RESUMEN

Hoy en día, para poder conocer las propiedades mecánicas de los elementos estructurales en edificaciones existentes, se tiene que recurrir a la toma de muestras de los materiales para luego ser ensayados en laboratorio y así obtener los valores reales de sus parámetros mecánicos, lo cual toma su tiempo y tiene su costo económico, adicionalmente, otra alternativa es realizar pruebas de cargas en situ para obtener las curvas de esfuerzo - deformación propias de cada elemento estructural existente en dicha edificación.

Estos antecedentes ponen de manifiesto el indudable interés que genera la posibilidad de disponer de una metodología alternativa para determinar las propiedades mecánicas de los elementos estructurales sin tener que recurrir a los métodos anteriormente descritos.

Por eso ésta monografía presenta una metodología de revisión para identificar los valores de los parámetros mecánicos como son el Módulo de Elasticidad (E), el Módulo de Rigidez al Cortante (G), el Coeficiente de Poisson (μ) y la Resistencia Específica del Hormigón (f´c) de los elementos estructurales sometidos a flexión, como son las vigas, de estructuras existentes de hormigón armado, mediante las técnicas de optimización y el cálculo estructural basado en elementos finitos.

INDICE

UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERIA
CENTRO DE POSTGRADOS

"TÉCNICAS DE OPTIMIZACIÓN PARA LA IDENTIFICACIÓN DE PARÁMETROS MECÁNICOS DE ESTRUCTURAS"

Monografía previa a la obtención del Título de Especialista en Análisis y Diseño de Estructuras.

Dirección :

Ing. Jaime Bojorque I. ,PhD.

Autor :

Ing. Fredy Leonardo Bacuilima G.

Cuenca, Julio de 2011

CAPITULO UNO : INTRODUCCIÓN

1. INTRODUCCIÓN

En los últimos años gracias al desarrollo de técnicas computacionales interactivas o automáticas se ha dado un gran impulso a una rama de la ingeniería conocida como "optimización estructural", mediante la cual se han mejorado los diseños, obteniendo una reducción de costos, materiales y tiempo en los procesos de diseño realizados por los ingenieros, estos diseños efectuados con técnicas de optimización deben cumplir con todas las condiciones de diseño, en cuanto a limitaciones y restricciones impuestas. Además la utilización de computadoras hace que la tarea de búsqueda del proyecto óptimo se torne bastante atractiva, de forma que los ingenieros puedan pasar más tiempo dedicado a la parte de concepción del proyecto lo que requiere mucha creatividad sin dejar a lado los cálculos matemáticos.

En general, se puede decir que las técnicas de optimización buscan un conjunto de valores de las variables de diseño, que haga mínima una función objetivo y cumpla a la vez una serie de restricciones que dependen de las mismas variables.

Ya formalmente se puede definir el problema de la optimización como:

$\min f(x)$

Sujeta a:

$g(x) = 0$

$h(x) \leq 0$

En donde:

x Variables de diseño.

f() Función objetivo.

g() Restricción de igualdad.

h() Restricción de desigualdad.

Las funciones f(), g() y h() pueden ser funciones lineales y/ó no lineales.

En la gran mayoría de los artículos que se han publicado con respecto al uso de las técnicas de optimización en las edificaciones de hormigón armado, tratan o estudian

principalmente a las vigas. Pocos trabajos representan la optimización de columnas o de pórticos espaciales, como se menciona en el artículo de Borda J. y Rodríguez G. (2010).

En la siguiente tabla que resume el estado de arte de los últimos 20 años, que a la vez, es una síntesis del artículo de Borda J. y Rodríguez G. (2010) del recuento histórico realizado hasta año 2010 de los trabajos de investigaciones desarrollados, en donde, se mencionan algunos autores de artículos científicos y sus contribuciones en cuanto a la optimización estructural.

AUTOR	AÑO	MÉTODO DE OPTIMIZACIÓN	APLICACIÓN EN LA OPTIMIZACIÓN ESTRUCTURAL
Al-Salloum y Siddiqi	1994	Multiplicadores de Lagrange.	Diseño de costo óptimo de vigas rectangulares de hormigón simplemente armadas.
Coello	1995	Algoritmos genéticos (GA).	Optimización de vigas rectangulares de hormigón armado (H°A).
Zielinski	1995	Penalización interna.	Diseño óptimo de columnas cortas de H°A.
Torrano y Martí	1997	Programación cuadrática (QP) mediante el software Disseny.	Optimización de secciones de forma arbitraria de H°A.
Pirzada	2000	Multiplicadores de Lagrange.	Minimización de los costos de vigas de H°A sujetas a una sola restricción de resistencia.
Martí	2001	Varias técnicas de optimización	Diseño óptimo de secciones y laminas de H°A.
Camp	2003	Algoritmos genéticos (GA).	Procedimiento para la optimización discreta de pórticos de H°A.
Elachachi y Djellouli	2004	Programación cuadrática secuencial (SQP) con ayuda del "Optimization Toolbox" de MATLAB	Optimización paramétrica de estructuras de H°A.
Andreczevski	2005	Optimización mediante el diseño experimental.	Minimización de costos de secciones transversales de vigas de H°A.
Rodrigues	2005	Optimización Multi-nivel.	Diseño óptimo de columnas de H°A en edificios altos.
Liang	2006	Optimización basada en desempeño de modelos Puntal-tensor.	Procedimiento para la optimización topológica de uniones viga-columna en pórticos de H°A.
Saini	2006	Redes neuronales artificiales (ANN).	Diseño óptimo de vigas de hormigón simple y doblemente armadas sometidas a cargas estáticas.

Quiroz	2007	Redes neuronales artificiales (ANN).	Diseño óptimo de pórticos de H°A sismorresistentes.
Fragiadakis y Papadrakakis	2008	Metodología automática basada en análisis por desempeño.	Diseño sísmico óptimo de pórticos de H°A.
Paya	2008	Cristalización simulada (SA).	Minimización del costo de estructuras aporticadas de H°A.
Tomás y Martí	2009	Módulo de optimización del software ANSYS.	Optimizaron la cantidad de refuerzo de acero en elementos finitos de placas y cáscaras de hormigón.
Borda y Rodríguez	2010	Algoritmos SQP y de Punto Interior del "Optimization Toolbox" de MATLAB	Diseño de costo óptimo de vigas de hormigón simplemente armadas.

1.1. MOTIVACIÓN

Hoy en día, para poder conocer las propiedades mecánicas de los elementos estructurales en edificaciones existentes, se tiene que recurrir a la toma de muestras de los materiales para luego ser ensayados en laboratorio y así obtener los valores reales de sus parámetros mecánicos, lo cual toma su tiempo y tiene su costo económico, adicionalmente, otra alternativa es realizar pruebas de cargas en situ para obtener las curvas de esfuerzo - deformación propias de cada elemento estructural existente en dicha edificación.

Estos antecedentes ponen de manifiesto el indudable interés que genera la posibilidad de disponer de una metodología alternativa para determinar las propiedades mecánicas de los elementos estructurales sin tener que recurrir a los métodos anteriormente descritos. Esta metodología debe ser versátil, debe poder implementarse fácilmente y debe dar la solución a los diferentes problemas que se presentan, independientemente del tamaño y geometría de la edificación, además de poderse validar numéricamente los resultados obtenidos, de ahí que, mediante las técnicas de optimización, las cuales se podrían utilizar como herramientas para la revisión de las estructuras existentes, puesto que al basarse en métodos inversos, se pueden ir variando los criterios de diseño impuestos originalmente (parámetros mecánicos, dimensiones de los elementos, propiedades de los materiales, etc.) para obtener las condiciones de frontera observadas o los resultados medidos en la realidad (deformaciones en los elementos, condiciones constructivas), para verificar si se han respetado y cumplido con los valores establecidos de los parámetros mecánicos de los elementos estructurales analizados, en sus diferentes etapas de diseño y construcción.

1.2. OBJETIVO GENERAL

El objetivo de ésta monografía es presentar una metodología de revisión para identificar los valores de los parámetros mecánicos como son el Módulo de Elasticidad (E), el Módulo de Rigidez al Cortante (G), el Coeficiente de Poisson (μ) y la Resistencia Específica del Hormigón (f´c) de los elementos estructurales sometidos a flexión, como son las vigas, de estructuras existentes de hormigón armado, mediante las técnicas de optimización y el cálculo estructural basado en elementos finitos.

1.3. OBJETIVOS ESPECÍFICOS

El principal objetivo es desarrollar una metodología de tipo general, aplicable directamente a un amplio espectro de problemas estructurales que se presentan independientemente de geometría y tamaño de la edificación, además se ha escogido utilizar paquetes de programas que son muy empleados y difundidos en nuestro medio como son el MATLAB y SAP2000.

Seleccionar en el entorno MATLAB, el método más adecuado basado en técnicas de optimización, de los ya desarrollados en centros de investigación que se va aplicar para realizar el análisis inverso.

Validar la metodología numéricamente, al garantizar la fiabilidad de los resultados obtenidos con la utilización del programa de cálculo estructural SAP2000 basado en el Método de Elementos Finitos (MEF).

Desarrollar un algoritmo de interfaz que ligue las herramientas de optimización existentes en MATLAB con las del cálculo estructural provenientes del SAP2000, para obtener una metodología coherente con el resto de los planteamientos y en la medida de lo posible, eficiente y sencilla.

1.4. PLANTEAMIENTO GENERAL DEL PROBLEMA

En términos matemáticos se formula el problema del diseño óptimo como la selección de un conjunto de variables de diseño, de forma que se minimice una función objetivo y se

verifiquen unas restricciones que se expresan en general mediante ecuaciones e inecuaciones.

El problema así descrito se conoce con el nombre de "problema general de minimización condicionada", y al conjunto de técnicas que permiten obtener su solución, así como la de otros problemas análogos, se las denomina con el nombre genérico de "programación matemática". La programación matemática es ya un área del conocimiento que envuelve una gran complejidad y que ha sido ya desarrollada por varios centros de investigación los cuales han sacado paquetes de programas que dan solución a esta problemática. Navarrina F. (1987)

Para esta monografía en particular, se parte de que existe una edificación de hormigón armado, de la cual se requiere conocer los parámetros mecánicos (E, G, μ, f´c) reales, no los de diseño o los teóricos, de una o varias vigas en particular de dicha edificación, para lo cual, se conocen a detalle la geometría de la edificación, además se conocen o se pueden obtener mediante mediciones las flechas reales en uno o varios puntos del elemento o elementos estructurales que se estén analizando.

En la ***Figura 1*** se esquematiza el proceso a realizar. Las variables de diseño se considerarán a los valores de los parámetros mecánicos (E, G, μ, f´c) de los elementos estructurales sometidas a flexión, es decir las vigas, de la estructura de hormigón armado que se esté considerando para la revisión. Estas variables de diseño tienen a su vez restricciones como valores máximos y mínimos que son requisitos que deben cumplir estos elementos para que se consideren como estructurales.

Se planteará en el programa de cálculo estructural SAP2000 basado en Elementos Finitos un modelo de un sistema estructural que represente lo más apegado a la realidad a la estructura continua de la edificación que se esté analizando o realizando la revisión.

Una vez planteado el modelo estructural en el SAP2000 se pueden obtener para cada valor de las variables de diseño (E, G, μ, f´c), valores de las flechas calculadas en diferentes puntos de la viga o vigas que se estén considerando para el análisis.

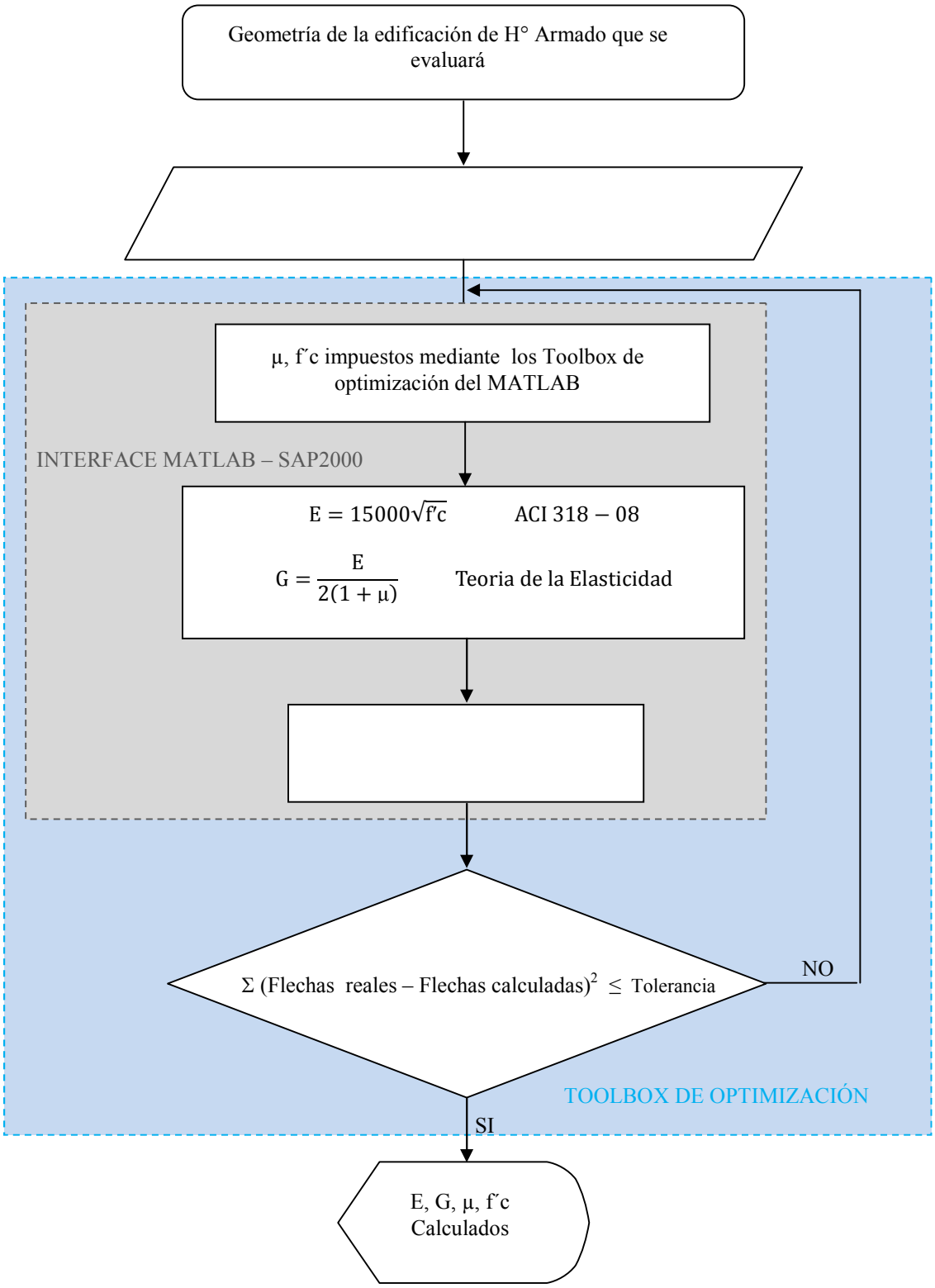


Figura 1: Proceso para encontrar los valores de parámetros mecánicos en vigas de estructuras existentes.

La función objetivo a minimizar será de la diferencia entre los valores de las flechas reales medidas y las flechas del modelo planteado en el programa SAP2000, puesto que, se pueden ir variando los valores de las variables de diseño (E, G, μ, f´c) a través de Toolbox de optimización en MATLAB, hasta que las flechas calculadas en el modelo sean casi o iguales la medidas en la realidad.

Se finalizará el proceso cuando la función objetivo sea cero o muy cercana a cero, de acuerdo a la tolerancia que se imponga, cuando haya concluido este proceso los valores que impusieron a las variables de diseño (E, G, μ, f´c), con los que se obtuvieron las flechas del modelo con las que se minimizo la función objetivo serán las que se asumirán como los valores reales de las propiedades mecánicas en los elementos estructurales analizados.

Hay que aclarar que para realizar este proceso se lo hará todo en el programa de cálculo numérico MATLAB y se tendrá que programar además una interface que lea y cambie los valores de los parámetros mecánicos del modelo estructural que se calculará en el SAP2000 para que todo el proceso sea automático.

1.5. ACTIVIDADES DESARROLLADAS

Para poder conseguir el objetivo planteado, ha sido necesario realizar las actividades que se detallan a continuación:

- Revisión de los principales comandos o funciones de los toolbox de optimización con los que cuenta el MATLAB para resolver problemas lineales y no-lineales. La versión que se ha utilizado del MATLAB es la 7.7(R2008b).
- Elección y detallamiento de la función de optimización más apropiada para ser aplicado a este proceso.
- Implementación de un modelo estructural de la edificación en el programa de cálculo estructural SAP2000 basado en el MEF. La versión empleada del SAP2000 es la 12.
- Realización del algoritmo de interface entre MATLAB y SAP2000 desarrollado en el entorno de programación MATLAB.

- Determinación de las propiedades mecánicas de los elementos estructurales analizados, a través del algoritmo de programación en MATLAB.
- Conclusiones y recomendaciones

1.6. CONTENIDO Y ORGANIZACIÓN DE LA MONOGRAFÍA

La presente monografía está dividida en cinco capítulos, el primero de los cuales corresponde a la presente introducción.

En el segundo capítulo se da una breve descripción de toolbox de optimización con los que cuenta el MATLAB para resolver los diferentes problemas de "programación matemática", además se explica a mayor detalle la función de optimización a ser utilizada de la caja de herramientas del MATLAB.

En el tercer capítulo se describen detalladamente las características que debe cumplir el modelo estructural del SAP2000 para que posteriormente pueda ser reconocido y ser enlazado con el MATLAB mediante la interface.

En el cuarto capítulo se realiza el análisis del caso de estudio, en donde se explicarán los parámetros de entrada y salida de la función de optimización utilizada y además los resultados de las propiedades mecánicas obtenidas del modelo estructural analizado.

En el quinto y último capítulo se presentan las principales conclusiones obtenidas como resultado del presente trabajo de Monografía, así como algunas posibles líneas de trabajos que se podrían desarrollar a futuro.

A continuación se citan las referencias bibliográficas consultadas durante la realización de este trabajo.

Finalmente, se presenta los apéndices, en el que se muestra el programa desarrollado en MATLAB para la aplicación de la metodología propuesta y las diferentes funciones creadas y utilizadas.

CAPITULO DOS : MÉTODOS DE OPTIMIZACIÓN: AMBIENTE MATLAB

2. MÉTOS DE OPTIMIZACIÓN: AMBIENTE MATLAB

El nombre de MATLAB proviene de la contracción de los términos MATrix LABoratory, es un programa interactivo para computación numérica y visualización de datos. Es ampliamente usado por Ingenieros de Control en el análisis y diseño, posee además una extraordinaria versatilidad y capacidad para resolver problemas en matemática aplicada, física, química, ingeniería, finanzas y muchas otras aplicaciones. Está basado en un sofisticado software de matrices para el análisis de sistemas de ecuaciones. Permite resolver complicados problemas numéricos sin necesidad de escribir un programa.

MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos.

Adicionalmente, MATLAB integra análisis numérico, cálculo matricial, proceso de señales y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional.

El programa permite desarrollar de un modo rápido la resolución numérica de problemas en un tiempo mucho menor que si se quisiesen resolver estos mismos problemas con lenguajes de programación tradicionales como pueden ser los lenguajes Fortran, Basic o C debido a las herramientas y funciones que posee.

Goza en la actualidad de un alto nivel de implantación en escuelas y centros universitarios, así como en departamentos de investigación y desarrollo de muchas compañías industriales nacionales e internacionales. En entornos universitarios, por ejemplo, MATLAB se ha convertido en una herramienta básica, tanto para los profesionales e investigadores de centros docentes, como una importante herramienta para la impartición de cursos universitarios, tales como sistemas e ingeniería de control, álgebra lineal, proceso digital de imágenes, señales, etc. En el mundo industrial, MATLAB está siendo utilizado como herramienta de investigación para la resolución de complejos problemas planteados en la realización y aplicación de modelos matemáticos en ingeniería. Los usos más característicos de la herramienta los encontramos en áreas de computación y

cálculo numérico tradicional, prototipaje algorítmico, teoría de control automático, estadística, análisis de series temporales para el proceso digital de señales.

