UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de programa de cómputo para el cálculo de deflexiones en elementos estructurales confinados

TESIS

Que para obtener el título de

Ingeniero Geomático

P R E S E N T A

Juan Pablo González Muñoz

DIRECTOR(A) DE TESIS

Ing. Roberto de la Cruz Sánchez

Ciudad Universitaria, Cd. Mx., 2023

Agradecimientos

A mi familia por su apoyo incondicional, su confianza y cariño; César y Ángeles por su enorme esfuerzo y sacrificio desde siempre, José por sus consejos y momentos compartidos. A mis tíos Checo y Silvia que siempre han demostrado su enorme cariño, a Coya y Chelo por compartir su tiempo y enseñanzas, así como a mis abuelos José, Alicia, Eliseo y Paula.

Agradezco al Ing. Joaquín Perea por su enorme paciencia, su franqueza, su sabiduría y sus consejos, al igual que su familia por su confianza.

A Fernanda, Tania, Ana Paula, Yolo, primos y sobrinos, Luis, Max, Emilio, Aldo, Carlita, Jessika, Anita, Karla, Deniss, Marifer, Alex, Yayo, Alitzel García, Alitzel Mejía, Anuar, Chucho, Pepe, Carlitos, Brayan, Marbel, Dalia y Omarcito por su compañía, amistad, sus ánimos y los momentos que hemos compartido.

También quiero agradecer a Iván Torres por su ayuda en la elaboración de este trabajo y del Ing. Ruvalcaba por sus asesorías. Al Ing. Roberto de la Cruz por su ayuda y sus conocimientos durante mi estancia en la universidad.

1 INTRODUCCIÓN

El sector de la construcción es la cuarta actividad económica generadora de empleos y riqueza del país contribuyendo con un 7% al PIB de acuerdo con datos del CEESCO [1] y además se relaciona con 183 ramas productivas de 262 que existen en México.

El desarrollo de la infraestructura nacional se torna más importante ya que proporciona elementos que permiten mejorar la calidad de vida de la sociedad mexicana como son vías terrestres, hospitales, escuelas, viviendas y obras civiles.

Actualmente, en el país existe una carencia de documentos y protocolos en normas y reglamentos que permitan establecer una respuesta y su actuar ante los procesos de desgaste en las edificaciones en cualquiera de sus etapas. Por ello, es importante que se considere implementar nuevas y actualizadas normas y protocolos que permitan seguir resguardando la seguridad de los usuarios y seguir mejorando el desarrollo de la infraestructura en el país.

1.1 Antecedentes

En un proyecto de una obra de infraestructura se contempla que tiene un ciclo de vida que se divide en cuatro etapas, de acuerdo con Antonio García Martínez (2010). Dichas etapas tienen un orden cronológico que consideran desde la extracción de materias primas hasta el mantenimiento y fin de la vida útil del edificio, las tres fases que se mencionan son: producción, diseño y proceso constructivo, uso y mantenimiento, y fase final de vida.

La producción se entiende como una serie de cadenas de procesos que incluye la extracción y suministro de materias primas, su transporte y su fabricación. (García Martínez, 2010)

El diseño y proceso de constructivo consta de todo el conjunto de planos, documentos, cálculos y especificaciones que se indica para el desarrollo constructivo con el fin de optimizar precios y tiempos. También es la fase donde se realiza la materialización del proyecto, es decir, se lleva a cabo la coordinación y el cronograma de actividades para el desarrollo de la obra.

La tercera etapa de una edificación es su uso y mantenimiento donde se consideran los procesos desarrollados durante la vida útil de la obra como el uso de ocupantes, reparaciones, renovaciones, etc. (García Martínez, 2010).

La última etapa comprende todas las operaciones relacionadas con la deconstrucción del edificio, desde el final de su vida útil a su eliminación final. (García Martínez, 2010).

[1] CEESCO: Centro de Estudios Económicos del Sector de la Construcción

Durante las etapas del ciclo de vida de una obra puede ocurrir algún evento o siniestro que pueda vulnerar la estabilidad o poner en riesgo su estructura. Estos siniestros pueden ser naturales tales como sismos, hundimientos, lluvias o también por acciones del hombre.

Actualmente existe una metodología llamada Modelado de Información de Construcción (BIM por sus siglas en inglés) que se emplea como una herramienta que logra gestionar y administrar documentación e información de alguna edificación durante las etapas de esta. En general, BIM es modelo inteligente que integra toda la información generando una representación gráfica de la edificación.

El ciclo de una edificación en BIM se divide en 7 dimensiones:

- 1D o Concepto. Se plantea la idea, diseño y bases del proyecto con estimaciones aproximadas (BIMnD, 2022).
- 2D o Boceto. Se establecen flujos de trabajo y procedimientos (BIMnD, 2022).
- 3D o Modelado. Se aplican los requisitos espaciales de la arquitectura, estructura e instalaciones del proyecto (BIMnD, 2022).
- 4D o Planificación. Se relaciona la variable del tiempo en la planificación del proyecto (BIMnD, 2022).
- 5D o Medición y presupuesto. Se emplea para el control de los costos de materiales, operaciones y mantenimiento de proyecto (BIMnD, 2022).
- 6D o Certificación energética. Ayuda para el cálculo y análisis energéticos (BIMnD, 2022).
- 7D o Mantenimiento. Documenta reparaciones e inspecciones para mantener la calidad del proyecto (BIMnD, 2022).

En la tecnología de BIM existen diferentes niveles de detalle acerca de la información de los elementos tanto de su geometría como sus características, también llamados niveles de desarrollo o LOD por sus siglas en inglés. En la actualidad existen siete niveles de desarrollo en la tecnología BIM, los cuales son:

- LOD 100. El nivel básico que representa la información de forma genérica y se definen parámetros como área, volumen, ubicación, etc. (EspacioBIM, 2020).
- LOD 200. Se adjunta información básica no geométrica a los elementos (EspacioBIM, 2020).
- LOD 300. Es un nivel detallado de las características de los elementos (EspacioBIM, 2020).
- LOD 350. Se documenta la construcción de los elementos, así como su interacción con varios sistemas (EspacioBIM, 2020).
- LOD 400. Se detalla la fabricación y montaje de los elementos, además de sus características más precisas como tamaños y cantidades e información no geométrica (EspacioBIM, 2020).

- LOD 500. Los elementos se modelan tal cual cómo se construyó, su pertenencia a un sistema constructivo específico y características geométricas y no geométricas de forma precisa y detallada. También, se aplica la normatividad correspondiente (EspacioBIM, 2020).
- LOD 600. Permite asignar información no geométrica y detallada a un elemento sin detalle, además que permite analizar información de deterioro de la edificación y darle un mantenimiento en su vida útil (EspacioBIM, 2020).

1.2 Identificación de la problemática

En México las herramientas tecnológicas creadas para emplearse en el sector de la construcción, como son las metodologías BIM, aún hace falta ser explotadas en su totalidad para su aplicación directa en la gestión de información hasta la creación de nuevas herramientas, en diferentes niveles de desarrollo, que permitan conocer a fondo el comportamiento de todos los sistemas que conforman una edificación.

Por otro lado, la topografía aplicada en la construcción es indispensable para poder desarrollar alguna obra civil en cualquiera de sus etapas, pero muchas veces se limita única y erróneamente al trabajo de medición. Todos los datos obtenidos mediante la topografía deberían almacenarse en sistemas de información para entender y optimizar temas de costos, tiempos, mantenimiento y monitoreo en cualquiera de las etapas de una edificación.

A pesar de que la topografía toma un papel muy importante en el sector de la construcción, en normas y reglamentos relacionadas para el desarrollo de infraestructura se desconoce si exista la documentación de algún protocolo para la obtención de datos de deformaciones en edificaciones a través de levantamientos topográficos, que en la actualidad sus equipos son capaces de hacer mediciones de alta precisión que facilitan y obligan a adquirir información relevante para identificar desgastes en alguna estructura.

La falta de aplicación de herramientas tecnológicas, así como la escasez de documentación de protocolos para la detección de deformaciones y desgastes en edificaciones evitan un desarrollo completo en las etapas de una construcción y que, a largo plazo, ante cualquier siniestro, dificulte dar la mejor solución tanto legal o estructuralmente puesto que se puede omitir información al faltar un respaldo de ella.

1.3 Objetivo

Proponer la implementación e instrumentación de un protocolo que pueda aplicarse en cualquier etapa constructiva para la identificación de deformaciones en elementos estructurales de las edificaciones y desarrollar una herramienta tecnológica capaz de procesar, calcular y generar información a detalle de desgastes en las edificaciones de acuerdo con la normatividad mexicana.

2 FUNDAMENTOS TEÓRICOS

2.1 Geometría Analítica

2.1.1 La recta en el espacio

Una recta es un lugar geométrico de todos los puntos contenidos en el plano tales, que tomados dos puntos cualesquiera P(*x,y*) y Q(*x,y*) de la recta, el valor de su pendiente es siempre constante. (FI División de Ciencias Básicas, Coordinación de Matemáticas, 2011).

2.1.2 Vector director

Un vector director es un vector que da la dirección y orientación de una recta. De tal forma que teniendo dos puntos que pertenecen a esa recta se puede calcular de la siguiente manera:

Sean P(x_p,y_p,z_p) y Q(x_q,y_q,z_q) dos puntos que pertenece a una recta su vector director (ū) en dirección P→Q queda definido:

$$\bar{u} = \left(x_q - x_p\ , y_q - y_p\ , z_q - z_p\right) \ldots (1)$$

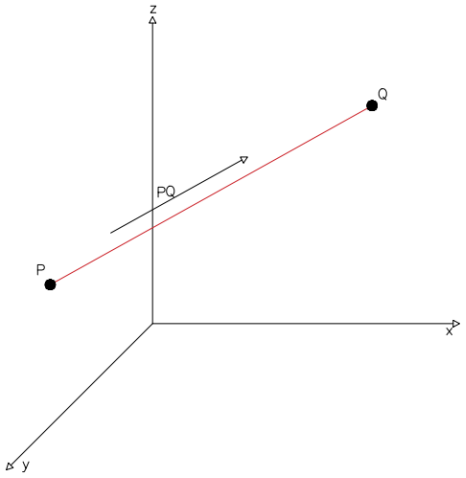


Imagen 1. Vector director $\bar{u}$

2.1.3 Ecuación vectorial y paramétrica de la recta

Sea P(x_0,y_0,z_0) un punto que pertenece a la recta ***r***; y sea ū = (u_x,u_y,u_z) un vector director de la misma recta ***r***, entonces obtenemos su ecuación vectorial de la recta:

$$(x, y, z) = (x_0, y_0, z_0) + \lambda \cdot \left(u_x, u_y, u_z\right) \ldots (2)$$

Siendo

(x, y, z), punto arbitrario de ***r;***

(x_0, y_0, z_0), coordenadas de un punto que pertenece a ***r;***

λ, parámetro perteneciente a los números reales;

(u_x, u_y, u_z), componentes del vector director de ***r.***

Igualando las componentes de la ecuación (2) obtendremos la ecuación paramétrica de la recta:

$$\begin{cases} x = x_0 + \lambda \cdot u_x \\ y = y_0 + \lambda \cdot u_y \\ z = z_0 + \lambda \cdot u_z \end{cases} \dots (3)$$

2.1.4 Distancia entre dos puntos en un espacio tridimensional

La distancia entre los puntos P(x_1,y_1,z_1) y Q(x_2,y_2,z_2) se calcula de la forma:

$$d(P, Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \dots (4)$$

2.1.5 Ecuación de un plano

Un plano esta determinad por al menos tres puntos no colineales o por un punto y dos vectores.

Sean los puntos P(x_p,y_p,z_p), Q(x_q,y_q,z_q) y R(x_r,y_r,z_r), y siendo que los vectores directores PQ y QR podemos generar la ecuación vectorial del plano π como,

$$\pi : (\mathrm{x}, \mathrm{y}, \mathrm{z}) = P + t\overline{PQ} + s\overline{QR};\ \ t, s \in \mathbb{R} \dots (5)$$

2.1.6 Ecuación normal y cartesiana del plano

Conociendo un punto P(x_p,y_p,z_p) que pertenece al plano π y un vector normal **n** al plano. Entonces, un punto *(x,y,z)* pertenece al plano si cumple con

$$n \cdot \big(P - (x, y, z)\big) = 0 \dots (6)$$

O también,

$$n \cdot (x, y, z) = n \cdot P \ \dots (7)$$

Siendo (7) la ecuación normal del plano. Si se desarrolla la ecuación (7) se obtiene la ecuación cartesiana del plano. Si **n** = *(a,b,c)* entonces,

$$(a, b, c) \cdot (x, y, z) = 0$$

$$ax + by + cz = d \ con \ d = n \cdot P$$

$$ax + by + cz + d = \ 0 \dots (8)$$

Siendo (8) la ecuación cartesiana del plano.

2.1.7 Ecuación del plano conociendo un punto y una recta

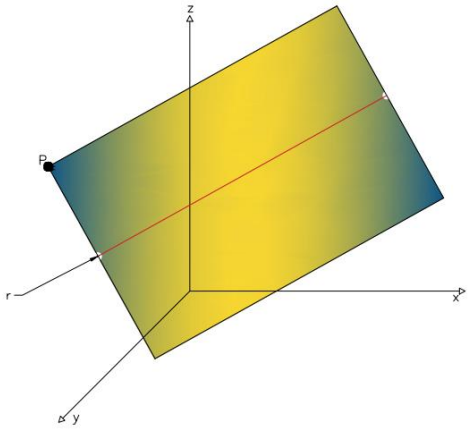


Imagen 2. Plano con recta r y punto P

Sea un punto P(x_p,y_p,z_p) no colineal con la recta ***r*** de ecuación paramétrica

$$r = \begin{cases} x = x_0 + \lambda \cdot u_x \\ y = y_0 + \lambda \cdot u_y \\ z = z_0 + \lambda \cdot u_z \end{cases}$$

Con su vector director ū = (u_x,u_y,u_z), que pertenecen al plano π. La ecuación cartesiana de π se obtiene de la siguiente forma:

Siendo $\lambda = 0$ en ***r*** entonces

$$R(x_0, y_0, z_0) \in r$$

Se obtiene el vector PR,

$$\bar{v} = (x_o - x_p, y_0 - y_p, z_0 - z_p) = (v_x, v_y, v_z)$$

Se lleva a cabo el producto cruz de $\bar{v} \times \bar{u}$ para encontrar el vector normal,

$$\bar{v} \times \bar{u} = \begin{vmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ v_x & v_y & v_z \\ u_x & u_y & u_z \end{vmatrix} = (v_y u_z - v_z u_y)\hat{\imath} + (v_x u_z - v_z u_x)\hat{\jmath} + (v_x u_y - v_y u_x)\hat{k}$$

Simplificando,

$$\bar{n} = \bar{v} \times \bar{u} = (A, B, C)$$

De acuerdo con la ecuación normal del plano (7) se tiene que,

$$RP \cdot \bar{n} = (x_0, y_0, z_0)(x, y, z) \cdot (A, B, C) = 0$$

Siendo P un punto cualquiera del plano y desarrollando:

$$RP \cdot \bar{n} = (x - x_0, y - y_0, z - z_0) \cdot (A, B, C) = 0$$

$$(x - x_0)A + (y - y_0)B + (z - z_0)C = 0$$

$$Ax + By + Cz + (-Ax_0 - By_0 - Cz_0) = 0$$

$$Ax + By + Cz + D = 0 \ldots (9)$$

Siendo (9) la ecuación cartesiana del plano π a partir de un punto conocido y una recta que contiene el plano.

2.1.8 Distancia de un punto a un plano

La distancia más corta entre un punto y un plano es la que se mide perpendicularmente del punto al plano.

Siendo P(x_p,y_p,z_p) un punto no perteneciente al plano π con ecuación cartesiana $Ax + By + Cz + D = 0$, entonces la distancia de P a π será:

$$d(P, \pi) = \frac{|Ax_p + By_p + Cz_p + D|}{\sqrt{A^2 + B^2 + C^2}} \ldots (10)$$

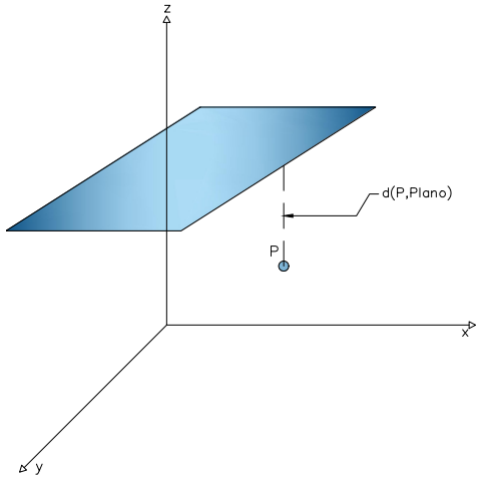


Imagen 3. Distancia de un punto a un plano

2.2 Cálculo de coordenadas a partir de un ángulo y una distancia

En la topografía, los levantamientos realizados se representan en un plano que estará referido a un sistema de ejes cartesianos, que generalmente, el eje Y sigue la dirección de la meridiana (Norte-Sur) y el eje X la dirección perpendicular a la meridiana (Este-Oeste). (García, 2014)

Por otro lado, para proyectar un punto en un plano XY se puede hacer mediante sus coordenadas polares. En la topografía es común combinar las coordenadas polares y cartesianes para obtener un plano.

2.2.1 Coordenadas Polares

Son un sistema de coordenadas bidimensional en donde cada punto del plano se determina a partir de un ángulo y una distancia. Un instrumento topográfico como la estación total, permite calcular coordenadas cartesianas a partir de la medición de coordenadas polares y de distancias reducidas.

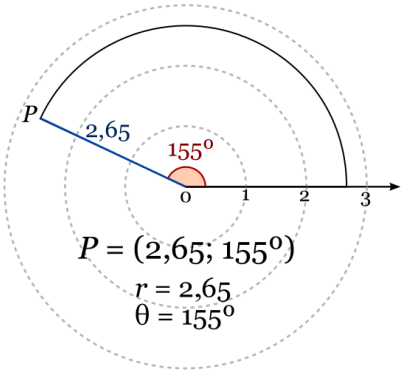


Imagen 4. Coordenadas Polares. Sánchez, Pedro (2009). Ejemplo de coordenadas polares. Recuperado de: https://commons.wikimedia.org/wiki/File:Coordenadas_polares.svg#metadata

2.2.2 Acimut

Es el ángulo que forma una línea con la dirección Norte-Sur o alguna otra dirección de referencia, medido de 0° a 360° y en sentido dextrógiro (sentido de las agujas del reloj). (Montes de Oca, 1989)

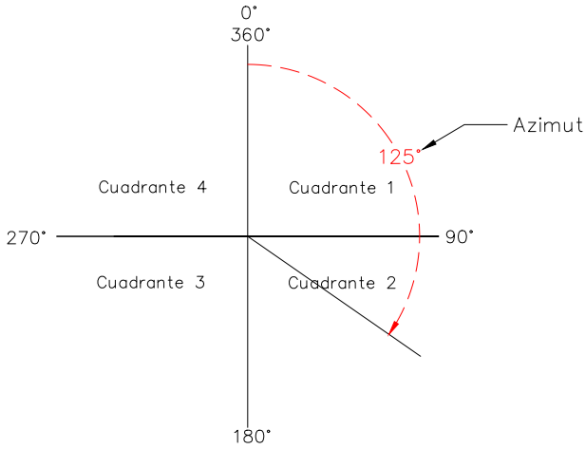


Imagen 5. Ejemplo de azimut en un plano

2.2.3 Distancia reducida

La distancia reducida hace referencia al cálculo de la distancia entre dos puntos con coordenadas P(x_1,y_1) y Q(x_2,y_2) en un plano horizontal, es decir:

$$distancia\ (P,Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \ldots (11)$$

2.2.4 Coordenadas Cartesianas

El sistema de coordenadas cartesianas consiste en dos ejes perpendiculares. Dichos ejes se cortan en un punto que será el origen de nuestro sistema de coordenadas.

Existen coordenadas absolutas o totales las cuales se conoce el origen del sistema, en cambio, existen las coordenadas relativas o parciales que se refieren a otro punto distintos del origen de coordenadas. (García, 2014)

2.3 Transformación de Coordenadas

2.3.1 Transformación de coordenadas polares a coordenadas cartesianas

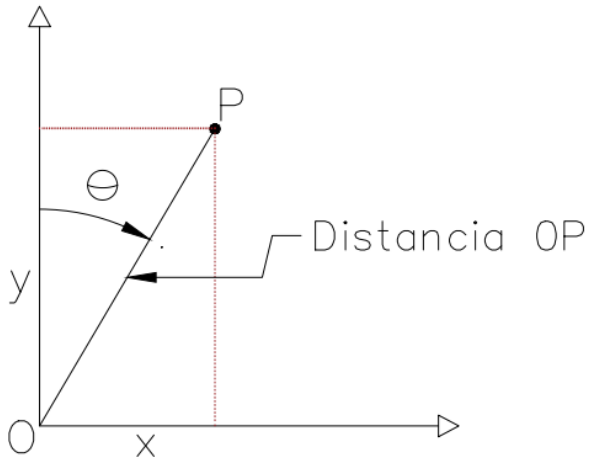


Imagen 6. Proyección para el cálculo de las coordenadas de P mediante coordenadas polares

Al tener las coordenadas polares de un punto, la distancia (*d)* y un acimut (*Θ)*, podemos calcular coordenadas cartesianas tanto absolutas como relativas. De acuerdo con la imagen 6, se puede deducir que las coordenadas del punto P son:

$$X_p = d_{OP} * sen(\Theta) \dots (12)$$

$$Y_p = d_{OP} * cos(\Theta) \dots (13)$$

2.3.2 Transformación de coordenadas cartesianas a coordenadas polares

Para obtener la distancia se realiza de forma análoga a la ecuación 11, al tomar los valres de X y Y de las coordenadas cartesianas.

Al tener un par coordenadas cartesianas, para el cálculo del acimut se emplea la siguiente expresión:

$$\Theta = ang\tan\frac{x_2 - x_1}{y_2 - y_1} = \frac{\Delta x}{\Delta y} \dots (14)$$

Existen en total 4 casos que dependen del cuadrante en el que esté ubicado el valor angular del acimut.

Cuadrante 1.

El ángulo del azimut se encuentra en el primer cuadrante si $\Delta x < 0$ y $\Delta y < 0$, entonces se aplica la ecuación 14:

$$\Theta = ang \tan \frac{x_2 - x_1}{y_2 - y_1} \ (15)$$

Cuadrante 2 y cuadrante 3.

El ángulo del azimut se encuentra en el segundo cuadrante si Δx > 0 y Δy < 0, entonces se aplica:

$$\Theta = ang \tan \frac{x_2 - x_1}{y_2 - y_1} + 180 \ldots (16)$$

Cuadrante 4.

El ángulo del azimut se encuentra en el cuarto cuadrante si Δx < 0 y Δy > 0, entonces se aplica:

$$\Theta = 360 + ang \tan \frac{x_2 - x_1}{y_2 - y_1} \ldots (17)$$

2.4 Análisis Estructural

En general, la palabra estructura hace referencia a un sistema o conjunto en donde sus elementos están distribuidos y en orden. (Cervera Ruiz & Blanco Díaz, 2014)

Al hablar particularmente en materia de mecánica estructural, una estructura es un conjunto estable de elementos resistentes capaz de mantener sus formas y cualidades con el paso del tiempo, bajo la acción de distintas fuerzas y cargas exteriores. (Carlos & José, 2009)

Cervera Ruiz y Blanco Diaz (2014) señalan que para el diseño de estructuras debe existir un análisis que permita determinar la viabilidad del comportamiento mecánico de estas mismas, mediante distintos métodos y técnicas, considerando al menos tres etapas a valorar:

- *Proyecto*, donde se valida y evalúa alternativas de materiales, dimensiones, etc.
- *Construcción*, donde se valida y evalúa diferentes metodologías de construcción.
- *Vida útil*, donde se aseguran las condiciones de funcionalidad y mantenimiento de la estructura.

El objetivo del análisis estructural consiste en determinar los estados de tensión y deformación cuando la estructura está sometida a acciones o estados de carga, en otras palabras, el análisis estructural consta de calcular las fuerzas internas y las deflexiones en un punto cualquiera de una estructura. (Camba Castañeda, Chacón García, & Pérez Arellano, 1982)

2.4.1 Condiciones del Análisis Estructural

El análisis estructural debe cumplir con tres condiciones:

A. *Equilibrio entre fuerzas internas y externas en todos los elementos estructurales*

Una estructura estará en equilibrio cuando el sistema de fuerzas externas en el que está sometido cumple con:

$$\sum F_x = 0 \qquad \sum M_x = 0$$
$$\sum F_y = 0 \qquad \sum M_y = 0$$
$$\sum F_z = 0 \qquad \sum M_z = 0$$

Siendo F la suma de sus fuerzas resultantes en un mismo plano y M el momento de una fuerza capas de alterar la velocidad de giro de un cuerpo. (L. Fernández, 2023)

Si la estructura está en equilibrio, cualquier elemento que se aísle también lo estará con la condición de que se apliquen las ecuaciones del equilibrio considerando las fuerzas internas provocadas por el sistema de fuerzas externas.

B. Compatibilidad de deformaciones de los elementos estructurales

Una estructura sometida a fuerzas externas o acciones de carga tendrá deformaciones y desplazamientos que deben de considerarse para su análisis. Por eso, al igual que las condiciones de equilibrio también deben existir condiciones de compatibilidad para las deformaciones, siendo tres de acuerdo con Cervera Ruiz y Blanco Díaz (2014):

- *Condiciones de apoyo*, las deformaciones deben de cumplir con las limitaciones de movimiento del apoyo.

  Como se muestra en la imagen 7, debido al empotramiento en el apoyo A imposibilita desplazamientos lineales en sus tres componentes al igual que giros, lo contrario con B, que al ser un apoyo articulado imposibilita desplazamientos lineales, pero no giros.

