4(cm)$	0.0129072	0.012907
$W_5(cm)$	-0.774822	-0.77482
$\theta x_5(cm)$	-0.0679873	-0.067987
$\theta y_5(cm)$	$1.084\ 10^{-14}$	$-0.25841\ 10^{-16}$

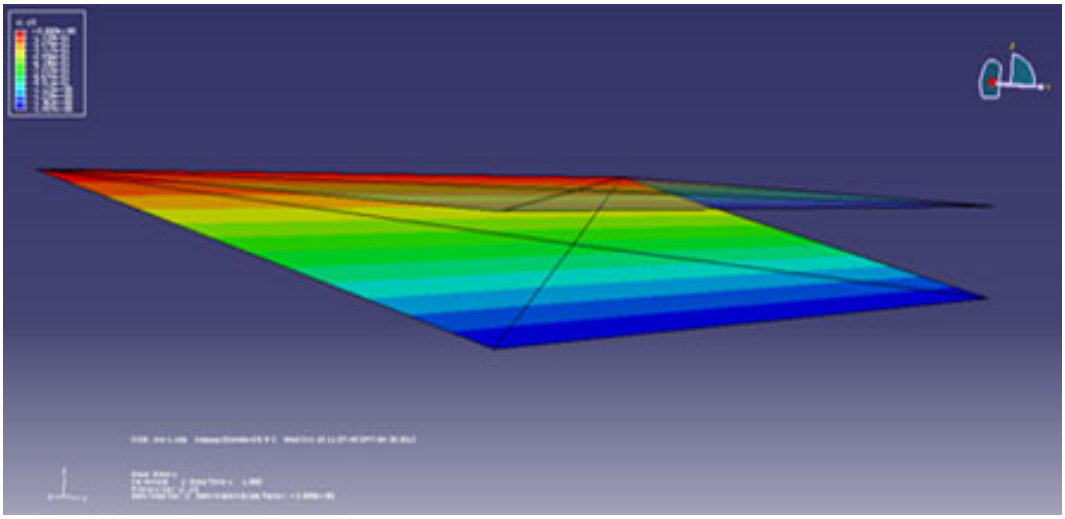

Figura 2.19: Resultados Gráficos de Abaqus. Cuatro Elementos Finitos DKT

2.3 Desarrollo del elemento finito Placa General (CSTDKT)

Los elementos de tipo placa general, son eficientes para modelar el comportamiento de la curvatura en las estructuras. Existen cuatro tipos de elementos de placa general estos son:

- Elemento Plano de Placa
- Elemento Curvo de Placa
- Elementos de Revolución
- Elemento Solido Placa de *Midlin*

El elemento plano de placa puede ser clasificado de acuerdo al espesor de la placa y la curvatura de la superficie. Dependiendo de este espesor el elemento puede separarse en placas delgadas y placas gruesas. Las placas delgadas son basadas en la teoría discreta de *Kirchhoff*, en la cual se desprecia la deformación del corte transversal. Las placas gruesas se basan en la teoría de *Midlin*, en la cual se toma en cuenta la deformación del corte transversal.

Los elementos de tipo placa pueden clasificarse, además, de acuerdo a la curvatura: en placas profundas y placas superficiales. Las placas superficiales son basadas en la teoría clásica de placas y pueden desarrollarse como combinación de los elementos de tipo membrana y tipo placa de flexión en su ecuación de energía.

El elemento de placa general es desarrollado superponiendo la rigidez de la membrana con la rigidez de la placa de flexión. Las fuerzas en los elementos membrana y placa flexión son totalmente independientes entre sí y por lo tanto no hay acoplamiento en el elemento combinado membrana - flexión. Esta es la mayor desventaja de los elementos de placa general.

El desarrollo del elemento placa general, por medio de la teoría clásica de placas es más engorroso y complejo, y muchas aproximaciones son requeridas para simplificar la solución. Los elemento de placa general son más sencillos de formular usando teorías previamente disponibles de membrana y de placas de flexión.

2.3.1 Elemento de Placa General

El elemento placa general está sometido a dos fuerzas en el plano y las fuerzas de flexión, por lo tanto el desarrollo de este elemento finito debe incluir una consideración de estas dos acciones. Un enfoque para el desarrollo del elemento placa es incluir la membrana y las propiedades de flexión de la placa mediante la combinación del elemento tipo membrana (CST) y el elemento placa de flexión (DKT).

Se consideraron dos tipos de elementos: (1) el elemento triangular de membrana (CST) desarrollado al principio del capítulo, el cual representa la parte de la membrana del elemento, y (2) el elemento DKT descrito anteriormente, el cual representa la parte de flexión del elemento placa. Se considera que el elemento está en el plano XY, el ensamblaje del elemento triangular placa general se muestra en la Figura 2.20

2.3.2 Desarrollo de la Matriz de rigidez para el elemento Placa General

En esta sección se discute la formulación general de la matriz de rigidez para los elementos de placa general. El desarrollo de la matriz de rigidez del elemento de

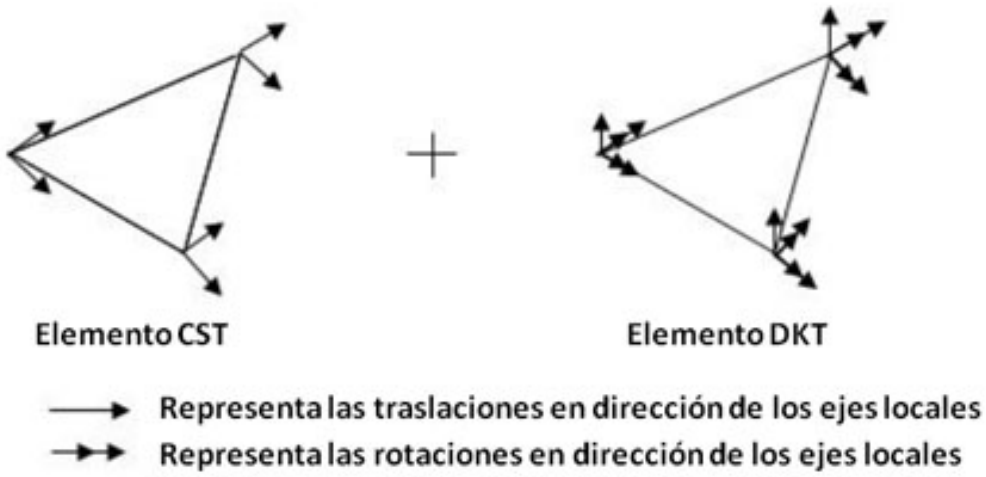

Figura 2.20: Combinación de los elementos CST y DKT.

membrana triangular se realizó al comienzo de este capítulo. El desarrollo de la matriz de rigidez para el elemento placa de flexión triangular se presentó anteriormente.

El elemento placa general triangular posee 6 grados de libertad por cada nodo, los desplazamientos del nodo son los siguientes:

$$\{U_i\} = \{ \; u_i \quad v_i \quad w_i \quad \theta_{xi} \quad \theta_{yi} \quad \theta_{zi} \; \} \tag{2.66}$$

Para $i = 1, 2, 3$

La matriz de rigidez del elemento membrana por cada nodo presenta dos (2) grados de libertad, su tamaño es de $2x2$ y se representa $[k_m]_{2x2}$. La matriz de rigidez del elemento placa flexión por cada nodo contiene tres (3) grados de libertad de tamaño $3x3$ y se representa $[k_f]_{3x3}$. La matriz de rigidez del elemento placa general presenta seis (6) grados de libertad por cada nodo y su tamaño es de $6x6$ representándose $[k_p]_{6x6}$. El ensamblaje de la matriz de rigidez del elemento membrana junto con la matriz de rigidez del elemento placa flexión, resultará en un valor cero (0) para la rotación del eje Z, θ_z, ya que esta rotación no se considera ni en el elemento membrana ni en el elemento placa flexión. La Figura 2.21 Represa los desplazamientos y las rotaciones para el elemento placa general.

Debido a la singularidad de esta rotación en las placas, se asume un valor de cero (0) y por lo tanto no afecta de manera significativa para el modelo. Por lo tanto la

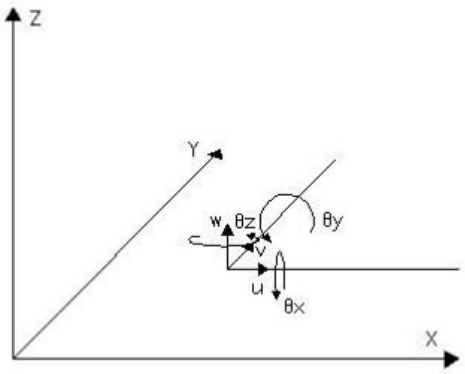

Figura 2.21: Grados de libertad para el elemento placa general.

Matriz de rigidez para el elemento placa general resulta de la siguiente manera:

$$[k_p]_i = \begin{bmatrix} [k_m]_{2x2} & [0]_{2x3} & 0 \\ [0]_{3x2} & [k_f]_{3x3} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.67}$$

Donde

$[k_p]_i$ = es la matriz de rigidez en cada nodo del elemento placa general.

$[k_m]_{2x2}$ = es la matriz de rigidez en cada nodo del elemento membrana.

$[k_f]_{3x3}$ = es la matriz de rigidez en cada nodo del elemento placa flexión.

Ya obtenida la matriz de rigidez de este elemento, la cual se necesita para realizar el cálculo de los desplazamientos del elemento finito ante cargas aplicadas en el plano.

Se procede a utilizar la ecuación $\{P\} = [k]\{U\}$ O $\{U\} = [k]^{-1}\{P\}$ donde el vector $\{P\}$ es el vector de Fuerzas externas, $[k]$ es la matriz de rigidez y el vector $\{U\}$ es el vector de desplazamientos.

Posteriormente obtenido el vector $\{U\}$ se procedió a implementar este elemento finito en *Fortran 90*, para incorporarlo al Programa Endógeno de Elementos Finitos (PEEF).

Realizando algunas pruebas y comparando el buen funcionamiento de este elemento finito con programas de simulación como lo es ABAQUS CAE, como se observa en las tablas, los resultados arrojaron buena precisión, demostrando así que el elemento finito para la flexión de la placa resultó ser tan bueno como el Elemento usado por ABAQUS

CAE.

2.3.3 Pruebas y Verificación del Elemento Finito Placa General

Se presenta los resultados obtenidos de el modelado de diferentes triángulos para la comprobación del elemento finito programado CSTDKT basado en la teoría de placas de *Kirchhoff*.

2.3.3.1 Prueba 1: Simulación de un elemento triangular

En la Figura 2.22 se ilustra el elemento finito a comprobar, el cual consta de un triángulo rectángulo sometido a una fuerza y a ciertas restricciones.

Especificaciones: Espesor = 1cm, E=215000, v=0.2

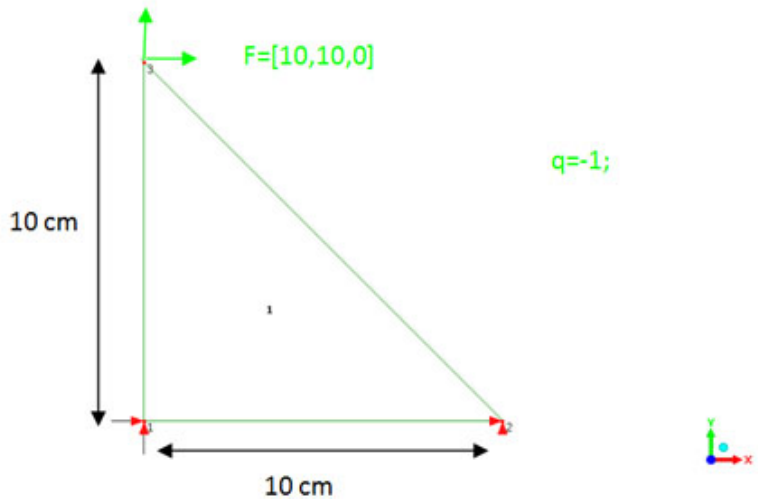

Figura 2.22: Elemento triangular y solicitaciones.

Los resultados obtenidos se muestran en la Tabla 2.8 y en el apéndice C se detalla el código en MatLab para este Elemento Finito (rigidezCombinado.m) :

En la Figuras 2.23 y 2.24 se observan graficamente los resultados obtenido por el sofware *Abaqus* en el cual se aplicó el elemento finito ***STRI3****: A3 -node triangular facet thin shell* para la simulación.

Tabla 2.8: Resultados de la simulación.

Variables	Matlab	Abaqus	PEEF
$Fu_1(Kg)$	-12	-11.9997	-12.000
$Fv_1(Kg)$	-20	-19.9972	-20.000
$Fw_1(Kg)$	33.3477	33.3477	33.348
$Mx_1(Kg.cm)$	121.194	121.194	121.19
$My_1(Kg.cm)$	-23.214	-23.214	-23.214
$Fu_2(Kg)$	2	1.99965	2.0000
$Fv_2(Kg)$	10	9.99717	10.000
$Fw_2(Kg)$	16.6523	16.6523	16.652
$Mx_2(Kg.cm)$	45.4731	45.4731	45.473
$My_2(Kg.cm)$	23.07	23.07	23.070
$U_3(cm)$	0.000223256	0.000223231	0.00022326
$V_3(cm)$	$8.93023\ 10^{-5}$	$8.92997\ 10^{-5}$	$8.9302\ 10^{-5}$
$W_3(cm)$	-0.0591565	-0.0591565	-0.059157
$\theta x_3(cm)$	-0.0095085	-0.0095085	-0.0095085
$\theta y_3(cm)$	-0.00376848	-0.00376848	-0.0037685

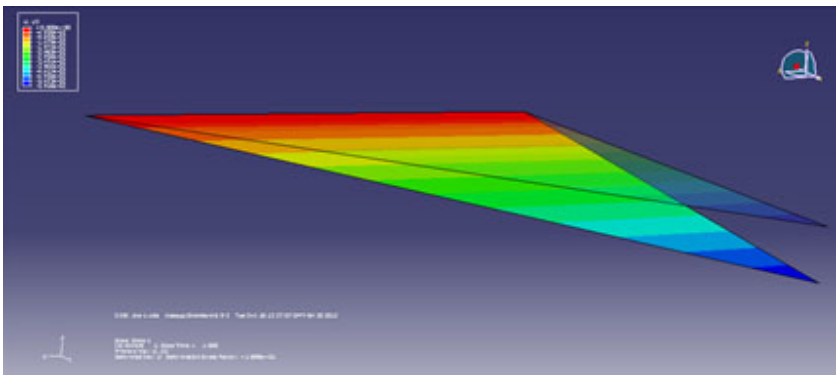

Figura 2.23: Resultados gráficos de *Abaqus* Flexión. Un Elemento Finito CSTDKT

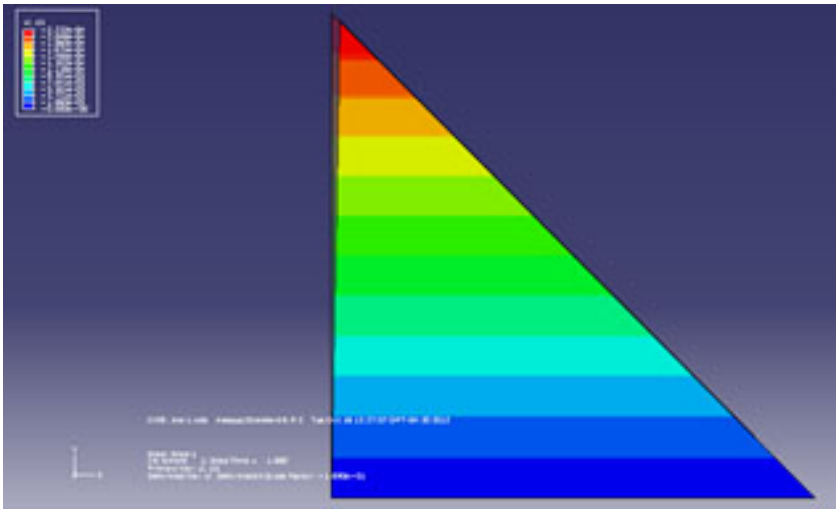

Figura 2.24: Resultados gráficos de *Abaqus* fuerzas en el plano. Un Elemento Finito CSTDKT

2.3.3.2 Prueba 2: Simulación de dos elementos triangulares

En la Figura 2.25 se ilustra el elemento finito a comprobar, el cual consta de dos triángulos rectángulos sometidos a fuerzas y a ciertas restricciones.

Especificaciones: Espesor = 1cm, E=215000, v=0.2

Los resultados obtenidos se muestran en la Tabla 2.9:

En las Figuras 2.26 y 2.27 se observa graficamente los resultados obtenido por el sofware *Abaqus* en el cual se aplicó el elemento finito ***STRI3****: A3 -node triangular facet thin shell* para la simulación.

Tabla 2.9: Resultados de la simulación.

Variables	Abaqus	PEEF
$Fu_1(Kg)$	19.9883	20.000
$Fv_1(Kg)$	9.91653	9.9199
$Fw_1(Kg)$	89.0241	89.024
$Mx_1(Kg.cm)$	86.6933	86.693
$My_1(Kg.cm)$	-897.583	-897.58
$U_2(cm)$	$-1.42616\ 10^{-5}$	$-1.4300\ 10^{-5}$
$V_2(cm)$	-0.000428487	-0.00042864
$W_2(cm)$	-1.5219	-1.5219
$\theta x_2(cm)$	0.00825282	0.0082528
$\theta y_2(cm)$	0.122078	0.12208
$U_3(cm)$	0.000200172	0.00020020
$V_3(cm)$	-0.000446713	-0.00044687
$W_3(cm)$	-1.37597	-1.3760
$\theta x_3(cm)$	0.0226767	0.022677
$\theta y_3(cm)$	0.0980174	0.098017
$Fu_4(Kg)$	-29.9883	-30.000
$Fv_4(Kg)$	0.0834693	0.080064
$Fw_4(Kg)$	110.976	110.98
$Mx_4(Kg.cm)$	-196.452	-196.45
$My_4(Kg.cm)$	-1102.42	-1102.4

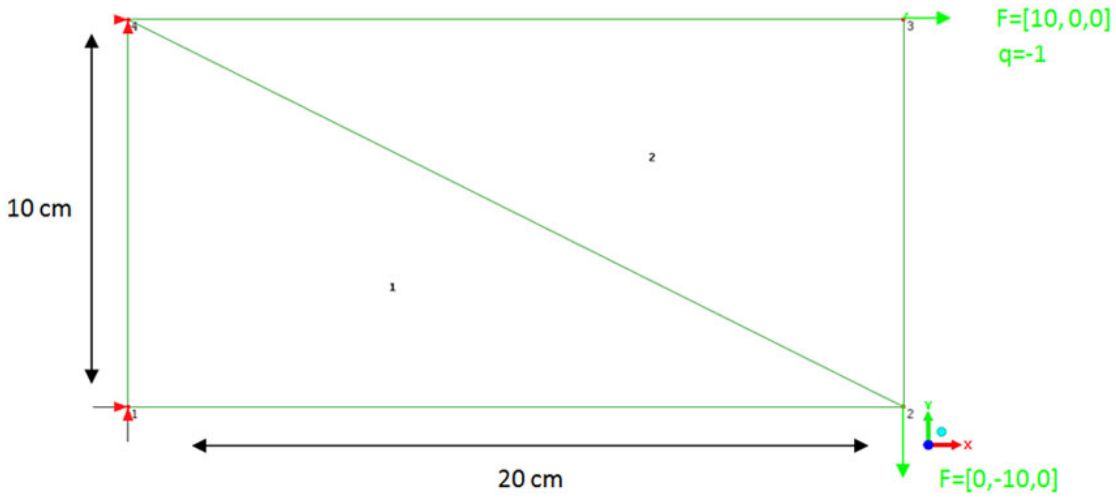

Figura 2.25: Dos elementos triangulares y solicitaciones.

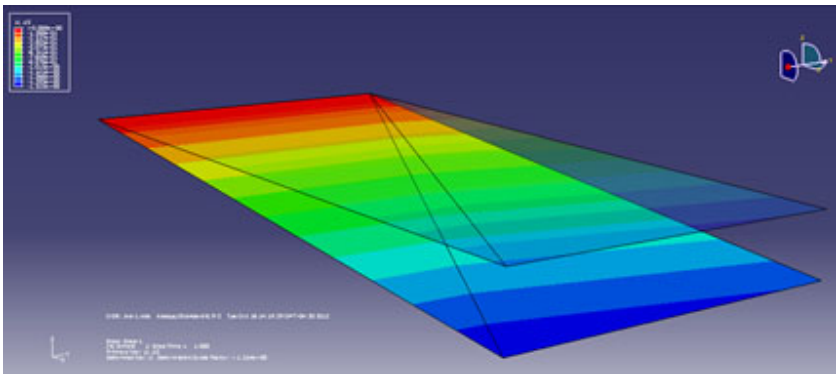

Figura 2.26: Resultados Gráficos de Abaqus para la flexión. Dos Elementos Finitos CSTDKT

2.3.3.3 Prueba 3: Simulación de cuatro elementos triangulares

En la Figura 2.28 se ilustra el elemento finito a comprobar, el cual consta de cuatro triángulos conectados de manera que formen una placa rectangular, los cuales son sometidos a fuerzas y a ciertas restricciones.

Especificaciones: Espesor = 1cm, E=215000, v=0.2

Los resultados obtenidos se muestran en la Tabla 2.10:

En la Figuras 2.29 y 2.30 se observa graficamente los resultados obtenido por el

Tabla 2.10: Resultados de la simulación.

Variables	Abaqus	PEEF
$Fu_1(Kg)$	-4.87899	-4.87860
$Fv_1(Kg)$	-4.99915	-5.0000
$Fw_1(Kg)$	400	400.00
$Mx_1(Kg.cm)$	4000	4000.0
$My_1(Kg.cm)$	-1551.04	1551.0
$Fu_2(Kg)$	-5.12101	-5.1214
$Fv_2(Kg)$	14.9991	15.000
$Fw_2(Kg)$	400	400.00
$Mx_2(Kg.cm)$	4000	4000.0
$My_2(Kg.cm)$	1551.04	1551.0
$U_3(cm)$	$5.22982\ 10^{-5}$	$5.2303\ 10^{-5}$
$V_3(cm)$	$-8.81434\ 10^{-5}$	$-8.8148\ 10^{-5}$
$W_3(cm)$	-1.48695	-1.4869
$\theta x_3(cm)$	-0.09593	-0.090593
$\theta y_3(cm)$	-0.0129072	-0.012907
$U_4(cm)$	$1.2897\ 10^{-4}$	$1.2898\ 10^{-4}$
$V_4(cm)$	$5.11603\ 10^{-5}$	$5.1164\ 10^{-5}$
$W_4(cm)$	-1.48695	-1.4869
$\theta x_4(cm)$	-0.090593	-0.090593
$\theta y_4(cm)$	0.0129072	0.012907
$U_5(cm)$	$2.92437\ 10^{-5}$	$2.9247\ 10^{-5}$
$V_5(cm)$	$-1.44728\ 10^{-5}$	$-1.4474\ 10^{-5}$
$W_5(cm)$	-0.774822	-0.77482
$\theta x_5(cm)$	-0.0679873	-0.067987
$\theta y_5(cm)$	$1.084\ 10^{-14}$	$-0.25841\ 10^{-16}$

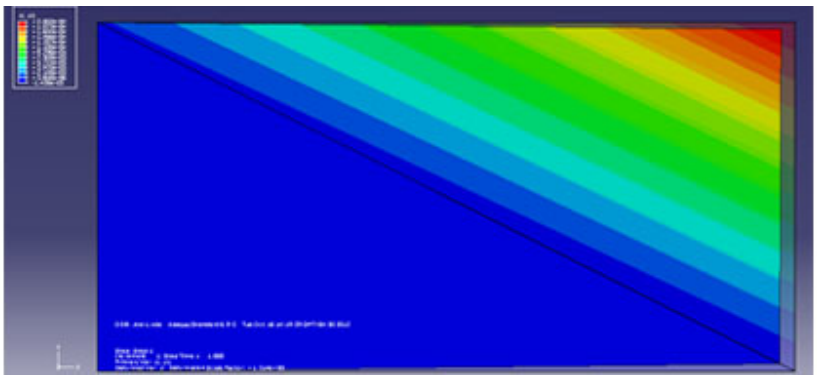

Figura 2.27: Resultados Gráficos de Abaqus para las fuerzas en el plano. Un Elementos Finitos CSTDKT

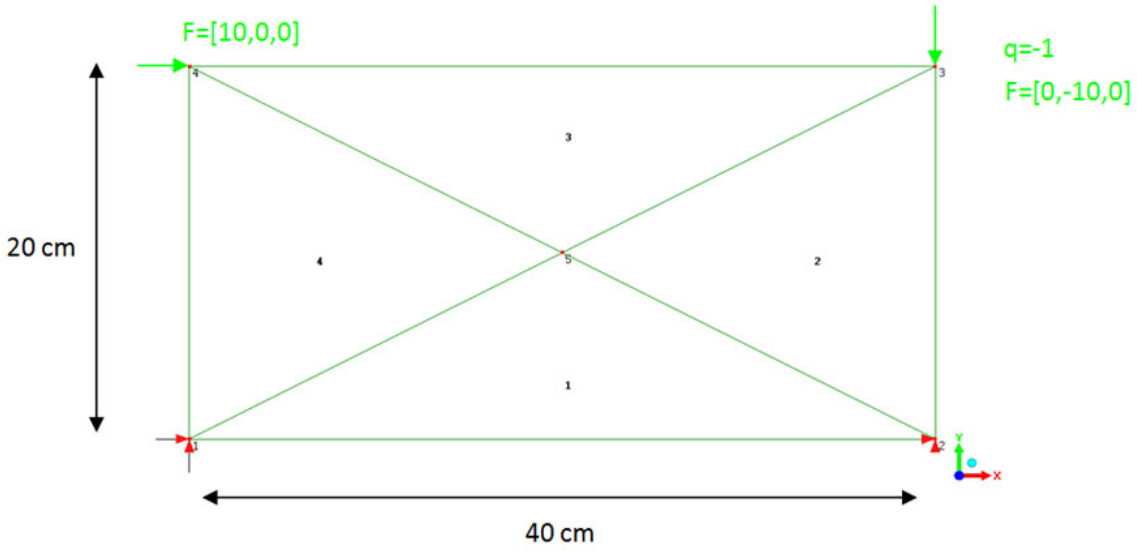

Figura 2.28: Cuatro elementos triangulares y solicitaciones.

sofware *Abaqus* en el cual se aplicó el elemento finito ***STRI3****: A3 -node triangular facet thin shell* para la simulación.

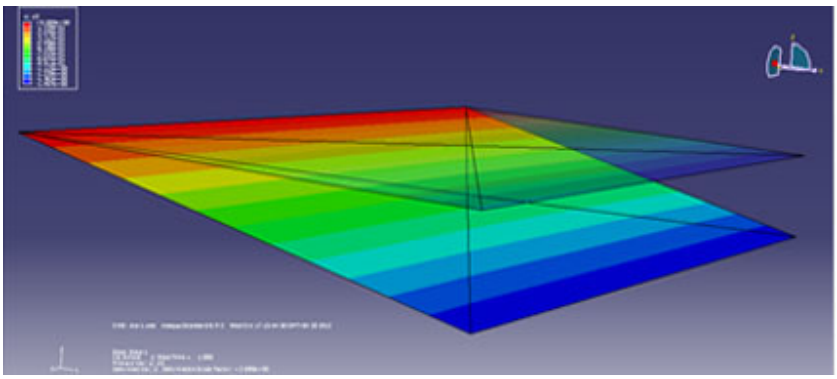

Figura 2.29: Resultados Gráficos de Abaqus para la flexión. Cuatro Elementos Finitos CSTDKT

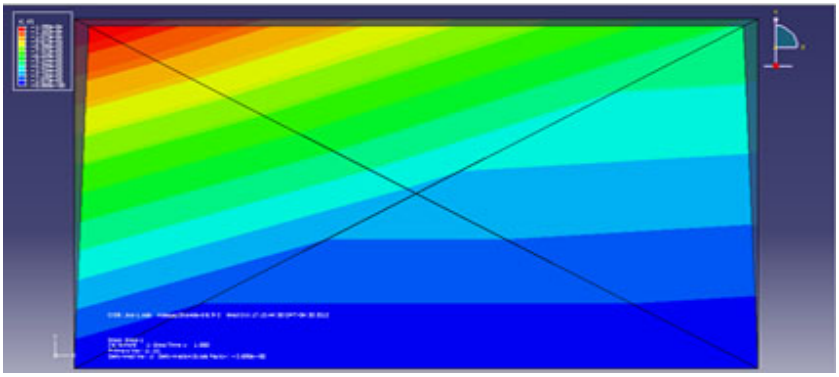

Figura 2.30: Resultados Gráficos de Abaqus para fuerzas en el plano. Cuatro Elementos Finitos CSTDKT

2.4 Análisis Dinámico para el elemento finito Placa General

Al realizar un análisis dinámico de un elemento finito, este se debe considerar la masa del elemento, tal propiedad permite generar una inercia en el mismo. Esta inercia al someterse ante una aceleración produce fuerzas adicionales a las presentes en la estructura en su estado de reposo.

Dado que solo en los nodos del elemento es donde se producen los desplazamientos, se asume que una aceleración solo se reflejará en estos. Al producirse movimiento de la placa, se tomara en cuenta como movimiento traslaciones, por lo cual las aceleraciones

angulares con respecto a los ejes se asumen 0. Quedando el vector de aceleración para el elemento de la siguiente forma:

$$\ddot{U} = \{ \ \ddot{u}_1 \quad \ddot{v}_1 \quad \ddot{u}_2 \quad \ddot{v}_2 \quad \ddot{u}_3 \quad \ddot{v}_3 \ \} \tag{2.68}$$

Donde

$\ddot{u}_i$ es la aceleración en dirección X

$\ddot{v}_i$ es la aceleración en dirección Y

e $i = 1, 2, 3$

Como el elemento placa presenta forma triangular, la masa del elemento se reparte en tres partes iguales, por lo que la matriz de masa para el elemento placa resulta así:

$$M^e = \frac{\rho A t}{3} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.69}$$

Donde

ρ = densidad del material

A= área del elemento

t = espesor de la placa

M = matriz de Masa para el elemento

Dado que el elemento tiene peso propio y hay presencia de movimiento, las fuerzas producidas deben ser adheridas a la ecuación general para realizar los cálculos.

$$\{Q\} = [K]\{U\} + [M^e]\{\ddot{U}\} - \{P\} \tag{2.70}$$

Donde

$\{Q\}$ es el vector de fuerzas internas del elemento

$[K]$ es la matriz de rigidez

$\{U\}$ el vector de desplazamiento

$[M]$ es la matriz de masa para el elemento

$\{\ddot{U}\}$ es el vector de aceleraciones

$\{P\}$ es el Vector de fuerzas externas

Desarrollado este elemento finito y comparado con su homólogo de *ABAQUS CAE*, le permite al usuario minimizar el tiempo en la introdución de datos referente a las cargas de la placa. Puesto que la placa solo aporta una carga (Fuerza) distribuida a lo largo del elemento viga, el analista solo se preocupará por el valor de cargas variables a las cuales debe someter la placa en su modelo.

Capítulo 3

Desarrollo del Servidor WEB

Dada la versatilidad y las múltiples ventajas que ofrece un servidor Web, como el control centralizado de los datos, el aumento de la escalabilidad y la gestión por acceso remoto, resulta ser mucho más factible aplicar una arquitectura de tipo Cliente - Servidor para el acceso al portal.

En este capítulo se describirá la arquitectura y los Servlets (Clases de JAVA) que permiten la comunicación y ejecución de diferentes funciones desde el Cliente PC.

3.1 Selección de la Tecnología

Para el diseño del servidor Web, se seleccionan las capas para su arquitectura. A continuación se especifica las capas usadas para el funcionamiento del servicio Web y en la Figura 3.1 se visualizan de forma gráfica para una mejor comprensión.

Sistema operativo Linux: Es un SO que utiliza un núcleo Linux, y su origen está basado en Debian. Ubuntu está orientado al usuario nivel y promedio, con un fuerte enfoque en la facilidad de uso y mejorar la experiencia de usuario. Está compuesto de múltiples *software* normalmente distribuido bajo licencias de *software* libre o de código abierto. Una versión evaluada fue Ubuntu 11.10. Ubuntu (2012) para el desarrollo de la aplicación.

Servidor Apache: Es un servidor Web HTTP de código abierto, para plataformas Unix, Microsoft Windows,Macintosh y otras, que implementa el protocolo

HTTP/1.1 y la noción de sitio virtual. Su desarrollo comenzó en 1995. El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido y se distribuye libremente con la licencia Apache. Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado según estadísticas de netcraft.com/. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios Web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. La versión evaluada para el servidor ha sido Apache HTTP Server 2.4.3. The Apache Software Foundation (2012a)

Apache Tomcat: Funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat es un servidor Web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor Web Apache. Tomcat fue escrito en JAVA, funciona en cualquier sistema operativo que disponga de la máquina virtual JAVA. La versión evaluada para el servidor ha sido Tomcat 6. The Apache Software Foundation (2012c) debido a la versatilidad, documentación y estabilidad que aporta.

Servlets: Los servlets son un conjunto de clases en el lenguaje de programación JAVA. Este conjunto modela una aplicación que corre dentro del servidor; además de ser utilizados para ampliar las capacidades del servidor donde se encuentra alojado. Aunque los servlets pueden responder a cualquier tipo de solicitudes, éstos son utilizados frecuentemente para extender las aplicaciones alojadas por servidores Web, de tal manera que pueden ser vistos como applets de JAVA que se ejecutan en servidores en vez de navegadores Web

Cliente GUI: Es el conjunto de clases que conforman la aplicación cliente, es decir, es la interfaz Hombre-Máquina albergada dentro de un Navegador Web para interactuar de forma sencilla con el usuario. Esta aplicación se comunica directamente

con el conjunto de servlets alojados en servidor remoto.

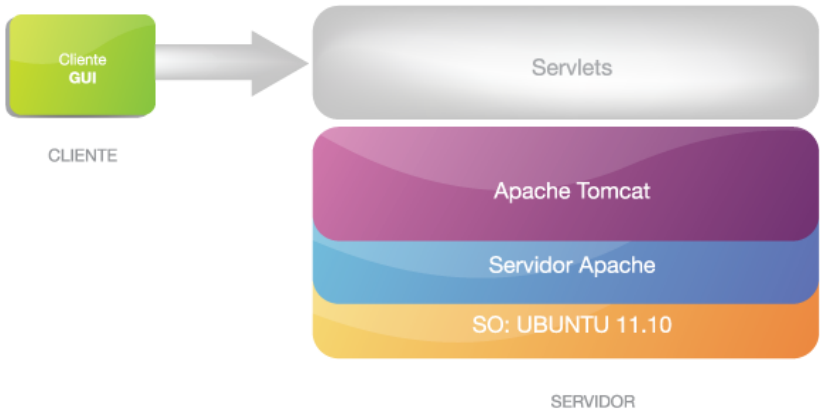

Figura 3.1: Capas del Servidor Web.

3.2 Manejadore de Acceso

Se entiende por gestión de usuarios y accesos un sistema integrado de políticas y procesos organizacionales que pretende facilitar y controlar el acceso a los sistemas de información y a las instalaciones.

Representa un conjunto de soluciones interrelacionadas que se utilizan para administrar diferentes procesos como: la autenticación de usuarios, restricciones de acceso, perfiles de cuentas, contraseñas y otros atributos necesarios para la administración de perfiles de usuario en una aplicación.

La gestión de usuarios o de acceso, es una parte fundamental dentro de un servicio Web, puesto que la información que se maneja dentro de este, es de suma importancia para el usuario y no puede correr el riesgo de perderla o dejarla a manos de cualquier actor que pueda acceder al servicio. En esta sección se presenta 4 alternativas para el manejo de usuarios, cada una en tecnologías diferentes.

3.2.1 Módulo RAMPART - Axis 2.0

Apache *Rampart* es el módulo Axis2 que proporciona funcionalidad de *WS-Security* para servicios Web *Axis2* y sus clientes. *Rampart* actualmente implementa *WS-Security*, *WS-SecurityPolicy*, *WS-SecureConversation* y especificaciones *WS-Trust*.

Suministra características de seguridad para servicios Web mediante la implementación de las siguientes especificaciones:

- WS-Security
- WS-SecurityPolicy
- WS-Trust
- WS-SecureConversation
- SAML 1.1
- SAML 2.0

La más reciente Versión del Módulo *Rampart* es la 1.6.2 (Abril, 2012), versión que fue probada e implementada para incorporar al portal. Esta versión presentaba problemas de estabilidad y ejecución al momento de realizar las pruebas, por tal motivo se decidió descartarla y probar con una tecnología más completa, más estable y de mayor popularidad.

3.2.2 Virtuoso Universal Server

Virtuoso Universal Server es un híbrido entre un motor de bases de datos y middleware que combina la funcionalidad de un Sistema Manejador de Bases de Datos Relacional (*RDBMS*) tradicional, Base de Datos Objeto-Relacional, base de datos virtual, Sistema de Base de Datos Federada (*RDF*), *XML*, texto libre, servidor de aplicaciones Web y la funcionalidad de servidor de archivos en un solo sistema.

En lugar de tener servidores dedicados para cada uno de los sistemas de funcionalidad antes mencionados, *Virtuoso* es un "servidor universal"; que permite en un único servidor multiproceso implementar múltiples protocolos. La edición de código abierto

de *Virtuoso Universal Server* también se conoce como *OpenLink Virtuoso*. El software ha sido desarrollado por *OpenLink Software* junto a Kingsley Uyi Idehen y Orri Erling como los arquitectos de software principales.

Algunas características de Virtuoso incluyen:

- Motor de base de datos objeto-relacional para (SQL, XML, RDF y texto)
- Servicios Web de la plataforma informática
- Servidor de aplicaciones Web
- Sistema de gestión de contenido Web (WCMS)
- Almacenamiento de Correo y Servidor proxy (POP3)
- Portabilidad de los datos

Virtuoso abarca una amplia gama de ámbitos, tradicionalmente distintos, en un único producto; de allí se deriva la versatilidad y el gran uso que puede obtener un producto con todo incluido. Sin embargo dada la experiencia con *Virtuoso* en uso y recopilación de información, en específico ejemplos y tutoriales para su uso, fue una experiencia desalentadora en el sentido de no poder encontrar alguna documentación que permitiera aplicar las bondades de este manejador de acceso con la aplicación a implementar; por tal motivo *Virtuoso* como servidor Web, fue descartado.