Así también, MATLAB dispone también en la actualidad de un amplio abanico de programas de apoyo especializados, denominados Toolbox (Caja de herramientas), que extienden significativamente el número de funciones incorporadas en el programa principal. Estos Toolbox cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación.

2.1. PRINCIPALES TOOLBOX DEL MATLAB

Entre los toolbox más importantes se encuentran:

- Curve fitting: Ajustes de modelos y análisis.
- Data Acquisition: Adquiere y envía datos a un instrumento electrónico conectado al computador. (sólo para Windows)
- Excel link: Permite usar Matlab con datos leídos directamente desde planillas Excel.
- Image processing: Permite el procesamiento de imágenes, análisis y desarrollo de algoritmos.
- Partial differential equation: Soluciona y analiza sistema de ecuaciones diferenciales parciales.
- Signal Processing: Permite el procesamiento de señales, análisis y desarrollo de algoritmos.
- Spline: Crea y manipula modelos de aproximación de datos Spline.
- Statistics: Permite aplicar modelos estadísticos y modelos de probabilidades.
- Structural Dynamics: Analiza modelos de elementos finitos y lleva a cabo análisis modales de sistemas mecánicos.
- Wavelet: Analiza, comprime y saca el ruido de señales e imágenes usando técnicas de wavelet.
- ***Optimization***: Consta de un conjunto de funciones que resuelven problemas de extremos, con o sin condiciones, de funciones reales las cuales son generalmente multivariables y no lineales.

2.2. TOOLBOX DE OPTIMIZACIÓN

Como ya se mencionó anteriormente la caja de herramientas de optimización consta de un conjunto de funciones o comandos que determinan el mínimo de una función lineal o no lineal que puede ser o no multivariable con restricciones de igualdad y desigualdad. Además, posee funciones para la resolución de algunos tipos de problemas matriciales en extremos.

Entre las principales funciones de la caja de herramientas de optimización tenemos:

- fminbnd: Esta función resuelve los problemas de optimización de funciones de una variable sin restricciones, se la conoce también como optimización escalar.
- fminsearch: Esa función en cambio resuelve los problemas de optimización de funciones de más de una variable sin restricciones.
- fminunc: Ésta al igual que la anterior proporciona el mínimo de una función de variables sin restricciones, pero, con la diferencia que utiliza información del gradiente y el hessiano de la función objetivo.
- fmincon: Ésta función determina el mínimo de una función multivariable con restricciones de igualdad y desigualdad, lineales y no lineales.
- quadprog: Realiza la minimización de una función cuadrática con restricciones de igualdad y desigualdad lineales.
- linprog: Esta función realiza la optimización de problemas de programación lineal.
- ***lsqnonlin***: Resuelve por mínimos cuadrados problemas no lineales de funciones o de ajuste de datos.

Se ha analizado cada una de estas funciones y su aplicación, por lo que, se puede decir que la función de optimización que más encaja para resolver el problema que se plantea en esta monografía es la lsqnonlin ya que la misma realiza la optimización de problemas no lineales por mínimos cuadrados mediante el ajuste de datos o valores que se van a obtener del programa SAP2000 mediante la interface que se realizará en MATLAB.

2.3. DESCRIPCIÓN DE LA FUNCIÓN lsqnonlin

lsqnonlin resuelve por mínimos cuadrados problemas no lineales, incluye el problema de ajuste de datos no lineales.

2.3.1. Ecuación

$$min_x \|f(x)\|^2 = min_x(f_1(x)^2 + f_2(x)^2 + \cdots + f_n(x)^2)$$

2.3.2. Sintaxis

```
x = lsqnonlin(fun,x0)
x = lsqnonlin(fun,x0,lb,ub)
x = lsqnonlin(fun,x0,lb,ub,options)
[x,resnorm] = lsqnonlin(...)
[x,resnorm,residual] = lsqnonlin(...)
[x,resnorm,residual,exitflag] = lsqnonlin(...)
[x,resnorm,residual,exitflag,output] = lsqnonlin(...)
[x,resnorm,residual,exitflag,output,lambda] = lsqnonlin(...)
[x,resnorm,residual,exitflag,output,lambda,jacobian] = lsqnonlin(...)
```

2.3.3. Descripción

En lugar de calcular el valor$\|f(x)\|^2$ (La suma de los cuadrados), `lsqnonlin` requiere la función definida por el usuario para calcular el vector de función con valores de:

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \dots \\ f_n(x) \end{bmatrix}$$

Es decir, en términos de vectores, se puede repetir este problema de optimización como:

$$min_x \|f(x)\|^2 = min_x(f_1(x)^2 + f_2(x)^2 + \cdots + f_n(x)^2)$$

En donde `x` es un vector y `f(x)` es una función que devuelve un valor del vector.

`x = lsqnonlin (fun, x0)` comienza en el punto `x0` y encuentra un mínimo de la suma de los cuadrados de las funciones descritas en `fun`. Se devuelve en `fun` un vector de valores y no la suma de los cuadrados de los valores. (El algoritmo implícitamente sumas y eleva al cuadrado `fun(x)`).

`x = lsqnonlin (fun, x0, lb, ub)` define un conjunto de límites inferior y superior en las variables de diseño en `x`, por lo que la solución siempre está en el rango de `lb ≤ x ≤ ub`. Cuando se pasen matrices vacías en `lb` y `ub` no existen los límites.

`x = lsqnonlin (fun, x0, lb, ub, opciones)` reduce al mínimo con las opciones de optimización especificadas en la estructura de `opciones`. Se utiliza `optimset` para establecer estas opciones. Para ésta monografía se va utilizar la optimización mediante el Jacobiano definido por el usuario en `fun` . La función `fun` debe devolver, en el segundo argumento de salida, el valor del Jacobiano en una matriz `J`.

```
J = ... % Jacobiano de la función evaluada en el último valor de x
```

Si `fun` devuelve un vector (matriz) de `m` componentes y `x` tiene una longitud `n`, donde `n` es la longitud de `x0`, el Jacobiano `J` es una matriz de `m` -por- `n` en donde `J (i,j)` es la derivada parcial de `F(i)` con respecto a `x(j)`. (El Jacobiano `J` es la transpuesta de la gradiente de `F`), es decir, la metodología que se va emplear para determinar los valores óptimos de la función que se requiere minimizar es aproximar los valores de su gradiente a cero.

`[x, resnorm] = lsqnonlin (...)` devuelve el valor de la potencia 2 del residuo en `x` : `suma (fun (x).^2)`.

`[x, resnorm, residual] = lsqnonlin(...)` devuelve el valor residual de `fun(x)` en la solución `x`.

`[x, resnorm, residual, exitflag] = lsqnonlin (...)` devuelve un valor `exitflag` que describe la condición de salida.

`[x, resnorm, residual, exitflag, output] = lsqnonlin (...)` devuelve una estructura `output` que contiene información acerca de la optimización.

`[x, resnorm, residual, exitflag, output, lambda] = lsqnonlin (...)` devuelve una estructura `lambda` cuyos campos contienen los multiplicadores de Lagrange en la solución `x`.

`[x, resnorm, residual, exitflag, output, lambda, jacobian] = lsqnonlin (...)` devuelve el Jacobiano de `fun` en la solución `x` .

Nota : Si los límites de entrada especificado por un problema son incompatibles, la salida de `x` es `x0` y las salidas `resnorm` y `residuales` son `[]`.Los componentes de `x0` que violan los límites `lb ≤ x ≤ ub` se restablecen en el interior del rango definido por los límites. Los componentes con respeto a los límites no se modifican.

CAPITULO TRES : MODELACIÓN NUMÉRICA Y OPTIMIZACIÓN DEL MODELO ESTRUCTURAL

3. MODELIZACIÓN NUMÉRICA Y OPTIMIZACIÓN DEL MODELO ESTRUCTURAL

Todas las estructuras deben ser diseñadas y construidas para que, con una seguridad aceptable, sea capaz de soportar las acciones que la puedan solicitar durante la construcción y el período de vida útil previsto en el proyecto así como la agresividad del medio en el que se encuentre.

El análisis estructural consiste en la determinación de los efectos originados por las acciones sobre la totalidad o parte de la estructura, con el objeto de efectuar comprobaciones en sus elementos estructurales resistentes, que es lo que se plantea en ésta presente monografía.

Para poder realizar la revisión de la estructura existente, se idealizan tanto la geometría de la estructura como las acciones y las condiciones de apoyo mediante un modelo matemático o numérico.

Para la realización de éste modelo numérico se va emplear el programa SAP2000, en donde, se representará atreves de un modelo estructural que debe ser capaz siempre de reproducir el comportamiento estructural dominante de la edificación de hormigón armado de la que se desea realizar la verificación e identificación de los valores de los parámetros mecánicos (E, G, μ, f´c) de las vigas a ser estudiadas.

SAP2000 es un programa sofisticado y de fácil manejo desarrollado por Computers and Structures Inc. (CSI).

En SAP2000 es posible crear y modificar un modelo, ejecutar el análisis del mismo, así como revisar y diseñar cada elemento. Los resultados se presentan de una manera gráfica en tiempo real. Posee una rápida solución de ecuaciones, esfuerzos y desplazamientos inducidos por cargas, elemento frame de sección no prismática, elemento shell muy exactos, análisis dinámicos, múltiples sistemas de coordenadas, varios tipos de constraint (restricciones) en nodos que ofrece la facilidad de fusionar mallas de elementos independientes.

Sap2000 posee además un módulo completo de diseño para acero y concreto reforzado incluido en el mismo programa usado para crear y analizar el modelo.

El método de análisis de SAP2000 se basa en la teoría de elementos finitos, la cual básicamente es dividir el elemento en partes pequeñas las cuales poseen las siguientes características:

Geometría:	Sistema de referencia.
Material:	Ley constitutiva.
Condiciones de frontera esenciales:	Apoyos.
Condiciones de frontera naturales:	Cargas.

3.1. PASOS QUE DEBE CUMPLIR EL MODELO ESTRUCTURAL EN EL SAP2000

Los siguientes pasos que se detalla a continuación deben cumplir obligatoriamente para que el modelo estructural del SAP2000 pueda posteriormente ser reconocido y ser enlazado con la interface del MATLAB que se creará:

- Unidades: En el SAP2000 siempre hay que definir primero las unidades en las que se van a trabajar antes de comenzar a crear el modelo de la estructura, claro que posteriormente en el proceso de creación del mismo se pueden cambiar las mismas, lo que hay que destacar es que si se requiere que la interface creada en MATLAB reconozca el modelo, las unidades en que se guarde el modelo deben ser siempre kilogramo fuerza (kgf), metro (m), ° Centígrados (C).
- Dibujo de la geometría del modelo estructural: Con las herramientas que cuenta el SAP2000 se dibuja el modelo de un sistema estructural que represente lo más apegado a la realidad a la estructura continua, considerando o tomando en cuenta la geometría y tamaño de la edificación que se esté analizando, aquí también se definen las secciones de los diferentes elementos estructurales que componen el pórtico que representa el modelo.
- Definición de los apoyos: El SAP2000 tiene múltiples condiciones de apoyo que se pueden definir, lo que hay que recalcar es que se debe escoger el más representativo a la realidad.

- Cargas: Las cargas que se deben definir en el modelo son las que va a estar sometida la edificación en el momento que se va a realizar la revisión, para que así el modelo represente lo que se va analizar. Aquí hay que destacar que no solamente involucra a las cargas sino también a las diferentes combinaciones de las mismas, las cuales posteriormente en la interface de MATLAB que se creará hay que definir ya que los valores que se van a tomar como resultado del análisis en el SAP2000 van a ser las flechas para el estado o combinación de carga que uno quiera analizar.
- Material: Se tienen que definir las propiedades mecánicas de todos los elementos estructurales que componen el pórtico del modelo estructural, pero como lo que se está planteando es encontrar justamente estos valores, se puede colocar en el modelo un valor cualquiera, ya posteriormente la interface del MATLAB lo cambiara automáticamente, se recomienda colocar los valores de las propiedades mecánicas que corresponden a un hormigón de un f´c=210 kgf/cm^2 ya que en la interface creada en el MATLAB toma como punto de partida esos valores. También lo que hay que recalcar es que en el modelo creado en el SAP2000 se tiene que tomar en cuenta el número del material de los elementos que se van analizar, este número se obtiene de la posición del material en la ventana de `Define Material` que se puede obtener en el menú del SAP2000.
- Definir el nombre de PORTICO para la base de datos que posteriormente leerá la interface de MATLAB: Esto se logra en la ventana de `Choose Tables for Display` que se obtiene del menú del SAP2000 en `Dislay\Show Tables…` En esta ventana hay que señalar las 54 tablas que conformarán la base de datos, además en esta misma ventana hay que definir el nombre de la base de datos en la parte que dice `Save Named Set` aquí se colocará PORTICO como nombre de la base de datos, ya que así se definió en la interface del MATLAB.
- Guardar la base de datos en el archivo PORTICO.xls: En la ventana de `Analysis Options` que se obtiene del menú principal del SAP2000 en `Analyze\Set Analysis Options…` en ésta ventana nos salen tres campos que hay que llenar de la siguiente manera:

`File Name`	Pórtico.xls

`Database Tables Named Set`	PORTICO
`Group`	All.

Esto nos asegura que cada vez que la interface que se realizada en MATLAB llame al SAP2000 guarde los resultados del análisis corrido por el programa de cálculo estructural en un archivo de Excel llamado pórtico.xls que luego será leído por la misma interface creada.

- Por último hay que guardar el archivo del modelo realizado en el SAP2000 en el directorio raiz `C:\` con cualquier nombre que uno desee. Esto se hace con la finalidad de que la interface creada en MATLAB sepa el lugar en donde buscar el modelo cuando ella lo requiera.

3.2. OPTIMIZACIÓN DEL MODELO ESTRUCTURAL

El método que se empleará para realizar la optimización del modelo estructural será mediante mínimos cuadrados, ajustando los valores o las variables de diseño con la utilización de la matriz Jacobiana.

La matriz Jacobiana `J`, que recibe éste nombre en honor al matemático Carl Gustav Jacobi, es una matriz formada por las derivadas parciales de primer orden de una función que para este caso será la función objetivo que se va a minimizar.

$$F(x) = \text{Función objetivo} = \text{Flechas reales} - \text{Flechas calculadas por el SAP2000}$$

$$x = \text{Variables de diseño}$$

$$y_i = F_i(x_i, \dots, x_n) \qquad y = F(x) = (F_1(x), \dots, F_m(x))$$

$$J = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

La matriz Jacobiana `J` es la transpuesta de la gradiente de `F(x)` y al determinante de la misma se le conoce como Jacobiano.

El Jacobiano nos da información muy importante sobre el comportamiento de `F(x)` en el punto `x` que se está analizando. Para empezar, una función `F(x)` es invertible cerca de `x` si el Jacobiano en `x` es no nulo. Más aún, el valor absoluto del determinante en `x` nos da el factor con el cual `F(x)` se expande o contrae su volumen cerca de `x`. Con ésta información el método va buscando los valores en donde la gradiente de la función cambia su signo lo cual no indica que el punto optimo esta próximo al mismo.

La interface que se creará en MATLAB, va a ser la encargada de tomar los valores de vector `x` dados por `lsqnonlin`, colocar dichos valores en el modelo estructural del SAP2000, calcular los valores de las flechas del modelo y de obtener la diferencia entre la flecha real medida y la del modelo calculado.

Cuando la diferencia entre las flechas reales y las calculadas sea menor que la tolerancia adoptada, la función `lsqnonlin` va a terminar su proceso y va a devolver los valores buscados de parámetros mecánicos (E, G, μ, f´c).

Para poder realizar el programa o algoritmo de optimización en MATLAB del modelo estructural creado en SAP2000 se utilizaron variables, comandos y sentencias, las cuales se explican a detalle en Anexo 1. Adicionalmente en el Anexo 2 se presenta el programa creado para el cálculo de las propiedades mecánicas en vigas. En el Anexo 3 se muestra la función creada en MATLAB para realizar la interface entre MATLAB y SAP2000.

3.3. PAQUETE ADICIONAL DE MIMICA DE MOVIMIENTO

Adicionalmente a las variables, comandos y sentencia, se utilizó un paquete de programa, que lleva el nombre de MIMICA DE MOVIMIENTO. Este paquete consta de tres funciones creadas (*.m) y por cuatro archivos de aplicación (*.dll). Las funciones creadas son las siguientes:

- WAITFORWINDOW.M: Esta función lo que realiza es esperar que se abra y luego activar la ventana que se especifica en esta función, la sintaxis que se utiliza es la siguiente:

```
waitforwindow(Nombre de la ventana)
```

En el anexo 4 se presenta esta función del paquete de MIMICA DE MOVIMIENTO.

- SIMINPUT.M: Esta función simula el movimiento del ratón al clicklear con el botón derecho o izquierdo según se especifique, también simula la entrada de alguna orden atraves del teclado, las sintaxis que se utilizarán son las siguientes:

`siminput('mouseMLC',[x y])`	Realiza un click con el botón izquierdo del ratón en la ventana activa, en las coordenadas de la pantalla `x y` , la coordenadas de la pantalla son [0 0] en la esquina superior izquierda de la pantalla y [1 1] en la esquina inferior derecha.
`siminput('mouseMRC',[x y])`	Realiza un click con el botón derecho del ratón en la ventana activa, en las coordenadas de la pantalla x y.
`siminput('keyb','{Teclas}')`	Simula la entrada de alguna orden en la ventana activa, atraves del teclado especificando el nombre de las `teclas` que conforman la orden.

En el anexo 5 se muestra esta función del paquete de MIMICA DE MOVIMIENTO.

La función `siminput.m` cuando simula alguna orden del teclado depende de la función `str2kc.m` que es la tercera función que contiene el paquete MIMICA DE MOVIMIENTO y que se detalla a continuación.

- STR2KC.M: Esta función arma la estructura que conforma la orden dada por el teclado para que la función `suminput.m` pueda trabajar, la sintaxis que se utilizará es la siguiente:

`str2kc({Teclas})` En donde `Teclas` son los nombres de las teclas que conforman la orden, por ejemplo, si se quiere dar la orden, que se un enter con el teclado la orden sería `ENTER`.

En el anexo 6 se presenta esta función del paquete de MIMICA DE MOVIMIENTO.

A más de las funciones, este paquete de MIMICA DE MOVIMIENTO contiene cuatro aplicaciones (*.dll) para que puedan trabajar las funciones anteriormente descritas, estas aplicaciones ya son definidas y no pueden ser cambiadas ya que vienen en lenguaje de la máquina y no se conoce su programación interna. Estos archivos de aplicación son:

- Getwindowname.dll
- Mousecontrol.dll
- Keycontrol.dll
- Maximizewindow.dll

NOTA: Para que el programa desarrollado en ésta monografía pueda trabajar, se debe cambiar temporalmente la configuración de Windows, de tal manera que todo archivo con extensión *.xls no se lo abra con Excel si no con el SAP2000, esto se logra configurando en el Explorador de Windows, haciendo un click con el botón derecho del ratón sobre cualquier archivo de Excel *.xls y se busca la opción `Abrir con` y se toma la que diga Abrir con SAP2000. Cuando ya no se desee utilizar el programa desarrollado se puede regresar a la configuración original de Windows.