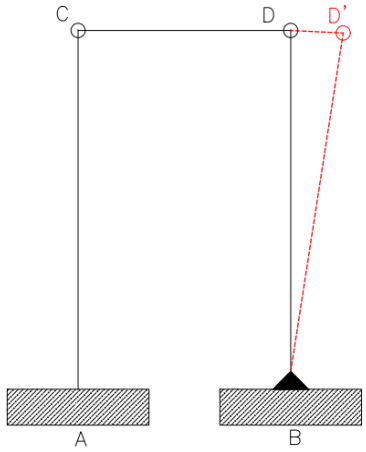


Imagen 7. Condición de apoyo

- *Continuidad en los nudos*, para los elementos que en sus extremos convergen en un nudo estructural su deformación debe cumplir con las limitaciones de movimiento del elemento enlace.

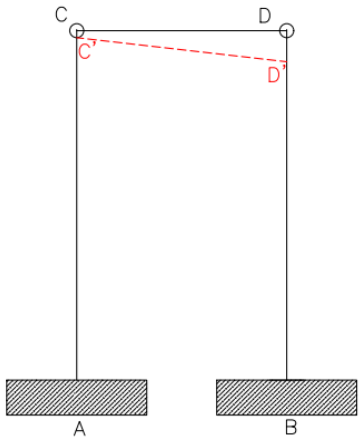


Imagen 8. Condición de continuidad en los nudos

- *Continuidad en las barras*, la deformación debe conservar la continuidad del elemento estructural evitando huecos.

C. Relación fuerza – desplazamiento

En general, la relación entre fuerzas y desplazamientos es lineal para poder aplicar el principio de superposición, es decir, que "los efectos que produce un sistema de fuerzas aplicado a una estructura son equivalentes a la suma de los efectos producidos por cada una de las fuerzas del sistema actuando independientemente". (Camba Castañeda, Chacón García, & Pérez Arellano, 1982)

Para aplicar dicho principio es necesario tomar en cuenta las siguientes hipótesis:

a) Las deformaciones (desplazamientos y giros) sean pequeñas que no afecten el sistema de fuerzas internas y de reacciones;
b) Que exista proporcionalidad entre esfuerzo y deformaciones, es decir, que cumpla en general la Ley de Hooke[2].
c) Que no exista una reducción de resistencia en los elementos estructurales debido a la relación de su longitud entre las dimensiones de su sección transversal sujetas a fuerzas axiales, llamado efecto de esbeltez. (Carrillo Cubillas, 2004)

[2] La *Ley de Hooke* establece que el alargamiento unitario que experimenta un cuerpo elástico es directamente proporcional a la fuerza aplicada sobre el mismo.

2.4.2 Elementos Estructurales

Los elementos que conforman una estructura se denominan de acuerdo con la función que ocupan, algunos de ellos son:

- Pilar / Soporte / Columna: Elemento vertical que resiste cargas paralelas a su eje de simetría, es decir, cargas axiales.

Imagen 9. Columna como elemento estructural

- Viga: Elemento horizontal sometido a cargas verticales y a esfuerzos de flexión.

Imagen 10. Viga de acero como elemento estructural

- Cable / Tirante: Elemento estructural sometido a tracción. (Cervera Ruiz & Blanco Díaz, 2014)

- Apoyo: Dispositivos destinados a la unión de la estructura al medio de sustentación. Sus funciones principales son el de impedir movimientos, limitar deformaciones y transmitir las cargas.

Imagen 11. Apoyo tipo empotramiento como elemento estructural

- Enlace / Nudo: Dispositivos destinados a unir entre sí los elementos de la estructura. Sus funciones principales son impedir movimientos relativos de los elementos y trasmitir las cargas.

Imagen 12. Nudos rígidos como elemento estructural

2.4.3 Acciones de carga sobre las estructuras

Sobre una estructura pueden actuar cargas de diferente naturaleza, por ello es importante clasificarlas para identificarlas y considerarlas para el análisis estructural. Para Cervera Ruiz y Blanco Díaz (2014) se pueden clasificar como:

- *Acciones permanentes*: cargas que actúan en todo momento y son constantes en posición y magnitud, que a la vez se clasifican en:

    - Cargas muertas, se consideran todos los pesos de los elementos constructivos, de los acabados y elementos que ocupan una posición permanente y tienen un peso que no cambia sustancialmente con el tiempo. (NTC para Diseño por Sismo, 2017)
    - Peso propio de los elementos estructurales.

- *Acciones permanentes de valor no constante,* como acciones reológicas (deformaciones en materia en estado líquido y sólido), etc.
- *Acciones variables*, cargas externas que pueden o no actuar sobre la estructura como:

    - Cargas vivas: son las fuerzas que no tienen carácter permanente y que se producen por el uso y ocupación de la edificación. (NTC para Diseño por Sismo, 2017)
    - Acción del viento, que depende de factores como las dimensiones, geometría de la estructura, la dirección, intensidad, etc.
    - Cargas térmicas, donde las variaciones de temperaturas pueden generar modificación a los elementos estructurales.
    - Cargas de nieve, etc.

- *Acciones accidentales*, como pueden ser inundaciones, sismos, impactos de vehículos, etc.

2.5 Estado límite en estructuras

Se puede definir un estado de límite en una estructura cuando tiende a un comportamiento que impide el buen funcionamiento mecánico – estructural por el cual fue diseñado. De acuerdo con Las Normas Técnicas Complementarias de Criterios y Acciones para el Diseño Estructural de las Edificaciones de la CDMX (2017), “se alcanza un estado límite de comportamiento en una construcción cuando se presenta una combinación de fuerzas, desplazamientos, niveles de fatiga, o varios de ellos, que determina el inicio o la ocurrencia de un modo de comportamiento inaceptable de dicha construcción”.

Existen dos clasificaciones de estados límites: a) estados límites de falla, es cuando la estructura presenta una falla que afecta el funcionamiento de la estructura y es insegura; y b) estados límites de servicio, son aquellos que el comportamiento afecta el funcionamiento de la estructura, pero sigue siendo segura.

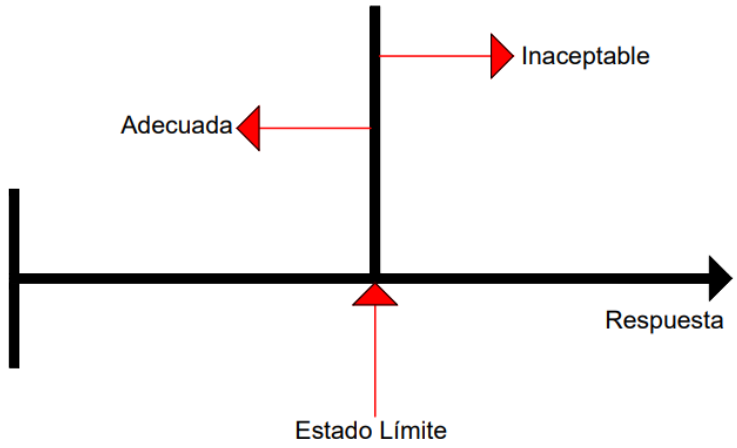


Imagen 13. Respuesta de una estructura (Figura tomada del libro: Meli, Roberto "Diseño Estructural" 2a. Edición (2009))

En el Artículo 147 del Reglamento de Construcciones para la CDMX (RCCDMX) del 2021 menciona que la estructura y cada uno de sus elementos debe cumplir:

I. Tener seguridad adecuada contra la aparición de todo estado límite de falla posible ante las combinaciones de acciones más desfavorables que puedan presentarse durante su vida esperada, y

II. No rebasar ningún estado límite de servicio ante combinaciones de acciones que corresponden a condiciones normales de operación.

Además, es importante para el diseño estructural contar con una serie de respuestas a acciones que rebasen los estados límites, mediante procedimientos y metodologías obtenidas del análisis estructural.

2.5.1 Estado límite de falla

En el Artículo 148 del RCCDMX se define al estado límite de falla como “cualquier situación que corresponda al agotamiento de la capacidad de carga de la estructura o de alguno de sus componentes, incluyendo la cimentación, o al hecho de que ocurran daños irreversibles que afecten significativamente su resistencia ante nuevas aplicaciones de carga”.

A la vez se pueden considerar dos tipos de estados límites de falla:

- Frágil, cuando se reduce la capacidad de carga bruscamente y los elementos estructurales no cuentan con deformaciones considerables;
- Dúctil, cuando los elementos estructurales pueden alcanzar deformaciones considerables antes de fallar y mantengan la capacidad de carga. (Cirilo Manzanares, 2013)

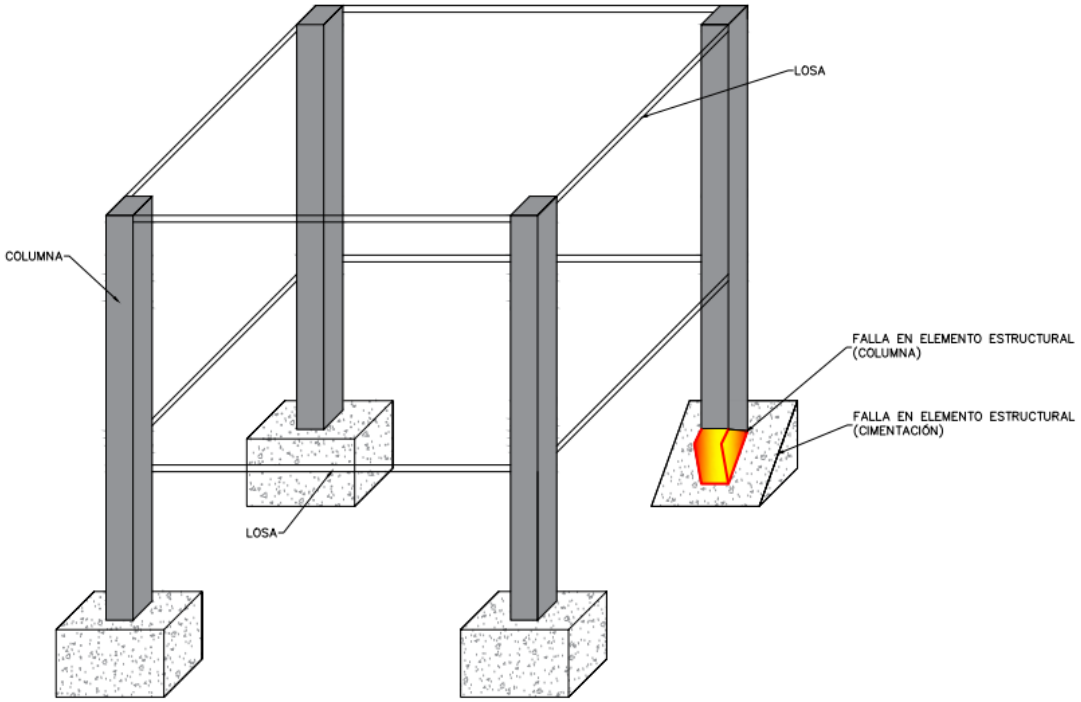


Imagen 14. Estado Límite de Falla en elementos estructurales

2.5.2 Estado límite de servicio

En el Artículo 149 del RCCDMX considera al estado límite de servicio como “la ocurrencia de desplazamientos, agrietamientos, vibraciones o daños que afecten el correcto funcionamiento de la edificación, pero que no perjudiquen su capacidad para soportar cargas”.

Si la estructura o alguno de sus elementos alcanza el estado límite de servicio puede generar reparaciones muy costosas o incluso el cierre de la construcción dependiendo de su uso y la percepción de los usuarios.

De acuerdo con las Normas Técnicas Complementarias (NTC) de Criterios y Acciones para el Diseño Estructural de las Edificaciones de la CDMX (2017) el cálculo del estado límite de servicio aplica para desplazamientos verticales y horizontales, así como para vibraciones.

Tabla 1.Estado límite y su respuesta estructural. Meli, Roberto "Diseño Estructural" 2a. Edición (2009)

Estado Límite de Falla	Estado Límite de Servicio
▪ ***Colapso***	▪ ***Desplazamientos verticales***
▪ ***Inestabilidad***	▪ ***Desplazamientos horizontales***
▪ ***Fatiga***	▪ ***Vibraciones***
▪ ***Daño irreversible***	▪ ***Agrietamientos***

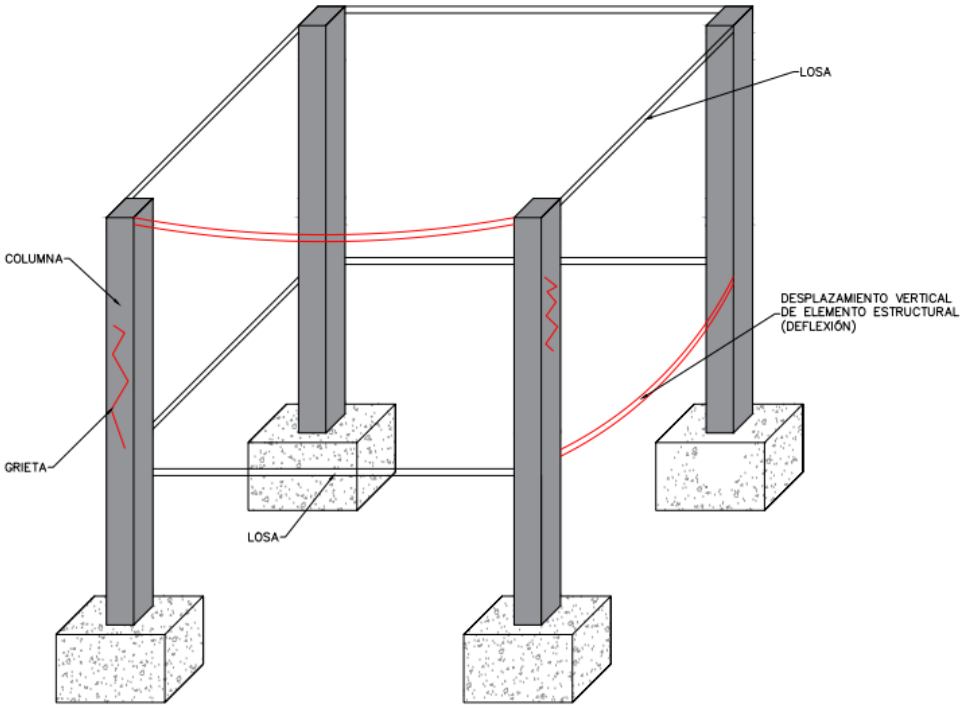


Imagen 15. Estado límite de servicio en elementos estructurales

2.5.3 Desplazamientos Verticales o Deflexiones

Cuando a un cuerpo o elemento se le aplica un sistema de fuerzas externas éste logrará deformarse hasta encontrar el equilibrio entre las fuerzas externas y las fuerzas internas, es decir, que el sistema de fuerzas externas realiza un trabajo el cual se almacena en el cuerpo y se denomina energía de deformación del cuerpo. Dicha energía de deformación aparece debido a fuerzas axiales, de flexión, cortantes, de torsión o combinadas. (Camba Castañeda, Chacón García, & Pérez Arellano, 1982)

La deflexión se define como el desplazamiento total de un punto de un elemento estructural de su posición antes de la aplicación de la carga a su posición después de someterlo a carga. (Armenta Mena, 1986)

Los elementos horizontales como vigas, trabes y losas sometidas a cargas son quienes presentan las deflexiones o también contraflechas. Dependiendo de las características físicas – químicas del material de estos elementos pueden llegar a ser muy flexibles al grado de tener desplazamientos verticales considerables y que afecten la funcionalidad de otros elementos no estructurales como muros, marcos de puertas y ventanas, etc. Por ello, es importante fijar sus valores de estado límite de servicio.

Cuando los desplazamientos verticales son exagerados se debe principalmente por las cargas gravitacionales vivas o muertas, efectos de temperatura, hundimientos diferenciales, errores en tolerancias y de cálculo estructural.

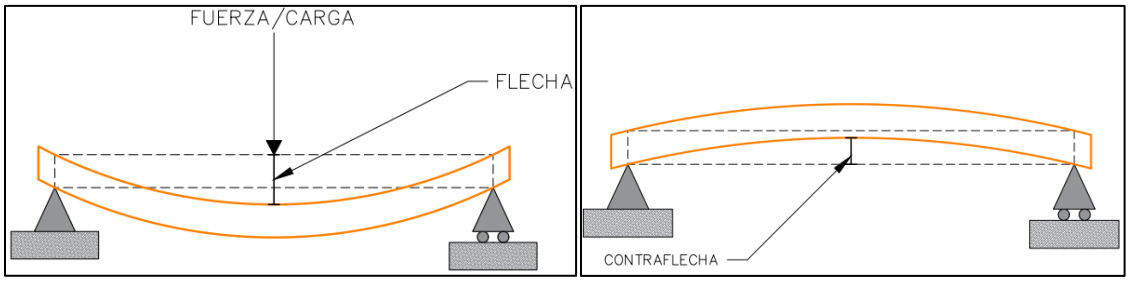


Imagen 16. Izquierda) Representación de flecha en viga; Derecha) Representación de contraflecha en viga

En el caso de las contraflechas, es común que se indique el grado de este desplazamiento desde el diseño estructural del elemento para fines estéticos o para que al aplicarle cargas tenga un ajuste en su superficie y exista un desplazamiento mínimo.

De acuerdo con la NTC de Criterios y Acciones para el Diseño Estructural de las Edificaciones de la CDMX (2017) para obtener los valores de los estado límite de servicio en desplazamientos verticales se aplica lo siguiente:

Desplazamiento vertical en trabes	Desplazamiento vertical en trabes voladizas
$\boldsymbol{ELS_v = \frac{l}{240}}$ *… (1)*	$\boldsymbol{ELS_v = \frac{l}{480}}$ *… (2)*

Donde:

ELS_v, estado límite de servicio en desplazamientos verticales;
l, longitud de la trabe o claro.

En caso de que estos desplazamientos verticales afecten a elementos no estructurales, el estado límite de servicio se calculará así:

Desplazamiento vertical en trabes	Desplazamiento vertical en trabes voladizas
$\boldsymbol{ELS_v = \frac{l}{480}}$ *… (3)*	$\boldsymbol{ELS_v = \frac{l}{960}}$ *… (4)*

De acuerdo a las Normas Técnicas Complementarias de la CDMX, estas ecuaciones se aplican a estructuras de acero y concreto.

Es importante mencionar que en estructuras de acero pueden existir contracciones y expansiones (cambios de dimensiones) por la temperatura, pero esto no debe afectar su funcionalidad ni su comportamiento en condiciones de servicio.

En caso de elementos estructurales de concreto no preforzados, para obtener su deflexión total se toman en cuenta dos tipos de deflexiones (inmediata y diferida) las cuales se calculan mediante momentos de inercia y módulos de elasticidad.

En el caso particular de elementos horizontales de madera los valores de estados límites de servicio para las deflexiones se obtienen de la siguiente manera.

En elementos de madera con claros menores a 3.5 m	En elementos de madera con claros mayores a 3.5 m
$\boldsymbol{ELS_v = \frac{l}{240}}$ *… (5)*	$\boldsymbol{ELS_v = \frac{l}{240} + 0.005\ m}$ *… (6)*

En caso de que estos desplazamientos verticales afecten a elementos no estructurales, el estado límite de servicio se calculará con las ecuaciones (7) y (8).

En elementos de madera con claros menores a 3.5 m

$$\boldsymbol{ELS_v = \frac{l}{480}} \quad \textit{... (7)}$$

En elementos de madera con claros mayores a 3.5 m

$$\boldsymbol{ELS_v = \frac{l}{480} + 0.003\ m} \quad \textit{... (8)}$$

Por otro lado, también existen normas internacionales que se consideran para el análisis estructural en los estado límite de servicio. A continuación, se presentan algunas de ellas.

De acuerdo con el Documento Básico SE Seguridad Estructural del Código Técnico de Edificación (2019) se menciona que el estado límite de servicio no debe exceder las siguientes expresiones

Pisos con tabiques frágiles o pavimentos rígidos sin juntas

$$\boldsymbol{ELS_v = \frac{l}{500}} \quad \textit{... (9)}$$

Pisos con tabiques ordinarios o pavimentos rígidos con juntas

$$\boldsymbol{ELS_v = \frac{l}{400}} \quad \textit{... (10)}$$

El resto de los casos

$$\boldsymbol{ELS_v = \frac{l}{300}} \quad \textit{... (11)}$$

En las normas norteamericanas, existen los Requisitos de Reglamento para Concreto Estructural (ACI 318S-14) del *American Concrete Institute* donde propone que el estado límite de servicio en trabes de concreto preforzadas no excedan las siguientes expresiones.

<table>
<tr><th>Miembro</th><th colspan="2">Condición</th><th>Deflexión considerada</th><th>Límite de deflexión</th></tr>
<tr><td>Cubiertas planas</td><td colspan="2" rowspan="2">Que no soporten ni estén ligados a elementos no estructurales susceptibles de sufrir daños debido a deflexiones grandes</td><td>Deflexión inmediata debida a cargas vivas, de nieve y lluvia</td><td>$\frac{l}{180}$</td></tr>
<tr><td>Entrepisos</td><td>Deflexión inmediata debida a carga viva</td><td>$\frac{l}{360}$</td></tr>
<tr><td rowspan="2">Cubiertas o entrepisos</td><td rowspan="2">Soporten o están ligados a elementos no estructurales</td><td>Susceptibles de sufrir daños debido a deflexiones grandes</td><td rowspan="2">La parte de la deflexión total que ocurre después de la unión de los elementos no estructurales</td><td>$\frac{l}{480}$</td></tr>
<tr><td>No susceptibles de sufrir daños debido a deflexiones grandes</td><td>$\frac{l}{240}$</td></tr>
</table>

Tabla 2. Deflexión máxima admisible calculada. ACI 318S-14

2.6 Topografía Pericial

Cuando se habla de peritaje se refiere a toda actividad de estudio realizada por una persona o un equipo de personas acreditadas por sus conocimientos y habilidades para la obtención de criterios certeros para fines de actividad procesal establecida por la ley. (Durán Mendietam, Martínez, & Nepomuceno Gutiérrez)

En el peritaje se involucran los siguientes elementos:

- Perito, persona especialista en alguna rama de ciencia, arte o técnica.
- El objeto que es investigado.
- La investigación o peritaje en donde se obtiene un informe siendo un documento técnico que sirve como prueba para realizar dictámenes ante un proceso judicial.
- La forma procesal en que se llevará a cabo el peritaje.

En México existe la oportunidad de realizar peritajes en materia de topografía, la persona que ejerza como topógrafo perito debe estar certificada, acreditada y pertenecer a un Padrón de Peritos Topógrafos Estatal.

La topografía pericial puede aplicarse en muchos aspectos como:

- Levantamientos topográficos y volumétricos
- Análisis de diseño y desarrollo de construcciones
- Evaluación de bienes
- Agrimensura y delimitación de terrenos
- Arqueología
- Diseño de vías de comunicación

La topografía pericial se vuelve necesaria cuando se requiere detectar o verificar características como ubicación, tamaño, alturas, distancias, deformaciones o accidentes geográficos en cualquier tipo de elemento y que son debidamente solicitados mediando procesos judiciales.

En un levantamiento topográfico, en este caso de carácter pericial, es recomendable realizar un informe que contenga la metodología empleada, las características técnicas del equipo de medición, evidencias fotográficas, los datos obtenidos, así como una conclusión sobre el estado del elemento medido.

2.6.1 Topografía pericial para la seguridad de edificaciones estructurales

Las construcciones tienen un cierto grado de vulnerabilidad de estabilidad en la estructura puesto que se ven amenazadas por fenómenos naturales como sismos, hundimientos, procesos de remoción de masas, lluvia o por acciones del hombre.

Es importante realizar revisiones y monitores de las edificaciones durante cualquiera de las etapas de construcción, principalmente posterior a un siniestro que pueda causar daños a la estructura. En estas condiciones nace la topografía pericial, para tener monitoreos de hundimientos y desplazamientos diferenciales, desplomes y verticalidad de columnas o deformaciones verticales y longitudinales en elementos de las edificaciones.

Imagen 17. Monitoreo de verticalidad de columnas con estación total

A partir de planos y datos obtenidos en los levantamientos topográficos periciales de las estructuras se elaborará un dictamen en dónde se determinará la acción a seguir como respuesta ante la problemática que incluso puede contemplar en modificar el diseño estructural del edificio.

Frecuentemente, para el monitoreo topográfico de edificaciones o de levantamientos periciales los datos se obtienen de forma gradual y también es necesario someterlos a un postproceso que falta sistematizar y automatizar, ocasionando alargar el tiempo de espera de los dictámenes y dar solución a cualquier problemática.

2.6.2 Topografía pericial aplicada para la identificación de desplazamientos verticales en elementos horizontales estructurales de una edificación

Después de algún siniestro que cause incertidumbre acerca de la estabilidad de una edificación es necesario la revisión de sus elementos estructurales, entre ellos, las trabes y vigas que la conforman.

Los elementos estructurales horizontales pueden presentar deformaciones longitudinales y desplazamientos verticales ocurridos por las fuerzas de cargas a las que son sometidas. Es importante monitorear y medir estos desplazamientos para asegurar que no excedan los límites calculados en su diseño y resguardar la estabilidad de la estructura.

La topografía pericial es conveniente para realizar las mediciones de estas deflexiones ya que, mediante una metodología de medición, equipo topográficos y herramientas se puede identificar la existencia de deformaciones. Para ello, es necesario destacar dos conceptos que se aplican en un levantamiento de deflexiones como son: la línea ideal y la catenaria.