3.2.3 Open Am

Es un sistema de gestión de identidades (idP) y de control de acceso *Open Source*. Es una herramienta, realmente buena, para clientes que deseen ampliar sus perímetros técnicos y abrir sus diferentes servicios y aplicaciones a sus colaboradores.

Las exigencias de seguridad y empresariales de dichas interfaces entre colaboradores obligan a controlar al máximo el nivel del riesgo. Por esta razón OPENAM permite introducir, en forma nativa, una infraestructura escalable redundante y segura. *OpenAm* está basado en *Sun Java System Access Manager*, que es el corazón, de los productos comerciales de administración de acceso y federación.

Entre las características más resaltantes de *OpenAm*

- Fácil de instalar y configurar para satisfacer las necesidades de los clientes más exigentes
- 100% JAVA que permite una amplia variedad de configuraciones de despliegue
- Se integra fácilmente en los sistemas existentes
- Soporta el último protocolo XACML
- Soporte completo para la Autenticación, ampliamente utilizado en los sitios de redes sociales

Debido a la complejidad encontrada para enlazar y ensamblar este manejador de acceso con el portal, se decidió descartar esta opción.

3.2.4 Módulo Acceso PHP - PDP3D

Dado los inconvenientes presentados, en este caso, para no poder utilizar un sistema de acceso ya desarrollado, surgió la iniciativa de implementar el acceso de usuarios vía Web, directamente aplicando formularios *HTML* y enlazándolos con una Base de Datos *MYSQL* usando comunicación vía *PHP*.

Esta solución, que en principio funciona correctamente, deja un nivel de seguridad bajo, vulnerable donde el riesgo a violar la seguridad está latente. Sin embargo se deja para una versión 2.0 mejorar el sistema de acceso implementando alguna de las tecnologías mencionadas anteriormente o una nueva bien sea el caso.

3.3 Desarrollo Servicio

Los servlets no son más que objetos JAVA que se adaptan a una interfaz específica para poder ser ejecutados por un servidor que soporte servlets y poder comunicarse con las peticiones que le lleguen desde un cliente Web. Los servlets son a la capa servidora de aplicaciones lo que los applets son a la capa del cliente, pero se diferencian de estos en que los servlets no son elementos gráficos.

Pueden mantener ciertos objetos activos entre distintas peticiones, que sean útiles para posteriores ejecuciones, como por ejemplo pueden mantener un agrupamiento de

conexiones con base de datos, con lo cual su ejecución es más rápida al no tener que crear dichas conexiones cada vez que se ejecuten.

Al ser ejecutados totalmente en el servidor para su uso, no es necesario que en el navegador se habilite el uso de JAVA, ya que pueden enviar respuestas exclusivamente con HTML, pero también pueden hacer uso de *applets*, *JavaBeans*, o cualquier otra tecnología en el cliente desee.

Para la comunicación los servlets, en su mayoría, usan el protocolo HTTP, el cual es el más implantado en la Web y por tanto su uso ha sido el más extendido. Si se desea hacer un servlet que sea independiente del protocolo se debe extender a la clase **GenericServlet**, mientras que si se requiere hacer un servlet específico para la comunicación mediante el protocolo HTTP se debe usar la clase **javax.servlet.http.HttpServlet**, lo cual en realidad es una clase que contiene toda la funcionalidad de **GenericServlet** extendiendo su funcionalidad específica del protocolo HTTP.

Los servlets son clases JAVA normales que contienen dos atributos:

javax.servlet.http.HttpServletRequest: Se usa para recoger la petición desde el cliente, es decir desde el navegador Web, parámetros que recibe, cabeceras HTTP, entre otros.

javax.servlet.http.HttpServletResponse: Usado para enviarle la respuesta al cliente tras gestionar la petición de su servicio, ponerle cabeceras, y de más.

En la Figura 3.2 se puede apreciar de forma gráfica como es la comunicación, usando el protocolo HTTP, entre el cliente (Un navegador Web), con el servidor.

Para el desarrollo del servicio es necesario ejecutar un conjunto de servlets que contienen funciones específicas como: escribir archivos de textos en el servidor, ejecutar el programa generador de propiedades, ejecutar el PEEF y obtener resultados que serán enviados al cliente (applet).

A continuación se explica el funcionamiento de cada uno de los servlets alojados en el Servidor.

Inp.java La clase Inp.java es un Servlet que deriva de la Clase HttpServlet. Este servlet contiene dos métodos. En la Figura 3.3 se ilustra la comunicacion del Servlet con la interfaz

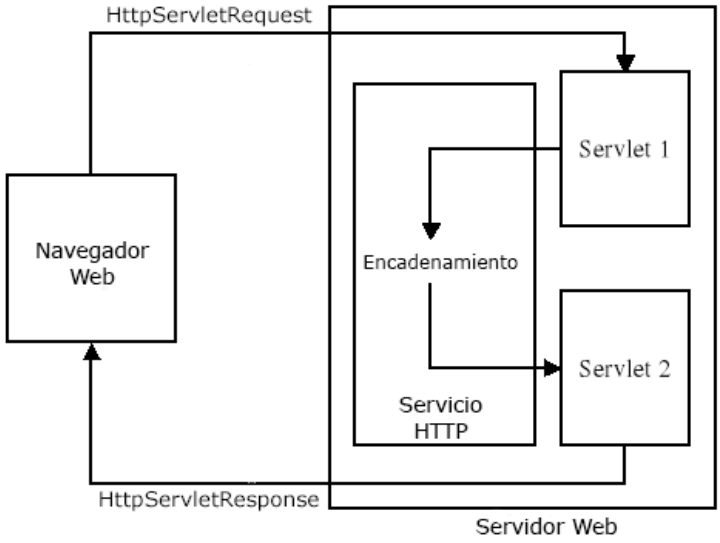

Figura 3.2: Diagrama de Comunicación Cliente - Servidor

doPost(HttpServletRequest request,HttpServletResponse response) (void): Método encargado de recibir una cadena de Texto, la cual contiene la información del archivo inp generada por cliente PC, esta cadena se deserializa y se convierte en un archivo de texto, con el nombre del proyecto y extensión inp. El archivo inp es almacenado en el Servidor y luego envía al Cliente PC la palabra "TRUE" como comprobación de que el archivo fue escrito con éxito.

generarArcInp(String nombre)(void): Método que se encarga de crear y almacenar el archivo de texto "ARCHIVOINP.txt"

Figura 3.3: Comunicación del Servlet Inp.java.

danio.java La clase danio.java es un Servlet que deriva de la Clase HttpServlet. Este servlet contiene dos métodos.En la Figura 3.4 se ilustra la comunicación del Servlet con la interfaz.

doPost(HttpServletRequest request,HttpServletResponse response) (void): Método encargado de recibir una cadena de Texto, la cual contiene la información acerca de la petición, si es nodo o elemento y su par de variables a graficar en los respectivos ejes coordenados. De acuerdo a la información obtenida de la petición, se busca en los archivos de resultados y se asigna a una cadena los datos, para posteriormente enviarlos al cliente.

leerDanio(String nombre, int ele) (String): Método que se encarga de leer desde el archivo de resultados según su valores recibidos por parámetro. Devuelve la cadena con la información.

Figura 3.4: Comunicación del Servlet danio.java.

leerRes.java La clase leerRes.java es un Servlet que deriva de la Clase HttpServlet. Este servlet contiene dos métodos. En la Figura 3.5 se ilustra la comunicación del Servlet con la interfaz.

doPost(HttpServletRequest request,HttpServletResponse response) (void): Método encargado de recibir una petición, la cual se indica al servlet que debe abrir el archivo "res" y enviar la información contenida como una cadena de texto. Esta cadena es recibida por el cliente e interpretada por la clase diagrama interacción para visualizar en pantalla las propiedades de la sección.

p3d.java La clase p3d.java es un Servlet que deriva de la Clase HttpServlet. Este

Figura 3.5: Comunicación del Servlet leerRes.java.

servlet contiene el siguiente método. En la Figura 3.6 se ilustra la comunicación del Servlet con la interfaz.

doPost(HttpServletRequest request,HttpServletResponse response) (void): Método encargado de recibir una cadena de Texto, la cual contiene la información del archivo p3d generada por Cliente PC, esta cadena se deserializa y se convierte en un archivo de texto, con el nombre del proyecto y extensión p3d. El archivo p3d es almacenado en el Servidor y luego envía al Cliente PC la palabra "TRUE" como comprobación de que el archivo fue escrito con éxito.

Figura 3.6: Comunicación del Servlet p3d.java.

proper.java La proper proper.java es un Servlet que deriva de la Clase HttpServlet. Este servlet contiene los siguientes métodos. En la Figura 3.7 se ilustra la comunicación del Servlet con la interfaz.

doPost(HttpServletRequest request,HttpServletResponse response) (void): Método encargado de recibir una cadena de Texto, la cual contiene el nombre de la sección del elemento para generar sus propiedades , esta petición

se realiza desde la clase principal del Cliente PC. El método devuelve la información de las propiedades en forma de cadena de caracteres, para ser enviada al cliente.

generarProp(String cad)(String): Esta función es la encargada de llamar y ensamblar el archivo de propiedades para la sección del elemento. Con el nombre de la sección recibida por parámetro, este método busca el archivo "res", el cual contiene la información de las propiedades de la sección, y extrae de él la información para luego serializarla en forma de cadena. El valor de retorno es la cadena con la información de las propiedades.

Figura 3.7: Comunicación del Servlet proper.java.

pruebaRun.java La clase pruebaRun.java es un Servlet que deriva de la Clase HttpServlet. Este servlet contiene el siguiente método. En la Figura 3.8 se ilustra la comunicación del Servlet con la interfaz.

doPost(HttpServletRequest request,HttpServletResponse response) (void): Este método se encarga de ejecutar el Programa Endógeno de Elementos Finitos (PEEF) para leer el archivo inp y producir los resultados correspondientes. Este Servlet es llamado desde la clase principal para enviar la petición de ejecución.

res_i.java res_i.java es un Servlet que deriva de la Clase HttpServlet. Este servlet contiene los siguientes métodos. En la Figura 3.9 se ilustra la comunicación del Servlet con la interfaz.

doPost(HttpServletRequest request,HttpServletResponse response) (void): Método encargado de recibir una cadena de texto con el nombre de la

Figura 3.8: Comunicación del Servlet pruebaRun.java.

sección del elemento y así generar su archivo res para el extremo i , esta petición se realiza desde la clase GrupoSecciones del Cliente PC.

generarRes(String nom)(String): Esta función es la encargada de escribir en el Servidor el archivo res de la sección para el extremo i. Solo recibe el nombre de la sección y reestructura el archivo res para adaptarlo a los ejes locales.

Figura 3.9: Comunicación del Servlet res_i.java.

res_j.java res_j.java es un Servlet que deriva de la Clase HttpServlet. Este servlet contiene los siguientes métodos. En la Figura 3.10 se ilustra la comunicación del Servlet con la interfaz.

doPost(HttpServletRequest request,HttpServletResponse response)(void): Método encargado de recibir una cadena de texto con el nombre de la sección del elemento y así generar su archivo res para el extremo j, esta petición se realiza desde la clase GrupoSecciones del Cliente PC.

generarRes(String nom)(String): Esta función es la encargada de escribir en el Servidor el archivo res de la sección para el extremo j. Solo recibe el nombre

de la sección y reestructura el archivo res para adaptarlo a los ejes locales.

Figura 3.10: Comunicación del Servlet res_j.java.

result.java result.java es un Servlet que deriva de la Clase HttpServlet. Este servlet contiene los siguientes métodos. En la Figura **??** se ilustra la comunicación del Servlet con la interfaz.

doPost(HttpServletRequest request,HttpServletResponse response) (void): Método encargado de recibir una cadena de texto con el número del elemento o nodo y las variables para ser graficadas en el cliente PC. Este servlet presenta comunicación directamente con la clase RGraficador del cliente PC.

leerResult(String nombre, int tipo,int ele,int ex,int ey)(String): Este método recibe la información necesaria para extraer de los archivos de texto, en el servidor, aquella que se enviará al cliente para posteriormente graficarla. El método retorna una cadena de texto con la información a ser graficada.

Figura 3.11: Comunicación del Servlet seccionTXTServlet.java

seccionTXTServlet.java La seccionTXTServlet.java es un Servlet que deriva de la Clase HttpServlet. Este servlet contiene los siguientes métodos. En la Figura

3.12 se ilustra la comunicación del Servlet con la interfaz.

doPost(HttpServletRequest request,HttpServletResponse response) (void): Método encargado de recibir la petición como una cadena de texto, la cual contiene la información del archivo sección txt. Esta información es enviada desde la clase Secciones.java del cliente PC. Al recibir la información esta se escribe en un archivo de texto dentro del servidor y posteriormente ejecuta el Generador (gen_new_m) para crear el archivo res asociado al archivo seccion.txt

generaRes()(boolean): Este método hace el llamado al generador para crear el archivo "res" de la sección correspondiente.

Figura 3.12: Comunicación del Servlet seccionTXTServlet.java

Una vez desarrollada la capa de Servlets encargada de funciones, que por parte del cliente no son posible ejecutar debido a las restricciones de seguridad de los applets, hacen el complemento ideal del cliente para una exitosa ejecución del Portal.

Capítulo 4

Desarrollo del Cliente PC

En esta sección se contempla la metodología y la referencia al código con el cual está diseñado la interfaz o el Cliente PC del Portal De Pórticos 3D (PDP3D). El PDP3D está desarrollado en un lenguaje de nivel intermedio como lo es *JAVA*.

Una de las ventajas de trabajar en java es la facilidad para desarrollar aplicaciones que corran en diferentes tecnologías, esto hace mucho más atractivo la escogencia del lenguaje, así esta aplicación podrá ser ejecutadas en múltiples plataformas como escritorio, móvil o Web.

Esta aplicación es la interfaz que permite al usuario interactuar de manera más amigable e intuitiva para el análisis de estructuras de concreto armado en 3 dimensiones. El cliente PC está implementado de manera modular donde se usan un conjunto de sub-rutinas para el buen desempeño del mismo y de mayor comprensión.

El Portal de pórticos 3D, como la mayoría de las aplicaciones actuales, trabaja con el uso de ventanas y menús. Estas ventanas y menús están diseñadas para facilitar el trabajo al usuario en el momento de introducir la información de la estructura.

4.1 Funcionamiento del Cliente PC

PORTAL DE PÓRTICOS 3D (PDP3D)

A través de este programa, se puede simular el comportamiento inelástico y el daño de las estructuras de concreto armado bajo la teoría de daño concentrado. El principal

objetivo es el de diagnóstico de las estructuras a través de un mapa de daño. El programa **PDP3D** es capaz de calcular la propagación de las grietas y el daño que se está generando en los elementos estructurales de los pórticos.

4.1.1 Requerimiento para el uso del programa

Hardware Los requerimientos físicos para la ejecución del PDP3D son:

- Una maquina con un procesador mayor a 1.2 GHz
- 512 MB de memoria RAM
- Conexión a internet

Software Los requerimientos de software son:

- JAVA Rutine Enviroment (JRE) 7 o superior
- Navegador WEB (Mozilla FireFox, internet Explorer, Google Chrome, entre otros)

4.2 Ambiente Principal del Programa

Para acceder al **PDP3D** es necesario conectarse desde el navegador a la dirección Web del Servidor, en este caso se usará un servidor local para la demostración del funcionamiento.

Luego de entrar a la dirección Web será necesaria la autenticación del usuario para acceder a la aplicación, de no ser así este debe registrarse. Este proceso se aclara en la Figura 4.1.

Como se indicó al final del capítulo 3, se decidió usar un manejador de acceso primitivo para el portal. Este sistema de acceso consta de un conjunto pequeño de módulos programados en php que permiten la autenticación de un usuario registrado a través del cotejamiento de un login y password, ingresado en el formulario, con los almacenados en la base de datos del portal.

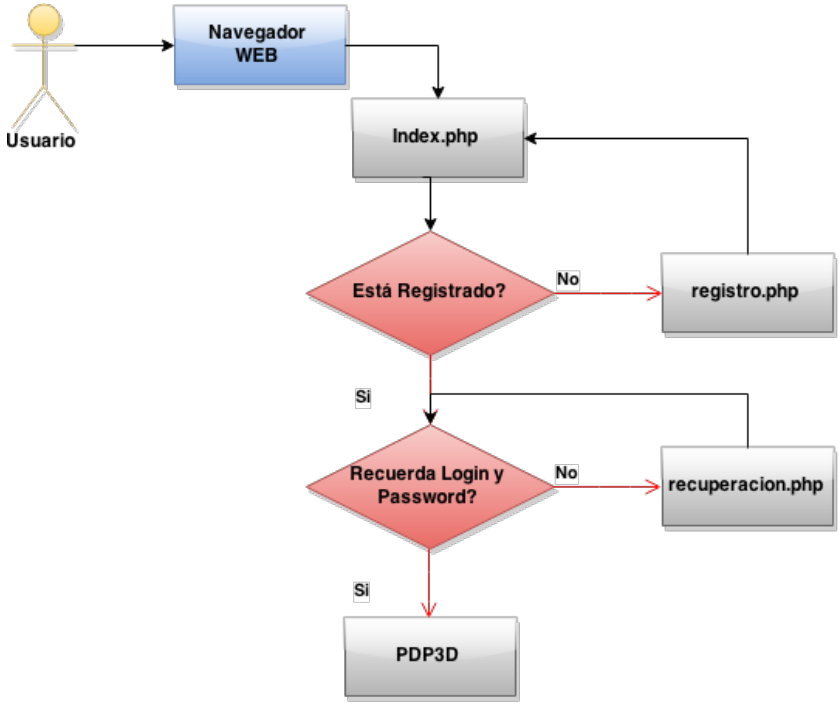

Figura 4.1: Diagrama para el acceso a PDP3D

La secuencia para el desarrollo de un buen y correcto análisis se ilustra en el diagrama de las Figura 4.2 y Figura 4.3, es decir, el usuario deberá completr cada paso en su totalidad para realizar un analisis correctamente.

El ambiente de la aplicación está compuesta por dos barras de Herramientas, una de opciones generales para visualización ubicada de forma vertical y otra de manejo de archivos (Nuevo proyecto, Abrir Proyecto, Abrir archivo INP, Guardar Proyecto, Guardar Como, Cerrar, además del nombre del Proyecto) ubicada en la parte superior de forma horizontal. En la Figura 4.4 se visualiza el entorno del programa.

Nuevo Proyecto

Al hacer click en este botón se despliega un cuadro de dialogo, donde deberá indicar el nombre que tendrá el proyecto. El programa crea un archivo de datos con la extensión P3D donde se guardan todos los datos del pórtico introducidos por el usuario.

Una vez creado se despliega una pantalla donde se deben definir el modelo, definiendo el nombre del proyecto, la opción si se trabajara con placa o no y por último la geometría donde se debe introducir la cantidad de niveles y tramos que llevara el

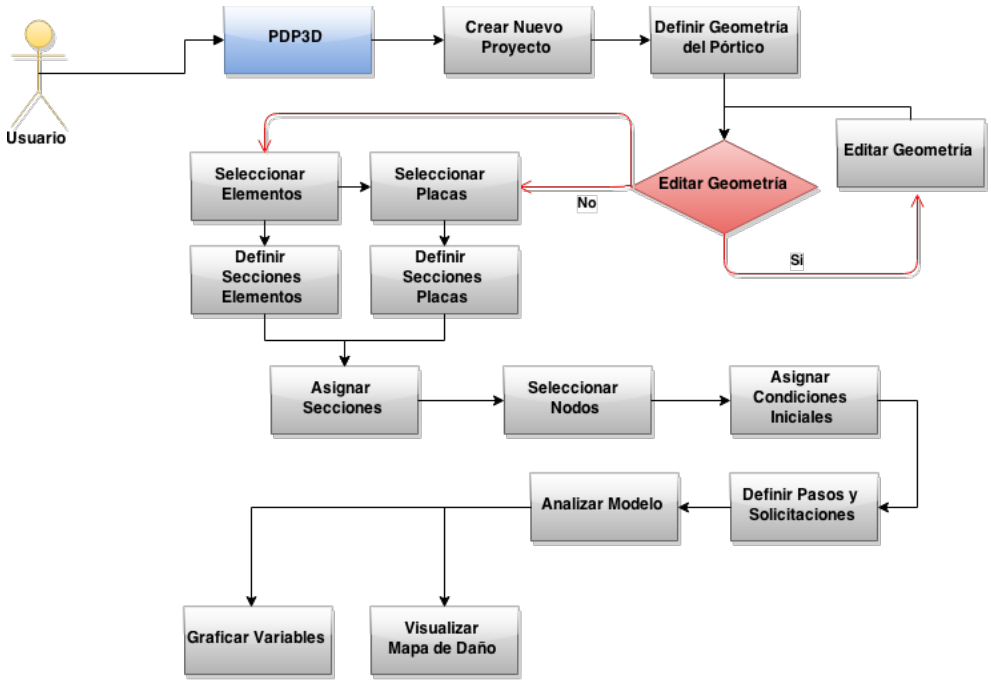

Figura 4.2: Diagrama de secuencia para el análisis de un nuevo modelo en PDP3D

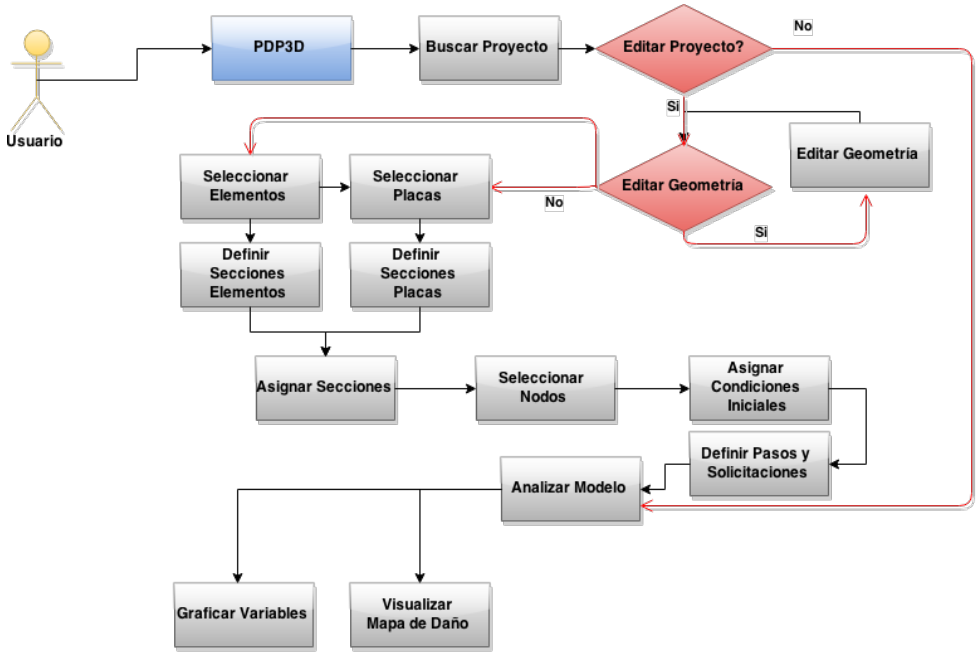

Figura 4.3: Diagrama de secuencia para el análisis de un modelo existente en PDP3D

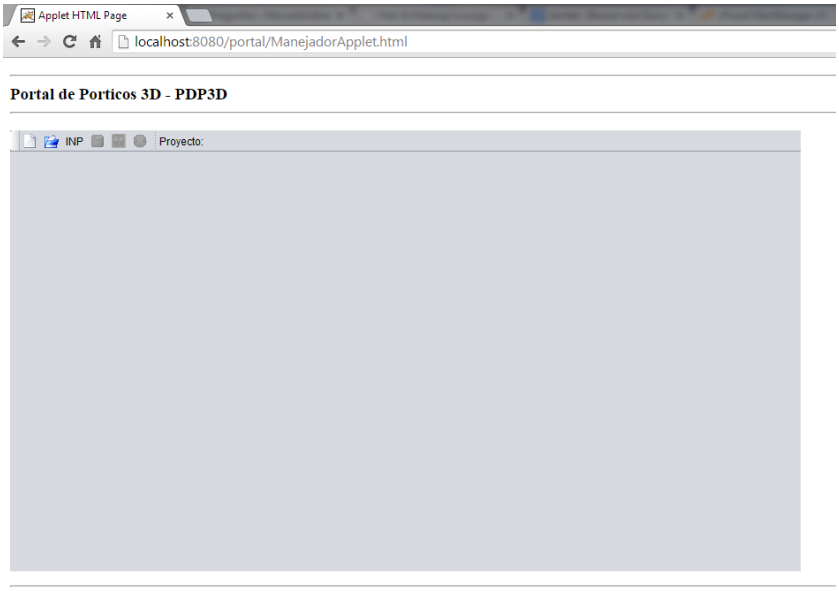

Figura 4.4: Entorno de la aplicación

proyecto.

Opciones del manejo de archivos

Abrir proyecto: da la opción de abrir un proyecto creado anteriormente.

Guardar: Despues de haber creado o abierto un proyecto se activara el botón dando la opción de guardarlo. En este particular se establece comunicación con el servidor para guardar el archivo de forma remota.

Guardar Como: Despues de haber creado o abierto un proyecto se activara el botón dando la opción de guardarlo donde el usuario considere. En este particular se establece comunicación con el servidor para guardar el archivo de forma remota.

IPN: Da la opción de abrir el archivo IPN de proyectos ya creados.

Digitalización de la estructura

Cuando es creado el proyecto, después de introducir el nombre, el programa pide establecer las características geométricas del modelo de la estructura en tres dimensiones a través de una nueva ventana llamada “Definición del modelo”,como se observa en la Figura 4.5, donde se debe especificar:

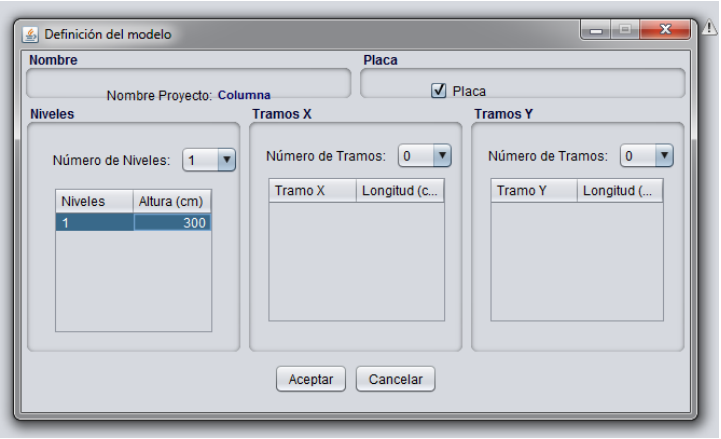

Figura 4.5: Digitalización de la Estructura Básica

- Si se quiere trabajar con placa o no
- Niveles, se selecciona el número de niveles que corresponde a la estructura a analizar. Y se debe especificar la altura de entrepiso en centímetros.
- Tramos, se debe definir el número de tramos tanto en la dirección X como en la dirección Y, indicando la longitud que tendrá cada uno de los tramos en centímetros.

Una vez digitalizada la estructura deseada, s e mostrará la pestaña "GEOMETRÍA" como se ilustra en la Figura 4.6. En esta ventana se definen las características geométricas del modelo, y estarán activos los siguientes tres botones:

Editar: Al hacer click a la opción "Editar" se despliegan las siguientes opciones:

Redefinir: abre nuevamente la ventana de "Definición de Modelo" y permite modificar la geometría del modelo creado ya sea en sus dimensiones, número de niveles y/o número de tramos.

Eliminar Elementos: despliega una ventana donde se debe seleccionar los elementos (Columnas y Vigas) que se requieren.

Agregar Elementos: adiciona elementos requeridos entre dos nodos existentes. Para agregar un elemento primero se debe seleccionar el número de elementos a agregar, luego indicar dónde estará ubicado cada elemento adicional.

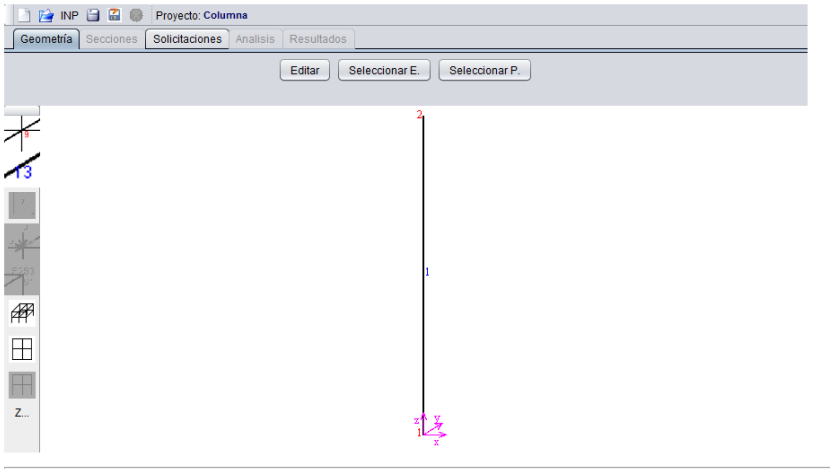

Figura 4.6: Entorno de la aplicación ampliada

Seleccionar Elementos: Cada elemento debe ser asignado a un grupo de elementos. Al crear los grupos de elementos, estos tendrán las mismas secciones, los cuales pueden ser creados haciendo click en la opción agregar (+) y eliminados en la opción quitar (-).

Al crear un conjunto, se debe seleccionar los elementos que contendrá dicho conjunto lo cual se hace doble click en el espacio vacío debajo de la columna "Elementos" y abrirá una nueva ventana donde lista los elementos presentes en la geometría del modelo.

Seleccionar Placas: Cada placa debe ser asignada a un grupo, los cuales pueden ser creados haciendo click en la opción (+) y eliminados en la opción (-).

Al crear un conjunto de placas, para seleccionar los elementos de este tipo que contendrá dicho conjunto se debe hacer doble click en el espacio vacío debajo de la columna "Placas", el cual desplegara una nueva ventana donde muestra todas las placas que existen en el pórtico modelado. Las placas que tengan características geométricas y condiciones de carga idénticas pueden ser asignadas a un mismo grupo.

En la pestaña "SECCIONES" se crean y asignan las secciones tanto para los elementos, como para las placas.

Sección Elementos: Al hacer click se despliega una ventana con la opción de crear diferentes secciones que llevara el proyecto, editarlas y/o eliminarlas. Al desplegar la ventana de "grupo de secciones" se activaran tres diferentes botones:

Nuevo: Para crear una nueva sección se debe seleccionar este botón, el cual desplegara otra pantalla adicional llamada "Definir Sección" donde se asigna toda la información de las secciones incluyendo las características del material.

Editar: Permite editar las seccionas ya creadas. Para modificar las secciones, se debe seleccionar la sección y se le da al botón "Editar", se desplegara la ventana de "Definir Sección" con todas las características de la sección seleccionada.

Eliminar: Permite eliminar una a una, las secciones ya creadas con solo seleccionarla.

DEFINIR SECCION:

En este módulo se presentan los campos necesarios para la introducción de los valores de secciones rectangulares. Graficamente el módulo puede observarse en la Figura 4.7.

Nombre de Sección: Define el nombre de la sección.

Tipo de Sección: En la parte superior derecha se selecciona el tipo de sección ya sea (columna, Viga X, Viga Y u Otro).

Área: Se definen las dimensiones (b, h) según la vista mostrada en la pantalla.

Masa: La masa del elemento en $\frac{(Ton*s^2)}{cm}$, puede calcularse considerando el peso del elemento. $Masa = \frac{(Volumen*\delta concreto)}{gravedad}$

Aceros Longitudinales: El programa por defecto trae dos capas de acero, compuesta por dos cabillas de $\frac{3}{4}$" con un recubrimiento de $2.5cm$.

Para definir el acero longitudinal se debe tomar en cuenta los siguientes aspectos:

- Definir el número de capas a incorporar, utilizando dos capas como trae por defecto o aumentándolo utilizando el botón agregar (+) o disminuyéndolo con el botón quitar (-) ubicados en la parte derecha.

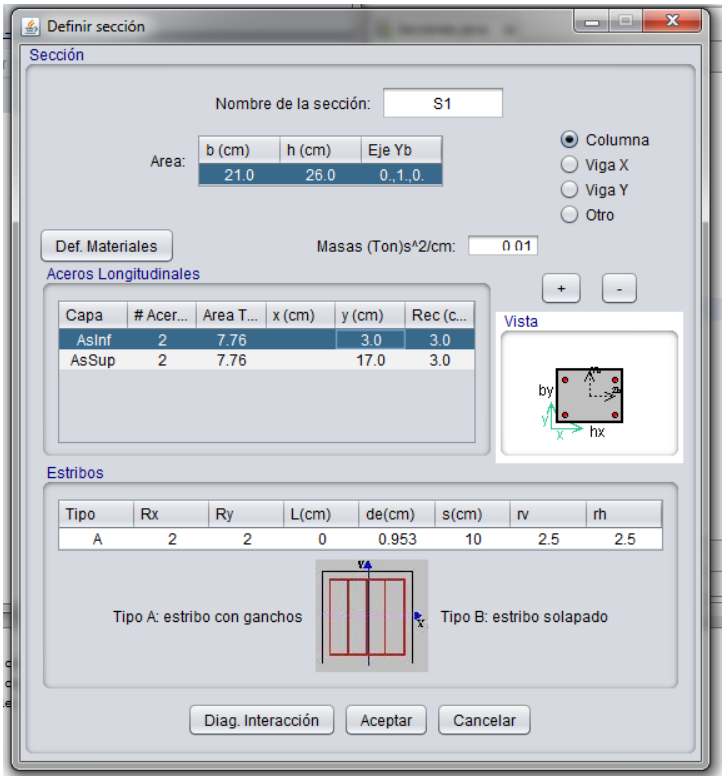

Figura 4.7: Digitalización de la Sección del Elemento Viga - Columna

- El número de barras a colocar que tendrá cada una de las capas.

- El área total del acero a utilizar por capa.

- La ubicación de las cabillas se podrá hacer de dos maneras:

  - Colocando en X la distancia horizontal que va desde el borde izquierdo de la sección hasta el centro de la barra y luego colocando el recubrimiento que será la distancia comprendida desde el borde inferior de la sección hasta el centro de la cabilla.

  - Colocando la altura que habrá desde el borde inferior de la sección hasta el centro de la barra y luego el recubrimiento que será la distancia en horizontal que habrá desde el borde izquierdo de la sección hasta el centro de la barra.

Estribos: Se agregan los datos correspondientes al acero transversal de los elementos:

Tipo: se define el tipo de estribo con el que se va a trabajar, teniendo dos tipos como opciones:

- Tipo A: estribo con ganchos

- Tipo B: estribo solapado

Tipo A: si se trabaja con estribos tipo A, se deben definir Rx y Ry

Rx: define el número de ramas en la dirección X de la sección transversal. Ry: define el número de ramas en dirección Y de la sección transversal.

Tipo B: si se elige trabajar con los estribos tipo B Rx y Ry son iguales a 0 y se debe proporcionar una longitud L (perímetro) de estribo empleada en el reforzamiento transversal.

Diámetro (de): se debe definir en centímetros el dímetro de los estribos.

Separación entre los estribos: se debe especificar la separación que tendrán los estribos en la zona confinada.

Recubrimiento: Rh y Rv serán los recubrimientos horizontales y verticales que tendrán los estribos partiendo de la cara del elemento al centro del estribo.

Definir Materiales: En esta opción se definen las especificaciones mecánicas usados para la estructura.

Para el concreto, el usuario debe proveer al sistema de los siguientes datos:

- Resistencia del concreto ($f'c$ en Kg/cm^2)
- Deformación máxima del concreto ($eo < 0.002$)
- Deformación ultima del concreto ($eoc < 0.004$)
- Módulo de elasticidad ($E, Kg/cm^2$)
- Máxima deformación última del concreto confinado ($eccu \geq 0.004$)
- Tasa de deformación (alta o baja) se refiere a la rapidez con la cual se aplica la carga al pórtico a analizar. Generalmente, se considera la tasa de deformación baja cuando las cargas son aplicadas lentamente, estáticas; mientras

que la opción alta se utiliza cuando las cargas son aplicadas rápidamente, dinámicas.

- Opciones de diseño (confinado, no confinado)

Para el acero, el usuario debe especificar la curva esfuerzo deformación del acero utilizado, suministrando los siguientes valores:

- Máxima deformación (esm)
- Deformación de cedencia (ey)
- Deformación al final de la cedencia (esh)
- Esfuerzo último ($fsu, Kg/cm^2$)
- Esfuerzo cedente ($fy, Kg/cm^2$)
- Esfuerzo cedente de refuerzo transversal ($fy, kg/cm^2$)

Se debe tomar en cuenta lo siguiente: $ey \leq esh \leq esm, fsu > fy$

Diagrama de Interacción: Permite al usuario visualizar los diagramas de interacción para cada uno de los elementos del sistema, tanto para el eje Yb como el eje Zb. Estos diagramas son referidos a los momentos críticos o de agrandamiento (Mcr), Momento plástico o de fluencia (Mp), y Momento Último (Mu) y la Curvatura Ultima con sus respectivos niveles de Carga Axial (C). En la Figura 4.8 se observa un ejemplo del Diagrama de Interacción. Tiene la capacidad de mostrar los momentos positivos y negativos, además de las curvaturas positivas y negativas.

Sección Placa: Al hacer click en este botón se activaran tres botones

Nuevo: permite crear una nueva placa, donde se despliega una nueva ventana llamada “Sección Placa”.

Sección Placa: donde se debe establecer las siguientes características de la placa

- Nombre Sección: define el nombre de la placa.
- Modulo de Elasticidad (E) en Kg/cm^2.

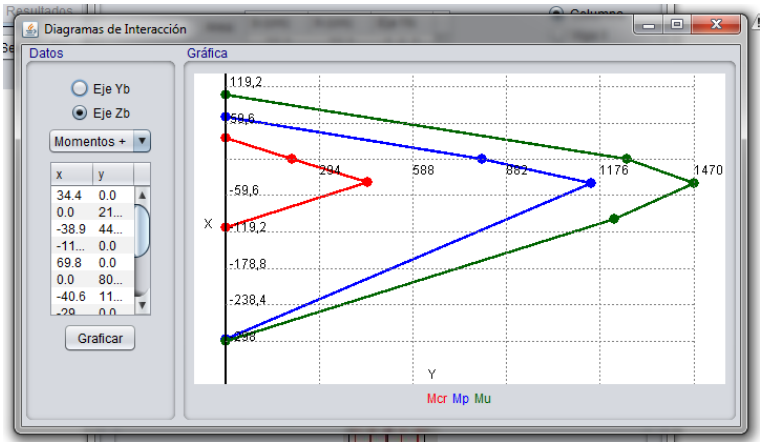

Figura 4.8: Diagrama de Interacción

- Coeficiente de Poisson (v)
- Espesor de la placa (t) en cm
- Densidad de masa (d) en Kg/m^3

Editar: permite editar las seccionas ya creadas. Para modificar las secciones, se debe seleccionar la sección y se le dá al botón "Editar", se desplegará la ventana de "Definir Sección" con todas las características de la sección seleccionada.