CAPITULO CUATRO : ANÁLISIS DEL CASO DE ESTUDIO

4. ANÁLISIS DEL CASO DE ESTUDIO

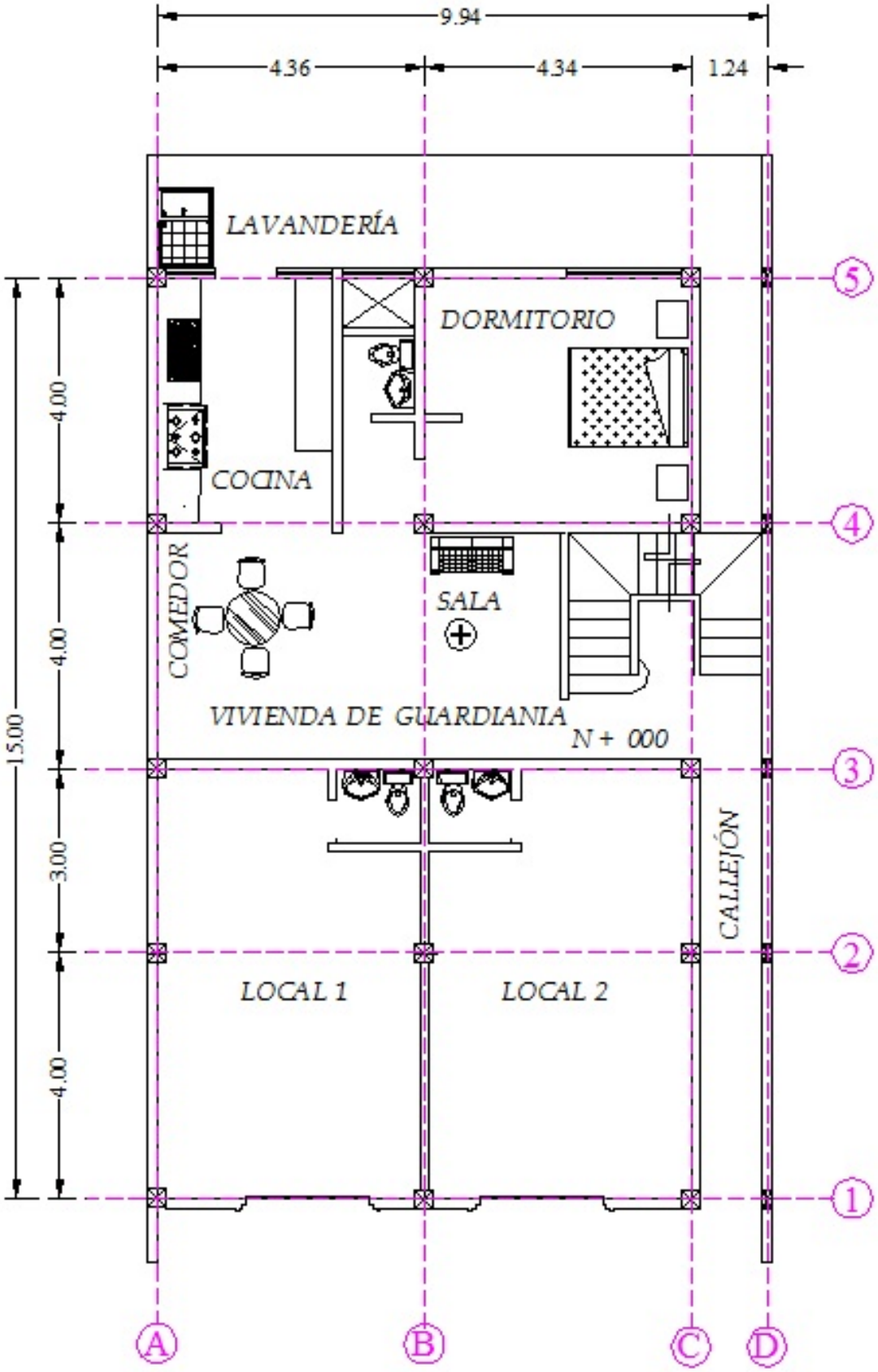
En la ***Figura 1*** del Capítulo 1 de ésta monografía se muestra el proceso que se va a seguir para encontrar los valores de parámetros mecánicos en vigas de la edificación que se está analizando, para ello se parte de la geometría y forma de la edificación. Para ésta monografía se ha tomado la geometría de un edificio de hormigón armado de cuatro plantas, de las cuales, en su planta baja está funcionado dos locales comerciales y la vivienda de guardianía, en las dos plantas superiores en cambio su uso es de departamentos residenciales y en su última planta que es de cubierta sirve a la vez de azotea del edificio. Ésta edificación tiene destinado un sitio específico para parqueo vehicular en la parte exterior baja del mismo.

Su ubicación por razones de reserva pedidas por su propietario no se da a conocer pero se puede decir que pertenece a la parroquia de Bellavista de la ciudad de Cuenca provincia del Azuay.

Cabe recalcar que la construcción de ésta edificación concluyo en Noviembre de 2008 y hasta la fecha, se lo ha dado uso al mismo. A finales del 2010 se empezaron a notar deflexiones no previstas en las vigas, especialmente de los pisos superiores de dicha edificación, para lo cual con consentimiento del propietario que desea conocer las causas que están provocando estas deflexiones, ha facilitado los planos arquitectónicos y estructurales del mismo. El propietario no desea que se saquen muestras o núcleos de los elementos estructurales con lo cual se podría comparar los resultados de los mismos con la nueva metodología que se está planteando en ésta monografía para la identificación de los parámetros mecánicos elementos estructurales.

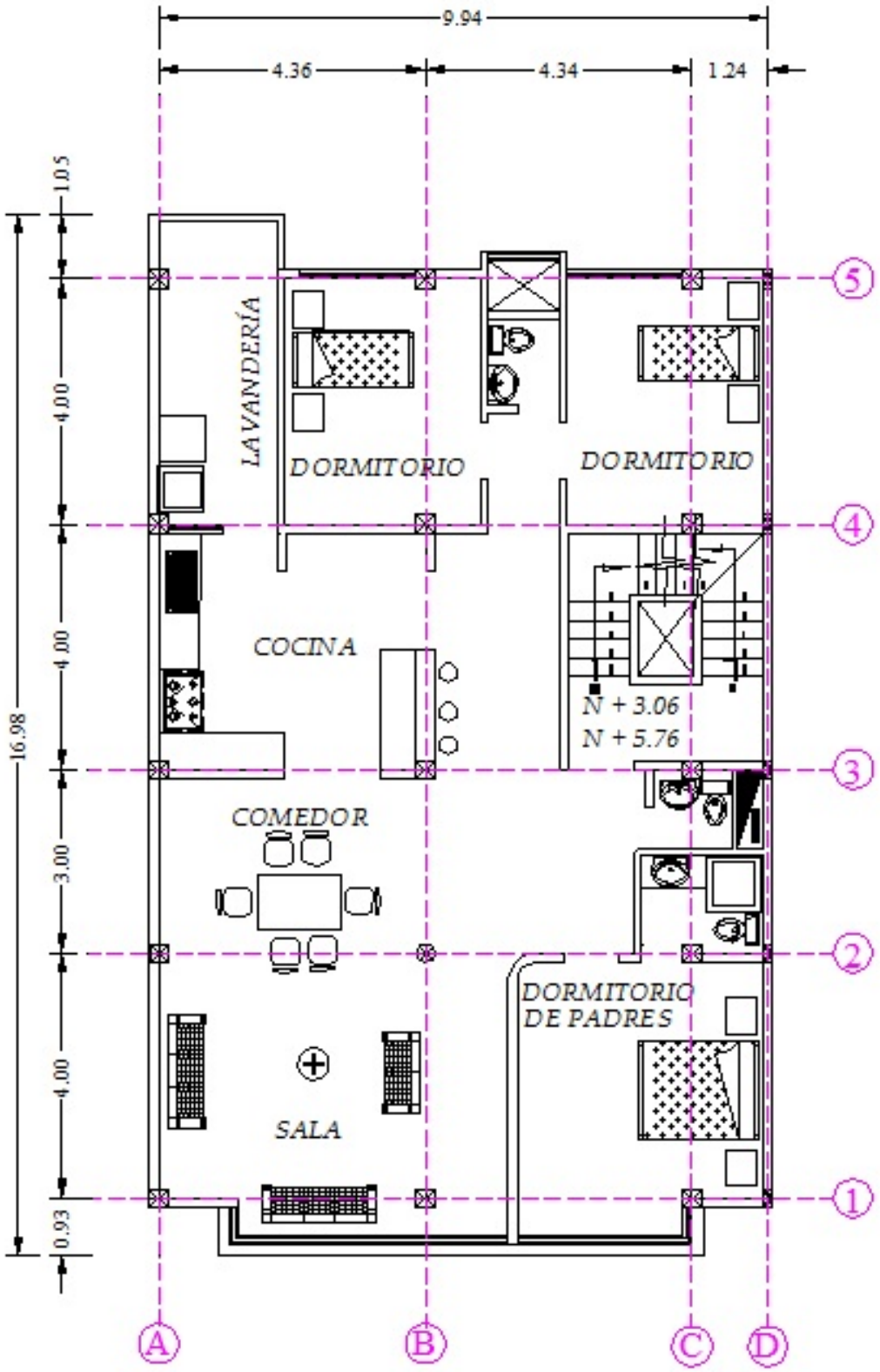
Para el análisis estructural se van a utilizar los criterios establecidos en el Código Ecuatoriano de la Construcción (CEC) y el Reglamento para Concreto Estructural ACI 318-08, además como ya se mencionó anteriormente el modelo estructural se va emplear el programa SAP2000 y para lo que es la optimización se va aprovechar la herramientas de optimización de `lsqnonlin` que posee el MATLAB.

A continuación en la ***Figura 2*** se indica las plantas y vistas arquitectónicas de la edificación que se va a tomar como caso de estudio.