Línea Ideal: línea imaginaria que se proyecta en el patín o cara inferior del elemento estructural dividiendo su centro geométrico, desde un punto del principio hasta otro del término de este. (Imagen 18)

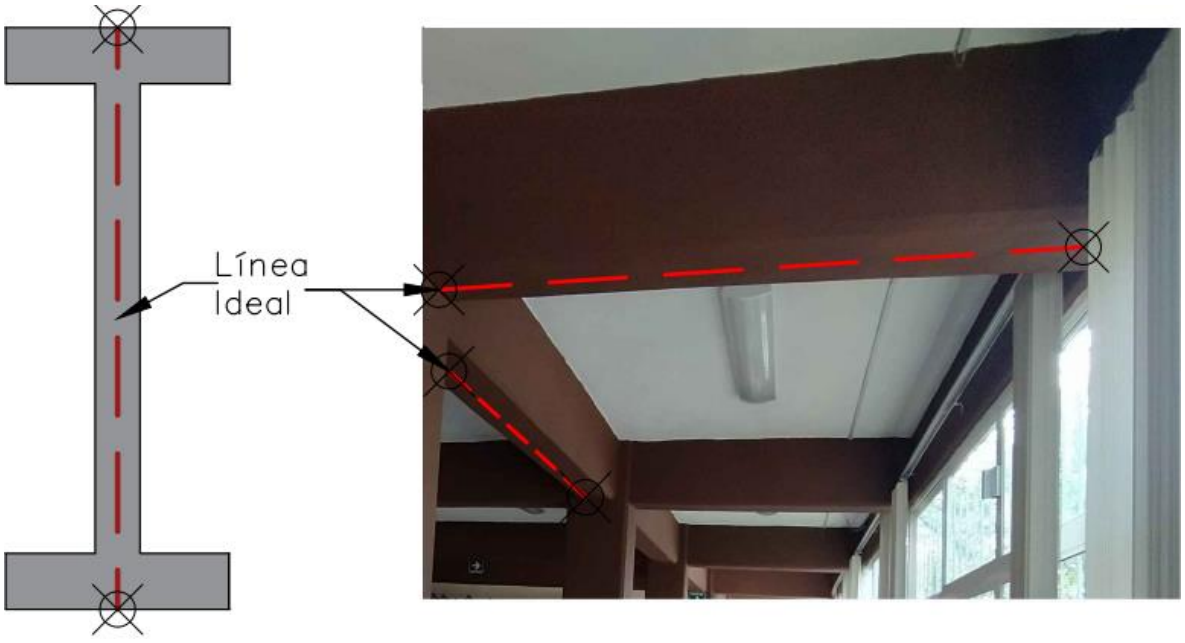


Imagen 18. Izquierda) Vista de Patín inferior de una viga marcando la Línea Ideal; Derecha) Vista de cara inferior de trabes marcando sus Líneas Ideales

Catenaria: se le denomina así a la curva generada por alguna cuerda, cadena o cable suspendida en sus dos extremos y sometida a la fuerza de la gravedad (RAE, 2022).

En este caso, se hace la analogía de la catenaria con respecto a la flecha que se producen por cargas en trabes y vigas en sus caras inferiores. Esta catenaria se puede dividir con puntos en partes proporcionales de acuerdo a sus dimensiones, como se muestra en la siguiente imagen.

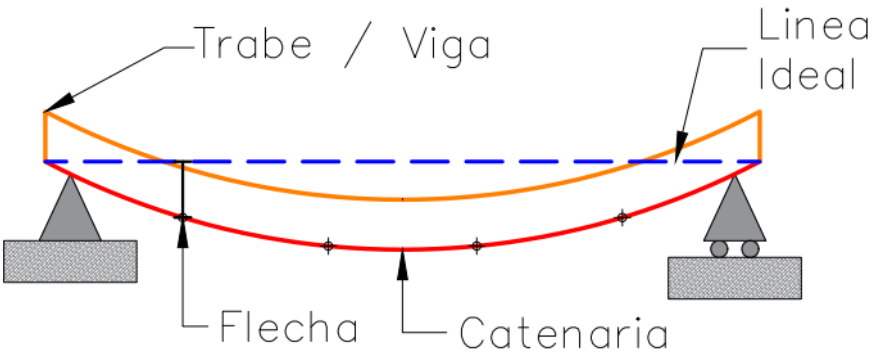


Imagen 19. Catenaria formada en trabe / viga y dividida en 5 segmentos

La línea ideal y la catenaria son dos conceptos de suma importancia, debido a que representan los elementos a considerar en un levantamiento de deflexiones y con la información obtenida se podrá calcular las flechas en trabes y vigas.

De esta manera, en el siguiente capítulo se describirá la forma en la que se aplican estos conceptos y la metodología o el proceso para obtener las deflexiones al considerar características físicas como el material y la geometría de los elementos, así como, el riesgo de estos desplazamientos de acuerdo con las normas mencionadas anteriormente.

3 ANÁLISIS, DISEÑO E IMPLEMENTACIÓN

3.1 Propuesta de la metodología: levantamiento de flechas en elementos estructurales horizontales confinadas con estación total

A continuación, se describirá una propuesta de procedimiento para realizar un levantamiento topográfico de flechas en una trabe confinada con el fin de documentar este tipo de trabajos técnicos que pueden emplearse y aplicarse en el monitoreo y mantenimiento de edificaciones. Es relevante mencionar que los alcances en este procedimiento no se contempla un análisis costo-beneficio comparado con otros métodos de levantamiento.

El material y equipo necesario para realizar este levantamiento es

« **Estación total con la opción de medición sin prisma**	
« **Tripié**	
« **Dianas reflectantes**	
« **Tiralíneas**	

Tabla 3. Material y equipo requerido para levantamiento

Los rendimientos de este procedimiento dependen de la cantidad de información a levantar, y de tiempos prolongados para realizar el levantamiento con el mínimo número de personal y equipo.

A continuación, se describe el procedimiento para realizar un levantamiento de deflexiones en trabes confinadas con equipo topográfico:

Paso 1. Reconocimiento y Orientación

Es importante hacer un reconocimiento previo del entorno donde se trabajará y considerar lugares estratégicos para colocarse con el equipo para evitar que cualquier objeto interfiera en la visualización de los elementos a levantar.

Es necesario realizar una orientación absoluta con el equipo, es decir, emplear un sistema de coordenadas correspondiente al proyecto con el fin de que en su postproceso la localización de los elementos concuerde con lo físico en la obra.

Paso 2. Preparación del elemento

Se procederá a marcar la línea ideal de la trabe en su centro geométrico. Esto se realiza al colocar de un extremo a otro el hilo del tiralíneas en el patín de la trabe y una vez tensado lo suficiente se tirará del mismo hilo para marcar una línea recta.

Esta propuesta también puede considerar medir las deflexiones en losas, para ello se deberá marcar la línea ideal con un tiralíneas en donde se considere o se requiera conocer su deformación.

Desde este paso también se podrá distinguir visualmente si el elemento estructural tuvo alguna deformación longitudinal en caso de que la línea marcada no este dividiendo en partes iguales desde el centro geométrico del patín de la trabe.

En caso de que el equipo a emplear tenga que medir con dianas reflectantes se tendrá que dividir el patín inferior del elemento en partes iguales y pegar sobre la línea ideal las dianas para poder medir los puntos de la catenaria. Para realizar una división distribuida en partes iguales se debe considerar la longitud de la trabe, así como su peralte teniendo en cuenta que el número de puntos o divisiones es inversamente proporcional a la dimensión del peralte y directamente proporcional a la longitud.

En este trabajo se propone que, para obtener el número de puntos o divisiones mínimos y necesarios al considerar la geometría del elemento, se apliquen las siguientes expresiones:

Losa	Trabe / Viga
$\boldsymbol{Num = \frac{L}{8P}}$	$\boldsymbol{Num = \frac{L}{4P}}$

Donde:
L = Longitud del elemento
P = Peralte del elemento

Tabla 2. Número de puntos a levantar de la catenaria recomendado considerando la geometría del elemento

Paso 3. Levantamiento

Una vez centrado, nivelado y orientado el equipo, se procederá a medir un punto en cada extremo del elemento estructural (***LI-A*** *y* ***LI-B*** respectivamente) mostrados en la Imagen No. 20 los cuales pertenecerán a la Línea Ideal de la trabe. Si las condiciones del lugar lo permiten, se trazará sobre el patín inferior de la trabe la línea con el tiralíneas para emplearlo como referencia para el siguiente paso.

Posteriormente se medirán los puntos que conforman la catenaria del elemento, estos deberán estar sobre la línea trazada de ***LI-A – LI-B*** medida anteriormente, el número de divisiones de la catenaria depende de la longitud del elemento, si el elemento es muy largo pueden hacerse hasta 6 divisiones.

Paso 4. Postproceso

Postproceso de la información obtenida mediante la herramienta tecnológica de ARROW, que se explicará en el capítulo 4 en la “Memoria descriptiva del programa ARROW”

Notas para considerar:

- Se recomienda conocer las dimensiones y geometría de las trabes.
- Se sugiere asignar un código único para los puntos pertenecientes a la Línea Ideal y otro para los puntos de la Catenaria.
- También se aconseja tener un identificador único para el nombre de los puntos que pertenezcan a la Línea Ideal y otro para los puntos de la Catenaria, en caso de olvidar cambiar el código del punto se pueda identificar el error y corregirlo.
- Para evitar que la información se duplique se recomienda marcar o realizar un croquis de las trabes y vigas ya levantadas.

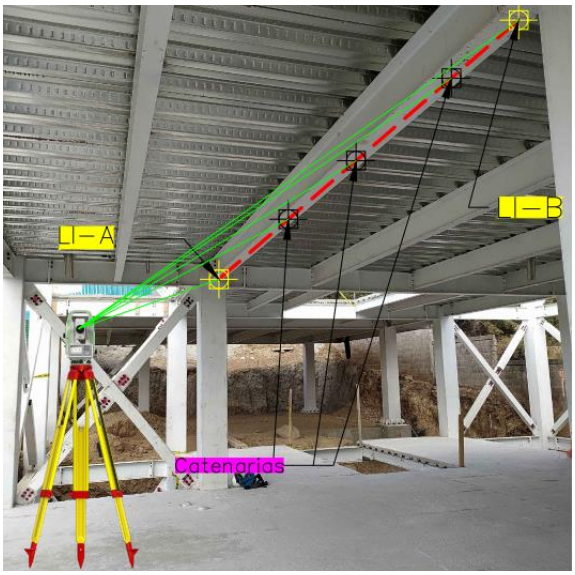


Imagen 20. Esquema del procedimiento para levantamiento de flechas

- Se debe tener cuidado que los puntos levantados en los extremos de la trabe se encuentren sobre la cara inferior del elemento y no sobre alguna conexión con otro o alguna deformación que exista y que pueda generar datos de deflexiones falsos.

Imagen 21. Conexiones de elementos estructurales

- Es necesario realizar una revisión y limpieza de los datos obtenidos en caso de ser necesario, principalmente, en los nombres y códigos de los puntos que pueden llegar a confundir y originar información errónea en su postproceso.
- Este procedimiento se aplica únicamente para trabes confinadas, es decir, que tiene algún apoyo o conexión en sus dos extremos. No aplica para elementos estructurales voladizos.

Muchas veces el trazo de la Línea Ideal sobre el patín inferior de la trabe o la medición de los puntos con las dianas reflectantes se vuelve complicado por las condiciones del entorno y su difícil acceso. Por ello, se recomienda que se utilice una estación total capaz de realizar mediciones sin prisma y sin dianas reflectantes para disminuir los tiempos de la medición y en consecuencia los costos que implica estos levantamientos.

Imagen 22. Diagrama de Flujo para procedimiento de levantamiento de flechas en trabes confinadas

3.2 Memoria descriptiva del programa ARROW

Este programa tiene la finalidad calcular desplazamientos verticales en elementos estructurales horizontales confinados con información obtenida mediante un levantamiento topográfico con estación total.

Arrow es una herramienta desarrollada con fundamentos teóricos multidisciplinarios como geometría analítica, topografía, cálculo estructural y programación con el objetivo de conocer la magnitud de desplazamientos y determinar si el elemento estructural se encuentra dentro del estado límite de servicio de acuerdo a la normatividad mexicana.

3.2.1 Lenguaje de programación

El programa Arrow esta desarrollado con el lenguaje de programación *Python Versión 3.9*, el cual se considera un lenguaje de alto nivel con una sintaxis fácil de leer y escribir, también apto para desarrollo de *software* y que es compatible con todos los sistemas operativos. (Amazon, 2023)

Python también es un lenguaje gratuito y de código abierto que permite tenga una comunidad grande.

La razón principal que se consideró para emplear este lenguaje fue por su amplia colección de bibliotecas y módulos el manejo de funciones matemáticas y otras específicas; el uso de datos de entrada y de salida de diferentes archivos y extensiones.

Específicamente, el programa emplea las siguientes bibliotecas y módulos:

- ***Tkinter***, es un módulo que proporciona las herramientas para administrar ventanas y la interfaz gráfica del programa. (Python, 2022)
- ***Os***, es un módulo para poder leer o escribir archivos, manipular rutas o crear archivos temporales. (Python, 2022)
- ***Pandas***, es una biblioteca para la manipulación y análisis de datos. Sirve para trabajar con datos estructurales y hacer operaciones entre tablas numéricas y series. (Python, 2022)
- ***Numpy***, es una biblioteca que permite manejar y hacer operaciones con matrices de N-dimensionales de forma rápida, además de que brinda funciones matemáticas como integrales, generador de número aleatorios, rutinas de algebra lineal, etc. (Python, 2022)
- ***Math***, es un módulo que proporciona funciones matemáticas desde teoría de números, algorítmicas, trigonométricas, conversiones angulares, hiperbólicas, constantes matemáticas, entre otras. (Python, 2022)
- ***Openpyxl***, es una biblioteca que permite leer y escribir archivos Excel con extensiones *xlsx, xlsm, xltx, xltm*. (Python, 2022)

- ***XlsxWriter***, es un módulo empleado para escribir archivos de texto, número, fórmulas e hipervínculo a archivos *.xlsx* de Excel. (Python, 2022)
- ***CSV***, es un módulo para poder leer y escribir datos tabulares en formatos csv (*Comma Separated Values*). (Python, 2022)
- ***Sys***, es un módulo que proporciona varias funciones y variables que se utilizan para manipular diferentes partes del entorno de ejecución de Python. (Lima, 2022)

Como se muestra en la imagen 23, se enlistan las distintas bibliotecas empleadas en este trabajo.

```
from tkinter import filedialog
from tkinter import ttk
import tkinter as tk
from tkinter import *
from PIL import ImageTk, Image
from tkinter import messagebox
import os
import pandas as pd
import numpy as np
import math
from openpyxl import Workbook
from openpyxl.chart import LineChart, Reference
import xlsxwriter
import csv
import sys
```

Imagen 23. Importación de módulos y bibliotecas empleados en el programa Arrow

3.2.2 Formato de ingreso del archivo de entrada a procesar

Generalmente la información de un equipo topográfico como es la estación total permite descargar los datos del levantamiento con diversos formatos y/o extensiones. Entre ellos está la posibilidad de obtener un archivo delimitado por comas con extensión .***CSV*** y dicho archivo podrá utilizarse en el programa desarrollado para este trabajo.

El archivo con los datos obtenidos en el levantamiento debe contener las siguientes características:

- *El formato de las coordenadas debe ser en el siguiente orden: Nombre, Norte, Este, Elevación, Código;*
- *El archivo no debe contener ningún encabezado o header;*
- *Es importante inspeccionar que el número de los puntos que pertenecen a las Líneas Ideales sea par y;*
- *Revisar que cada par de coordenadas que pertenezcan a una Línea Ideal sean consecutivas.*

Es importante que previo a procesar los datos del archivo se haga una inspección, revisión y en dado caso una limpieza de los datos para evitar errores de nombre o datos duplicados.

3.2.3 Diagrama de Flujo General

El programa se dividió en cuatro bloques: 1) La selección del archivo de entrada a procesar, es decir, un archivo con extensión ***.CSV*** que contenga la información del levantamiento topográfico; 2) El filtrado y ordenamiento de la información de acuerdo con los códigos pertenecientes a la Línea Ideal y Catenaria; 3) Una serie de cálculos para la obtención de las deflexiones y escritura de los archivos de salida; 4) La generación de tres archivos de salida, como se observa en el siguiente diagrama de flujo del programa ARROW.

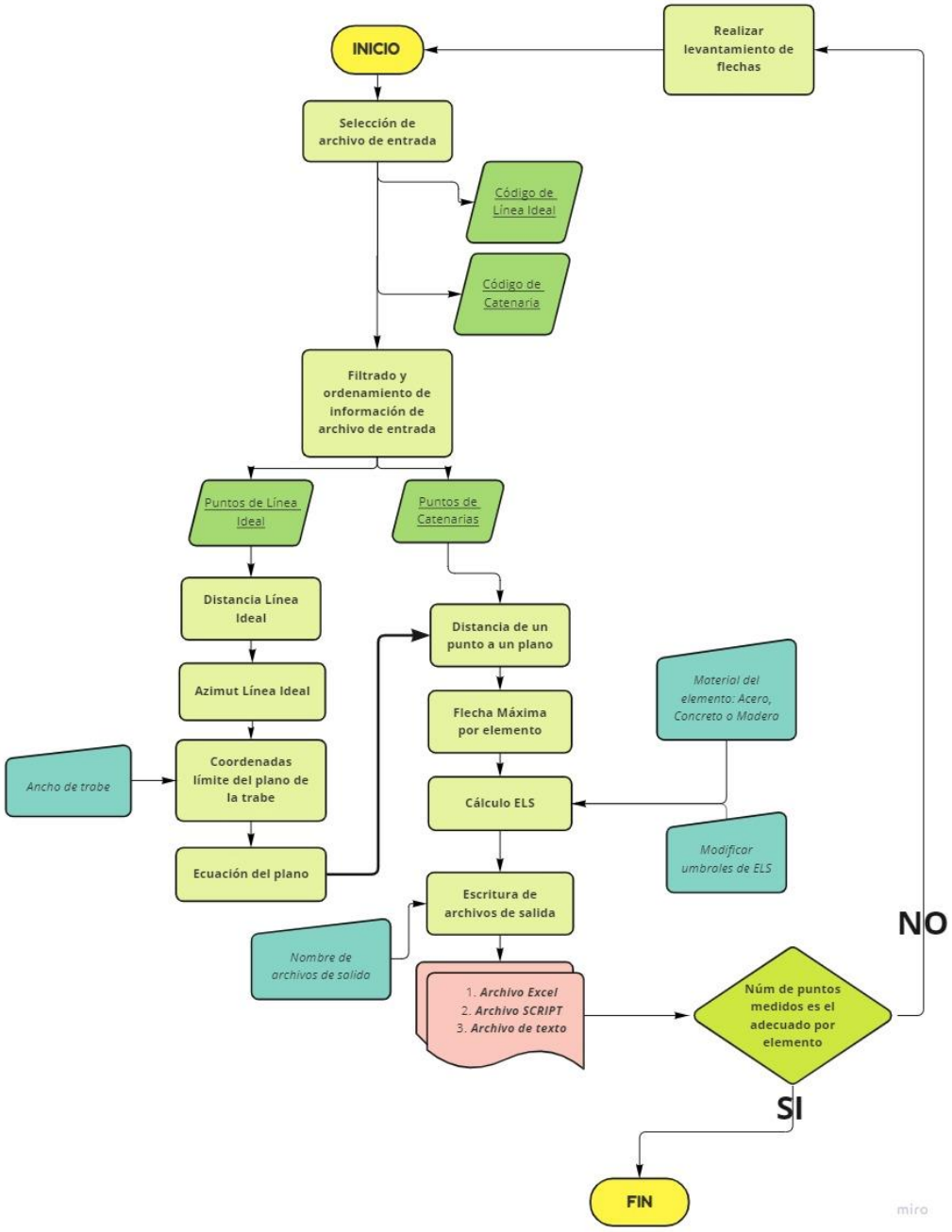


Imagen 24. Diagrama de flujo del programa

3.2.4 Desarrollo del programa

Al inicio del código se desarrollaron las características para la interfaz del programa, la cual consta de una ventana con nombre de "**A R R O W**" y contiene lo siguiente:

- Las instrucciones del formato ***.csv*** de entrada.
- La ruta donde se encuentra el archivo ***.csv*** de entrada.
- Un menú con las opciones del material de la estructura: acero, concreto y madera.
- Un menú para seleccionar el tipo de elemento: Losa o trabe/viga
- Una celda para escribir en metros el peralte del elemento.
- Una celda para escribir en metros el ancho de las trabes.
- Dos menús desplegables que leen los códigos del archivo ***.csv*** de entrada y se seleccionará al que pertenece a las Líneas Ideales y Catenarias respectivamente.
- Una celda para que el usuario escriba el nombre que desee dar a los archivos de salida.
- Una celda para modificar y aumentar los umbrales de los estados límites de servicio si lo desea el usuario.
- Un botón de "continuar" para realizar los cálculos.

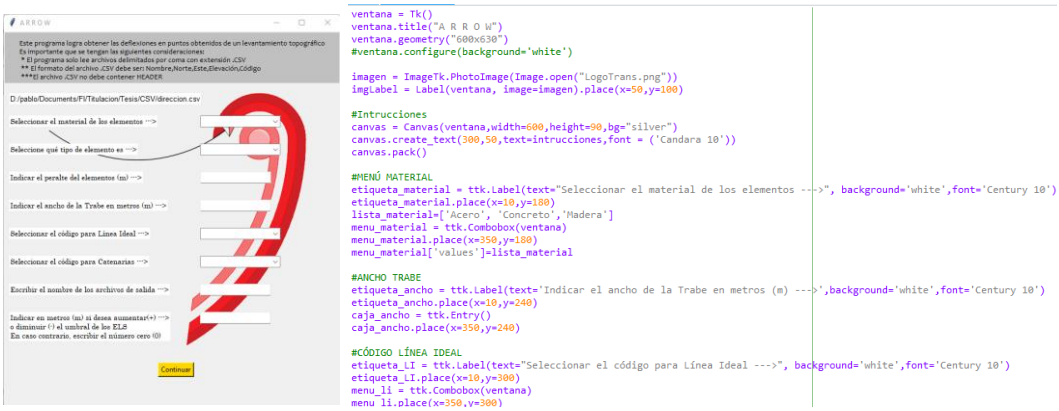


Imagen 25. Ventana principal del programa

Al momento de correr el programa aparecerá un mensaje de aviso para indicar que primero será necesario seleccionar el archivo .**csv** de entrada:

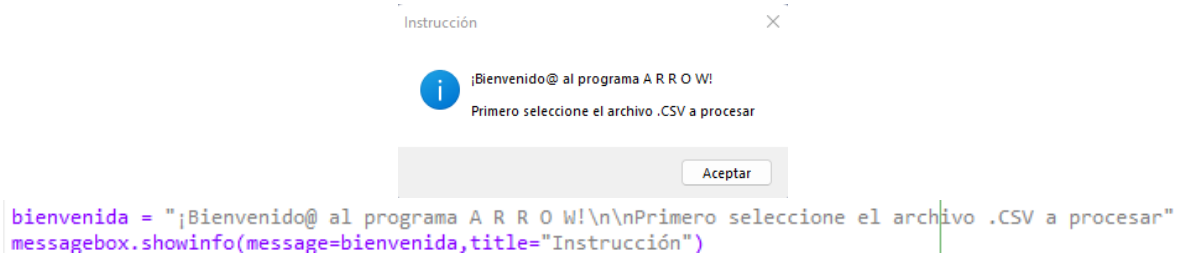
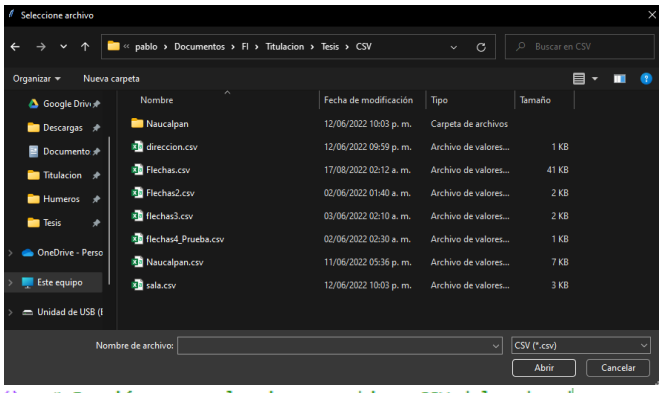


Imagen 26. Ventana de aviso para selección de archivo de entrada.