Eliminar: permite eliminar una a una, las placas ya creadas con solo seleccionarla.

Asignar Sección: Despliega una ventana que permite asignar las secciones y placas definidas al conjunto de elementos y conjunto de placas.

SOLICITACIONES: Para asignar las solicitaciones, conjunto de cargas (Fuerzas puntuales o distribuidas), desplazamientos o aceleraciones, a las cuales se someterá la estructura, se deben seleccionar los nodos, después se debe indicar las condiciones iniciales y finalmente introducir los pasos de análisis.

Seleccionar Nodos: Para seleccionar los nodos deben crearse los conjuntos de nodos, estos conjuntos llevaran las mismas restricciones y solicitaciones. Haciendo click al botón "+" cada conjunto de nodos puede ser renombrado presionando doble click en la casilla debajo de la columna "nombre".

Condiciones iniciales: Dependiendo del conjunto de nodos se debe asignar las diferentes restricciones que tendrán para establecer las condiciones de apoyo del pórtico, ya

sea la restricción de los desplazamientos en las diferentes direcciones (Ux,Uy,Uz), o la restricción de las rotaciones posibles $(\theta x, \theta y, \theta z)$.

Pasos de análisis: En esta opción podemos crear los pasos de análisis que tendrá el pórtico, la estructura o elemento pudiendo elegir la duración del paso, el nodo al que será sometido y la magnitud de la fuerza, desplazamiento o aceleración, y la dirección que esta tendrá. En la Figura 4.9 se visualiza el diseño de la ventana.

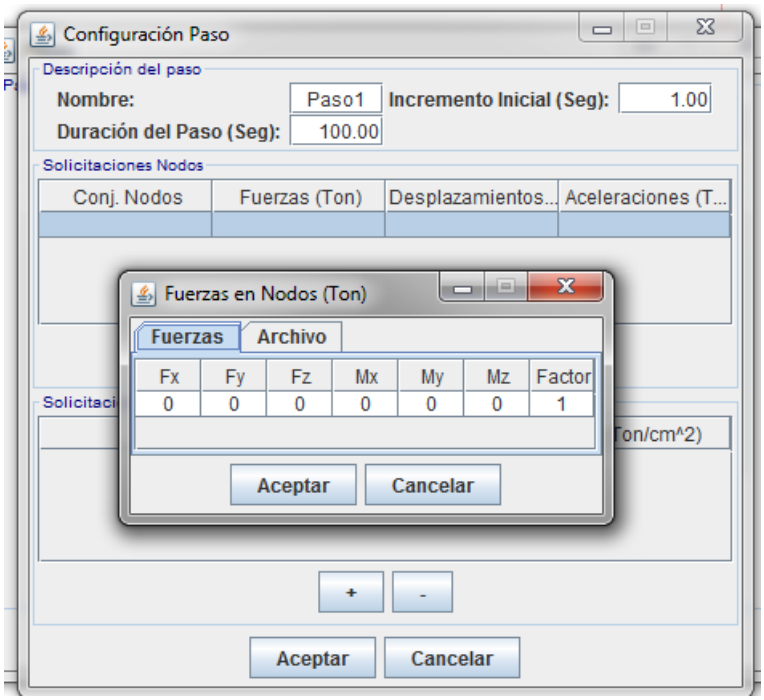

Figura 4.9: Configuración del Paso de análisis

Para introducir los pasos de análisis se oprime el botón "Pasos Análisis" que abrirá la ventana "Pasos" Luego para crear un nuevo paso, en la opción Nuevo se abre una nueva ventana "Configuración Paso" donde se piden las características principales del paso los cuales son necesarios para el análisis Las caracterizas necesarias son:

Nombre: Permite al usuario asignarle un nombre para identificar el paso creado.

Incremento Inicial (seg): valor del incremento del paso en segundos, con el cual el programa empieza el proceso. En caso de ser un análisis dinámico es referido al primer incremento inicial de tiempo que tiene el registro de aceleraciones.

Los registros de aceleraciones contenidos en la carpeta del programa tienen un incremento de 0.01seg.

Duración del Paso (seg): Para el caso de un análisis estático, la duración del paso indica el tiempo en el cual se desea aplicar la carga o desplazamiento. En caso de ser un análisis dinámico, la duración se refiere al tiempo que será sometida la estructura al sismo.

Luego, el usuario podrá elegir dependiendo del conjunto de nodo o conjunto de placas, las solicitaciones que tendrá por paso ya sea, Fuerzas, Desplazamientos o Aceleraciones que serán aplicadas. Para introducir las solicitaciones primero se debe crear haciendo click en el botón "+" luego se tiene que seleccionar el conjunto de Nodos al cual le será aplicada la solicitación. A continuación, se asignara las solicitaciones, que serán aplicadas en el paso a analizar. Estas pueden ser:

Fuerzas (Ton): Para introducir una fuerza se hace doble click en el espacio vacío ubicado en la segunda columna. En esta opción se pueden introducir fuerzas y/o momentos eligiendo la dirección requerida, se introduce la magnitud de la fuerza en toneladas y el momento en $Ton.cm$ en la casilla correspondiente.

Desplazamiento en Nodos (cm): Igualmente se hace doble click en el espacio vacío correspondiente. En esta opción el usuario podrá introducir tanto desplazamientos, como rotaciones introduciendo la magnitud en la dirección requerida. Los desplazamientos deben ser introducidos en centímetros (cm) y las rotaciones en radianes

Aceleraciones (Ton/cm^2): En este caso el usuario podrá escoger entre una lista de registros que se encuentran almacenados en la carpeta del programa o bien introducir su propio archivo de registros teniendo cuidado que el mismo cumpla con el formato que se presenta y la extensión *amp* para que pueda ser leído por el sistema.

Estos registros se presentan como archivos de texto con extensión *amp* contendido en ocho (8) columnas separadas por comas (,) donde se especifica el intervalo de tiempo así como el registro de aceleraciones de los mismos.

Los sismos tienen una duración de 29.99segundos y la amplitud varía según la intensidad del sismo. (La primera columna corresponde al tiempo y la segunda a las aceleraciones registradas).

ANALIZAR: Luego de crear la geometría, las secciones, solicitaciones y paso de análisis del modelo, se realiza el análisis de la estructura. Para comenzar el análisis es necesario guardar el proyecto, si este es guardado correctamente aparecerá una pequeña ventana indicando que ha sido guardado con éxito.

Una vez guardado se activara el botón de "Analizar". Este botón establece la comunicación y emite la petición para llamar a PEEF y proceder al análisis. Al comenzar se podrá observar en una ventana emergente el proceso de simulación indicando el el tiempo de itecaión de ejecución del mismo. Si el análisis se realiza de manera correcta y completa al final de la ventana emergente indicara que su proceso ha sido culminado exitosamente. De lo contrario esta lista quedará incompleta.

RESULTADOS: Una vez culminado el análisis correctamente, se activara la pestaña de "Resultados" donde se grafican y visualizan las variables calculadas durante el análisis. Se presentaran dos opciones: Gráficas y Mapa de Daño.

Gráficas: En la primera opción se realizan las graficascbidimensionales correspondientes a los nodos o elementos. Un ejemplo se puede observar en la Figura 4.10. Las variables a graficar están constituidas por dos ejes X y Y.

Para graficar se debe elegir entre las dos variables: nodos o elementos. Si se desea graficar para un nodo:

Se debe elegir cuál de los nodos existentes se desea graficar. El usuario podrá elegir cualquiera de las variables en los dos componentes.

Las variables son:

- Tiempo
- Desplazamiento en X - U1
- Desplazamiento en Y - U2
- Desplazamiento en Z - U3

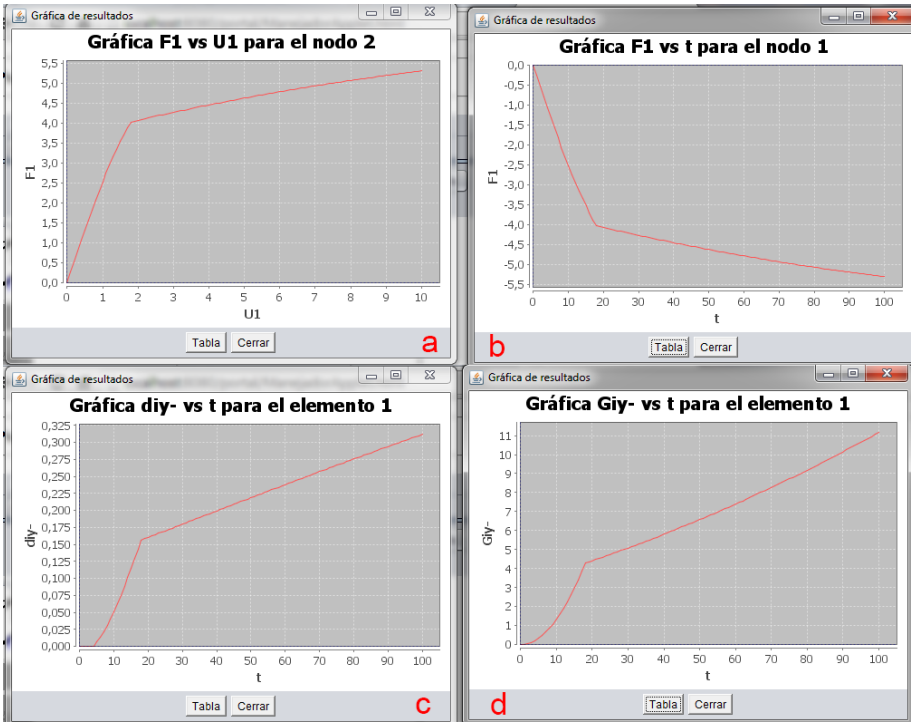

Figura 4.10: Resultados gráficos del portal

- Rotación en X - R1
- Rotación en Y - R2
- Rotación en Z - R3
- Fuerza en X - F1
- Fuerza en Y - F2
- Fuerza en Z - F3
- Momento en X - M1
- Momento en Y - M2
- Momento en Z - M3

Por otra parte si se desea graficar para Elementos aparecen los valores correspondientes a:

- $\phi iy - \phi jy$= son las rotaciones por flexión de la tangente del elemento en los extremos con respecto a la cuerda del plano.
- $\phi iz - \phi jz$= son las rotaciones por flexión en el plano XY
- δ= representa el alargamiento axial.
- ϕx= rotación torsional.
- $Miy - Mjy$= representan los momentos flectores en los extremos i y j del elemento en el plano XZ.
- N= es la carga axial del elemento.
- $Miz - Mjz$= representan los momentos flectores en i y j en el plano XY.
- Mx= es el torque.
- $\lambda i - \lambda j$= son multiplicadores plásticos.
- $diy^+ - djy^+ - diz^+ - djz^+$= representan el agrietamiento debido a acciones positivas.
- $diy^- - djy^- - diz^- - djz^-$= representan el agrietamiento debido a acciones negativas.
- $\Phi iyp - \Phi izp$= deformaciones plásticas para la rotula i.
- $\Phi jyp - \Phi jzp$= deformaciones plásticas para la rotula j.
- $Giy^+ - Gjy^+ - Giz^+ - Gjz^+ - Giy^- - Gjy^- - Giz^- - Gjz^-$= representan la tasa de disipación de energía para cada rotula del miembro de pórtico.

Como se observa Figura 4.10, la gráfica a, representa la relación entre el desplazamiento en X y la fuerza en X para el nodo 2. La gráfica b, representa la relación entre el tiempo t y la fuerza en X para el nodo 1. La gráfica c, representa la relación entre el tiempo t y daño en la dirección negativa del eje Y local para el elemento 1. La gráfica d, representa la disipación de energia en dirección negativa del eje Y local a medida que transcurre el tiempo t para el elemento 1.

Mapa de Daño: En esta segunda opción se puede ver gráficamente y numéricamente el daño de la estructura en función del tiempo, Figura 4.11. El tiempo se puede cambiar manualmente, o presionando el botón CORRER y donde el programa reproducirá el daño iteración a iteración.

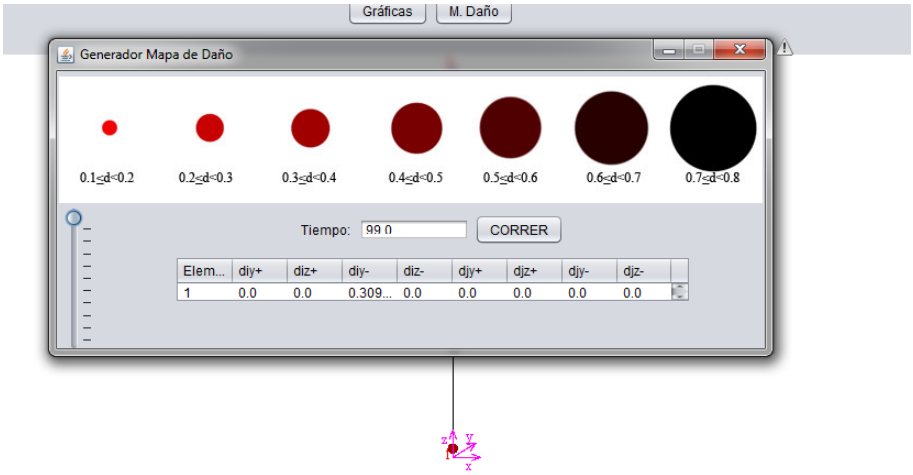

Figura 4.11: Mapa de Daño

4.2.1 Archivos generados por el PDP3D

Nombre_Proyecto.INP: El archivo es generado por el portal y contiene la información sobre la geometría de la estructura las propiedades de los elementos que la conforman y la información sobre los pasos de análisis para realizar la simulación. Es el archivo de entrada o archivo que alimenta el programa PEEF

Nombre_Proyecto.P3D El archivo es generado por el portal, el cual contiene la información y los datos que el usuario administra al portal, como por ejemplo, geometría de la estructura, secciones de los elementos, pasos de análisis para la simulación, materiales de los elementos, y más. Este archivo solo es entendido por el Portal.

Nombre_Proyecto.ERR En este archivo se guardan los errores por los cuales no pudo converger la simulación de la estructura. Este archivo es generado por PEEF

Nombre_Proyecto_E Elemento.txt Los archivos obtenidos por PEEF, al simular correctamente una estructura, contienen los resultados de cada una de las variables de estado para los elementos. Este archivo se refiriere exclusivamente a los resultados obtenidos en el elemento con identificador Elemento.

Nombre_Proyecto_E.txt Los archivos obtenidos por PEEF, al simular correctamente una estructura, contienen los resultados de cada una de las variables de estado para los elementos, Este archivo se refiriere exclusivamente a los resultados obtenidos en todos los elementos de la estructura. Este archivo es el cual se utiliza para visualizar los resultados en el Portal.

Nombre_Proyecto_N Nodo.txt Los archivos obtenidos por PEEF, al simular correctamente una estructura, contienen los resultados de cada una de las variables de estado para los nodos, Este archivo se refiriere exclusivamente a los resultados obtenidos en el nodo con identificador Nodo.

Nombre_Proyecto_N.txt Los archivos obtenidos por PEEF, al simular correctamente una estructura, contienen los resultados de cada una de las variables de estado para los nodos, Este archivo se refiriere exclusivamente a los resultados obtenidos en todos los nodos de la estructura. Este archivo es el cual se utiliza para visualizar los resultados en el Portal.

ArchivoINP.txt Este archivo contiene e indica a PEEF cual archivo con extensión INP debe simular.

RES Este archivo sin extensión es el resultado de ejecutar el programa generador, contiene la información sobre las propiedades de la sección en sus ejes locales.

RES_seccion Este archivo contiene la información sobre las propiedades de la sección en sus ejes locales en el extremo i de la sección.

RES_seccion_j Este archivo contiene la información sobre las propiedades de la sección en sus ejes locales en el extremo j de la sección.

seccion.txt El archivo que crea el portal con la información sobre la sección, la cual contiene información de los materiales y la distribución de los aceros. Este programa funciona como archivo de entrada para el programa generador

4.3 Pruebas y Depuraciones del Cliente PC

Luego de la programación del Cliente PC en una interfaz gráfica, sencilla, intuitiva y agradable para el usuario. Se llevó la ardua tarea de realizar numerosas pruebas del programa.

Esta serie de pruebas y depuraciones por las cuales pasó el PDP3D se resumen en 3 etapas fundamentales.

La primera etapa puede definirse como la prueba del post -procesador; recordando que el Post-proceso se refiere a la visualización de los resultados para ser entendidos e interpretados por el especialista.

En esta etapa se probó exhaustivamente el portal con diferentes modelos teóricos y reales. Estas pruebas se realizaron en conjunto con la Ingeniero Maylett Uzcátegui, siendo ella el primer usuario dentro del área en probar el Portal de Pórticos 3D.

Entre algunas de las pruebas realizadas se destaca la visualización de los resultados obtenidos de un Edificio pequeño en forma de L sometido ante un sismo en particular.

Simulación numérica de una estructura en "L" y dos niveles: En este caso la estructura se sometió varias veces a historias de aceleraciones en su base derivadas del sismo artificial Z5S2GB2 donde:

- Z5: Según el mapa de zonificación (Mérida, Municipio Libertador).
- S2: Forma espectral tipificada de acuerdo al terreno de fundación.
- GB2: Clasificación de la edificación según el uso.

Se realizaron un total de 6 simulaciones numéricas, en donde el registro de aceleraciones se multiplicó por una magnitud según la tabla 4.1, para las direcciones X y Y respectivamente:

En la Figura 4.12 se muestra el mapa de daño obtenido en cada simulación.

Tabla 4.1: Magnitud de la variable aceleración del registro de aceleraciones.

Simulación	Sismo en X	Sismo en Y
1	0.20g	0.10g
2	0.40g	0.20g
3	0.60g	0.30g
4	0.80g	0.40g
5	1.00g	0.50g
6	1.20g	0.60g

En la simulación 1 y 2 los valores de daño son muy pequeños, inferiores en la mayoría de los elementos a 0.20, el máximo ocurre en la viga 4 con un valor de daño de 0.24. También se puede observar que los daños en las columnas aparecen a partir de la simulación 5, sin embargo no se incrementan de forma considerable en la simulación 6, y se mantiene con valores inferiores a 0.40. Por el contrario, en las vigas 4, 6, 8 y 10 el daño aumenta con cada simulación, alcanzando valores entre 0.50 y 0.60 en la última simulación.

La segunda etapa para las pruebas y depuración del Cliente Pc incluyo probar y depurar el Preprocesador junto con el Procesador, de esta manera el objetivo de esta etapa era refinar la escritura del archivo INP, archivo que contiene la información del modelo. Estas Pruebas fueron realizadas por Lisbeth Maldonado y Diego Uzcátegui, para su proyecto de grado Maldonado L. (2012).

Los resultados obtenidos por parte de esta segunda etapa fueron los siguientes:

- En la primera parte se corrigieron errores en los archivos de extensión .INP que generaba el preprocesador, tales como, errores en la frecuencia, errores en la ubicación de las coordenadas de los nodos, entre otros, los cuales se comprobaban a través del programa ABAQUS CAE.

- En el menú de edición del grupo de secciones, no se podían deshacer los cambios a la sección, por ende una vez modificados, no se lograba regresar.

- Al eliminar algún elemento no era posible realizar el análisis de la estructura.

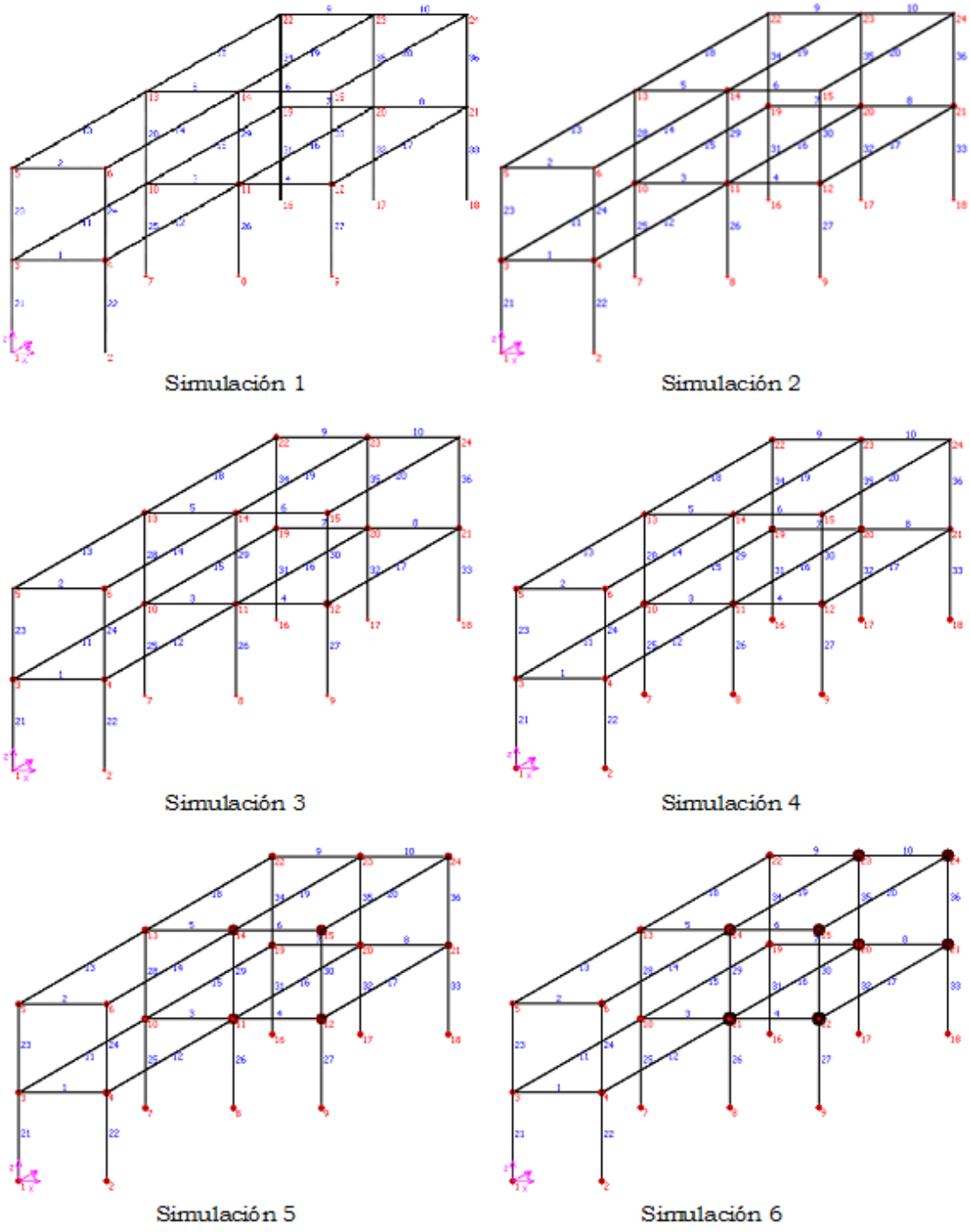

Figura 4.12: Mapa de daño al final de cada simulación

La tercera y última etapa para la depuración del PDP3D fue un trabajo en equipo junto a la Universidad Centroccidental "Lisandro Alvarado". Esta etapa se enmarcó en el uso y manejo del portal de una forma mucho más rigurosa para la obtención de posibles errores en el PDP3D.

Un aporte adicional a este trabajo fue la realización de un Manual de Usuario y un Tutorial (ver Apéndice D), a cargo de Gabriela Terán, para el correcto uso del PDP3D.

El trabajo realizado por Andrea Guzmán y Katherine Mejías, en su trabajo de Grado Guzmán Andrea (2014), consistía en probar múltiples modelos de estructuras ante diferentes solicitaciones. Los modelos que se usaron para el estudio y depuración del portal van desde modelos teóricos, hasta modelos de edificios reales, obteniendo como resultados más importantes los siguientes:

- Se encontró un error con el generador el cual se localiza en el módulo de secciones en la opción definir la sección, al momento de colocar el área de acero no quería generar los diagramas ni aceptar la cantidad de acero que se definía.
- Al momento de crear un paso donde se tuviera que cargar un registro sísmico (nombre.AMP) el programa no procesaba el archivo.
- Cuando se eliminaba algún paso para luego generar uno nuevo la pantalla emergente se colocaba en blanco (se colgaba) no era posible realizar el análisis de la estructura.
- Al momento de colocar el recubrimiento del acero longitudinal no permitía un recubrimiento menor a 3cm; ya que generaba un problema con el recubrimiento del acero transversal.

Dado que el proceso de prueba y depuración del Portal de Pórticos 3D fue realizado por diversas personas en diferentes momentos, evidencia la evolución, el mejoramiento y el fortalecimiento del mismo. Comenzando por una simple aplicación que mostraba solo resultados de forma gráfica hasta convertirse en un portal que permite a un usuario con cierto nivel de conocimiento poder realizar análisis de pórticos desde la comodidad de su casa.

4.4 Diagrama General de Clases

Como en toda aplicación de *software* desarrollada orientada a objeto, contiene un conjunto de clases enlazadas entre sí para englobar un producto final. En el caso del Portal De Pórticos 3D, en la Figura 4.13, se puede observar el diagrama de clases desarrolladas en JAVA para esta aplicación.

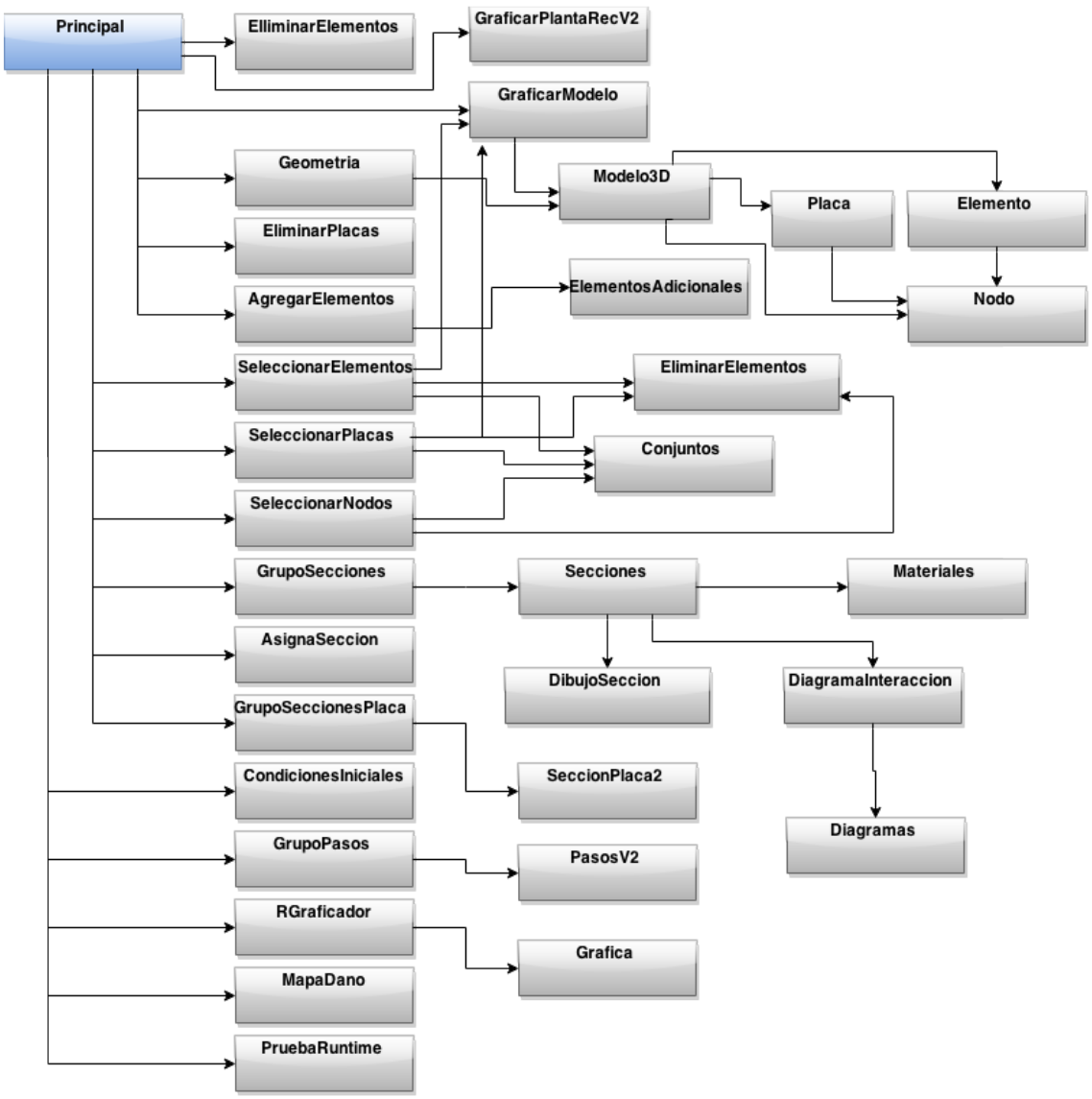

Figura 4.13: Diagrama de Clases para la Aplicación Portal de Pórticos 3D.

Como se observa el corazón de la aplicación es un Applet, con el nombre principal.java. Este Applet directamente funciona como clase principal conteniendo e interactuando con la mayoría de las clases del portal, es decir, sincronizando cada una de sus funciones. Esta clase principal.java hace llamados a un grupo específico de clases, como se ilustra en el diagrama de la Figura 4.14.

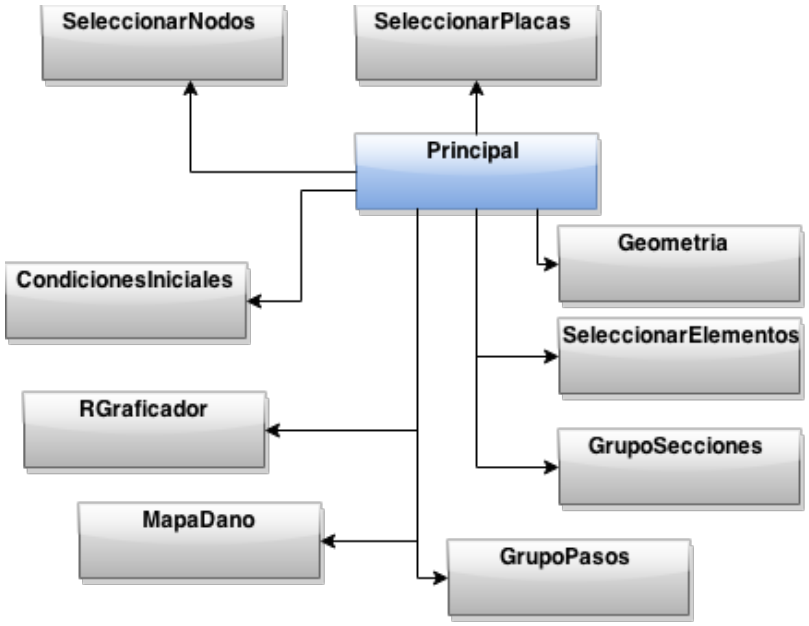

Figura 4.14: Diagrama de Clases para la Clase principal.java

Entre las clases, más importantes, que llama principal.java se encuentran:

Geometría: Esta clase se encarga directamente de contener agrupado los datos de nodos, elementos y placas que contiene el modelo estructural.

SeleccionarElementos: Clase cuyo objetivo es el de proporcionarle al usuario una interfaz para seleccionar los diferentes elementos, Viga o Columna, que compartan la misma sección. Estos elementos seleccionados resaltan en color rojo con respecto a los demás.

SeleccionarPlacas: En esta clase se despliega visualmente una lista de placas para ser seleccionadas por el usuario. Estas placas seleccionadas resaltan en un color amarillo para diferenciarlas del resto. Los diferentes conjuntos seleccionados donde almacenados en una clase llamada Conjunto.java

SeleccionarNodos: Clase con propósito muy similar al de SeleccionarElemento.java, pero que se diferencia en la información a almacenar. Esta clase almacena la información en un objeto de la clase Conjunto.java.

GrupoSecciones: Esta clase derivada de la JFrame permite al usuario gestionar las diferentes secciones de elementos con los cuales se diseña el pórtico.

CondicionesIniciales: Es una clase que contiene la interfaz, sencilla e intuitiva para la selección de condiciones de contorno, o condiciones iniciales para los nodos del portal, almacenando la información en un vector de tipo *boolean*.

GrupoPasos: Esta clase derivada de la JFrame permite al usuario gestionar las diferentes solicitaciones(Fuerza, Desplazamientos o Aceleraciones) a las cuales está sometida la estructura aporticada.

RGraficador: Clase encargada de proporcionarle al usuario las diferentes alternativas para visualizar los datos en un plano cartesiano XY. Esta clase contiene tres JComboBox, que sirven para la selección de los ejes y el tipo de gráfica que se desea visualizar.

MapaDanio: Clase derivada de JFrame, la cual contiene un serie de objetos como un JSlide, para ver el transcurso del tiempo, una JTable donde se registran los resultados para el daño del elemento y un panel con la leyenda para interpretar el resultado numérico.

Otra clase importante a la cual hay que hacer referencia es la clase GraficarModelo.java, clase cuyo objetivo principal es el de graficar o visualizar el pórtico en 3D. Esta clase incluye la clase Molelo3D.java, clase base encargada de almacenar en ArrayList la información de la estructura, utilizando objetos de tipo Elemento, Placa y Nodo. Estos objetos contienen las coordenadas de ubicación, y los atributos característicos de cada elemento físico. La relación entre ellas se representa en el diagrama de la Figura 4.15.

Las secciones de los elementos son determinista para el modelo de la estructura aporticada, por tal motivo la clase Secciones.java contiene y maneja la información

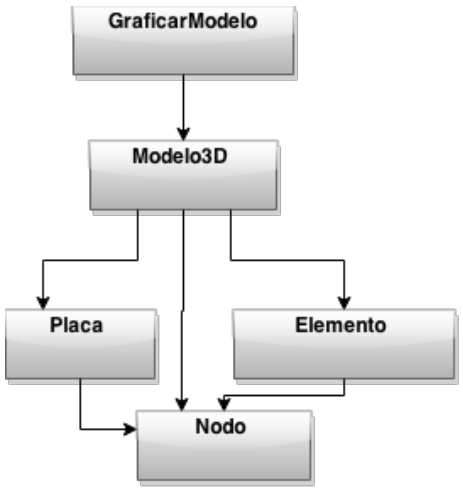

Figura 4.15: Diagrama de la Clase GraficarModelo.java y Modelo3D.java.

detallada para esta característica física. Esta Clase deriva de la JFrame y en ella se encuentran los atributos importantes de la sección almacenados en tres JTables, uno para el área de la sección, otro para la ubicación de los aceros longitudinales y otro que contiene la información de los estribos. Además esta clase realiza llamados a la Clase DibujoSeccion.java, DiagramaInteraccion.java y Materiales.java como se muestra en el diagrama de la Figura 4.16.

DibujoSeccion.java: Clase que deriva de un JPanel y cuyo propósito es graficar la sección, para darle un bosquejo al usuario de como resulta su diseño.

Materiales.java: Clase derivada de la JFrame, y que permite al usuario editar los parámetros en los materiales de construcción, concreto y acero, a utilizar en el modelo.

DiagramaInteraccion.java: Clase derivada de un JFrame, que contiene la estructura visual para contener un objeto grafico de tipo panel y mostrar al usuario a través de un JTable los datos del diagrama para una determinada sección. Esta clase llama la clase Diagrama.java, derivada de la clase Canvas.java, permite graficar el diagrama de interacción y ubicarlo en el objeto panelcentral de la clase DiagramaInteraccion.java

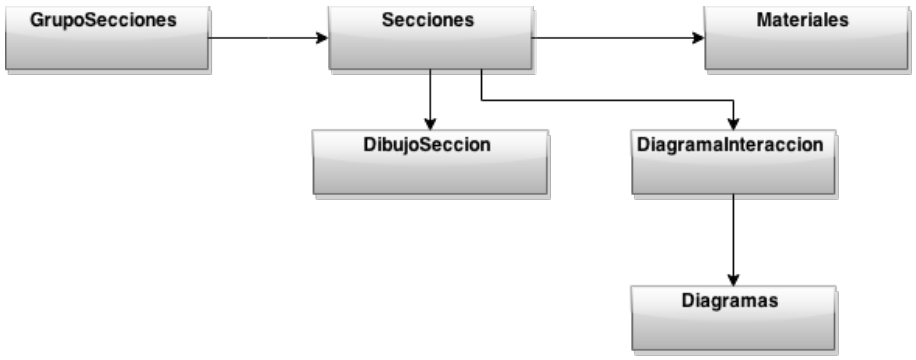

Figura 4.16: Diagrama de la Clase SeccionV2.java

La clase RGraficador.java encargada de proporcionar los controles al usuario presenta la peculiaridad que llama a la clase Grafica.java. Esta clase hace un llamado a librería JFreeChart que proporciona los métodos para la visualización de gráficas, ver Figura 4.17.

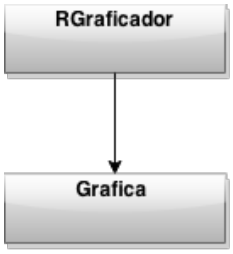

Figura 4.17: Diagrama de la Clase Rgraficador.java

La documentación más extendida y con mayor claridad queda expuesta en el Apéndice A del presente documento; en esta sección queda expuesta todo el desarrollo y funcionalidad del Cliente PC, como una aplicación Applet, alojada en un Navegador Web que realiza simulaciones de estructuras aporticadas desde cualquier computador con acceso a internet, sin atarse a una arquitectura física en particular.

Capítulo 5

Conclusiones

En la Universidad de Los Andes (ULA) desde los años 90 se ha desarrollado un programa de investigación en el área de simulación numérica, a partir de la creación de diferentes elementos finitos que permiten específicamente la evaluación de estructuras aporticadas de concreto armado. Actualmente cuentan con el programa de elementos finitos (PEEF), sin embargo su uso estaba limitado ya que no existía ninguna interfaz amigable para los usuarios.

Por tal motivo, en busca de un instrumento de apoyo, en esta tesis se desarrolló el *Portal De Pórticos 3D (PDP3D)*. El cual es un programa de elementos finitos con el que se realizan modelos estructurales de sistemas aporticados de concreto armado, para el análisis y evaluación del daño que puedan presentar las estructuras ante diferentes solicitaciones. Adicionalmente, se escribió un elemento finito que permite incorporar el efecto de las losas de entrepisos.