PLANTA BAJA NIVEL +000

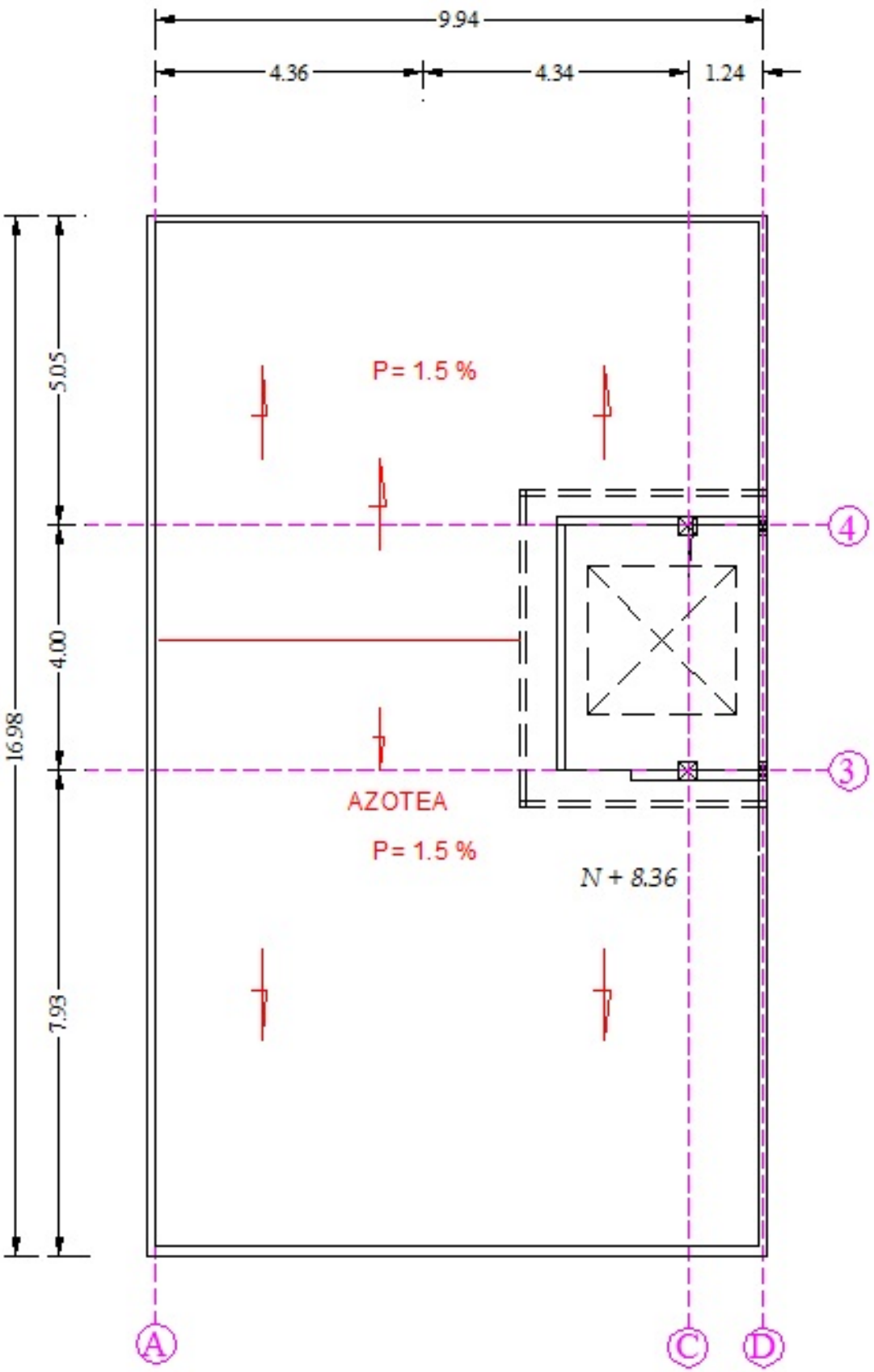
Figura 2a: Proyecto arquitectónico



PRIMERA PLANTA ALTA NIVEL +3.060

Y SEGUNDA PLANTA ALTA NIVEL +5.760

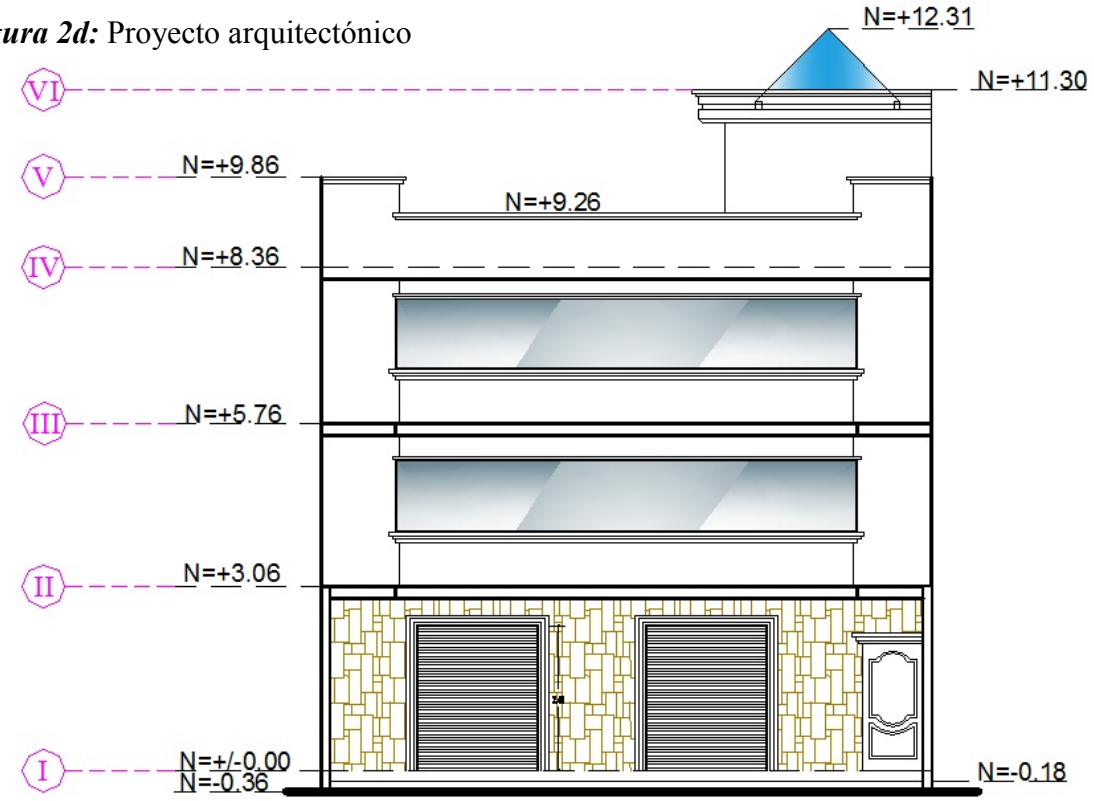
Figura 2b: Proyecto arquitectónico



PLANTA DE CUBIERTA NIVEL +8.360

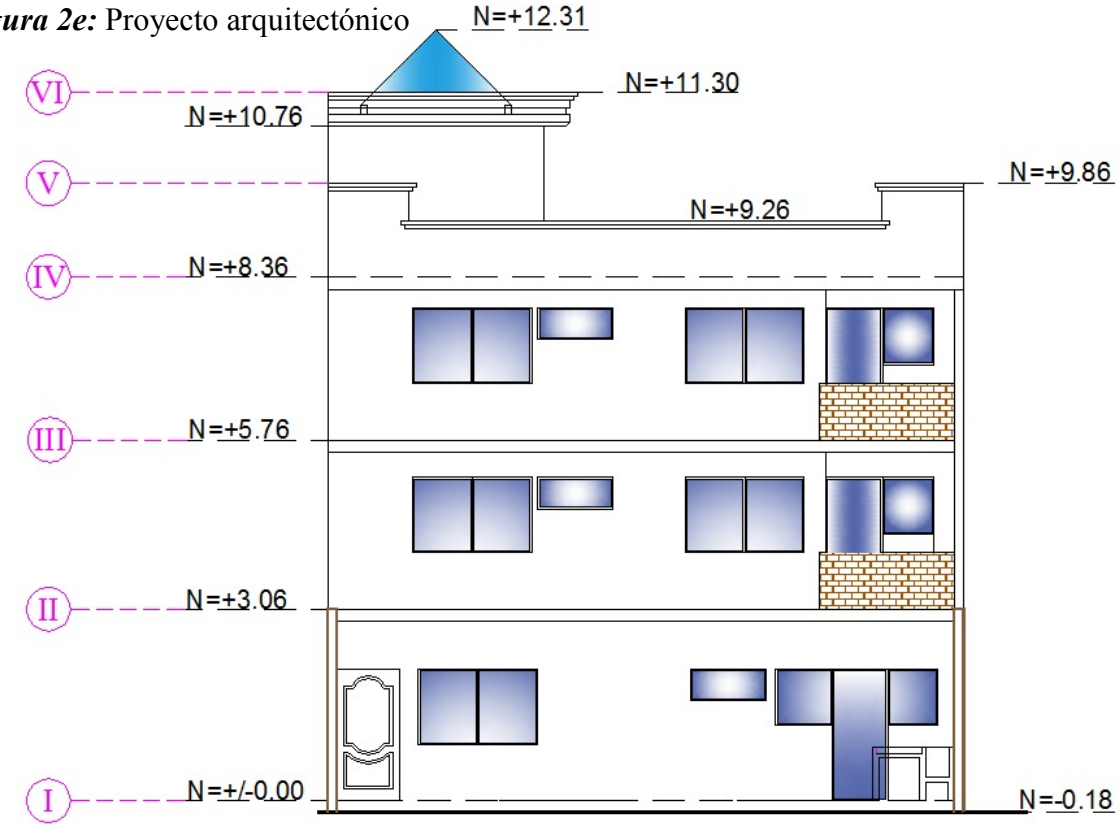
Figura 2c: Proyecto arquitectónico

Figura 2d: Proyecto arquitectónico



FACHADA FRONTAL

Figura 2e: Proyecto arquitectónico



FACHADA POSTERIOR

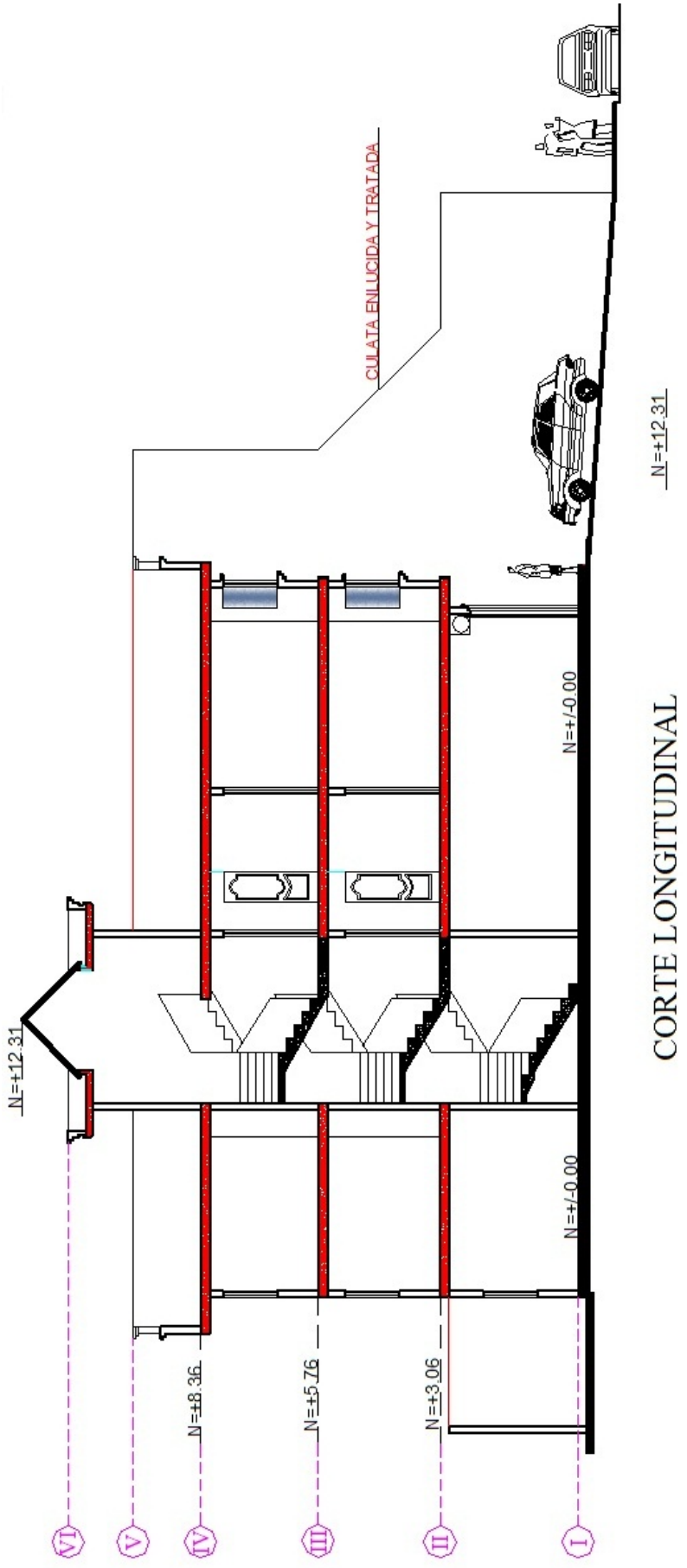


Figura 2f: Proyecto arquitectónico

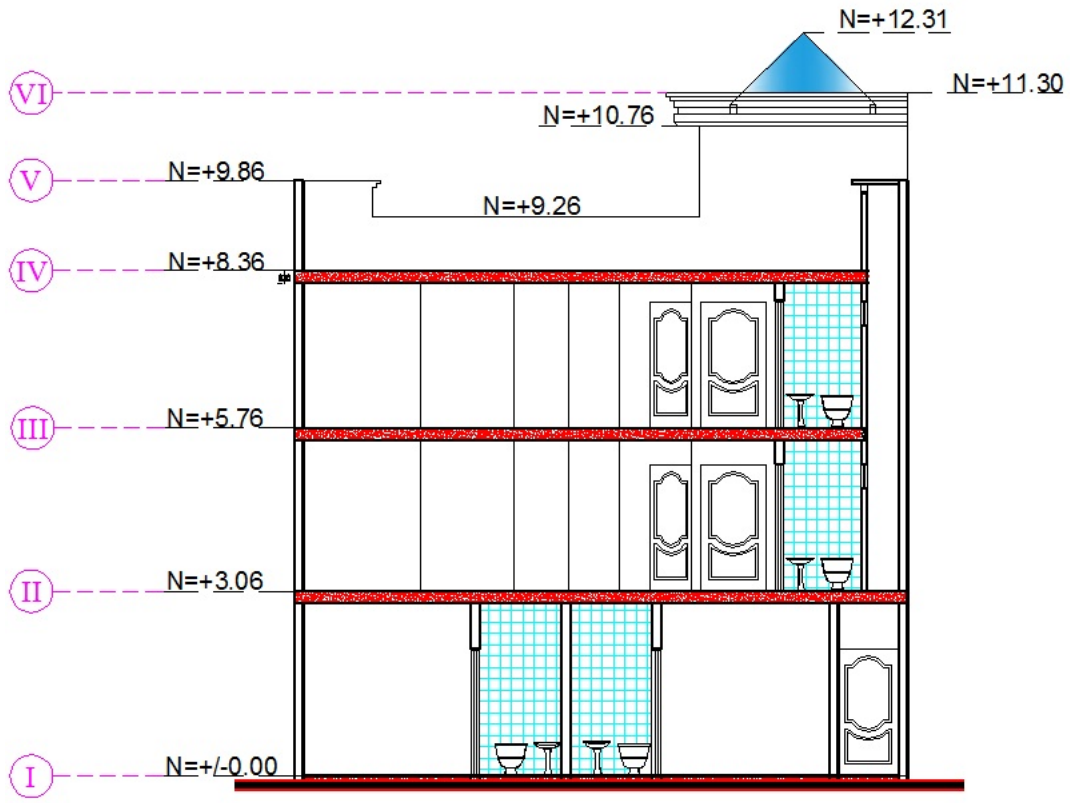


Figura 2g: Proyecto arquitectónico

4.1. ANÁLISIS DE CARGAS

Para el análisis de cargas muertas se han tomado las dimensiones de los elementos estructurales ya construidos y los siguientes pesos volumétricos de los materiales de acuerdo a los existentes en el mercado:

- Hormigón armado 2400 kg/m^3
- Mortero de cemento y arena 1800 kg/m^3
- Ladrillo macizo 1800 kg/m^3
- Bloque hueco de concreto 1200 kg/m^3
- Bloque hueco de pomes 880 kg/m^3
- Azulejo o loseta 1800 kg/m^3
- Vidrio para ventanas 2600 kg/m^3

En cambio para las cargas vivas que son producidas por el uso y ocupación de la edificación se han considerado, las siguientes de acuerdo al CEC:

- Residencias 200 kg/m^3
- Almacenes minoristas 400 kg/m^3
- Para pasillos, escaleras, rampas 500 kg/m^3
- Azoteas 100 kg/m^3

4.1.1. Carga muerta de la losa

La carga muerta de la losa se calcula por metro cuadrado, en donde se toma en cuenta el peso de los materiales que se han utilizado para su construcción como se muestra en la ***Figura 3***.

Peso de loseta	=	2400*0.05*1*1	=	120.0 kg/m^2
Peso de los nervios	=	2400*(1*0.2+0.8*0.2)*0.15	=	129.6 kg/m^2
Peso de bloques de pomes	=	880*0.15*0.8*0.8	=	84.5 kg/m^2
Peso de alisado de piso	=	1800*0.03*1*1	=	54. 0 kg/m^2
Peso de azulejo	=	1800*0.02 *1*1	=	36. 0 kg/m^2

TOTAL= 424.1 kg/m^2

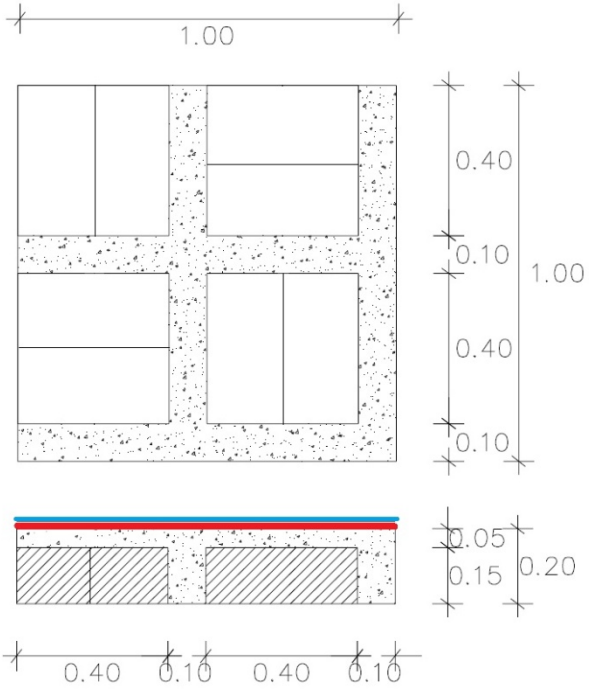


Figura 3: Corte transversal de la losa

El dato que se ingresará en el modelo del SAP2000 para representar esta losa nervada será como un elemento Shell de espesor de 0.10 m y de peso volumétrico de 2400 kg/m^3 que es aproximadamente el peso de la losa si se consideraría maciza y para lo que es la inercia, hay que corregir la misma por un factor de 3.05 para que la losa maciza tenga la misma inercia que la losa nervada propuesta. En el anexo 7 se detalla cómo se obtiene este factor de corrección del momento de inercia.

4.1.2. Carga muerta de las Paredes

La carga muerta de las paredes se calculará por metro lineal de pared y se la aplicará sobre la viga que soportará a la misma. Para la edificación que se está analizando tenemos los siguientes casos:

Paredes de la primera panta alta:

- Solo pared:

Alto de pared	=	2.50 m
Espesor pared de bloque	=	0.15 m
Espesor del enlucido dos lados	=	0.03 m
Peso del bloque =1200*0.15*2.5 =		450.0 kg/m
Peso del enlucido =1800*0.03*2.5=		135.0 kg/m
TOTAL=		585.0 kg/m

- Pared con ventana:

Alto de pared incluido ventana	=	2.50 m
Alto de la ventana	=	1.20 m
Espesor pared de bloque	=	0.15 m
Espesor del enlucido dos lados	=	0.03 m
Peso del bloque =1200*0.15*1.3 =		234.0 kg/m
Peso del enlucido =1800*0.03*1.3=		70.20 kg/m
Peso del vidrio =2600*0.006*1.2=		18.7 kg/m
TOTAL=		322.9 kg/m

Paredes de la segunda panta alta:

- Solo pared:

Alto de pared	=	2.40 m
Espesor pared de bloque	=	0.15 m
Espesor del enlucido dos lados	=	0.03 m

Peso del bloque =1200*0.15*2.4 = 432.0 kg/m

Peso del enlucido =1800*0.03*2.4= 129.6 kg/m

TOTAL= 561.6 kg/m

- Pared con ventana: Alto de pared incluido ventana = 2.40 m

  Alto de la ventana = 1.20 m

  Espesor pared de bloque = 0.15 m

  Espesor del enlucido dos lados = 0.03 m

  Peso del bloque =1200*0.15*1.2 = 216.0 kg/m

  Peso del enlucido =1800*0.03*1.2= 64.80 kg/m

  Peso del vidrio =2600*0.006*1.2= 18.7 kg/m

  TOTAL= 299.5 kg/m

Paredes de la Azotea:

- Pared alta: Alto de pared = 1.50 m

  Espesor pared de bloque = 0.15 m

  Espesor del enlucido dos lados = 0.03 m

  Peso del bloque =1200*0.15*1.5 = 270.0 kg/m

  Peso del enlucido =1800*0.03*1.5= 81.00 kg/m

  TOTAL= 251.0 kg/m

- Pared baja: Alto de pared = 0.90 m

  Espesor pared de bloque = 0.15 m

  Espesor del enlucido dos lados = 0.03 m

  Peso del bloque =1200*0.15*0.9 = 162.0 kg/m

  Peso del enlucido =1800*0.03*0.9= 48.60 kg/m

  TOTAL= 210.6 kg/m

4.1.3. Combinaciones de carga

No se va a considerar el efecto de sismo ya que como se menciono anteriormente el modelo debe reflejar a la edificación en el momento que se realiza la revisión de la misma, de ahí, el sismo no se encuentra presente. Lo que si hay que tomar en cuenta son las combinaciones de las cargas muerta y viva:

- Combinación 1=Muera + Viva sin mayorar C1=D+L
- Combinación 2=Muera + Viva mayoradas C2=1.4D+1.7L ACI 318-08

4.2. SECCIONES DE ELEMENTOS ESTRUCTURALES

Las secciones que se van a indicar a continuación en la ***Figura 4*** fueron obtenidas del plano estructural proporcionado por el propietario de la edificación. Solamente las dimensiones de las secciones fueron verificadas en el edificio ya construido puesto que no se permitió ninguna rotura en el mismo para poder chequear el acero que fue colocado.

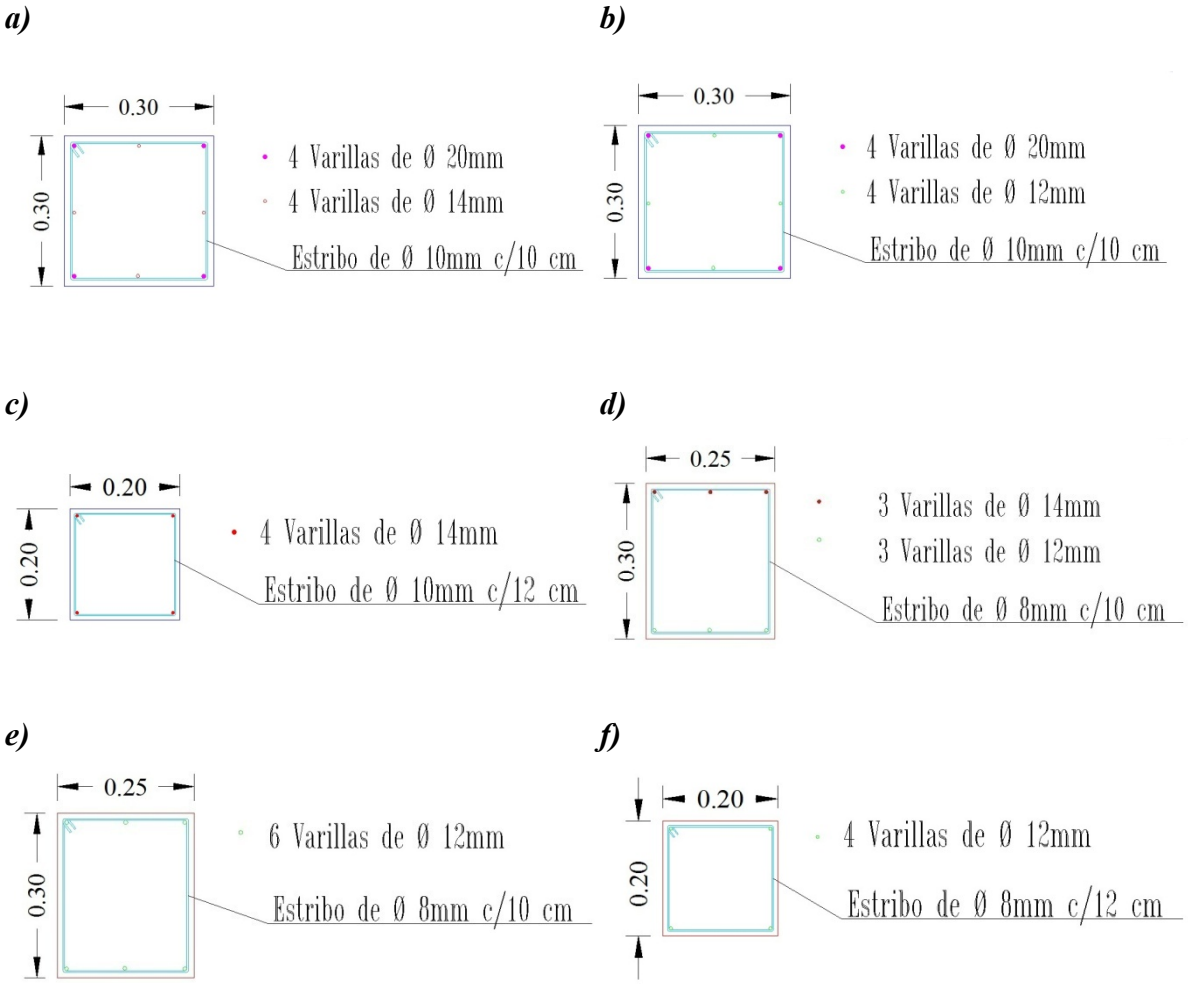


Figura 4: Secciones de columnas y vigas

En la ***Figura 4a*** se indica la sección de todas las columnas que se encuentran en el eje B desde el nivel I al IV, en cambio en la ***Figura 4b*** pertenecen las columnas que se encuentran en los ejes A y C desde el nivel I al IV. La ***Figura 4c*** agrupa a todas las columnas del eje D desde el nivel I al VI además a las columnas del eje C desde el nivel IV al VI.

Pertenecen a la ***Figura 4d*** todas las vigas que se encuentran en los ejes B, 2, 3 y 4 desde el nivel I al IV. En la ***Figura 4e*** engloba a todas las vigas de los ejes A, C, 1 y 5 desde el nivel I al VI. Por último en a ***Figura 4f*** pertenecen las vigas del eje D a más de todas las vigas que se encuentran entre los niveles IV y VI.

4.3. MODELO MATEMATICO INTRODUCIDO EN EL SAP2000

En el programa de cálculo estructural basado en MEF SAP2000 se introdujeron los datos de la geometría de la edificación basándose a los planos arquitectónicos mostrados en la ***Figura 2***, en cambio para las secciones de los elementos estructurales se introdujeron de acuerdo a la ***Figura 4*** y para los que son las cargas se colocaron de a cuerdo a lo descrito en éste capítulo en la sección **4.1**.

A continuación en la ***Figura 5*** se presenta el modelo matemático ya introducido en el programa SAP2000.

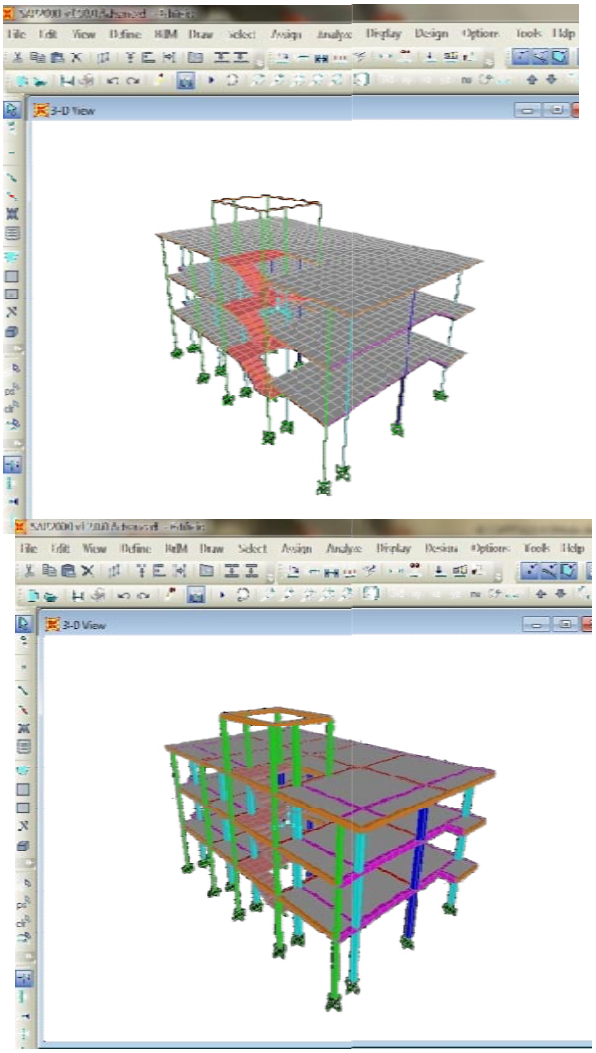


Figura 5: Modelo estructural

4.4. DATOS DE LA FLECHAS MEDIDAS

A más del modelo matemático ya introducido en el SAP2000 se necesitan los datos de las flechas reales medidas en los elementos estructurales que se van analizar para poder obtener los valores de parámetros mecánicos de dichos elementos de la edificación que se está realizando la revisión.

Los elementos estructurales que se van analizar son los que pertenecen a la viga A-B, en el eje 4, en el nivel 5.76 de la ***Figura 2b***.