Para buscar en el ordenador y seleccionar el archivo de entrada se definió una función llamada "*seleccionar_archivo*", dentro de esta función se encuentra un módulo con su extensión llamado *filedialog.askopenfilename* que permite abrir el buscador de documentos en el ordenador y seleccionar un archivo, en este caso, se indica que los archivos deben ser con extensión ***.csv***




```
def seleccionar_archivo():  # Función para seleccionar archivo .CSV del ordenador
    global archivo
    archivo = filedialog.askopenfilename(initialdir="/",title="Seleccione archivo", filetypes=(("CSV", "*.csv")

seleccionar_archivo()
```

Imagen 27. Función para seleccionar archivo de entrada

Una vez seleccionado el archivo se procede a leerlo y agregar el nombre a cada columna en el siguiente orden: Nombre, N (norte), E (este), Z (elevación) y Código. Después, en la última columna se obtienen los valores únicos para crear una lista de los códigos que contiene el archivo sin repetirlos los cuales se guardan en la variable "*codigosA*".

```
In [1]: runfile('D:/pablo/Documents/FI/Titulacion/Tesis/Programas/Arrows App/ArrowApp_08/
ARROW.py', wdir='D:/pablo/Documents/FI/Titulacion/Tesis/Programas/Arrows App/ArrowApp_08')
    Nombre        N         E        Z     CÓDIGO
0       51  3006.419  2001.546  102.586     LIDEAL
1       52  3005.398  2003.593  102.566     LIDEAL
2       53  3005.520  2003.402  102.568  CATENARIA
3       54  3005.795  2002.806  102.571  CATENARIA
4       55  3006.033  2002.367  102.578  CATENARIA
5       56  3006.251  2001.864  102.582  CATENARIA
6       57  3005.396  2003.659  102.589     LIDEAL
7       58  3004.824  2004.832  102.577     LIDEAL
8       59  3004.911  2004.590  102.578  CATENARIA
9       60  3005.113  2004.193  102.578  CATENARIA
10      61  3005.287  2003.775  102.582  CATENARIA
11      62  2997.592  2001.315  102.476     LIDEAL
12      63  2999.164  1998.090  102.483     LIDEAL
13      64  2998.835  1998.760  102.474  CATENARIA
14      65  2998.530  1999.382  102.470  CATENARIA
15      66  2998.109  2000.245  102.472  CATENARIA
16      67  2997.846  2000.785  102.476  CATENARIA
LIDEAL CATENARIA
```

```
#----------------------------- DATAFRAMES DE L IDEAL Y CATENARIA
csv = pd.read_csv(archivo, header=None)  # Leemos archivo CSV
# Agregamos HEADER al archivo CSV
csv.columns = ["Nombre", "N", "E", "Z", "CÓDIGO"]
print(csv)
#Leemos los códigos que existen en el csv
codigosA = str(csv['CÓDIGO'].unique()).replace("[", "").replace("]","").replace("'","" )
print(codigosA)
```

Imagen 28. Lectura de archivo .CSV y obtención de códigos

La lista de códigos obtenida anteriormente será guardada en forma de dos menús y se carga en la ventana principal del programa. Posteriormente, se crea una función nombrada "***calc***" en donde primero se leen los parámetros seleccionados en la interfaz principal del programa como los códigos de la línea ideal y catenarias, respectivamente, el nombre de los archivos de salida introducidos por el usuario y la selección del material.

```
menu_li['values']=codigosA
print(menu_li,"MENU LINEA IDEAL")
menu_cat['values']=codigosA
print(menu_cat,"MENU CATENARIA")

def calc():

    csv = pd.read_csv(archivo, header=None)
    csv.columns = ["Nombre", "N", "E", "Z", "CÓDIGO"]
    codigosA = str(csv['CÓDIGO'].unique()).replace("[", "").replace("]","").replace("'","" )
    lideal_1 = str(menu_li.get())
    print("SELECCIONÓ :",lideal_1," para la L IDEAL")
    catenarias_1 = str(menu_cat.get())
    print("SELECCIONÓ :",catenarias_1," para CATENARIA")
    #------------------------------- CREAMOS ARCHIVOS DE SALIDA
    archivo_out = str(caja_nombrearchivo.get())
    print("ESCRIBIÓ : ",archivo_out," para NOMBRE")
    nom_txt=archivo_out+'.txt'
    nom_scr=archivo_out+'.scr'
    nom_excel=archivo_out+'.xlsx'
    #----------------------------- SELECCIÓN MATERIAL
    material = str(menu_material.get())
    print("SELECCIONÓ EL MATERIAL DE: ",material)
```

```
In [1]: runfile('D:/pablo/Documents/FI/Ti
Documents/FI/Titulacion/Tesis/Programas/A
.!combobox2 MENU LINEA IDEAL
.!combobox3 MENU CATENARIA
SELECCIONÓ : LIDEAL  para la L IDEAL
SELECCIONÓ : CATENARIA  para CATENARIA
ESCRIBIÓ :  Prueba1  para NOMBRE
SELECCIONÓ EL MATERIAL DE:  Concreto
```

Imagen 29. Lectura de parámetros introducidos en interfaz principal del programa

Luego se crearán dos estructuras de columnas y renglones que guardarán y almacenarán las coordenadas, una para los puntos asociados al código de la línea ideal seleccionado y otro para los puntos asociados al código de catenaria seleccionado.

```
# Creamos dataframe de Líneas Ideales
df_lideal = csv.loc[csv['CÓDIGO'] == lideal_1]
df_Icoord = df_lideal[['N','E','Z']]
print("COORDENADAS L IDEALES : ", df_Icoord)
# Creamos otro DF con las coordeandas de las Líneas Ideales
df_catenarias = csv.loc[csv['CÓDIGO'] == catenarias_1]
df_Ccoord = df_catenarias[['N', 'E', 'Z']]
print("COORDENADAS CATENARIAS : ", df Ccoord)
```

```
COORDENADAS L IDEALES :          N        E       Z
0   3006.419  2001.546  102.586
1   3005.398  2003.593  102.566
6   3005.396  2003.659  102.589
7   3004.824  2004.832  102.577
11  2997.592  2001.315  102.476
12  2999.164  1998.090  102.483
COORDENADAS CATENARIAS :          N        E       Z
2   3005.520  2003.402  102.568
3   3005.795  2002.806  102.571
4   3006.033  2002.367  102.578
5   3006.251  2001.864  102.582
8   3004.911  2004.590  102.578
9   3005.113  2004.193  102.578
10  3005.287  2003.775  102.582
13  2998.835  1998.760  102.474
14  2998.530  1999.382  102.470
15  2998.109  2000.245  102.472
16  2997.846  2000.785  102.476
```

Imagen 30. Creación de dataframes para los puntos asociados a la Línea Ideal y para las Catenarias

Se creará un archivo de Excel que contendrá dos hojas: "Flechas" y "Gráficas", de igual forma se modifican el tamaño de las columnas y se crean los formatos de texto que se asignarán a las celdas del archivo.

```
#------------------------------EXCEL
cad_txt = open(nom_txt,'w')
cad_scr = open(nom_scr,'w')
excel = xlsxwriter.Workbook(nom_excel)
hoja = excel.add_worksheet('Flechas')
hoja2 = excel.add_worksheet("Gráficas")
 #---------------------------- FORMATO A ARCHIVO EXCEL
    #Tamaño Columnas
hoja.set_column('A:A',13.00)
hoja.set_column('B:B',28.00)
hoja.set_column('C:C',55.00)
hoja.set_column('D:D',27.00)
hoja.set_column('E:E',14.00)
hoja.set_column('F:F',2.00)
hoja.set_column('G:G',12.00)
#Formatos Texto
titulo00 = excel.add_format({'bold':1,'border':1,'align':'center','fg_color':'#FFC000'})
t1 = excel.add_format({'align':'center'}).set_text_wrap()
t2 = excel.add_format({'align':'left'}).set_text_wrap()
t5 = excel.add_format({'align':'vcenter','border':1,})
t5.set_text_wrap()
```

Imagen 31. Creación de hojas y edición de archivo Excel

Se empezará un bucle ***while()*** que lee el ancho de las trabes introducido por el usuario en la ventana principal y se dividirá a la mitad para emplearse más adelante. También, se crearán contadores de Excel que servirán para escribir en las celdas en cada una de las hojas.

```
while True:
------------------------------ ANCHO TRABES
     trabe = float(caja_ancho.get())
     ancho = trabe/2
     linea = 0
---------------------- CONTADORES EXCEL
---------------------- HOJA 1
     titulo = 1
     renglon2 = 2
     renglon3 = 3
     renglon4 = 4
     renglon5 = 5
     renglon6 = 6
     renglon7 = 7
     renglon8 = 8
     renglon9 = 9
     renglon10 = 10
     renglon11 = 11
```

```
In [1]: runfile('D:/pablo/Documents/FI/Titulacio
ArrowApp_08/ARROW.py', wdir='D:/pablo/Documents/
Arrows App/ArrowApp_08')

2022-09-14T14:26:50.989ZE [13404:ShellIpcClient]
called on MessageLoop that's already been Quit!
EL ANCHO DE LA TRABE ES DE:  0.35
```

Imagen 32. Inicio de ciclo while() con contadores del archivo Excel y lectura del ancho de la trabe

El siguiente paso será la obtención del Azimut de cada Línea Ideal, para ello dentro de un ciclo ***for()*** que recorrerá el *dataframe* que contiene los puntos de las líneas ideales calculará su distancia, así como una segunda distancia llamada *hip* que hace referencia a la hipotenusa que se forma en el triángulo ABC de la figura 33. También calculará la diferencia en X y en Y entre los puntos AB, que más adelante definirán la condición para el cálculo del azimut.

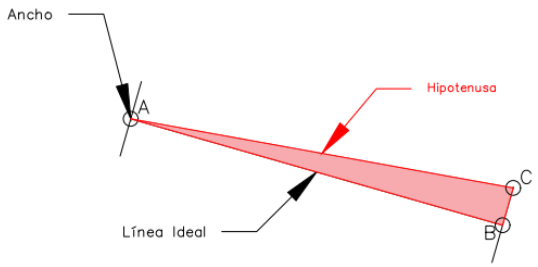


Imagen 33. Triángulo ABC formado por la línea ideal y por el ancho de la trabe.

```
------------------ CÁLCULOS AZIMUT
for i in range(0, len(df_Icoord), 2):
     # Obtenemos la distancia de la Línea Ideal
     pot1 = (df_Icoord.iloc[i+1,1]-df_Icoord.iloc[i,1])**2
     pot2 = (df_Icoord.iloc[i+1,0]-df_Icoord.iloc[i,0])**2
     sumapots = pot1 + pot2
     raizpots = sumapots ** 0.5
     DistABB = raizpots #Distacia plano 2D
     #DistABB = math.dist(df_Icoord.iloc[i+1], df_Icoord.iloc[i]) #Distancia 3D
     DistAB = round(DistABB,3)
     print("La distancia de la línea 1 es de: ",DistAB)
     # Obtenemos la Hipotenusa que se forma de un triángulo ABC
     hip = math.sqrt(DistAB**2 + ancho**2)
     #print("La HIPOTENUSA del triángulo es de : ",hip)
     # Diferencia de X's para aplicar ecuación Tangente
     deltax = df_Icoord.iloc[i+1, 1] - df_Icoord.iloc[i, 1]
     #print("Diferencia de X: ",deltax)
     # Diferencia de Y's para aplicar ecuación Tangente
     deltay = df_Icoord.iloc[i+1, 0] - df_Icoord.iloc[i, 0]
     #print("Diferencia de Y: ",deltay)
     # Obtenemos el Azimut de la Línea Ideal
```

```
La distancia de la línea 1 es de:  2.238
La HIPOTENUSA del triángulo es de :  2.2448316195207156
Diferencia de X:  2.0
Diferencia de Y:  1.0
```

Imagen 34. Obtención de distancia de la Línea Ideal, hipotenusa y diferencia de X y Y

Al tener la diferencia en X y Y se establecerá la condición de acuerdo con la ecuación 14 del ángulo tangente para obtener el azimut y aplicarla al cuadrante correspondiente.

```
#---------------------------- SEGUNDO CUADRANTE
        elif deltax > 0 and deltay < 0:
            AzimutAB = math.degrees(math.atan(deltax/deltay))+180
            print("El AZIMUT de la LÍNEA IDEAL es de: ",AzimutAB)
            print("Esta línea se encuentra en 2° CUADRANTE")
            # Límite 1
            alfa = AzimutAB - 90
            DistXLím1 = math.sin(math.radians(alfa))*ancho
            DistYLím1 = math.cos(math.radians(alfa))*ancho
            Lím1 = [df_Icoord.iloc[i, 0]+DistYLím1,
                    df_Icoord.iloc[i, 1]+DistXLím1, df_Icoord.iloc[i, 2]]
            print("El Límite 1 es : ",Lím1)
            # Límite 2
            Lím2 = [df_Icoord.iloc[i, 0]-DistYLím1,
                    df_Icoord.iloc[i, 1]-DistXLím1, df_Icoord.iloc[i, 2]]
            print("El Límite 2 es : ",Lím2)
            # Límite 3
            alfa2 = math.degrees(math.atan(ancho/DistAB))
            AzimutALím3 = AzimutAB - alfa2
            DistXLím3 = math.sin(math.radians(AzimutALím3))*hip
            DistYLím3 = math.cos(math.radians(AzimutALím3))*hip
            if AzimutALím3 < 90:
                DistYLím3 = DistYLím3 * -1
                Lím3 = [df_Icoord.iloc[i, 0]+DistYLím3,
                        df_Icoord.iloc[i, 1]+DistXLím3, df_Icoord.iloc[i+1, 2]]
            elif AzimutALím3 > 90:
                Lím3 = [df_Icoord.iloc[i, 0]+DistYLím3,
                        df_Icoord.iloc[i, 1]+DistXLím3, df_Icoord.iloc[i+1, 2]]
            print("El Límite 3 es : ",Lím3)
            # Límite 4
            alfa3 = alfa2
            AzimutALím4 = AzimutAB + alfa3
            DistXLím4 = math.sin(math.radians(AzimutALím4))*hip
            DistYLím4 = math.cos(math.radians(AzimutALím4))*hip
```

```
EL ANCHO DE LA TRABE ES DE:  0.35
El AZIMUT de la LÍNEA IDEAL es de:  63.43494882292201
Esta línea se encuentra en 1° CUADRANTE
El Límite 1 es :  [10.156524758424986, 9.921737620787507, 10.0]
El Límite 2 es :  [9.843475241575014, 10.078262379212493, 10.0]
El Límite 3 es :  [11.157388785153891, 11.92346567424532, 9.91]
El Límite 4 es :  [10.844339268303921, 12.079990432670304, 9.91]
El AZIMUT de la LÍNEA IDEAL es de:  103.59648186609768
Esta línea se encuentra en 2° CUADRANTE
El Límite 1 es :  [9.217095701506874, 9.757139425480732, 1.0]
El Límite 2 es :  [8.876904298493127, 9.674860574519267, 1.0]
El Límite 3 es :  [8.696623198567782, 11.909093043401972, 0.9]
El Límite 4 es :  [8.356431795554036, 11.826814192440509, 0.9]
El AZIMUT de la LÍNEA IDEAL es de:  253.7453542108045
Esta línea se encuentra en 3° CUADRANTE
El Límite 1 es :  [8.475995246740084, 8.536983700167243, 100.0]
El Límite 2 es :  [8.812004753259917, 8.439016299832756, 100.0]
El Límite 3 es :  [7.8963081436180325, 6.548767448731305, 100.2]
El Límite 4 es :  [8.232317650137867, 6.450800048396821, 100.2]
El AZIMUT de la LÍNEA IDEAL es de:  306.9206141027554
Esta línea se encuentra en 4° CUADRANTE
El Límite 1 es :  [9.844092997596166, 7.924876117469085, 50.5]
El Límite 2 es :  [10.123907002403834, 8.135123882530912, 50.5]
El Límite 3 es :  [11.366887524543971, 5.898223254076392, 51.0]
El Límite 4 es :  [11.646701529351642, 6.10847101913822, 51.0]
```

Imagen 35. Cálculo de azimut, así como los límites del plano de la trabe.

Es importante mencionar que para los límites en los valores de elevación (Z) se les asigna la misma elevación de los puntos de la Línea Ideal dependiendo del lado en que se encuentren los límites, esto con el fin de poder generar un plano, siendo la cara inferior de la trabe o viga.

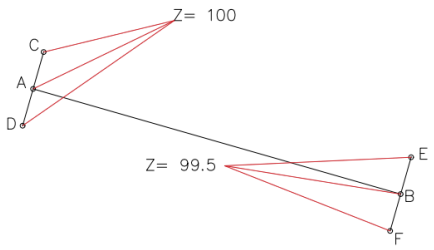


Imagen 36. Asignación de elevaciones para delimitar plano de la cara inferior de la trabe o viga

Posteriormente es necesario identificar las coordenadas máximas y mínimas de los límites obtenidos para así acotar más adelante la búsqueda de los puntos de catenarias que se encuentren dentro del plano formado. Para ello se usan la funciones ***max()*** y ***min()*** para las coordenadas en **X, Y** y **Z**. En el caso particular de la Z, se suman y se restan 20 centímetros para poder acotar los puntos de catenarias en sus elevaciones y evitar errores.

```
# Obtendremos las coordenas límites en X,Y,Z
 xmin = min(Lim1[1], Lim2[1], Lim3[1], Lim4[1])
 xmax = max(Lim1[1], Lim2[1], Lim3[1], Lim4[1])
 ymin = min(Lim1[0], Lim2[0], Lim3[0], Lim4[0])
 ymax = max(Lim1[0], Lim2[0], Lim3[0], Lim4[0])
 zmin = min(Lim1[2], Lim2[2], Lim3[2], Lim4[2])-0.2
 zmax = max(Lim1[2], Lim2[2], Lim3[2], Lim4[2])+0.2
 print("X Min: ",xmin," X Max: ",xmax)
 print("Y Min: ",ymin," Y Max: ",ymax)
 print("Z Min: ",zmin," Z Max: ",zmax)
```

```
X Min:  9.921737620787507  X Max:  12.079990432670304
Y Min:  9.843475241575014  Y Max:  11.157388785153891
Z Min:  9.71  Z Max:  10.2
El Límite 2 es :  [8.876904298493127, 9.674860574519267, 1.0]
X Min:  9.674860574519267  X Max:  11.909093043401972
Y Min:  8.356431795554036  Y Max:  9.217095701506874
Z Min:  0.7  Z Max:  1.2
X Min:  6.450800048396821  X Max:  8.536983700167243
Y Min:  7.8963081436180325  Y Max:  8.812004753259917
Z Min:  99.8  Z Max:  100.4
X Min:  5.898223254076392  X Max:  8.135123882530912
Y Min:  9.844092997596166  Y Max:  11.646701529351642
Z Min:  50.3  Z Max:  51.2
```

Imagen 37. Obtención de máximos y mínimos del plano generado por los límites

Al tener una una línea que se encuentra en el plano (Línea Ideal) y además cuatro puntos no colineales a la Línea Ideal se puede encontrar una ecuación del plano, como se explica en el capítulo 2 de este trabajo. Es decir, se obtiene el vector director de AB y el otro vector del límite C al punto A, con lo anterior se puede calcular el producto cruz y generar la ecuación del plano al desarrollar un producto punto. Dicha ecuación se guardará en el programa como una lista para facilitar posteriormente el manejo de cada uno de sus elementos, la ecuación del plano se guarda como:

$$Ecuación\ del\ plano = Ax + By + Cz + D \Longrightarrow plano = [A\ , B, C, D]$$

```
---------------- ECUACIÓN DEL PLANO
 # Obtenemos el vector director de AB
 v = [df_Icoord.iloc[i+1, 1]-df_Icoord.iloc[i, 1], df_Icoord.iloc[i+1,0]-df_Icoord.iloc[i, 0], df_Icoord.iloc[i+1, 2]

 # Ahora se calcula el vector CLA, siendo C la "esquina del plano" obtenida anteriormente
 CLA = [df_Icoord.iloc[i, 1]-Lim1[1], df_Icoord.iloc[i, 0] - Lim1[0], df_Icoord.iloc[i, 2]-Lim1[2]]

 # Hacemos producto cruz para obtener vector normal al plano, siendo CLA x V
 n = np.cross(CLA, v)
 print("PRODUCTO CRUZ DE VECTORES: ",n)
 plano = [n[0], n[1], n[2], -(n[0]*df_Icoord.iloc[i, 1])-(n[1]*df_Icoord.iloc[i, 0])-(n[2]*df_Icoord.iloc[i, 2])]
 print("LA ECUACIÓN DEL PLANO ES:  ", plano)
```

```
PRODUCTO CRUZ DE VECTORES:  [ 0.00313202 -0.00156219  0.40031226]
LA ECUACIÓN DEL PLANO ES:   [0.003132023728402887, -0.0015621867253045442, 0.4003122609288951, -42.638735312638154]
PRODUCTO CRUZ DE VECTORES:  [ 0.00188754 -0.00092044  0.2283809 ]
LA ECUACIÓN DEL PLANO ES:   [0.0018875374403924785, -0.0009204359896885202, 0.2283809003068515, -24.445074920193022]
PRODUCTO CRUZ DE VECTORES:  [ 1.10114881e-03 -5.36746025e-04  6.27852810e-01]
LA ECUACIÓN DEL PLANO ES:   [0.0011011488112188683, -0.0005367460251886057, 0.6278528096820586, -64.93464456696596]
```

Imagen 38. Obtención de la ecuación del plano de la cara inferior de cada trabe o viga

Al tener la ecuación del plano se puede calcular la distancia perpendicular de un punto a dicho plano al aplicar la ecuación 10 del capítulo 2 de este trabajo. Para ello, dentro de un segundo ciclo ***for()*** se leerá el *dataframe* correspondiente a los puntos de la catenaria con una condicional de que se encuentren dentro de los límites del plano calculados anteriormente.

Se guardarán todas las flechas calculadas de cada plano en una lista que posteriormente se identificará el valor máximo que indicará la deflexión mayor calculada.

```
---------- CÁLCULO FLECHAS
aa =[]
for j in range(0, len(df_Ccoord)):
    if xmin <= df_Ccoord.iloc[j, 1] <=xmax and ymin <= df_Ccoord.iloc[j, 0] <=ymax and zmin <= df_Ccoord.iloc[j, 2]
        numerador = abs((plano[0]*df_Ccoord.iloc[j, 1]) + (
        plano[1]*df_Ccoord.iloc[j, 0]) + (plano[2]*df_Ccoord.iloc[j, 2]) + plano[3])
        denominador = math.sqrt(
            plano[0]**2 + plano[1]**2 + plano[2]**2)
        flecha2 = numerador/denominador
        flecha = round(flecha2,3)
        print("FLECHA          :      ",flecha)
        bb = [flecha,df_Ccoord.iloc[j,0],df_Ccoord.iloc[j,1],df_Ccoord.iloc[j,2]]
        aa.append(bb)
---------- FLECHA MÁXIMA
for m in range(0,len(aa)):
    flechamax = max(aa[:])
cc = flechamax[1:]
flechamaxpoint = flechamax[0]
flecharedon = round(flechamaxpoint,3)
print("LA FLECHA MÁXIMA ES: ",flecharedon)
```

```
FLECHA          :       0.0
FLECHA          :       0.003
FLECHA          :       0.0
FLECHA          :       0.001
LA FLECHA MÁXIMA ES:  0.003
FLECHA          :       0.001
FLECHA          :       0.005
FLECHA          :       0.006
LA FLECHA MÁXIMA ES:  0.006
FLECHA          :       0.008
FLECHA          :       0.01
FLECHA          :       0.006
FLECHA          :       0.001
LA FLECHA MÁXIMA ES:  0.01
```

Imagen 39. Lectura del Dataframe de catenarias para encontrar los puntos que caen dentro del plano que contiene la línea ideal conforme a los límites calculados previamente.

Al momento de realizar el levantamiento es poco probable que un punto de catenaria se halle exactamente sobre la Línea Ideal y para que en el momento de generar algún plano la visualización de las flechas sea sencilla es necesario calcular una proyección de coordenadas que se encuentre sobre la línea ideal respetando la coordenada Z del punto a proyectar.

La obtención de las coordenadas se calcula de la misma manera como se ha hecho anteriormente, con la diferencia en X y Y del punto C con el A (Imagen 19), se aplican los mismos criterios para calcular el Azimut de la línea AC (Ecuación 14).

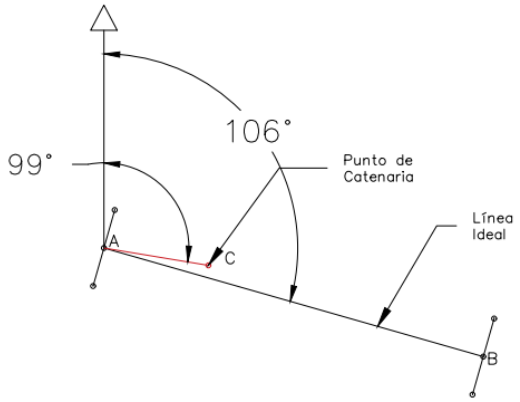


Imagen 40. Azimut de la Línea AC, siendo C un punto de catenaria y A el origen de la Línea Ideal

```
---------- Calculamos un punto C sobre la LIDEAL que contiene la Flecha Máxima
DistAC = math.dist(df_Icoord.iloc[i],cc)
------------------- AZIMUT A CADA PUNTO DE CATENARIAS
#Diferencia de X's para aplicar ecuación Tangente
deltax2 = cc[1] - df_Icoord.iloc[i, 1]
# Diferencia de Y's para aplicar ecuación Tangente
deltay2 = cc[0] - df_Icoord.iloc[i, 0]

----------------------- PRIMER CUADRANTE
if deltax2 > 0 and deltay2 > 0:
    AzimutAC = math.degrees(math.atan(deltax2/deltay2))
---------------------- SEGUNDO CUADRANTE
elif deltax2 > 0 and deltay2 < 0:
    AzimutAC = math.degrees(math.atan(deltax2/deltay2))+180
---------------------- TERCER CUADRANTE
elif deltax2 < 0 and deltay2 < 0:
    AzimutAC = math.degrees(math.atan(deltax2/deltay2))+180
---------------------- CUARTO CUADRANTE
elif deltax2 < 0 and deltay2 > 0:
    AzimutAC = 360+math.degrees(math.atan(deltax2/deltay2))

beta = abs(AzimutAC-AzimutAB)
cosbeta = math.cos(math.radians(beta))
ca = DistAC * cosbeta
xc = math.sin(math.radians(AzimutAB))*ca
yc = math.cos(math.radians(AzimutAB))*ca
C = [df_Icoord.iloc[i,1] + xc, df_Icoord.iloc[i,0] + yc,cc[2]]
print("La Flecha Máxima se encuentra en las coordenadas:  ",C)
```

```
La Flecha Máxima se encuentra en las coordenadas:   [2002.8042897518367, 3005.7913918726795, 102.571]
La Flecha Máxima se encuentra en las coordenadas:   [2003.7957888758438, 3005.329296473161, 102.582]
La Flecha Máxima se encuentra en las coordenadas:   [1999.3836565767049, 2998.533417631448, 102.47]
```

Imagen 41. Obtención de Flecha Máxima sobre la Línea Ideal

Dependiendo del material seleccionado (acero, concreto o madera) en la ventana principal del programa se obtendrán el estado límite de servicio de acuerdo con la normatividad mencionada en el capítulo 2, dichas expresiones dependen de la longitud de la trabe que en este caso será la misma que la distancia de la línea ideal calculada.