Para lograr este objetivo se diseñaron cinco (5) módulos que permiten definir la geometría y características del modelo, incorporar las solicitaciones, realizar el análisis y revisar los resultados obtenidos. Así mismo se incorporó en el programa PEEF la nueva subrutina (CSTDKT).

Para verificar el correcto funcionamiento del portal, se llevó a cabo la simulación numérica de un gran número de ejemplos correspondientes a ensayos experimentales encontrados en la literatura o de edificios reales. Estos resultados fueron comparados de manera exitosa con los obtenidos experimentalmente o mediante los programas:

Portal de Pórticos 2D y *Abaqus CAE*.

Uno de los aportes más importantes de esta tesis consiste en ofrecer un portal de elementos finitos de aplicación específica que puede ser accedido exclusivamente a través de la Web, por lo tanto permitirá un mayor número de usuarios de mediano y alto nivel, manteniendo el estándar tecnológico y científico. En tal sentido el programa Portal De Pórticos 3D cuenta con las siguientes características:

Libre acceso: Podrá ser accedido vía Web, desde la computadora del usuario.

Simple: Es un portal de aplicación específica con interfaces agradables al usuario.

Alta capacidad tecnológica: Las simulaciones numéricas no se realizarán en la computadora personal del usuario sino en un servidor de alto rendimiento.

Alta capacidad científica: Las teorías usadas en los elementos finitos se basan en la "Teoría del Daño Concentrado".

Adicionalmente, el Portal de Pórticos 3D cuenta tanto con un manual de usuario como un tutorial para su uso. Estos documentos han sido elaborados en conjunto con la Universidad Centroccidental Lisandro Alvarado (UCLA).

Durante el desarrollo de la tesis se suscitaron varios inconvenientes como la elección del manejador de usuarios. Estas plataformas se probaron, generando resultados desfavorables, en este caso, bien sea por la poca documentación encontrada o por la incompatibilidad con la versión en el caso del Módulo Rampart de Axis2. Para solventar el inconveniente se desarrolló un sistema de acceso sencillo y básico, implementado en MYSQL y PHP, que en principio funciona correctamente, mas deja vulnerable el servicio Web.

Sin embargo en una posterior oportunidad se recomienda fortalecer la seguridad del servicio aplicando protocolos de seguridad tracidicionales como SSL, TLS y HTTPS que cifran los mensajes transferidos de entre dos puntos o protocolos más avanzados como WebServices Security que proporciona protección a nivel de mensajes entre dos extremos.

Como parte adicional y tratando de incluir el PD3D a la tendencia actual esta tesis impulsa la posterior creación de una serie de aplicaciones APP para acceder al

servicio desde dispositivos móviles y tener una herramienta para el análisis de pórticos al alcance de la mano.

Bibliografía

3DS Dassault Systemes (2012). *Abaqus Overview*. http://www.3ds.com/.

ANSYS INC. (2012). *ANSYS Software Products*. http://www.ansys.com/.

Batoz, J. L. (1980). A study of three-noded triangular plate bending elements. *International Journal for Numerical Methods in Engineering*, 15:1771–1812.

Celigüeta Lizarza, J. T. (2008). *Método de los Elementos Finitos para Análisis Estructural*. Navarra, España.

Chaves Eduardo, M. R. (2010). *Mecánica computacional en la ingeniería con aplicaciones en MATLAB*. Ciudad Real, España.

Comer, D. E. (1995). *Redes globales de información con internet y TCP/IP*. Pearson Education, México.

Computers and Structures, INC. (2012a). *ETABS Overview*. http://www.csiberkeley.com/etabs.

Computers and Structures, INC. (2012b). *SAP2000 Overview*. http://www.csiberkeley.com/sap2000/.

Flórez López, J. (1999). *Plasticidad y Fractura en Estructuras Aporticadas*. Monografía CIMNE IS-35 Universidad Politécnica de Madrid. Madrid, España.

Guzmán Andrea, M. K. (2014). *DEPURACIÓN Y VALIDACIÓN DEL PROGRAMA DE ANÁLISIS ESTRUCTURAL PEEF*. Barquisimeto,Venezuela.

Haklay, M. (2012). *Ciencia Ciudadana*. http://blogs.casa.ucl.ac.uk/?s=citizen+science.

Ibarra A., S. (2012). *Definición de Cliente Servidor*. Universidad de Colima, México, urlhttp://docente.ucol.mx/sadanary/public_html/bd/.

Maldonado L., U. D. (2012). *DESARROLLO DE UNA HERRAMIENTA COMPUTACIONAL, CON INTERFAZ WEB, PARA REALIZAR ANÁLISIS EN TRES DIMENSIONES DE LA VULNERABILIDAD SÍSMICA DE ESTRUCTURAS CIVILES*. Mérida,Venezuela.

Márquez Avendaño, Bertha Mariel; Zulaica Rugarcía, J. M. (2004). Implementación de un reconocedor de voz gratuito a el sistema de ayuda a invidentes dos-vox en español. *Universidad de las Américas Puebla*.

MSC Software Corporation (2012). *MSC Nastran for FEA*. http://www.mscsoftware.com/Contents/Products/CAE-Tools/MSC-Nastran.aspx.

PRISA, G. (2012). *Gráfico: Modelo cliente-servidor para el intercambio de información.* http://www.kalipedia.com/tecnologia/tema/comunicaciones/graficos-modelo-cliente-servidor.html?x1=20070821klpinginf_30.Ees&x=20070821klpinginf_71.Kes.

Saffirio, M. (2006). *¿Qué son los Web Services?* http://msaffirio.wordpress.com/2006/02/05/¿que-son-los-web-services/.

Taboada G., J. A. (2009). *ANÁLISIS Y DISEÑO DE EDIFICIOS ASISTIDO POR COMPUTADORAS*. Tesis pregrado, PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ, Lima,Perú.

TERÁN TARAZONA, G. I. (2014). *ELABORACION DEL MANUAL DE TEORIA Y DEL MANUAL DE USUARIO DEL PROGRAMA DE ANALISIS DE ESTRUCTURA PEEF VERSION 1.0*. Barquisimeto,Venezuela.

The Apache Software Foundation (2012a). *About the Apache HTTP Server Project*. http://httpd.apache.org/ABOUT_APACHE.html.

The Apache Software Foundation (2012b). *Apache Axis2*. http://axis.apache.org/axis2/java/core/index.html.

The Apache Software Foundation (2012c). *Apache Tomcat*. http://tomcat.apache.org/.

The World Wide Web Consortium (W3C) (2012). *Web Services Glossary*. http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/.

Ubuntu (2012). *The Projec Ubuntu*. http://www.ubuntu.com/project.

Universidad de Los Andes (1990). *Portal De Pórticos*. http://www.portaldeporticos.ula.ve/.

University at Buffalo (1987). *IDARC 2D*. http://civil.eng.buffalo.edu/idarc2d50/.

University of California (1993). *The Earthquake Engineering*. http://nisee.berkeley.edu/elibrary/Software/DRAIN2DXZIP.

Uzcátegui Flores, M. Y. (2007). *Un portal de cálculo para la simulación numérica del colapso de estructuras de concreto armado*. IEEE 5º Congreso Internacional en Innovación y Desarrollo Tecnológico, Cuernavaca, Morelos, México.

Uzcátegui Flores, M. Y. (2012). *DESARROLLO DE UN PROGRAMA DE ELEMENTOS FINITOS TRIDIMENSIONAL BASADO EN LA WEB*. Tesis doctoral, UNIVERSIDAD DE LOS ANDES, Mérida,Venezuela.

WIKIPEDIA (2012a). *Client–server model*. http://en.wikipedia.org/wiki/Client–server_model.

WIKIPEDIA (2012b). *Servicio Web*. http://es.wikipedia.org/wiki/Servicio_web/.

WIKIPEDIA (2012c). *Web Services*. http://en.wikipedia.org/wiki/Web_service/.

Zienkiewicz O. C., T. R. L. (1994). *El Método de los Elementos Finitos*. McGraw-Hill.

Apéndice A

Clases del Portal

En el siguiente apéndice se documenta el conjunto de clases utilizada para la implentación del Portal De Pórticos 3D **PDP3D**

A.1 Principal.java

El portal de Porticos 3D poseé una clase principal llamada **Principal.java** la cual se encarga de las funciones más comunes dentro de una aplicación con ventanas. En la Figura A.1 se ilustran los objetos y métodos de la clase.

init(): Es el constructor de la clase principal, este método se encarga de asignarle memoria a cada objeto contenido en la clase, aparte de asignarle a cada ícono una imagen para representarlo. La clase init() funciona como un Macro de la interfaz la cual contiene las funciones más importantes de la aplicaciones que manejan archivos. Estas funciones son Archivo Nuevo, Abrir Archivo, Guardar Archivo y Cerrar Archivo. Además de contener el menú de herramientas y el resto de clases que permiten modelar la estructura tridimensional.

asignarSeleccionElementos(): Como su nombre lo indica, esta función asigna los elementos (Viga y Columna) seleccionados al conjunto elegido por el usuario. Este método es un procedimiento que no retorna ningún valor.

asignarSeleccionNodos(): Este método asigna los nodos seleccionados al conjunto elegido por el usuario. Este método es un procedimiento que no retorna ningún

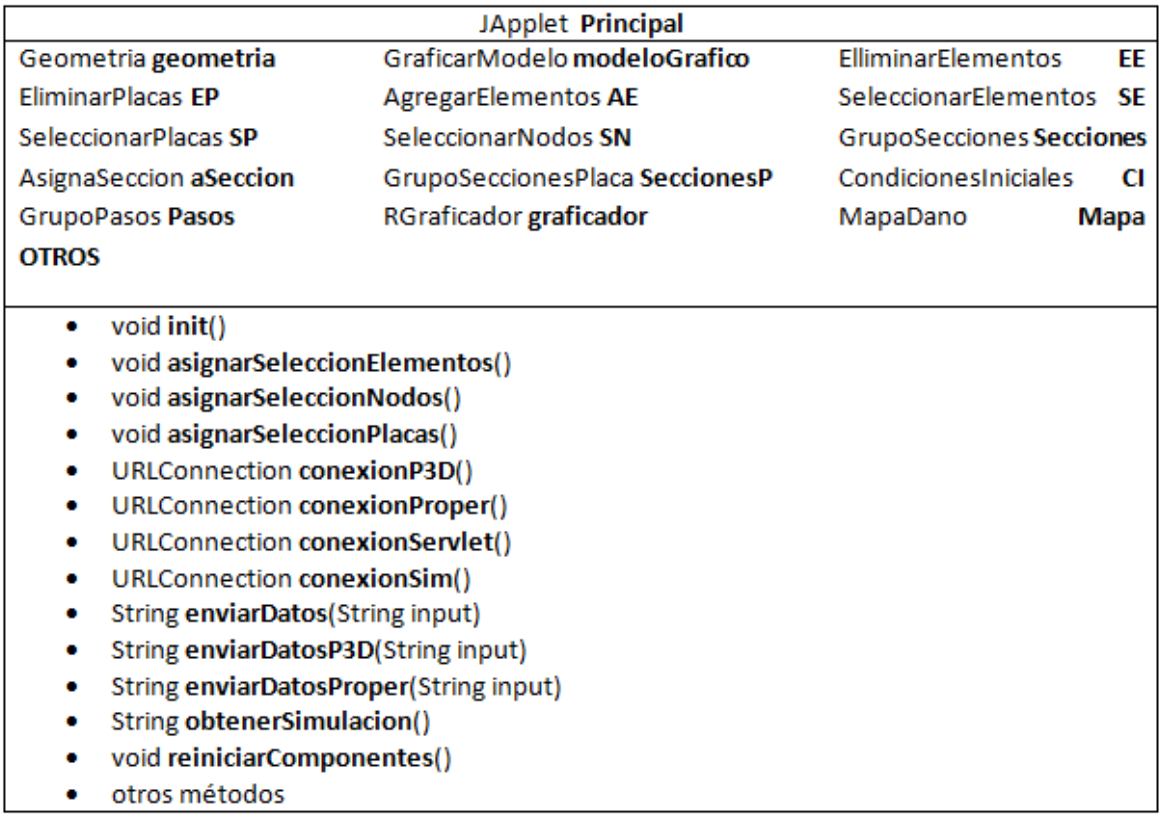

Figura A.1: Clase Principal.java

valor.

asignarSeleccionPlacas(): El objetivo de este método es asignar los elementos tipo placa seleccionados al conjunto elegido por el usuario. Este método es un procedimiento que no retorna ningún valor.

conexionP3D(): Este método devuelve el objeto URLConnection que permite la conexión con el Servlet para realizar la escritura del archivo .P3D en el servidor.

conexionProper(): El método devuelve el objeto URLConnection que permite la conexión con el Servlet para realizar la escritura del archivo PROPERTY.txt en el servidor.

conexionServlet(): Este método devuelve el objeto URLConnection que permite la conexión con el Servlet para realizar la escritura del archivo .INP y el archivo ARCHIVOINP.txt en el servidor.

conexionSim(): Este método devuelve el objeto URLConnection que permite

la conexión con el Servlet para realizar la ejecucion del ejecutable ProgrmaPrincipal (PEEF) en el servidor.

enviarDatos(String input): Este método permite la conexión directa con el Servlet que escribe el archivo .inp en el servidor, se envía el archivo serializado en una cadena de texto, la cual es introducida por parámetro de la función, y se envía esta cadena al servidor, este la decodifica y la escribe como archivo de texto .inp.

enviarDatosP3D(String input): Este método permite la conexión directa con el Servlet que escribe el archivo .p3d en el servidor, se envía el archivo serializado en una cadena de texto, la cual es introducida por parámetro de la función, y se envía esta cadena al servidor, este la decodifica y la escribe como archivo de texto .p3d

enviarDatosProper(String input): Este método permite la conexión directa con el Servlet que escribe el archivo PROPERTY.txt en el servidor, se envía la orden en una cadena de texto, para la construcción del archivo de texto, y se envía esta orden al servidor, este Servlet la interpreta y construye el de texto.

obtenerSimulacion(): Este método permite la conexión directa con el Servlet que envia el archivo de resultados serializado en el servidor, se envía la orden en una cadena de texto, para la interpretación del archivo de texto en el cliente, y se guarda en memoria los resultados.

reiniciarComponentes(): Este método asigna nueva dirección de memoria a los objetos de la clase.

otros metodos: Entre los otros métodos de la clase se incluyen los eventos correspondientes a cada uno de los bótenes contenidos en la interfaz.

A.2 AgregarElementos.java

La clase AgregarElementos permite adicionar elementos a la geometría del pórtico. En la Figura A.2

AgregarElementos(int nnodos): Es el constructor paramétrico que recibe el numero de nodos.

setNNodos(int n): Asigna memoria al objeto EA (Ver Figura A.2) y agrega este objeto a esta clase.

JFrame **AgregarElementos**
ElementosAdicionales **EA**
• AgregarElementos(int nnodos) • setNNodos(int n)

Figura A.2: Clase AgregarElementos.java

A.3 AsignaSeccion.java

Es la clase que se encarga de enlazar los conjuntos de elementos definidos por el usuario con las secciones diseñadas. En la Figura A.3 se observa los métodos y atributos más reprensentativos de esta clase.

JFrame **AsignaSeccion**
JTable **Elementos** JTable **Secciones** JTable **Tabla** Entero **nEle**
• AsignaSeccion() • AsignaSeccion(JTable TSec, JTable TEle) • AsignarDatos(JTable TSec, JTable TEle,JTable TASec) • Rotular(String nombre,JPanel P) • SeccionXElemento(int n)

Figura A.3: Clase AsignaSeccion.java

AsignaSeccion(): Es el constructor por defecto, solo se encarga de asignar memoria a los objetos.

AsignaSeccion(JTable TSec, JTable TEle): Constructor que asigna memoria a los datos y configura la ventana para la introducción de los datos.

AsignarDatos(JTable TSec, JTable TEle,JTable TASec): Es la rutina que tiene como función enlazar las secciones con los conjuntos de elementos. Recibe 3 JTable para su funcionamiento. La información suministrada por el usuario es almacenada en TASec con la información aportada por TSec y TELe.

Rotular(String nombre,JPanel P): Es una función que se encarga de darle formato al panel que recibe como parámetro, esta rutina solo recibe el nombre del panel y el panel a rotular.

SeccionXElemento(int n): esta función se encarga de generar una tabla con todos los elemento y asignarle a cada elemento la sección asignada en cada extremo, en la primera columna guarda el numero de elemento en la segunda la sección al nodo i, y la tercera al nodo j. Para el correcto funcionamiento la rutina necesita el número de elementos presentes en el modelo.

A.4 CondicionesIniciales.java

Es una clase que permite al usuario definir condiciones iníciales para el modelo al usuario. Al final de rutina se devuelve un vector booleano con la información sobre las restricciones. Los objetos más importantes de la clase se muestran en la Figura A.4.

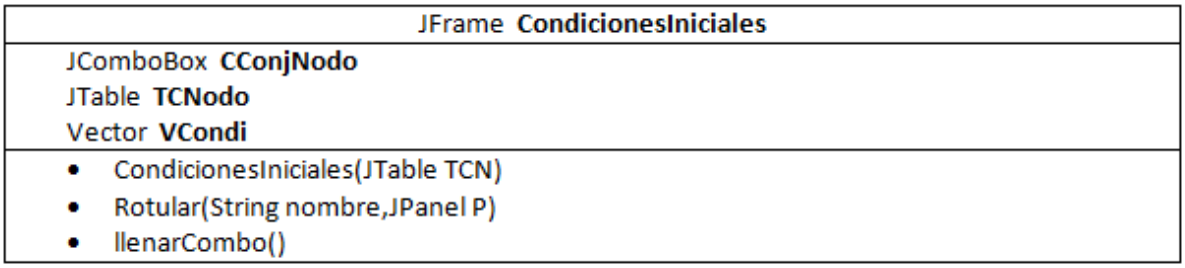

Figura A.4: Clase CondicionesIniciales.java

CondicionesIniciales(JTable TCN): Es el constructor de la clase, este método se encarga de asignarle memoria a los objetos presentes en la clase, así como de la configuración de la ventana de visualización, es necesario para el funcionamiento una tabla que contenga el conjunto de nodos definido por el usuario.

Rotular(String nombre,JPanel P): Es una función que se encarga de darle formato al panel que recibe como parámetro, esta rutina solo recibe el nombre del panel y el panel a rotular.

llenarCombo(): Esta rutina permite llenar el objeto CConjNodo, que presenta la ventana, con el conjunto de nodos definidos por el usuario.

A.5 Conjuntos.java

Es una clase que permite agrupar elementos que comparten características comunes. En la Figura A.5 se muestran los objetos más importantes de la clase.

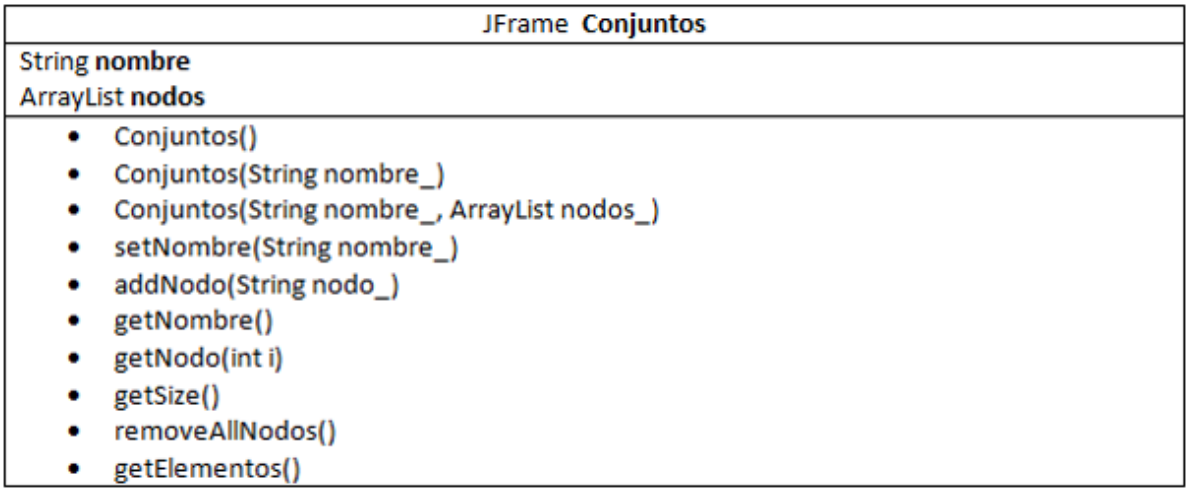

Figura A.5: Clase Conjuntos.java

Conjuntos(): Es el constructor por defecto de la clase, asigna memoria a los objetos de ella.

Conjuntos(String nombre_): Es el constructor paramétrico de la clase, asigna el nombre que recibe por el parámetro.

Conjuntos(String nombre_, ArrayList nodos_): Es el constructor paramétrico de la clase, asigna el nombre que recibe por el parámetro, y la lista de elementos.

setNombre(String nombre_): Asigna el nombre del conjunto

getNombre(): Obtiene el nombre del conjunto

getNodo(int i): Obtiene el nodo en la posición i

getSize(): Obtiene el tamaño de la lista

removeAllNodos(): Remueve todos los nodos de la lista

A.6 DiagramaInteraccion.java

Esta clase funciona como un contenedor de los objetos para visualizar el diagrama de interacción para las secciones de los elementos. En la Figura A.6 se puede observar los atributos y métodos más representativos de la clase.

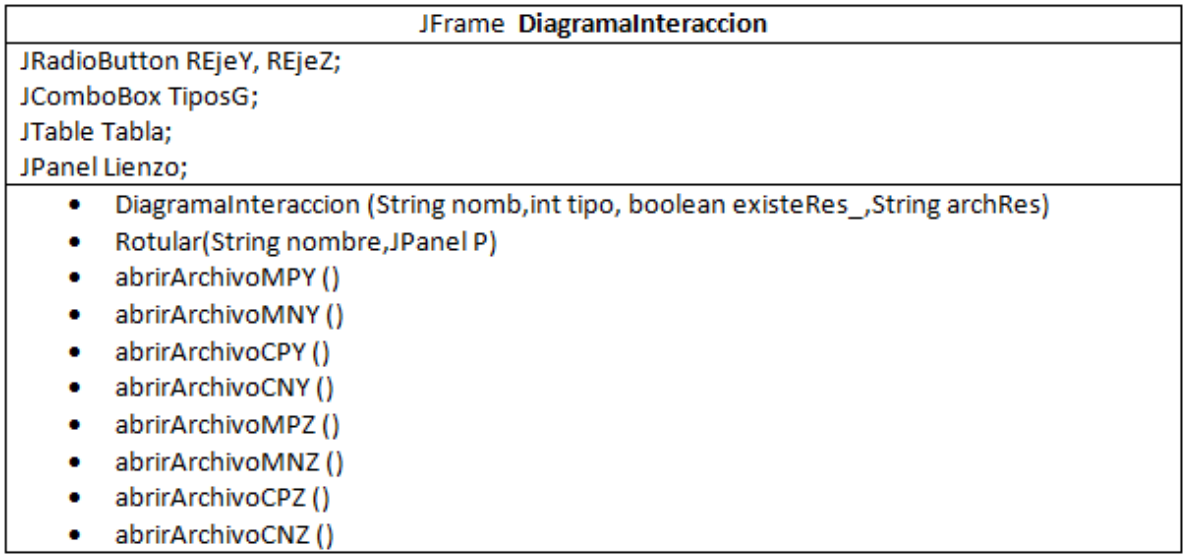

Figura A.6: Clase DiagramaInteraccion.java

DiagramaInteraccion(String nomb,int tipo, boolean existeRes_,String archRes): Este método es el constructor paramétrico el cual se encarga de asignarle memoria a los objetos de la clase. Recibe el nombre de la sección, el tipo y el

Rotular(String nombre,JPanel P): Este método básicamente le asigna nombre al panel que recibe por parámetro para identificarlo de manera gráfica.

abrirArchivoMPY(): Este procedimiento permite descomponer la cadena de texto, la cual contiene la información de las propiedades de la sección y grafica los momentos correspondientes (Momento plástico, momento crítico, momento ultimo) al eje y local en la dirección positiva.

abrirArchivoMNY(): Este procedimiento permite descomponer la cadena de texto, la cual contiene la información de las propiedades de la sección y grafica los momentos correspondientes (Momento plástico, momento crítico, momento ultimo) al eje y local en la dirección negativa.

abrirArchivoMPZ(): Este procedimiento permite descomponer la cadena de texto, la cual contiene la información de las propiedades de la sección y grafica los momentos correspondientes (Momento plástico, momento crítico, momento ultimo) al eje z local en la dirección positiva.

abrirArchivoMNZ(): Este procedimiento permite descomponer la cadena de texto, la cual contiene la información de las propiedades de la sección y grafica los momentos correspondientes (Momento plástico, momento crítico, momento ultimo) al eje z local en la dirección negativa.

abrirArchivoCPY(): Este procedimiento permite descomponer la cadena de texto, la cual contiene la información de las propiedades de la sección y grafica la curvatura correspondiente al eje y local en la dirección positiva.

abrirArchivoCNY(): Este procedimiento permite descomponer la cadena de texto, la cual contiene la información de las propiedades de la sección y grafica la curvatura correspondiente al eje y local en la dirección negativa. abrirArchivoCPZ():Este procedimiento permite descomponer la cadena de texto, la cual contiene la información de las propiedades de la sección y grafica la curvatura correspondiente al eje z local en la dirección positiva.

abrirArchivoCNZ(): Este procedimiento permite descomponer la cadena de texto, la cual contiene la información de las propiedades de la sección y grafica la curvatura correspondiente al eje z local en la dirección negativa.

A.7 Diagramas.java

Esta clase cumple con el propósito de dibujar el grafico tanto de los momentos como de la curvatura en la sección correspondiente. Los atributos y métodos se visualizan en la Figura A.7.

Diagramas(JTable Datos): Constructor paramétrico el cual recibe una tabla de datos, para dibujar sobre el cambas el diagrama.

paint(Graphics g): Método que se encarga de dibujar sobre el objeto g:Graphics y asignarlo al canva.

Canvas **Diagramas**
JTable **TablaDatos**
• Diagramas(JTable Datos) • paint(Graphics g)

Figura A.7: Clase Diagramas.java

A.8 DibujoSeccion.java

Es una clase que contiene la rutina para visualizar el diseño de la sección. Esta clase es llamada dentro de la clase Secciones.java, la salida de esta clase es un gráfico plano de la sección transversal donde se visualiza el corte de la sección. Los principales objetos que maneja la clase se observan en la Figura A.8.

Canvas **DibujoSeccion**
JTable DatosA JTable DatosS Entero eje String Vect
• DibujoSeccion() • DibujoSeccion(JTable TS,JTable TA, int e, String V) • AsignaVector() • DibujarSeccion (Graphics g) • dibujarFlecha(Graphics g, Point P0, Point Pf) • paint(Graphics g)

Figura A.8: Clase DibujoSeccion.java

DibujoSeccion(): Es el constructor por defecto que asigna memoria a los objetos de la clase.

DibujoSeccion(JTable TS,JTable TA, int e, String V): Es el constructor paramétrico que asigna valores del argumento a los correspondientes objetos de la clase.

AsignaVector(): Descompone la cadena Vect, que contiene la información del vector dirección, en sus componentes vx, vy, vz.

DibujarSeccion (Graphics g): Funciona como el método principal de la clase,

recibe de parámetro un tipo gráfico, y la función se encarga de construir una sección plana rectangular con la información que obtuvo del usuario. Así como se puede visualizar la cantidad de aceros suministrada por el usuario.

dibujarFlecha(Graphics g, Point P0, Point Pf): Esta rutina recibe un tipo gráfico y dos tipo Point. El objetivo de este procedimiento es dibujar en el gráfico una flecha entre el punto P0 y el punto Pf.

paint(Graphics g): Es la función ,en java, que le indica al sistema que debe dibujar. Esta función llama a DibujarSeccion (Graphics g)

A.9 Elemento.java

La clase Elemento modela o contiene las características principales de un elemento viga, columna para el modelo del pórtico. En la Figura A.9 se observan los atributos y métodos mas reprentativos de la Clase.

Elemento
Nodo **nodoi, nodoj** int **identificador** double **longitud** boolean **visible** int **tipo** String **Si,Sj**
• **Elemento**(int ind, Nodo i, Nodo j,int tip) • Nodo **getNodoi**() • Nodo **getNodoj**() • double **getlongitud**() • int **getindicador**() • boolean **isVisible**() • boolean **setNodoi**(Nodo i) • boolean **setNodoj**(Nodo j) • void **setIdentificador**(int i) • boolean **setVisible**(boolean t) • int **getTipo**() • void **setSi**(String Si_) • void **setSj**(String Sj_)

Figura A.9: Clase Elemento.java

getNodoi(): el método obtiene el nodo asignado al extremo i del elemento

getNodoj(): el método obtiene el nodo asignado al extremo j del elemento

getlongitud(): el método obtiene la longitud del elemento y la devuelve de tipo doublé

isVisible(): Devuelve si el elemento es visible o no

getIdentificador(): Devuelve el identificador del elemento

setNodoi(Nodo i): Asigna un objeto nodo al elemento en el extremo i

setNodoj(Nodo j): Asigna un objeto nodo al elemento en el extremo j

setIdentificador(int i): Asigna un el identificador i al elemento en cuenstion

setVisible(boolean t): Asigna TRUE o FALSE, para configurar la visialización del elemento

getTipo(): obtiene el tipo de seccion (1: Columna, 2: Viga X, 3: Viga Y, 4: Otro)

setSi(String Si_): Asigna el nombre de la seccion al extremo I del elemento

setSj(String Sj_): Asigna el nombre de la seccion al extremo j del elemento

A.10 ElementosAdicionales.java

Esta clase contiene los métodos necesarios para agregar elementos adicionales entre 2 nodos existentes. En la Figura A.10 se visualiza los atributos y métodos de la Clase.

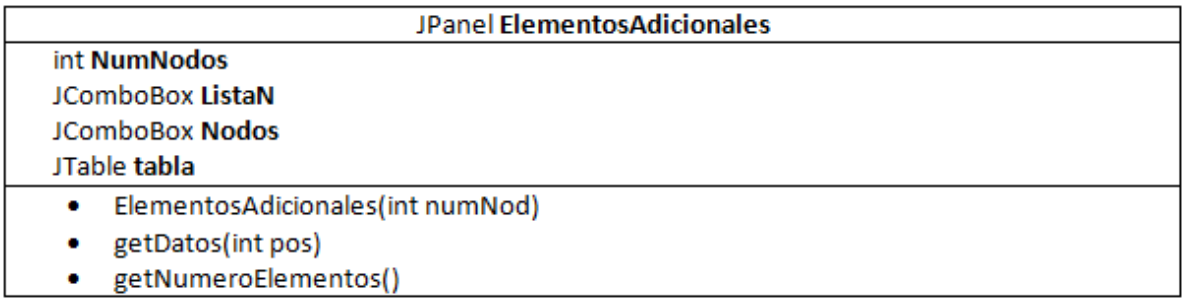

Figura A.10: Clase ElementosAdicionales.java

ElementosAdicionales(int numNod): Es el constructor de la clase, el cual recibe como parámetro el número de nodos del modelo. Este constructor asigna memoria a cada uno de los atributos de la clase.

getDatos(int pos): Este método devuelve un vector con los nodos del elemento adicional.

getNumeroElementos(): Retorna el total de elementos que constituyen el modelo del pórtico.

A.11 EliminarElemento.java

Es una clase que se encarga de manejar los elementos para facilitar su selección. Esta clase contiene los siguientes objetos visibles en la Figura A.11.

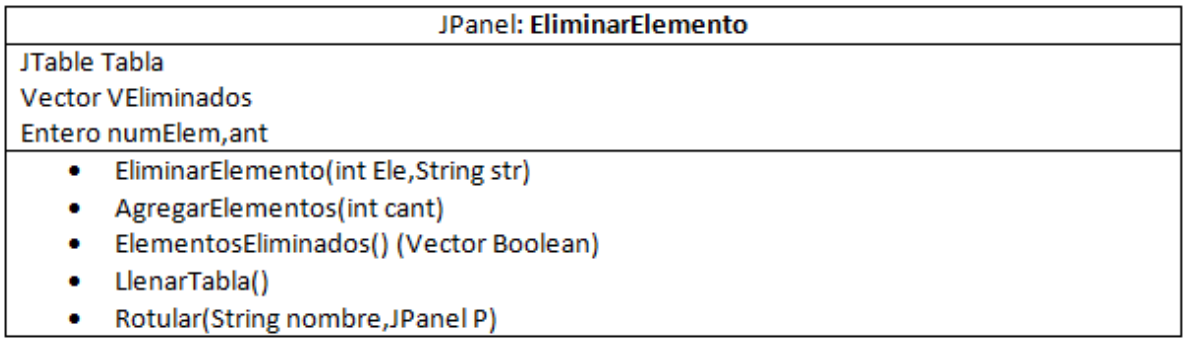

Figura A.11: Clase EliminarElemento.java

EliminarElemento(int Ele,String str): Es el constructor el cual se encarga de asignar memoria a cada objeto de la clase. Recibe como parámetros el número de elementos y una cadena con la palabara "Elementos"

AgregarElementos(int cant): Esta función agrega los elementos adicionales a la tabla de elementos ya creada.

ElementosEliminados(): Es una rutina que devuelve un vector booleano del tamaño igual al número de elementos. Este vector contiene marcado como verdadero los elementos que han sido eliminados por el usuario.

LlenarTabla(): Esta rutina se encarga de marcar como falso todas las casillas de la tabla de los elementos.

Rotular(String nombre,JPanel P): Es una función que se encarga de darle formato al panel que recibe como parámetro, esta rutina solo recibe el nombre del

panel y el panel a rotular.

A.12 EliminarPlacas.java

Es una clase que se encarga de manejar los elementos tipo placa para facilitar su selección. Esta clase contiene los siguientes objetos mostrados en la Figura A.12.

JFrame: **EliminarPlacas**
Modelo3D **modeloEstrucutura** EliminarElemento **pEliminarE**
• EliminarPlacas(Modelo3D mod) • reenumerarModelo(Modelo3D modeloE) • setModelo(Modelo3D mod)

Figura A.12: Clase EliminarPlacas.java

EliminarPlacas(Modelo3D mod): Es el constructor el cual se encarga de asignar memoria a cada atributo de la clase. Recibe como parámetros un objeto de tipo Modelo3D para vincularse con el Modelo graficado.

reenumerarModelo(Modelo3D modeloE): Este método reestructura o reenumera el modelo del pórtico tras eliminar elementos.

setModelo (Modelo3D mod): Asigna al atributo de la clase el objeto de tipo Modelo3D recibido por parámetro.

A.13 Geometria.java

Es una clase que se encarga de capturar los datos de la geometría del modelo para construir el cubo, esta aplicación es una ventana que contiene tres tablas relacionadas con cada dimensión. Los atrributos y métodos se pueden observar en la Figura A.13.

Geometria(String name): Es el constructor de la clase el cual se encarga de asignar memoria a cada objeto contenido en ella. Recibe como parámetro el nombre del archivo y devuelve los datos del modelo luego de capturarlos por la pantalla.

Rotular(String nombre,JPanel P): Es una función que se encarga de darle formato al panel que recibe como parámetro, esta rutina solo recibe el nombre del

JFrame: **Geometria**
JTable **modeloZ, modeloX, modeloY** Modelo3D **modelo**
• Geometria(String name) • Rotular(String nombre,JPanel P) • construirModelo(javax.swing.JTable TablaZ, javax.swingJTable TablaX, javax.swing.JTable TablaY) • Modelo3D getModelo() • reenumNodos() • setModelo(ArrayList<Nodo> listanodos, ArrayList<Elemento> listaelementos,ArrayList<Placa> listaPlacas, String name) • tipoElemento(Nodo ni, Nodo nj)

Figura A.13: Clase Geometria.java

panel y el panel a rotular.

A.14 Grafica1.java

Es una clase que contiene los elementos necesarios, atributos y métodos, para realizar una gráfica sobre una ventana a partir de un conjunto de datos de entrada, x y y. En la Figura A.14 se puede visualizar los métodos y atributos de la Clase.

JFrame: **Grafica1**
• Vector **EjeX, EjeY** • String **Nombre** • **Int x0,xN,y0,yN, NE, ele, ejex, ejey** • JFrame **Ventana, VTabla**
• Grafica(int NE_, int ele_, int ejex_,int ejey_) • AsignaParam(int NE_,int ele_,int ejex_,int ejey_) • abrirArchivo () • paint(Graphics g)

Figura A.14: Clase Grafica1.java

Grafica(int NE_, int ele_, int ejex_,int ejey_): es el constructor por defecto encargado de asignarle memoria a cada objeto de la clase. Así de configurar la ventana

en la que se visualizará la gráfica.

AsignaParam(int NE_,int ele_,int ejex_,int ejey_): esta función tiene como objetivo asignarle los parámetros esenciales para la construcción de los resultados en forma de gráfica. Estos parámetros son pasados a los objetos para llamar a la función que permite visualizar.

abrirArchivo (): es una rutina que permite abrir el archivo de texto correspondiente a la solicitación hecha desde el manejador de gráficas.

paint(Graphics g): Es la función en java, encargada de indicarle al sistema lo que debe dibujar. Esta función muestra por pantalla la relación entre los el conjunto de datos de manera visual.

A.15 GraficarModelo.java

Clase cuya función principal es contener los métodos para dibujar pórticos en 3 dimensiones, esta clase permite visualizar y manejar las características del pórtico en 3D. Los elementos más importantes de la clase se m+observan en la Figura A.15.

GraficarModelo(): Constructor por defecto, este método se encarga de asignarle memoria a los atributos de la clase una vez creado el objeto.

GraficarModelo (Modelo3D modelo_): Constructor paramétrico de la clase, encargado de asignar memoria a los objetos de la clase. Recibe un objeto de tipo Modelo3D el cual contiene cada dimensión del pórtico, la ubicación de los nodos, distribución de los elementos de construcción viga, columna y placa.

AjustarFactorEscala(int anchoPantalla,int altoPantalla): Esta función ajusta la es factor de escala para representar el dibujo en la mayor área de pantalla posible.

AjustarGrados(double Grados): función devuelve las componentes x y y del ángulo suministrado, este es el ángulo de perspectiva.

AlOrigen(Graphics g): Es una función que traslada el origen del objeto g al origen de coordenadas de referencia propuesto.

Diagonales(JTable Dia,Vector ¡Boolean¿ VE): Asigna el objeto JTable a Diagonales y el Vector booleano a EEliminado.