Ésta viga se ha dividido en 9 elementos como se puede observar en la Figura 6, estos elementos se encuentran entre los nodos 1315 al 1324 respectivamente del modelo estructural introducido en el SAP2000.

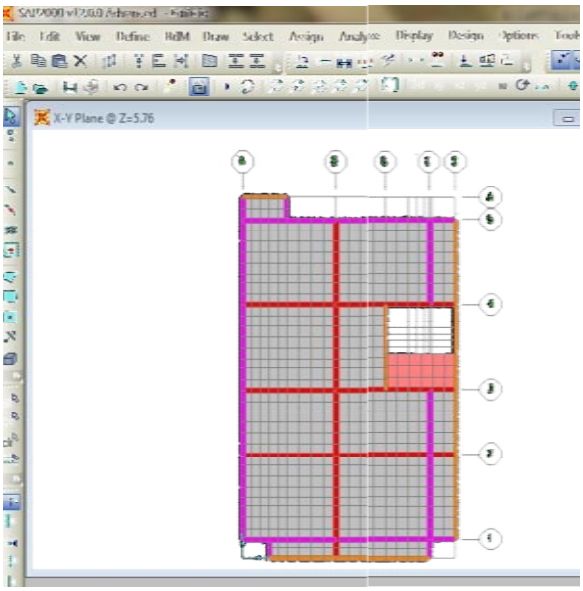


Figura 6: Elementos estructurales que se van analizar

Los datos de las flechas reales de los elementos estructurales que se van a analizar se impondrán ya que no se cuenta con la información adecuada, las mismas que se introducirán mediante un archivo de texto *.txt, que tiene que contener la siguiente información. En su primera línea el nombre del archivo del SAP2000 que contiene el modelo estructural sin ninguna extensión, en la segunda línea se debe dar el número de la combinación de carga y el número del material que se va analizar. En el tercer renglón los tiempos en segundos que se necesitan para abrir el archivo del SAP2000 y el tiempo que se

demora en correr el archivo en el SAP2000. Desde la cuarta línea en adelante y para cada línea dependiendo de los elementos que se estudiarán se debe introducirse el número del nodo y la flecha medida en ese nodo en mm de las esquinas de los elementos seleccionados.

Así para nuestro caso de estudio el archivo de texto con la información se lo ha llamado datos.txt y contiene la siguiente información:

Edificio
3 7
210 180
1316 1.5
1317 2.5
1318 3.0
1319 3.8
1320 4.0
1321 3.8
1322 3.0
1323 2.0

Hay que aclarar que las flecha medidas son en los nodos y no se han considerado los nodos 1315 y 1324 puesto que esos nodos son en las columnas y allí no se pueden realizar ninguna medición.

4.5. APLICACIÓN DE LA FUNCIÓN lsqnonlin PARA EL CASO DE ESTUDIO

Se aplicará la siguiente forma de la función `lsqnonlin` para realizar la optimización y poder encontrar los valores de los parámetros mecánicos (E, G, μ, f´c) de las vigas de la estructura o de la edificación de hormigón armado estudiada.

```
[x,resnorm,residual,exitflag,output,lambda,jacobian]=lsqnonlin(fun,x0,lb,ub,options)
```

Los parámetros mecánicos (f´c, μ) para este caso en particular corresponderían al vector `x` de las variables, las cuales la función `lsqnonlin` va encontrar para que `fun` sea mínima aplicando el método de mínimos cuadrados. Los valores de E se determinará ya una vez calculado f'c con la siguiente expresión:

$$E = 15000\sqrt{f'c} \qquad \text{según ACI } 318 - 08$$

En donde:

$$E \text{ al igual que } f'c \text{ esta en } kg/cm^2$$

G también se calculará posteriormente al igual que E mediante la siguiente expresión:

$$G = \frac{E}{2(1+\mu)} \qquad \text{según la Teoria de la Elasticidad}$$

En donde:

G en las mismas unidad de E

`fun` va a ser la función que sirva de interface entre MATLAB y SAP2000, ya que ella va a ser la encargada de tomar los valores del vector `x` dados por `lsqnonlin`, colocar dichos valores en el modelo estructural del SAP2000, calcular los valores de las flechas del modelo y de obtener la diferencia entre la flecha real medida y la del modelo calculado.

Cuando la diferencia entre las flechas reales y las calculadas sea menor que la tolerancia adoptada, la función `lsqnonlin` va a terminar su proceso y va a devolver los valores buscados de parámetros mecánicos (E, G, μ, f´c).

El vector `x0` va a contener los valores de los parámetros mecánicos (f´c, μ) con los que la función `lsqnonlin` va a tomar como punto de partida para comenzar a buscar los valores que hagan mínima a `fun`. Estos valores de `x0` se van a adoptar a los correspondientes a un hormigón de f´c=210 kg/cm^2 ya que según las normas del ACI 318-08 es la menor resistencia para que un hormigón se considere como estructural, es decir `x0=[fc0,μ0]=[210,0.2]`. Hay que aclarar que este vector `x0` pude iniciar con valores adoptados al azar, lo que si va a influir es en el tiempo y el número de interacciones para que la función `lsqnonlin` encuentre los valores que busca para que minimice a `fun`.

Los vectores `lb ub` corresponden al conjunto de límites inferior y superior de las variables de diseño `x`, que para el presente estudio serian los límites de los parámetros mecánicos que se están buscando. Para el vector `lb` se van a adoptar a los correspondientes a un hormigón de f´c=100 kg/cm^2, este valor se toma bastante bajo para que la respuesta

tenga un rango mayor de posibilidad de encontrar su resultado esperado, en cambio para el vector `ub` se van a adoptar a los correspondientes a un hormigón de f´c=900 kg/cm^2, el coeficiente μ se considerará que varia de 0.15 a 0.25, con lo que quedaría establecido como: `lb=[fclb,μlb]=[100,0.15] ub=[fcub,μub] =[900,0.25].`

En `options` se especifica la estructura con la que `lsqnonlin` realizará optimización, se utiliza `optimset` para establecer o cambia los valores de `options`, a continuación se va mostrar la configuración utilizada:

```
options=optimset('Jacobian','on','Hessian',off','LargeScale','on',…
                                                …'TolX',TolXv,'TolFun',TolFunv)
```

En donde:

`Jacobian` Si se usa '`on`' , `lsqnonlin` utiliza el `Jacobian` definido por el usuario en `fun` para la optimización. La función `fun` debe devolver, en el segundo argumento de salida, el valor del Jacobiano en una matriz `J`.

```
J = ... % Jacobiano de la función evaluada en el último
valor de x
```

Si `fun` devuelve un vector (matriz) de `m` componentes y `x` tiene una longitud `n`, donde `n` es la longitud de `x0`, el Jacobiano `J` es una matriz de `m` -por- `n` en donde `J (i,j)` es la derivada parcial de `F(i)` con respecto a `x(j)`. (El Jacobiano `J` es la transpuesta de la gradiente de `F`).

`Hessian` Si se coloca '`off`' `lsqnonlin` no utiliza el `Hesiano` para la optimización ya que está utilizando otra opción, en este caso ya utiliza el Jacobiano.

`LargeScale` Si se coloca '`on`' para resolver problemas en estructuras de grandes escalas.

`TolX` Es el valor de la tolerancia en `x`, debe ser un escalar positivo. El valor que se coloca se guarda en `TolXv=0.01`.

`TolFun` Es el valor de la tolerancia de la función, debe ser un escalar positivo. El valor que se coloca se guarda en `TolFunv =0.01`.

`MaxIter` El número máximo de iteraciones permitido, debe ser un entero positivo. El valor que se colocó para el caso de estudio es `800`.

`MaxFunEvals` Es el número máximo de evaluaciones que la función permite, debe ser un número entero positivo. Se colocó `800`.

`DiffMaxChange` Máximo cambio en las variables de las gradientes de diferencias finitas (un escalar positivo). El valor adoptado fue `0.01`

`DiffMinChange` Mínimo cambio en las variables de las gradientes de diferencias finitas (debe ser un escalar positivo). El valor que se colocó es `0.001`

En cuanto a los demás argumentos de salida como son `resnorm residual exitflag output lambda jacobian` que nos da la función `lsqnonlin`, se van a utilizar únicamente `residual` y `exitflag` que darán los valores del residuo de la función calculada y si la función a convergido o no respectivamente.

4.6. RESULTADOS OBTENIDOS PARA EL CASO DE ESTUDIO

Si no se aplicara la optimización para la obtención de los valores de los parámetros mecánicos se tendría que ir variando los valores de los mismos e ir calculando atraves del SAP2000 las flechas en los nodos que se analizan para así comparar con las flechas reales, hasta que la diferencia de las mismas sea mínima. Se realizó éste proceso, cambiando únicamente en el programa principal la parte de la optimización, con dos bucles que hacen que varíen f´c y μ desde los valores de los vectores `lb ub` que corresponden al conjunto de límites inferior y superior de las variables de diseño. Dicho proceso tardo, para éste caso en particular, aproximadamente 16 horas para obtener los valores de las flechas, su diferencia con las flechas reales y además la sumatoria de las diferencias de las mismas elevadas al cuadrado. En la ***Figura 7*** se puede apreciar Σ(Flechas reales – Flechas calculadas)^2 cuando se varían los parámetros mecánicos.

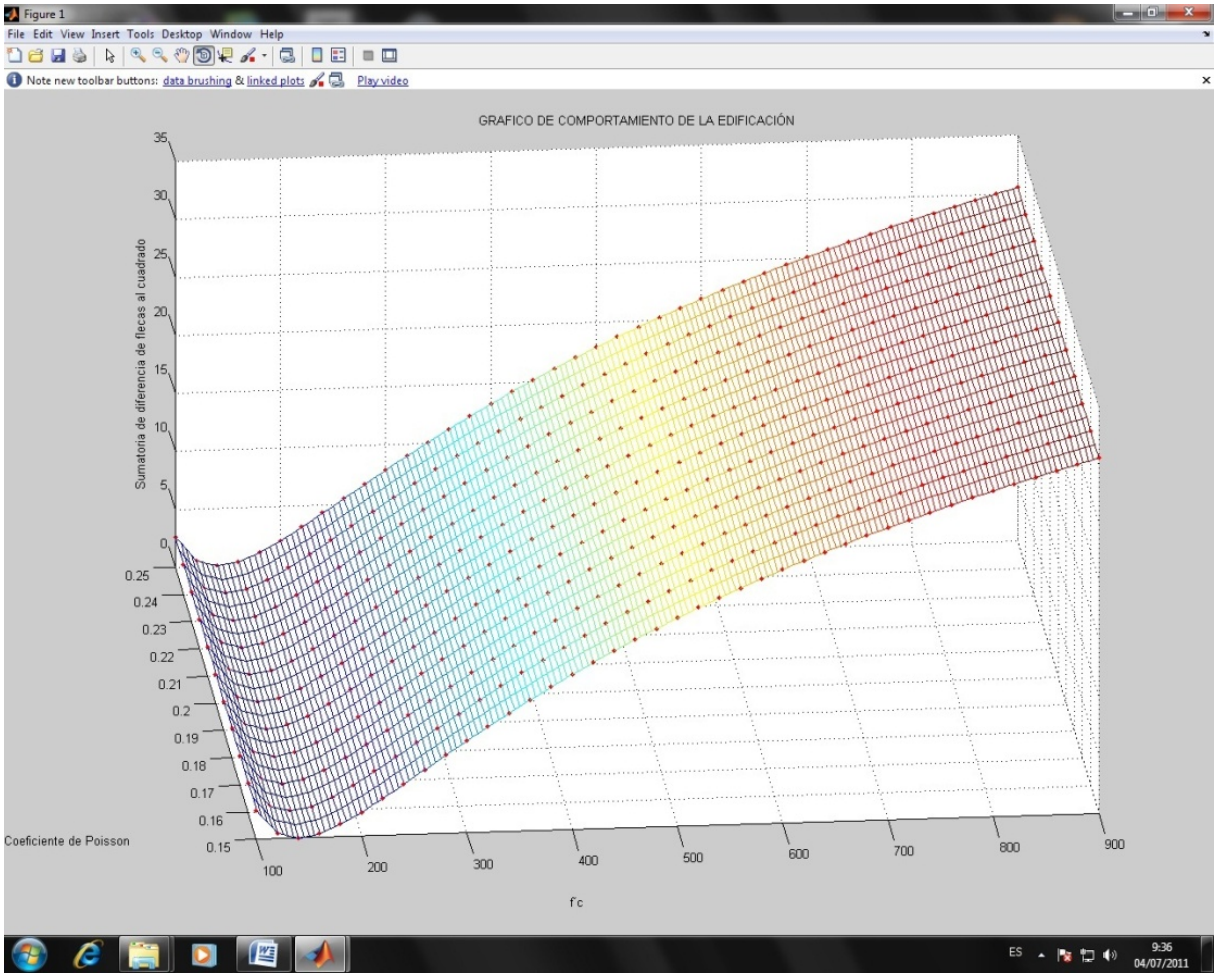


Figura 7: Σ(Flechas reales – Flechas calculadas)^2 al variar los parámetros mecánicos

Como se puede observar en la figura anterior el comportamiento de la sumatoria de las diferencias de las flechas elevadas al cuadrado tiene una forma parabólica y su valor mínimo está muy cercano a un hormigón con un f´c de 140 kg/cm^2. Además se puede decir que la variación del Coeficiente de Poisson (μ) no influye en el resultado de las diferencias de las flechas entre las calculadas y las reales, ya que, si se mantiene constante el f'c siempre me va a dar una línea paralela al eje de f´c en el eje de las Σ(Flechas reales – Flechas calculadas)^2 .

En cambio realizando la optimización con la función `lsqnonlin` se obtuvo los mismos resultados (f´c=138.9 kg/cm^2 μ=0.17) pero con una variación de tiempo considerable, aproximadamente 2 horas con lo queda justificado la aplicación de ésta función para la obtención de los parámetros mecánicos. Se puede acotar además que éste valor obtenido en la resistencia específica del hormigón muy cercano a 140 kg/cm^2 es muy bajo y representa a un hormigón muy pobre que no se considera estructural (hormigón estructural f´c≥ 210

kg/cm^2), lo que sería una de las causas para que las deflexiones en las vigas sean considerables. En la ***Figura 8*** se puede ver claramente los puntos rojos que representan como la función `lsqnonlin` varía los valores de los parámetros mecánicos (f´c, μ) hasta encontrar la respuesta más aproximada a la realidad.

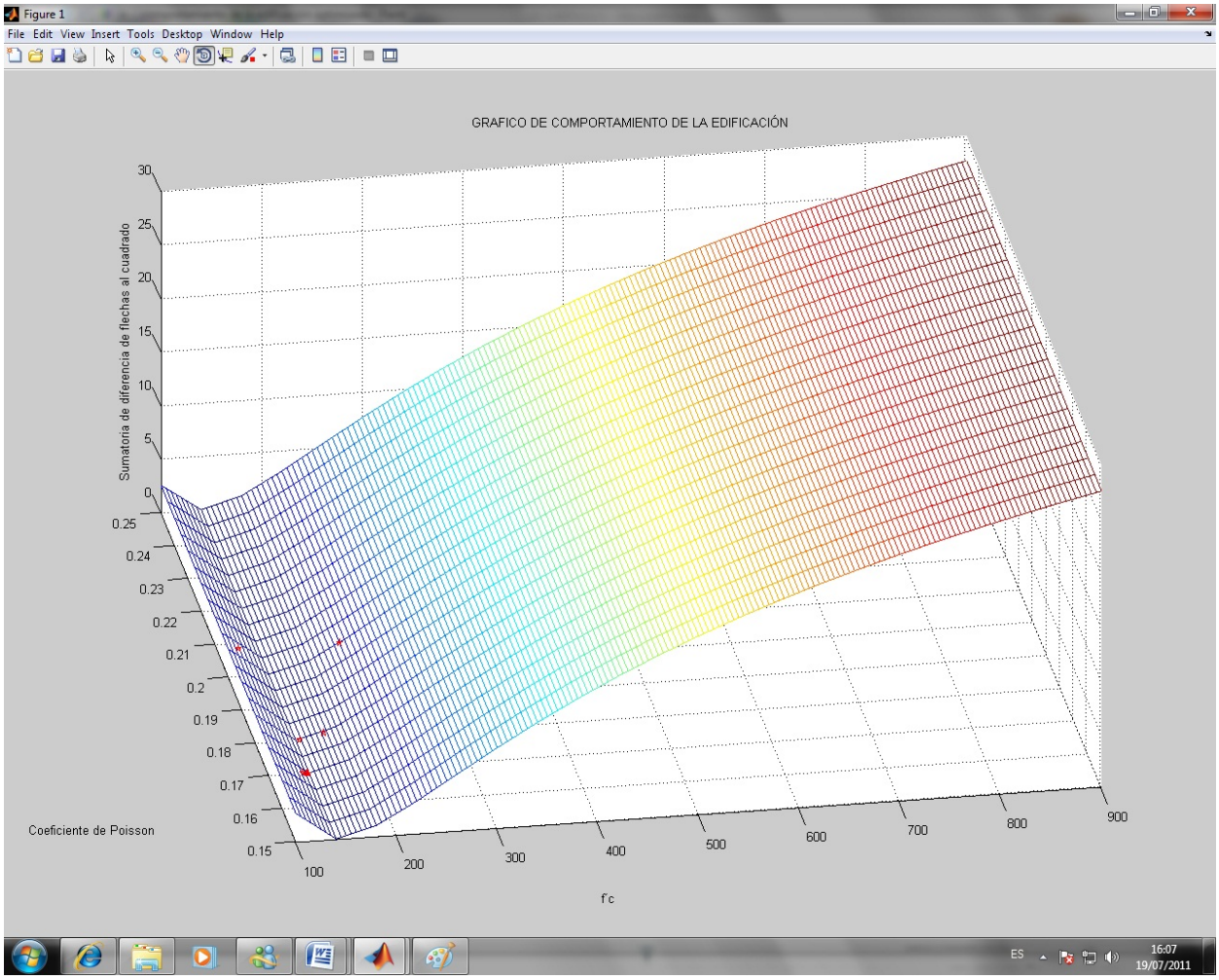


Figura 8: Σ(Flechas reales – Flechas calculadas)^2 optimizando los parámetros mecánicos

El programa realizando también da como resultado la siguiente tabla en un archivo de texto, el cual en sus primeras cuatro columnas se muestra los valores de los parámetros mecánicos que la función `lsqnonlin` fue adoptando en cada iteración, en la quinta columna en cambio se observa los valores de Σ(Flechas reales – Flechas calculadas)^2 y a partir de la sexta columna nos da los valores de las flechas calculadas en mm en los nodos analizados.

F´c	u	E	G	SunF2	Nodo1316	Nodo1317	Nodo1318	Nodo1319	Nodo1320	Nodo1321	Nodo1322	Nodo1323
110.0	0.20	150000.00	62500.00	2.653	1.820	2.775	3.709	4.394	4.724	4.507	3.613	2.469
210.0	0.20	217370.65	90571.10	2.554	1.256	1.915	2.559	3.032	3.260	3.110	2.493	1.704
167.7	0.18	194244.10	82253.20	0.593	1.405	2.144	2.866	3.395	3.651	3.482	2.791	1.906
133.6	0.17	173367.67	74389.78	0.114	1.575	2.402	3.212	3.806	4.092	3.903	3.128	2.136
143.6	0.18	179739.67	76467.70	0.068	1.519	2.317	3.097	3.670	3.946	3.764	3.017	2.060
138.1	0.17	176246.04	75186.37	0.068	1.549	2.363	3.159	3.743	4.025	3.839	3.077	2.101
136.8	0.17	175419.86	74881.75	0.077	1.556	2.374	3.174	3.761	4.044	3.857	3.091	2.111
139.1	0.17	176883.20	75393.86	0.064	1.543	2.354	3.148	3.729	4.010	3.825	3.066	2.094
138.4	0.17	176489.20	75249.36	0.066	1.547	2.359	3.155	3.738	4.019	3.834	3.072	2.098
138.9	0.17	176784.78	75357.77	0.064	1.544	2.355	3.149	3.731	4.012	3.827	3.067	2.095

En el cuadro anterior en su última fila se puede observar que las flechas calculadas en los nodos analizados se aproximan a las flechas reales que se impusieron como dato (página 39), cometiendo un error de 0.064 en la suma de las diferencias al cuadrado entre las flechas reales y las calculadas, cuyo valor se puede tomar como válido ya que es muy cercano a cero considerando las unidades de mm^2.