También, se agrega una variable más, la cual permite modificar los umbrales de los ELS dependiendo si se desea aumentar o disminuir dichos límites, esto es a consideración de los requisitos y conocimientos del estructurista o del usuario.

```
-------------------- ESTADOS LÍMITE DE SERVICIO
#Se calcularán los Estados Límites de Servicio de acuerdo con el material
umbral = float(caja_umbral.get())
if material == 'Acero': #ACERO
    els1_mx = round(DistAB/240,3) + umbral
    els2_mx = round(DistAB/480,3) + umbral
    els3_mx = round(DistAB/480,3) + umbral
    els4_mx = round(DistAB/960,3) + umbral
    print("ELS 1: ",els1_mx)
    print("ELS 2: ",els2_mx)
    print("ELS 3: ",els3_mx)
    print("ELS 4: ",els4_mx)
```

```
ELS 1:  0.009
ELS 2:  0.005
ELS 3:  0.005
ELS 4:  0.002
```

Imagen 42. Obtención de los ESL de una trabe para el material de acero

Cada material tiene establecido un diseño de tabla diferente que indican casos específicos en los que se aplican las normas. Cada elemento de la tabla se escribe a través de comandos del módulo ***XlsxWriter*** con sus diferentes diseños de formato de texto y celdas.

En cada archivo de Excel se crea una segunda hoja de nombre “Gráficas” en donde se aprecia el comportamiento de las deflexiones en trabes a través de un perfil.

```
---------- GRÁFICAS

celda_tih2 ='A'+str(h2titulo)+':H'+str(h2titulo)
texto_tih2 = 'LINEA '+str(linea)
hoja2.merge_range(celda_tih2,texto_tih2,titulo00)
for p in range(0,len(aa),1):
    hoja2.write(renglon2,0,'Punto',t5)
    hoja2.write(renglon3,0,'Flecha (m)',t5)
    hoja2.write(renglon2,1+p,p,t5)
    hoja2.write(renglon3,1+p,aa[p][0]*-1,t5)
    chart = excel.add_chart({'type':'line'})
    chart.add_series({'values': ['Gráficas',renglon3,1,renglon3,len(aa)], 'line':{'color':'red'},})
    chart.set_y_axis({'name':'Flecha','num_font':{'arial black':True},})
    chart.set_x_axis({'minor_unit':0})
    chart.set_size({'width':445,'height':159})
    chart.set_title({'name':texto_tih2, 'num_font':{'name':'arial black','size':12},})
    chart.set_legend({'position':'none'},)
    grafpos = "I"+str(h2titulo)
    hoja2.insert_chart(grafpos, chart)
```

LINEA 2

Punto	0	1	2
Flecha (m)	-0.001	-0.005	-0.006

LINEA 3

Punto	0	1	2	3
Flecha (m)	-0.008	-0.01	-0.006	-0.001

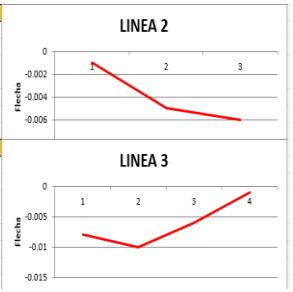


Imagen 43. Creación de la hoja “Gráficas” para perfiles de las deflexiones.

```
celda_titulo ='A'+str(titulo)+':F'+str(titulo)
texto_titulo ='LINEA '+str(linea)
hoja.merge_range(celda_titulo,texto_titulo,titulo00)
celda_distancia ='A'+str(renglon2)
hoja.write(celda_distancia,'Distancia (m)',t2)
celda_distancia2 = 'B'+str(renglon2)
hoja.write(celda_distancia2,DistAB,t1)
celda_max = 'C'+str(renglon2)
hoja.write(celda_max,'Deflexión Máx (m)',t2)
celda_max2 = 'D'+str(renglon2)
hoja.write(celda_max2,flecharedon,t1)
celda_espacio1 = 'A'+str(renglon3)+':F'+str(renglon3)
hoja.merge_range(celda_espacio1,'')

celda_ntc = 'A'+str(renglon4)+':A'+str(renglon7)
hoja.merge_range(celda_ntc,'NTC CDMX',t5)
ntc1 = 'B'+str(renglon4)+':B'+str(renglon5)
hoja.merge_range(ntc1,'Trabes / Vigas',t5)
ntc2 = 'B'+str(renglon6)+':B'+str(renglon7)
hoja.merge_range(ntc2,'Trabes / Vigas que afecten a elementos NO estructurales',t5)

c4 = 'C'+str(renglon4)
hoja.write(c4,'No voladizas',t5)
c5 = 'C'+str(renglon5)
hoja.write(c5,'Voladiza',t5)
c6 = 'C'+str(renglon6)
hoja.write(c6,'No voladiza',t5)
c7 = 'C'+str(renglon7)
hoja.write(c7,'Voladiza',t5)
```

Imagen 44. Escritura de tabla de excel

LINEA 19				
Distancia (m)	1.659	Deflexión Máx (m)	0.001	
NTC CDMX	Trabes / Vigas	No voladizas	ELS = 0.007	SÍ CUMPLE
		Voladiza	ELS = 0.003	NO APLICA
	Trabes / Vigas que afecten a elementos NO estructurales	No voladiza	ELS = 0.003	SÍ CUMPLE
		Voladiza	ELS = 0.002	NO APLICA
ACI 318S-140	Trabes / Vigas	Debido a cargas de nieve y lluvia	ELS = 0.009	SÍ CUMPLE
		Debido a carga viva	ELS = 0.005	SÍ CUMPLE
	Trabes / Vigas que afecten a elementos NO estructurales	Susceptibles de sufrir daños por deflexiones grandes	ELS = 0.003	SÍ CUMPLE
		No susceptibles de sufrir daños debido a deflexiones grandes	ELS = 0.007	SÍ CUMPLE

Punto	0	1	2	3
Flecha (m)	0	0	0.001	0.001
El núm. de puntos por trabe es el recomendado				
Num de puntos recomendado:				
	1.4			

Imagen 45. Ejemplo de escritura de datos tabulares

Finalmente se escriben en un archivo de texto y en un script ***(.scr)*** los comandos necesarios para poder dibujar en un programa de diseño asistido por computadora como AutoCad cada línea ideal con su respectiva numeración y también el punto donde se localiza la flecha máxima en la misma línea.

```
---------------------- ESCRIBIMOS EN ARCHIVOS TXT Y SCR
line_cad ="LINE",str((df_Icoord.iloc[i, 1])).replace(" ","")+","+str(df_Icoord.iloc[i, 0]).replace(" ","")+","+str(df_Icoord.iloc[i, 2]).replace(" ","")+" "+str(df_Ic
print(*line_cad,file=(cad_txt))
print(*line_cad,file=(cad_scr))
point_cad ="POINT",str(C[0]).replace(" ","")+","+str(C[1]).replace(" ","")+","+str(C[2]).replace(" ","")
print(*point_cad,file=(cad_txt))
print(*point_cad,file=(cad_scr))
textline_cad ="TEXT",str(df_Icoord.iloc[i, 1]).replace(" ","")+","+str(df_Icoord.iloc[i, 0]).replace(" ","")+","+str(df_Icoord.iloc[i, 2]).replace(" ","")+" 0.02 0 Lí
print(*textline_cad,file=(cad_txt))
print(*textline_cad,file=(cad_scr))
pointext_cad ="TEXT",str(C[0]).replace(" ","")+","+str(C[1]).replace(" ","")+","+str(C[2]).replace(" ","")+" 0.02 0 Flecha_Max "+str(flecharedon).replace(" ","")+"  "
print(*pointext_cad,file=(cad_txt))
print(*pointext_cad,file=(cad_scr))
```

Imagen 46. Escritura de comandos en los archivos de texto y .SCR para poder dibujar los planos correspondientes.

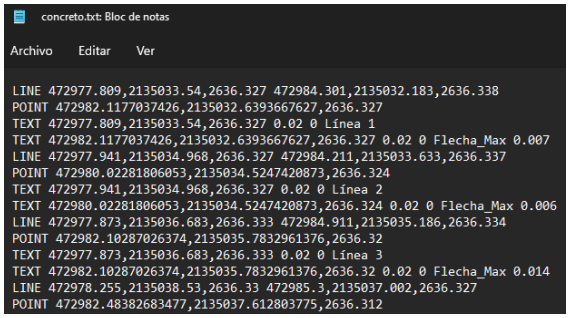


Imagen 47. Ejemplo de escritura de comandos en archivos de texto

3.2.5 Archivos de salida

El programa genera tres archivos al finalizar sus cálculos:

1. Una hoja de cálculo con extensión ***.xlsx*** la cual contiene una serie de tablas diferentes dependiendo del material seleccionado en la ventana principal.

   Cada elemento tiene su propia tabla donde se indica: su longitud, la flecha máxima obtenida, diferentes ELS calculados dependiendo de las características de las trabes y además si las deflexiones cumplen o no la normatividad aplicada. También se anexa una tabla que indica el número de puntos levantados y la flecha obtenida en cada uno de ellos.

LINEA 13										
Distancia (m)	6.276	Deflexión Máx (m)	0.006							
					Punto	0	1	2	3	
NTC CDMX	Trabes / Vigas	No voladizas	ELS = 0.026	SÍ CUMPLE	Flecha (m)	0.003	0.006	0.006	0.003	
		Voladiza	ELS = 0.013	NO APLICA	Se recomienda medir más puntos en este elemento					
	Trabes / Vigas que afecten a elementos NO estructurales	No voladiza	ELS = 0.013	SÍ CUMPLE	Num de puntos recomendado:					
		Voladiza	ELS = 0.007	NO APLICA		5.1				

Imagen 48. Ejemplo de la tabla de una trabe de acero, se indica su longitud, la deflexión máxima que se calculó y dependiendo de las características de la trabe se muestra el ESL máximo y permisible para conocer si las flechas obtenidas cumplen con la normatividad. Del lado derecho se muestra la cantidad de puntos de catenarias levantadas en esa trabe junto con su flecha calculada.

Nota: Es importante poner a consideración los resultados con un especialista en la materia.

LINEA 9										
Distancia (m)	1.676	Deflexión Máx (m)	0.002							
					Punto	0	1	2	3	
NTC CDMX	Trabes / Vigas	No voladizas	ELS = 0.007	SÍ CUMPLE	Flecha (m)	0.001	0.002	0.001	0.001	
		Voladiza	ELS = 0.003	NO APLICA	El núm. de puntos por elemento es el recomendado					
	Trabes / Vigas que afecten a elementos NO estructurales	No voladiza	ELS = 0.003	SÍ CUMPLE	Num de puntos recomendado:					
		Voladiza	ELS = 0.002	NO APLICA		1.4				
ACI 318S-140	Trabes / Vigas	Debido a cargas de nieve y lluvia	ELS = 0.009	SÍ CUMPLE						
		Debido a carga viva	ELS = 0.005	SÍ CUMPLE						
	Trabes / Vigas que afecten a elementos NO estructurales	Susceptibles de sufrir daños por deflexiones grandes	ELS = 0.003	SÍ CUMPLE						
		No susceptibles de sufrir daños debido a deflexiones grandes	ELS = 0.007	SÍ CUMPLE						

Imagen 49. Ejemplo de la tabla de una trabe de concreto donde se indica su longitud, la flecha máxima, así como los ELS obtenidos de acuerdo con la NTC de la CDMX y del ACI 318S-140 Del Instituto Americano del Concreto y dependiendo de la característica de la trabe cumple con dichas normas. Del lado derecho se muestra la cantidad de puntos de catenarias levantadas en esa trabe junto con su flecha calculada.

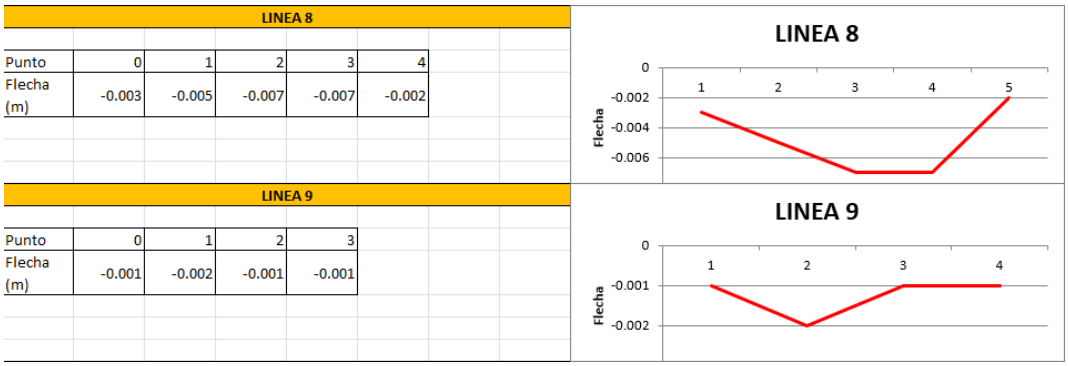
Nota: Es importante poner a consideración los resultados con un especialista en la materia.

LINEA 8											
Distancia (m)	6.362	Deflexión Máx (m)	0.007								
					Punto	0	1	2	3	4	
NTC CDMX	Trabes / Vigas	< 3.5 m	ELS = 0.027	SÍ CUMPLE	Flecha (m)	0.003	0.005	0.007	0.007	0.002	
		> 3.5 m	ELS = 0.013	SÍ CUMPLE	Se recomienda medir más puntos en este elemento						
	Trabes / Vigas que afecten a elementos NO estructurales	< 3.5 m	ELS = 0.032	SÍ CUMPLE	Num de puntos recomendado:						
		> 3.5 m	ELS = 0.016	SÍ CUMPLE		5.2					

Imagen 50. Ejemplo de la tabla de una trabe de madera donde se indica su longitud, la flecha máxima, así como los ELS dependiendo de las características de la trabe de acuerdo con la NTC de la CDMX. Del lado derecho se muestra la cantidad de puntos de catenarias levantadas en esa trabe junto con su flecha calculada.

Nota: Es importante poner a consideración los resultados con un especialista en la materia.

En una segunda hoja de nombre “Gráfica” se pueden visualizar los perfiles del comportamiento de la catenaria por cada elemento. En este caso, los valores de las deflexiones son negativos con el único fin de apreciar visualmente la catenaria.



LINEA 8					
Punto	0	1	2	3	4
Flecha (m)	-0.003	-0.005	-0.007	-0.007	-0.002

LINEA 9				
Punto	0	1	2	3
Flecha (m)	-0.001	-0.002	-0.001	-0.001

Imagen 51. Ejemplo de perfil del comportamiento de la catenaria en trabes

2. Un archivo ***script*** (***.scr)*** que es capaz de abrirlo desde un programa asistido por computadora como Autocad o Civil3D con el comando: SCR o SCRIPT, el cual contiene una serie de comandos escritos que permiten visualizar las líneas ideales levantadas y el punto donde se encuentra la flecha máxima en cada elemento.
3. Un archivo de texto ***(.txt)*** que contiene la misma serie de comandos escritos en el ***script***, el motivo de generar un archivo como este es en caso de alguna aclaración o duda de los comandos y que se pueda revisar en este documento.

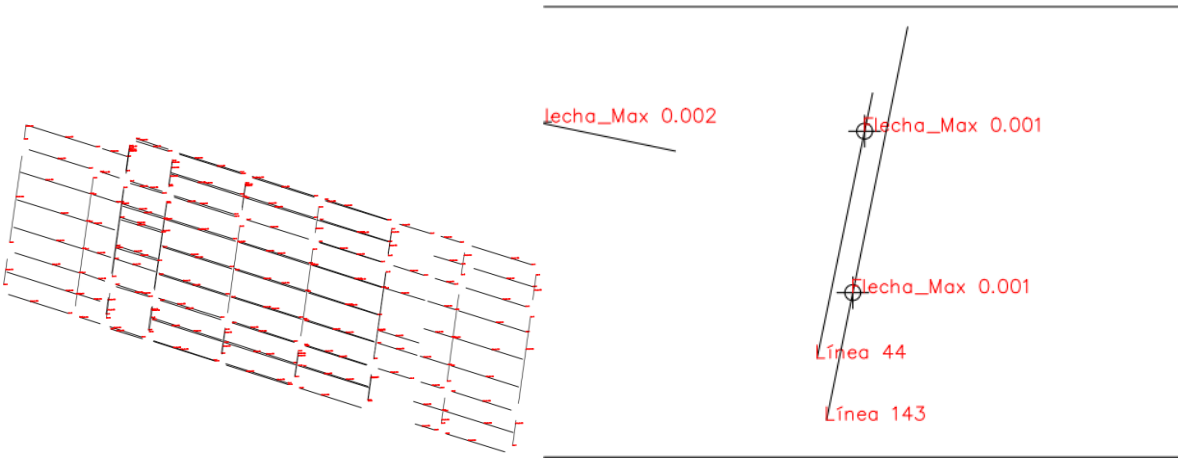


Imagen 52. Plano generado a partir del script obtenido en el programa

3.2.6 Ventajas y límites del programa ARROW

Previo al desarrollo de este programa, la obtención de las deflexiones a través de la información levantada en campo era un procesamiento lento, tardado y tedioso. Su postproceso constaba de obtener los perfiles de las catenarias de cada elemento en un programa de diseño asistido por computadora mediante una serie de comandos, el cual se complica cuando es necesario hacer uno por uno y multiplicarlo por el número de trabes. Además, únicamente se obtenían los perfiles sin conocer realmente si las deflexiones estaban dentro del ELS establecidos en las normas y sin contemplar características físicas como el tipo de material.

El programa ARROW permite realizar un postproceso automatizado y sistematizado que permite reducir tiempo y costos. ARROW es un programa especializado para el tratamiento de información obtenida directamente en campo mediante equipos topográficos, este tipo de programas que no está al alcance del público en general y no ha sido desarrollado en otro lado.

Esta herramienta tecnológica únicamente se puede aplicar para elementos estructurales horizontales confinados, no para elementos voladizos, siendo una limitante al contemplar únicamente un tipo de elemento dentro de un gran espectro existente. También, se consideraron características físicas como el material y la longitud, sin embargo, se podrían agregar otras variables específicas como formas de patines y peraltes, resistencias del material, características químicas de los elementos, etc. Sin embargo, la herramienta contempla lo mínimo para poder calcular una deflexión.

3.3 Documentación y resultados

Para la elaboración de este trabajo se realizó un levantamiento de flechas en elementos estructurales horizontales. La manera en cómo se obtuvo la información y su procesamiento ayudaron a entender las necesidades y problemáticas específicas para desarrollar el programa ARROW.

La metodología empleada en los levantamientos fue la que se propone en este trabajo y se realizó con una estación total Sokkia IM-55 capaz de hacer mediciones sin prisma. A continuación, se muestra evidencia de dicho levantamiento y su procesamiento con el programa ARROW

3.3.1 Levantamiento de flechas en el Centro de Justicia para Mujeres de la Fiscalía de la CDMX en la alcaldía La Magdalena Contreras

Localización

Calle Soledad *s/n* en la colonia San Bernabé Ocotepec, Magdalena Contreras en la CDMX.

Imagen 53. Localización de Centro de Justicia para Mujeres en la CDMX

En total se levantaron 147 trabes de acero distribuidas en los tres pisos del proyecto en su primera etapa de construcción, el tiempo en el que se hizo el levantamiento fue de dos jornadas completas de trabajo en febrero de 2022.

Resultados

La obtención de flechas de este levantamiento se hizo de dos maneras: la primera empleando un programa de dibujo asistido por computadora, y el segundo con el programa ARROW.

Postproceso con CIVILCAD

Previo a importar la información al programa se debe tener un archivo que contengan únicamente los puntos correspondientes a las líneas ideales y otro donde incluya los que pertenecen a las catenarias; previamente será necesario revisar que no exista información duplicada o algún error en los códigos y nombres de los puntos.

Al importar los puntos de las líneas ideales se deben unir cada par con una línea, de esta forma se verificará de forma visual que no exista algún error. Posteriormente, se tendrá que obtener un perfil de dicha línea mediante una serie de comandos indicando aspectos como ejes de proyecto, estaciones, escalas horizontales y verticales.

Posteriormente, es necesario importar los puntos pertenecientes a las catenarias y obtener un segundo perfil mediante la misma serie de comandos mencionados anteriormente, la diferencia es que las estaciones se deben de ingresar de forma manual uno por uno al igual que su elevación, ya que los puntos no se encuentran sobre la línea ideal dibujada y el programa no logra detectarlos ni marcar su cadenamiento.

El tiempo de postproceso de la información fue aproximadamente de día y medio de jornada para la obtención solamente de un plano con el perfil de cada elemento estructural y sus flechas.

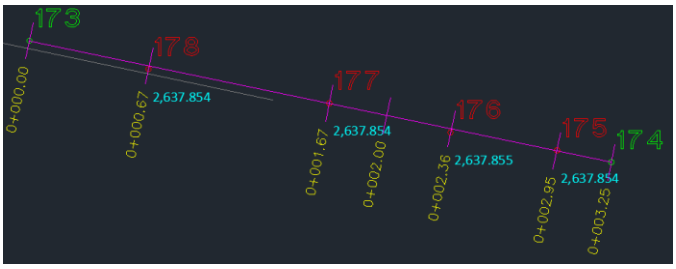


Imagen 54. Ejemplo un eje con cadenamientos, siendo realmente una Línea Ideal con Catenarias en CivilCAD

Finalmente se debe obtener el perfil, en donde el programa calculará un relleno y corte puesto que se toma en cuenta como un eje de vía terrestre, estos conceptos realmente serán las flechas y contraflechas que se buscan obtener.

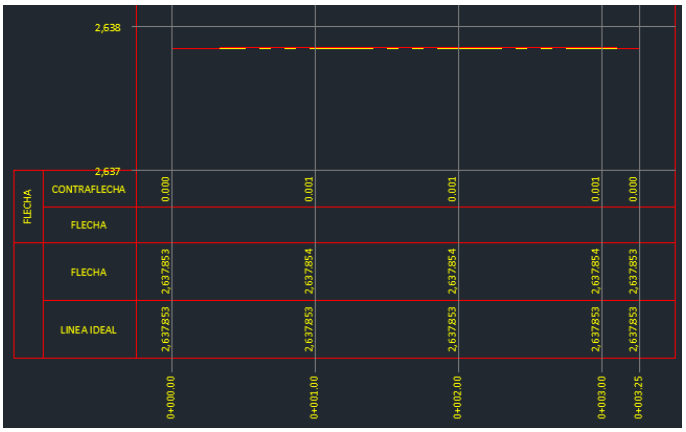


Imagen 55. Ejemplo de obtención de perfil de una trabe con su respectivo cadenamiento indicando las flechas y contraflechas existentes.

Postproceso con ARROW

El tiempo de postproceso de información fue instantáneo para la obtención de un plano con la ubicación de la flecha máxima en cada trabe, y además una hoja de cálculo donde indica si las deflexiones se encuentran dentro del estado límite de servicio en cada elemento y la gráfica del comportamiento de la catenaria.

Al considerar la trabe del ejemplo anterior, este programa logra obtener en el plano la línea ideal e identifica la coordenada exacta donde se encuentra la deflexión máxima.

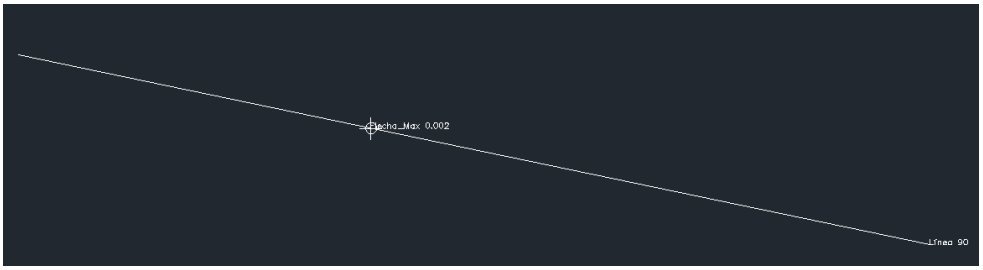


Imagen 56. Ejemplo de trabe con su deflexión máxima

Además, en la hoja de cálculo de salida se indica una tabla con el ELS dependiendo de su longitud y si cumple o no de acuerdo con las NTC CDMX al ser de acero; también, se anexa otra tabla con todas las deflexiones obtenidas de la misma trabe.

LINEA 90				
Distancia (m)	3.279	Deflexión Máx (m)	0.002	
NTC CDMX	Trabes / Vigas	No voladizas	ELS = 0.014	SÍ CUMPLE
		Voladiza	ELS = 0.007	SÍ CUMPLE
	Trabes / Vigas que afecten a elementos NO estructurales	No voladiza	ELS = 0.007	SÍ CUMPLE
		Voladiza	ELS = 0.003	SÍ CUMPLE

Punto	0	1	2	3
Flecha (m)	0.001	0.002	0.002	0.002

Imagen 57. Tabla de datos de los ESL de un elemento estructural con sus respectivas flechas calculadas

En otra hoja del mismo archivo se puede apreciar una gráfica del comportamiento de cada elemento conforme a las flechas obtenidas.

Comparación de postprocesos

CivilCAD	Arrow
✓ Obtención de perfiles a detalle ✓ Procesamiento manual	✓ Procesamiento más rápido ✓ Cuenta con referencias a normas establecidas ✓ Contempla material del elemento ✓ Obtención de perfiles ✓ Procesamiento automático ✓ Obtención de datos tabulares y generación de planos

Tabla 4. Comparación de los postprocesos realizados

Evidencia fotográfica

Se muestra la evidencia del levantamiento de deflexiones realizado con estación total en febrero del 2022 en la alcaldía La Magdalena Contreras.

Imagen 58. Levantamiento de deflexiones de trabes en el sótano del edificio.