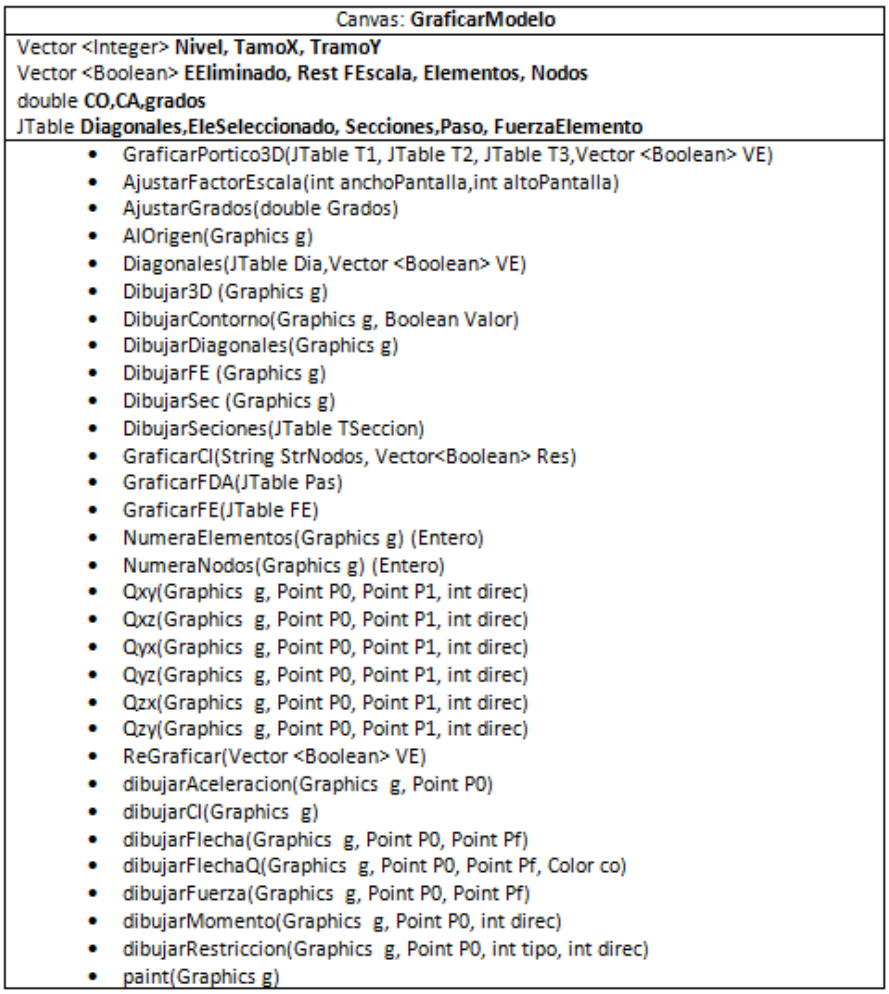

Figura A.15: Clase GraficarModelo.java

Dibujar3D (Graphics g): Función que permite el dibujado del pórtico en tres dimensiones. Esta rutina hace el llamado de varias rutinas para mostrar las características de manera visual al usuario.

DibujarContorno(Graphics g, Boolean Valor): Función cuyo propósito es dibujar el contorno de la estructura resaltando los elementos visibles desde la perspectiva.

DibujarDiagonales(Graphics g): Esta función se encarga de dibujar sobre el pórtico 3D los elementos diagonales que se han definido previamente. Devuelve el total de elementos.

DibujarFE (Graphics g): Esta función se encarga de mostrar por pantalla la fuerza en los elementos.

DibujarSec (Graphics g): Rutina encargada de mostrar por pantalla las secciones asignadas a los objetos en el pórtico 3D.

DibujarSeciones(JTable TSeccion): Rutina encargada de asignar al objeto Secciones el valor del parámetro.

GraficarCI(String StrNodos, Vector¡Boolean¿ Res): Función cuyo propósito es asignarle los valores de los parámetros a los objetos Rest y Strnodo respectivamente.

GraficarFDA(JTable Pas): función que asigna el valor del parámetro al objeto Paso.

public void GraficarFE(JTable FE): función que asigna el valor del parámetro al objeto FuerzaElemento.

NumeraElementos(Graphics g): Esta función se encarga de numerar los elementos en el pórtico 3D.

NumeraNodos(Graphics g): Función que se encarga de numerar los nodos en el pórtico 3D.

Qxy(Graphics g, Point P0, Point P1, int direc): rutina que se encarga de dibujar en la pantalla la carga distribuida del pórtico 3D en el plano X-Y. Comenzando en el P0 y terminando en el P1, con la dirección direc.

Qxz(Graphics g, Point P0, Point P1, int direc): rutina que se encarga de dibujar en la pantalla la carga distribuida del pórtico 3D en el plano X-Z. Comenzando en el P0 y terminando en el P1, con la dirección direc.

Qyx(Graphics g, Point P0, Point P1, int direc): rutina que se encarga de dibujar en la pantalla la carga distribuida del pórtico 3D en el plano Y-X. Comenzando en el P0 y terminando en el P1, con la dirección direc.

Qyz(Graphics g, Point P0, Point P1, int direc): rutina que se encarga de dibujar en la pantalla la carga distribuida del pórtico 3D en el plano Y-X. Comenzando en el P0 y terminando en el P1, con la dirección direc.

Qzx(Graphics g, Point P0, Point P1, int direc): rutina que se encarga de dibujar en la pantalla la carga distribuida del pórtico 3D en el plano X-Z. Comenzando en el P0 y terminando en el P1, con la dirección direc.

Qzy(Graphics g, Point P0, Point P1, int direc): rutina que se encarga de dibujar en la pantalla la carga distribuida del pórtico 3D en el plano Z-Y. Comenzando en el P0 y terminando en el P1, con la dirección direc.

ReGraficar(Vector ¡Boolean¿ VE): Actualiza el Valor de EEliminado según el parámetro.

dibujarAceleracion(Graphics g, Point P0): Dibuja un gráfico de líneas en zig-zag en el punto P0.

dibujarCI(Graphics g): Grafica en la pantalla las restricciones del pórtico 3D. Según el valor del objeto Res.

dibujarFlecha(Graphics g, Point P0, Point Pf): Dibuja un flecha entre los puntos P0 y Pf.

dibujarFlechaQ(Graphics g, Point P0, Point Pf, Color co): dibuja la flecha de las cargas distribuidas.

dibujarFuerza(Graphics g, Point P0, Point Pf): Dibuja la Fuerza en el Nodo con coordenada, P0. Es una flecha que comienza en el punto P0 y termina en el punto Pf.

dibujarMomento(Graphics g, Point P0, int direc): Dibuja una línea curva definiendo la dirección del momento.

dibujarRestriccion(Graphics g, Point P0, int tipo, int direc): Dibuja las restricciones en los nodos establecidos por P0, el tipo de restricción la asigna tipo, y la dirección direc.

paint(Graphics g): Es la función ,en java, que le indica al sistema que debe dibujar. Esta función llama a Dibujar3D(g)

A.16 GraficarPlantaRecV2.java

Esta clase se encarga de presentar al usuario un dibujo plano de la planta de la estructura a analizar. Los metodos y atributos de la clase son los mostrados en la Figura A.16.

GraficarPlantaRecV2(JTable taxx, JTable tazz): es el constructor de la clase, donde se encarga de asignarle memoria a cada objeto. Además de extraer la

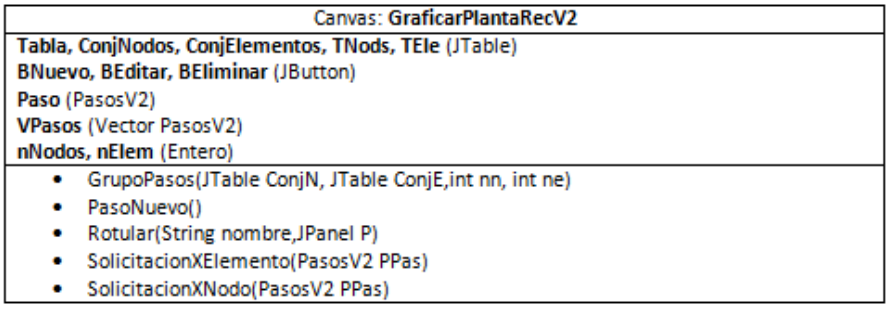

Figura A.16: Clase GraficarPlantaRecV2.java

información de los parámetros y asignarle los valores a los objetos.

DibujarPlanta(Graphics g): es la función que se encarga de dibujar la planta en la pantalla, obteniendo los valores de los objetos.

NumeroPorticos (): Devuelve el número de pórticos que tiene la planta.

dibujarFlecha(Graphics g, Point P0, Point Pf): función que se encarga de dibujar una flecha entre el punto P0 y el Pf, esta función es utilizada para graficar el eje coordenado del plano donde se encuentra dibujada la planta.

paint(Graphics g): Es la función ,en java, que le indica al sistema que debe dibujar. Esta función llama a DibujarPlanta(Graphics g).

A.17 GrupoPasos.java

Esta clase se encarga de manejar el conjunto de pasos que el usuario requiere para ejecutar el análisis. Entre las funciones que esta clase puede ejecutar se encuentra los objetos más importantes de la clase se ilustran en la Figura A.17.

GrupoPasos(JTable ConjN, JTable ConjE,int nn, int ne): Constructor paramétrico encargado de asignarle memoria a cada objeto de la clase asi como es el encargado de manejar las diferentes acciones que ejecutan los botones.

PasoNuevo(): Es una rutina que restablece los valores por defecto y visualizar la ventana para introducir los valores.

Rotular(String nombre,JPanel P): Es una función que se encarga de darle formato al panel que recibe como parámetro, esta rutina solo recibe el nombre del

JPanel: **GrupoPasos**
Tabla, ConjNodos, ConjElementos, TNods, TEle (JTable) **BNuevo, BEditar, BEliminar** (JButton) **Paso** (PasosV2) **VPasos** (Vector PasosV2) **nNodos, nElem** (Entero)
• GrupoPasos(JTable ConjN, JTable ConjE,int nn, int ne) • PasoNuevo() • Rotular(String nombre,JPanel P) • SolicitacionXElemento(PasosV2 PPas) • SolicitacionXNodo(PasosV2 PPas)

Figura A.17: Clase GrupoPasos.java

panel y el panel a rotular.

SolicitacionXElemento(PasosV2 PPas): es una función que retorna una tabla con la información de las solicitaciones realizadas a cada elemento, se genera una tabla por paso de análisis.

SolicitacionXNodo(PasosV2 PPas): es una función que retorna una tabla con la información de las solicitaciones realizadas a cada nodo, se genera una tabla por paso de análisis.

A.18 GrupoSecciones.java

Esta clase contiene los objetos y las funciones necesarias para el manejo de secciones definidas por el usuario, entre las aplicaciones que maneja esta clase se encuentran nueva sección, editar una sección y eliminar una sección. Los atributos más importantes de la clase se observan en la Figura A.18.

JPanel : **GrupoSecciones**
BNuevo, BEditar, BEliminar (JButton) **Tabla** (JTable) **Sec** (Secciones) **VSec** (Vector Secciones)
• GrupoSecciones() • Rotular(String nombre,JPanel P)

Figura A.18: Clase GrupoSecciones.java

GrupoSecciones(): Función cuyo propósito fundamental es la asignación de memoria a cada objeto así como la manipulación de los diferentes botones, estos botones

tienen sus funciones desarrolladas dentro de esta rutina.

Rotular(String nombre,JPanel P): Es una función que se encarga de darle formato al panel que recibe como parámetro, esta rutina solo recibe el nombre del panel y el panel a rotular.

A.19 GrupoSeccionesPlaca.java

Esta clase contiene los objetos y las funciones necesarias para el manejo de secciones de placa definidas por el usuario, entre las aplicaciones que maneja esta clase se encuentran nueva sección, editar una sección y eliminar una sección. Los atributos más importantes de la clase se visualizan en la Figura A.19.

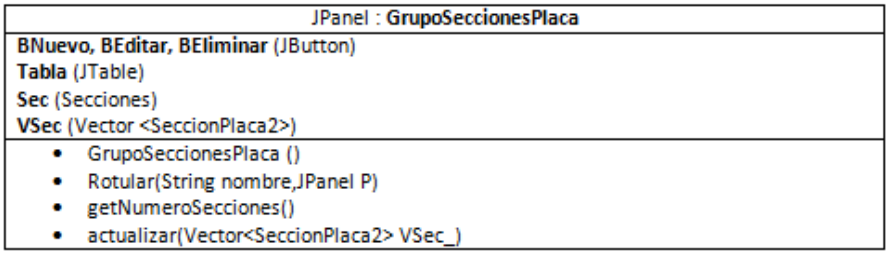

Figura A.19: Clase GrupoSeccionesPlaca.java

GrupoSeccionesPlaca (): Función cuyo propósito fundamental es la asignación de memoria a cada objeto así como la manipulación de los diferentes botones, estos botones tienen sus funciones desarrolladas dentro de esta rutina.

Rotular(String nombre,JPanel P): Es una función que se encarga de darle formato al panel que recibe como parámetro, esta rutina solo recibe el nombre del panel y el panel a rotular.

getNumeroSecciones(): Retorna el numero de secciones que existen y están almacenadas en el vector de secciones.

actualizar(Vector¡SeccionPlaca2¿ VSec_): Este procedimiento sobrescribe el vector existente por el que recibe en el parámetro.

A.20 MapaDano.java

Esta clase, contiene los procedimientos y atributos necesarios para visualizar el daño en la estructura al terminar la simulación. Permite de manera animada visualizar el aumento del daño en las estructuras. Los atributos y métodos más importantes de esta clase se ilustran en la Figura A.20.

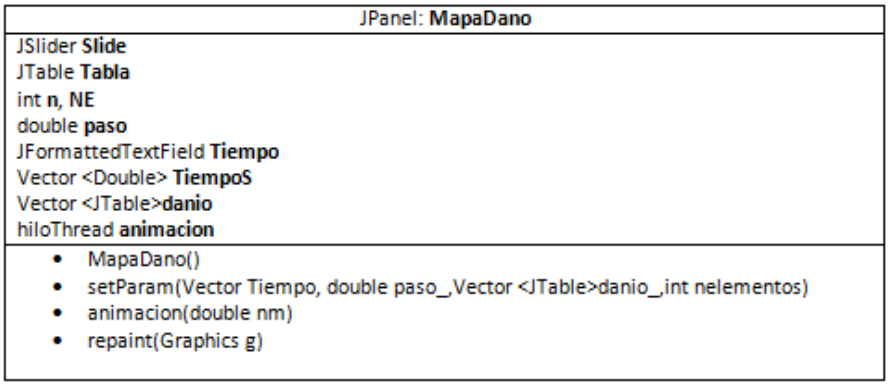

Figura A.20: Clase MapaDano.java

MapaDano(): El propósito fundamental de esta clase, es la asignación de memoria a cada objeto así como la manipulación de los diferentes botones, estos botones tienen sus funciones desarrolladas dentro de esta rutina.

setParam(Vector Tiempo, double paso_,Vector ¡JTable¿danio_,int nelementos): Se copia la información recibida por los parámetros directamente a los atributos propios de la clase.

animacion(double nm): Recibe como parámetro el numero de muestras que se van a graficar a medida que transcurre el tiempo de simulación. Es aquí donde se visualiza cada valor de daño, para un tiempo específico, en la estructura.

repaint(Graphics g): Dibuja las esferas del daño en los elementos.

A.21 Materiales .java

Clase que contiene los objetos y métodos necesarios para albergar y manejar la información relacionada con materiales. Los objetos más importantes de la clase se observan

en la Figura A.21.

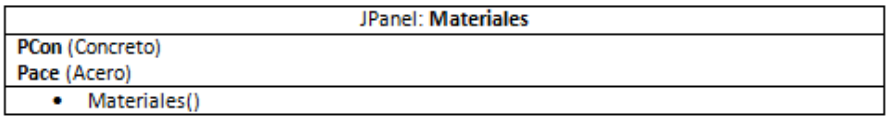

Figura A.21: Clase Materiales.java

Materiales(): Este es el constructor por defecto que asigna memoria a los objetos de la clase.

Además de los objetos y métodos, la clase contiene dos clases anidadas correspondientes al manejo de la Información del Acero y del Concreto.

Class Acero, contiene los siguiente objetos.

- MaxDef ,DefCed, DefFCe, EsfUlt, EsfTra, EsfTRT (JTextField)

Y su constructor **Acero()** como único método, encargado de asignar memoria a los objetos y valores por defecto, además de la configuración por pantalla.

Class Concreto, contiene los siguiente objetos.

- TResistencia, TDeformacionM, TDeformacionU, TModuloE, TECCU (JTextField)
- TDAlta ,TDBaja (JRadioButton)
- ODConfinado, ODNConfinado (JRadioButton)

Y su constructor **Concreto()** como único método, encargado de asignar memoria a los objetos y valores por defecto, además de la configuración por pantalla.

A.22 Modelo3D.java

La clase Modelo3D se encarga de representar la información pertinente al pórtico modelado, entiéndase niveles y tramos de la estructura, además de contener la geometría del mismo. En la Figura A.22 se observan las características de la clase.

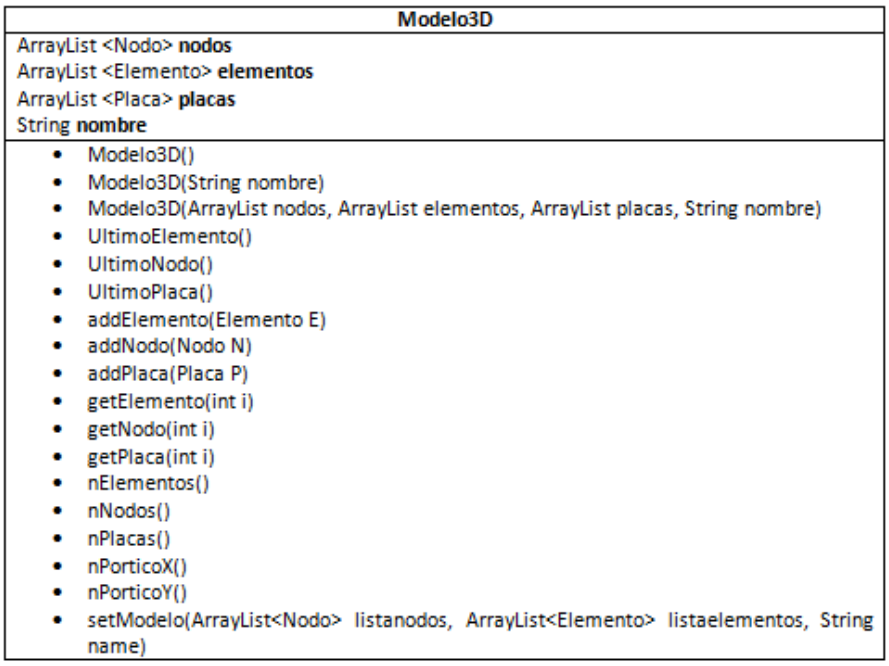

Figura A.22: Clase Modelo3D.java

Modelo3D(): Este es el constructor por defecto, el cual se encarga de asignarle memoria a cada atributo de la clase.

Modelo3D(String nombre): Es el constructor paramétrico que recibe el nombre de la estructura

Modelo3D(ArrayList nodos, ArrayList elementos, ArrayList placas, String nombre): Este constructor paramétrico recibe la geometría del pórtico en arreglos, y la asigna al atributo correspondiente de la clase.

UltimoElemento(): Devuelve el último elemento de la lista de elementos.

UltimoNodo(): Devuelve el último nodo de la lista de nodos.

UltimoPlaca(): Devuelve el último objeto de tipo placa contenido en la lista de placas.

addElemento(Elemento E): Agrega un elemento a la lista de elementos.

addNodo(Nodo N): Agrega un nodo a la lista de nodos.

addPlaca(Placa P): Agrega una placa a la lista de placas.

getElemento(int i): Obtiene el elemento de la lista en la posición i

getNodo(int i): Obtiene el nodo de la lista en la posición i

getPlaca(int i): Obtiene la placa de la lista en la posición i

nElementos(): Devuelve el número de elementos existentes en el modelo

nNodos(): Devuelve el número de nodos existentes en el modelo

nPlacas(): Devuelve el número de placas existentes en el modelo

nPorticoX(): Retorna un vector con las posiciones en el eje x del pórtico.

nPorticoY(): Retorna un vector con las posiciones en el eje y del pórtico.

setModelo(ArrayList¡Nodo¿ listanodos, ArrayList¡Elemento¿ listaelementos, String name): Actualiza el modelo existente según los nodos, elementos, que recibe por parámetros.

A.23 Nodo.java

La clase Nodo permite al usuario modelar un nodo con sus propiedades más representativas, es decir un nodo contiene 3 coordenadas, visibilidad e identificación. En la Figura A.23 se observan los atributos y métodos de la clase.

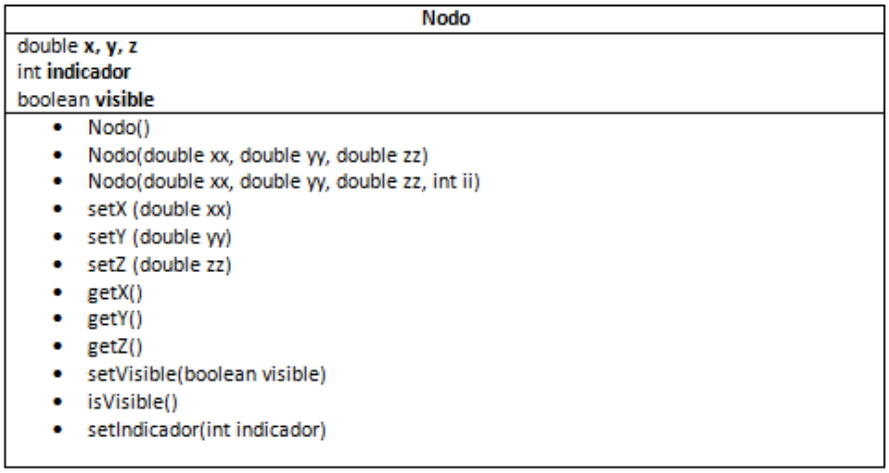

Figura A.23: Clase Nodo.java

Nodo(): Este es el constructor por defecto de la clase, encargado de asignarle memoria a cada atributo.

Nodo(double xx, double yy, double zz): Constructor paramétrico, el cual se encarga de asignarle los valores de x, y y z.

Nodo(double xx, double yy, double zz, int ii): Constructor paramétrico, el cual se encarga de asignarle los valores de x, y y z, además de asignarle un indicador.

setX (double xx): Asigna el valor de la coordenada x

setY (double yy): Asigna el valor de la coordenada y

setZ (double zz): Asigna el valor de la coordenada z

getX(): Devuelve el valor x del nodo

getY(): Devuelve el valor y del nodo

getZ(): Devuelve el valor z del nodo

setVisible(boolean visible): Asigna un valor lógico, para la visibilidad del nodo.

isVisible(): Devuelve un valor lógico de acuerdo con la visibilidad asignada

setIndicador(int indicador): Asigna un valor para el identificador del nodo

A.24 PasosV2.java

Clase que contiene objetos y métodos necesarios para la manipulación de la información concerniente a los pasos de análisis. Los objetos más importantes de la clase se observan en la Figura A.24.

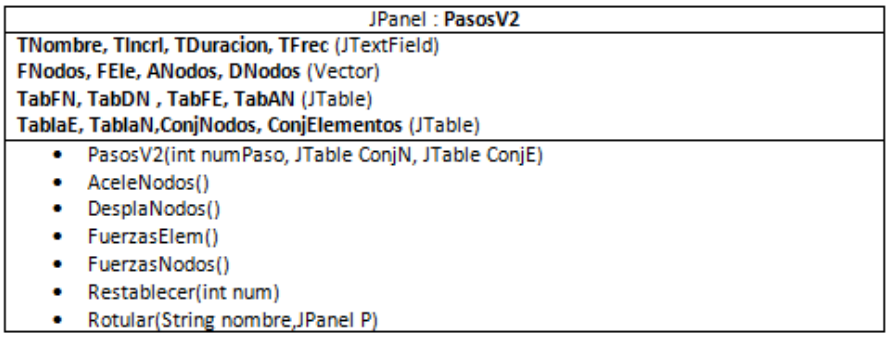

Figura A.24: Clase PasosV2.java

PasosV2(int numPaso, JTable ConjN, JTable ConjE): constructor paramétrico, que se encarga de asignar memoria a los objetos de la clase, así como también es el encargado de contener los atributos necesarios para la visualización por pantalla de la información pertinente.

AceleNodos(): Función que maneja una ventana para mostrar la interfaz de cómo el usuario introduce la información sobre la aceleración en los nodos.

DesplaNodos(): Función que maneja una ventana para mostrar la interfaz de cómo el usuario introduce la información sobre los desplazamientos impuestos en los nodos.

FuerzasElem(): Función que maneja una ventana para mostrar la interfaz de cómo el usuario introduce la información sobre las fuerzas en los elementos.

FuerzasNodos(): Función que maneja una ventana para mostrar la interfaz de cómo el usuario introduce la información sobre las fuerzas en los nodos.

Restablecer(int num): asigna los valores por defecto a un paso según el parámetro.

Rotular(String nombre,JPanel P): Es una función que se encarga de darle formato al panel que recibe como parámetro, esta rutina solo recibe el nombre del panel y el panel a rotular.

A.25 Placa.java

Clase encargada de modelar la placa, estructura que sirve de soporte o sostén entre cuatro columnas. Esta clase contiene los atributos para la geometría de la placa y sus características para definir su sección. En la Figura A.25 se observan los métodos y los atributos de la misma.

Placa(int ind, Nodo n1, Nodo n2, Nodo n3, double EE, double vv, double tt, double dd): Constructor paramétrico, el cual recibe los valores por parámetro y los asigna a los atributos de la clase.

getArea(): Devuelve el área de la placa.

getNodo1(): Retorna el nodo 1 de la placa.

getNodo2(): Retorna el nodo 2 de la placa.

getNodo3(): Retorna el nodo 3 de la placa.

getPropiedades(): Devuelve un arreglo con las propiedades de la placa

getSeccion(): Retorna una cadena con el nombre de la sección asignada

getindicador(): Devuelve un entero con el numero del indicador.

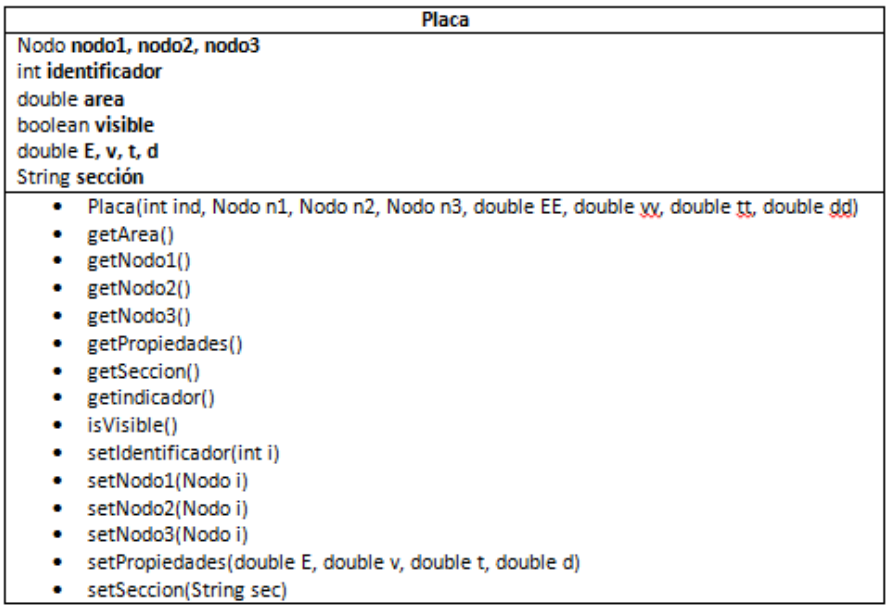

Figura A.25: Clase Placa.java

isVisible(): Devuelve un valor lógico que indica si la placa esta visible o no.

setIdentificador(int i): Asigna un valor entero para el identificador de la placa

setNodo1(Nodo i): Asigna el Nodo 1 a la placa.

setNodo2(Nodo i): Asigna el Nodo 2 a la placa

setNodo3(Nodo i): Asigna el Nodo 3 a la placa

setPropiedades(double E, double v, double t, double d): Asigna los valores de la propiedades a la placa

setSeccion(String sec): Asigna el nombre pasado por parámetro a la placa.

A.26 PruebaRuntime.java

Clase que contiene la información de objetos y métodos para la conexión del portal con el programa de elementos finitos. La clase funciona como una rutina de conexión donde no posee ningún objeto propio y tiene solo como método el constructor paramétrico. En la Figura A.26 se muestran los atributos y los métodos de la clase.

PruebaRuntime(String Nombre) recibe el nombre del archivo INP que se está analizando y conecta con el programa de elementos finitos. Para lograr el objetivo de la

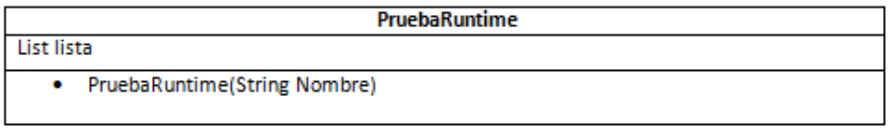

Figura A.26: Clase PruebaRuntime.java

clase es necesario que exista un archivo llamado ArchivoINP.txt que sirve de enrutador entre el programa de elementos finitos y el archivo INP generado por el portal.

A.27 RGraficador.java

Es una clase que contiene los atributos y métodos adecuados para enlazar a los resultados gráficos del análisis del pórtico. Esta clase permite la interfaz del usuario para visualizar los resultados de forma gráfica. Los principales atributos de la clase se muestran en la Figura A.27.

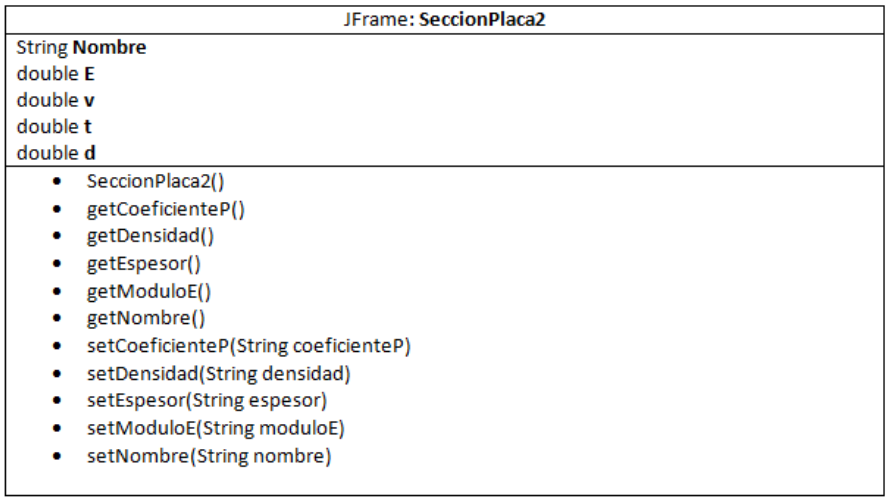

Figura A.27: Clase RGraficador.java

RGraficador(int no, int el): es el constructor paramétrico encargado de asignarle memoria a cada atributo de la clase, así de configurar la ventana que servirá de interfaz al usuario.

Rotular(String nombre,JPanel P): es una función que se encarga de darle

formato al panel que recibe como parámetro, esta rutina solo recibe el nombre del panel y el panel a rotular.

llenarcomboE(): función cuyo objetivo es asignarle a los comboBox el conjunto de variables pertenecientes a los elementos.

llenarcomboN(): función cuyo objetivo es asignarle a los comboBox el conjunto de variables pertenecientes a los nodos.

A.28 SeccionPlaca2.java

Clase que contiene los atributos y métodos que permiten modelar una sección del elemento tipo placa. Esta clase contiene los siguientes atributos considerados de relevancia en la Figura A.28.

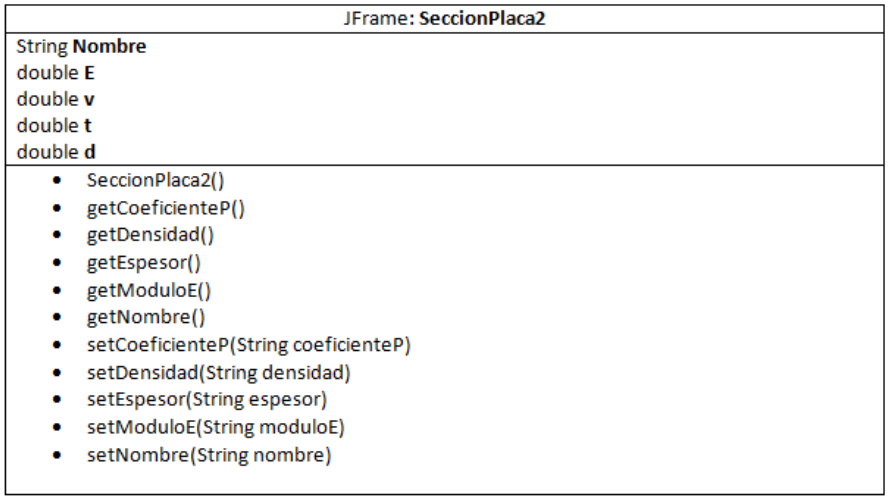

Figura A.28: Clase SeccionPlaca2.java

SeccionPlaca2(): Es el constructor por defecto encargado de asignarle memoria a cada uno de los objetos de la clase así como de configurar la ventana contenedora para visualizar e interactuar con el usuario. Además de contener los eventos de la ventana.

getCoeficienteP(): Devuelve una cadena con el valor de v (Coeficiente de Poisson)

getDensidad(): Devuelve una cadena con el valor de d (Densidad del Concreto)

getEspesor(): Devuelve una cadena con el valor de t (Espesor de la placa)

getModuloE(): Devuelve una cadena con el valor de E (Modulo de Elasticidad del Concreto)

getNombre(): Devuelve una cadena con el nombre de la sección

setCoeficienteP(String coeficienteP): Asigna el valor de v (Coeficiente de Poisson) pasado por parámetro

setDensidad(String densidad): Asigna el valor de d (Densidad del Concreto) pasado por parámetro

setEspesor(String espesor): Asigna el valor de t (Espesor de la Placa) pasado por parámetro

setModuloE(String moduloE): Asigna el valor de E (Modulo de Elasticidad del Concreto) pasado por parámetro

setNombre(String nombre): Asigna nombre de la sección pasado por parámetro

A.29 Secciones.java

Esta clase contiene los atributos y métodos que permiten modelar una sección rectangular de manera virtual. Esta clase contiene los siguientes atributos considerados de relevancia. En la Figura A.29 se ilustra las características mas resaltantes de la clase.

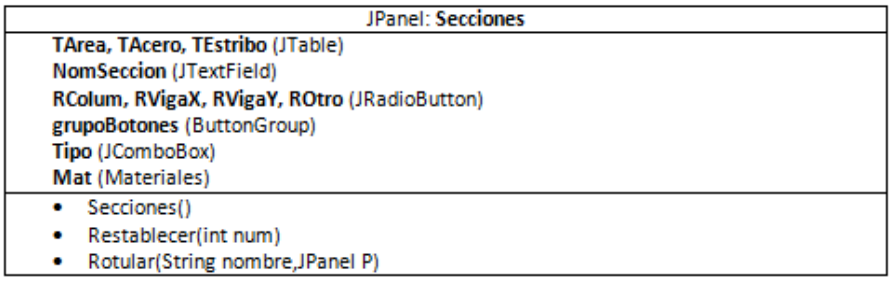

Figura A.29: Clase Secciones.java

Secciones(): es el constructor por defecto encargado de asignarle memoria a cada uno de los objetos de la clase así como de configurar la ventana contenedora para visualizar e interactuar con el usuario. Además de contener los eventos de la ventana.

Restablecer(int num): asigna los valores por defecto a una sección según el parámetro dado.

Rotular(String nombre,JPanel P): Es una función que se encarga de darle formato al panel que recibe como parámetro, esta rutina solo recibe el nombre del panel y el panel a rotular.

A.30 SeleccionarElementos.java

Clase que contiene los atributos y métodos los cuales permiten definir, manejar y visualizar el conjunto de elementos (Viga Columna) por el usuario. Esta clase funciona como enlace entre el la ventana que maneja los elementos y el portal. Contiene los atributos más importante de la clase se listan en la Figura A.30.

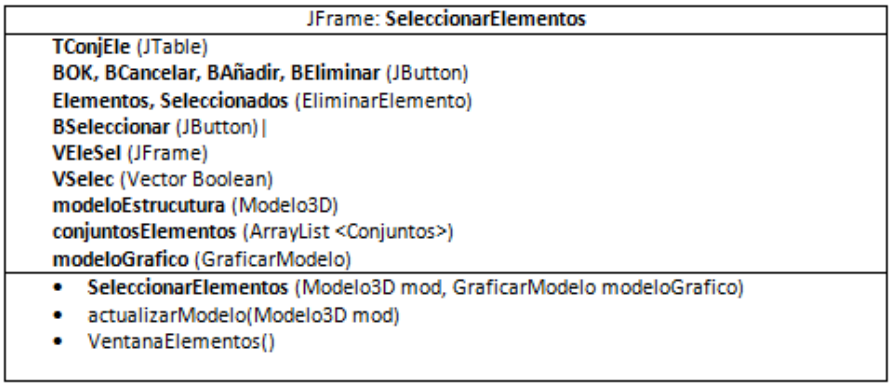

Figura A.30: Clase SeleccionarElementos.java

SeleccionarElementos (Modelo3D mod, GraficarModelo modeloGrafico): Es el constructor por paramétrico el cual asigna memoria a los atributos de la clase, recibe un objeto Modelo3D y un objeto GraficarModelo para visualizar la selección de los elementos.

actualizarModelo(Modelo3D mod): Este método recibe un objeto de tipo Modelo3D y lo asigna a su atributo, con el fin de actualizar el modelo.

VentanaElementos(): método que permite visualizar por pantalla la ventana manejadora de elementos, además este método identificar en la ventana de elementos aquellos que ya han sido seleccionados.

A.31 SeleccionarNodos.java

Clase que contiene los objetos y métodos los cuales permiten definir, manejar y visualizar el conjunto de nodos por el usuario. Esta clase funciona como enlace entre el la ventana que maneja los nodos y el portal. Contiene los atributos más importante de la clase se listan en la Figura A.31.