También hay que aclarar que el análisis que se ha realizado en ésta monografía trata de las deflexiones inmediatas y que no se ha considerado el agrietamiento y los efectos a largo plazo que son muy importantes para el cálculo de las deflexiones en los elementos estructurales. De ahí que sería importante trabajar y ahondar más en estos temas para futuras investigaciones, ya que son diversas las causas que conducen al agrietamiento del concreto, siendo las fundamentales, las deformaciones debidas a cambios volumétricos y los esfuerzos ocasionados por fuerzas de tensión, por momentos flexionantes, o por las fuerzas cortantes. El agrietamiento además si no se controla nos disminuye la sección y por ende el momento de inercia baja lo que ocasiona que las deflexiones aumenten ya que cambian las distribuciones de esfuerzos de tensión en el hormigón y el acero de refuerzo, disminuyendo la adherencia entre los mismos, lo que ocasionaría un incremento en el ancho de las grietas lo que influye a la vez en la protección del refuerzo contra la corrosión por eso es que preferible muchas grietas muy finas (capilares) que pocas grietas anchas.

Por otra parte los efectos a largo plazo en las deflexiones que son provocadas por los efectos de la contracción y la fluencia lenta, la formación de nuevas fisuras, y el ensanchamiento de las fisuras existentes.

Los efectos de la contracción y la fluencia lenta se deben estimar, ya que la distribución de las deformaciones y tensiones es variable en la altura y la longitud de la viga. Las propiedades de la sección (resistencia, módulo de elasticidad, contracción y fluencia lenta) también varían según la composición de la mezcla, las condiciones de curado y la edad.

CAPITULO CINCO : CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

5. CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

5.1. CONCLUSIONES

Se ha propuesto una nueva metodología para la revisión de edificaciones de hormigón armado existentes con la cual se obtienen los parámetros mecánicos (E, G, μ, f´c) de las vigas de la estructura analizada para lo cual se ha desarrollado un algoritmo que integra las técnicas de optimización por mínimos cuadrados y el cálculo estructural basado en elementos finitos. Dicho algoritmo ha sido implementado con éxito en el entorno de programación MATLAB para aplicarlo a la resolución de diferentes casos prácticos que se puedan presentar.

Lo que hay que recalcar de ésta nueva metodología es que puede implementarse directamente a un amplio espectro de problemas estructurales que se presentan independientemente de geometría y tamaño de la edificación existente de hormigón armado ya que como parte de un modelo estructural generado a través de las herramientas con las que cuenta en el programa de cálculo estructural SAP2000, se pueden abarcar casi en su totalidad a los casos reales presentados, todo va a estar en función de la calidad del modelo estructural plantado en el SAP2000.

Otra punto que hay que destacar es que la variación del Coeficiente de Poisson (μ) desde 0.15 a 0.25 no influye en el resultado cuando se calculan las flechas de los elementos analizados y por ende la diferencias de las flechas entre las calculadas y las reales tampoco varia.

5.2. FUTURAS LÍNEAS DE INVESTIGACIÓN

A partir de ésta metodología dada en el presente trabajo de investigación se pueden tomar en cuenta la influencia de otras variables en la deflexión de las vigas como pueden ser:

- Problemas de agrietamiento ver como varia la inercia, el modulo de elasticidad, la adherencia entre acero y el concreto.
- Efecto de las deflexiones a largo.

Todo lo anteriormente señalado pude ir a la par con una validación experimental en laboratorios en donde se pueda cumplir con todas las normas y restricciones que exige un trabajo de esta calidad.

REFERENCIAS

[1] **Borda J., Rodríguez G.**, *"Optimización de secciones transversales de Pórticos de Hormigón Armado mediante uso de la Técnica de Programación Cuadrática Secuencial SQP"*, Mecánica Computacional Vol. XXIX, Buenos Aires- Argentina, 2010.

[2] **Codigo Ecuatoriano de la Construcción CEC**, *"Requisitos generales de diseño"*, Instituto Ecuatoriano de normalización INEN, Quito-Ecuador, 2001.

[3] **Comité ACI 318**, *"Requisitos de Reglamento para Concreto Estructural (ACI 318S-08) y Comentario"*, American Concrete Institute, Farmington Hills, MI – USA, 2008.

[4] **Computers and Structures Inc.**, *"SAP2000 Advanced 12.0 Structural Analysis Program"*, User Guide, Berkeley, California – USA, 2008.

[5] **Garcia J., Rodriguez J., Vidal J.**, *"Aprenda MATLAB 7.0 como si estuviera en primero"*, Universidad Politécnica de Madrid, 2005.

[6] **Herrea C. H.,** , *"Curso de manejo Programa SAP2000 Diseño y Análisis por medio de elementos Finitos"*, 2004.

[7] **Holzleitner L., Mahmoud K. G.**, *"Structural Shape Optimization using MSC/NASTRAN and Sequential Quadratic Programming"*, Computers & Structures, 70,487-514, 1999.

[8] **Liu G.R., Han X.**, "*Computational inverse techniques in nondestructive evaluation"*, CRC Press, Florida – USA, 2003.

[9] **Navarrina F.**, *"Una metodología para optimización estructural en diseño asistido por ordenador"*, Universidad Politécnica de Catalunya, 1987.

[10] **Olhoff N., Bendsoe M.P. and Rasmussen J**., "*On CAD-Integrated Structural Topology and Design Optimization"*. Comp. Mech. In Appl. Mech. And Engng. 89, 259-279, 1991.

[11] **Oñate E**, "*Cálculo de Estructuras por el Método de Elementos Finitos"*. Centro Internacional de Métodos Numéricos en Ingeniería, 1995.

[12] **Payá I.**, "*Optimización Heurística de Pórticos de edificación de Hormigón Armado",* Universidad Politécnica de Valencia, 2007.

[13] **The Mathworks Inc.**, "*MATLAB 7.7(R2008b)",* User Guide, MA – USA, 2008.

[14] **Torrano S.**; "*DISHA: Diseño interactivo de secciones de hormigón armado".* Cartagena (España): Documento de trabajo interno. Departamento de Estructuras y Construcción. UPCT. 2000.

ANEXO 1: Programación en MATLAB

PROGRAMACIÓN EN MATLAB

En MATLAB se trabaja sobre `Command Window` que es la ventana inicial donde ingresamos comandos y los ejecutamos directamente. Frecuentemente una serie de comandos debe ser ejecutada varias veces durante una misma sesión, para evitarnos el trabajo de ingresarlos continuamente existen los scripts. Este término inglés script significa: escrito, guión, nota; el término guión es el que más se utiliza en las traducciones al español y no es más que una serie o conjunto de comandos que permitan realizar una tarea o función específica.

En el MATLAB se puede crear funciones y scripts, que no son más, que archivos de texto ASCII, con la extensión *.m, que contienen funciones o conjuntos de comandos respectivamente. MATLAB cuenta con un editor propio que permite tanto crear y modificar estos archivos *.m, en donde también se puede ejecutarlos paso a paso los comandos o funciones que contienen estos archivos, para ver si los mismos contienen o no errores (a esto se lo conoce como proceso de Debug o depuración, en donde, se eliminan errores del programa). También Matlab permite que utilicemos cualquier editor (edit de DOS, Word, Notepad, etc.), ya que los archivos son sólo de texto.

Otra forma de ejecutar un script es sencillamente introduciendo su nombre en la línea de comandos del `Command Window de MATLAB`, mientras que una función se ejecuta de forma parecida que un script pero va acompañada de sus argumentos que deben pasarse entre paréntesis y separados por coma.

Para poder realizar un programa o algoritmo en MATLAB se utilizan variables, funciones de librerías o comandos, sentencias. Las cuales se van a explicar a continuación:

- Una variable es un nombre que se da a una entidad numérica, que puede ser una matriz, un vector o un escalar. Por defecto MATLAB trabaja con variables de punto flotante y doble precisión (double), aunque también en estas variables se pueden guardar valores enteros, complejos, lógicos e inclusive cadena de caracteres que van entre apóstrofos o comillas simples.

Además el valor de la variable, e incluso el tipo de entidad numérica que representa, puede cambiar a lo largo de una sesión de MATLAB o a lo largo de la ejecución de un programa. La forma más normal de cambiar el valor de una variable es colocándola a la izquierda del operador (=).

```
variable = expresión
```

También se pueden incluir varias expresiones en una misma línea separándolas por comas (,) o puntos y comas (;). Si una expresión termina en punto y coma (;) su resultado se calcula, pero no se escribe en la pantalla. Esta posibilidad es muy interesante, tanto para evitar la escritura de resultados intermedios, como para evitar la impresión de grandes cantidades de números cuando se trabaja con matrices de gran tamaño.

A semejanza del lenguaje C, MATLAB distingue entre mayúsculas y minúsculas en los nombres de variables. Los nombres de variables deben empezar siempre por una letra y pueden constar de hasta 63 letras y números. El carácter guión bajo (_) se considera como una letra.

A diferencia del lenguaje C, no hace falta declarar las variables que se vayan a utilizar. Esto hace que se deba tener especial cuidado con no utilizar nombres erróneos en las variables, porque no se recibirá ningún aviso del ordenador.

- Una función de librería o comando son funciones incorporadas en el propio código ejecutable del programa, existen muchas de estas funciones incorporadas en MATLAB. Estas funciones de librería se ejecutan de la misma forma que las funciones creadas por los usuarios en los archivos de texto *.m .

  Toda función contiene tres elementos: nombre de la función, valor o valores de retorno y argumento o argumentos de entrada.

  ```
  [valores de retorno]=nombre de la función(argumentos de entrada)
  ```

Los argumentos de entrada de cada función van a continuación del nombre de la función entre paréntesis (y separados por comas si hay más de uno). Los valores de retorno son el resultado de la función y sustituyen a ésta en la expresión donde la función aparece y van entre corchetes (y separados por comas) pero si solo existe un valor de retorno no es necesario colocar corchetes. Existen pero son muy pocas las funciones de librería que no tienen argumentos de entrada por lo que solo para estos casos no llevarían paréntesis.

Las funciones de MATLAB nunca devuelven modificadas las variables que se pasan como argumentos de entrada, se podía decir entonces que, los argumentos de entrada de una función de MATLAB siempre se pasan por valor y nunca por referencia.

- MATLAB al igual que cualquier otro lenguaje de programación dispone de sentencias para realizar bifurcaciones y bucles. Las bifurcaciones son las que nos permiten realizar una u otra operación según se cumpla o no una determinada condición. Se puede observar en la ***Figura 9*** las posibles formas de bifurcaciones que se pueden realizar en MATLAB.

  Los bucles son los que permiten repetir las mismas o análogas operaciones sobre datos distintos. En la Figura 10 se muestran dos posibles formas de bucles, con el control situado al principio o al final del mismo.

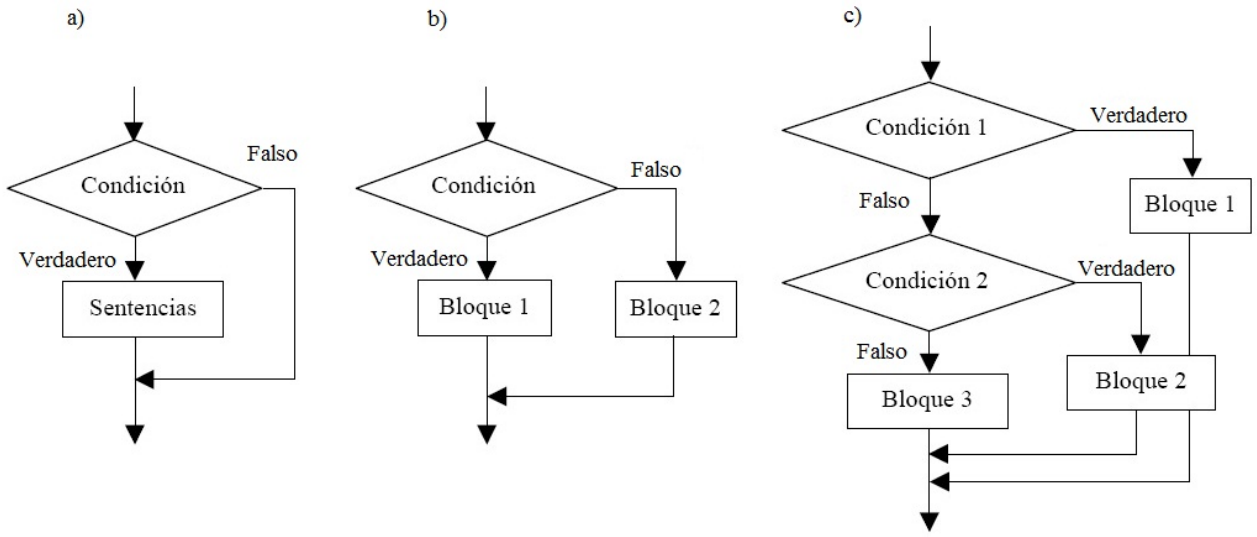


Figura 9: Gráficos de bifurcaciones con los que cuenta el MATLAB

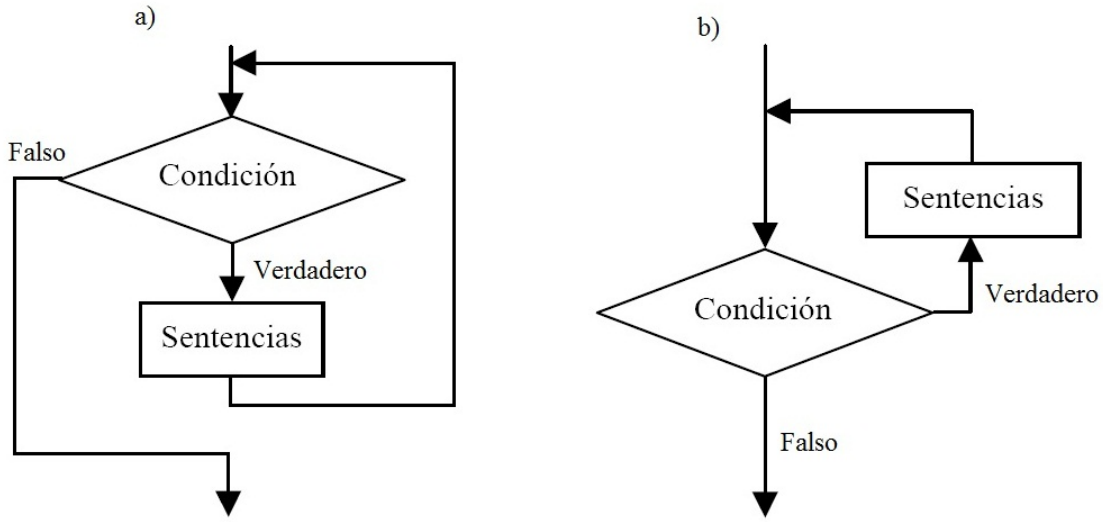


Figura 10: Gráficos de bucles con control al principio y final

FUNCIONES DE LIBRERÍA O COMANDOS UTILIZADOS

Las siguientes funciones de librerías o comandos del MATLAB se han empleado para realizar el programa que sea capaz de encontrar los parámetros mecánicos de las vigas de una edificación existente y a la vez que interactúe con el SAP2000:

- ADDPATH: Éste comando adicionar nuevos directorios al Path del MATLAB, sus sintaxis más utilizadas son las siguientes:

```
addpath('Directorio a adicionar');
addpath('Directorio a adicionar 1', 'Directorio a adicionar 2',...);
```

- CLS: Limpia la ventana de Command, su sintaxis es la siguiente:

```
clc;
```

- CD: Cambia a otro directorio en donde se va a trabajar, su sintaxis más usada es la siguiente:

```
cd('Directorio al que se va a cambiar');
```

- CLEAR: Éste comando se utiliza para borrar variables y funciones, su sintaxis más usadas son las siguientes:

`clear` Sin argumentos, clear elimina todas las variables creadas (excepto las variables globales).

`clear A, b`	Borra las variables indicadas en este caso `A` y `b`.
`clear global`	Borra las variables globales.
`clear functions`	Borra las funciones.
`clear all`	Borra todas las variables, incluyendo las globales, y las funciones.

- DISP: Imprime en la pantalla un mensaje de texto o el valor de una matriz, pero sin imprimir su nombre. En realidad, `disp` siempre imprime vectores y/o matrices: las cadenas de caracteres son un caso particular de vectores, su sintaxis más usada es la siguiente:

`disp(A);`	Imprime en la pantalla el contenido de la variable A

- DOS: Nos sirve para ejecutar un comando del DOS, su sintaxis más usada es la siguiente:

```
dos('Comando de DOS a ser ejecutado');
```

- FPRINTF: Dirige su salida formateada hacia la pantalla o un fichero indicado por su identificador, sus sintaxis más generales son:

```
fprintf (fi,'Cadena de Texto',variable1,variable2,...);
```

En donde `fi` es el indicador del fichero o archivo en donde desea escribirse la salida de `fprint`.

```
fprintf ('Cadena de Texto',var1,var2,...);
```

Se imprime en la pantalla lo que se indica en `fprint`.

- GLOBAL: Sirve para declarar que una o más variables son globales, para realizar esto se coloca la palabra global y luego las variables que se van a declarar separadas por espacios en blanco, su sintaxis es la siguiente:

`global A b C;` Se declara como globales la variables `A`, `b` y `C` .

- IMPORTDATA: Carga los datos de un archivo al espacio de trabajo o a una variable, su sintaxis más usada es la siguiente:

```
Variable= importdata(Nombre del archivo);
```

- INPUT: Permite imprimir un mensaje en la línea de comandos de MATLAB y recuperar como valor de retorno un valor numérico o el resultado de una expresión tecleada por el usuario. Después de imprimir el mensaje, el programa espera que el usuario teclee el valor numérico o la expresión. Cualquier expresión válida de MATLAB es aceptada por este comando. El usuario puede teclear simplemente un vector o una matriz. En cualquier caso, la expresión introducida es evaluada con los valores actuales de las variables de MATLAB y el resultado se devuelve como valor de retorno, para ello la sintaxis es la siguiente:

```
A = input ('Mensaje');
```

Otra posible forma de esta función o comando es la siguiente (obsérvese el parámetro 's'):

```
texto = input('Mensaje','s');
```

En este caso el texto tecleado como respuesta se lee y se devuelve sin evaluar, con lo que se almacena en variable texto. Así pues, en este caso, si se teclea una fórmula, se almacena como texto sin evaluarse.

- PAUSE: Realiza una pausa en la corrida del programa, su sintaxis más usada es la siguiente:

`pause(n);` Realiza una pausa de n segundos en la corrida del programa.

- XLSREAD: Se utiliza para leer datos de un archivo de Excel, las sintaxis que utilizaremos para la creación del programa son las siguientes:

```
Variable=xlsread(Nombre de archivo de Excel,Hoja)
Variable=xlsread(Nombre del archivo de Excel,Hoja,'Rango')
```

En la primera expresión se guarda en `Variable` todo en contenido de la hoja de Excel que se lo indica. En cambio para la segunda expresión guarda en `Variable` el contenido del rango señalado de la hoja de Excel especificada.

- XLSWRITE: Se utiliza para escribir datos en un archivo de Excel, las sintaxis que utilizaremos en el programa es la siguiente:

```
xlswrite (Nombre de archivo de Excel,A,Hoja,'Rango')
```

En ésta expresión se escribe el contenido de la variable `A` en rango señalado de la hoja de Excel especificada.