Imagen 59. Levantamiento de deflexiones de trabes del cubo de escaleras y elevador de la obra

3.4 Manual del programa ARROW

1. Se entra a la página de *Arrow App*:

   https://site-9000923-386-2002.mystrikingly.com

2. Se descarga el archivo comprimido .rar con el botón de: *Descargar .exe*

Imagen 58. Página principal

3. Se descomprime y ejecuta como administrador el archivo ***ARROW.exe***

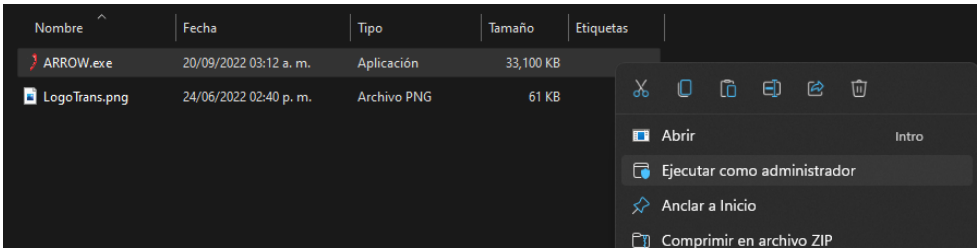


Imagen 59. Ejecutable del programa ARROW

4. Se abrirá una ventana de bienvenida e instrucciones para seleccionar el archivo de entrada, dar en aceptar.

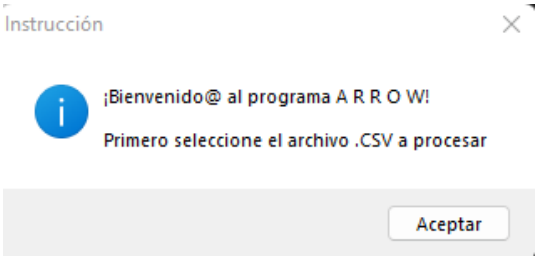


Imagen 60. Ventana de bienvenida

5. Se busca en el explorador de archivos el documento .CSV, se selecciona y dar *click* en abrir.

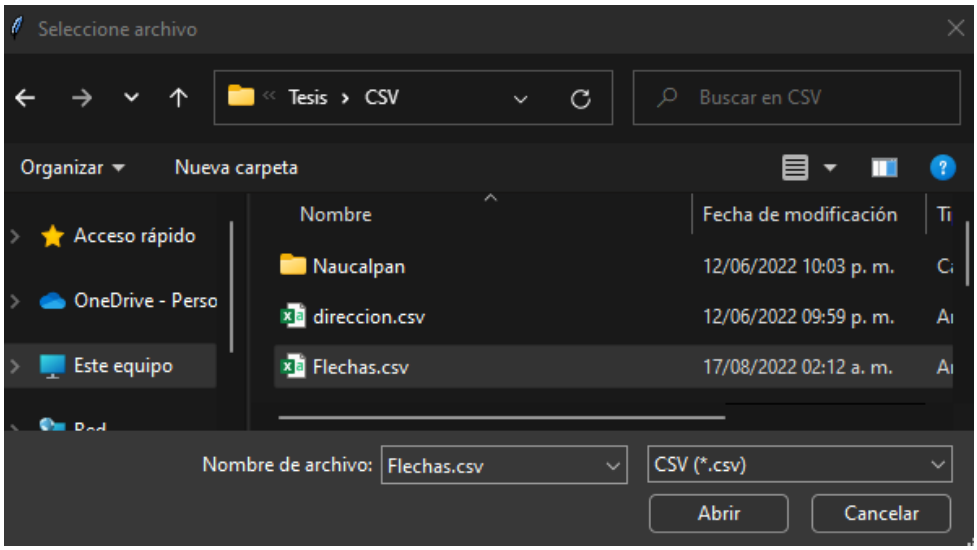


Imagen 6159. Selección de archivo de entrada

6. En la venta principal se ingresa de forma manual los siguientes datos:
    a. Seleccionar el material del elemento (acero, concreto o madera)
    b. Seleccionar el tipo de elemento estructural (losa o trabe/viga)
    c. Indicar el peralte del elemento en metros
    d. Indicar en metros el ancho del elemento
    e. Seleccionar el código perteneciente a la Línea Ideal
    f. Seleccionar e código perteneciente a la Catenaria
    g. Escribir el nombre de los archivos de salida
    h. Indicar en metros si se desea modificar los umbrales del ELS. Si se desea disminuir se debe escribir en negativo el valor, en caso contrario escribir el valor positivo.
    En caso de no querer modificar los umbrales escribir 0 (cero)

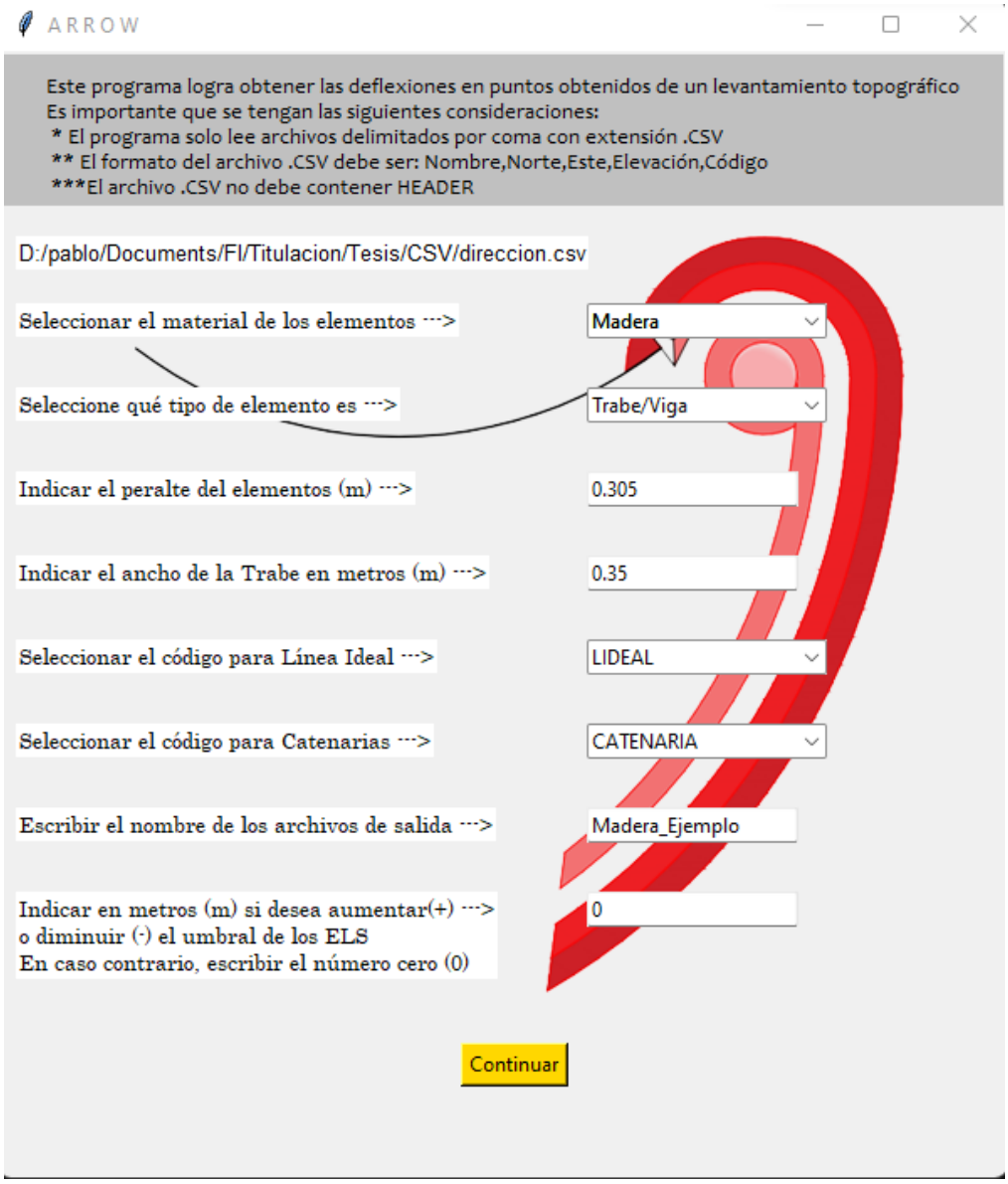


Imagen 62. Ventana principal del programa

7. Dar *click* en el botón de **“CONTINUAR”**, en caso de faltar alguna variable se indicará con una ventana de emergencia.

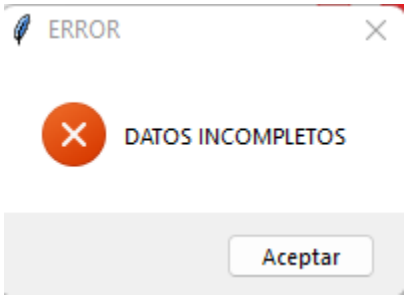


Imagen 63.. Ventana de emergencia

Al terminar de procesar los datos, aparecerá una ventana que indicará que el proceso se realizó correctamente.

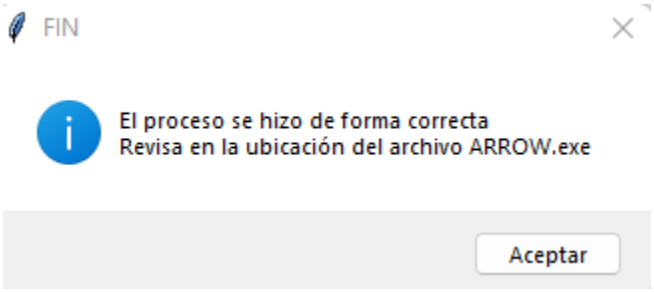


Imagen 64. Ventana de finalización de cálculos

8. Los archivos de salida generados se encontrarán en la misma ruta donde se encuentra el archivo ejecutable ***ARROW.exe***

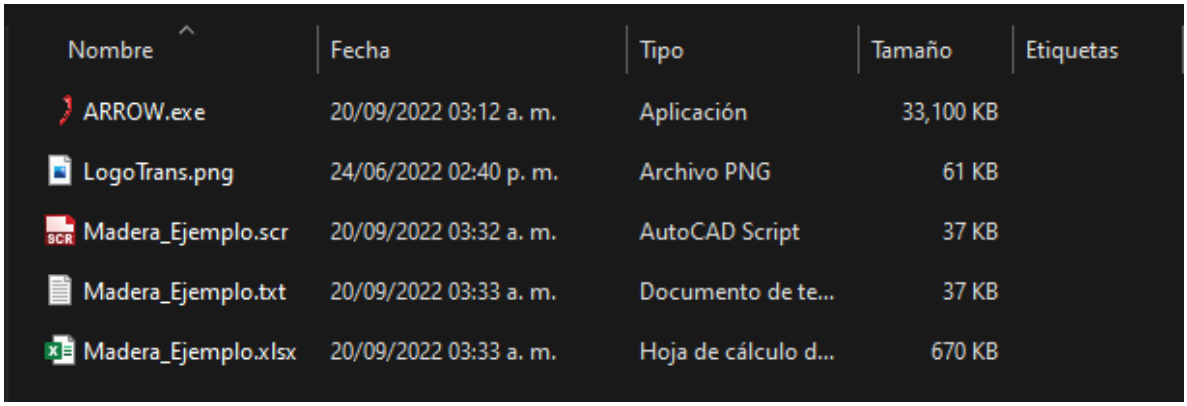


Imagen 65. Localización de archivos de salida

4 CONCLUSIONES

La propuesta y evidencia de la implementación de un protocolo para conocer el desempeño de los elementos de una edificación permite empezar a tener un registro junto con documentación de métodos que pueden difundirse para su aplicación, específicamente en la topografía aplicada a la construcción y al peritaje, para tener certeza en la calidad de la recolección y manejo de datos para asegurar la correcta toma de decisiones para el funcionamiento de las edificaciones durante su vida útil.

Actualmente existe la necesidad de desarrollar herramientas tecnológicas que automaticen el manejo y procesamiento de la información con el fin de optimizar tiempos y costos. La herramienta elaborada en este trabajo además de optimizar deja sentado una base para que pueda seguir evolucionando hasta incluirlo en modelos de información de construcción o también llamados métodos BIM para entender a detalle y de manera específica el comportamiento de una edificación contemplando todas las variables que puedan existir.