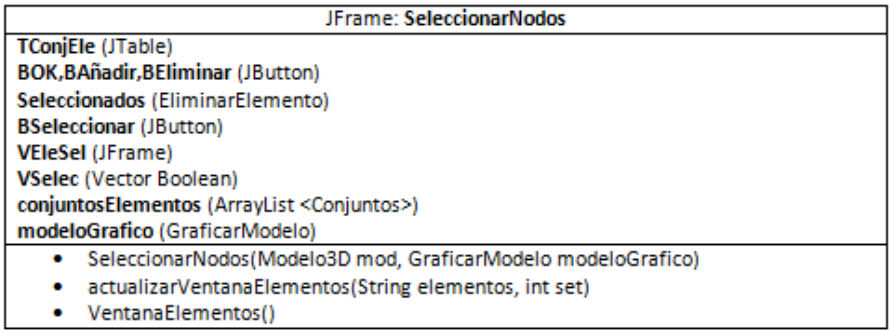

Figura A.31: Clase SeleccionarNodos.java

SeleccionarNodos(Modelo3D mod, GraficarModelo modeloGrafico): Es el constructor por paramétrico el cual asigna memoria a los atributos de la clase, recibe un objeto Modelo3D y un objeto GraficarModelo para visualizar la selección de los elementos.

actualizarVentanaElementos(String elementos, int set): Este método recibe en una cadena los nodos y la posición , indica en la lista de nodos cuales están seleccionados y cuáles no.

VentanaElementos(): método que permite visualizar por pantalla la ventana manejadora de elementos, además este método identificar en la ventana de elementos aquellos que ya han sido seleccionados.

A.32 SeleccionarPlacas.java

Clase que contiene los objetos y métodos los cuales permiten definir, manejar y visualizar el conjunto de placas por el usuario. Esta clase funciona como enlace entre el la ventana que maneja los nodos y el portal. Contiene los atributos más importante de

la clase se listan en la Figura A.32.

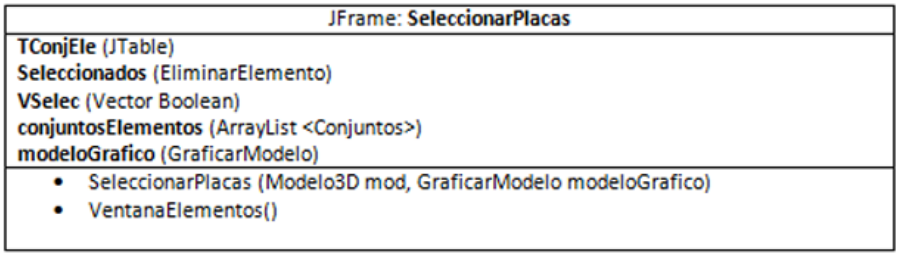

Figura A.32: Clase SeleccionarPlacas.java

SeleccionarPlacas (Modelo3D mod, GraficarModelo modeloGrafico): Es el constructor por paramétrico el cual asigna memoria a los atributos de la clase, recibe un objeto Modelo3D y un objeto GraficarModelo para visualizar la selección de los elementos tipo placa.

VentanaElementos(): Método que permite visualizar por pantalla la ventana manejadora de elementos, además este método identificar en la ventana de elementos aquellos que ya han sido modificados.

Apéndice B

Códigos Servlets

En este apéndice se muetran los códigos usados para generar los servlets que se ejecutan en el servidor.

Listing B.1: danio.java

```
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
import java.net.*;
import java.util.Vector;
import javax.swing.table.DefaultTableModel;

/**
 * Simple demonstration for an Applet <-> Servlet communication.
 */
public class danio extends HttpServlet {
        /**
         * Get a String-object from the applet and send it back.
         */
        public void doPost(
                HttpServletRequest request,
                HttpServletResponse response)
                throws ServletException, IOException {
                try {
                        response.setContentType("application/x-java-
                            serialized-object");
```

```
                    // read a String-object from applet
                    // instead of a String-object, you can transmit any object, which
                    // is known to the servlet and to the applet
                    InputStream in = request.getInputStream();
                    ObjectInputStream inputFromApplet = new
                        ObjectInputStream(in);
                    String echo = (String) inputFromApplet.readObject
                        ();

                    System.out.println(echo);
                    //

                    String aux[]=echo.split("#");
                    int elementos=Integer.parseInt(aux[1]);
                    String Nombre=aux[0];
                    System.out.println(Nombre+"-"+elementos);

                    String resul=leerDanio(Nombre,elementos);
                    System.out.println(resul);

                    OutputStream outstr = response.getOutputStream();
                    ObjectOutputStream oos = new ObjectOutputStream(
                        outstr);
                    oos.writeObject(resul);
                    oos.flush();
                    oos.close();


            } catch (Exception e) {
                    e.printStackTrace();
            }
    }

    public String leerDanio(String nombre, int ele){

        FileReader AAbierto;
```

```
String Cadena = "", result="";


int elementos=ele;
String Nombre=nombre;

try{
    AAbierto = new FileReader ("../webapps/portal/Archivos/"+
        Nombre+"_E.txt");
    BufferedReader BAA = new BufferedReader(AAbierto);
    Cadena = BAA.readLine();

    while(Cadena!=null){

    //System.out.println(Cadena);
    String str2[]=Cadena.split(",");

    if(str2.length==1){

      result=result+Cadena+"#";

    }else{

        for(int i=0; i<elementos;i++){

            result=result+(i+1)+"#";
            result=result+(Double.parseDouble(str2[15]))+"#";
            result=result+(Double.parseDouble(str2[16]))+"#";
            result=result+(Double.parseDouble(str2[17]))+"#";
            result=result+(Double.parseDouble(str2[18]))+"#";
            result=result+(Double.parseDouble(str2[19]))+"#";
            result=result+(Double.parseDouble(str2[20]))+"#";
            result=result+(Double.parseDouble(str2[21]))+"#";
            result=result+(Double.parseDouble(str2[22]))+"#";

            Cadena = BAA.readLine();
```

```
                    if(Cadena!=null)
                        str2=Cadena.split(",");
                }

                if(str2.length==1){
                    result=result+"~"+(Cadena)+"#";
                }

            }
            Cadena = BAA.readLine();
        }

          //System.out.println(Danio);

        AAbierto.close();
        }
        catch(FileNotFoundException fnfe) {}
        catch (IOException ioe) {}

        System.out.println(result);
        return result;
    }

}
```

Listing B.2: leerRes.java

```
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
import java.net.*;
import java.util.Vector;

/**
 * Simple demonstration for an Applet <-> Servlet communication.
 */
public class leerRes extends HttpServlet {
        /**
```

```
     * Get a String-object from the applet and send it back.
     */
    public void doPost(
            HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
            try {
                    response.setContentType("application/x-java-
                        serialized-object");

                    // read a String-object from applet
                    // instead of a String-object, you can transmit any object, which
                    // is known to the servlet and to the applet
                    InputStream in = request.getInputStream();
                    ObjectInputStream inputFromApplet = new
                        ObjectInputStream(in);
                    String echo = (String) inputFromApplet.readObject
                        ();

                    System.out.println(echo);


                    FileReader AAbierto;
                    String Cadena = "",aRes="";

                    try{

                        AAbierto = new FileReader ("../webapps/portal
                            /Archivos/"+"res");

                        //System.out.println("Entre al try");


                        BufferedReader BAA = new BufferedReader(
                            AAbierto);

                        while(Cadena!=null){
```

```
                    Cadena = BAA.readLine();
                    aRes=aRes+Cadena+"#";
                }

                AAbierto.close();

            }
            catch(FileNotFoundException fnfe) {}
            catch (IOException ioe) {}



            System.out.println(aRes);
            OutputStream outstr = response.getOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(
                outstr);
            oos.writeObject(aRes);
            oos.flush();
            oos.close();


        } catch (Exception e) {
                e.printStackTrace();
        }
    }


}
```

Listing B.3: p3d.java

```
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;

/**
 * Simple demonstration for an Applet <-> Servlet communication.
 */
```

```
public class p3d extends HttpServlet {
    /**
     * Get a String-object from the applet and send it back.
     */
    public void doPost(
        HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        try {
            response.setContentType("application/x-java-serialized-object
                ");

            // read a String-object from applet
            // instead of a String-object, you can transmit any object, which
            // is known to the servlet and to the applet
            InputStream in = request.getInputStream();
            ObjectInputStream inputFromApplet = new ObjectInputStream(in)
                ;
            String echo = (String) inputFromApplet.readObject();


            FileWriter fichero = null;
                PrintWriter pw = null;

            try{

                        String cadena[]=echo.split("~");

                        //Genera el inp
                        String nombreArchivo= cadena[0]+".p3d";
                        String Archivo = cadena[1];
                    fichero = new FileWriter("../webapps/portal/Archivos/
                        "+nombreArchivo);
                        pw = new PrintWriter(fichero);

                    pw.println(Archivo);
            System.out.println("Escribiendo Archivo P3D...");
```

```
                        echo="True";

            fichero.close();


               }
               catch (Exception e) {
                 e.printStackTrace();
                } finally {
                try {
                    // Nuevamente aprovechamos el finally para
                    // asegurarnos que se cierra el fichero.
                    if (null != fichero)
                            fichero.close();
                    } catch (Exception e2) {
                            e2.printStackTrace();
                }
                }

            // echo it to the applet
            OutputStream outstr = response.getOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(outstr);
            oos.writeObject(echo);
            oos.flush();
            oos.close();

        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }


}
```

Listing B.4: proper.java

```
import java.io.*;
```

```
import javax.servlet.ServletException;
import javax.servlet.http.*;
import java.net.*;

/**
 * Simple demonstration for an Applet <-> Servlet communication.
 */
public class proper extends HttpServlet {
    /**
     * Get a String-object from the applet and send it back.
     */
    public void doPost(
        HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        try {
            response.setContentType("application/x-java-serialized-object
                ");

            // read a String-object from applet
            // instead of a String-object, you can transmit any object, which
            // is known to the servlet and to the applet
            InputStream in = request.getInputStream();
            ObjectInputStream inputFromApplet = new ObjectInputStream(in)
                ;
            String echo = (String) inputFromApplet.readObject();

                        System.out.println(echo);
                        //
                        String dia=generarProp(echo);
                        System.out.println(dia);

                        OutputStream outstr = response.getOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(outstr);
            oos.writeObject(dia);
            oos.flush();
            oos.close();
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private String generarProp(String cad) {


        String dat_[]=cad.split("#");
        String conjunto = dat_[0];
        String seci = dat_[1], secj=dat_[2];
        double lon = Double.parseDouble(dat_[3]);
        double b = Double.parseDouble(dat_[4]);
        double h = Double.parseDouble(dat_[5]);
        double E = Double.parseDouble(dat_[6]);
        double n1 = Double.parseDouble(dat_[7]);
        double n2 = Double.parseDouble(dat_[8]);
        int eje = Integer.parseInt(dat_[9]);
        double d14 = Double.parseDouble(dat_[10]);
        double d04 = Double.parseDouble(dat_[11]);
        double d05 = Double.parseDouble(dat_[12]);
        String Salida="";


        String NombreArchivo = "PROPERTY.txt";
        FileWriter Archi = null;
        FileReader Seccioni = null;
        FileReader Seccioniz = null;
        FileReader Seccionj = null;
        FileReader Seccionjz = null;
        FileReader Secc = null;
        String Cadena = "";

        try{
```

```
Archi = new FileWriter ("../webapps/portal/Archivos/"+
    NombreArchivo);
Seccioni = new FileReader ("../webapps/portal/Archivos/res_"+
    seci);
BufferedReader Seci = new BufferedReader(Seccioni);
Seccioniz = new FileReader ("../webapps/portal/Archivos/res_"
   +seci);
BufferedReader Seciz = new BufferedReader(Seccioniz);
Seccionj = new FileReader ("../webapps/portal/Archivos/res_"+
    secj+"_j");
BufferedReader Secj = new BufferedReader(Seccionj);
Seccionjz = new FileReader ("../webapps/portal/Archivos/res_"
   +secj+"_j");
BufferedReader Secjz = new BufferedReader(Seccionjz);
Secc = new FileReader ("../webapps/portal/Archivos/res_"+seci
    );
BufferedReader Sec_ = new BufferedReader(Secc);

int Cont =1;


Archi.write("*PROPERTY,ELSET="+conjunto+"\n");

            // Calculo de propiedades por conjunto de elementos
            System.out.println(lon);
            //System.out.println(TElementos.getRowCount());

            double RP=0.2, A, G, J,a,c;
            a=b;
            c=h;
            String str[]=Cadena.split(",");
            // Area
            A=b*h;
            G=E/(2*(1+RP));

            a=Math.max(b, h);
            c=Math.min(b, h);
```

```
double r1 = (1-Math.pow(c, 4)/(12*Math.pow(a,4)))
    ;
double r2 = 0.21*(c/a)*r1;
J=a*Math.pow(c, 3)*((1.0/3.0) - r2);


double diyp = 0;
double diyn = 0;
double dizp = 0;
double dizn = 0;
double djyp = 0;
double djyn = 0;
double djzp = 0;
double djzn = 0;


if(eje==1){

    dizp = djzn = d14;
    djzp = dizn = b - d04;
    diyn = djyp = h - d05;
    diyp = djyn = h - d05;

}


if(eje==2){

    diyp = djyn = b - d05;
    djyp = diyn = b - d05;
    dizn = djzp = d14;
    dizp = djzn = h - d04;

}


if(eje==3){
```

```
                diyp = djyn = b - d05;
                djyp = diyn = b - d05;
                dizn = djzp = h - d04;
                dizp = djzn = d14;

            }

            Cadena = Sec_.readLine();//1
            Cadena = Sec_.readLine();//2
            Cadena = Sec_.readLine();//3
            Cadena = Sec_.readLine();//4
            Cadena = Sec_.readLine();//5

            //str=Cadena.split(",");
            //System.out.println(str[7]);
            //String Masa = str[7];

            Cadena = Sec_.readLine();//6
            str=Cadena.split(",");

            String Masa = str[0];

            String eiz = str[1];
            String ea = str[2];

            for (int i=0 ; i<=17 ;i++)
                Cadena = Sec_.readLine();//i --- 17
            str=Cadena.split(",");
            String eiy = str[1];

            double v21=Double.parseDouble(eiy);
            double v22=Double.parseDouble(eiz);
            double v23=Double.parseDouble(ea);


        for(int i=1; i<39; i++){
```

```
if(i!=21 && i!=22 && i!=23 && i!=24 && i!=25 && i!=26 &&
    i!=27 && i!=28 && i!=29 && i!=30 && i!=31 && i!=32 &&
    i!=33 && i!=34 && i!=35 && i!=36 && i!=37)
     Archi.write("0.0");

if(i==21)
     Archi.write(""+eiy);

if(i==22)
     Archi.write(""+eiz);

if(i==23)
     Archi.write(""+ea);

if(i==24)
     Archi.write(""+(G*J));

if(i==25)
     Archi.write(""+A);

if(i==26)
     Archi.write(""+J);

if(i==27)
     Archi.write(""+2);

if(i==28)
     Archi.write(""+Masa);

if(i==29)
     Archi.write(""+0.25);

if(i==30)
     Archi.write(""+diyp);

if(i==31)
```

```
                Archi.write(""+diyn);

            if(i==32)
                Archi.write(""+dizp);

            if(i==33)
                Archi.write(""+dizn);

            if(i==34)
                Archi.write(""+djyp);

            if(i==35)
                Archi.write(""+djyn);

            if(i==36)
                Archi.write(""+djzp);

            if(i==37)
                Archi.write(""+djzn);


            if(Cont%8==0 && i>0)
                Archi.write(",\n");
            else
                Archi.write(",");

            Cont++;
        }

        for(int i=0; i<=26; i++){
            Cadena = Seci.readLine();
        }

        str=Cadena.split(",");

        Archi.write(str[2]+","+str[3]+",\n");
        Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+",");
```

```
double v39 = Double.parseDouble(str[2]);
double v42 = Double.parseDouble(str[5]);



Cadena = Seci.readLine();//27
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+","); //
    52

double v50 = Double.parseDouble(str[5]);

Cadena = Seci.readLine(); //28
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+","); //60

Cadena = Seci.readLine(); //29
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+","); //
    68

double v68 = Double.parseDouble(str[7]);

Cadena = Seci.readLine(); //30
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+","); //
    76

double v76 = Double.parseDouble(str[7]);

Cadena = Seci.readLine(); //31
str=Cadena.split(",");
```

```
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+",");
    //84

double v84 = Double.parseDouble(str[7]);

Cadena = Seci.readLine(); //32
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+","); //
    92



Cadena = Seci.readLine(); //33
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+","); //
    100

Cadena = Seci.readLine(); //34
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");
Archi.write(str[4]+","+str[5]+","); //106

double v102 = Double.parseDouble(str[1]);

Cadena = Seciz.readLine();

for(int i=0; i<=7; i++){
     Cadena = Seciz.readLine();
}

//System.out.println(Cadena);
str=Cadena.split(","); //8
Archi.write(str[2]+","+str[3]+","+str[4]+","+str[5]+","+str
    [6]+","+str[7]+",\n"); //112
```

```
double v110 = Double.parseDouble(str[5]);

Cadena = Seciz.readLine();//9
str=Cadena.split(",");  //120
double v118 = Double.parseDouble(str[5]);
Archi.write(Cadena+",\n");

Cadena = Seciz.readLine();//10
str=Cadena.split(",");  //128
Archi.write(Cadena+",\n");

Cadena = Seciz.readLine();//11
str=Cadena.split(",");  //136
double v136 = Double.parseDouble(str[7]);
Archi.write(Cadena+",\n");

Cadena = Seciz.readLine();//12
str=Cadena.split(",");  //144
double v144 = Double.parseDouble(str[7]);
Archi.write(Cadena+",\n");

Cadena = Seciz.readLine();//13
str=Cadena.split(",");  //152
double v152 = Double.parseDouble(str[7]);
Archi.write(Cadena+",\n");

Cadena = Seciz.readLine();//14
str=Cadena.split(",");  //160
Archi.write(Cadena+",\n");

Cadena = Seciz.readLine();//15
str=Cadena.split(",");  //168
Archi.write(Cadena+",\n");

Cadena = Seciz.readLine();//16
str=Cadena.split(",");  //174
double v170 = Double.parseDouble(str[1]);
```

```
Archi.write(Cadena+",");


for(int i=0; i<=26; i++){
    Cadena = Secj.readLine();
}

str=Cadena.split(",");

Archi.write(str[2]+","+str[3]+",\n"); //176
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+",");//
    180
double v178 = Double.parseDouble(str[5]);

Cadena = Secj.readLine();//27
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");//
    184
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+",");//
    188
double v186 = Double.parseDouble(str[5]);

Cadena = Secj.readLine(); //28
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");//
    192
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+",");//
    196

Cadena = Secj.readLine(); //29
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");//
    200
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+","); //
    204
double v204 = Double.parseDouble(str[7]);
```

```
Cadena = Secj.readLine(); //30
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");//
    208
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+","); //
    212
double v212 = Double.parseDouble(str[7]);

Cadena = Secj.readLine(); //31
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");//
    216
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+","); //
    220

double v220 = Double.parseDouble(str[7]);
Cadena = Secj.readLine(); //32
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");//
    224
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+",");//
    228
double v222 = Double.parseDouble(str[1]);

Cadena = Secj.readLine(); //33
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");//
    232
Archi.write(str[4]+","+str[5]+","+str[6]+","+str[7]+","); //
    236

Cadena = Secj.readLine(); //34
str=Cadena.split(",");
Archi.write(str[0]+","+str[1]+","+str[2]+","+str[3]+",\n");//
    240
Archi.write(str[4]+","+str[5]+","); //242
double v238 = Double.parseDouble(str[1]);
```

```
Cadena = Secjz.readLine();

for(int i=0; i<=7; i++){
    Cadena = Secjz.readLine();
}

//System.out.println(Cadena);
str=Cadena.split(",");  //8
Archi.write(str[2]+","+str[3]+","+str[4]+","+str[5]+","+str
    [6]+","+str[7]+",\n");  //248
double v246 = Double.parseDouble(str[5]);

Cadena = Secjz.readLine();//9
str=Cadena.split(",");  //256
Archi.write(Cadena+",\n");
double v254 = Double.parseDouble(str[5]);

Cadena = Secjz.readLine();//10
str=Cadena.split(",");  //264
Archi.write(Cadena+",\n");

Cadena = Secjz.readLine();//11
str=Cadena.split(",");  //272
Archi.write(Cadena+",\n");
double v272 = Double.parseDouble(str[7]);

Cadena = Secjz.readLine();//12
str=Cadena.split(",");  //280
Archi.write(Cadena+",\n");
double v280 = Double.parseDouble(str[7]);

Cadena = Secjz.readLine();//13
str=Cadena.split(",");  //288
Archi.write(Cadena+",\n");
double v288 = Double.parseDouble(str[7]);
```

```
        Cadena = Secjz.readLine();//14
        str=Cadena.split(",");  //296
        Archi.write(Cadena+",\n");


        Cadena = Secjz.readLine();//15
        str=Cadena.split(",");  //304
        Archi.write(Cadena+",\n");

        Cadena = Secjz.readLine();//16
        str=Cadena.split(",");  //310
        Archi.write(Cadena+",");
        double v306 = Double.parseDouble(str[1]);

      /*
        System.out.println(v21+" *21");
        System.out.println(v22+" *22");
        System.out.println(v23+" *23");
        System.out.println(v39+" *39");
        System.out.println(v42+" *42");
        System.out.println(v50+" *50");
        System.out.println(v68+" *68");
        System.out.println(v76+" *76");
        System.out.println(v89+" *89");
        System.out.println(v102+" *102");
        System.out.println(v110+" *110");
        System.out.println(v118+" *118");
        System.out.println(v136+" *136");
        System.out.println(v144+" *144");
        System.out.println(v152+" *152");
        System.out.println(v170+" *170");
        System.out.println(v178+" *178");
        System.out.println(v186+" *186");
        System.out.println(v204+" *204");
        System.out.println(v212+" *212");
        System.out.println(v222+" *222");
```

```
System.out.println(v238+" *238");
System.out.println(v246+" *246");
System.out.println(v254+" *254");
System.out.println(v272+" *272");
System.out.println(v280+" *280");
System.out.println(v288+" *288");
System.out.println(v306+" *306");

 */



// Momento torsor (311 312)
Archi.write(0.0+","+0.0+",\n");

// Momento Torsor (313 314 315 316 317 318 319 320)
Archi.write(0.0+","+0.0+","+0.0+","+0.0+","+0.0+","+0.0+","
    +0.0+","+n1+",\n");
Archi.write(n2+",50,10000,3, 6, 10, 4, 8,\n");
Archi.write("4, 4, 5,4\n");



Archi.close();
Secc.close();
Seccioni.close();
Seccioniz.close();
Seccionj.close();
Seccionjz.close();

FileReader Seccion;
Seccion = new FileReader ("../webapps/portal/Archivos/"+"
    PROPERTY.txt");
BufferedReader Temp = new BufferedReader(Seccion);



double Fac = 1e-6;
double cr1 = Math.min(v68,v102)*Fac*0.5*Math.min(diyp,diyn);
```

```
double cr2 = Math.min(v204,v238)*Fac*0.5*Math.min(djyp,djyn);
double cr3 = (v39/v23)*Fac*lon*100;
double cr4 = Math.min(v136,v170)*Fac*0.5*Math.min(dizp,dizn);
double cr5 = Math.min(v272,v306)*Fac*0.5*Math.min(djzp,djzn);
double cr6 = Fac;
double T1 = Math.min(Math.min(v50,v84),Math.min(v118,v152))*
    Fac;
double T2 = Math.min(Math.min(v186,v220),Math.min(v254,v288))
    *Fac;
double T3 = (v42*v42)*lon*Fac/(6*v21);
double T4 = (v212*v212)*lon*Fac/(6*v21); ///// Este Valor Cambia
    v178 por v212
double T5 = (v110*v110)*lon*Fac/(6*v22);
double T6 = (v280*v280)*lon*Fac/(6*v22); //// Este Valor cambia
    v246 por v280
double T7 = (v76*v76)*lon*Fac/(6*v21);
double T8 = (v178*v178)*lon*Fac/(6*v21); ///// Este Valor Cambia
    v212 por v178
double T9 = (v144*v144)*lon*Fac/(6*v22);
double T10 = (v246*v246)*lon*Fac/(6*v22); //// Este Valor cambia
    v280 por v246
double g1 = cr1;
double g2 = cr2;
double g3 = cr4;
double g4 = cr5;



Cadena = Temp.readLine();
for(int i=0;Cadena!=null;i++){

    if(i==1){

        str=Cadena.split(",");
        Salida=Salida+cr1+","+cr2+","+cr3+","+cr4+","+cr5+","
            +cr6+","+T1+","+T2+","+"\n";

    }

    if(i==2){
```

```
                    str=Cadena.split(",");
                    Salida=Salida+T3+","+T4+","+T5+","+T6+","+T7+","+T8+"
                        ,"+T9+","+T10+","+"\n";

                }

                if(i==3){

                    str=Cadena.split(",");
                    Salida=Salida+g1+","+g2+","+g3+","+g4+","+str[4]+","+
                        str[5]+","+str[6]+","+str[7]+","+"\n";

                }

                if(i==0 || i>3){

                Salida=Salida+Cadena+"\n";
                }

                Cadena = Temp.readLine();
            }
            Seccion.close();

        }
        catch(FileNotFoundException fnfe) {}
        catch (IOException ioe) {}


        return Salida;
    }



}
```

Listing B.5: resi.java

```
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
import java.net.*;

/**
 * Simple demonstration for an Applet <-> Servlet communication.
 */
public class resi extends HttpServlet {
        /**
         * Get a String-object from the applet and send it back.
         */
        public void doPost(
                HttpServletRequest request,
                HttpServletResponse response)
                throws ServletException, IOException {
                try {
                        response.setContentType("application/x-java-
                            serialized-object");

                        // read a String-object from applet
                        // instead of a String-object, you can transmit any object, which
                        // is known to the servlet and to the applet
                        InputStream in = request.getInputStream();
                        ObjectInputStream inputFromApplet = new
                            ObjectInputStream(in);
                        String echo = (String) inputFromApplet.readObject
                            ();


                           String dia=generarRes(echo);


                        System.out.println(dia);
                        OutputStream outstr = response.getOutputStream();
                        ObjectOutputStream oos = new ObjectOutputStream(
                            outstr);
```

```
                oos.writeObject(dia);
                oos.flush();
                oos.close();


        } catch (Exception e) {
                e.printStackTrace();
        }
}


public boolean generaRes(){

File aRES = new File("../webapps/portal/Archivos/"+"res");


    try {


    if(aRES.exists()){
        aRES.delete();
        System.out.println("Res anterior eliminado");
   }


    Process p = Runtime.getRuntime().exec("../webapps/portal/
        Archivos/gen_new_M.out");

    InputStream is = p.getInputStream();

    BufferedReader br = new BufferedReader(new InputStreamReader(
        is));

    String aux = br.readLine();

    //System.out.println("aux"+aux);
```

```
            while (aux != null) {
                // Se escribe la linea en pantalla
                System.out.println(aux);
                // y se lee la siguiente.
                aux = br.readLine();
            }




        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("Excepcion del generaRes");
        }

            if(aRES.exists())
                return true;
            else
                return false;
    }

    private String generarRes(String nom) {


        FileReader AAbierto, AAbierto2, AAbierto3, AAbierto4;
        FileWriter aa=null;
        File AA;


        String Cadena = "";

        try{
                    AAbierto = new FileReader ("../webapps/portal/
                        Archivos/"+"res");
                    AAbierto2 = new FileReader ("../webapps/portal/
                        Archivos/"+"res");
```

```
AAbierto3 = new FileReader ("../webapps/portal/
    Archivos/"+"res");
AAbierto4 = new FileReader ("../webapps/portal/
    Archivos/"+"res");

aa = new FileWriter ("../webapps/portal/Archivos/"+"
    res_"+nom);

BufferedReader BAA = new BufferedReader(AAbierto);
BufferedReader BAA2 = new BufferedReader(AAbierto2);
BufferedReader BAA3 = new BufferedReader(AAbierto3);
BufferedReader BAA4 = new BufferedReader(AAbierto4);

for(int i=0;i<8;i++){
    Cadena = BAA.readLine();
    aa.write(Cadena+"\n");
}

Cadena = BAA.readLine();

String str[] = Cadena.split(",");
aa.write(str[0]+","+str[1]+",");

for(int i=0;i<13;i++){
    Cadena = BAA2.readLine();
}

str = Cadena.split(",");
aa.write(str[4]+","+str[5]+","+str[6]+","+str[7]+",")
    ;

for(int i=0; i<3;i++){
    Cadena = BAA2.readLine();
    str = Cadena.split(",");
    aa.write(str[0]+","+str[1]+"\n"+str[2]+","+str
        [3]+","+str[4]+","+str[5]+","+str[6]+","+str
        [7]+",");
```

```
        }

        Cadena = BAA2.readLine();
        str = Cadena.split(",");
        aa.write(str[0]+","+str[1]+"\n"+str[2]+","+str[3]+","
            +str[4]+","+str[5]+",");

        for(int i=0;i<9;i++){
            Cadena = BAA3.readLine();
       }

        str = Cadena.split(",");
        aa.write(str[2]+","+str[3]+","+str[4]+","+str[5]+"\n"
            +str[6]+","+str[7]+",");

        for(int i=0; i<3;i++){
            Cadena = BAA3.readLine();
            str = Cadena.split(",");
            aa.write(str[0]+","+str[1]+","+str[2]+","+str[3]+
                ","+str[4]+","+str[5]+"\n"+str[6]+","+str[7]+"
                ,");
        }

        Cadena = BAA3.readLine();
        str = Cadena.split(",");
        aa.write(str[0]+","+str[1]+","+str[2]+","+str[3]+"\n"
            );

        for(int i=0;i<17;i++){
            Cadena = BAA.readLine();

            if(i>7)
            aa.write(Cadena+"\n");
        }

        Cadena = BAA.readLine();
```

```
str = Cadena.split(",");
aa.write(str[0]+","+str[1]+",");

for(int i=0;i<14;i++){
    Cadena = BAA2.readLine();
}

str = Cadena.split(",");
aa.write(str[4]+","+str[5]+","+str[6]+","+str[7]+",")
    ;

for(int i=0; i<3;i++){
    Cadena = BAA2.readLine();
    str = Cadena.split(",");
    aa.write(str[0]+","+str[1]+"\n"+str[2]+","+str
        [3]+","+str[4]+","+str[5]+","+str[6]+","+str
        [7]+",");
}

Cadena = BAA2.readLine();
str = Cadena.split(",");
aa.write(str[0]+","+str[1]+"\n"+str[2]+","+str[3]+","
    +str[4]+","+str[5]+",");

for(int i=0;i<14;i++){
    Cadena = BAA3.readLine();
}

str = Cadena.split(",");
aa.write(str[2]+","+str[3]+","+str[4]+","+str[5]+"\n"
    +str[6]+","+str[7]+",");

for(int i=0; i<3;i++){
    Cadena = BAA3.readLine();
    str = Cadena.split(",");
```

```
                        aa.write(str[0]+","+str[1]+","+str[2]+","+str[3]+
                            ","+str[4]+","+str[5]+"\n"+str[6]+","+str[7]+"
                            ,");
                    }

                    Cadena = BAA3.readLine();
                    str = Cadena.split(",");
                    aa.write(str[0]+","+str[1]+","+str[2]+","+str[3]+"\n"
                        );

                    AAbierto.close();
                    aa.close();
                }
                catch(FileNotFoundException fnfe) {}
                catch (IOException ioe) {}


        return "";
    }



}
```

Listing B.6: resj.java

```
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
import java.net.*;

/**
 * Simple demonstration for an Applet <-> Servlet communication.
 */
public class resj extends HttpServlet {
        /**
         * Get a String-object from the applet and send it back.
         */
```

```
    public void doPost(
            HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
            try {
                    response.setContentType("application/x-java-
                        serialized-object");

                    // read a String-object from applet
                    // instead of a String-object, you can transmit any object, which
                    // is known to the servlet and to the applet
                    InputStream in = request.getInputStream();
                    ObjectInputStream inputFromApplet = new
                        ObjectInputStream(in);
                    String echo = (String) inputFromApplet.readObject
                        ();


                    String dia=generarRes(echo);


                    System.out.println(dia);
                    OutputStream outstr = response.getOutputStream();
                    ObjectOutputStream oos = new ObjectOutputStream(
                        outstr);
                    oos.writeObject(dia);
                    oos.flush();
                    oos.close();


            } catch (Exception e) {
                    e.printStackTrace();
            }
    }


public boolean generaRes(){
```

```
File aRES = new File("../webapps/portal/Archivos/"+"res");


    try {


    if(aRES.exists()){
        aRES.delete();
        System.out.println("Res anterior eliminado");
    }


    Process p = Runtime.getRuntime().exec("../webapps/portal/
        Archivos/gen_new_M.out");

    InputStream is = p.getInputStream();

    BufferedReader br = new BufferedReader(new InputStreamReader(
        is));

    String aux = br.readLine();

    while (aux != null) {
        // Se escribe la linea en pantalla
        System.out.println(aux);
        // y se lee la siguiente.
        aux = br.readLine();
    }




} catch (Exception e) {
    e.printStackTrace();
    System.out.println("Excepcion del generaRes");
}
```

```
        if(aRES.exists())
            return true;
        else
            return false;
    }

    private String generarRes(String nom) {


        FileReader AAbierto, AAbierto2, AAbierto3, AAbierto4;
        FileWriter aa=null;

        String Cadena = "";

        try{
                    AAbierto = new FileReader ("../webapps/portal/
                        Archivos/"+"res_"+nom);
                    AAbierto2 = new FileReader ("../webapps/portal/
                        Archivos/"+"res_"+nom);
                    AAbierto3 = new FileReader ("../webapps/portal/
                        Archivos/"+"res_"+nom);
                    AAbierto4 = new FileReader ("../webapps/portal/
                        Archivos/"+"res_"+nom);

                    aa = new FileWriter ("../webapps/portal/Archivos/"+"
                        res_"+nom+"_j");

                    BufferedReader BAA = new BufferedReader(AAbierto);
                    BufferedReader BAA2 = new BufferedReader(AAbierto2);
                    BufferedReader BAA3 = new BufferedReader(AAbierto3);
                    BufferedReader BAA4 = new BufferedReader(AAbierto4);

                    for(int i=0;i<8;i++){
                        Cadena = BAA.readLine();
                        aa.write(Cadena+"\n");
                    }
```

```
 Cadena = BAA.readLine();

 String str[] = Cadena.split(",");
 aa.write(str[0]+","+str[1]+",");

 for(int i=0;i<13;i++){
      Cadena = BAA2.readLine();
}

 str = Cadena.split(",");
 aa.write(str[4]+","+str[5]+","+str[6]+","+str[7]+",")
     ;

 for(int i=0; i<3;i++){
      Cadena = BAA2.readLine();
      str = Cadena.split(",");
      aa.write(str[0]+","+str[1]+"\n"+str[2]+","+str
          [3]+","+str[4]+","+str[5]+","+str[6]+","+str
          [7]+",");
 }

 Cadena = BAA2.readLine();
 str = Cadena.split(",");
 aa.write(str[0]+","+str[1]+"\n"+str[2]+","+str[3]+","
    +str[4]+","+str[5]+",");

 for(int i=0;i<9;i++){
      Cadena = BAA3.readLine();
}

 str = Cadena.split(",");
 aa.write(str[2]+","+str[3]+","+str[4]+","+str[5]+"\n"
    +str[6]+","+str[7]+",");

 for(int i=0; i<3;i++){
      Cadena = BAA3.readLine();
```

```
        str = Cadena.split(",");
        aa.write(str[0]+","+str[1]+","+str[2]+","+str[3]+
            ","+str[4]+","+str[5]+"\n"+str[6]+","+str[7]+"
            ,");
    }

    Cadena = BAA3.readLine();
    str = Cadena.split(",");
    aa.write(str[0]+","+str[1]+","+str[2]+","+str[3]+"\n"
        );

    for(int i=0;i<26;i++){
        Cadena = BAA4.readLine();
        if(i>16)
            aa.write(Cadena+"\n");
    }

    Cadena = BAA4.readLine();
    str = Cadena.split(",");
    aa.write(str[0]+","+str[1]+",");

    Cadena = BAA4.readLine();
    Cadena = BAA4.readLine();
    Cadena = BAA4.readLine();
    Cadena = BAA4.readLine();

    str = Cadena.split(",");
    aa.write(str[4]+","+str[5]+","+str[6]+","+str[7]+",")
        ;

    for(int i=0; i<3;i++){
        Cadena = BAA4.readLine();
        str = Cadena.split(",");
        aa.write(str[0]+","+str[1]+"\n"+str[2]+","+str
            [3]+","+str[4]+","+str[5]+","+str[6]+","+str
            [7]+",");
    }
```

```
                Cadena = BAA4.readLine();
                str = Cadena.split(",");
                aa.write(str[0]+","+str[1]+"\n"+str[2]+","+str[3]+","
                    +str[4]+","+str[5]+",");


                for(int i=0;i<14;i++)
                    Cadena = BAA3.readLine();

                str = Cadena.split(",");
                aa.write(str[2]+","+str[3]+","+str[4]+","+str[5]+"\n"
                    +str[6]+","+str[7]+",");

                for(int i=0; i<3;i++){
                    Cadena = BAA3.readLine();
                    str = Cadena.split(",");
                    aa.write(str[0]+","+str[1]+","+str[2]+","+str[3]+
                        ","+str[4]+","+str[5]+"\n"+str[6]+","+str[7]+"
                        ,");
                }
                Cadena = BAA3.readLine();
                str = Cadena.split(",");
                aa.write(str[0]+","+str[1]+","+str[2]+","+str[3]+"\n"
                    );

                AAbierto.close();
                aa.close();
            }
            catch(FileNotFoundException fnfe) {}
            catch (IOException ioe) {}

        return "True";
    }
```

```
}
```