SENTENCIA UTILIZADA

Una de las sentencias que se utilizo fue la sentencia de bifurcación IF en su forma más simple que corresponde a la ***Figura 9 a*** que se escribe de la siguiente forma:

```
if condición
   sentencia
end
```

La sentencia para los bucles que se empleo es la FOR la cual repite un conjunto de operaciones un número predeterminado de veces y corresponde a la ***Figura 10 b*** que se escribe de la siguiente forma:

```
for i=1:n
    sentencias
end
```

ANEXO 2: Programa creado en MATLAB para el cálculo de las propiedades mecánicas en vigas

```
% Programa que encuentra los valores de los parámetros mecánicos de vigas
% de edificaciones existentes mediante el paquete de optimización del
% MATLAB y el programa de calculo estructural SAP2000 basado en elementos
% finitos
% Se va adicionar nuevos directorios de trabajo en el Path del Matlab
addpath('C:\Monografia\CorridaMatlab');
addpath('C:\Monografia\CorridaMatlab\MIMICA_MOVIMIENTO');
addpath('C:\Monografia\PorticosSap');
addpath('C:\Monografia\CorridaSap');
clc; % Para limpiar la ventana del Command
clear all % Se borra todas las variables y funciones del Workspace
% Se va eliminar todo el contenido del directorio
% C:\Monografia\CorridaSap\ para poder trabajar en el mismo
cd('C:\Monografia\CorridaSap\');% Se va ha cambiar de al directorio que se
borrara
comandoborrar=['del *.* /Q'];% Se guarda la orden de borrado para DOS
dos(comandoborrar);% Se ejecuta la orden de borrado para el DOS
cd('C:\Monografia\CorridaMatlab\');% Se cambia a este nuevo directorio
% de Trabajo del Matlab
% Se declaran como globales a estas variables
global NumMaterial NumCombinaciones NombreArchivoSap Nodos FlechasRealesMm
m direccion3 NumCorridasSap2000 ValoresParametrosMecanicos FlechasTotales
DiferenciaFlechaTotales ValorSumaCuadrados TolXv T1 T2
% Se va copiar un archivo C:\Monografia\CorridaMatlab\limpiar.bat
% al C:\Monografia\CorridaSap\limpiar.bat que se necesita cada vez para
% eliminar todos los archivos secundarios que genera el SAP2000 al
% ejecutarce
Auxiliar1='copy C:\Monografia\CorridaMatlab\limpiar.bat C:
\Monografia\CorridaSap\limpiar.bat';
dos(Auxiliar1);%Ejecuto la orden de copiado para el DOS
clc
% Se va a leer el archivo de datos para la corrida, este archivo es un
% archivo de texto *.txt que contiene en su primera línea
% el nombre del pórtico,en la segunda fila tiene el número de combinaciones
% y el numero del material. En su tercera fila los tiempos en segundos
% para abrir el archivo del SAP200 y el tiempo que se demora en correr
% el archivo en el SAP200 y en las siguientes líneas contiene el número de
% nodo y la flecha real medida en ese nodo
ArchivoDatos=input('Ingrese el nombre del archivo de datos ','s');
NombreArchivoDatos=[ArchivoDatos,'.txt'];
Datos=importdata(NombreArchivoDatos);
NombreArchivoSap=Datos.textdata{1,1};
NumCombinaciones=Datos.data(1,1);
NumMaterial=Datos.data(1,2);
T1=Datos.data(2,1);
T2=Datos.data(2,2);
Nodos=Datos.data(3:end,1);
FlechasRealesMm=Datos.data(3:end,2);
```

```
Tamano=size(Nodos);% Tamaño de la matriz de nodos
m=Tamano(1,1);% Número de nodos que se va analizar
clc
% Se copiará el archivo C:\NombreArchivoSap.$2K
% al C:\Monografia\CorridaSap\NombreArchivoSap.$2K
direccion1=['C:\',NombreArchivoSap,'.$2K'];
direccion2=['C:\Monografia\CorridaSap\',NombreArchivoSap,'.$2K'];
comandocopy=['copy ',direccion1,' ',direccion2];% Se coloca la orden de
copiado para DOS
dos(comandocopy);% Ejecuto la orden de copiado para el DOS
clc
disp('ESTA CORRIENDO EL PROGRAMA SAP2000.....');
% Se llama a la función que realiza la corrida del Programa SAP2000
% La variable de NombreArchivoSap es solo el nombre del portico que se
% quiere ejecutar la corrida en el SAP2000 sin ninguna extención
% La variable direccion2 contiene la dirección completa del archivo a abrir
% en el SAP2000 con su extención (la cual puede ser *.$2K *.SDB *.XLS)
NumCorridasSap2000=1;
CorridaProgramaSap2000(NombreArchivoSap,direccion2,T1,T2)
clc
direccion3=['C:\Monografia\CorridaSap\portico.xls'];%Archivo de exel
% Función que realiza la lectura de datos de la base de datos en Excel del
% portico corrido del Programa SAP2000
[Fc1,E1,U1,G1,Flechas1]=LecturaDatosSap2000(direccion3,NumMaterial,Nodos,m,
NumCombinaciones);
% Se va a guardar los valores de los parámetros mecánicos en la matriz
% ValoresParametrosMecanicos
ValoresParametrosMecanicos(1,1)=Fc1;
ValoresParametrosMecanicos(2,1)=U1;
ValoresParametrosMecanicos(3,1)=E1;
ValoresParametrosMecanicos(4,1)=G1;
Flechas=Flechas1';
FlechasTotales(:,1)=Flechas;
% Se va a comparar los valores de las flechas para ver si es necesario
% realizar la optimización
DiferenciaFlecha=(FlechasRealesMm-Flechas);
DiferenciaFlechaTotales(:,1)=DiferenciaFlecha;
suma=0;
for i=1:m
suma=suma+(DiferenciaFlecha(i)).^2;
end;
ValorSumaCuadrados(1)=suma;
if suma>0.01
disp('Esta corriendo la optimización con Matlab y Sap2000');
%=========================================================================

%Optimización de lsqnonlin para calcular los coeficientes de
% una función por el método de minimos cuadrados
```

```
%=========================================================================
% Forma de emplear la Función lsqnonlin
%[x,resnorm,residual,exitflag,output,lambda,jacobian] = ....
%..... lsqnonlin(fun,x0,lb,ub,options)

%=========================================================================

% Argumentos de Entrada para esta aplicación

%=========================================================================
X0=[210,0.2];% Punto de partida para la solución f'c=210Kg/cm2 u=0.20
Xlb=[100,0.15];% Límite inferior de la solución f'c=100Kg/cm2 u=0.15
Xlu=[900,0.25];% Límite superior de la solución f'c=900Kg/cm2 u=0.25
TolXv=1;
TolFunv=0.001;
% Se va a reducir los argumentos de entrada a la unidad para poder llamar a
% la función lsqnonlin
Escala=[100,0.1];% Vector de escala que reduce los argumentos de entrada a
la unidad
X0esc=X0./Escala;% Punto de partida para la solución reducido a la unidad
Xlbesc=Xlb./Escala;% Límite inferior de la solución reducido a la unidad
Xluesc=Xlu./Escala;% Límite superior de la solución reducido a la unidad
opciones= optimset('Jacobian','on','Hessian', 'off','LargeScale','on',...
'TolX',TolXv,'TolFun',TolFunv,'MaxIter',800,'MaxFunEvals',
800,'DiffMaxChange',1,'DiffMinChange',0.1);
%=========================================================================

% Aplicación de la función lsqnonlin

%=========================================================================
[X,Resnorm,Residuo,EXITFLAG,OUTPUT,LAMBDA,JACOB]=lsqnonlin
(@funInterfaceSapJ,X0esc,Xlbesc,Xluesc,opciones);
X1=X(1,1)+0.1;
X2=X(1,2)+0.1;
X0esc=[X1,X2];% Punto de partida para aproximarse a la solución
TolXv=0.1;
opciones= optimset('Jacobian','on','Hessian', 'off','LargeScale','on',...
'TolX',TolXv,'TolFun',TolFunv,'MaxIter',800,'MaxFunEvals',
800,'DiffMaxChange',0.1,'DiffMinChange',0.01);
[X,Resnorm,Residuo,EdXITFLAG,OUTPUT,LAMBDA,JACOB]=lsqnonlin
(@funInterfaceSapJ,X0esc,Xlbesc,Xluesc,opciones);
X1=X(1,1)+0.01;
X2=X(1,2)+0.01;
X0esc=[X1,X2];% Punto de partida para aproximarse más a la solución
TolXv=0.01;
opciones= optimset('Jacobian','on','Hessian', 'off','LargeScale','on',...
'TolX',TolXv,'TolFun',TolFunv,'MaxIter',800,'MaxFunEvals',
800,'DiffMaxChange',0.01,'DiffMinChange',0.001);
[X,Resnorm,Residuo,EdXITFLAG,OUTPUT,LAMBDA,JACOB]=lsqnonlin
```

```
(@funInterfaceSapJ,X0esc,Xlbesc,Xluesc,opciones);

%=========================================================================

end;
% Matriz de resultados
Resultados=ValoresParametrosMecanicos';
Resultados(:,5)=ValorSumaCuadrados(:)';
Flecha=FlechasTotales';
Nodo=Nodos';
fprintf(' F´c u E G SunF2 ');
fprintf('Nodo%g ',Nodo);
fprintf('\n');
k=NumCorridasSap2000;
for i=1:k;
fprintf('%3.1f %1.2f %7.2f %7.2f %2.3f ',Resultados(i,1),
Resultados(i,2),Resultados(i,3),Resultados(i,4),Resultados(i,5));
fprintf('%1.3f ',Flecha(i,:));
fprintf('\n');
end;
NombreArchivoSalida=['SalidaConOptimizacionDe',NombreArchivoSap,'.txt'];
ArchivoSalida=fopen(NombreArchivoSalida,'w');
fprintf(ArchivoSalida,' F´c u E G SunF2 ');
fprintf(ArchivoSalida,'Nodo%g ',Nodo);
fprintf(ArchivoSalida,'\n');
for i=1:k;
fprintf(ArchivoSalida,'%3.1f %1.2f %7.2f %7.2f %2.3f ',
Resultados(i,1),Resultados(i,2),Resultados(i,3),Resultados(i,4),Resultados
(i,5));
fprintf(ArchivoSalida,'%1.3f ',Flecha(i,:));
fprintf(ArchivoSalida,'\n');
end;
fclose(ArchivoSalida);
% Grafico de resultados
A=Resultados;
x=A(:,1);
y=A(:,2);
z=A(:,5);
figure(1);
plot3(x,y,z,'r.');
grid
title('GRAFICO DE COMPORTAMIENTO DE LA EDIFICACIÓN')%titulo del grafico
xlabel('f´c')%Etiqueta al eje x
ylabel('Coeficiente de Poisson')%Etiqueta al eje y
zlabel('Sumatoria de diferencia de flechas al cuadrado')%Etiqueta al eje z
```

ANEXO 3: Función creada en MATLAB para realizar la interface entre MATLAB y SAP2000

```
% Función de interface entre MATLAB y SAP2000
function [DifFlecha J]=funInterfaceSapJ(X)
global NumMaterial NumCombinaciones NombreArchivoSap Nodos FlechasRealesMm
m direccion3 NumCorridasSap2000 ValoresParametrosMecanicos FlechasTotales
DiferenciaFlechaTotales ValorSumaCuadrados TolXv T1 T2

%Se va ha eliminar todos los archivos secundarios que genera el SAP2000 al
% ejecutarce
cd('C:\Monografia\CorridaSap\');% Cambio de directorio
Comandolimpiar=['limpiar.bat'];% Orden de limpiado de archivos para el DOS
dos(Comandolimpiar);% Ejecuto la orden de limpiado de archivos para el DOS

clc

cd('C:\Monografia\CorridaMatlab\');% Se cambia a este nuevo directorio
% Se va copiar un archivo C:\Monografia\CorridaSap\portico.xls
% al C:\Monografia\CorridaSap\NombreArchivoSap.xls
comandocopy2=['Copy ',direccion3,' C:\Monografia\CorridaSap\',
NombreArchivoSap,'.xls'];% pongo la orden de copiado para DOS
dos(comandocopy2);% Ejecuto la orden de copiado para el DOS

clc

disp('Esta corriendo la optimización con Matlab y Sap2000 Cambiando Valores
de f´c u E G');
% Se va ha calcular los valores de F´c u y E
FcOptimoKgm2=X(1,1).*1000000;
uOptimo=X(1,2)./10;
EOptimoKgm2=150000000*((FcOptimoKgm2./10000)^(0.5));
% Se cambia el valor de f'c del archivo
% C:\Monografia\CorridaSap\NombreArchivo.xls
% xlswrite (nombre de archivo, M, hoja, rango)
ArchivoExcel=['C:\Monografia\CorridaSap\',NombreArchivoSap,'.xls'];%Archivo
de exel
HojaExcelFc='MatProp 03b - Concrete Data';%hoja del archivo de excel que
tiene la informacion
RangoFc=['B',num2str(NumMaterial+1,2)];
xlswrite(ArchivoExcel,FcOptimoKgm2,HojaExcelFc,RangoFc);%Se escribe en
% el archivo de Excel
% Se cambia el valor de E del archivo
% C:\Monografia\CorridaSap\NombreArchivo.xls
% xlswrite (nombre de archivo, M, hoja, rango)
HojaExcelE='MatProp 02 - Basic Mech Props';%hoja del archivo de excel que
tiene la informacion
RangoE=['D',num2str(NumMaterial+3,2)];
xlswrite(ArchivoExcel,EOptimoKgm2,HojaExcelE,RangoE);%Se escribe en
% el archivo de Excel
% Se cambia el valor de u del archivo
% C:\Monografia\CorridaSap\NombreArchivo.xls
% xlswrite (nombre de archivo, M, hoja, rango)
```

```
Rangou=['F',num2str(NumMaterial+3,2)];
xlswrite(ArchivoExcel,uOptimo,HojaExcelE,Rangou);%Se escribe en
% el archivo de Excel

clc

disp('ESTA CORRIENDO EL PROGRAMA SAP2000.....');
% Se llama a la función que realiza la corrida del Programa SAP2000
% La variable de NombreArchivoSap es solo el nombre del portico que se
% quiere ejecutar la corrida en el SAP2000 sin ninguna extención
% La variable direccion2 contiene la dirección completa del archivo a abrir
% en el SAP2000 con su extención (la cual puede ser *.$2K *.SDB *.XLS)
NumCorridasSap2000=NumCorridasSap2000+1
Fcactual=X(1,1).*100;
fprintf('f´c = %3.2f ',Fcactual);
%X(1,2)./10
pause(2);
CorridaProgramaSap2000(NombreArchivoSap,ArchivoExcel,T1,T2)
clc
disp('Esta corriendo la optimización con Matlab y Sap2000 Obteniendo las
Flechas del Modelo');
% Función que realiza la lectura de datos de la base de datos en Excel del
% portico corrido del Programa SAP2000
[Fci,Ei,Ui,Gi,Flechasi]=LecturaDatosSap2000(direccion3,NumMaterial,Nodos,m,
NumCombinaciones);
% Se va a guardar los valores de los parámetros mecánicos en la matriz
% ValoresParametrosMecanicos
ValoresParametrosMecanicos(1,NumCorridasSap2000)=Fci;
ValoresParametrosMecanicos(2,NumCorridasSap2000)=Ui;
ValoresParametrosMecanicos(3,NumCorridasSap2000)=Ei;
ValoresParametrosMecanicos(4,NumCorridasSap2000)=Gi;
Flechas=Flechasi';
FlechasTotales(:,NumCorridasSap2000)=Flechas;
% Se va a calcular los valores de las diferencias entre flechas
DiferenciaFlecha=abs(FlechasRealesMm-Flechas);
DiferenciaFlechaTotales(:,NumCorridasSap2000)=DiferenciaFlecha;
suma=0;
for i=1:m
suma=suma+(DiferenciaFlecha(i)).^2;
end;
ValorSumaCuadrados(NumCorridasSap2000)=suma;
% Calculo del Jacobiano
for i=m
J(i,1)=DiferenciaFlecha(i)./(TolXv);
J(i,2)=DiferenciaFlecha(i)./(TolXv);
end
DifFlecha=DiferenciaFlecha
```

```
% Función que ejecuta la corrida de un archivo del Programa SAP2000
% La variable de NombreArchivoSap es solo el nombre del portico que se
% quiere ejecutar la corrida en el SAP2000 sin ninguna extención
% La variable direccion contiene la dirección completa del archivo a abrir
% en el SAP2000 con su extención (la cual puede ser *.$2K *.SDB *.XLS)
function CorridaProgramaSap2000(NombreArchivoSap,direccion,T1,T2)
direccionSap=["",direccion,'" F &'];
dos(direccionSap);% Se abre el portico en el Sap2000
pause(T1)% Tiempo en segundos para cargarse el archivo de SAP2000
VentanaSap=['SAP2000 v12.0.0 Advanced - ',NombreArchivoSap];% Nombre de la
ventana en el SAP2000
waitforwindow(VentanaSap);% Se espera que se carge la ventana del archivo
en el Sap2000
pause(5);% Realiza un pausa de 5 segundos
siminput('mouseMLC',[.5 .5]) % Click adentro del Sap
siminput('keyb','{F5}') ; % Se ordena el Run al Sap2000
pause(4);% Realiza un pausa de 4 segundo
waitforwindow('Set Load Cases to Run');% Se espera que se carge la ventana
de Run
siminput('mouseMLC',[.286 .617]) % Click adentro del Sap
pause(1)
siminput('mouseMLC',[.688 .604]) % Click adentro del Sap
%siminput('keyb','{ENTER}');% Se hace un enter en el Run del SAp2000
pause(T2);% Tiempo en segundos para Run del archivo de SAP2000
siminput('mouseMLC',[.5 .5]) % Click adentro del Sap
pause(1)% Realiza un pausa de 1 segundo
waitforwindow(VentanaSap);maximizewindow;% espera que se carge la ventana
% y se expande la ventana del archivo en el Sap2000
pause(1);% Realiza un pausa de 1 segundo
siminput('mouseMLC',[.99 .01]) % Se hace un click en la esquina superior
derecha del
%programa para cerrar el SAp2000
pause(0.4)% Realiza un pausa de 0.4 segundos
waitforwindow('SAP2000');%Se espera que se abra la ventana para guardar el
archivo
pause(0.4);% Realiza un pausa de 0.4 segundos
siminput('keyb','{ENTER}');% Se hace un {enter} en {yes} para que guarde el
archivo
pause(6)% Realiza un pausa de 6 segundos
waitforwindow('C:\Windows\system32\cmd.exe');%Se espera que se active la
ventana del Dos
pause(1)% Realiza un pausa de 1 segundo
siminput('keyb','{exit}{ENTER}');%Se ordena que se salga de la ventana del Dos

clc
```

```
% Función que realiza la lectura de datos de la base de datos en Excel del
% portico corrido del Programa SAP2000
function [Fc,E,U,G,Flechas]=LecturaDatosSap2000(ArchivoExcel,NumMaterial,
Nodos,NumeroNodos,NumCombinaciones)
% Se busca el valor de f'c
% xlsread(nombre de archivo Excel,hoja)
HojaExcelFc='MatProp 03b - Concrete Data';%hoja del archivo de excel que
tiene la informacion
DatosDeFc= xlsread(ArchivoExcel,HojaExcelFc);%Leo el archivo de Excel y lo
guardo en la variable
FcKgm2=DatosDeFc(NumMaterial-2,1);
FcKgcm2=FcKgm2/10000;
FcKgmm2=FcKgcm2/100;
Fc=FcKgcm2;
% Se busca el valor de E
% xlsread(nombre de archivo Excel,hoja)
HojaExcelE='MatProp 02 - Basic Mech Props';%hoja del archivo de excel que
tiene la informacion
DatosDeE= xlsread(ArchivoExcel,HojaExcelE);%Leo el archivo de Excel y lo
guardo en la variable
EKgm2=DatosDeE(NumMaterial,3);
EKgcm2=EKgm2/10000;
EKgmm2=EKgcm2/100;
E=EKgcm2;
% Se busca el valor de U
U=DatosDeE(NumMaterial,5);
% Se busca el valor de G
GKgm2=DatosDeE(NumMaterial,4);
GKgcm2=GKgm2/10000;
GKgmm2=GKgcm2/100;
G=GKgcm2;
% Se busca el valor de las flecha en los nodos correspondientes
% xlsread(nombre de archivo,hoja)
HojaExcelFlechas='joint Displacements';%hoja del archivo de excel que tiene
la informacion
DatosDeFlecha= xlsread(ArchivoExcel,HojaExcelFlechas);%Se lee el archivo de
Excel y lo guardo en la variable
for i=1:NumeroNodos
FlechaMin=DatosDeFlecha((Nodos(i))*(NumCombinaciones+1),7);
FlechaMax=DatosDeFlecha((Nodos(i))*(NumCombinaciones+1)-1,7);
FlechaM=max(abs(FlechaMin),abs(FlechaMax));
FlechaCm=FlechaM*100;
FlechaMm=FlechaM*1000;
Flechas(i)=FlechaMm;
end
```