i) Anexo Código

```
1.from tkinter import filedialog
2.from tkinter import ttk
3.import tkinter as tk
4.from tkinter import *
5.from PIL import ImageTk, Image
6.from tkinter import messagebox
7.import os
8.import pandas as pd
9.import numpy as np
10.    import math
11.    from openpyxl import Workbook
12.    from openpyxl.chart import LineChart, Refere
  nce
13.    import xlsxwriter
14.    import csv
15.    import sys
16.
17.
18.    intrucciones="Este programa logra obtener
  las deflexiones en puntos obtenidos de un
  levantamiento topográfico\nEs importante que se
  tengan las siguientes consideraciones:\n * El
  programa solo lee archivos delimitados por coma
  con extensión .CSV\n ** El formato del archivo
  .CSV debe ser:
  Nombre,Norte,Este,Elevación,Código\n ***El
  archivo .CSV no debe contener HEADER"
19.
20.    ventana = Tk()
21.    ventana.title("A R R O W")
22.    ventana.geometry("600x670")
23.    #ventana.configure(background='white')
24.
25.    imagen = ImageTk.PhotoImage(Image.open("Logo
  Trans.png"))
26.    imgLabel = Label(ventana, image=imagen).plac
  e(x=50,y=100)
27.
```

```
28.    #Intrucciones
29.    canvas = Canvas(ventana,width=600,height=90,
  bg="silver")
30.    canvas.create_text(300,50,text=intrucciones,
  font = ('Candara 10'))
31.    canvas.pack()
32.
33.    #MENÚ MATERIAL
34.    etiqueta_material = ttk.Label(text="Seleccio
  nar el material de los elementos ---
  >", background='white',font='Century 10')
35.    etiqueta_material.place(x=10,y=150)
36.    lista_material=['Acero', 'Concreto','Madera'
  ]
37.    menu_material = ttk.Combobox(ventana)
38.    menu_material.place(x=350,y=150)
39.    menu_material['values']=lista_material
40.
41.    #TRABE o LOSA
42.    etiqueta_tipo = ttk.Label(text="Seleccione
  qué tipo de elemento es ---
  >",background='white',font='Century 10' )
43.    etiqueta_tipo.place(x=10,y=200)
44.    lista_tipo=['Losa','Trabe/Viga']
45.    menu_tipo = ttk.Combobox(ventana)
46.    menu_tipo.place(x=350,y=200)
47.    menu_tipo['values']=lista_tipo
48.
49.    #PERALTE
50.    etiqueta_peralte = ttk.Label(text='Indicar
  el peralte del elementos (m) ---
  >',background='white',font='Century 10')
51.    etiqueta_peralte.place(x=10,y=250)
52.    caja_peralte = ttk.Entry()
53.    caja_peralte.place(x=350,y=250)
54.
55.    #ANCHO TRABE
56.    etiqueta_ancho = ttk.Label(text='Indicar el
  ancho de la Trabe en metros (m) ---
  >',background='white',font='Century 10')
57.    etiqueta_ancho.place(x=10,y=300)
```

```
58.    caja_ancho = ttk.Entry()
59.    caja_ancho.place(x=350,y=300)
60.
61.
62.
63.    #CÓDIGO LÍNEA IDEAL
64.    etiqueta_LI = ttk.Label(text="Seleccionar el
  código para Línea Ideal ---
  >", background='white',font='Century 10')
65.    etiqueta_LI.place(x=10,y=350)
66.    menu_li = ttk.Combobox(ventana)
67.    menu_li.place(x=350,y=350)
68.
69.
70.    #CÓDIGO CATENARIA
71.    etiqueta_cat = ttk.Label(text="Seleccionar
  el código para Catenarias ---
  >", background='white',font='Century 10')
72.    etiqueta_cat.place(x=10,y=400)
73.    menu_cat = ttk.Combobox(ventana)
74.    menu_cat.place(x=350,y=400)
75.
76.    #NOMBRAR ARCHIVO
77.    etiqueta_nombrearchivo = ttk.Label(text="Esc
  ribir el nombre de los archivos de salida ---
  >",background='white',font='Century 10')
78.    etiqueta_nombrearchivo.place(x=10,y=450)
79.    caja_nombrearchivo = ttk.Entry()
80.    caja_nombrearchivo.place(x=350,y=450)
81.
82.    #MODIFICAR UMBRAL DE ELS
83.    etiqueta_umbral = ttk.Label(text='Indicar en
  metros (m) si desea aumentar(+) --->\no diminuir
  (-) el umbral de los ELS\nEn caso contrario,
  escribir el número cero
  (0)',background='white',font='Century 10')
84.    etiqueta_umbral.place(x=10,y=500)
85.    caja_umbral = ttk.Entry()
86.    caja_umbral.place(x=350,y=500)
87.
```

```
88.    #----------------------------SELECCIÓN
  ARCHIVO
89.    bienvenida = "¡Bienvenido@ al programa A R R
  O W!\n\nPrimero seleccione el archivo .CSV a
  procesar"
90.    messagebox.showinfo(message=bienvenida,title
  ="Instrucción")
91.
92.    archivo=str()
93.    def seleccionar_archivo():  # Función para
  seleccionar archivo .CSV del ordenador
94.        global archivo
95.        archivo = filedialog.askopenfilename(ini
  tialdir="/",title="Seleccione
  archivo", filetypes=(("CSV", "*.csv"), ("Texto",
   "*.txt"),("all files", "*.*")))
96.
97.    seleccionar_archivo()
98.    #RUTA
99.    etiqueta_ruta = ttk.Label(text=archivo,backg
  round='white',font='Arial 10')
100.   etiqueta_ruta.place(x=10,y=110)
101.
102.
103.
104.
105.   #----------------------------- DATAFRAMES
  DE L IDEAL Y CATENARIA
106.   csv = pd.read_csv(archivo, header=None)  #
  Leemos archivo CSV
107.   # Agregamos HEADER al archivo CSV
108.   csv.columns = ["Nombre", "N", "E", "Z", "CÓD
  IGO"]
109.   #print(csv)
110.   #Leemos los códigos que existen en el csv
111.   codigosA = str(csv['CÓDIGO'].unique()).repla
  ce("[", "").replace("]","").replace("'","" )
112.   #print(codigosA)
113.
114.
115.   menu_li['values']=codigosA
```

```
116.   #print(menu_li,"MENU LINEA IDEAL")
117.   menu_cat['values']=codigosA
118.   #print(menu_cat,"MENU CATENARIA")
119.
120.   def calc():
121.
122.       csv = pd.read_csv(archivo, header=None)
123.       csv.columns = ["Nombre", "N", "E", "Z",
  "CÓDIGO"]
124.       codigosA = str(csv['CÓDIGO'].unique()).r
  eplace("[", "").replace("]","").replace("'","" )
125.       lideal_1 = str(menu_li.get())
126.   #    print("SELECCIONÓ :",lideal_1," para la
  L IDEAL")
127.       catenarias_1 = str(menu_cat.get())
128.   #    print("SELECCIONÓ :",catenarias_1,"
  para CATENARIA")
129.       #-------------------------------
  CREAMOS ARCHIVOS DE SALIDA
130.       archivo_out = str(caja_nombrearchivo.get
  ())
131.   #    print("ESCRIBIÓ : ",archivo_out," para
  NOMBRE")
132.       nom_txt=archivo_out+'.txt'
133.       nom_scr=archivo_out+'.scr'
134.       nom_excel=archivo_out+'.xlsx'
135.       #--------------------------- SELECCIÓN
  MATERIAL
136.       material = str(menu_material.get())
137.   #    print("SELECCIONÓ EL MATERIAL DE:
  ",material)
138.
139.
140.
141.       # Creamos dataframe de Líneas Ideales
142.       df_lideal = csv.loc[csv['CÓDIGO'] == lid
  eal_1]
143.       df_Icoord = df_lideal[['N','E','Z']]
144.       #print("COORDENADAS L IDEALES : ",
  df_Icoord)
```

```
145.        # Creamos otro DF con las coordeandas de
  las Líneas Ideales
146.        df_catenarias = csv.loc[csv['CÓDIGO'] ==
   catenarias_1]
147.        df_Ccoord = df_catenarias[['N', 'E', 'Z'
  ]]
148.        #print("COORDENADAS CATENARIAS : ",
  df_Ccoord)
149.        #------------------------------
  EXCEL
150.        cad_txt = open(nom_txt,'w')
151.        cad_scr = open(nom_scr,'w')
152.        excel = xlsxwriter.Workbook(nom_excel)
153.        hoja = excel.add_worksheet('Flechas')
154.        hoja2 = excel.add_worksheet("Gráficas")
155.         #------------------------------ FORMATO
  A ARCHIVO EXCEL
156.            #Tamaño Columnas
157.        hoja.set_column('A:A',13.00)
158.        hoja.set_column('B:B',28.00)
159.        hoja.set_column('C:C',55.00)
160.        hoja.set_column('D:D',27.00)
161.        hoja.set_column('E:E',14.00)
162.        hoja.set_column('F:F',2.00)
163.        hoja.set_column('G:G',12.00)
164.        #Formatos Texto
165.        titulo00 = excel.add_format({'bold':1,'b
  order':1,'align':'center','fg_color':'#FFC000'})
166.        t1 = excel.add_format({'align':'center'}
  ).set_text_wrap()
167.        t2 = excel.add_format({'align':'left'}).
  set_text_wrap()
168.        t5 = excel.add_format({'align':'vcenter'
  ,'border':1,})
169.        t5.set_text_wrap()
170.
171.
172.        while True:
173.
174.            if len(material)!=0 and len(lideal_1
  )!=0 and len(catenarias_1)!=0 and len(caja_nombr
```

```
  earchivo.get())!=0 and len(caja_ancho.get())!=0
  and len(caja_umbral.get())!=0:
175.                messagebox.showinfo(message="El
  proceso se hizo de forma correcta\nRevisa en la
  ubicación del archivo ARROW.exe",title="FIN")
176.            else:
177.                messagebox.showerror("ERROR","DA
  TOS INCOMPLETOS")
178.
179.    #------------------------------ ANCHO
  TRABES
180.            trabe = float(caja_ancho.get())
181.            #print("EL ANCHO DE LA TRABE ES DE:
  ",trabe)
182.            ancho = trabe/2
183.            linea = 0
184.    #------------------------ CONTADORES
  EXCEL
185.    #------------------------ HOJA 1
186.            titulo = 1
187.            renglon2 = 2
188.            renglon3 = 3
189.            renglon4 = 4
190.            renglon5 = 5
191.            renglon6 = 6
192.            renglon7 = 7
193.            renglon8 = 8
194.            renglon9 = 9
195.            renglon10 = 10
196.            renglon11 = 11
197.    #------------------------ HOJA 2
198.            h2titulo = 1
199.            h2renglon2 = 2
200.            h2renglon3 = 3
201.            h2renglon4 = 4
202.            h2renglon5 = 5
203.            h2renglon6 = 6
204.            h2renglon7 = 7
205.            h2renglon8 = 8
206.            h2renglon9 = 9
207.            h2renglon10 = 10
```

```
208.           h2renglon11 = 11
209.   #------------------------ CÁLCULOS AZIMUT
210.           for i in range(0, len(df_Icoord), 2)
  :
211.               # Obtenemos la distancia de la
  Linea Ideal
212.               pot1 = (df_Icoord.iloc[i+1,1]-
  df_Icoord.iloc[i,1])**2
213.               pot2 = (df_Icoord.iloc[i+1,0]-
  df_Icoord.iloc[i,0])**2
214.               sumapots = pot1 + pot2
215.               raizpots = sumapots ** 0.5
216.               DistABB = raizpots #Distacia
  plano 2D
217.               #DistABB =
  math.dist(df_Icoord.iloc[i+1],
  df_Icoord.iloc[i]) #Distancia 3D
218.               DistAB = round(DistABB,3)
219.               #print("La distancia de la
  línea 1 es de: ",DistAB)
220.               # Obtenemos la Hipotenusa que
  se forma de un triángulo ABC
221.               hip = math.sqrt(DistAB**2 +
  ancho**2)
222.               #print("La HIPOTENUSA del
  triángulo es de : ",hip)
223.               # Diferencia de X's para
  aplicar ecuación Tangente
224.               deltax = df_Icoord.iloc[i+1, 1]
   - df_Icoord.iloc[i, 1]
225.               #print("Diferencia de X:
  ",deltax)
226.               # Diferencia de Y's para
  aplicar ecuación Tangente
227.               deltay = df_Icoord.iloc[i+1, 0]
   - df_Icoord.iloc[i, 0]
228.               #print("Diferencia de Y:
  ",deltay)
229.               # Obtenemos el Azimut de la
  Línea Ideal
230.
```

```
231.     #------------------------- PRIMER
  CUADRANTE
232.                 if deltax > 0 and deltay > 0:
233.
234.                     AzimutAB = math.degrees(mat
  h.atan(deltax/deltay))
235.                     #print("El AZIMUT de la
  LÍNEA IDEAL es de: ",AzimutAB)
236.                     #print("Esta línea se
  encuentra en 1° CUADRANTE")
237.                     # Límite 1
238.                     alfa = AzimutAB - 90
239.                     DistXLim1 = math.sin(math.r
  adians(alfa))*ancho
240.                     DistYLim1 = math.cos(math.r
  adians(alfa))*ancho
241.                     Lim1 = [df_Icoord.iloc[i, 0
  ]+DistYLim1,
242.                             df_Icoord.iloc[i, 1
  ]+DistXLim1, df_Icoord.iloc[i, 2]]
243.   #                  print("El Límite 1 es :
  ",Lim1)
244.
245.
246.                     # Límite 2
247.                     Lim2 = [df_Icoord.iloc[i, 0
  ]-DistYLim1,
248.                             df_Icoord.iloc[i, 1
  ]-DistXLim1, df_Icoord.iloc[i, 2]]
249.    #                  print("El Límite 2 es :
  ",Lim2)
250.
251.                     # Límite 3
252.                     alfa2 = math.degrees(math.a
  tan(ancho/DistAB))
253.                     AzimutALim3 = AzimutAB -
  alfa2
254.                     DistXLim3 = math.sin(math.r
  adians(AzimutALim3))*hip
255.                     DistYLim3 = math.cos(math.r
  adians(AzimutALim3))*hip
```

```
256.                     if AzimutALim3 < 0:
257.                         DistXLim3 = DistXLim3 *
  -1
258.                         Lim3 = [df_Icoord.iloc[
  i, 0]+DistYLim3,
259.                                 df_Icoord.iloc[
  i, 1]+DistXLim3, df_Icoord.iloc[i+1, 2]]
260.                     elif AzimutALim3 > 0:
261.                         Lim3 = [df_Icoord.iloc[
  i, 0]+DistYLim3,
262.                                 df_Icoord.iloc[
  i, 1]+DistXLim3, df_Icoord.iloc[i+1, 2]]
263.     #                print("El Límite 3 es :
  ",Lim3)
264.
265.                     # Límite 4
266.                     alfa3 = alfa2
267.                     AzimutALim4 = AzimutAB +
  alfa3
268.                     DistXLim4 = math.sin(math.r
  adians(AzimutALim4))*hip
269.                     DistYLim4 = math.cos(math.r
  adians(AzimutALim4))*hip
270.                     if AzimutALim4 > 90:
271.                         DistYLim4 = DistYLim4 *
  -1
272.                         Lim4 = [df_Icoord.iloc[
  i, 0]+DistYLim4,
273.                                 df_Icoord.iloc[
  i, 1]+DistXLim4, df_Icoord.iloc[i+1, 2]]
274.                     elif AzimutALim4 < 90:
275.                         Lim4 = [df_Icoord.iloc[
  i, 0]+DistYLim4,
276.             df_Icoord.iloc[i, 1]+DistXLim4, df_
  Icoord.iloc[i+1, 2]]
277.      ##                print("El Límite 4 es :
  ",Lim4)
278.    #---------------------------- SEGUNDO
  CUADRANTE
279.                 elif deltax > 0 and deltay < 0:
```

```
                    AzimutAB = math.degrees(mat
  h.atan(deltax/deltay))+180
                    #print("El AZIMUT de la
  LÍNEA IDEAL es de: ",AzimutAB)
                    #print("Esta línea se
  encuentra en 2° CUADRANTE")
                    # Límite 1
                    alfa = AzimutAB - 90
                    DistXLim1 = math.sin(math.r
  adians(alfa))*ancho
                    DistYLim1 = math.cos(math.r
  adians(alfa))*ancho
                    Lim1 = [df_Icoord.iloc[i, 0
  ]+DistYLim1,
                            df_Icoord.iloc[i, 1
  ]+DistXLim1, df_Icoord.iloc[i, 2]]
                    #print("El Límite 1 es :
  ",Lim1)
                    # Límite 2
                    Lim2 = [df_Icoord.iloc[i, 0
  ]-DistYLim1,
                            df_Icoord.iloc[i, 1
  ]-DistXLim1, df_Icoord.iloc[i, 2]]
                    #print("El Límite 2 es :
  ",Lim2)
                    # Límite 3
                    alfa2 = math.degrees(math.a
  tan(ancho/DistAB))
                    AzimutALim3 = AzimutAB -
  alfa2
                    DistXLim3 = math.sin(math.r
  adians(AzimutALim3))*hip
                    DistYLim3 = math.cos(math.r
  adians(AzimutALim3))*hip
                    if AzimutALim3 < 90:
                        DistYLim3 = DistYLim3 *
  -1
                        Lim3 = [df_Icoord.iloc[
  i, 0]+DistYLim3,
                                df_Icoord.iloc[
  i, 1]+DistXLim3, df_Icoord.iloc[i+1, 2]]
```

```
303.                     elif AzimutALim3 > 90:
304.                         Lim3 = [df_Icoord.iloc[
  i, 0]+DistYLim3,
305.                                 df_Icoord.iloc[
  i, 1]+DistXLim3, df_Icoord.iloc[i+1, 2]]
306.                     #print("El Límite 3 es :
  ",Lim3)
307.                     # Límite 4
308.                     alfa3 = alfa2
309.                     AzimutALim4 = AzimutAB +
  alfa3
310.                     DistXLim4 = math.sin(math.r
  adians(AzimutALim4))*hip
311.                     DistYLim4 = math.cos(math.r
  adians(AzimutALim4))*hip
312.                     if AzimutALim4 > 180:
313.                         DistXLim4 = DistXLim4 *
  -1
314.                         Lim4 = [df_Icoord.iloc[
  i, 0]+DistYLim4,
315.                                 df_Icoord.iloc[
  i, 1]+DistXLim4, df_Icoord.iloc[i+1, 2]]
316.                     elif AzimutALim4 < 180:
317.                         Lim4 = [df_Icoord.iloc[
  i, 0]+DistYLim4,
318.                                 df_Icoord.iloc[
  i, 1]+DistXLim4, df_Icoord.iloc[i+1, 2]]
319.                     #print("El Límite 4 es :
  ",Lim4)
320.
321.    #---------------------------- TERCER
  CUADRANTE
322.                 elif deltax < 0 and deltay < 0:
323.
324.                     AzimutAB = math.degrees(mat
  h.atan(deltax/deltay))+180
325.                     #print("El AZIMUT de la
  LÍNEA IDEAL es de: ",AzimutAB)
326.                     #print("Esta línea se
  encuentra en 3° CUADRANTE")
327.                     # Límite 1
```

```
328.                     alfa = AzimutAB - 90
329.                     DistXLim1 = math.sin(math.r
  adians(alfa))*ancho
330.                     DistYLim1 = math.cos(math.r
  adians(alfa))*ancho
331.                     Lim1 = [df_Icoord.iloc[i, 0
  ]+DistYLim1,
332.                             df_Icoord.iloc[i, 1
  ]+DistXLim1, df_Icoord.iloc[i, 2]]
333.                     #print("El Límite 1 es :
  ",Lim1)
334.
335.                     # Límite 2
336.                     Lim2 = [df_Icoord.iloc[i, 0
  ]-DistYLim1,
337.                             df_Icoord.iloc[i, 1
  ]-DistXLim1, df_Icoord.iloc[i, 2]]
338.                     #print("El Límite 2 es :
  ",Lim2)
339.
340.                     # Límite 3
341.                     alfa2 = math.degrees(math.a
  tan(ancho/DistAB))
342.                     AzimutALim3 = AzimutAB -
  alfa2
343.                     DistXLim3 = math.sin(math.r
  adians(AzimutALim3))*hip
344.                     DistYLim3 = math.cos(math.r
  adians(AzimutALim3))*hip
345.                     if AzimutALim3 > 270:
346.                         DistYLim3 = DistYLim3 *
  -1
347.                         Lim3 = [df_Icoord.iloc[
  i, 0]+DistYLim3,
348.                                 df_Icoord.iloc[
  i, 1]+DistXLim3, df_Icoord.iloc[i+1, 2]]
349.                     elif AzimutALim3 < 270:
350.                         Lim3 = [df_Icoord.iloc[
  i, 0]+DistYLim3,
351.                                 df_Icoord.iloc[
  i, 1]+DistXLim3, df_Icoord.iloc[i+1, 2]]
```

```
352.                     #print("El Límite 3 es :
  ",Lim3)
353.
354.                     # Límite 4
355.                     alfa3 = alfa2
356.                     AzimutALim4 = AzimutAB +
  alfa3
357.                     DistXLim4 = math.sin(math.r
  adians(AzimutALim4))*hip
358.                     DistYLim4 = math.cos(math.r
  adians(AzimutALim4))*hip
359.                     if AzimutALim4 < 180:
360.                         DistXLim4 = DistXLim4 *
  -1
361.                         Lim4 = [df_Icoord.iloc[
  i, 0]+DistYLim4,
362.                                 df_Icoord.iloc[
  i, 1]+DistXLim4, df_Icoord.iloc[i+1, 2]]
363.                     elif AzimutALim4 > 180:
364.                         Lim4 = [df_Icoord.iloc[
  i, 0]+DistYLim4,
365.                                 df_Icoord.iloc[
  i, 1]+DistXLim4, df_Icoord.iloc[i+1, 2]]
366.                     #print("El Límite 4 es :
  ",Lim4)
367.
368.    #------------------------ CUARTO CUADRANTE
369.                 elif deltax < 0 and deltay > 0:
370.
371.                     AzimutAB = 360+math.degrees
  (math.atan(deltax/deltay))
372.                     #print("El AZIMUT de la
  LÍNEA IDEAL es de: ",AzimutAB)
373.                     #print("Esta línea se
  encuentra en 4° CUADRANTE")
374.                     # Límite 1
375.                     alfa = AzimutAB - 90
376.                     DistXLim1 = math.sin(math.r
  adians(alfa))*ancho
377.                     DistYLim1 = math.cos(math.r
  adians(alfa))*ancho
```

```
                    Lim1 = [df_Icoord.iloc[i, 0]+DistYLim1,
                                df_Icoord.iloc[i, 1]+DistXLim1, df_Icoord.iloc[i, 2]]
                    #print("El Límite 1 es : ",Lim1)

                    # Límite 2
                    Lim2 = [df_Icoord.iloc[i, 0]-DistYLim1,
                                df_Icoord.iloc[i, 1]-DistXLim1, df_Icoord.iloc[i, 2]]
                    #print("El Límite 2 es : ",Lim2)

                    # Límite 3
                    alfa2 = math.degrees(math.atan(ancho/DistAB))
                    AzimutALim3 = AzimutAB - alfa2
                    DistXLim3 = math.sin(math.radians(AzimutALim3))*hip
                    DistYLim3 = math.cos(math.radians(AzimutALim3))*hip
                    if AzimutALim3 < 270:
                        DistYLim3 = DistYLim3 * -1
                        Lim3 = [df_Icoord.iloc[i, 0]+DistYLim3,
                                    df_Icoord.iloc[i, 1]+DistXLim3, df_Icoord.iloc[i+1, 2]]
                    elif AzimutALim3 > 270:
                        Lim3 = [df_Icoord.iloc[i, 0]+DistYLim3,
                                    df_Icoord.iloc[i, 1]+DistXLim3, df_Icoord.iloc[i+1, 2]]
                    #print("El Límite 3 es : ",Lim3)

                    # Límite 4
                    alfa3 = alfa2
```

```
403.                     AzimutALim4 = AzimutAB +
  alfa3
404.                     DistXLim4 = math.sin(math.r
  adians(AzimutALim4))*hip
405.                     DistYLim4 = math.cos(math.r
  adians(AzimutALim4))*hip
406.                     if AzimutALim4 > 360:
407.                         DistXLim4 = DistXLim4 *
  -1
408.                         Lim4 = [df_Icoord.iloc[
  i, 0]+DistYLim4,
409.                                 df_Icoord.iloc[
  i, 1]+DistXLim4, df_Icoord.iloc[i+1, 2]]
410.                     elif AzimutALim4 < 360:
411.                         Lim4 = [df_Icoord.iloc[
  i, 0]+DistYLim4,
412.                                 df_Icoord.iloc[
  i, 1]+DistXLim4, df_Icoord.iloc[i+1, 2]]
413.                     #print("El Límite 4 es :
  ",Lim4)
414.    #------------------------
  LÍMITES
415.                # Obtendremos las coordenas
  límites en X,Y,Z
416.                 xmin = min(Lim1[1], Lim2[1], Li
  m3[1], Lim4[1])
417.                 xmax = max(Lim1[1], Lim2[1], Li
  m3[1], Lim4[1])
418.                 ymin = min(Lim1[0], Lim2[0], Li
  m3[0], Lim4[0])
419.                 ymax = max(Lim1[0], Lim2[0], Li
  m3[0], Lim4[0])
420.                 zmin = min(Lim1[2], Lim2[2], Li
  m3[2], Lim4[2])-0.2
421.                 zmax = max(Lim1[2], Lim2[2], Li
  m3[2], Lim4[2])+0.2
422.                 #print("X Min: ",xmin," X Max:
  ",xmax)
423.                 ##print("Y Min: ",ymin," Y Max:
  ",ymax)
```

```
424.                 #print("Z Min: ",zmin," Z Max:
  ",zmax)
425.    #-------------------------- ECUACIÓN DEL
  PLANO
426.                 # Obtenemos el vector director
  de AB
427.                 v = [df_Icoord.iloc[i+1, 1]-
  df_Icoord.iloc[i, 1], df_Icoord.iloc[i+1,0]-
  df_Icoord.iloc[i, 0], df_Icoord.iloc[i+1, 2]-
  df_Icoord.iloc[i, 2]]
428.
429.                 # Ahora se calcula el vector
  CLA, siendo C la "esquina del plano" obtenida
  anteriormente
430.                 CLA = [df_Icoord.iloc[i, 1]-
  Lim1[1], df_Icoord.iloc[i, 0] -
  Lim1[0], df_Icoord.iloc[i, 2]-Lim1[2]]
431.
432.                 # Hacemos producto cruz para
  obtener vector normal al plano, siendo CLA x V
433.                 n = np.cross(CLA, v)
434.                 #print("PRODUCTO CRUZ DE
  VECTORES: ",n)
435.                 plano = [n[0], n[1], n[2], -
  (n[0]*df_Icoord.iloc[i, 1])-
  (n[1]*df_Icoord.iloc[i, 0])-
  (n[2]*df_Icoord.iloc[i, 2])]
436.                 #print("LA ECUACIÓN DEL PLANO
  ES:  ", plano)
437.    #---------------------------- CÁLCULO
  FLECHAS
438.                 aa =[]
439.                 for j in range(0, len(df_Ccoord
  )):
440.                     if xmin <= df_Ccoord.iloc[j
  , 1] <=xmax and ymin <= df_Ccoord.iloc[j, 0] <=y
  max and zmin <= df_Ccoord.iloc[j, 2] <=zmax:
441.                         numerador = abs((plano[
  0]*df_Ccoord.iloc[j, 1]) + (
```

```
                        plano[1]*df_Ccoord.iloc
  [j, 0]) + (plano[2]*df_Ccoord.iloc[j, 2]) +
  plano[3])
                        denominador = math.sqrt
  (
                            plano[0]**2 +
  plano[1]**2 + plano[2]**2)
                        flecha2 = numerador/den
  ominador
                        flecha = round(flecha2,
  3)
                        #print("FLECHA        :
       ",flecha)
                        bb = [flecha,df_Ccoord.
  iloc[j,0],df_Ccoord.iloc[j,1],df_Ccoord.iloc[j,2
  ]]
                        aa.append(bb)
    #--------------------------- FLECHA
  MÁXIMA
                for m in range(0,len(aa)):
                    flechamax = max(aa[:])
                cc = flechamax[1:]
                flechamaxpoint = flechamax[0]
                flecharedon = round(flechamaxpo
  int,3)
                #print("LA FLECHA MÁXIMA ES:
  ",flecharedon)

    #--------------------------- Calculamos un
  punto C sobre la LIDEAL que contiene la Flecha
  Máxima
                DistAC = math.dist(df_Icoord.il
  oc[i],cc)
     #--------------------------- AZIMUT A
  CADA PUNTO DE CATENARIAS
                #Diferencia de X's para aplicar
  ecuación Tangente
                deltax2 = cc[1] -
  df_Icoord.iloc[i, 1]
                # Diferencia de Y's para
  aplicar ecuación Tangente
```

```
                deltay2 = cc[0] -
  df_Icoord.iloc[i, 0]

    #--------------------------------- PRIMER
  CUADRANTE
                if deltax2 > 0 and deltay2 > 0:

                    AzimutAC = math.degrees(mat
  h.atan(deltax2/deltay2))
    #--------------------------------- SEGUNDO
  CUADRANTE
                elif deltax2 > 0 and deltay2 <
  0:
                    AzimutAC = math.degrees(mat
  h.atan(deltax2/deltay2))+180
    #--------------------------------- TERCER
  CUADRANTE
                elif deltax2 < 0 and deltay2 <
  0:
                    AzimutAC = math.degrees(mat
  h.atan(deltax2/deltay2))+180
    #--------------------------------- CUARTO
  CUADRANTE
                elif deltax2 < 0 and deltay2 >
  0:
                    AzimutAC = 360+math.degrees
  (math.atan(deltax2/deltay2))

                beta = abs(AzimutAC-AzimutAB)
                cosbeta = math.cos(math.radians
  (beta))
                ca = DistAC * cosbeta
                xc = math.sin(math.radians(Azim
  utAB))*ca
                yc = math.cos(math.radians(Azim
  utAB))*ca
                C = [df_Icoord.iloc[i,1] +
  xc, df_Icoord.iloc[i,0] + yc,cc[2]]
                #print("La Flecha Máxima se
  encuentra en las
  coordenadas:  ",C)
```

```
486.
487.                 #print("La Ecuación del Plano
  es: ", plano[0],"x + ",plano[1],"y +
  ",plano[2],"z + ",plano[3]," = 0")
488.                 linea = linea + 1
489.                 #print("\nDistancia de la Línea
  ideal", linea, " es de:", DistAB)
490.                 #print("El AZIMUT de la Línea
  Ideal", linea," es de:", AzimutAB, " GRADOS")
491.
492.     #-----------------------------------------
  ---------- ESCRIBIMOS EN EXCEL ---------------
  ----------------------------------
493.                 tipo = str(menu_tipo.get())
494.                 peralte = float(caja_peralte.ge
  t())
495.                 if tipo == 'Losa':
496.                     num_ptos = round(DistAB
  / (8*peralte),1)
497.
498.                 elif tipo == 'Trabe/Viga' :
499.                     num_ptos = round(DistAB
  / (4*peralte),1)
500.
501.
502.
503.     #---------------------------- GRÁFICAS
504.
505.
506.                 celda_tih2 ='A'+str(h2titulo)+'
  :H'+str(h2titulo)
507.                 texto_tih2 = 'LINEA
  '+str(linea)
508.                 hoja2.merge_range(celda_tih2,te
  xto_tih2,titulo00)
509.                 for p in range(0,len(aa),1):
510.                     hoja2.write(renglon2,0,'Pun
  to',t5)
511.                     hoja2.write(renglon3,0,'Fle
  cha (m)',t5)
```

```
                    hoja2.write(renglon2,1+p,p,
  t5)
                    hoja2.write(renglon3,1+p,aa
  [p][0]*-1,t5)
                    chart = excel.add_chart({'t
  ype':'line'})
                    chart.add_series({'values':
   ['Gráficas',renglon3,1,renglon3,len(aa)], 'line
  ':{'color':'red'},})
                    chart.set_y_axis({'name':'F
  lecha','num_font':{'arial black':True},})
                    chart.set_x_axis({'minor_un
  it':0})
                    chart.set_size({'width':445
  ,'height':159})
                    chart.set_title({'name':tex
  to_tih2, 'num_font':{'name':'arial
  black','size':12},})
                    chart.set_legend({'position
  ':'none'},)
                    grafpos = "I"+str(h2titulo)
                    hoja2.insert_chart(grafpos,
   chart)

                h2titulo = h2titulo + 7
                h2renglon2 = h2renglon2 + 7
                h2renglon3 = h2renglon3 + 7
                h2renglon4 = h2renglon4 + 7
                h2renglon5 = h2renglon5 + 7
                h2renglon6 = h2renglon6 + 7
                h2renglon7 = h2renglon7 + 7
                h2renglon8 = h2renglon8 + 7
                h2renglon9 = h2renglon9 + 7
                h2renglon10 = h2renglon10 + 7
                h2renglon11 = h2renglon11 + 7


    #------------------------------ ESTADOS
  LÍMITE DE SERVICIO
                #Se calcularán los Estados
  Límites de Servicio de acuerdo con el material
```

```
539.                 umbral = float(caja_umbral.get(
  ))
540.                 if material == 'Acero': #ACERO
541.                     els1_mx = round(DistAB/240,
  3) + umbral
542.                     els2_mx = round(DistAB/480,
  3) + umbral
543.                     els3_mx = round(DistAB/480,
  3) + umbral
544.                     els4_mx = round(DistAB/960,
  3) + umbral
545.                     # print("ELS 1: ",els1_mx)
546.                     # print("ELS 2: ",els2_mx)
547.                     # print("ELS 3: ",els3_mx)
548.                     # print("ELS 4: ",els4_mx)
549.
550.
551.                     celda_titulo ='A'+str(titul
  o)+':F'+str(titulo)
552.                     texto_titulo ='LINEA
  '+str(linea)
553.                     hoja.merge_range(celda_titu
  lo,texto_titulo,titulo00)
554.                     celda_distancia ='A'+str(re
  nglon2)
555.                     hoja.write(celda_distancia,
  'Distancia (m)',t2)
556.                     celda_distancia2 = 'B'+str(
  renglon2)
557.                     hoja.write(celda_distancia2
  ,DistAB,t1)
558.                     celda_max = 'C'+str(renglon
  2)
559.                     hoja.write(celda_max,'Defle
  xión Máx (m)',t2)
560.                     celda_max2 = 'D'+str(renglo
  n2)
561.                     hoja.write(celda_max2,flech
  aredon,t1)
562.                     celda_espacio1 = 'A'+str(re
  nglon3)+':F'+str(renglon3)
```

```
563.                     hoja.merge_range(celda_espa
  cio1,'')
564.
565.                     celda_ntc = 'A'+str(renglon
  4)+':A'+str(renglon7)
566.                     hoja.merge_range(celda_ntc,
  'NTC CDMX',t5)
567.                     ntc1 = 'B'+str(renglon4)+':
  B'+str(renglon5)
568.                     hoja.merge_range(ntc1,'Trab
  es / Vigas',t5)
569.                     ntc2 = 'B'+str(renglon6)+':
  B'+str(renglon7)
570.                     hoja.merge_range(ntc2,'Trab
  es / Vigas que afecten a elementos NO
  estructurales',t5)
571.
572.                     c4 = 'C'+str(renglon4)
573.                     hoja.write(c4,'No
  voladizas',t5)
574.                     c5 = 'C'+str(renglon5)
575.                     hoja.write(c5,'Voladiza',t5
  )
576.                     c6 = 'C'+str(renglon6)
577.                     hoja.write(c6,'No
  voladiza',t5)
578.                     c7 = 'C'+str(renglon7)
579.                     hoja.write(c7,'Voladiza',t5
  )
580.
581.                     d4_pos = 'D'+str(renglon4)
582.                     d4_nombre = 'ELS =
  '+str(els1_mx)
583.                     hoja.write(d4_pos,d4_nombre
  ,t5)
584.                     d5_pos = 'D'+str(renglon5)
585.                     d5_nombre = 'ELS =
  '+str(els2_mx)
586.                     hoja.write(d5_pos,d5_nombre
  ,t5)
587.                     d6_pos = 'D'+str(renglon6)
```

```
588.                    d6_nombre = 'ELS =
  '+str(els3_mx)
589.                    hoja.write(d6_pos,d6_nombre
  ,t5)
590.                    d7_pos = 'D'+str(renglon7)
591.                    d7_nombre = 'ELS =
  '+str(els4_mx)
592.                    hoja.write(d7_pos,d7_nombre
  ,t5)
593.
594.                    if flecharedon <= els1_mx:
595.                        celda_cumple4 = 'E'+str
  (renglon4)
596.                        hoja.write(celda_cumple
  4,'SÍ CUMPLE',t5)
597.                    elif flecharedon > els1_mx:
598.                        celda_cumple4 = 'E'+str
  (renglon4)
599.                        hoja.write(celda_cumple
  4,'NO CUMPLE',t5)
600.                    if flecharedon <= els2_mx:
601.                        celda_cumple5 = 'E'+str
  (renglon5)
602.                        hoja.write(celda_cumple
  5,'NO APLICA',t5)
603.                    elif flecharedon > els2_mx:
604.                        celda_cumple5 = 'E'+str
  (renglon5)
605.                        hoja.write(celda_cumple
  5,'NO APLICA',t5)
606.                    if flecharedon <= els3_mx:
607.                        celda_cumple6 = 'E'+str
  (renglon6)
608.                        hoja.write(celda_cumple
  6,'SÍ CUMPLE',t5)
609.                    elif flecharedon > els3_mx:
610.                        celda_cumple6 = 'E'+str
  (renglon6)
611.                        hoja.write(celda_cumple
  6,'NO CUMPLE',t5)
612.                    if flecharedon <= els4_mx:
```

```
613.                         celda_cumple7 = 'E'+str
  (renglon7)
614.                         hoja.write(celda_cumple
  7,'NO APLICA',t5)
615.                     elif flecharedon > els4_mx:
616.                         celda_cumple7 = 'E'+str
  (renglon7)
617.                         hoja.write(celda_cumple
  7,'NO APLICA',t5)
618.
619.                     for q in range(0,len(aa),1)
  :
620.                         hoja.write(renglon2,6,'
  Punto',t5)
621.                         hoja.write(renglon3,6,'
  Flecha (m)',t5)
622.                         hoja.write(renglon2,7+q
  ,q,t5)
623.                         hoja.write(renglon3,7+q
  ,aa[q][0],t5)
624.                     if len(aa) >= num_ptos:
625.                         hoja.write(renglon4,6,'
  El núm. de puntos por elemento es el
  recomendado')
626.                         hoja.write(renglon5,6,'
  Num de puntos recomendado:')
627.                         hoja.write(renglon6,7,n
  um_ptos)
628.                     elif len(aa) < num_ptos:
629.                          hoja.write(renglon4,6,
  'Se recomienda medir más puntos en este
  elemento')
630.                             #text_ptos = 'Se
  recomienda medir al menos',num_ptos,' # de
  puntos'
631.                          hoja.write(renglon5,6,
  'Num de puntos recomendado:')
632.                          hoja.write(renglon6,7,
  num_ptos)
633.
634.                     titulo = titulo + 7
```

```
635.                     renglon2 = renglon2 + 7
636.                     renglon3 = renglon3 + 7
637.                     renglon4 = renglon4 + 7
638.                     renglon5 = renglon5 + 7
639.                     renglon6 = renglon6 + 7
640.                     renglon7 = renglon7 + 7
641.                     renglon8 = renglon8 + 7
642.                     renglon9 = renglon9 + 7
643.                     renglon10 = renglon10 + 7
644.                     renglon11 = renglon11 + 7
645.
646.
647.                 elif material == 'Concreto': #C
  ONCRETO
648.                     els1_mx = round(DistAB/240,
  3) + umbral
649.                     els2_mx = round(DistAB/480,
  3) + umbral
650.                     els3_mx = round(DistAB/480,
  3) + umbral
651.                     els4_mx = round(DistAB/960,
  3) + umbral
652.                     els1_aci = round(DistAB/180
  ,3) + umbral
653.                     els2_aci = round(DistAB/360
  ,3) + umbral
654.                     els3_aci = round(DistAB/480
  ,3) + umbral
655.                     els4_aci = round(DistAB/240
  ,3) + umbral
656.
657.                     celda_titulo ='A'+str(titul
  o)+':F'+str(titulo)
658.                     texto_titulo ='LINEA
  '+str(linea)
659.                     hoja.merge_range(celda_titu
  lo,texto_titulo,titulo00)
660.                     celda_distancia ='A'+str(re
  nglon2)
661.                     hoja.write(celda_distancia,
  'Distancia (m)',t2)
```

```
                    celda_distancia2 = 'B'+str(renglon2)
                    hoja.write(celda_distancia2,DistAB,t1)
                    celda_max = 'C'+str(renglon2)
                    hoja.write(celda_max,'Deflexión Máx (m)',t2)
                    celda_max2 = 'D'+str(renglon2)
                    hoja.write(celda_max2,flecharedon,t1)
                    celda_espacio1 = 'A'+str(renglon3)+':F'+str(renglon3)
                    hoja.merge_range(celda_espacio1,'')
                    celda_ntc = 'A'+str(renglon4)+':A'+str(renglon7)
                    hoja.merge_range(celda_ntc,'NTC CDMX',t5)
                    celda_aci = 'A'+str(renglon8)+':A'+str(renglon11)
                    hoja.merge_range(celda_aci,'ACI 318S-140',t5)
                    ntc1 = 'B'+str(renglon4)+':B'+str(renglon5)
                    hoja.merge_range(ntc1,'Trabes / Vigas',t5)
                    ntc2 = 'B'+str(renglon6)+':B'+str(renglon7)
                    hoja.merge_range(ntc2,'Trabes / Vigas que afecten a elementos NO estructurales',t5)
                    aci1 = 'B'+str(renglon8)+':B'+str(renglon9)
                    hoja.merge_range(aci1,'Trabes / Vigas',t5)
                    aci2 = 'B'+str(renglon10)+':B'+str(renglon11)
```

```
681.                     hoja.merge_range(aci2,'Trab
  es / Vigas que afecten a elementos NO
  estructurales',t5)
682.                     c4 = 'C'+str(renglon4)
683.                     hoja.write(c4,'No
  voladizas',t5)
684.                     c5 = 'C'+str(renglon5)
685.                     hoja.write(c5,'Voladiza',t5
  )
686.                     c6 = 'C'+str(renglon6)
687.                     hoja.write(c6,'No
  voladiza',t5)
688.                     c7 = 'C'+str(renglon7)
689.                     hoja.write(c7,'Voladiza',t5
  )
690.                     c8 = 'C'+str(renglon8)
691.                     hoja.write(c8,'Debido a
  cargas de nieve y lluvia',t5)
692.                     c9 = 'C'+str(renglon9)
693.                     hoja.write(c9,'Debido a
  carga viva',t5)
694.                     c10 = 'C'+str(renglon10)
695.                     hoja.write(c10,'Susceptible
  s de sufrir daños por deflexiones grandes',t5)
696.                     c11 = 'C'+str(renglon11)
697.                     hoja.write(c11,'No
  susceptibles de sufrir daños debido a
  deflexiones grandes',t5)
698.
699.                     d4_pos = 'D'+str(renglon4)
700.                     d4_nombre = 'ELS =
  '+str(els1_mx)
701.                     hoja.write(d4_pos,d4_nombre
  ,t5)
702.                     d5_pos = 'D'+str(renglon5)
703.                     d5_nombre = 'ELS =
  '+str(els2_mx)
704.                     hoja.write(d5_pos,d5_nombre
  ,t5)
705.                     d6_pos = 'D'+str(renglon6)
```

```
706.                        d6_nombre = 'ELS =
  '+str(els3_mx)
707.                        hoja.write(d6_pos,d6_nombre
  ,t5)
708.                        d7_pos = 'D'+str(renglon7)
709.                        d7_nombre = 'ELS =
  '+str(els4_mx)
710.                        hoja.write(d7_pos,d7_nombre
  ,t5)
711.                        d8_pos = 'D'+str(renglon8)
712.                        d8_nombre = 'ELS =
  '+str(els1_aci)
713.                        hoja.write(d8_pos,d8_nombre
  ,t5)
714.                        d9_pos = 'D'+str(renglon9)
715.                        d9_nombre = 'ELS =
  '+str(els2_aci)
716.                        hoja.write(d9_pos,d9_nombre
  ,t5)
717.                        d10_pos = 'D'+str(renglon10
  )
718.                        d10_nombre = 'ELS =
  '+str(els3_aci)
719.                        hoja.write(d10_pos,d10_nomb
  re,t5)
720.                        d11_pos = 'D'+str(renglon11
  )
721.                        d11_nombre = 'ELS =
  '+str(els4_aci)
722.                        hoja.write(d11_pos,d11_nomb
  re,t5)
723.
724.                        if flecharedon <= els1_mx:
725.                            celda_cumple4 = 'E'+str
  (renglon4)
726.                            hoja.write(celda_cumple
  4,'SÍ CUMPLE',t5)
727.                        elif flecharedon > els1_mx:
728.                            celda_cumple4 = 'E'+str
  (renglon4)
```

```
729.                         hoja.write(celda_cumple
  4,'NO CUMPLE',t5)
730.                     if flecharedon <= els2_mx:
731.                         celda_cumple5 = 'E'+str
  (renglon5)
732.                         hoja.write(celda_cumple
  5,'NO APLICA',t5)
733.                     elif flecharedon > els2_mx:
734.                         celda_cumple5 = 'E'+str
  (renglon5)
735.                         hoja.write(celda_cumple
  5,'NO APLICA',t5)
736.                     if flecharedon <= els3_mx:
737.                         celda_cumple6 = 'E'+str
  (renglon6)
738.                         hoja.write(celda_cumple
  6,'SÍ CUMPLE',t5)
739.                     elif flecharedon > els3_mx:
740.                         celda_cumple6 = 'E'+str
  (renglon6)
741.                         hoja.write(celda_cumple
  6,'NO CUMPLE',t5)
742.                     if flecharedon <= els4_mx:
743.                         celda_cumple7 = 'E'+str
  (renglon7)
744.                         hoja.write(celda_cumple
  7,'NO APLICA',t5)
745.                     elif flecharedon > els4_mx:
746.                         celda_cumple7 = 'E'+str
  (renglon7)
747.                         hoja.write(celda_cumple
  7,'NO APLICA',t5)
748.                     if flecharedon <= els1_aci:
749.                         celda_cumple8 = 'E'+str
  (renglon8)
750.                         hoja.write(celda_cumple
  8,'SÍ CUMPLE',t5)
751.                     elif flecharedon > els1_aci
  :
752.                         celda_cumple8 = 'E'+str
  (renglon8)
```

```
753.                         hoja.write(celda_cumple
  8,'NO CUMPLE',t5)
754.                     if flecharedon <= els2_aci:
755.                         celda_cumple9 = 'E'+st
  r(renglon9)
756.                         hoja.write(celda_cumpl
  e9,'SÍ CUMPLE',t5)
757.                     elif flecharedon > els2_aci
  :
758.                         celda_cumple9 = 'E'+st
  r(renglon9)
759.                         hoja.write(celda_cumpl
  e9,'NO CUMPLE',t5)
760.                     if flecharedon <= els3_aci:
761.                         celda_cumple10 = 'E'+s
  tr(renglon10)
762.                         hoja.write(celda_cumpl
  e10,'SÍ CUMPLE',t5)
763.                     elif flecharedon > els3_aci
  :
764.                         celda_cumple10 = 'E'+s
  tr(renglon10)
765.                         hoja.write(celda_cumpl
  e10,'NO CUMPLE',t5)
766.                     if flecharedon <= els4_aci:
767.                         celda_cumple11 = 'E'+s
  tr(renglon11)
768.                         hoja.write(celda_cumpl
  e11,'SÍ CUMPLE',t5)
769.                     elif flecharedon > els4_aci
  :
770.                         celda_cumple11 = 'E'+s
  tr(renglon11)
771.                         hoja.write(celda_cumpl
  e11,'NO CUMPLE',t5)
772.
773.
774.                     for q in range(0,len(aa),1)
  :
775.                         hoja.write(renglon2,6,'
  Punto',t5)
```

```
776.                         hoja.write(renglon3,6,'
  Flecha (m)',t5)
777.                         hoja.write(renglon2,7+q
  ,q,t5)
778.                         hoja.write(renglon3,7+q
  ,aa[q][0],t5)
779.                     if len(aa) >= num_ptos:
780.                         hoja.write(renglon4,6,'
  El núm. de puntos por elemento es el
  recomendado')
781.                         hoja.write(renglon5,6,'
  Num de puntos recomendado:')
782.                         hoja.write(renglon6,7,n
  um_ptos)
783.                     elif len(aa) < num_ptos:
784.                          hoja.write(renglon4,6,
  'Se recomienda medir más puntos en este
  elemento')
785.                             #text_ptos = 'Se
  recomienda medir al menos',num_ptos,' # de
  puntos'
786.                          hoja.write(renglon5,6,
  'Num de puntos recomendado:')
787.                          hoja.write(renglon6,7,
  num_ptos)
788.
789.
790.
791.                     titulo = titulo + 11
792.                     renglon2 = renglon2 + 11
793.                     renglon3 = renglon3 + 11
794.                     renglon4 = renglon4 + 11
795.                     renglon5 = renglon5 + 11
796.                     renglon6 = renglon6 + 11
797.                     renglon7 = renglon7 + 11
798.                     renglon8 = renglon8 + 11
799.                     renglon9 = renglon9 + 11
800.                     renglon10 = renglon10 + 11
801.                     renglon11 = renglon11 + 11
802.
```

```
803.                 elif material == 'Madera': #MAD
  ERA
804.
805.
806.                     els1_mx = round(DistAB/240,
  3) + umbral
807.                     els2_mx = round(DistAB/480,
  3) + umbral
808.                     els3_mx = round(DistAB/240
  + 0.005,3) + umbral
809.                     els4_mx = round(DistAB/480
  + 0.003,3) + umbral
810.
811.                     celda_titulo ='A'+str(titul
  o)+':F'+str(titulo)
812.                     texto_titulo ='LINEA
  '+str(linea)
813.                     hoja.merge_range(celda_titu
  lo,texto_titulo,titulo00)
814.                     celda_distancia ='A'+str(re
  nglon2)
815.                     hoja.write(celda_distancia,
  'Distancia (m)',t2)
816.                     celda_distancia2 = 'B'+str(
  renglon2)
817.                     hoja.write(celda_distancia2
  ,DistAB,t1)
818.                     celda_max = 'C'+str(renglon
  2)
819.                     hoja.write(celda_max,'Defle
  xión Máx (m)',t2)
820.                     celda_max2 = 'D'+str(renglo
  n2)
821.                     hoja.write(celda_max2,flech
  aredon,t1)
822.                     celda_espacio1 = 'A'+str(re
  nglon3)+':F'+str(renglon3)
823.                     hoja.merge_range(celda_espa
  cio1,'')
824.
```

```
                    celda_ntc = 'A'+str(renglon
  4)+':A'+str(renglon7)
                    hoja.merge_range(celda_ntc,
  'NTC CDMX',t5)
                    ntc1 = 'B'+str(renglon4)+':
  B'+str(renglon5)
                    hoja.merge_range(ntc1,'Trab
  es / Vigas',t5)
                    ntc2 = 'B'+str(renglon6)+':
  B'+str(renglon7)
                    hoja.merge_range(ntc2,'Trab
  es / Vigas que afecten a elementos NO
  estructurales',t5)

                    c4 = 'C'+str(renglon4)
                    hoja.write(c4,'< 3.5 m',t5)
                    c5 = 'C'+str(renglon5)
                    hoja.write(c5,'> 3.5 m',t5)
                    c6 = 'C'+str(renglon6)
                    hoja.write(c6,'< 3.5 m',t5)
                    c7 = 'C'+str(renglon7)
                    hoja.write(c7,'> 3.5 m',t5)

                    d4_pos = 'D'+str(renglon4)
                    d4_nombre = 'ELS =
  '+str(els1_mx)
                    hoja.write(d4_pos,d4_nombre
  ,t5)
                    d5_pos = 'D'+str(renglon5)
                    d5_nombre = 'ELS =
  '+str(els2_mx)
                    hoja.write(d5_pos,d5_nombre
  ,t5)
                    d6_pos = 'D'+str(renglon6)
                    d6_nombre = 'ELS =
  '+str(els3_mx)
                    hoja.write(d6_pos,d6_nombre
  ,t5)
                    d7_pos = 'D'+str(renglon7)
                    d7_nombre = 'ELS =
  '+str(els4_mx)
```

```
852.                     hoja.write(d7_pos,d7_nombre
  ,t5)
853.
854.                     if flecharedon <= els1_mx:
855.                         celda_cumple4 = 'E'+str
  (renglon4)
856.                         hoja.write(celda_cumple
  4,'SÍ CUMPLE',t5)
857.                     elif flecharedon > els1_mx:
858.                         celda_cumple4 = 'E'+str
  (renglon4)
859.                         hoja.write(celda_cumple
  4,'NO CUMPLE',t5)
860.                     if flecharedon <= els2_mx:
861.                         celda_cumple5 = 'E'+str
  (renglon5)
862.                         hoja.write(celda_cumple
  5,'SÍ CUMPLE',t5)
863.                     elif flecharedon > els2_mx:
864.                         celda_cumple5 = 'E'+str
  (renglon5)
865.                         hoja.write(celda_cumple
  5,'NO CUMPLE',t5)
866.                     if flecharedon <= els3_mx:
867.                         celda_cumple6 = 'E'+str
  (renglon6)
868.                         hoja.write(celda_cumple
  6,'SÍ CUMPLE',t5)
869.                     elif flecharedon > els3_mx:
870.                         celda_cumple6 = 'E'+str
  (renglon6)
871.                         hoja.write(celda_cumple
  6,'NO CUMPLE',t5)
872.                     if flecharedon <= els4_mx:
873.                         celda_cumple7 = 'E'+str
  (renglon7)
874.                         hoja.write(celda_cumple
  7,'SÍ CUMPLE',t5)
875.                     elif flecharedon > els4_mx:
876.                         celda_cumple7 = 'E'+str
  (renglon7)
```

```
877.                         hoja.write(celda_cumple
  7,'NO CUMPLE',t5)
878.
879.                     for q in range(0,len(aa),1)
  :
880.                         hoja.write(renglon2,6,'
  Punto',t5)
881.                         hoja.write(renglon3,6,'
  Flecha (m)',t5)
882.                         hoja.write(renglon2,7+q
  ,q,t5)
883.                         hoja.write(renglon3,7+q
  ,aa[q][0],t5)
884.                     if len(aa) >= num_ptos:
885.                         hoja.write(renglon4,6,'
  El núm. de puntos por elemento es el
  recomendado')
886.                         hoja.write(renglon5,6,'
  Num de puntos recomendado:')
887.                         hoja.write(renglon6,7,n
  um_ptos)
888.                     elif len(aa) < num_ptos:
889.                          hoja.write(renglon4,6,
  'Se recomienda medir más puntos en este
  elemento')
890.                             #text_ptos = 'Se
  recomienda medir al menos',num_ptos,' # de
  puntos'
891.                          hoja.write(renglon5,6,
  'Num de puntos recomendado:')
892.                          hoja.write(renglon6,7,
  num_ptos)
893.
894.                     titulo = titulo + 7
895.                     renglon2 = renglon2 + 7
896.                     renglon3 = renglon3 + 7
897.                     renglon4 = renglon4 + 7
898.                     renglon5 = renglon5 + 7
899.                     renglon6 = renglon6 + 7
900.                     renglon7 = renglon7 + 7
901.                     renglon8 = renglon8 + 7
```

```
                    renglon9 = renglon9 + 7
                    renglon10 = renglon10 + 7
                    renglon11 = renglon11 + 7
    #---------------------------------
  ESCRIBIMOS EN ARCHIVOS TXT Y SCR
                line_cad ="LINE",str((df_Icoord
  .iloc[i, 1])).replace("
  ","")+","+str(df_Icoord.iloc[i, 0]).replace("
  ","")+","+str(df_Icoord.iloc[i, 2]).replace("
  ","")+" "+str(df_Icoord.iloc[i+1, 1]).replace("
  ","")+","+str(df_Icoord.iloc[i+1, 0]).replace("
  ","")+","+str(df_Icoord.iloc[i+1, 2]).replace("
  ","")+" "
                print(*line_cad,file=(cad_txt))
                print(*line_cad,file=(cad_scr))
                point_cad ="POINT",str(C[0]).re
  place(" ","")+","+str(C[1]).replace("
  ","")+","+str(C[2]).replace(" ","")
                print(*point_cad,file=(cad_txt)
  )
                print(*point_cad,file=(cad_scr)
  )
                textline_cad ="TEXT",str(df_Ico
  ord.iloc[i, 1]).replace("
  ","")+","+str(df_Icoord.iloc[i, 0]).replace("
  ","")+","+str(df_Icoord.iloc[i, 2]).replace("
  ","")+" 0.02 0 Línea "+str(linea).replace("
  ","")+"  "
                print(*textline_cad,file=(cad_t
  xt))
                print(*textline_cad,file=(cad_s
  cr))
                pointext_cad ="TEXT",str(C[0]).
  replace(" ","")+","+str(C[1]).replace("
  ","")+","+str(C[2]).replace(" ","")+" 0.02 0
  Flecha_Max "+str(flecharedon).replace("
  ","")+"  "
                print(*pointext_cad,file=(cad_t
  xt))
                print(*pointext_cad,file=(cad_s
  cr))
```

```
918.
919.
920.           ventana.destroy()
921.           cad_txt.close()
922.           cad_scr.close()
923.           excel.close()
924.           sys.exit()
925.
926.
927.
928.
929.   Button(text='Continuar',bg='gold',command=ca
  lc).place(x=275,y=590)
930.   ventana.mainloop()
```