Listing B.7: result.java

```
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
import java.net.*;
import java.util.Vector;
import javax.swing.table.DefaultTableModel;

/**
 * Simple demonstration for an Applet <-> Servlet communication.
 */
public class result extends HttpServlet {
        /**
         * Get a String-object from the applet and send it back.
         */
        public void doPost(
                HttpServletRequest request,
                HttpServletResponse response)
                throws ServletException, IOException {
                try {
                        response.setContentType("application/x-java-
                            serialized-object");

                        // read a String-object from applet
                        // instead of a String-object, you can transmit any object, which
                        // is known to the servlet and to the applet
                        InputStream in = request.getInputStream();
                        ObjectInputStream inputFromApplet = new
                            ObjectInputStream(in);
                        String echo = (String) inputFromApplet.readObject
                            ();

                        System.out.println(echo);
                        //
```

```
                    String aux[]=echo.split("#");

                    String Nombre=aux[0];
                    int tipo=Integer.parseInt(aux[1]);
                    int ele=Integer.parseInt(aux[2]);
                    int ex=Integer.parseInt(aux[3]);
                    int ey=Integer.parseInt(aux[4]);

                    String resul=leerResult(Nombre,tipo,ele,ex,ey);
                    System.out.println(resul);

                    OutputStream outstr = response.getOutputStream();
                    ObjectOutputStream oos = new ObjectOutputStream(
                        outstr);
                    oos.writeObject(resul);
                    oos.flush();
                    oos.close();


            } catch (Exception e) {
                    e.printStackTrace();
            }
    }

    public String leerResult(String nombre, int tipo,int ele,int ex,int
        ey){
        String result="";

        int NE=tipo;
        int EjeX=ex;
        int EjeY=ey;
        int Elem=ele;

        FileReader AAbierto;
        String Cadena = "";
        Vector <Double> VX = new Vector <Double> (), VY = new Vector <
            Double> ();
```

```
try{

 //System.out.println(NE);

if(NE==1)
    AAbierto = new FileReader ("../webapps/portal/Archivos/"+
        nombre+"_N.txt");
else
    AAbierto = new FileReader ("../webapps/portal/Archivos/"+
        nombre+"_E.txt");

BufferedReader BAA = new BufferedReader(AAbierto);

Cadena = BAA.readLine();
VX.add(0.0);
VY.add(0.0);

while(Cadena!=null){

Cadena=Cadena.replace("␣","");

//System.out.println(Cadena);
String str2[]=Cadena.split(",");

if(str2.length==1){

    //System.out.println(Double.parseDouble(str2[0]));
    if(EjeX==0){
            VX.add(Double.parseDouble(str2[0]));
            ////TagX="t";
    }

    if(EjeY==0){
            VY.add(Double.parseDouble(str2[0]));
            ////TagY="t";
    }
```

```
            }
            else
                if(NE==1){   /// Abrimos el Archivo donde se encuentran los resultados de
                    los nodos

                if(Integer.parseInt(str2[0])==Elem){

                    if(EjeX==1){
                        VX.add(Double.parseDouble(str2[1]));
                        ////TagX="U1";
                    }

                    if(EjeX==2){
                        VX.add(Double.parseDouble(str2[2]));
                        ////TagX="U2";
                    }

                    if(EjeX==3){
                        VX.add(Double.parseDouble(str2[3]));
                        ////TagX="U3";
                    }

                    if(EjeX==4){
                        VX.add(Double.parseDouble(str2[4]));
                        ////TagX="R1";
                    }

                    if(EjeX==5){
                        VX.add(Double.parseDouble(str2[5]));
                        ////TagX="R2";
                    }

                    if(EjeX==6){
                        VX.add(Double.parseDouble(str2[6]));
                        ////TagX="R3";
                    }

                    if(EjeX==7){
```

```
            VX.add(Double.parseDouble(str2[7]));
            ////TagX="F1";
        }

        if(EjeX==8){
            VX.add(Double.parseDouble(str2[8]));
            ////TagX="F2";
        }

        if(EjeX==9){
            VX.add(Double.parseDouble(str2[9]));
            ////TagX="F3";
        }

        if(EjeX==10){
            VX.add(Double.parseDouble(str2[10]));
            ////TagX="M1";
        }

        if(EjeX==11){
            VX.add(Double.parseDouble(str2[11]));
            ////TagX="M2";
        }

        if(EjeX==12){
            VX.add(Double.parseDouble(str2[12]));
            ////TagX="M3";
        }

        if(EjeY==1){
            VY.add(Double.parseDouble(str2[1]));
            ////TagY="U1";
        }

        if(EjeY==2){
            VY.add(Double.parseDouble(str2[2]));
            ////TagY="U2";
```

```
}

if(EjeY==3){
    VY.add(Double.parseDouble(str2[3]));
    ////TagY="U3";
}

if(EjeY==4){
    VY.add(Double.parseDouble(str2[4]));
    ////TagY="R1";
}

if(EjeY==5){
    VY.add(Double.parseDouble(str2[5]));
    ////TagY="R2";
}

if(EjeY==6){
    VY.add(Double.parseDouble(str2[6]));
    ////TagY="R3";
}

if(EjeY==7){
    VY.add(Double.parseDouble(str2[7]));
    ////TagY="F1";
}

if(EjeY==8){
    VY.add(Double.parseDouble(str2[8]));
    ////TagY="F2";
}

if(EjeY==9){
    VY.add(Double.parseDouble(str2[9]));
    ////TagY="F3";
}
```

```
                if(EjeY==10){
                    VY.add(Double.parseDouble(str2[10]));
                    ////TagY="M1";
                }

                if(EjeY==11){
                    VY.add(Double.parseDouble(str2[11]));
                    ////TagY="M2";
                }

                if(EjeY==12){
                    VY.add(Double.parseDouble(str2[12]));
                    ////TagY="M3";
                  }

                }

            } else{

               if(Integer.parseInt(str2[0])==Elem){

                if(EjeX==1){
                    VX.add(Double.parseDouble(str2[1]));
                    ////TagX='ŏ3A6'+"iy";
                }

                if(EjeX==2){
                    VX.add(Double.parseDouble(str2[2]));
                    ////TagX='ŏ3A6'+"jy";
                }

                if(EjeX==3){
                    VX.add(Double.parseDouble(str2[3]));
                    ////TagX=""+'ŏ3B4';
                }

                if(EjeX==4){
```

```
            VX.add(Double.parseDouble(str2[4]));
            ////TagX='ŏ3A6'+"iz";
        }

        if(EjeX==5){
            VX.add(Double.parseDouble(str2[5]));
            ////TagX='ŏ3A6'+"jz";
        }

        if(EjeX==6){
            VX.add(Double.parseDouble(str2[6]));
            ////TagX='ŏ3A6'+"x";
        }

        if(EjeX==7){
            VX.add(Double.parseDouble(str2[7]));
            ////TagX="Miy";
        }

        if(EjeX==8){
            VX.add(Double.parseDouble(str2[8]));
            ////TagX="Mjy";
        }

        if(EjeX==9){
            VX.add(Double.parseDouble(str2[9]));
            ////TagX="N";
        }

        if(EjeX==10){
            VX.add(Double.parseDouble(str2[10]));
            ////TagX="Miz";
        }

        if(EjeX==11){
            VX.add(Double.parseDouble(str2[11]));
            ////TagX="Mjz";
```

```
}

if(EjeX==12){
    VX.add(Double.parseDouble(str2[12]));
    ////TagX="Mx";
}

if(EjeX==13){
    VX.add(Double.parseDouble(str2[13]));
    ////TagX='&#03BB'+"i";
  }

if(EjeX==14){
    VX.add(Double.parseDouble(str2[14]));
    ////TagX='&#03BB'+"j";
  }

if(EjeX==15){
    VX.add(Double.parseDouble(str2[15]));
    ////TagX="diy+";
  }

if(EjeX==16){
    VX.add(Double.parseDouble(str2[16]));
    ////TagX="djy+";
  }

if(EjeX==17){
    VX.add(Double.parseDouble(str2[17]));
    ////TagX="diz+";
  }

if(EjeX==18){
    VX.add(Double.parseDouble(str2[18]));
    ////TagX="djz+";
  }
```

```
if(EjeX==19){
    VX.add(Double.parseDouble(str2[19]));
    ////TagX="diy-";
  }

if(EjeX==20){
    VX.add(Double.parseDouble(str2[20]));
    ////TagX="djy-";
  }

if(EjeX==21){
    VX.add(Double.parseDouble(str2[21]));
    ////TagX="diz-";
  }

if(EjeX==22){
    VX.add(Double.parseDouble(str2[22]));
    ////TagX="djz-";
  }

if(EjeX==23){
    VX.add(Double.parseDouble(str2[23]));
    ////TagX='ŏ3A6'+"iyp";
  }

if(EjeX==24){
    VX.add(Double.parseDouble(str2[24]));
    ////TagX='ŏ3A6'+"jyp";
  }

if(EjeX==25){
    VX.add(Double.parseDouble(str2[25]));
    ////TagX='ŏ3A6'+"izp";
  }

if(EjeX==26){
    VX.add(Double.parseDouble(str2[26]));
```

```
        ////TagX='Ŏ3A6'+"jzp";
    }

if(EjeX==27){
    VX.add(Double.parseDouble(str2[27]));
    ////TagX="Giy+";
    }



if(EjeX==28){
    VX.add(Double.parseDouble(str2[28]));
    ////TagX="Gjy+";
    }

if(EjeX==29){
    VX.add(Double.parseDouble(str2[29]));
    ////TagX="Giz+";
    }

if(EjeX==30){
    VX.add(Double.parseDouble(str2[30]));
    ////TagX="Gjz+";
    }

if(EjeX==31){
    VX.add(Double.parseDouble(str2[31]));
    ////TagX="Giy-";
    }

if(EjeX==32){
    VX.add(Double.parseDouble(str2[32]));
    ////TagX="Gjy-";
    }

if(EjeX==33){
    VX.add(Double.parseDouble(str2[33]));
    ////TagX="Giz-";
```

```
                }

                if(EjeX==34){
                    VX.add(Double.parseDouble(str2[34]));
                    ////TagX="Gjz-";
                }

        ////////////////////////////////////////
                if(EjeY==1){
                    VY.add(Double.parseDouble(str2[1]));
                    ////TagY='Ŏ3A6'+"iy";
                }

                if(EjeY==2){
                    VY.add(Double.parseDouble(str2[2]));
                    ////TagY='Ŏ3A6'+"jy";
                }

                if(EjeY==3){
                    VY.add(Double.parseDouble(str2[3]));
                    ////TagY=""+'Ŏ3B4';
                }

                if(EjeY==4){
                    VY.add(Double.parseDouble(str2[4]));
                    ////TagY='Ŏ3A6'+"iz";
                }

                if(EjeY==5){
                    VY.add(Double.parseDouble(str2[5]));
                    ////TagY='Ŏ3A6'+"jz";
                }

                if(EjeY==6){
                    VY.add(Double.parseDouble(str2[6]));
                    ////TagY='Ŏ3A6'+"x";
                }
```

```
if(EjeY==7){
    VY.add(Double.parseDouble(str2[7]));
    ////TagY="Miy";
}

if(EjeY==8){
    VY.add(Double.parseDouble(str2[8]));
    ////TagY="Mjy";
}

if(EjeY==9){
    VY.add(Double.parseDouble(str2[9]));
    ////TagY="N";
}

if(EjeY==10){
    VY.add(Double.parseDouble(str2[10]));
    ////TagY="Miz";
}

if(EjeY==11){
    VY.add(Double.parseDouble(str2[11]));
    ////TagY="Mjz";
}

if(EjeY==12){
    VY.add(Double.parseDouble(str2[12]));
    ////TagY="Mx";
  }

if(EjeY==13){
    VY.add(Double.parseDouble(str2[13]));
    ////TagY='03BB'+"i";
  }

if(EjeY==14){
```

```
        VY.add(Double.parseDouble(str2[14]));
        ////TagY='ŏ3BB'+"j";
      }

    if(EjeY==15){
        VY.add(Double.parseDouble(str2[15]));
        ////TagY="diy+";
      }

    if(EjeY==16){
        VY.add(Double.parseDouble(str2[16]));
        ////TagY="djy+";
      }

    if(EjeY==17){
        VY.add(Double.parseDouble(str2[17]));
        ////TagY="diz+";
      }

    if(EjeY==18){
        VY.add(Double.parseDouble(str2[18]));
        ////TagY="djz+";
      }

    if(EjeY==19){
        VY.add(Double.parseDouble(str2[19]));
        ////TagY="diy-";
      }

    if(EjeY==20){
        VY.add(Double.parseDouble(str2[20]));
        ////TagY="djy-";
      }

    if(EjeY==21){
        VY.add(Double.parseDouble(str2[21]));
        ////TagY="diz-";
```

```
    }

if(EjeY==22){
    VY.add(Double.parseDouble(str2[22]));
    ////TagY="djz-";
    }

if(EjeY==23){
    VY.add(Double.parseDouble(str2[23]));
    ////TagY='Ŏ3A6'+"iyp";
    }

if(EjeY==24){
    VY.add(Double.parseDouble(str2[24]));
    ////TagY='Ŏ3A6'+"jyp";
    }

if(EjeY==25){
    VY.add(Double.parseDouble(str2[25]));
    ////TagY='Ŏ3A6'+"izp";
    }

if(EjeY==26){
    VY.add(Double.parseDouble(str2[26]));
    ////TagY='Ŏ3A6'+"jzp";
    }

if(EjeY==27){
    VY.add(Double.parseDouble(str2[27]));
    ////TagY="Giy+";
    }


if(EjeY==28){
    VY.add(Double.parseDouble(str2[28]));
    ////TagY="Gjy+";
    }
```

```
                if(EjeY==29){
                    VY.add(Double.parseDouble(str2[29]));
                    ////TagY="Giz+";
                  }

                if(EjeY==30){
                    VY.add(Double.parseDouble(str2[30]));
                    ////TagY="Gjz+";
                  }

                if(EjeY==31){
                    VY.add(Double.parseDouble(str2[31]));
                    ////TagY="Giy-";
                  }

                if(EjeY==32){
                    VY.add(Double.parseDouble(str2[32]));
                    ////TagY="Gjy-";
                  }

                if(EjeY==33){
                    VY.add(Double.parseDouble(str2[33]));
                    ////TagY="Giz-";
                  }

                if(EjeY==34){
                    VY.add(Double.parseDouble(str2[34]));
                    ////TagY="Gjz-";
                  }

                }

            }

            Cadena = BAA.readLine();
        }
```

```
            AAbierto.close();

            // Se crea la tabla de Datos

            result="";
            for(int i=0; i<VX.size(); i++) {
                result=result+(VX.get(i)+"~"+VY.get(i))+"~";
            }



            }
            catch(FileNotFoundException fnfe) {}
            catch (IOException ioe) {}



          return result;
        }

}
```

Listing B.8: seccionTXTServlet.java

```
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
import java.net.*;

/**
 * Simple demonstration for an Applet <-> Servlet communication.
 */
public class seccionTXTServlet extends HttpServlet {
        /**
         * Get a String-object from the applet and send it back.
         */
```

```
public void doPost(
        HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        try {
                response.setContentType("application/x-java-
                    serialized-object");

                // read a String-object from applet
                // instead of a String-object, you can transmit any object, which
                // is known to the servlet and to the applet
                InputStream in = request.getInputStream();
                ObjectInputStream inputFromApplet = new
                    ObjectInputStream(in);
                String echo = (String) inputFromApplet.readObject
                    ();

                FileWriter fichero = null;
                PrintWriter pw = null;

                try{


                String nombreArchivo= "seccion.txt";

                fichero = new FileWriter("../webapps/portal/
                    Archivos/"+nombreArchivo);
               pw = new PrintWriter(fichero);

                pw.println(echo);
                System.out.println("Escribiendo seccion.txt ...")
                    ;
                fichero.close();


            } catch (Exception e) {
                  e.printStackTrace();
```

```
                } finally {
                try {
                        // Nuevamente aprovechamos el finally para
                        // asegurarnos que se cierra el fichero.
                        if (null != fichero)
                                fichero.close();
                        } catch (Exception e2) {
                                e2.printStackTrace();
                }
                }

                // echo it to the apple
                String dia="True";


        try {

                java.io.File aRES = new java.io.File("../webapps
                    /portal/Archivos/res");



                System.out.println("Eliminando res anterior...")
                    ;

                 if(aRES.exists()){
                   Runtime.getRuntime().exec("../webapps/portal/
                       Archivos/gen_new_M.out");



                System.out.println("Res actual creado...");
                dia="True";

                 }else
                      dia="False";
```

```
                } catch (Exception ee) {
                    ee.printStackTrace();
                }
                    System.out.println(dia);
                    OutputStream outstr = response.getOutputStream();
                    ObjectOutputStream oos = new ObjectOutputStream(
                        outstr);
                    oos.writeObject(dia);
                    oos.flush();
                    oos.close();


            } catch (Exception e) {
                    e.printStackTrace();
            }
    }


    public boolean generaRes(){

    File aRES = new File("../webapps/portal/Archivos/"+"res");


        try {


        if(aRES.exists()){
            aRES.delete();
            System.out.println("Res anterior eliminado");
       }


        Process p = Runtime.getRuntime().exec("../webapps/portal/
            Archivos/gen_new_M.out");

        InputStream is = p.getInputStream();
```

```
            BufferedReader br = new BufferedReader(new InputStreamReader(
                is));

            String aux = br.readLine();

            //System.out.println("aux"+aux);

            while (aux != null) {
                // Se escribe la linea en pantalla
                System.out.println(aux);
                // y se lee la siguiente.
                aux = br.readLine();
            }




        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("Excepcion del generaRes");
        }

            if(aRES.exists())
                return true;
            else
                return false;
    }


}
```

Apéndice C

Código del Elemento Finito Placa General desarrollado en MatLab

Listing C.1: rigidezMembrana.m

```
%
function [K]=rigidezMembrana(XY,E,v,t)

x1=XY(1,1);
y1=XY(1,2);
x2=XY(2,1);
y2=XY(2,2);
x3=XY(3,1);
y3=XY(3,2);

A=(x1*(y2-y3)+x2*(y3-y1)+x3*(y1-y2))/2;

B=[y2-y3  0  y3-y1  0  y1-y2  0;
    0  x3-x2  0  x1-x3  0  x2-x1;
    x3-x2  y2-y3  x1-x3  y3-y1  x2-x1  y1-y2]/(2*A);

%El=E/((1+v)*(1-2*v))*[1-v v 0;v 1-v 0; 0 0 (1-2*v)/2];
El=E/(1-v^2)*[1  v  0;  v  1  0;  0  0  (1-v)/2];

K=t*A*(B*El*B);
```

Listing C.2: rigidDKT.m

```
function [ke,Ae,Dk]=rigidDKT(x,E,nu,t);
%
%Funcion que obtiene la matriz de rigidez de un elemento DKT
%trabajano en 2 dimensiones (en el plano forjado)
%en coordenadas globales
%
%Salida:
% ke-¿ Matriz de rigidez del elemento DKT
% Ae-¿Area del elemento
%
%Recibe como Datos:
% x-¿ matriz con las coordenadas de los nodos del elemento
% E-¿ modulos de elasticidad del material
% nu-¿ coeficiente de poisson del material
% t-¿ espesor del material
%

if nargin ~=4,
    error(Número de Argumentos incorrectos, compruebe los datos de entrada);
end

%Numero de nodos por elemeto y la dimension del problema
%a partir de las dimensiones de la matriz de coordenadas x

[nne,dim]=size(x);
if nne ~=3,
    error(El elemento DKT ha de tener tres nodos);
end

%El numero de grados de libertad es igual a 3
ngn=3;

%Matriz constitutiva
Dk=mcons(E,nu,t)

%Valores Auxiliares
```

```
KOD=[1  2  3  4  5  6  7  8  9;
     1  3  2  4  6  5  7  9  8];

 PP=[12  4  4;
      4  2  1;
      4  1  2];

  BB=zeros(nne,1);
  CC=zeros(nne,1);

  BB(1)=x(2,2)-x(3,2); BB(2)=x(3,2)-x(1,2); BB(3)=x(1,2)-x(2,2)
  CC(1)=x(3,1)-x(2,1); CC(2)=x(1,1)-x(3,1); CC(3)=x(2,1)-x(1,1)

  DET=(BB(1)*CC(2)-BB(2)*CC(1))*24

  DD=zeros(nne*ngn);

   for i= 1:nne,
      for j= 1:nne,
         for k1= 1:nne,
              ii=(i-1)*nne+k1;
              for k2=1:nne,
                   jj=(j-1)*nne+k2;
                   DD(ii,jj)=Dk(i,j)*PP(k1,k2)/DET;
              end
         end
      end
   end


DD

ALS=zeros(nne,1);
RS=zeros(2,nne);

for i=1:nne,
    ALS(i)=BB(i)*BB(i)+CC(i)*CC(i);
```

```
74      PT(1,i)=6*CC(i)/ALS(i);
75      PT(2,i)=6*BB(i)/ALS(i);
76      RS(1,i)=3*CC(i)^2/ALS(i);
77      RS(2,i)=3*BB(i)^2/ALS(i);
78      Q(i)=3*BB(i)*CC(i)/ALS(i);
79  end
80
81  ALS
82  PT
83  RS
84  Q
85
86  GG=zeros(nne*ngn+1,nne*ngn);
87
88  for  i=1:2,
89       ii=(i-1)*5;
90      GG(ii+1,KOD(i,1))=PT(i,3);
91      GG(ii+2,KOD(i,1))=-PT(i,2);
92      GG(ii+3,KOD(i,1))=-PT(i,3);
93      GG(ii+4,KOD(i,1))=PT(i,2)-PT(i,3);
94      GG(ii+5,KOD(i,1))=PT(i,2);
95      GG(ii+1,KOD(i,2))=-Q(3);
96      GG(ii+2,KOD(i,2))=-Q(2);
97      GG(ii+3,KOD(i,2))=Q(3);
98      GG(ii+4,KOD(i,2))=Q(2)+Q(3);
99      GG(ii+5,KOD(i,2))=Q(2);
100     GG(ii+1,KOD(i,3))=-1-RS(i,3);
101     GG(ii+2,KOD(i,3))=-1-RS(i,2);
102     GG(ii+3,KOD(i,3))=RS(i,3);
103     GG(ii+4,KOD(i,3))=RS(i,2)+RS(i,3);
104     GG(ii+5,KOD(i,3))=RS(i,2);
105     GG(ii+1,KOD(i,3))=-1-RS(i,3);
106     GG(ii+1,KOD(i,4))=-PT(i,3);
107     GG(ii+3,KOD(i,4))=PT(i,3);
108     GG(ii+4,KOD(i,4))=PT(i,1)+PT(i,3);
109     GG(ii+1,KOD(i,5))=-Q(3);
110     GG(ii+3,KOD(i,5))=Q(3);
```

```
111       GG(ii+4,KOD(i,5))=Q(3)-Q(1);
112       GG(ii+1,KOD(i,6))=1-RS(i,3);
113       GG(ii+3,KOD(i,6))=RS(i,3);
114       GG(ii+4,KOD(i,6))=RS(i,3)-RS(i,1);
115       GG(ii+2,KOD(i,7))=PT(i,2);
116       GG(ii+4,KOD(i,7))=-PT(i,1)-PT(i,2);
117       GG(ii+5,KOD(i,7))=-PT(i,2);
118       GG(ii+2,KOD(i,8))=-Q(2);
119       GG(ii+4,KOD(i,8))=Q(2)-Q(1);
120       GG(ii+5,KOD(i,8))=Q(2);
121       GG(ii+2,KOD(i,9))=1-RS(i,2);
122       GG(ii+4,KOD(i,9))=RS(i,2)-RS(i,1);
123       GG(ii+5,KOD(i,9))=RS(i,2);
124 end
125
126 GG
127
128 QQ=zeros(nne*ngn);
129
130 for i=1:nne*ngn,
131       QQ(1,i)=BB(2)*GG(1,i)+BB(3)*GG(2,i);
132       QQ(2,i)=2*BB(2)*GG(3,i)+BB(3)*GG(4,i);
133       QQ(3,i)=BB(2)*GG(4,i)+2*BB(3)*GG(5,i);
134       QQ(4,i)=-CC(2)*GG(6,i)-CC(3)*GG(7,i);
135       QQ(5,i)=-2*CC(2)*GG(8,i)-CC(3)*GG(9,i);
136       QQ(6,i)=-CC(2)*GG(9,i)-2*CC(3)*GG(10,i);
137       QQ(7,i)=CC(2)*GG(1,i)+CC(3)*GG(2,i)-BB(2)*GG(6,i)-BB(3)*GG(7,i);
138       QQ(8,i)=2*CC(2)*GG(3,i)+CC(3)*GG(4,i)-2*BB(2)*GG(8,i)-BB(3)*GG(9,i);
139       QQ(9,i)=CC(2)*GG(4,i)+2*CC(3)*GG(5,i)-BB(2)*GG(9,i)-2*BB(3)*GG(10,i);
140 end
141
142 QQ
143
144 SS=zeros(nne*ngn);
145
146 for i= 1:nne*ngn,
147       for j=1:nne*ngn,
```

```
148             for i1=1:nne*ngn,
149                 SS(i,j)=SS(i,j)+DD(i,i1)*QQ(i1,j);
150             end
151         end
152 end
153
154 SS
155 %Matriz de Rigidez del elemento DKT en globales
156
157 ke=zeros(nne*ngn);
158
159 for i=1:nne*ngn,
160     for j=1:nne*ngn,
161         for i1=1:nne*ngn,
162             ke(i,j)=ke(i,j)+QQ(i1,i)*SS(i1,j);
163         end
164     end
165 end
166
167 ke
168 %Calculo del area del elemento mediante la formula del determinante
169
170 if nargout == 2,
171     Ae=abs(1/2*(det([x ones(nne,1)])));
172 end
```

Listing C.3: rigidezCombinado.m

```
1 function [K,Ae]=rigidezCombinado(x,E,v,t)
2
3 Ae=(x(1,1)*(x(2,2)-x(3,2))+x(2,1)*(x(3,2)-x(1,2))+x(3,1)*(x(1,2)-x(2,2)))
      /2;
4
5 KEM=rigidezMembrana(x,E,v,t)
6 KER=rigidDKT(x,E,v,t)
7
8
9 K11=[KEM(1,1) KEM(1,2) 0 0 0;
10       KEM(2,1) KEM(2,2) 0 0 0;
```

```
11        0  0 KER(1,1) KER(1,2) KER(1,3);
12        0  0 KER(2,1) KER(2,2) KER(2,3);
13        0  0 KER(3,1) KER(3,2) KER(3,3);
14       ];
15
16 K12=[KEM(1,3) KEM(1,4)  0  0  0;
17        KEM(2,3) KEM(2,4)  0  0  0;
18        0  0 KER(1,4) KER(1,5) KER(1,6);
19        0  0 KER(2,4) KER(2,5) KER(2,6);
20        0  0 KER(3,4) KER(3,5) KER(3,6);
21       ];
22
23 K13=[KEM(1,5) KEM(1,6)  0  0  0;
24        KEM(2,5) KEM(2,6)  0  0  0;
25        0  0 KER(1,7) KER(1,8) KER(1,9);
26        0  0 KER(2,7) KER(2,8) KER(2,9);
27        0  0 KER(3,7) KER(3,8) KER(3,9);
28       ];
29
30 K21=[KEM(3,1) KEM(3,2)  0  0  0;
31        KEM(4,1) KEM(4,2)  0  0  0;
32        0  0 KER(4,1) KER(4,2) KER(4,3);
33        0  0 KER(5,1) KER(5,2) KER(5,3);
34        0  0 KER(6,1) KER(6,2) KER(6,3);
35       ];
36
37 K22=[KEM(3,3) KEM(3,4)  0  0  0;
38        KEM(4,3) KEM(4,4)  0  0  0;
39        0  0 KER(4,4) KER(4,5) KER(4,6);
40        0  0 KER(5,4) KER(5,5) KER(5,6);
41        0  0 KER(6,4) KER(6,5) KER(6,6);
42       ];
43
44 K23=[KEM(3,5) KEM(3,6)  0  0  0;
45        KEM(4,5) KEM(4,6)  0  0  0;
46        0  0 KER(4,7) KER(4,8) KER(4,9);
47        0  0 KER(5,7) KER(5,8) KER(5,9);
```

```
48        0  0 KER(6,7) KER(6,8) KER(6,9);
49       ];
50
51 K31=[KEM(5,1) KEM(5,2)  0  0  0;
52       KEM(6,1) KEM(6,2)  0  0  0;
53        0  0 KER(7,1) KER(7,2) KER(7,3);
54        0  0 KER(8,1) KER(8,2) KER(8,3);
55        0  0 KER(9,1) KER(9,2) KER(9,3);
56       ];
57
58 K32=[KEM(5,3) KEM(5,4)  0  0  0;
59       KEM(6,3) KEM(6,4)  0  0  0;
60        0  0 KER(7,4) KER(7,5) KER(7,6);
61        0  0 KER(8,4) KER(8,5) KER(8,6);
62        0  0 KER(9,4) KER(9,5) KER(9,6);
63       ];
64
65 K33=[KEM(5,5) KEM(5,6)  0  0  0;
66       KEM(6,5) KEM(6,6)  0  0  0;
67        0  0 KER(7,7) KER(7,8) KER(7,9);
68        0  0 KER(8,7) KER(8,8) KER(8,9);
69        0  0 KER(9,7) KER(9,8) KER(9,9);
70       ];
71
72 K=[K11  K12  K13;
73     K21  K22  K23;
74     K31  K32  K33;];
```

Listing C.4: mcons.m

```
1 function Dk=mcons(E,nu,t)
2
3 D=E*t^3/(12*(1-nu^2))
4
5 Dk=[D  nu*D  0;
6       nu*D D  0;
7       0  0  D*(1-nu)/2];
```

Listing C.5: SEF1.m

```
1
2 clc
3 clear all
4 close all
5
6 x=[0 0; 10 0; 0 10];
7
8 %KE=rigidezDKT(x,215000,0.2,1)
9
10 E= 215000;
11 v=0.2;
12 t=1;
13
14 [K,AE]=rigidezCombinado(x,E,v,t);
15
16 q=-1;
17
18 syms u1 v1 w1 tx1 ty1 u2 v2 w2 tx2 ty2 u3 v3 w3 tx3 ty3;
19 U=[0 0 0 0 0 0 0 0 0 0 u3 v3 w3 tx3 ty3];P=(q*AE/3)*[0 0 1 0 0 0 0 1 0 0 0 0 1 0 0];
20
21 syms Fu1 Fu2 Fu3 Fv1 Fv2 Fv3 Fw1 Fw2 Fw3 Mx1 Mx2 Mx3 My1 My2 My3;
22 F=[Fu1 Fv1 Fw1 Mx1 My1 Fu2 Fv2 Fw2 Mx2 My2 10 10 0 0 0];Q=(K*U-P);Q-F
```

Apéndice D

Tutorial para el Uso del Cliente PC

El presente tutorial corresponde a un ejemplo utilizando el Portal de Pórticos 3D, PDP3D como herramienta de Analisis. Este tutorial es tomado de TERÁN TARAZONA (2014), incluido como aporte para el proceso de depuración del PDP3D.

Tutorial

Se presenta una estructura aporticada de concreto armado que consta de 2 niveles, con una altura en el primer piso de $2.5m$ y en el segundo de $2.2m$, un tramo en la dirección X con una longitud de $2m$ y un tramo en la dirección Y de longitud $2m$.

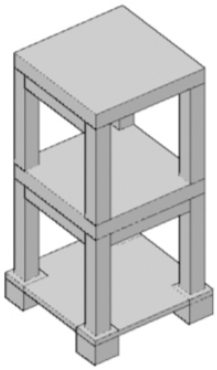

Figura D.1: Vista 3D del ejemplo

Para crear el nuevo proyecto, hacemos click en Nuevo, y le asignamos el nombre del proyecto. Presionamos el botón “Aceptar”.

Luego asignamos la geometría del modelo, agregamos 2 niveles, el primero de $250cm$ y el segundo de $220cm$. Luego agregamos un tramo en X, y uno en Y de $200cm$.

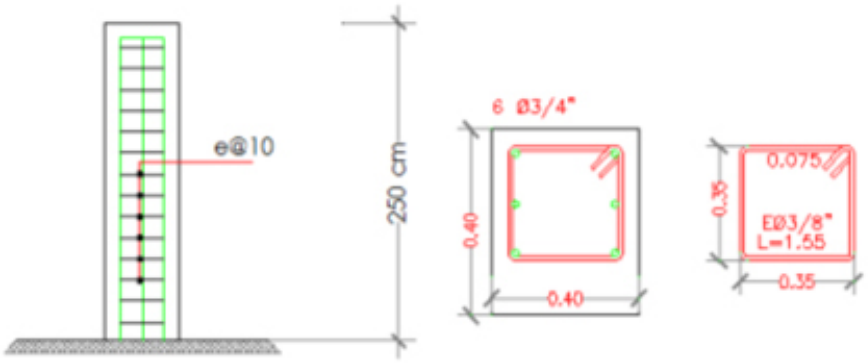

Figura D.2: Sección para Columnas del primer piso

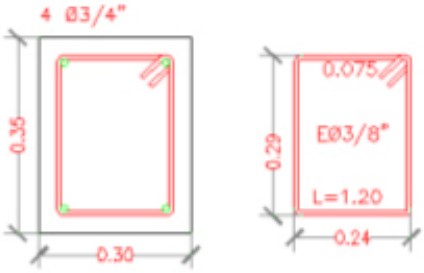

Figura D.3: Sección para Columnas del segundo piso

Dejamos el botón de placa activada. Al hacer esto debemos oprimir el botón “Aceptar” (Ver figura D.6).

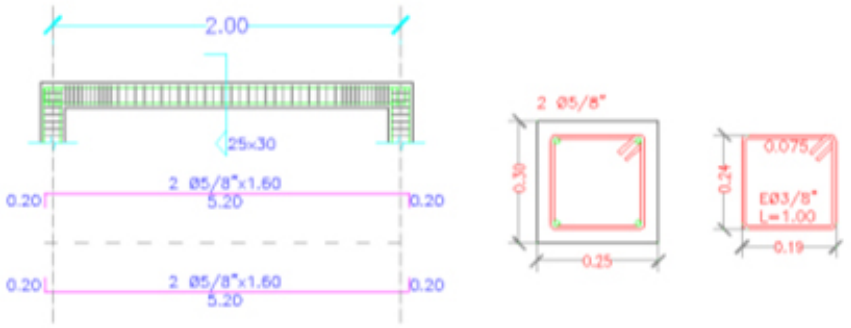

Figura D.4: Sección para Vigas en ambos pisos y ambas direcciones

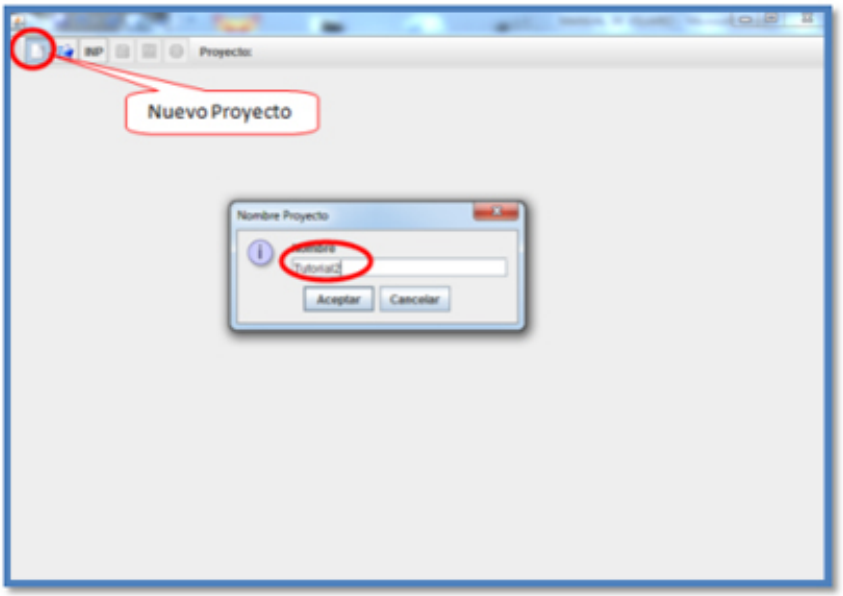

Figura D.5: Vista del nuevo proyecto “Tutorial”

Luego asignamos 4 grupos:

Un grupo para las columnas del primer piso. Un grupo para las del segundo piso.

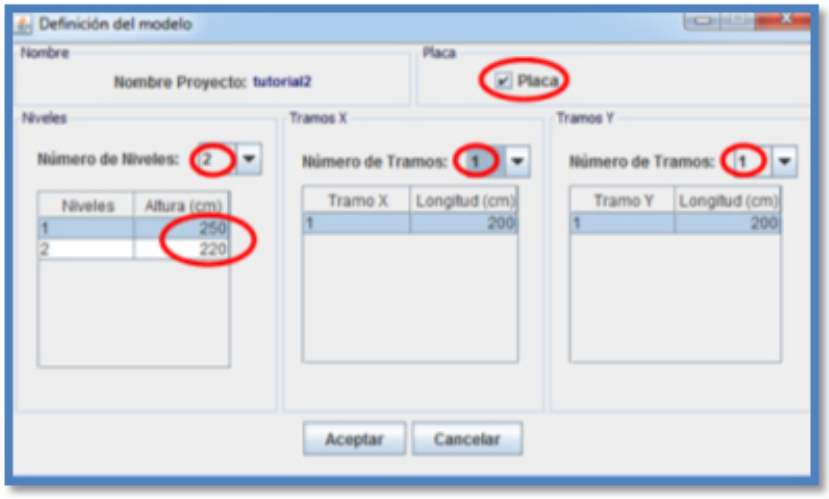

Figura D.6: Geometría del ejemplo

Un grupo para las vigas en X. Por último un grupo para las vigas en Y. Para crear el grupo hacemos click en "Seleccionar E." y luego añadimos el conjunto de elementos haciendo click en el símbolo "+". Para asignar los elemento hacemos doble click en el espacio vacío debajo de elementos el cual abrirá una nueva ventana se selección.

El programa asignara automáticamente un numero para cada elemento, debemos seleccionar los elementos según su número y su grupo correspondiente. Al finalizar se debe oprimir "OK" y luego "Aceptar".

Luego debemos asignar las placas, haciendo el mismo procedimiento anterior (Ver figura D.8)

Después se activa la pestaña "Secciones" donde debemos crear y asignar las secciones que tendrá la estructura. Para hacer esto debemos hacer click en "Sección E." donde debe aparecer una nueva ventana. Para crear la sección se debe oprimir el botón "Nuevo".

Al hacer click en "Nuevo" aparecerá una nueva ventana donde salen las distintas características de la sección.