ANEXO 4: Función waitforwindow del paquete de Mímica de Movimiento

```
function waitforwindow(name,Tmax)
% waitforwindow('windowname',Tmax)
%
% Waits for a window with the specified name for Tmax seconds
if nargin==1
Tmax=3;
end
nu=cputime;
w=0;
while ~strncmp(name,getwindowname,length(name))
pause(.2)
if cputime-nu>5*Tmax
error(['Window ''' name ''' not found within ' num2str(5*Tmax) '
seconds.'])
elseif cputime-nu>Tmax&w==0
warning(['Window ''' name ''' not found within ' num2str(Tmax) '
seconds.'])
end
end
```

ANEXO 5: Función siminput del paquete de Mímica de Movimiento

```
function siminput(varargin)
% Simulate user input
%
% SIMINPUT('type1','args1','type2','args2',...)
%
% 'type' describes the type of user input. Possible values are
% 'keyb' : keyboard input
% 'mouseM' : mouse move
% 'mouseLC' : left click
% 'mouseRC' : right click
% 'mouseMLC' : mouse move, followed by left click
% 'mouseMRC' : mouse move, followed by right click
%
% 'args' gives the arguments for the action.
% a string giving the desired keystrokes (see STR2KC)
% a matrix with two coordinates for the mousemove actions
% an empty matrix for the static mouse actions
while ~isempty(varargin)
typ=varargin{1};
args=varargin{2};
switch typ
case 'keyb'
[KC,ACT]=str2kc(args);
keycontrol(KC,ACT)
case 'mouseM'
if length(args)==2
mousecontrol(2,2^16*args)
end
case 'mouseLC'
mousecontrol(1,1)
case 'mouseRC'
mousecontrol(1,2)
case 'mouseMLC'
if length(args)==2
mousecontrol(2,2^16*args)
end
mousecontrol(1,1)
case 'mouseMRC'
if length(args)==2
mousecontrol(2,2^16*args)
end
mousecontrol(1,2)
otherwise
error('unknown input command')
end
varargin=varargin(3:end);
pause(.2)
end
```

ANEXO 6: Función str2kc del paquete de Mímica de Movimiento

```
function [KC,ACT]=str2kc(inp)
% converts a string to a KeyCode and an Action
%
% The input is given as a string, special keys between accolades,
% like {ENTER}, {TAB}, ... Combinations with Control, Alt or Shift are
% done with ^, % and + respectively (as for the function SendKeys in
% Excel). If it is the purpose to send one of these strokes, they should
% be between accolades. Parentheses also have a special meaning: ^, % or
% + followed by (...) means that the control, alt or shift key is
% pressed for all of the keys between ().
%
% Example:
% +abcd results in Abcd
% +(abcd) results in ABCD
% %{F4} results in Alt-F4
% ^s results in Ctrl-s
% ABCD123 results in ABCD123
%
% KC contains an array of KeyCodes (see MSDN website for information) and
% ACT defines the action: 1 = press, 2 = release and 3 = press and
% immediately release.
KC=[];
ACT=[];
if isstr(inp)
str=inp;
l=length(str);
k1=0;
while k1<l
k1=k1+1;
t1=str(k1);
switch t1
case '{'
k2=k1+1;
t2=str(k2);
while t2~='}'
k2=k2+1;
t2=str(k2);
end
switch str(k1+1:k2-1)
case 'BACKSPACE'; KC=[KC 8];ACT=[ACT 3];
case 'TAB'; KC=[KC 9];ACT=[ACT 3];
case 'ENTER'; KC=[KC 13];ACT=[ACT 3];
case 'PAUSE'; KC=[KC 19];ACT=[ACT 3];
case 'CAPSLOCK'; KC=[KC 20];ACT=[ACT 3];
case 'ESC'; KC=[KC 27];ACT=[ACT 3];
case 'SPACE'; KC=[KC 32];ACT=[ACT 3];
case ' '; KC=[KC 32];ACT=[ACT 3];
case 'PGUP'; KC=[KC 33];ACT=[ACT 3];
case 'PGDN'; KC=[KC 34];ACT=[ACT 3];
case 'END'; KC=[KC 35];ACT=[ACT 3];
case 'HOME'; KC=[KC 36];ACT=[ACT 3];
```

```
case 'LEFT'; KC=[KC 37];ACT=[ACT 3];
case 'UP'; KC=[KC 38];ACT=[ACT 3];
case 'RIGHT'; KC=[KC 39];ACT=[ACT 3];
case 'DOWN'; KC=[KC 40];ACT=[ACT 3];
case 'PRTSC'; KC=[KC 44];ACT=[ACT 3];
case 'INS'; KC=[KC 45];ACT=[ACT 3];
case 'DEL'; KC=[KC 46];ACT=[ACT 3];
case 'F1'; KC=[KC 112];ACT=[ACT 3];
case 'F2'; KC=[KC 113];ACT=[ACT 3];
case 'F3'; KC=[KC 114];ACT=[ACT 3];
case 'F4'; KC=[KC 115];ACT=[ACT 3];
case 'F5'; KC=[KC 116];ACT=[ACT 3];
case 'F6'; KC=[KC 117];ACT=[ACT 3];
case 'F7'; KC=[KC 118];ACT=[ACT 3];
case 'F8'; KC=[KC 119];ACT=[ACT 3];
case 'F9'; KC=[KC 120];ACT=[ACT 3];
case 'F10'; KC=[KC 121];ACT=[ACT 3];
case 'F11'; KC=[KC 122];ACT=[ACT 3];
case 'F12'; KC=[KC 123];ACT=[ACT 3];
case 'F13'; KC=[KC 124];ACT=[ACT 3];
case 'F14'; KC=[KC 125];ACT=[ACT 3];
case 'F15'; KC=[KC 126];ACT=[ACT 3];
case 'F16'; KC=[KC 127];ACT=[ACT 3];
case 'NUMLOCK'; KC=[KC 144];ACT=[ACT 3];
case 'SCROLLOCK'; KC=[KC 145];ACT=[ACT 3];
case '(';KC=[KC 16 57 16];ACT=[ACT 1 3 2];
case ')';KC=[KC 16 48 16];ACT=[ACT 1 3 2];
case '%'; KC=[KC 16 53 16];ACT=[ACT 1 3 2];
case '^'; KC=[KC 16 54 16];ACT=[ACT 1 3 2];
case '+'; KC=[KC 107];ACT=[ACT 3];
case 'ACCOLLEFT'; KC=[KC 16 219 16];ACT=[ACT 1 3 2];
case 'ACCOLRIGHT'; KC=[KC 16 221 16];ACT=[ACT 1 3 2];
case '';
otherwise [KC,ACT]=str2kc(str(k1+1:k2-1));
%error(['Unknown special key between {}, at
location ' int2str(k1) ' - ' int2str(k2)])
end
k1=k2;
%
%
================================================================================
========
%
case '('
k2=k1+1;
t2=str(k2);
level=1;
while t2~=')'|level~=0;
k2=k2+1;
t2=str(k2);
switch t2
case '(';level=level+1;
case ')';level=level-1;
end
end
```

```
k3=k2-1;
t3=str(k3);
while t3~=' '
k3=k3-1;
t3=str(k3);
end
N=str2num(str(k3+1:k2-1));
[KC1,ACT1]=str2kc(str(k1+1:k3-1));
KC=repmat(KC1,1,N);
ACT=repmat(ACT1,1,N);
k1=k2;
%
%
=========================================================================================
=======
%
case '+'
if str(k1+1)=='('
k2=k1+2;
t2=str(k2);
level=1;
if t2=='(';level=2;end
while t2~=')'|level~=0
k2=k2+1;
t2=str(k2);
switch t2
case '(';level=level+1;
case ')';level=level-1;
end
end
[KC2,ACT2]=str2kc(str(k1+2:k2-1));
KC=[KC 16 KC2 16];
ACT=[ACT 1 ACT2 2];
k1=k2;
else
KC=[KC 16];ACT=[ACT 3];
end
%
%
=========================================================================================
=======
%
case '^'
if str(k1+1)=='('
k2=k1+2;
t2=str(k2);
level=1;
if t2=='(';level=2;end
while t2~=')'|level~=0
k2=k2+1;
t2=str(k2);
switch t2
case '(';level=level+1;
case ')';level=level-1;
end
```

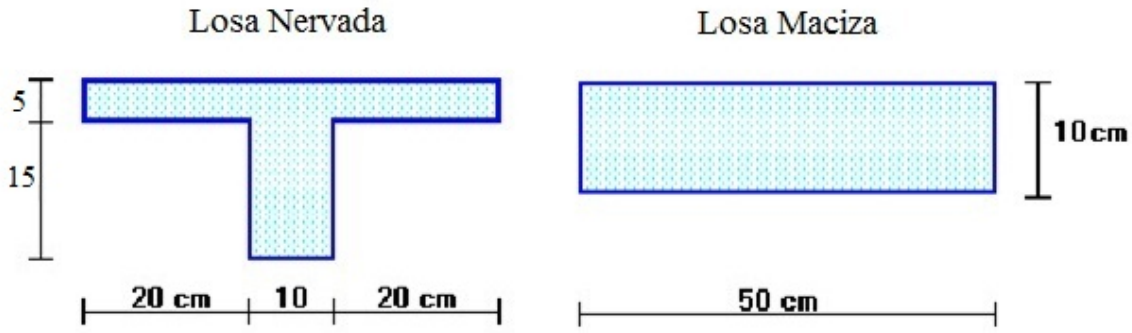
```
end
[KC2,ACT2]=str2kc(str(k1+2:k2-1));
KC=[KC 17 KC2 17];
ACT=[ACT 1 3*ones(1,length(KC2)) 2];
k1=k2;
else
KC=[KC 17];ACT=[ACT 3];
end
%
%
=========================================================================
========
%
case '%'
if str(k1+1)=='('
k2=k1+2;
t2=str(k2);
level=1;
if t2==')';level=2;end
while t2~=')'|level~=0
k2=k2+1;
t2=str(k2);
switch t2
case '(';level=level+1;
case ')';level=level-1;
end
end
[KC2,ACT2]=str2kc(str(k1+2:k2-1));
KC=[KC 18 KC2 18];
ACT=[ACT 1 3*ones(1,length(KC2)) 2];
k1=k2;
else
KC=[KC 18];ACT=[ACT 3];
end
%
%
=========================================================================
========
%
otherwise % 0-9, a-z of leestekens
if double(t1)>=48 & double(t1) <=57 %0-9
KC=[KC double(t1)+48];
ACT=[ACT 3];
elseif double(t1)>=97&double(t1)<=122 %a-z
KC=[KC double(t1)-32];
ACT=[ACT 3];
elseif double(t1)>=65&double(t1)<=90
KC=[KC 16 double(t1) 16];
ACT=[ACT 1 3 2];
else
switch t1
case '`'; KC=[KC 192];ACT=[ACT 3];
case '~'; KC=[KC 16 192 16];ACT=[ACT 1 3 2];
case '!'; KC=[KC 16 49 16];ACT=[ACT 1 3 2];
case '@'; KC=[KC 16 50 16];ACT=[ACT 1 3 2];
```

```
case '#'; KC=[KC 16 51 16];ACT=[ACT 1 3 2];
case '$'; KC=[KC 16 52 16];ACT=[ACT 1 3 2];
case '&'; KC=[KC 16 55 16];ACT=[ACT 1 3 2];
case '*'; KC=[KC 106];ACT=[ACT 3];
case '-'; KC=[KC 109];ACT=[ACT 3];
case '_'; KC=[KC 16 189 16];ACT=[ACT 1 3 2];
case '='; KC=[KC 187];ACT=[ACT 3];
case '['; KC=[KC 219];ACT=[ACT 3];
case ']'; KC=[KC 221];ACT=[ACT 3];
case ';'; KC=[KC 186];ACT=[ACT 3];
case ':'; KC=[KC 16 186 16];ACT=[ACT 1 3 2];
case '"'; KC=[KC 222];ACT=[ACT 3];
case '"'; KC=[KC 16 222 16];ACT=[ACT 1 3 2];
case '\'; KC=[KC 220];ACT=[ACT 3];
case '|'; KC=[KC 220];ACT=[ACT 1 3 2];
case ','; KC=[KC 188];ACT=[ACT 3];
case '<'; KC=[KC 16 188 16];ACT=[ACT 1 3 2];
case '.'; KC=[KC 190];ACT=[ACT 3];
case '>'; KC=[KC 16 190 16];ACT=[ACT 1 3 2];
case '/'; KC=[KC 191];ACT=[ACT 3];
case '?'; KC=[KC 16 191 16];ACT=[ACT 1 3 2];
case ' ';
otherwise error(['Unknown character at position '
int2str(k1) ' in ' str])
end
end
end
end
elseif iscell(inp)
for i=1:size(inp,1);
str=inp{i};
[KC2,ACT2]=str2kc(str);
KC=[KC KC2];
ACT=[ACT ACT2];
end
else
error('Input should be a string or a cell array')
end
```

ANEXO 7: Momento de inercia para una losa nervada

ALTERNATIVA UNO:

Se calcula el momento de inercia de la losa nervada como una viga T (con 1 solo nervio).



- Losa Maciza I_m= Momento de inercia losa maciza = $bh^3/12 = 50*10^3/12 = 4166.67\ cm^4$

- Losa Nervada I_n= Momento de inercia losa nervada

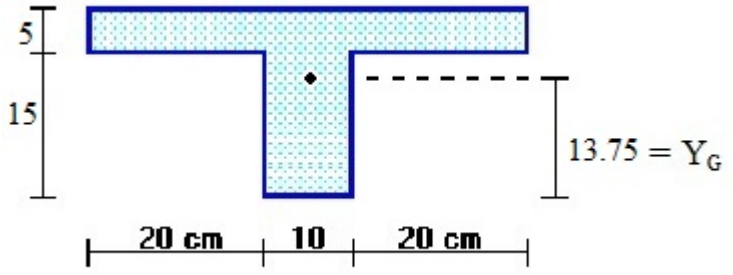
  A = Área de losa nervada = 5*50 + 15*10 = 400.00 cm2

  Se calcula el momento que produce la viga T con respecto a su base:

  M = [(5*50)* 17.5] + [(15*10)*7.5] = 5500.00 cm3

  Se calcula la posición del centro de gravedad de la viga T con relación a la base del alma:

  $Y_G = M/A = 5500/400 = 13.75$ cm



Se calcula la inercia de la viga **T** con relación a su centro de gravedad:

$I_n= 50*5^3/12+50*5*(17.5-13.75)^2+10*15^3/12+10*15*(13.75-7.5)^2$

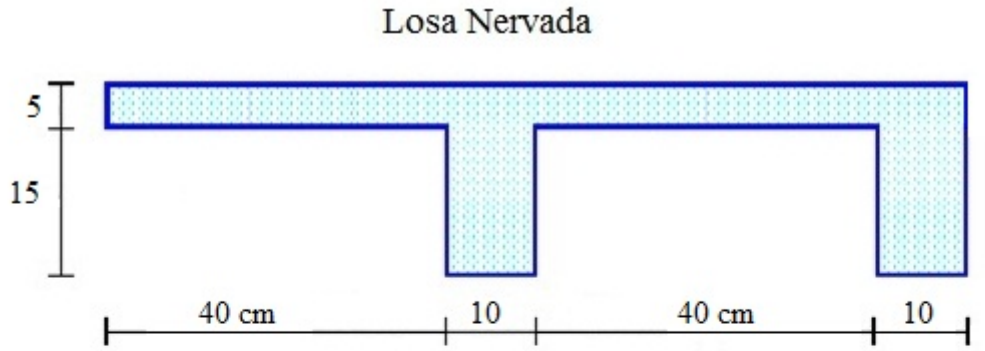
I_n= 520.8333 + 3515.625 + 2812.50 + 5859.375

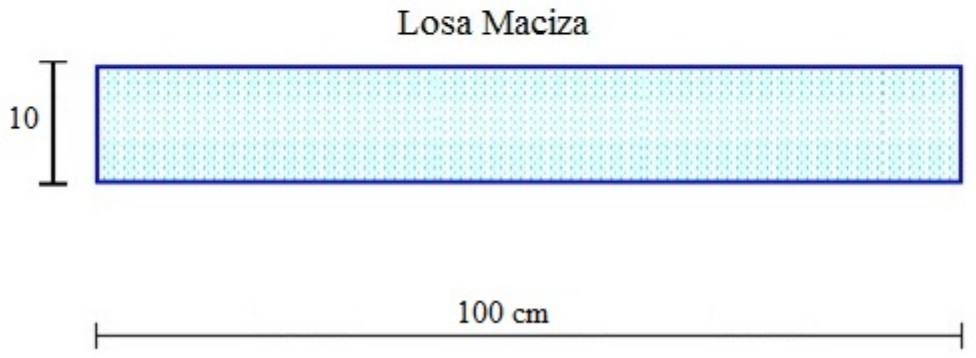
I_n= 12708.33 cm^4

- Factor de Corrección — F_{CI}=Factor de corrección de inercia = I_n/ I_m = 12708.33/4166.67 =3.05

ALTERNATIVA DOS:

Se calcula el momento de inercia de la losa nervada considerando un metro de losa (con 2 nervios).





- Losa Maciza — I_m= Momento de inercia losa maciza = $bh^3/12 = 100*10^3/12 = 8333.33\ cm^4$

- Losa Nervada — I_n= Momento de inercia losa nervada

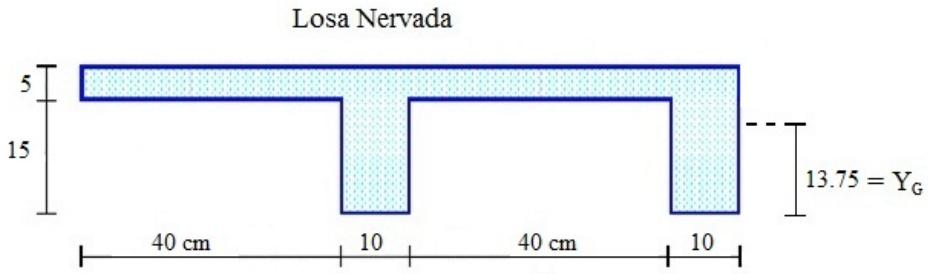
A = Área de losa nervada = 5*100 + 15*10 + 15*10 = 800.00 cm2

Se calcula el momento que produce la losa nervada con respecto a su base:

$M = [(5*100)* 17.5] + [(15*10)*7.5] + [(15*10)*7.5] = 11000.00 \text{ cm}3$

Se calcula la posición del centro de gravedad de la losa nervada con relación a la base del alma:

$Y_G = M/A = 7625/450 = 13.75 \text{ cm}$



Se calcula la inercia de la losa nervada con relación a su centro de gravedad:

$I_n = 100*5^3/12+100*5*(17.5-13.75)^2+2*[10*15^3/12+10*15*(13.75-7.5)^2]$

$I_n = 1041.67 + 7031.25 + 17343.75$

$I_n = 25416.67 \text{ cm}^4$

- Factor de Corrección

F_{CI}=Factor de corrección de inercia = $I_n / I_m = 25416.67/8333.33 = 3.05$

El dato que se ingresará en el modelo del SAP2000 para representar esta losa nervada será como un elemento Shell de espesor de 0.10 m y de peso volumétrico de 2400 kg/m^3 que es aproximadamente el peso de la losa si se consideraría maciza y para lo que es la inercia, hay que corregir la misma por un factor de 3.05 para que la losa maciza tenga la misma inercia que la losa nervada propuesta.