ii) Anexo página del programa Arrow

Se proporciona la liga de acceso a la página de ARROW App para la descarga del programa:

https://site-9000923-386-2002.mystrikingly.com/

iii) Bibliografía

Amazon. (2023). *AWS*. Obtenido de ¿Qué es Python?: https://aws.amazon.com/es/what-is/python/

American Concrete Institute. (2014). *Requisitos de Reglamento para Concreto Estructural (ACI 318S-14).* Recuperado el 5 de Mayo de 2022

Armenta Mena, C. (1986). *Deflexiones en Elementos de Concreto Preforzado.* Recuperado el 29 de Abril de 2022, de http://132.248.9.195/pmig2018/0013723/0013723.pdf

BIMnD, E. (2022). *Las 7 dimensiones BIM*. Recuperado el 30 de 08 de 2022, de https://www.bimnd.es/7dimensionesbim/

Camba Castañeda, J. L., Chacón García, F., & Pérez Arellano, F. (1982). *Repositorio Digital de la Facultad de Ingeniería, UNAM.* Recuperado el 27 de Abril de 2022, de http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/814/An%c3%a1lisis%20Estructural_CAMBA_ocr.pdf?sequence=1&isAllowed=y

Carlos, U. N., & José, P. C. (2009). *Universidad Carlos III de Madrid.* Recuperado el 27 de Abril de 2022, de Open Course Ware: https://ocw.uc3m.es/mecanica-de-medios-continuos-y-teoria-de-estructuras/ingenieria-estructural/material-de-clase-1/apuntes/Capitulo_1_I_.-Introduccion_a_las_estructuras.pdf

Carrillo Cubillas, M. M. (14 de Mayo de 2004). *Bibliotecas UDLAP.* Obtenido de http://catarina.udlap.mx/u_dl_a/tales/documentos/lic/carrillo_c_mm/capitulo5.pdf

Cervera Ruiz, M., & Blanco Díaz, E. (2014). *Mecánica de Estructuras.* Barcelona, España: Centro Internacional de Métodos Numéricos en Ingeniería (CIMNE). Obtenido de http://cervera.rmee.upc.edu/libros/Mec%C3%A1nica%20de%20Estructuras.pdf

Cirilo Manzanares, C. (2013). *Revisión de Estados Límite en Estructuras.* Recuperado el 28 de Abril de 2022, de http://132.248.9.195/ptd2013/abril/0691611/0691611.pdf

Clark, C., & Gazoni, E. (24 de 05 de 2022). *Openpyxl*. Obtenido de openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files: https://openpyxl-readthedocs-

io.translate.goog/en/stable/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc

Código Técnico de la Edificación de España. (2019). *Documento Básico SE Seguridad Estructural.* España. Recuperado el 3 de Mayo de 2022, de https://www.codigotecnico.org/pdf/Documentos/SE/DBSE.pdf

Colaborades. (24 de Abril de 2022). *Ley de elasticidad de Hooke*. Recuperado el 27 de Abril de 2022, de Wikipedia: https://es.wikipedia.org/w/index.php?title=Ley_de_elasticidad_de_Hooke&oldid=143112157

Durán Mendietam, J., Martínez, T., & Nepomuceno Gutiérrez, O. (s.f.). *Centro de Estudios Geográficos.* Recuperado el 06 de Julio de 2022, de Centro de Estudios Geográficos: http://itgeo.com.mx/ceg/wp-content/uploads/2015/05/Peritajes-en-materia-de-topograf%C3%ADa.pdf

EspacioBIM. (16 de 01 de 2020). *BIM o Metodología BIM (Qué es) más que tecnologia*. Recuperado el 30 de 08 de 2022, de https://www.espaciobim.com/bim

FI División de Ciencias Básicas, Coordinación de Matemáticas. (Abril de 2011). *División de Ciencias Básicas.* Recuperado el 29 de Junio de 2022, de http://dcb.fi-c.unam.mx/cerafin/bancorec/capsulasmatematicas/larecta.pdf

García Martínez, A. (2010). *Análisis del Ciclo de Vida (ACV) de Edificios.* Sevilla: Universidad de Sevilla.

García, M. (2014). *Topografía.* Universidad Politécnica de Cartagena.

Imasgal. (2022). *Nivel de Desarrollo (LOD) BIM*. Recuperado el 30 de 08 de 2022, de https://imasgal.com/nivel-desarrollo-bim-lod/

Ing geek. (12 de 10 de 2012). *¿Sabes qué es un LOD en BIM?* Recuperado el 30 de 08 de 2022, de https://www.ingegeek.site/2021/10/12/sabes-que-es-un-lod-en-bim/

L. Fernández, J. (2023). *FisicaLab*. Obtenido de FísicaLab: https://www.fisicalab.com/apartado/momento-fuerza

Lima, A. (2022). *Acervo Lima*. Obtenido de Módulo Python Sys: https://es.acervolima.com/modulo-python-sys/

Marco, G., & Mora, W. (2018). *Universidad de los Andes.* Recuperado el 30 de Junio de 2022, de http://funes.uniandes.edu.co/12975/1/Gutierrez2018Vectores.pdf

Meli Piralla, R. (2009). *Diseño Estructural* (Segunda ed.). Limusa. Obtenido de https://www.aldeatdo.com/wp-content/uploads/2020/12/Diseno-Estructural-Meli-Piralla-ARQUILIBROS-AL-3.pdf

Montes de Oca, M. (1989). *Topografía.* D.F. : Alfaomega, S.A. de C.V.

Numpy. (2022). *Numpy*. Obtenido de https://numpy.org/

Python. (2022). *La Biblioteca Estándar de Python*. Obtenido de https://docs.python.org/es/3.9/library/

RAE. (2022). *Diccionario de la lengua española*, 23.5 en línea. (23, Editor) Recuperado el 05 de Julio de 2022, de https://dle.rae.es/catenario#7wZ57Sr

Sociedad Mexicana de Ingeniería Estructural, A.C. (SMIE). (2017). Recuperado el 27 de Abril de 2022, de https://www.smie.org.mx/archivos/informacion-tecnica/normas-tecnicas-complementarias/normas-tecnicas-complementarias-diseno-sismo-2017.pdf

SPHINX. (11 de 07 de 2022). *Python 3.9.13 documentation*. Obtenido de csv — CSV File Reading and Writing: https://docs.python.org/3/library/csv.html

SPHINX. (11 de 07 de 2022). *Python 3.9.13 documentation*. Obtenido de math - Funciones matemáticas : https://docs-python-org.translate.goog/3/library/math.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc

SPHINX. (11 de 07 de 2022). *Python 3.9.13 documentation*. Obtenido de https://docs.python.org/es/3.10/library/os.html

SPHINX. (11 de 07 de 2022). *Python 3.9.13 documentation*. Obtenido de https://docs.python.org/es/3/library/tk.html

SPHINX. (2022). *XlsxWriter*. Obtenido de Creating Excel files with Python and XlsxWriter: https://xlsxwriter.readthedocs.io/

Wikipedia. (26 de 01 de 2022). *Pandas (software)*. Obtenido de https://es.wikipedia.org/wiki/Pandas_(software)

iv) Índice de imágenes

v) Índice de tablas