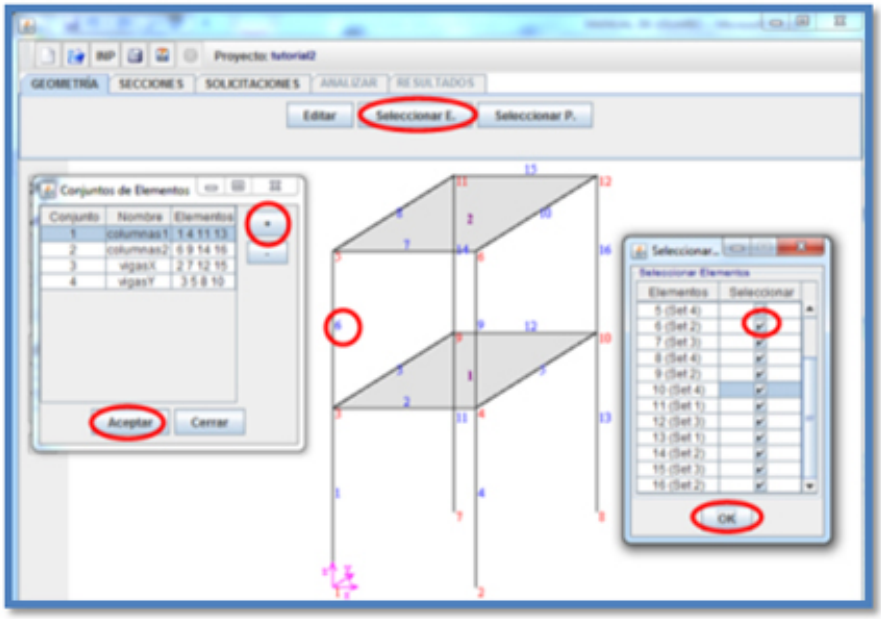

Figura D.7: Asignación de conjunto de elementos

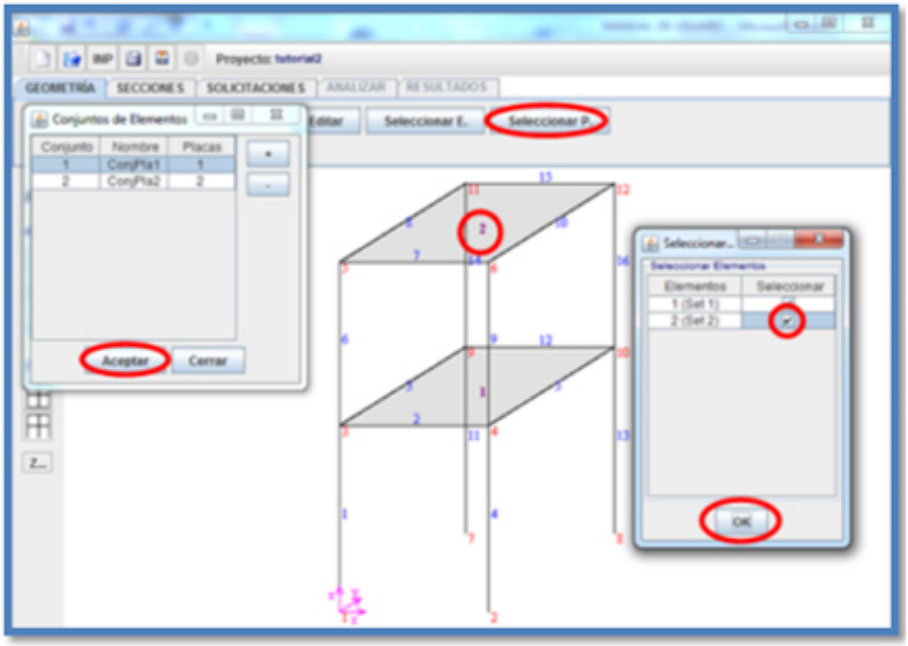

Figura D.8: Asignación de conjunto de placas

En la pantalla Definir Sección se proporciona la información referente a los elementos columnas (Ver figuras D.13, D.12, D.11 y D.10)

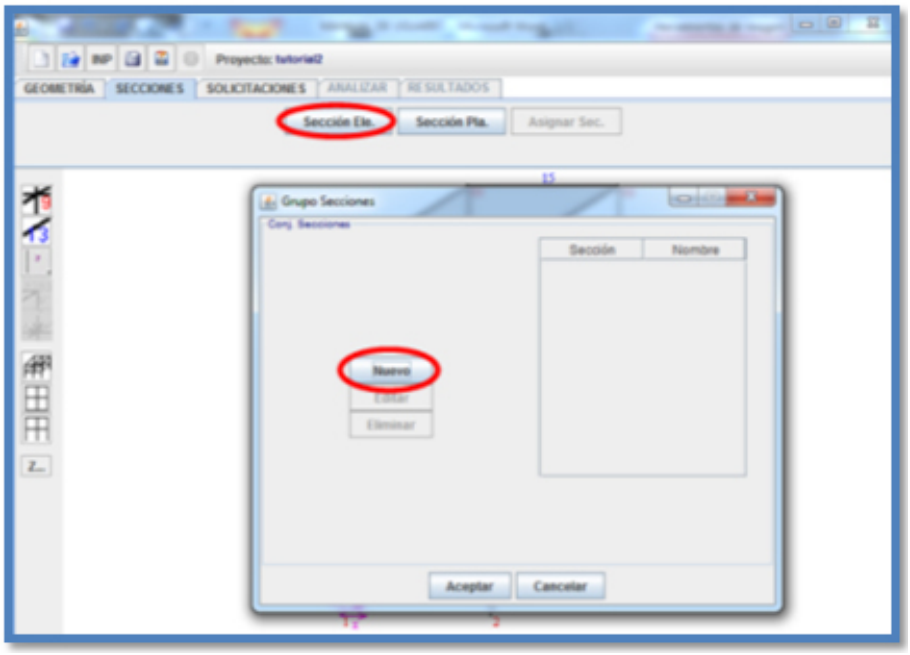

Figura D.9: Ventana de grupo de secciones

Nombre de la sección: Nombre asignado a la sección, "Columna1".

Tipo de sección: Se selecciona el tipo de sección (Columna, viga X, viga Y, otro), "Columna".

Área: Se definen las dimensiones de la sección (b, h) según la vista mostrada, "$b = 40cm$, $h = 40cm$".

Masa: Usaremos el peso propio de cada elemento que será de $9.78E{-}4(TN{*}Seg^2)/cm$.

Aceros Longitudinales: En los Aceros Longitudinales se define primero el número de capas, nuestro número de capas serán "2", y serán 3 barras para cada capa, el cual tendrán cada una un área de acero de $8.55cm^2$. La altura para la primera capa será de $3.3983cm$ y para la segunda capa será de $36.6cm$ ambos con un recubrimiento de $3.3983cm$.

Estribos: En la opción de Estribos se introducen los datos correspondientes al acero transversal de los elementos.En la opción Estribos se indica el tipo de estribo, seleccionando la opción "A". Donde: rh: $2.5cm$ rv: $2.5cm$ dE: $0.953cm$ s: $10cm$

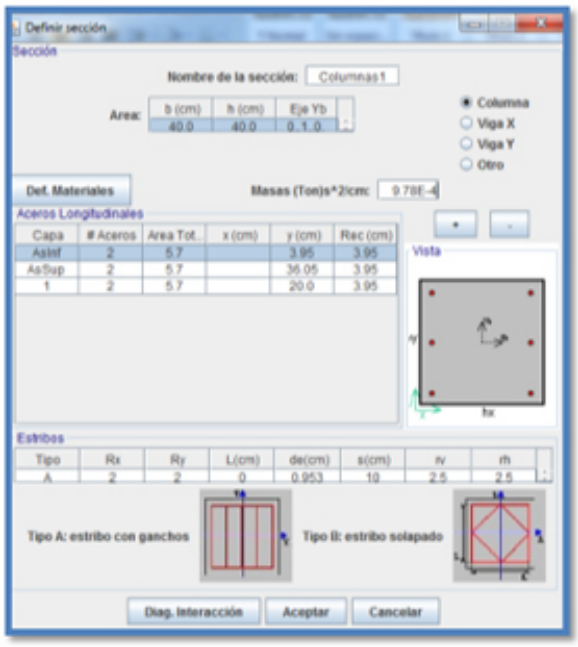

Figura D.10: Sección de las columnas del primer piso

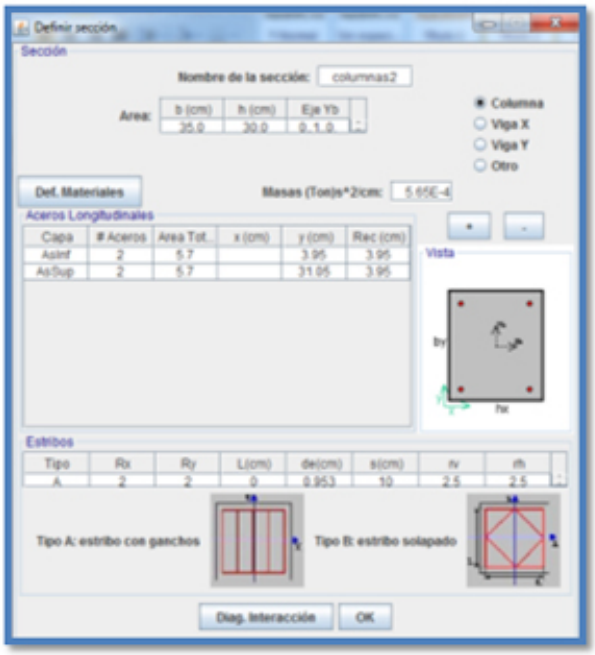

Figura D.11: Sección de las columnas del segundo piso

Y para las características de los materiales, usaremos los valores predeterminados del programa (Ver figura D.14).

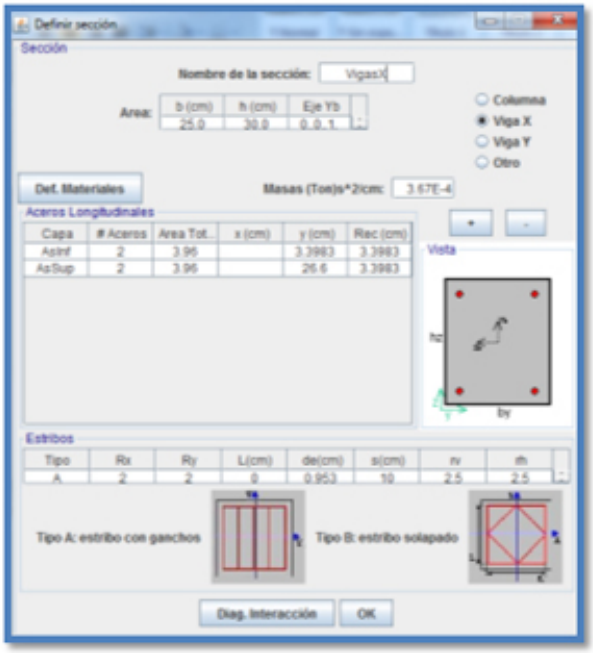

Figura D.12: Sección de las Vigas en dirección x

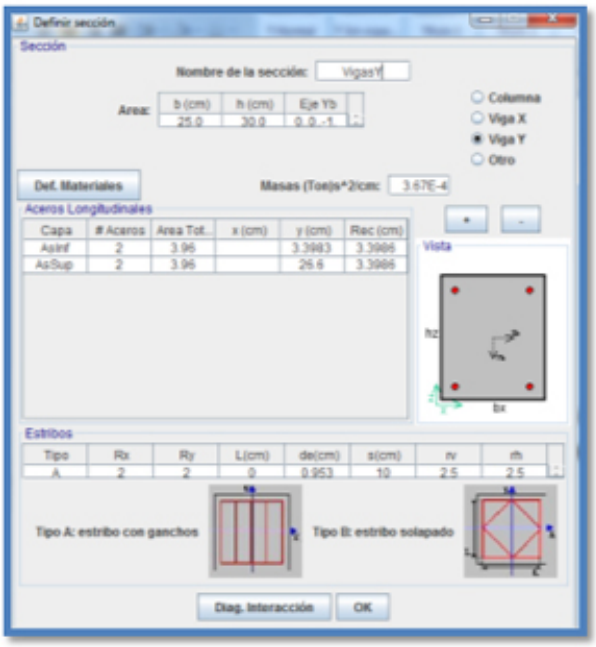

Figura D.13: Sección de las Vigas en dirección Y

Luego creamos una sección para ambas placas haciendo click en “Seccion Pla.”. Después aparecerá una ventana “Conj Secciones de Placas” donde debemos oprimir el

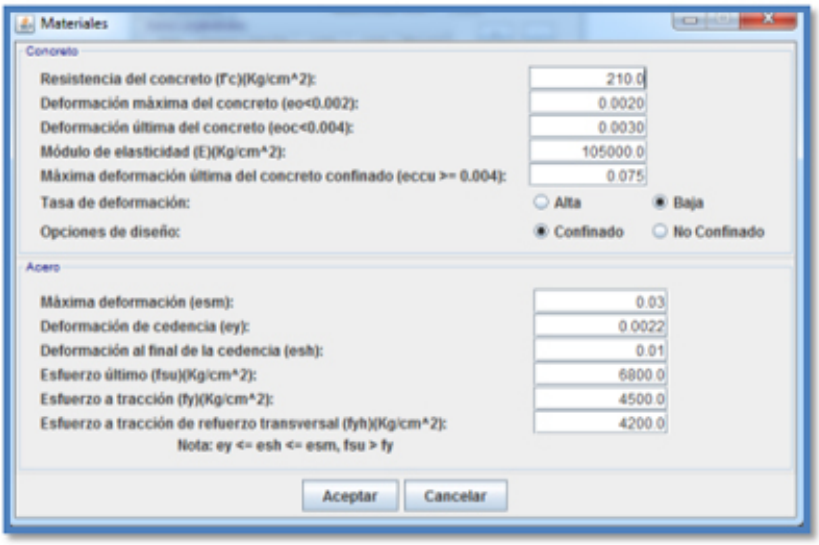

Figura D.14: Características de los materiales utilizados

botón "Nuevo" al oprimirlo surgirá una segunda ventana donde asignaremos el espesor de la placa y sus propiedades (Ver figura D.15).

El siguiente paso es asignar las secciones a los elementos, haciendo click en el botón "Asignar Sec." Lo que desplegará una ventana donde seleccionamos las secciones creadas anteriormente, según el grupo correspondiente. Al finalizar se debe hacer click en "Aceptar".

Después debemos asignar las solicitaciones. Primero asignamos grupos de nodos para cada piso. Para crear los grupos y asignarlos hacemos click en "Sel. Nodos" el cual desplegara una nueva ventana llamada "Conjunto de Nodos".

Para crear los grupos se le da al botón que tiene el símbolo "+", y luego para asignar los nodos hacemos doble click en el espacio vacío debajo de la columna de "Nodos", luego en la nueva ventana asignaremos los nodos correspondientes a cada piso.

Se debe tomar en cuenta la placa para q todo el piso se mueva en conjunto. El programa asigna para las 2 placas, los 2 números finales, en este caso 13 para la placa del primer piso y 14 para la del segundo piso. Al finalizar se debe hacer click en "OK" y luego en "Aceptar".

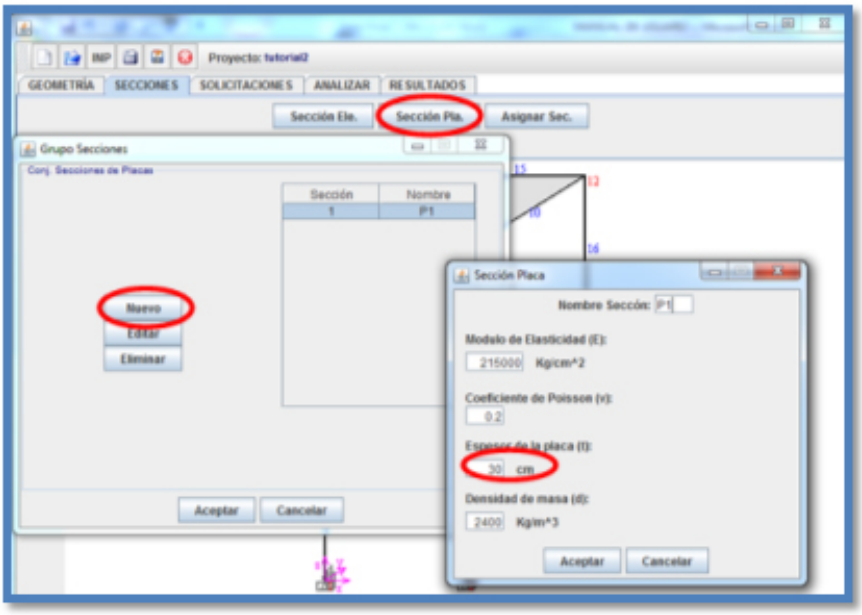

Figura D.15: Sección de las placas del ejemplo

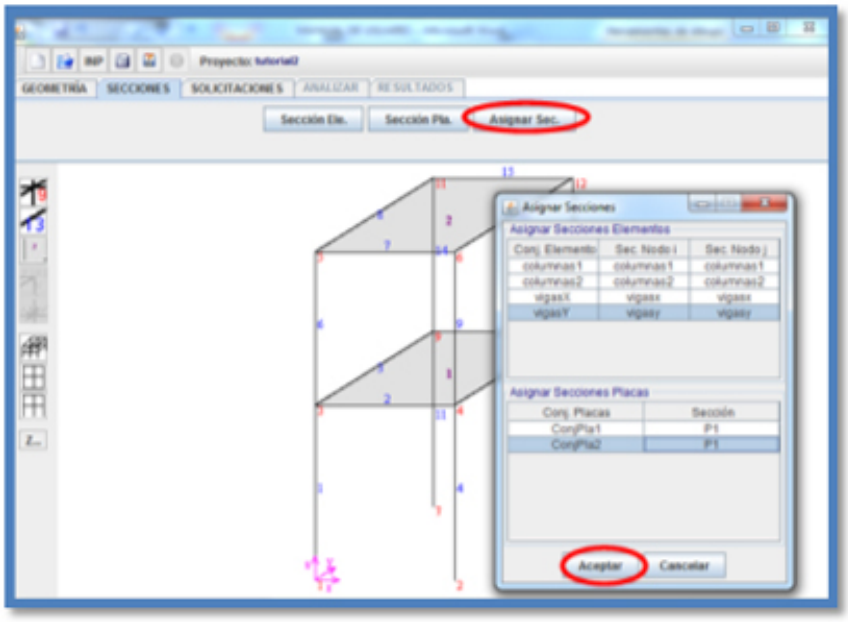

Figura D.16: Asignación de secciones

Luego para asignar las restricciones que tendrán cada conjunto de nodos. Para

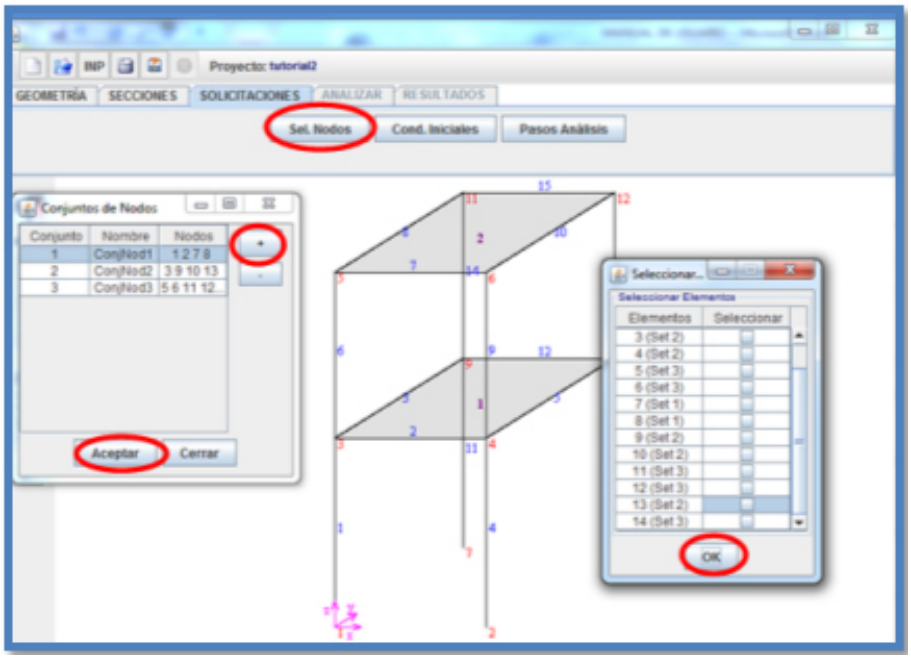

Figura D.17: Asignación de conjunto de nodos para el ejemplo

hacerlo le damos click en el botón "Cond. Iniciales" y en la ventana siguiente seleccionamos todos los movimientos ya que es una estructura aporticada. Al finalizar hacemos click en "Aceptar".

El siguiente paso es introducir pasos de análisis haciendo click en el botón "Pasos Análisis" el cual desplegara una nueva ventana llamada "Pasos" para crear un nuevo paso se oprime el botón "Nuevo" (Ver figura D.19).

En la ventana "Configuración de Paso" primero debemos verificar el incremento inicial y la duración del paso, que para este caso será de $1Seg$ y de $10Seg$. Luego para introducir la fuerza en la placa hacemos click en el símbolo "+" para crear el paso, seleccionamos la placa a la que será sometida la fuerza, para nuestro ejemplo será el conjunto de nodos 2, después hacemos doble click en el espacio vacío debajo de la columna de Fuerzas Distribuidas (Ton/cm^2) (Ver figura D.20).

Después en la pequeña ventana llamada "Fuerzas en Elementos (Ton)" en la columna "Fx" se debe introducir la fuerza de $-6.5E-5TN$. Al terminar de teclear debemos pulsar el botón "ENTER" o "INTRO" para que sea introducido correctamente, luego hacemos click en la opción "Aceptar" en la ventana de "Fuerzas de Nodos"

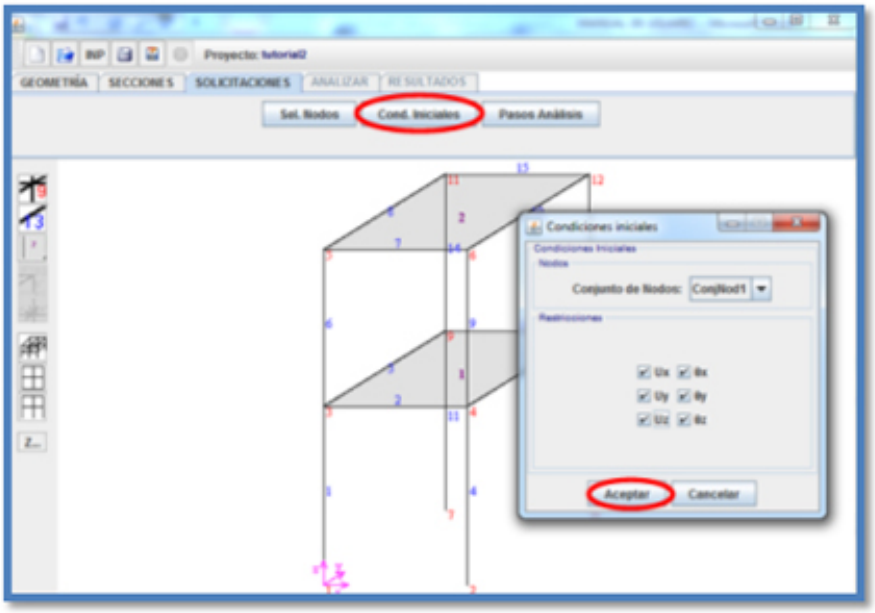

Figura D.18: Condiciones iniciales para el ejemplo

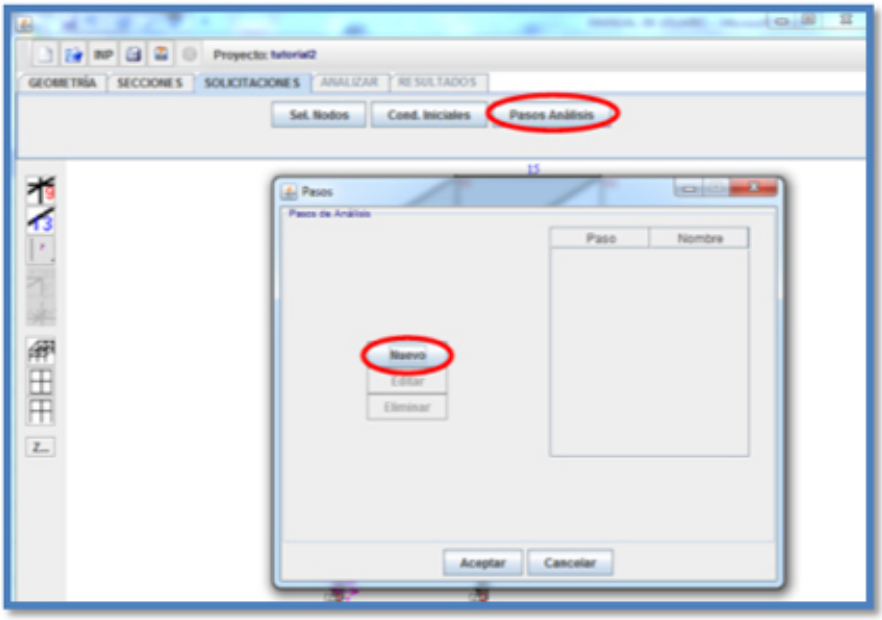

Figura D.19: Ventana de Pasos de análisis

y en la ventana “Configuración Paso” (Ver figura D.21).

Para introducir el archivo de sismo, repetimos el mismo paso anterior, hacemos click

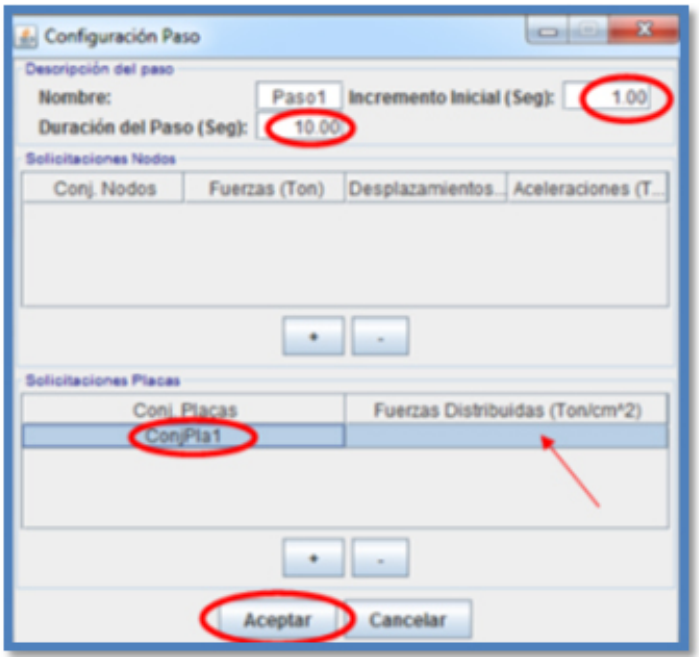

Figura D.20: Configuración del paso 1 para el ejemplo

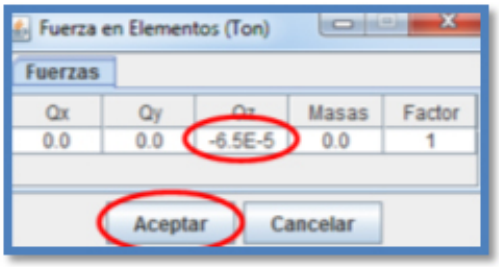

Figura D.21: Asignación de la fuerza para el paso 1 del ejemplo

en el botón "Nuevo" el cual creara el "Paso 2", verificamos los valores de incremento inicial y duración del paso esta vez serán de $0.01Seg$ y de $29Seg$ respectivamente, luego

seleccionamos nuevamente el conjunto de nodos 1 y hacemos doble click en la columna vacía debajo de "Aceleraciones" (Ver figura D.22).

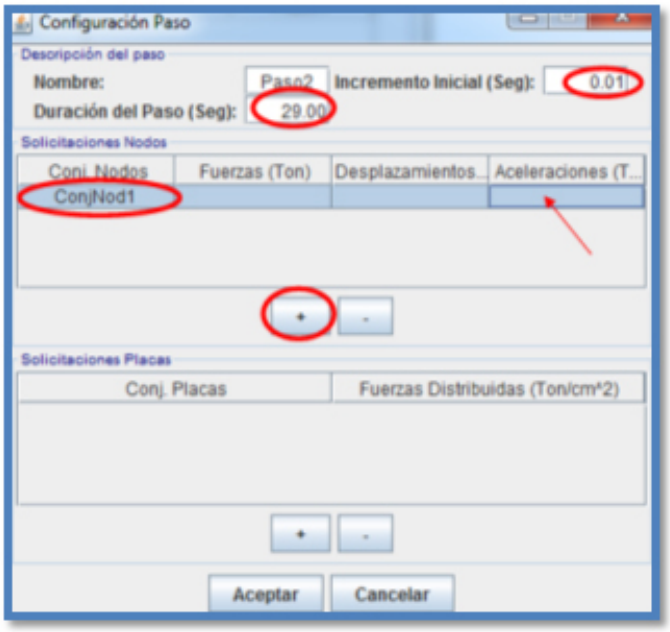

Figura D.22: Configuración del paso 2 para el ejemplo

En la ventana emergente llamada "Aceleración de Nodos", para subir el archivo, hacemos click en la pestaña "Archivo", seleccionamos Ax, para que aplique el sismo en esa dirección, luego introducimos el valor de sismo que será de 981. Luego hacemos click en el botón "Subir" y buscamos el archivo que debe estar en la misma carpeta del programa y presionamos el botón "Abrir" +++++++++(Ver figuras D.23 y D.24).

Y finalizamos haciendo click en "Aceptar" en ambas ventanas y en la ventana de "Pasos"

Para el análisis del elemento debemos pulsar "Guardar" donde debemos indicar el lugar donde se quiera almacenar el archivo. Luego de guardar el archivo, este automáticamente abrirá una ventana emergente que indica que se ha guardado correctamente y se habilitara la pestaña "Analizar".

NOTA: Es recomendable guardar los archivos en la misma carpeta del programa.

Luego de guardado, hacemos click en "Analizar" y este abrirá una ventana emergente donde mostrara el proceso de simulación indicando el porcentaje de ejecución.

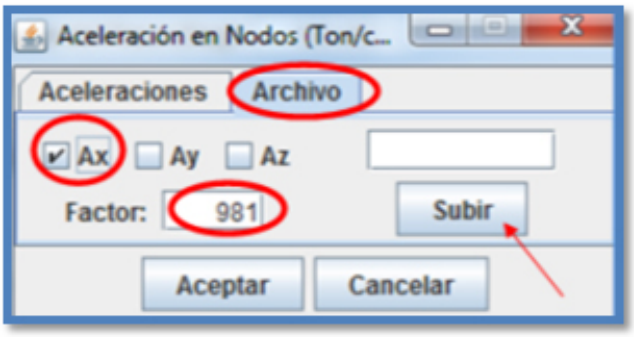

Figura D.23: Asignación del sismo para el paso 2 del ejemplo

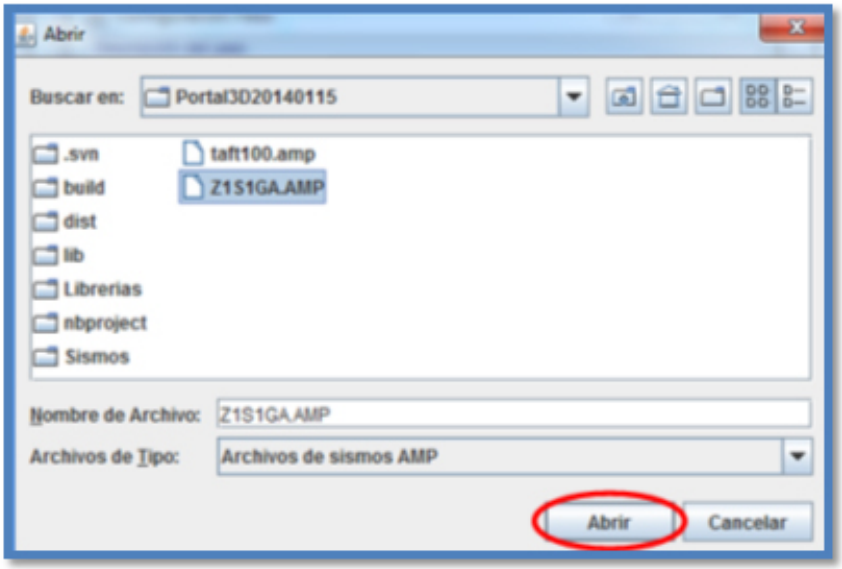

Figura D.24: Archivo de sismo para el ejemplo

Al finalizar el análisis correctamente esta indicara que el análisis ha sido finalizado correctamente (Ver figura D.25).

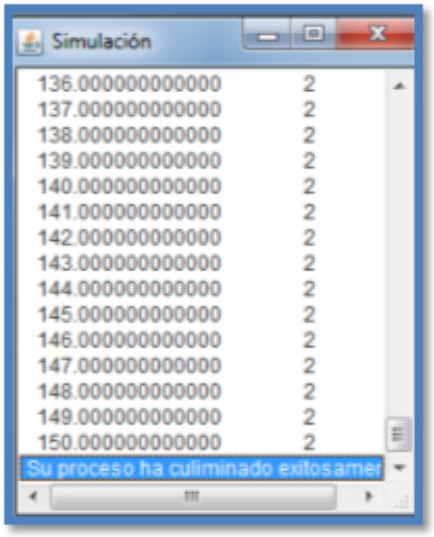

Figura D.25: Proceso exitoso del análisis

Una vez analizada correctamente nuestra columna se activara la pestaña de "Resultados" donde podremos observar los resultados obtenidos. Para observar las graficas obtenidas, a través del botón "Gráficas", al oprimirlo nos aparece una ventana llamada "Graficador" donde pide indicar las variables, en nuestro ejemplo graficaremos Tiempo vs el alargamiento axial. Seleccionamos la opción de "Elementos" y luego elegimos el Elemento 2, continuamente escogemos las dos variables, y finalmente hacemos click en "Aceptar" (Ver figura D.26).

Para observar el daño ocasionado por las solicitaciones, hacemos click en "M. Daño" lo cual desplegara una nueva ventana llamada "Generador Mapa de Daño". Para observar la magnitud de los daños hacemos click en "CORRER" (Ver figuras D.27 y D.28).

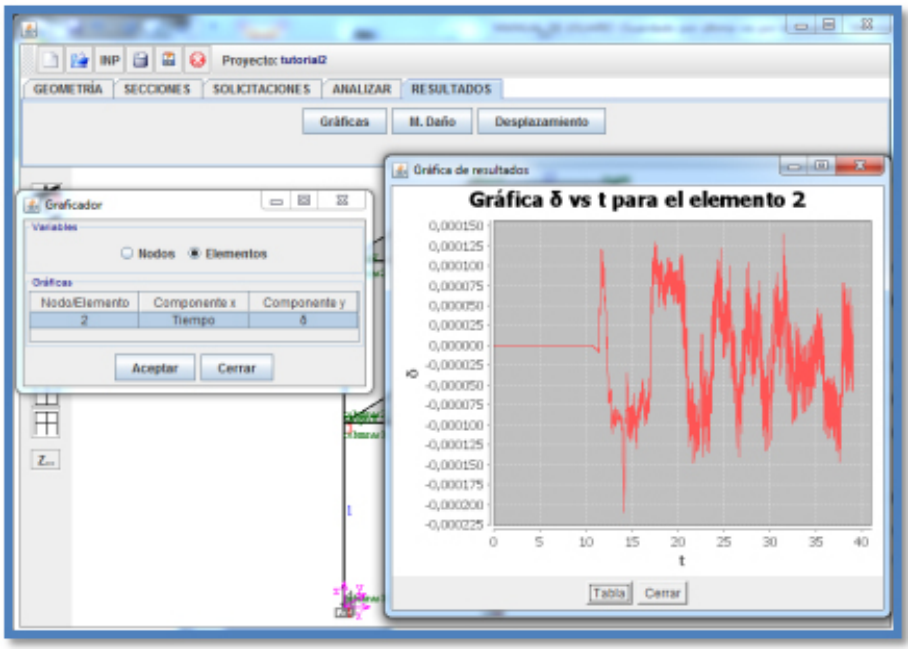

Figura D.26: Grafica Tiempo vs Alargamiento axial para el ejemplo

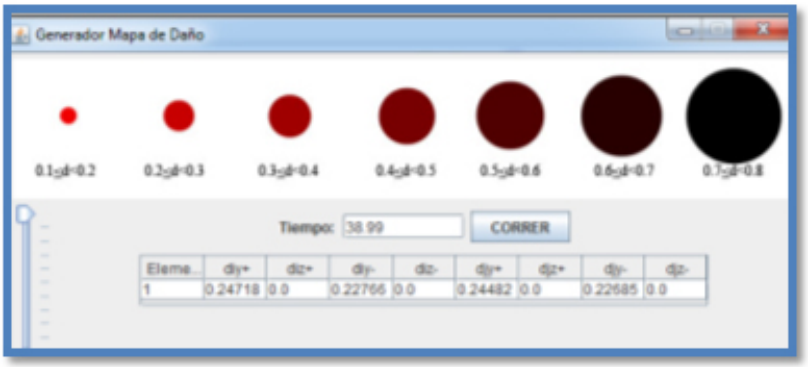

Figura D.27: Generador de daños para el ejemplo

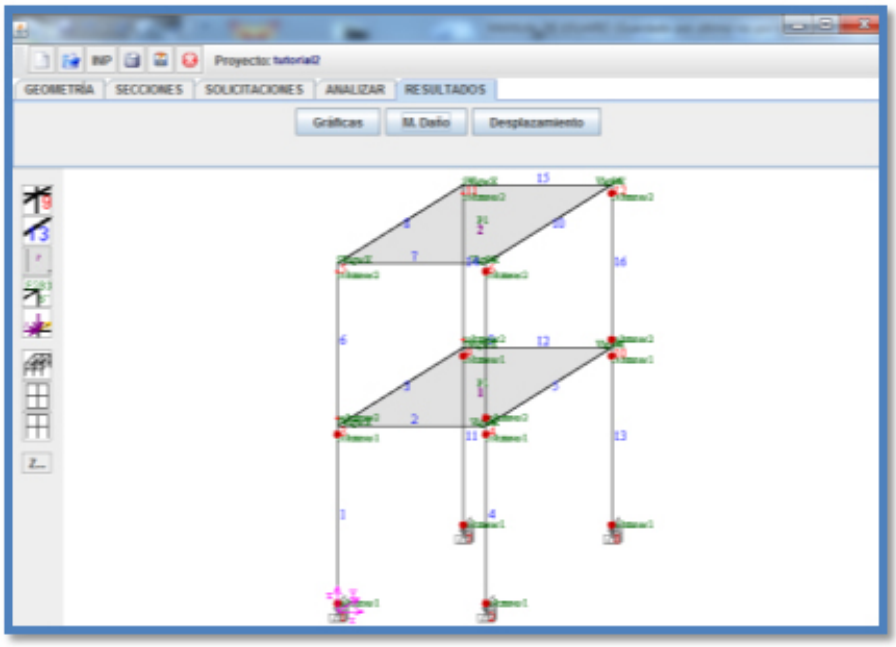

Figura D.28: Vista 3D de los daños ocurridos para el ejemplo